Visual Basic 6
למתכנתים



אילן אמויאל

2001

Visual Basic 6 למתכנתים

(מהדורה דיגיטלית)

מאת אילן אמויאל

זכויות יוצרים

2001 © כל הזכויות שמורות לאילן אמויאל

הספר מוגש לציבור ללא תשלום, לשימוש אישי ולמטרות לימוד.

אין בהפצתו ללא תשלום משום ויתור על זכויות היוצרים. אין להעתיק, לשכפל, למכור או לעשות שימוש מסחרי בספר, בשלמותו או בחלקו, ללא אישור מראש ובכתב מאת בעל זכויות היוצרים.

הגבלת אחריות

המידע, ההסברים ודוגמאות הקוד בספר נועדו למטרות לימוד והעשרה בלבד.

המחבר וההוצאה עשו כמיטב יכולתם להבטיח את נכונות המידע המובא בספר, אולם אין הם מתחייבים לשלמותו, לדיוקו או להתאמתו למטרה מסוימת.

דוגמאות הקוד נכתבו ונבדקו בסביבת Visual Basic 6 על מערכת ההפעלה Windows.

השימוש במידע ובדוגמאות הקוד נעשה באחריות הקורא בלבד. המחבר וההוצאה לא יישאו באחריות לנזק ישיר או עקיף העלול להיגרם כתוצאה מהשימוש בהם.

הוצאת מויאל

אתר ההוצאה: https://moyal.es

אתר "כיתה ברשת": https://kita.co.il

תוכן העניינים

הקדמה 13

אודות שפת Visual Basic 13
תוכן הספר 13
ידע מוקדם 14
קובצי התוכניות 14
אי נטילת אחריות 14
הערות מאוחרות להקדמה 15

הקדמת ביניים משנת 2010 15
תוספת משנת 2026 15

שיעור 1: סקירה 17

אודות Visual Basic 17

יתרונות הפיתוח ב-Visual Basic 17
חסרונות הפיתוח ב-Visual Basic 17

הכרת סביבת הפיתוח המשולבת (Integrated Development Environment) 18

תיבת הדו-שיח New Project 18
שורת התפריטים 19
סרגל הכלים הסטנדרטי של Visual Basic 20
חלון ה-Project Explorer 21
חלון ה-Properties 21
חלון ה-Form Layout 22
ה-Toolbox (ארגז הכלים) 22
ה-Code Editor (עורך הקוד) 23

שיעור 2: טפסים ותפריטים 25

פרויקט 25

יצירת פרויקט חדש 26
הרצת פרויקט 26
שמירת פרויקט 26

ה-Form הסטנדרטי 27
מרכיבי Form 27
Properties (מאפיינים), Methods (שיטות), ו-Events (אירועים) 28
Properties של Form 28
Methods של Form 30
Events של Form 30
פונקציות ראשונות 33
הפונקציה Print 33
הפונקציה MsgBox 33
הוספת הערות לקוד 36
MDI Form (טופס MDI) 37
Properties של MDI Form 37
Methods של MDI Form 37
Menus (תפריטים) 38
Properties של Menu 38
Events של Menu 38
עבודה עם צבעים 38
הפונקציה QBColor 38
הפונקציה RGB 39
אוסף הקבועים ColorConstants 39
אוסף הקבועים SystemColorConstants 39
תרגילים 47
שיעור 3: הפקדים הסטנדרטיים I 49
Label (תווית) 49
Properties של Label 49
Events של Label 50
TextBox (תיבת טקסט) 50
Properties של TextBox 50

Events של TextBox 51
CommandButton (לחצן פקודה) 51
Properties של CommandButton 51
HScrollBar (פס גלילה אופקי) ו-VScrollBar (פס גלילה אנכי) 52
Properties של פסי גלילה 52
Events של פסי גלילה 52
פונקציות לעבודה עם מחרוזות 53
הפונקציה Len 53
הפונקציה Val 53
הפונקציה Str 53
הפונקציה UCase 53
הפונקציה LCase 54
הפונקציות Trim, LTrim ו-Rtrim 54
הפונקציות Mid, Left ו-Right 54
האופרטור & 55
הקבוע vbCrLf 55
הפונקציה Asc 55
התנאי If...Then...End If 56
תרגילים 57

שיעור 4: מרכיבי השפה I 59

Variables (משתנים) ו-Constants (קבועים) 59
סוגי משתנים 59
הצהרה על משתנים 62
הצהרה על קבועים 63
Control Structures (מבני בקרה)..... 64
המבנה If...Then...Else 64
המבנה While...Wend 65
המבנה Do...Loop 66
המבנה For...Next 67

המבנה Select Case 68
הפונקציות Beep, InputBox ו-IIf 69
הרוטינה Beep 69
הפונקציה InputBox 69
הפונקציה IIf 70
תרגילים 72
שיעור 5: הפקדים הסטנדרטיים II 73
CheckBox (תיבת סימון) 73
Properties של CheckBox 74
Events של CheckBox 74
OptionButton (לחצן אפשרות) 74
Properties של OptionButton 74
Frame (מסגרת) 74
Properties של Frame 74
Methods של Frame 75
ListBox (תיבת רשימה) 78
Properties של ListBox 78
Methods של ListBox 79
Events של ListBox 79
ComboBox (תיבה משולבת) 80
Properties של ComboBox 80
Line (קו) ו-Shape (צורה) 80
Properties של Line ו-Shape 80
Timer (פקד שעון) 81
Properties של Timer 81
Events של Timer 81
תרגילים 89

שיעור 6: מרכיבי השפה II 91

המילה With ***92***
מערכים (Arrays) ***92***
מערכים חד ממדיים (One-Dimensional Arrays) 92
מערכים רב ממדיים (Multi-Dimensional Arrays) 93
מערכים דינמיים (Dynamic Arrays) 93
בדיקת גודל מערך בזמן ריצה באמצעות LBound ו-UBound 94
מערכי פקדים (Control Array) ***96***
גישה לפקד בודד מתוך Control Array 96
בדיקת גודל המערך וגבולותיו 97
אירועים של פקדים השייכים ל-Control Array 97
שלבי יצירת Control Array 97
יצירת פקדים בזמן ריצה 97
פרוצדורות ב-Visual Basic ***101***
רוטינות ופונקציות שלב I 101
קריאות לרוטינות ופרוצדורות 106
רוטינות ופרוצדורות שלב II 107
מודולי קוד סטנדרטיים *.bas ***110***
הפעלת פונקציות מחלון ה-Immediate ***110***
Enumerations (אוספי קבועים) ***113***
User-Defined Types (סוגי נתונים מוגדרים על ידי המשתמש) ***114***

שיעור 7: הרצה וניפוי שגיאות 117

חלונות תצוגה שונים לאבחון ***117***
שלושת המצבים של תוכנית ***117***
התפריט Run (הרצה) ***118***
Start (התחל) 118
Start With Full Compile (התחל לאחר קימפול מלא) 118
Break (עבור ל-Break Mode) 118
End (סיים) 119

Restart (התחל מחדש) 119
התפריט Debug (ניפוי שגיאות) 119
Step Into (צעד לתוך) 119
Step Over (צעד מעבר) 120
Step Out (צא החוצה) 120
Run To Cursor (הרץ עד הסמן) 120
Add Watch (הוספת תצפית) 120
Edit Watch (ערוך תצפית) 122
Quick Watch (תצפית מהירה) 122
Toggle Breakpoint (הפוך מצב של נקודת עצירה) 123
Clear All Breakpoints (הסר את כל נקודות העצירה) 123
Set Next Statement (קבע את המשפט הבא לביצוע) 123
Show Next Statement (הצג את המשפט הבא לביצוע) 123
האובייקט Debug 124
טיפול בשגיאות בקוד התוכנית..... 125
האובייקט Err 125
On Error... 126
המשפט Resume 127
הפונקציה Error 128
שיעור 8: תכנות מבוסס רכיבים - COMPONENTWARE DEVELOPMENT 129
תכנות פרוצדורלי לעומת תכנות מונחה עצמים 129
Procedural Programming 129
Object Oriented Programming 133
Event Oriented Programming (תכנות מונחה אירועים) 138
Component Object Model (COM) 138
התמיכה של Visual Basic ב-OOP ו-COM 141
רכיבי ActiveX של Visual Basic 141
שיעור 9: תכנות מונחה עצמים ב-Visual Basic 143
Classes (מחלקות) 143

Properties של Class Module 143
Events של Class Module 144
הגדרת משתנים ב-Class 144
הוספת פרוצדורות ל-Class 144
הצהרה על אובייקטים מסוג המחלקה 146
Reference Counting (ספירת התייחסויות) 148
מימוש Interface 164
אבחון תכונות אובייקט בזמן ריצה 165
Early Binding לעומת Late Binding 171

הגדרת תכונות של פרוצדורות ב-Interface 176

צפייה בתכונות באמצעות Object Browser 176
צפייה באמצעות היישום OLE VIEW 178
הגדרת תכונות 179

Collections (אוספים) 181

תחביר 182
הלולאה For each ..In .. Next 183
מחלקות Collection מותאמות אישית 189

שיעור 10: ActiveX DLL 201

מושגים 201
יצירת ActiveX DLL 204

יצירת אובייקטים 204
הצגת ממשק משתמש 204
ניהול קבוצת פרויקטים במקביל 205

Instantiation (יצירת מופעים)..... 212
הגדרת כתובות בסיס לרכיבי DLL 214
טיפול בשגיאות ברכיב..... 216

שיעור 11: כתיבת Stored Procedures ב-SQL Server 223

Stored Procedures 223

תהליך הביצוע של משפט SQL הנשלח על ידי לקוח 223

תהליך הביצוע של בקשה להפעלת Stored Procedure מתוך תוכנית לקוח 225
כתיבת Stored Procedures ב-SQL Server 225
פרוצדורה פשוטה ללא פרמטרים 225
פרוצדורה המקבלת פרמטרים 226
הגדרת ערך ברירת מחדל לפרמטרים 227
החזרת ערך מפרוצדורה באמצעות משתנה 228
שימוש במשתנים מקומיים 229
יציאה מפרוצדורה באמצעות RETURN 230
שיעור 12: ACTIVEX DATA OBJECTS 233
סקירה של טכנולוגיות הגישה לנתונים 233
ODBC (Open Database Connectivity) 234
DAO (Data Access Object) 235
DC (Data Control) 236
RDO (Remote Data Object) 236
RDC (Remote Data Control) 236
ODBCDirect 237
OLE DB 237
ADO (ActiveX Data Objects) 238
MSDASQL 238
ADODC (ADO Data Control) 238
ADO (ActiveX Data Objects) 248
Data Link 249
האובייקט Connection 262
Event-ים של אובייקט Connection 268
האובייקט Recordset..... 274
המאפיין CursorType 279
תנועה ב-Recordset 280
Cursors (סמנים)..... 280
המאפיין CursorLocation 280
המאפיין LockType 281

מאפיינים נוספים 281
האובייקט Command 281
ADO Data control..... 282
האובייקט Recordset 282
Properties של ADODC 283
Methods של ADODC 284
Events של ADODC 284
עיצוב הנתונים..... 284
הערות לגבי Data Bound Controls 284
שיעור 13: COM+ 287
אודות COM+ וניהול טרנזקציות..... 287
יצירת רכיבים המותאמים לפעולה תחת COM+..... 291
המאפיין MTSTransactionMode 291
האובייקט ObjectContext 291
יכולות נוספות של COM+ 297

הקדמה

אודות שפת Visual Basic

Visual Basic 6.0 היא שפת תכנות וסביבת פיתוח של Microsoft, המיועדת לפיתוח יישומים למערכת ההפעלה Windows. היא מאפשרת לפתח במהירות יישומים בעלי ממשק גרפי, ומשמשת ליצירת תוכנות עצמאיות, כלי עזר ומערכות מידע עסקיות.

אפשר לפגוש את Visual Basic גם בתוך יישומים אחרים. באמצעות VBA אפשר לכתוב פקודות מאקרו והרחבות ליישומי Microsoft Office וליישומים נוספים, כגון AutoCAD. גם ביישומי ASP אפשר לכתוב קוד שרת באמצעות VBScript, שפה ממשפחת Visual Basic.

Visual Basic משלבת תכנות מונחה אירועים עם תמיכה בתכנות מונחה עצמים ובטכנולוגיית COM. שילוב זה מאפשר ליצור רכיבי תוכנה ולהשתמש ברכיבים קיימים.

תוכן הספר

הספר מוביל מן היסודות אל נושאים מתקדמים. בתחילתו נלמדים סביבת הפיתוח, טפסים, תפריטים, פקדים, משתנים, מבני בקרה, מערכים, פרוצדורות וניפוי שגיאות. בהמשך אני עוסק בתכנות מונחה עצמים, בממשקים ובאוספים, ביצירת רכיבי ActiveX DLL, בגישה לנתונים באמצעות ADO, בכתיבת Stored Procedures ב-SQL Server ובשימוש ב-COM+ לניהול טרנזקציות.

השתדלתי להציג את הנושאים באופן תמציתי ומסודר, באמצעות דוגמאות ממוקדות והנחיות המובילות את הקורא שלב אחר שלב. לאחר כל דוגמה מובא הסבר על פעולת הקוד ועל הרעיונות שביסודו. הספר מבוסס, בין היתר, על חומרי לימוד שפיתחתי לקורסים שהעברתי בחברות טכנולוגיה.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

ישנם תחומי משנה נוספים שאינם נידונים בספר, או נידונים בו בקצרה. עם זאת, רצף הנושאים והדוגמאות נועד להקנות בסיס חזק להבנת השפה ולשליטה בה, ולהביא את הקורא לרמה שמאפשרת לו לעבוד בה ולהמשיך ללמוד לפי צורכי העבודה.

ידע מוקדם

הספר מיועד לקוראים בעלי ידע בסיסי בתכנות וניסיון בכתיבת קוד בשפה כלשהי. נדרשת גם היכרות עם ייצוג מספרים בבסיס הבינארי, בבסיס האוקטלי ובבסיס ההקסדצימלי.

קובצי התוכניות

קובץ ה-ZIP מכיל כולל את קוד המקור של התוכניות המופיעות כדוגמאות והצעות לפתרונות התרגילים שבסוף חלק מהפרקים.

אי נטילת אחריות

המידע, ההסברים ודוגמאות הקוד בספר מובאים למטרות לימוד בלבד. עשיתי כמיטב יכולתי להציג מידע נכון, אך איני מתחייב לשלמותו, לדיוקו או להתאמתו לצורך מסוים. דוגמאות הקוד נכתבו ונבדקו בסביבת Visual Basic 6 במערכת ההפעלה Windows.

השימוש בספר, בהנחיות ובדוגמאות הקוד, בין שנעשה בידי אדם פרטי ובין שנעשה בידי חברה או גוף אחר, הוא באחריות המשתמש בלבד. האחריות לבדיקת הקוד ולהתאמתו לסביבת העבודה ולמטרת השימוש מוטלת על המשתמש. איני נוטל אחריות לנזק ישיר או עקיף שייגרם עקב שימוש בספר או הסתמכות עליו.

אני מקווה שהספר יסייע לך ללמוד את Visual Basic, להבין את עקרונותיה ולשמש לך מקור עזר בעבודה.

בברכה,

אילן אמויאל

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

הערות מאוחרות להקדמה

הקדמת ביניים משנת 2010

הספר מבוסס על חומרי לימוד וקורסים שהעברתי בחברות טכנולוגיה. לאחר שנים שבהן שימש את הלומדים בו, החלטתי לאפשר להפיץ אותו לציבור ללא תשלום. כדברי חז"ל, "אין אדם יוצא מן העולם וחצי תאוותו בידו", ואני שמח להעמיד את החומר לרשות מי שמעוניין ללמוד ממנו.

זכויות היוצרים בספר נשארות שלי. מותר להפיץ את הספר בשלמותו, ללא תשלום וללא שינוי. אין להפיץ קטעים ממנו או לעשות בו שימוש מסחרי ללא אישורי.

תוספת משנת 2026

יותר מרבע מאה לאחר כתיבת הספר, מערכות Legacy עדיין פועלות בארגונים רבים, ובהם משרדי ממשלה, בנקים וחברות ביטוח. בין המערכות הוותיקות שעדיין נדרשות לתחזוקה אפשר למצוא גם יישומים שנכתבו ב-Visual Basic 6. גם VBA ממשיכה לשמש ביישומי Office לשולחן העבודה, ובהם Excel ו-Word.

אני מפרסם כאן את הספר כדי לאפשר למי שמתחזקים מערכות Legacy ללמוד את VB6 במהירות ובעומק, מן היסודות ועד לרבדים הארגוניים (Enterprise) של השפה והטכנולוגיות שסביבה. הספר עוסק, בין היתר, בתכנות מונחה עצמים, ב-COM, ברכיבי ActiveX, בגישה למסדי נתונים וב-COM+. אף שיש תחומי משנה שאינם מכוסים בו, הוא מספק בסיס חזק להבנת השפה ולעבודה מעשית בה.

קוד המקור לדוגמאות והתרגילים שבספר זמין במאגר ה-GitHub:

https://github.com/IlanAmoyal/visual-basic-6-for-programmers

מוזמנים להתחבר
אליי גם ב-LinkedIn

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 1:

סקירה

בשיעור זה:

- תכיר את יתרונות הפיתוח שב-Visual Basic
- תכיר רכיבים בסיסיים בסביבת הפיתוח המשולבת של Visual Basic

אודות Visual Basic

- Visual Basic 6.0 היא שפת תכנות וסביבת פיתוח משולבת מבית Microsoft.
- Visual Basic כוללת מרכיבי שפה (Basic) ומרכיבים גרפיים המאפשרים פיתוח אפליקציות בעלות ממשק גרפי.

יתרונות הפיתוח ב-Visual Basic

- קלות פיתוח
- חזותיות
- מהירות פיתוח
- כלי Debug הנם חלק אינטגרלי מסביבת הפיתוח
- ניתן לנפות שגיאות ברכיבי VB6 מקומפלים גם באמצעות Visual C++, כאשר הרכיבים נבנו עם מידע ניפוי מתאים.
- ניתן להכניס תוספות (Add-Ins) לסביבת הפיתוח
- ניתן לנהל מספר פרויקטים במקביל, תחת קבוצה (Group) אחת
- קיימת תמיכה בשימוש וביצירה של רכיבי ActiveX.

חסרונות הפיתוח ב-Visual Basic

- פיתוח ב-Visual Basic פירושו פיתוח אפליקציות לסביבת Microsoft Windows בלבד

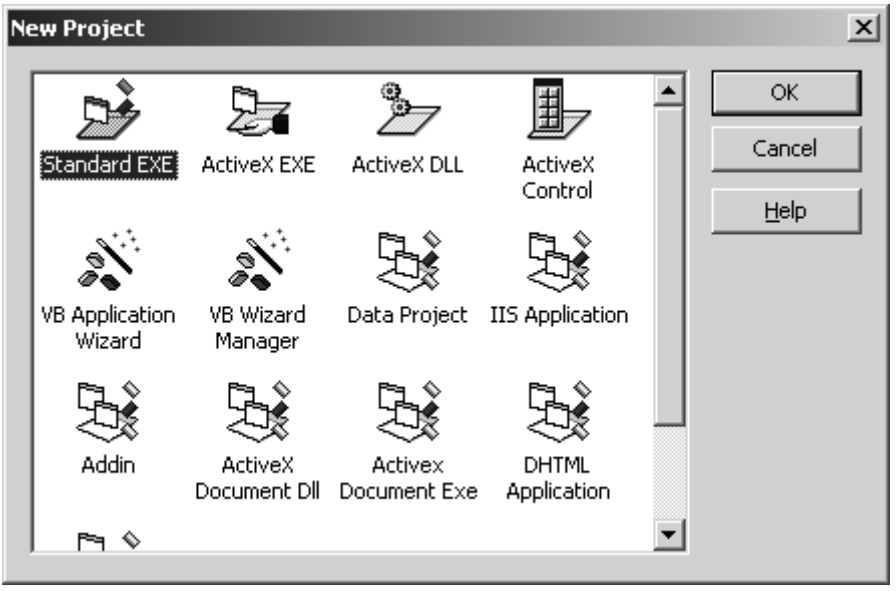
© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ביצועי תוכניות VB6 תלויים באופי היישום ובאופן הקימפול. השפה אינה מיועדת ליישומי Real Time קשיחים, המחייבים עמידה בזמני תגובה מובטחים.
- בעת הרצה מתוך ה ,IDE-קוד התוכנית פועל בתוך התהליך של סביבת הפיתוח ולא כתהליך EXE עצמאי. לכן תקלה קריטית בקוד עלולה לגרום לקריסת ה IDE-עצמו.
- פיתוח מרובה תהליכונים מוגבל ב-VB6; רכיבי ActiveX תומכים ב- Apartment Threading, ובפרויקטי ActiveX EXE אפשר גם להגדיר Thread per Object או Thread Pool.

הכרת סביבת הפיתוח המשולבת (Integrated Development Environment)

סביבת הפיתוח של Visual Basic כוללת מגוון רחב של כלים וחלונות. במהלך הספר תכיר רבים מהם, ונתחיל בהיכרות עם המרכיבים המרכזיים של סביבת הפיתוח.

תיבת הדו-שיח New Project



© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- תיבה זו נפתחת אוטומטית בעת הפעלת Visual Basic, או בעת בחירת הפריט New מהתפריט File
- תיבה זו מאפשרת לבחור פרויקט חדש מסוגים שונים
- תיבה זו כוללת Templates (תבניות) ו-Wizards (אשפים) המאפשרים קיצור משך הפיתוח

שורת התפריטים



- שורת התפריטים של Visual Basic כוללת מספר רב של פריטים. נציג את החשובים שבהם:
- התפריט File מאפשר יצירת פרויקטים חדשים, שמירת פרויקט פתוח, סגירת פרויקט, קומפילציה (הידור) של תוכנית, ועוד.
- התפריט Edit מאפשר פעולות עריכה שונות כגון העתקה, גזירה, הדבקה, חיפוש והחלפה אוטומטיים, השלמה אוטומטית של מילים שמורות ושמות פונקציות, ועוד.
- התפריט View שולט בעיקר על אופן התצוגה של Visual Basic ועל רכיבי ה-IDE המוצגים.
- התפריט Project כולל אפשרויות להוספת רכיבים לפרויקט, והסרה של רכיבים, שליטה על סוג הפרויקט ומאפייניו, ועוד.
- התפריט Format מאפשר לעצב את הפקדים על הטופס.
- התפריט Debug מאפשר להריץ ולנפות שגיאות מהתוכנית.
- התפריט Run מאפשר הרצה, השהייה והפסקה של התוכנית.
- התפריט Tools מאפשר הוספת שלדי פונקציות, מתודות ומאפיינים לרכיבי קוד בפרויקט, הוספת תפריטים לפרויקט, כמו גם אפשרויות מתקדמות להגדרת אופן ההתנהגות של Visual Basic.
- התפריט Add-Ins מאפשר לצרף תוספות של יצרן צד שלישי לממשק Visual Basic.
- התפריט Window מאפשר שליטה על אופן פריסת החלונות בחלון ה-IDE.
- התפריט Help מאפשר קבלת עזרה.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

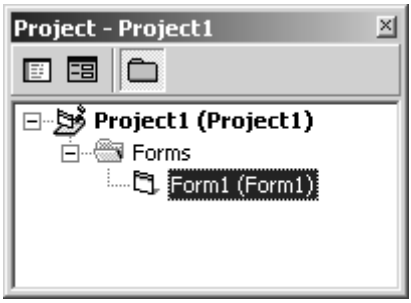
סרגל הכלים הסטנדרטי של Visual Basic



- סרגל הכלים הסטנדרטי כולל את אפשרויות התפריט השימושיות ביותר
- ניתן להוסיף סרגלי כלים נוספים לתצוגה ע"י בחירה מתאימה מתוך התפריט View, או ע"י לחיצה ימנית על אזור סרגלי הכלים ובחירה בתפריטים המתאימים.
- להלן הסבר כללי על פריטי סרגלי הכלים, שחלק לא קטן מהם אתה מכיר:
- הפריט  מוסיף פרויקט נוסף
- הפריט  מוסיף רכיבים לפרויקט
- הפריט  מוסיף תפריטים לתוכנית
- הפריטים  מאפשרים פתיחה ושמירה של פרויקט
- הפריטים  מאפשרים גזירה, הדבקה, העתקה, חיפוש והחלפה של טקסט
- הפריטים  מאפשרים ביטול והפעלה מחדש
- הפריטים  מאפשרים הפעלה, השהייה והפסקה של התוכנית
- הפריט  מציג/מסתיר את ה-Project Explorer
- הפריט  מציג/מסתיר את ה- Properties Window
- הפריט  מציג/מסתיר את ה-Form Layout Window
- הפריט  מציג/מסתיר את ה-Toolbox

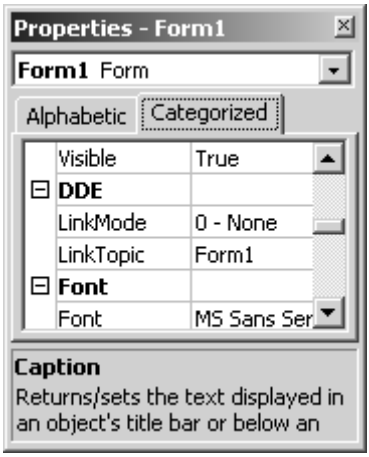
© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

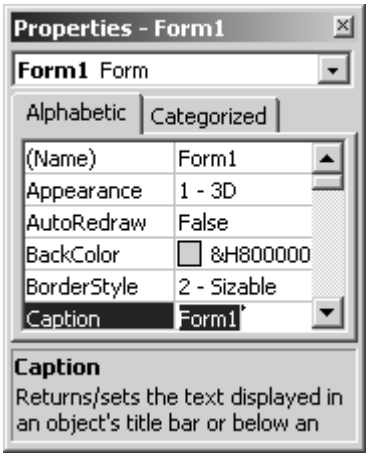
חלון ה-Project Explorer



- בחלון זה רואים את רכיבי הפרויקט
- כאשר מנהלים מספר פרויקטים במקביל, רואים אותם ב-Project Explorer תחת Group אחד

חלון ה-Properties

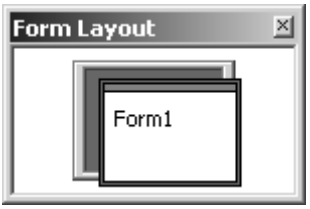




- חלון ה-Properties מציג את מאפייני הרכיב או האובייקט הנבחר.
- האובייקט הנבחר יכול להיות Form (טופס), Control (פקד), הפרויקט עצמו, או קבוצת רכיבים.
- ניתן לצפות במאפיינים בסדר אלפביתי או מסודרים לפי קטגוריות.

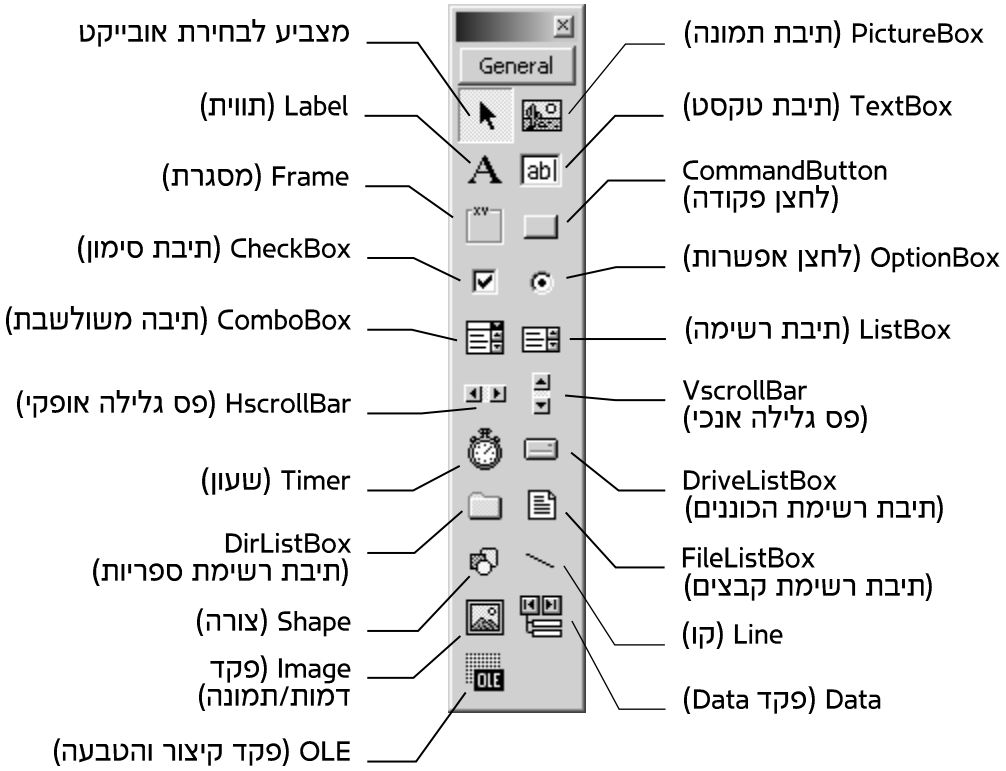
© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

חלון ה-Form Layout



- ניתן לראות ולקבוע כיצד ייפרסו הטפסים על המסך בעת פתיחתם Form Layout-ב
- ניתן לקבוע את אופן פריסת הטפסים מחלון זה או מתוך חלון מאפייני הטופס

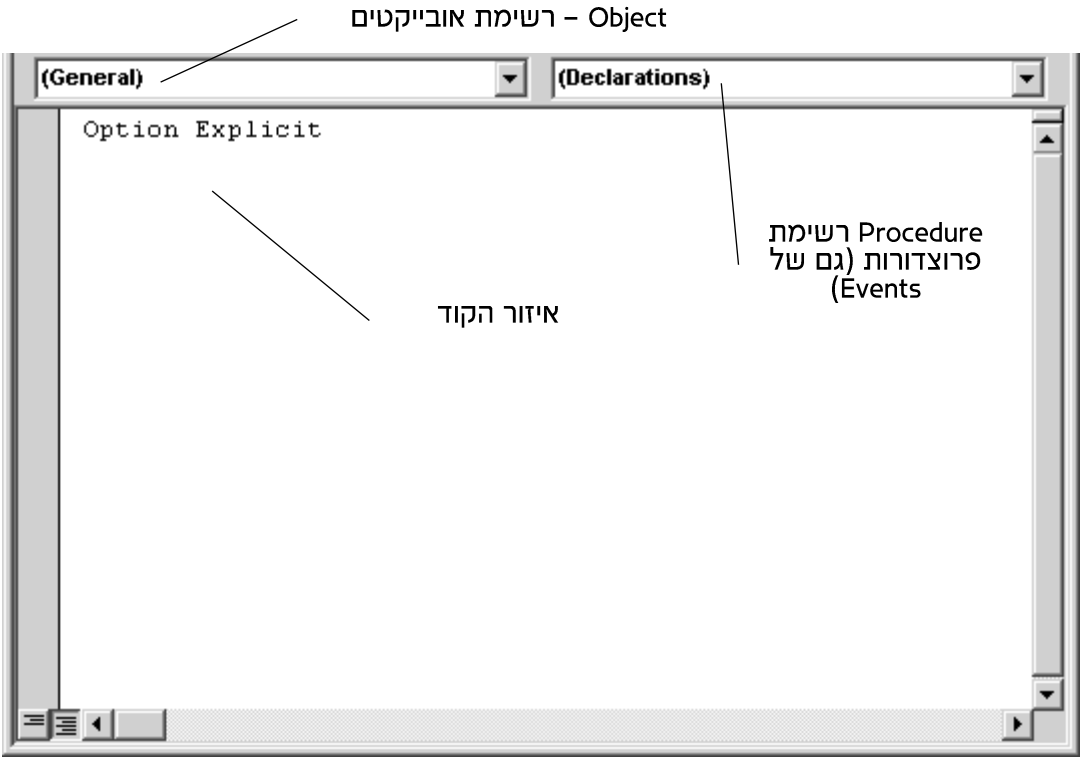
ה-Toolbox (ארגז הכלים)



© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ה-ToolBox של Visual Basic מציג במצב התחלתי את הפקדים הסטנדרטיים של Visual Basic
- ניתן להוסיף פקדים נוספים ל-ToolBox
- ניתן לארגן את הפקדים השונים לפי קבוצות

ה-Code Editor (עורך הקוד)



- במרכז המסך ממוקם חלון הקוד – כאן רושמים את שורות הקוד השונות
- התיבה המשולבת (ComboBox) משמאל מציגה את רשימת האובייקטים הזמינים לעריכת קוד
- התיבה המשולבת (ComboBox) מימין מציגה את אירועי האובייקט שנבחר, או את הפרוצדורות וההצהרות כאשר נבחר (General)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 2:

טפסים ותפריטים

בשיעור זה:

- תבין מהו פרויקט ב-Visual Basic
- תכיר את הפרויקט מהסוג Standard EXE
- תכיר את מבנה החלון בסביבת Microsoft Windows
- תבין את המושגים Properties, Methods ו-Events
- תכיר את המושגים Design Time ו-Run Time
- תכיר את שני סוגי ה-Forms (טפסים) העיקריים ב-Visual Basic: טופס רגיל וטופס MDI
- תלמד לעבוד עם Menus (תפריטים)
- תכיר את הפונקציות Print ו-MsgBox
- תלמד להוסיף הערות לקוד
- תלמד כיצד ניתן לשלוט על מיקום ופריסת הטפסים ביחס למסך, ביחס לחלון אב, וביחס לחלון MDI
- תכיר את סוגי הקבצים המאחסנים את נתוני הפרויקט ואת נתוני ה-Forms

פרויקט

- פרויקט ב-Visual Basic הוא אוסף קבצים המאוגדים כקבוצה אחת לשם יצירת תוכנית מסוג כלשהו
- ב-Visual Basic ניתן ליצור פרויקטים מסוגים שונים ולנהל מספר פרויקטים
- פרויקט ב-Visual Basic מאוחסן בקובץ שהסיומת שלו היא vbp.*. קובץ זה הוא קובץ טקסט המכיל מידע מגוון אודות הפרויקט

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- לעתים יש צורך לערוך ידנית את קובץ הפרויקט. למשל, כשמידע מסוים בקובץ שגוי וגורם לבעיות בטעינת הפרויקט

יצירת פרויקט חדש

- בעת הפעלת Visual Basic נפתח באופן אוטומטי חלון New Project
- מתוך חלון ה-New Project ניתן לבחור את סוג הפרויקט שרוצים לפתח
- לשם יצירת פרויקט חלונות סטנדרטי יש לבחור את Standard EXE
- ניתן להפעיל את חלון ה-New Project גם מתוך התפריט File
- הפרויקט ורכיביו מוצגים בחלון ה-Project Explorer
- לאחר סימון הפרויקט ב-Project Explorer ניתן לקבוע את שמו ב-Properties Window
- ניתן לקבוע את שם הפרויקט ומאפיינים נוספים של הפרויקט ע"י בחירת *[Project name]* Properties... מתוך התפריט Project, כאשר [Project Name] הוא שם הפרויקט שבפיתוח. נכיר את המאפיינים השונים של הפרויקט בהמשך הספר

הרצת פרויקט





- ניתן להריץ את התוכנית שבפיתוח ע"י לחיצה על הכפתור Start או ע"י לחיצה על המקש F5
- השלב שבו מפתחים את הפרויקט, נקרא Design Time
- השלב שבו הפרויקט נמצא במצב ריצה נקרא Run Time
- כדי להפסיק (באופן לא מסודר) את ריצת התוכנית שבפיתוח יש ללחוץ על הלחצן Stop
- לחיצה על Stop עוצרת באופן מידי את התוכנית תוך דילוג על אירועי הסיום המובנים בתוכנית

שמירת פרויקט

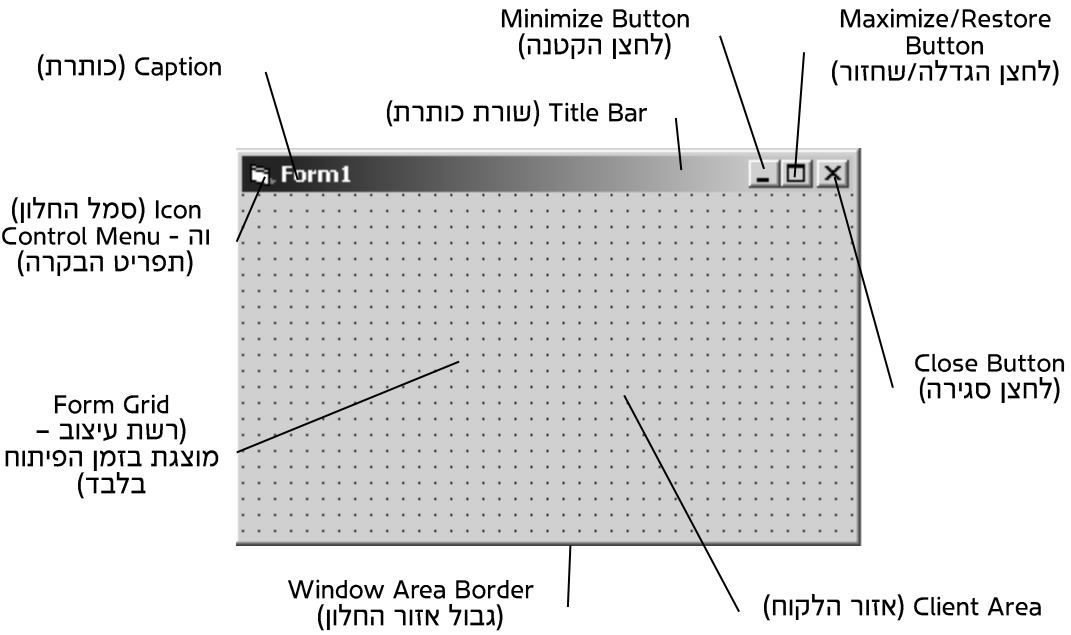
- הנתונים אודות הפרויקט עצמו ומאפייניו מאוחסנים בקובץ שהסיומת שלו היא *.vbp
- במקרה הצורך, ניתן לערוך קובץ זה באמצעות ה-Notepad
- קובצי .vbp ו-.frm הם קובצי טקסט. כאשר טופס מכיל מידע בינארי, כגון תמונות, נשמר לצדו גם קובץ .frx, שיש לשמור יחד איתו.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

ה-Form הסטנדרטי

- בעת יצירת פרויקט מסוג Standard EXE נוסף Form רגיל לפרויקט
- ניתן להוסיף Forms נוספים ע"י בחירת Add Form מהתפריט Project
- טפסים נשמרים בקבצים בעלי הסיומת *.frm
- קובץ זה הוא קובץ טקסט הכולל את הקוד השייך לטופס (קוד זה מוצג בעת טעינת הטופס ב-Visual Basic), כמו גם נתונים נוספים הממוקמים בראש הקובץ, כגון מאפיינים (שגם הם מוצגים בחלון המאפיינים)
- אם קיים צורך, ניתן לערוך קובץ זה ידנית מתוך ה-Notepad

מרכיבי Form



© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Properties (מאפיינים), Methods (שיטות), ו-Events (אירועים)

- **Property** (מאפיין) של אובייקט הוא ערך מסוג מסוים הקובע פן מסוים בהתנהגות האובייקט. למשל, המאפיין Caption של Form קובע את הטקסט המופיע בפס הכותרת של הטופס.
- **Method** (שיטה) של אובייקט היא שגרה המופעלת מתוך האובייקט ומשפיעה על אופן תפקודו. למשל, המתודה Move של טופס משנה את מיקומו על המסך.
- **Event** (אירוע) הוא הודעה על התרחשות הקשורה לאובייקט. שגרת הטיפול באירוע (Event Handler) מופעלת בתגובה לאירוע. למשל, האירוע Load של Form מופעל עם טעינת הטופס.
- לרוב האובייקטים (כולל טפסים ופקדים) קיימים Properties, Methods ו-Events
- הכרה של Properties, Methods ו-Events של Form או Control חוסכת לימוד חוזר שלהם עבור אובייקטים שמכירים לאחר מכן, כיוון שאופן ההתנהגות הוא בדרך כלל זהה.

שים לב: בסעיפים הבאים אפרט את חלק ניכר מהמאפיינים, השיטות והאירועים של טופס. לעתים קרובות המאפיינים, השיטות והאירועים הללו זהים בין אובייקטים שונים, ולכן לא אסביר אותם מחדש כאשר אסביר לך אודות אובייקטים נוספים, כגון טופס MDI, CommandButton וכו', אלא אם יש הבדל באופן פעולתם העקרוני.

Properties של Form

- **Name**: השם המוקצה לאובייקט ה-Form. באמצעות שם זה פונים למאפייני ה-Form או למתודות שלו
- **BackColor**: קובע את צבע הרקע של ה-Form
- **BorderStyle**: קובע את סגנון הגבול של הטופס
- **Caption**: קובע את הטקסט המוצג בפס הכותרת
- **ControlBox**: קובע האם יוצגו תפריטי הבקרה
- **Enabled**: קובע האם ה-Form יהיה פעיל או לא (Enabled או Disabled). Form פעיל יכול להגיב לאירועים המתרחשים, כגון לחיצת עכבר וכו'
- **FillColor**: צבע המילוי עבור צורות המצוירות באמצעות שגרות גרפיות על גבי ה-Form

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **FillStyle**: סגנון המילוי (קיימים שמונה סגנונות מילוי שונים: Solid - מלא, Transparent - שקוף, HorizontalLine - קווים אופקיים, ועוד...)
- **Font**: סגנון הגופן עבור פעולות גרפיות הכותבות טקסט על גבי ה-Form
- **ForeColor**: קובע את צבע הקידמה, כלומר, את הצבע באמצעותו יצוירו גבולות הצורות הגרפיות והטקסט
- **Height**: גובה ה-Form
- **Icon**: סמל הטופס
- **KeyPreview**: קובע אם ה-Form יקבל את אירועי המקלדת לפני הפקד בעל המיקוד (Focus).
- **Left**: מיקום נקודת הקצה השמאלית של ה-Form
- **MaxButton**: קובע האם כפתור ההגדלה שבתפריט הבקרה יהיה פעיל או לא
- **MDIChild**: קובע האם הטופס יהיה טופס בן של טופס ה-MDI בתוכנית
- **MinButton**: קובע האם כפתור המזעור בתפריט הבקרה יהיה פעיל
- **Moveable**: קובע האם ה-Form יהיה נייד במהלך ריצת התוכנית
- **NegotiateMenus**: קובע האם ה-Form ישתף תפריטים של האובייקטים שבתוכו בשורת התפריטים שלו
- **Picture**: הוספת תמונה לרקע של הטופס
- **RightToLeft**: קובע את כיוון הצגת הטקסט
- **ScaleHeight**: גובה אזור הלקוח ב-Form לפי יחידות המדידה המצוינות ב-ScaleMode
- **ScaleLeft**: ערך קצהו השמאלי של אזור הלקוח ב-Form
- **ScaleMode**: קובע את יחידות המדידה של אזור הלקוח (Client Area). ניתן גם להגדיר יחידות מדידה מותאמות אישית, ולקבוע את הערכים ScaleLeft, ScaleTop, ScaleHeight ו-ScaleWidth.
- **ScaleTop**: ערך קצהו העליון של אזור הלקוח ב-Form
- **ScaleWidth**: רוחב אזור הלקוח של ה-Form לפי יחידות המדידה שנקבעו ב-ScaleMode
- **ShowInTaskbar**: קובע האם יוצג סמל התוכנית בשורת המשימות
- **StartUpPosition**: קובע את מיקום ה-Form ביחס למסך או לחלון האב
- **Top**: מיקום נקודת הקצה העליונה של ה-Form

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Visible**: קובע האם ה-Form יהיה נראה (Visible) או לא
- **Width**: רוחב ה-Form
- **WindowState**: קובע את מצבו ההתחלתי של ה-Form (מוגדל/מוקטן/נורמלי)

Methods של Form

- **Cls**: מנקה ציורים וטקסט שנוספו ל-Form בזמן הריצה
- **Hide**: מסתיר את הטופס אך לא פורק (Unloading) אותו
- **Move**: משנה את מיקומו ו/או את גודלו של ה-Form
- **PopupMenu**: מציג תפריט מוקפץ
- **Refresh**: מרענן את תוכן ה-Form כך שיצטייר מחדש
- **SetFocus**: מעביר את ה-Focus (מיקוד) ל-Form

Events של Form

- **Activate**: מתרחש כשה-Form הופך להיות פעיל
- **Click**: מתרחש כשלחצן העכבר נלחץ ומשוחרר על גבי ה-Form
- **DblClick**: מתרחש בעת לחיצה כפולה על הטופס
- **Deactivate**: מתרחש כשה-Form הופך ללא פעיל
- **GotFocus**: מתרחש כאשר ה-Form מקבל את ה-Focus (מיקוד)
- **Initialize**: מתרחש בעת אתחול ה-Form
- **KeyDown**: מתרחש בעת לחיצת מקש
- **KeyPress**: מתרחש בעקבות הקשה על מקש המייצר תו
- **KeyUp**: מתרחש לאחר שחרור מקש
- **Load**: מתרחש בעת טעינת ה-Form
- **LostFocus**: מתרחש בעת איבוד ה-Focus (מיקוד)
- **MouseDown**: מתרחש בעת לחיצה על אחד הלחצנים בעכבר
- **MouseMove**: מתרחש בעת תזוזת עכבר
- **MouseUp**: מתרחש בעת שחרור אחד הלחצנים בעכבר
- **Paint**: מתרחש בעת ציור תוכן הטופס מחדש
- **QueryUnload**: מתרחש לפני פירוק (Unloading) של הטופס

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Resize**: מתרחש בעת שינוי גודל ה-Form
- **Terminate**: מתרחש בעת סיום פעולת הטופס
- **Unload**: מתרחש בעת פריקת הטופס

האירועים Initialize, Terminate, Load, Unload, Activate, Deactivate, GotFocus, LostFocus ו-QueryUnload

- האירוע Initialize מתרחש בעת יצירת האובייקט בזיכרון (תמיד מתרחש לפני Load). לאחר Initialize ה-Form קיים כאובייקט אך עדיין לא כחלון
- האירוע Terminate מתרחש בעת שחרור האובייקט מהזיכרון (מתרחש תמיד אחרי Unload, אך אינו מתרחש אם התוכנית הסתיימה בצורה לא נורמלית)
- האירוע Load מתרחש בעת טעינת האובייקט מהזיכרון והפיכתו לחלון – כלומר בשלב זה, לחלון יש ידית (hWnd), והפקדים שבו נוצרו ונטענו
- האירוע Unload מתרחש בעת פריקת החלון וכל הפקדים שבו. עם זאת, ה-Form עדיין קיים כאובייקט בזיכרון
- האירוע QueryUnload מתרחש לפני פריקת ה-Form (Unloading)
- Activate מתרחש בעת מעבר ל-Form מטופס אחר באותה התוכנית
- Deactivate מתרחש בעת מעבר ל-Form אחר באותה התוכנית
- GotFocus ו-LostFocus הם האירועים המתרחשים בעת קבלת המיקוד ואיבודו, בהתאמה. SetFocus היא שיטה להעברת המיקוד.

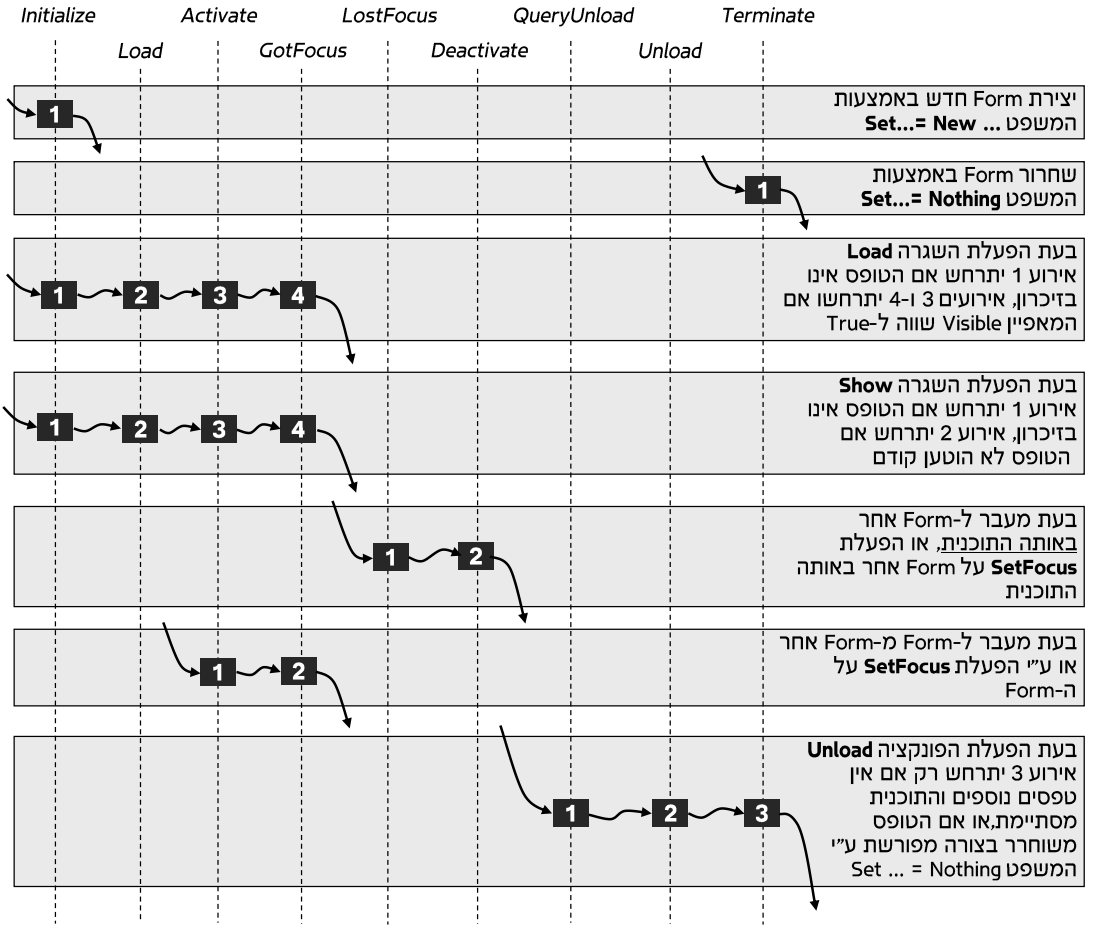
ארבעת המצבים ש-Form יכול להימצא בהם

קיימים ארבעה מצבים שבהם יכול להימצא Form:

- **Created but not loaded**: ה-Form נוצר כאובייקט בזיכרון אך לא הוטען עדיין כחלון
- **Loaded but not shown**: ה-Form הוטען כחלון אך לא הוצג
- **Shown**: ה-Form הוצג
- **Memory and resources completely reclaimed**: הזיכרון והמשאבים שהוקצו עבור ה-Form שוחררו במלואם

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- הגרף הבא מתאר את סדר התרחשות האירועים הנ"ל בתגובה לפעילויות מסוימות של המשתמש או פונקציות שהמתכנת מפעיל.



האירועים Click ו-DblClick

- האירוע Click באובייקט מתרחש לאחר הורדת לחצן עכבר (MouseDown) ושחרור אותו לחצן (MouseUp) על גבי אותו אובייקט

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- האירוע DblClick באובייקט מתרחש לאחר לחיצת עכבר (MouseDown) שבאה בעקבות האירוע Click תוך פרק זמן הקצר או שווה לזה שמוגדר בלוח הבקרה (Control Panel) של המערכת

פונקציות ראשונות

בסעיף זה נכיר שתי דרכים להצגת הודעה.

הפונקציה Print

- הפונקציה Print מדפיסה טקסט על גבי הטופס

דוגמה

```
Print "Hello, world!"
```

- שורת קוד זו מדפיסה את המחרוזת Hello, world! על גבי ה-Form

הפונקציה MsgBox

- הפונקציה MsgBox מציגה תיבת הודעה למשתמש

דוגמה

```
MsgBox "Hello, world!"
```

- שורת קוד זו מציגה תיבת הודעה עם הכיתוב Hello, world!

דוגמה 1: התוכנית HelloWorld

- בדוגמה זו ניצור תוכנית המדפיסה את המחרוזת Hello, world! על גבי הטופס ובאמצעות תיבת הודעה.
- בהדרגה נוסיף פריטים לתוכנית זו, תוך כדי למידה של נושאים נוספים

> **שים לב**: פעולות שחוזרות על עצמן, כגון יצירת פרויקט, מתן שם לפרויקט וכו', אפרט פעם אחת בדוגמה הראשונה שבה נדרשת פעולה זו. אם בדוגמאות מתקדמות שכחת כיצד לבצע פעולה זו, חזור לדוגמאות הראשונות כדי לרענן את זיכרונך.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

1. **צור תוכנית מסוג Standard EXE:**
   - הפעל את Visual Basic
   - בחר באפשרות New Project מתוך התפריט File
   - בחר בפרויקט Standard EXE, ולאחר מכן לחץ על OK
2. **הקצה שם לפרויקט:**
   - בחר באפשרות Project1 Properties... מתוך התפריט Project
   - בשדה Project Name רשום HelloWorld (זהו שם הפרויקט שבחרתי לפרויקט זה, כמובן שתוכל לבחור בשם אחר אם רצונך בכך)
3. **הקצה מאפיינים ל-Form:**
   - לחץ על ה-Form במקום כלשהו
   - מאפייני ה-Form צריכים להופיע בחלון המאפיינים (אם אינך רואה את חלון המאפיינים, בחר באפשרות Properties Window בתפריט View, או לחץ על F4)
   - הגדר את המאפיינים הבאים:

מאפיין	ערך
Name	FrmHello
Caption	Hello World Program

4. **שמור את הפרויקט:**
   - בחר באפשרות Save Project מתוך התפריט File
   - הקפד ליצור ספרייה נפרדת עבור כל פרויקט
   - שים לב ש-Visual Basic מציעה שמות התואמים את שמות הפרויקט והטופס כפי שהקצת להם בתוכנית

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

5. הוסף קוד:



- לחץ לחיצה כפולה על הטופס או על הלחצן View Code בחלון ה-Project Explorer
- בחלון ה-Code Editor (עורך הקוד) בחר ב-Click מתוך רשימת הפרוצדורות הזמינות והקלד את הקוד הבא:

```
Private Sub Form_Click()
   Print "Hello, world"!
End Sub
```

6. הרץ את התוכנית ובדוק אותה:

- לחץ על F5
- לחץ על ה-Form המוצג. הכיתוב Hello, world! אמור להופיע על הטופס.
- כל לחיצה נוספת תוסיף שורה נוספת של ההודעה על הטופס
- מזער את החלון והגדל אותו בחזרה – ההודעות נעלמו
- לחץ לחיצה נוספת על הטופס, שוב תיכתב ההודעה Hello, world! מיד לאחר השורה האחרונה שבה היא נרשמה קודם
- עצור את ריצת התוכנית ע"י סגירת החלון

7. הנחה את הטופס לזכור את תוכנו כתמונת מפת סיביות:

- שנה את המאפיין AutoRedraw של frmHello לערך True
- הרץ ובדוק את התוכנית שוב. שים לב שהפעם לאחר מזעור והגדלה של ה-Form תוכנו נשמר
- סיים את ריצת התוכנית

8. הפעלת תיבת הודעה בתגובה ללחיצת מקש:

- בחלון הקוד בחר ב- KeyPress מתוך רשימת הפרוצדורות הזמינות, והקלד את הקוד הבא:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Private Sub Form_KeyPress(KeyAscii As Integer(
   MsgBox "Hello, world"!
End Sub
```

9. הרץ את התוכנית ובדוק אותה:

- לאחר הרצת התוכנית לחץ על מקש כלשהו (מבין מקשי האותיות והמספרים)
- צריכה להופיע תיבת הודעה עם המסר Hello, world!

10. שמור את התוכנית:

- שמור את התוכנית ע"י בחירת Save Projectמתוך תפריט File
- בהמשך נשוב למספר שינויים בתוכנית זו
- אם תרצה תוכל לסגור את התוכנית ע"י בחירת Remove Project מתוך התפריט File או ע"י סגירת Visual Basic

הוספת הערות לקוד

- מומלץ להוסיף הערות כדי לשפר את הבנת הקוד
- הערות ב-Visual Basic ניתן להוסיף לאחר הסימן ' (גרש בודד)

המשך דוגמה 1

11. פתח את התוכנית HelloWorld:

- מצא את התוכנית בספרייה שבה שמרת אותה
- הקלק על הקובץ HelloWorld.vbp – התוכנית תיפתח אוטומטית באמצעות Visual Basic

12. הוסף הערות לקוד:

- הוסף את ההערות הבאות לקוד (ההערות נצבעות אוטומטית בצבע ירוק)

```
Private Sub Form_Click()
   Print "Hello, world!" 'Print the message on the form
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer(
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   MsgBox "Hello, world!" 'Display a message box to the user
End Sub
```

13. **מקם את ה-Form במרכז המסך:**
    - קבע את המאפיין StartUpPosition של frmHello לערך 2-CenterScreen
14. **קבע את גבול החלון כקבוע:**
    - שנה את המאפיין BorderStyle של frmHello לערך 2-FixedSingle
15. **הרץ ובדוק את התוכנית:**
    - כאשר תריץ את התוכנית, תשים לב כי החלון מוקם במרכז המסך
    - נסה לשנות את גודל ה-Form – שים לב שזה בלתי אפשרי

MDI Form (טופס MDI)

- MDI הם ראשי התיבות של Multiple Document Interface – ממשק מרובה מסמכים
- לפרויקט ניתן להוסיף MDI Form אחד לכל היותר
- MDI Form הוא Form שיכול להכיל בתוכו מספר Form-ים
- במקרה כזה ה-MDI Form ייקרא חלון האב של החלונות הכפופים לו
- כדי להפוך Form לטופס השייך לחלון ה-MDI יש לשנות את המאפיין MDI Child של אותו Form ל-True

Properties של MDI Form

- רוב ה-Properties ה-Methods וה-Events של MDI Form זהים לאלה שבסעיף ה-Form הסטנדרטי
- **AutoShowChildren:** קובע האם חלונות בנים (MDI Children) יוצגו באופן אוטומטי בזמן הטעינה

Methods של MDI Form

- **Arrange:** מסדר את החלונות הבנים הפתוחים בתוך ה-MDI Form

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Menus (תפריטים)

- ניתן להוסיף תפריטים לטפסים ע"י בחירת Menu Editor... מתפריט Tools
- תפריט הוא כמו אובייקט רגיל ב-Visual Basic. ניתן לגשת אל מאפייניו באמצעות שמו
- ניתן להוסיף & לפני אות מסוימת בכותרת תפריט על מנת להפוך אותו למקש מהיר (Accelerator)
- כדי ליצור קו מפריד בתפריט, יש ליצור פריט שה-Caption שלו הוא מקף רגיל – (הסימן מינוס)

Properties של Menu

- **Checked:** קובע האם יוצג הסימן V ליד פריט התפריט
- **NegotiatePosition:** קובע את האופן שבו יסודרו התפריטים על גבי החלון המכיל יחד עם תפריטים מאובייקטים שונים
- **Shortcut:** קובע מקשי קיצור עבור פריט התפריט
- **Tag:** מאפשר להצמיד מידע נוסף (מחרוזתי) לאובייקט התפריט
- **WindowList:** קובע אם אובייקט התפריט ינהל רשימת חלונות (מסוג MDI Child) הפתוחים בתוך MDI Form
- **Parent:** ייחוס לחלון האב של התפריט

Events של Menu

- **Click:** מתרחש בעת לחיצה על התפריט

עבודה עם צבעים

בסעיף זה נכיר מספר פונקציות וקבועים המחזירים ערכים המייצגים צבע.

הפונקציה QBColor

- פונקציה זו יודעת ליצור ערך המייצג צבע עבור 16 צבעים שונים מוגדרים מראש
- פונקציה זו מקבלת אחד מהערכים 0-15 ומחזירה ערך המייצג צבע

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה

```
Form1.BackColor = QBColor(8)
```

- בשורת קוד זו, נקבע צבע הרקע של Form1 לצבע Gray (אפור)

הפונקציה RGB

- הפונקציה RGB יוצרת צבע מותאם אישית
- הפונקציה מקבלת שלושה ערכים, כל ערך מייצג את רמת מרכיב הצבע – אדום, ירוק או כחול – בצבע התוצאה
- כל מרכיב יכול לנוע החל מהערך 0 עד לערך 255
- יש להפעיל את הפונקציה באופן הבא:

```
RGB(red_value, green_value, blue_value)
```

דוגמה

```
Form1.ForeColor = RGB(0, 0, 255)
```

- בשורת קוד זו נקבע צבע הקדמה (צבע הטקסט והגבולות של צורות המצוירות ע"י פונקציות גרפיות) לצבע כחול
- טיפים:
  - אם רמת כל הצבעים מוגדרת כ-0 נוצר צבע שחור
  - אם רמת כל הצבעים מוגדרת כ-255 נוצר צבע לבן
  - אם רמת כל הצבעים מוגדרת לערך זהה נוצר גוון אפור
  - ככל שהערכים גבוהים יותר הצבע בהיר יותר

אוסף הקבועים ColorConstants

- אוסף קבועים זה מכיל שמונה צבעים מוגדרים מראש

אוסף הקבועים SystemColorConstants

- אוסף של 25 קבועים שונים המגדירים צבעים כפי שהוגדרו ע"י המשתמש בלוח הבקרה (Control Panel) של המערכת (למעשה קיימים 27 אך שניים מהם זהים)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה 2 – התוכנית Colors

בדוגמה זו ניצור תוכנית MDI עם שלושה חלונות, ונבצע משחקי צבעים וגרפיקה פשוטים על הטפסים. כמו כן, נוסיף תפריטים לתוכנית

1. **צור תוכנית מסוג Standard EXE**
2. **הוסף Forms לפרויקט:**
   - בחר ב-Add Form מתוך התפריט Project
   - מתוך תיבת הדו-שיח Add Form בחר ב-Form (הפריט הראשון ברשימה)
   - חזור על פעולה זו כך שתוסיף טופס שלישי לתוכנית
3. **הוסף MDIForm לפרויקט:**
   - בחר ב-Add MDI Form מהתפריט Project
   - מתיבת הדו-שיח Add MDI Form בחר ב-MDI Form
4. **הקצה שמות לפרויקט ולטפסים:**
   - שנה את שמו של הפרויקט ל-Colors
   - שנה את שמות הטפסים לשמות frmQBColors, frmRGB ו-frmSystemColors
   - שנה את שם טופס ה-MDI ל-mdiColors
5. **שמור את הפרויקט**
6. **הגדר את המאפיינים הבאים:**

אובייקט	מאפיין	ערך
FrmQBColors	Caption	QBColors
	MDIChild	True
frmRGB	Caption	RGB
	MDIChild	True

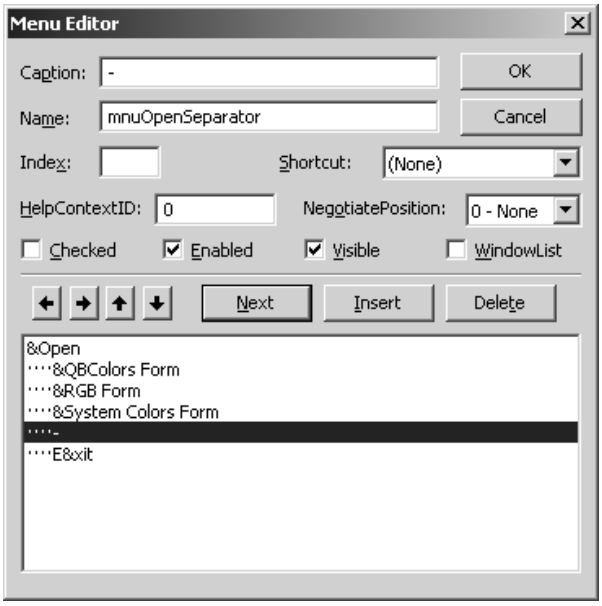
© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

FrmSystemColors	Caption	System Colors
	MDIChild	True

7. שנה את ה-Startup Object:

- פתח את Colors Properties... מתוך התפריט Project
- שנה את האובייקט בשדה :Startup Object כך שיהיה MDIColors

8. הוסף תפריט לטופס mdiColors:



- בחר ב-Menu Editor מתוך תפריט Tools
- בשדה Caption רשום &Open ובשדה Name רשום mnuOpen
- לחץ על Next
- השדות התרוקנו להזנת הפריט הבא
- בשדה Caption רשום QBColors Form, ובשדה Name רשום mnuOpenQBColorsForm
- השתמש בחץ המופנה כלפי ימין כדי להזיח את הפריט הזה מקום אחד ימינה, כך שיופיע כתת פריט של mnuOpen

שים לב: הסימן – (מינוס) יוצר הפרדה בתפריטים

- הוסף את הפריטים המופיעים ברשימה הבאה כך שהחלון ייראה כמו זה שמופיע כאן (בסוגריים רשומים השמות)

Caption ***Name***

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

&Open	mnuOpen	
	&QBColors Form	mnuOpenQBColorsForm
	&RGB Form	mnuOpenRGBForm
	&System Colors Form	mnuOpenSystemColorsForm
	-	mnuOpenSeparator
	E&xit	mnuOpenExit

Caption	***Name***	
&Window	mnuWindow	
	&Cascade	mnuWindowCascade
	Tile &Horizontally	mnuWindowTileHorizontally
	Tile &Vertically	mnuWindowTileVertically
	&Arrange Icons	mnuWindowArrangeIcons

9. באופן דומה הוסף את התפריטים הבאים:

◄ לטופס frmQBColors:

Caption	***Name***	
&Colors	mnuColors	
	&Red	mnuColorsRed
	&Green	mnuColorsGreen
	&Blue	mnuColorsBlue
	-	mnuColorsSeparator1
	B&lack	mnuColorsBlack
	&White	mnuColorsWhite
	-	mnuColorsSeparator2
	&Quit	mnuColorsQuit

◄ לטופס frmRGB:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Caption	*Name*	
&Colors	mnuColors	
	&Red	mnuColorsRed
	&Green	mnuColorsGreen
	&Blue	mnuColorsBlue
	-	mnuColorsSeparator1
	&Cyan	mnuColorsCyan
	&Magenta	mnuColorsMagenta
	&Yellow	mnuColorsYellow
	-	mnuColorsSeparator2
	&Quit	mnuColorsQuit

◄ לטופס frmSystemColors:

Caption	*Name*	
&Colors	mnuColors	
	&Button Face	mnuColorsButtonFace
	&Tool Tip	mnuColorsToolTip
	&Desktop	mnuColorsDesktop
	-	mnuColorsSeparator
	&Quit	mnuColorsQuit

10. הוסף קוד:

◄ לטופס mdiColors:

```
Private Sub mnuOpenExit_Click()
   Unload Me
End Sub

Private Sub mnuOpenQBColorsForm_Click()
   ' Loads and displays frmQBColors
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   Load frmQBColors
   frmQBColors.Show
End Sub

Private Sub mnuOpenRGBForm_Click()
   ' Loads and displays frmRGB
   Load frmRGB
   frmRGB.Show
End Sub

Private Sub mnuOpenSystemColorsForm_Click()
   ' Loads and displays frmSystemColors
   Load frmSystemColors
   frmSystemColors.Show
End Sub

Private Sub mnuWindowArrangeIcons_Click()
   Arrange vbArrangeIcons
End Sub

Private Sub mnuWindowCascade_Click()
   Arrange vbCascade
End Sub

Private Sub mnuWindowTileHorizontally_Click()
   Arrange vbTileHorizontal
End Sub

Private Sub mnuWindowTileVertically_Click()
   Arrange vbTileVertical
End Sub
```

◄ לטופס frmQBColors:

```
Private Sub mnuColorsBlack_Click()
   BackColor = QBColor(0) ' Black
End Sub

Private Sub mnuColorsBlue_Click()
   BackColor = QBColor(1) ' Blue
End Sub
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Private Sub mnuColorsGreen_Click()
   BackColor = QBColor(2) ' Green
End Sub

Private Sub mnuColorsQuit_Click()
   Unload Me
End Sub

Private Sub mnuColorsRed_Click()
   BackColor = QBColor(4) 'Red
End Sub

Private Sub mnuColorsWhite_Click()
   BackColor = QBColor(15) 'white
End Sub
```

◄ לטופס frmRGB:

```
Private Sub mnuColorsMagenta_Click()
   BackColor = RGB(255, 0, 255) 'Magenta
End Sub

Private Sub mnuColorsBlue_Click()
   BackColor = RGB(0, 0, 255) ' Blue
End Sub

Private Sub mnuColorsCyan_Click()
   BackColor = RGB(0, 255, 255) ' Cyan
End Sub

Private Sub mnuColorsGreen_Click()
   BackColor = RGB(0, 255, 0) ' Green
End Sub

Private Sub mnuColorsQuit_Click()
   Unload Me
End Sub

Private Sub mnuColorsRed_Click()
   BackColor = RGB(255, 0, 0) ' Red
End Sub
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Private Sub mnuColorsYellow_Click()
   BackColor = RGB(255, 255, 0) ' Yellow
End Sub
```

◄ לטופס frmSystemColors:

```
Private Sub mnuColorsButtonFace_Click()
   BackColor = SystemColorConstants.vbButtonFace
End Sub

Private Sub mnuColorsDesktop_Click()
   BackColor = SystemColorConstants.vbDesktop
End Sub

Private Sub mnuColorsQuit_Click()
   Unload Me
End Sub

Private Sub mnuColorsToolTip_Click()
   BackColor = SystemColorConstants.vbInfoBackground ' ToolTip
background
End Sub
```

11. הרץ ובדוק את התוכנית

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגילים

1. כתוב תוכנית בשם SimpleColors ובה שלושה Forms. קבע צבע רקע שונה לכל Form וטען את ה-Form השני בעת לחיצה על ה-Form הראשון, ואת ה-Form השלישי בעת לחיצה כפולה על ה-Form השני.
   בעת לחיצת מקש על ה-Form השלישי ייסגרו כל ה-Forms ותסתיים התוכנית.

2. כתוב תוכנית בשם TraceEvents ובה שלושה Forms: Form1, Form2 ו-Form3.
   ב-Form1 הוסף תפריט המאפשר את טעינת Form2 ו-Form3, ניקוי ה-Form (בצע ניקוי באמצעות השגרה Cls) וכן, אפשרות יציאה מהתוכנית (הסוגרת את כל ה-Forms). כמו כן, בטל את לחצן הסגירה ב-Form1 כך שניתן יהיה לסגור את התוכנית רק באמצעות האופציה המתאימה בתפריט, ולבסוף, הגדר את AutoRedraw ב-Form1 כ-True ע"מ שתוכן החלון יישמר לאחר הסתרה של חלק ממנו.
   בכל אחד מה-Forms האחרים (Form2 ו-Form3), הוסף שורות קוד המדפיסות על גבי Form1 את האירוע שהתרחש עבור כל אחד מהאירועים הבאים: Initialize, Load, QueryUnload, Unload, GotFocus, LostFocus, Activate ו-Deactivate. כך לדוגמה, באירוע Activate שב-Form3 תרשום:

```
Private Sub Form_Activate()
   Form1.Print "Form3: Activate"
End Sub
```

   תוכל להיעזר בחלון ה-Form Layout כדי להגדיר בקלות את מיקומי הטפסים על גבי המסך בעת הרצת התוכנית. לאחר הפעלת התוכנית, עקוב אחר סדר התרחשות ה-Events ב-Form2 ו-Form3 כפי שהם מודפסים על גבי Form1.

3. כתוב תוכנית בשם ArrangeForms עם MDIForm ושלושה Forms רגילים המוגדרים כ-MDIChild.
   הוסף ל-MDIForm תפריט לטעינת הטפסים הללו, וכן תפריט בשם Window המאפשר לפרוס את החלונות שבו באופנים שונים: Tile Horizontally, Tile Vertically, Cascade, ו-Arrange Icons (השתמש בשגרה Arrange של MDIForm לשם כך). כמו כן הגדר את האפשרות WindowList עבור התפריט Window כ-True על מנת שרשימת ה-Forms

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

הפתוחים תוצג בתפריט זה. (אל תשכח לשנות את ה-Startup Object ל-MDIForm בהגדרות הפרויקט).

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 3:

הפקדים הסטנדרטיים I

בשיעור זה:

- תכיר את ה-Controls הבאים:
  - Label (תווית)
  - TextBox (תיבת טקסט)
  - CommandButton (לחצן פקודה)
  - HScrollBar (פס גלילה אופקי)
  - VScrollBar (פס גלילה אנכי)
- תכיר את הפונקציות הבאות לעבודה עם מחרוזות: Len, Val, Str, RTrim, LTrim, Trim, Right, Left, Mid, ו-Asc
- תלמד לשרשר מחרוזות ולהוסיף מעברי שורה
- תכיר את התנאי If... Then... End If

הערה: בדרך כלל, אני לא חוזר על Properties, Methods ו-events שהזכרתי בטפסים שפעולתם זהה לאופן הפעולה ב-Controls שבפרק זה. בכל Control אני מציין את ה-Properties, Methods ו-Events שלא צוינו קודם.

Label (תווית)

- Label נועד בעיקר להצגת טקסט, בד"כ בצמוד לפקדים אחרים

Properties של Label

- **Alignment:** קובע את סוג יישור הטקסט בתוך ה-Label

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Appearance:** קובע את אופן התצוגה של ה-Label (אפקטים תלת ממדיים או ללא אפקטים תלת ממדיים)
- **AutoSize:** קובע האם גודל ה-Label ישתנה באופן אוטומטי בהתאם לתוכן
- **BackStyle:** קובע את סגנון הרקע (Transparent – שקוף, או Opaque – אטום, לא שקוף)
- **Caption:** טקסט התוכן המוצג ב-Label
- **Container:** קובע את ה-Container Control של ה-Label (Control Container הוא Control מכולה, כלומר Control המסוגל להכיל בתוכו פקדים אחרים. מהפקדים הסטנדרטיים, PictureBox ו-Frame הם ה-Control Containers היחידים)
- **Index:** מספרו הסידורי של ה-Label אם הוא שייך למערך של Labels
- **RightToLeft:** קובע את כיוון קריאת הטקסט
- **TabIndex:** קובע את מיקומו הסידורי של ה-Label בסדר מעבר הטאב. כיוון שה-Label עצמו אינו מקבל מיקוד, מקש גישה שהוגדר לו יכול להעביר את המיקוד לפקד הבא בסדר הטאב
- **Tag:** מאפשר להצמיד מידע נוסף ל-Label
- **ToolTipText:** טקסט עזרה המוצג כאשר סמן העכבר ממוקם מעל ה-Label
- **UseMnemonic:** קובע האם להתייחס לתו & כתו המאפשר הגדרת מאיץ מקלדת או להציגו כמו שהוא
- **WordWrap:** קובע האם לאשר גלישת טקסט הארוך מרוחב ה-Label

Events של Label

- **Change:** מתרחש כאשר משתנה תוכן ה-Label

TextBox (תיבת טקסט)

- השימוש הנפוץ ב-TextBox הוא לקבלת קלט מהמשתמש

Properties של TextBox

- **HideSelection:** קובע האם טקסט מסומן יישאר מואר כאשר הפוקוס עובר ל-Control אחר

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Locked:** קובע האם ה-TextBox יכול לקבל פעולות עריכה
- **MaxLength:** קובע את המספר המקסימלי של תווים שניתן להזין ל-TextBox (0 מציין אי הגבלה)
- **MultiLine:** קובע אם ה-TextBox יכול להיות מרובה שורות
- **PasswordChar:** קובע איזה תו יחליף את התו המוקלד (בתצוגה בלבד) במקרה שה-TextBox משמש לקבלת סיסמה
- **ScrollBars:** קובע אילו פסי גלילה יופיעו ב-TextBox כאשר MultiLine=True
- **TabStop:** קובע אם סדר מעבר הטאב יפעל גם על ה-TextBox
- **Text:** מכיל את הטקסט הנמצא ב-TextBox

Events של TextBox

- **Change:** מתרחש בכל פעם שהטקסט שונה (לאחר שהשינוי הוצג ב-TextBox)
- **KeyPress:** מתרחש בעת הקשה המייצרת קוד תו ANSI, לרבות תווי בקרה מסוימים כגון Backspace ו-Enter. מאפשר לטפל בתו לפני הצגתו ב-TextBox

CommandButton (לחצן פקודה)

- שימושו הנפוץ של CommandButton הוא לאפשר למשתמש פעולה מסוימת המשויכת ל-CommandButton. לפיכך, בדרך כלל, מגיבים לאירוע Click ב-CommandButton

Properties של CommandButton

- **Cancel:** קובע אם ה-CommandButton ידמה לחיצה עליו בתגובה ללחיצה על המקש Escape
- **Default:** קובע אם ה-CommandButton ידמה לחיצה עליו בתגובה ללחיצה על המקש Enter (Return)
- **DisabledPicture:** קובע איזה תמונה תוצג על גבי ה-CommandButton כאשר הוא במצב Disabled, כלומר, כאשר המאפיין Enabled שווה ל-False (יש לוודא שהמאפיין Style של ה CommandButton יהיה Graphical)
- **DownPicture:** קובע איזו תמונה תוצג על גבי הלחצן בזמן לחיצה עליו (שוב, יש להגדיר את המאפיין Style של ה-CommandButton ל-Graphical בכדי שמאפיין זה יפעל)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Enabled:** קובע האם הלחצן יאופשר או לא (זהו אחד המאפיינים השימושיים ביותר של CommandButton)
- **MaskColor:** מאפשר להגדיר איזה צבע בתמונה יהיה שקוף (יש להגדיר גם UseMaskColor כ-True כדי שמאפיין זה יפעל)
- **Picture:** קובע איזו תמונה תוצג על גבי הלחצן במצב רגיל.
- **Style:** קובע את סגנון הלחצן. Standard – מציג רק טקסט ו-Graphical – משתמש גם בתמונות המוגדרות.
- **UseMaskColor:** קובע האם ה-CommandButton ישתמש בצבע המוגדר במאפיין MaskColor כצבע שקוף.

HScrollBar (פס גלילה אופקי) ו-VScrollBar (פס גלילה אנכי)

- פסי גלילה מאפשרים גלילת טקסט, תמונות וכו', וזהו בדרך כלל שימושם העיקרי

Properties של פסי גלילה

- **LargeChange:** קובע את גודל הקפיצה כאשר לחצן העכבר לוחץ במקום כלשהו בין ה-Bar לבין אחד החצים, או כאשר נלחץ אחד מהמקשים PageUp או PageDown
- **Max:** מגדיר את הערך המקסימלי (הקצה הימני בפס אופקי או הקצה התחתון בפס אנכי) של פס הגלילה
- **Min:** מגדיר את הערך המינימלי של פס הגלילה (אם Min גדול מ-Max פס הגלילה משנה את כיוון התקדמותו מימין לשמאל בפס אופקי, ומלמטה כלפי מעלה בפס אנכי)
- **SmallChange:** מגדיר את גודל הצעד הקטן עבור התקדמות ה-Bar

Events של פסי גלילה

- **Change:** מתרחש בעת שינוי בערכו של פס הגלילה.
- **Scroll:** מתרחש בעת גרירת ה-Bar שעל פס הגלילה באמצעות העכבר.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

פונקציות לעבודה עם מחרוזות

- בסעיף זה תכיר מספר פונקציות לעבודה עם מחרוזות

הפונקציה Len

- הפונקציה Len מאפשרת למצוא את אורכה של מחרוזת

דוגמה

```
Len("Hello, world!")
```

- בשורת קוד זו, הפונקציה Len מחזירה את המספר 13 שהוא אורך המחרוזת Hello, world!

הפונקציה Val

- הפונקציה Val מחזירה את הערך הנומרי של מחרוזת

דוגמה

```
Val("12abc")
```

- בשורת קוד זו, מוחזר הערך 12 ע"י הפונקציה Val

הפונקציה Str

- הפונקציה Str יוצרת מחרוזת הזהה לערך נומרי המועבר לה.

דוגמה

```
Str(12.45)
```

- פונקציה זו יוצרת את המחרוזת "12.45 " הכוללת רווח מוביל השמור לסימן המספר החיובי

הפונקציה UCase

- הפונקציה UCase מקבלת מחרוזת ומחזירה את אותה המחרוזת לאחר הפיכת האותיות הקטנות במחרוזת לאותיות גדולות

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה

```
UCase("I have something 4 u...")
```

- שורת קוד זו מחזירה את המחרוזת "I HAVE SOMETHING 4 U..."

הפונקציה LCase

- הפונקציה LCase מקבלת מחרוזת ומחזירה את אותה המחרוזת לאחר הפיכת האותיות הגדולות במחרוזת לאותיות קטנות

דוגמה

```
LCase("HOW R u 2 dayyYy...?")
```

- שורת קוד זו מחזירה את המחרוזת "how r u 2 dayyyy...?"

הפונקציות Trim, LTrim ו-Rtrim

- הפונקציה RTrim מעתיקה מחרוזת חדשה ממחרוזת נתונה, תוך הסרת כל הרווחים הנמצאים בקצה הימני של המחרוזת.
- הפונקציה LTrim מעתיקה מחרוזת חדשה ממחרוזת נתונה, תוך הסרת כל הרווחים הנמצאים בקצה השמאלי של המחרוזת.
- הפונקציה Trim מעתיקה מחרוזת חדשה ממחרוזת נתונה, תוך הסרת כל הרווחים שבקצות המחרוזת

דוגמה

```
LTrim ("      Hello      How r u       ")
```

- פונקציה זו תחזיר את המחרוזת "Hello   How r u"

הפונקציות Mid, Left ו-Right

- הפונקציה Left יוצרת תת מחרוזת שמאלית באורך נתון ממחרוזת נתונה
- הפונקציה Right יוצרת תת מחרוזת ימנית באורך נתון ממחרוזת נתונה
- הפונקציה Mid יוצרת תת מחרוזת המתחילה במיקום נתון, ובאורך נתון מתוך מחרוזת נתונה.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמאות

```
Right("Hello, world!",6)
```

- שורת קוד זו מחזירה את המחרוזת world!

```
Left("Hello, world!",4)
```

- שורת קוד זו מחזירה את המחרוזת Hell

```
Mid("Hello, world!",8,3)
```

- שורת קוד זו מחזירה את המחרוזת wor

האופרטור &

- האופרטור & מאפשר לשרשר מחרוזות

דוגמאות

```
"Ilan" & " " & "Amoyal"
```

- שורה זו הופכת למחרוזת "Ilan Amoyal"

```
txtOut.Text = txtInput1.Text & txtInput2.Text
```

- בשורת קוד זו, לתוך ה-txtOut מוזנת המחרוזת המשורשרת מ-txtInput1 ו-txtInput2

הקבוע vbCrLf

- הקבוע vbCrLf מייצג מעבר השורה

דוגמה

```
Text1.Text = Text1.Text & vbCrLf & "Hello, world!"
```

- שורת קוד זו מוסיפה מעבר שורה ל-Text1 ומיד לאחר מכן משרשרת את המחרוזת "Hello, world!"

הפונקציה Asc

- מחזירה את קוד ה ANSI-של התו הראשון במחרוזת, בהתאם לקידוד המערכת (לקבלת קוד Unicode משתמשים ב-AscW).

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמאות

```
Asc(“a”)
```

- בשורת קוד זו, הפונקציה Asc מחזירה את הערך 97, שזהו ערכו של התו “a” לפי טבלת ה-ASCII

```
Asc("Goal")
```

- בשורת קוד זו הפונקציה Asc מחזירה את 71, שזהו ערכו ה-ASCII של התו G (התו הראשון במחרוזת)

התנאי If...Then...End If

- תחביר התנאי הוא כדלקמן:

```
If condition Then
   do_something
End If
```

- התנאי (condition) יכול להיות מורכב מהקשרים And ו/או Or

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגילים

1. כתוב תוכנית בשם Focuses ובה ארבעה CommandButtons:
   - לחיצה על ה-CommandButton הראשון תעביר את ה-Focus לשני
   - לחיצה על השני תעביר את ה-Focus לשלישי
   - לחיצה על השלישי תעביר את ה-Focus לרביעי
   - לחיצה על הרביעי תחזיר את ה-Focus ל-CommandButton הראשון

   כל CommandButton בעת קבלת ה-Focus ידפיס על גבי ה-Form את שמו (Name) (לאחר ניקוי התוכן הקודם של ה-Form), וכמו כן, יעדכן את כותרת החלון (Form.Caption) כך שתהיה זהה לכיתוב (Caption) של הלחצן שקיבל את המיקוד.

2. כתוב תוכנית בשם ThreeColors ובה שלושה CommandButtons עם הכיתובים Red, Green ו-Blue. לחיצה על אחד הפקדים תשנה את צבע הרקע של ה-Form, ותעדכן את כותרת ה-Form בהתאם.

3. כתוב תוכנית בשם Concatenate והנח בה שלושה פקדי TextBox ו-CommandButton אחד. סמוך ל-TextBox הראשון מקם Label עם הכיתוב "Input1". סמוך ל-TextBox השני מקם Label עם הכיתוב "Input2", וליד השלישי מקם Label עם הכיתוב "Output". כמו כן, נעל את ה-TextBox השלישי מפני עריכה.
   לחיצה על ה-CommandButton תשרשר את שתי המחרוזות שבשני פקדי ה-TextBoxes.

4. כתוב תוכנית זהה לקודמת בשם Concatenate2 עם השינויים הבאים:
   - ה-CommandButton יהיה Default
   - יוכנס מעבר שורה בין שתי המחרוזות המשורשרות
   - לאחר השרשור, יימחקו תכני תיבות הקלט וה-Focus יוחזר לתיבת הקלט הראשונה.

5. כתוב תוכנית בשם TextBoxCopy:
   - על ה-Form הנח שני פקדי TextBox האחד לקלט (Input) והשני לפלט (Output).

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- התוכנית צריכה להעתיק את המחרוזת המוזנת כקלט ב-TextBox הראשון אל ה-TextBox של הפלט.
- כל העתקה תיכנס לשורה חדשה בפלט מבלי למחוק את השורות הקודמות.
- בתוכנית זו הוסף שלושה CommandButtons:
  - אחד להעתקה (זה יהיה פקד ה-Default)
  - שני לניקוי תיבת הפלט
  - ושלישי לסגירת התוכנית (זה יהיה פקד ה-Escape).
- לאחר העתקת הטקסט, יש להחזיר את ה-Focus לתיבת הקלט תוך סימון הטקסט הרשום בה.

6. כתוב תוכנית בשם DigitTextBox, שתכלול TextBox שיאפשר קלט של ספרות, מקש Backspace, ומקש Enter למעבר שורה. בנוסף הנח שני CommandButtons, האחד לניקוי ה-TextBox והשני ליציאה (שיגיב למקש ה-Escape בתוכנית).

7. כתוב תוכנית בשם SlideColors:
ב-Form הנח שלושה ScrollBars והגדר את הטווח שלהם מ-0 עד 255. כמו כן, הנח Label על הטופס. בעת שינוי ערכו של אחד מה-ScrollBars יש לעדכן את צבע הרקע של ה-Label באמצעות הפונקציה RGB, כך שה-ScrollBar הראשון יהווה את המרכיב Red בפונקציה, ה-ScrollBar השני יהווה את מרכיב ה-Green בפונקציה, וה-ScrollBar השלישי יהווה את מרכיב ה-Blue בפונקציה.
צור ממשק נוח למשתמש תוך שימוש ב-Labels לפי הצורך.

8. צור תוכנית זהה לתוכנית הקודמת בשם SlideColors2 תוך הוספת השיפורים הבאים:

- דאג לכך שבזמן שהתוכנית עולה, צבע ה-Label יהיה תואם לערכים שב-ScrollBars
- בסמוך לכל ScrollBar תוצב תיבת טקסט שתשקף את הערך שבאותו ScrollBar
- צבע הרקע של כל TextBox יהיה צבע המרכיב שאותו הוא מייצג
- כמו כן, נעל את תיבות הטקסט באמצעות המאפיין Locked, והגדר את המאפיין LargeChange של כל אחד מפסי הגלילה ל-10

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 4:

מרכיבי השפה I

בשיעור זה:

- תלמד להצהיר על משתנים ועל קבועים ותכיר את סוגם
- תבין את ההבדל בין משתנה מקומי בפרוצדורה או פונקציה לבין משתנה משותף במודול
- תבין את ההבדלים בין Public Dim ו-Private בהקשר של הצהרות משתנים
- תכיר את מבני הבקרה הקיימים ב-Visual Basic
- תכיר את הפונקציות InputBox, IIf ואת הרוטינה Beep

Variables (משתנים) ו-Constants (קבועים)

בסעיף זה נסקור את:

- סוגי המשתנים ב-Visual Basic
- אופן ההצהרה על משתנים
- אזורי הצהרה שונים במודול
- הסתרה או חשיפה של משתנים באמצעות המילים Private או Public (הרשאות גישה)

סוגי משתנים

- ב-Visual Basic קיימים סוגי משתנים לאחסון מספרים שלמים, מספרים ממשיים (עם נקודה צפה) וכן מחרוזות וטיפוסים נוספים.
- למספר סוגים ישנם תווים מיוחדים להצהרה על משתנים (Type-declaration character), תווים אלו הנם שריד של הגרסאות המוקדמות של Visual Basic. לסוגי הנתונים החדשים (למשל Byte ו-Boolean) מעצבי Visual Basic לא הקצו תווים מיוחדים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

(כנראה כיוון שזה רעיון גרוע מלכתחילה), בכל אופן עדיף להשתמש בהצהרה ברורה ולא בתווי הצהרה

- הסוג Variant הוא סוג כללי המאפשר להכיל בתוכו את כל סוגי המשתנים (מלבד מחרוזות בגודל קבוע)
- הסוג Variant מאפשר טיפול גמיש מאוד בנתונים, וכן פעילות אוטומציה בין יישומים שונים
- בטבלה הבאה מפורטים סוגי המשתנים השימושיים ביותר:

סוג	גודל אחסון בזיכרון	תחום ערכים	תו הצהרה
Byte	בית אחד	0 עד 255	אין
Boolean	2 בתים	True או False	אין
Integer	2 בתים	-32,768 עד 32,767	% (אחוז)
Long	4 bytes	-2,147,483,648 עד 2,147,483,647	& (אמפרסנד)
Single (מספר ממשי ברמת דיוק בודדת)	4 בתים	בתחום השלילי: -3.402823E38 עד -E-451.401298 בתחום החיובי: 1.401298E-45 עד E383.402823	! (סימן קריאה)
Double	8 בתים	בתחום השלילי:	# (סולמית)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

(מספר ממשי ברמת דיוק כפולה)		-1.79769313486231E308 עד E-4.94065645841247-324 בתחום החיובי: E-4.94065645841247 324 עד E3081.79769313486232	
Date	8 בתים	1 לינואר שנת 100 עד ה-31 לדצמבר שנת 9999	אין
Object	4 בתים	הפנייה כלשהי לאובייקט	אין
String (בגודל משתנה)	10 בתים + אורך המחרוזת	0 עד 2 מיליארד (בערך)	$ (דולר)
String (בגודל קבוע)	כאורך המחרוזת	1 עד 65,400 (בערך)	$ (דולר)
Variant (עם מספרים)	16 בתים	כל ערך מספרי עד מקסימום לתחום המספרים מסוג Double	
Variant (עם תווים)	22 בתים + אורך המחרוזת	אורך זהה כשל מחרוזת בגודל משתנה	

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

הצהרה על משתנים

- ניתן להצהיר על משתנים במקומות שונים בתוכנית:
  - ברמת המודול (חלק ה-General)
  - בתוך פרוצדורה או פונקציה
- תחביר ההצהרה על משתנים הוא:

```
Dim varname [As type]
```

- Dim זוהי המילה המורה ל-Visual Basic להקצות מקום אחסון למשתנה
- בתוך פרוצדורה או פונקציה משתמשים בדרך כלל ב-Dim להצהרה על משתנים.
- ברמת המודול ניתן להשתמש במילים Public או Private במקום Dim:
  - **Public (ציבורי):** מאפשר גישה למשתנה לקטעי קוד הנמצאים במודולים אחרים
  - **Private (פרטי):** מאפשר גישה למשתנה אך ורק לפרוצדורות והפונקציות הרשומות בתוך אותו מודול בו הוצהר המשתנה
  - אם משתמשים ב-Dim, ההחלטה אם המשתנה יהיה Private או Public במודול תלויה בסוג המודול שלו. למשל, עבור Form (שהוא למעשה Class Module) ברירת המחדל היא Private
- **varname:** זהו שם המשתנה
- **Type:** זהו סוג המשתנה, אם לא מצוין הסוג, המשתנה יהיה מסוג Variant (אלא אם נעשה שימוש ב-Def*Type* במודול – נושא זה אינו נידון בספר)
- ניתן להצהיר על Strings באורך קבוע באמצעות שימוש בסימן *

דוגמאות

- nGrade - משתנה Integer

```
Dim nGrade As Integer
```

- strFirstName - משתנה String

```
Private strFirstName As String
```

- m_strLName - משתנה String באורך קבוע של 20 תווים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Private m_strLName As String * 20
```

- lBillGatesSalary - משתנה מסוג Long

```
Public lBillGatesSalary As Long
```

- nStage – משתנה מסוג Integer

```
Dim nStage%
```

הצהרה על קבועים

- ניתן להצהיר על קבועים במקומות שונים בתוכנית:
  - ברמת המודול (חלק ה-General)
  - בתוך פרוצדורה או פונקציה
- תחביר ההצהרה על קבועים הוא:

```
[Private | Public] Const constname [As type] = expression
```

- המילה Const משמשת להצהרת קבועים
- קבועים על פי ברירת המחדל הם מסוג Private אלא אם נרשם Public במפורש
- ב-Class Module לסוגיו השונים (Form הוא דוגמה ל-Class Module) לא ניתן להצהיר על קבועים כ-Public (Class Modules נלמד בהמשך הספר)
- ניתן להצהיר על Consts גם בתוך פונקציות ופרוצדורות, אך הם אינם קיימים מחוץ לפונקציות או פרוצדורות אלה

דוגמאות

- קבוע Double המאותחל לערך 17

```
Const cdTaxPercentage As Double = 17#
```

- קבוע String המאותחל לערך "Sales"

```
Private Const cstrDepartment = "Sales"
```

- קבוע Long המאותחל לערך 4405

```
Public Const clErrNumber as Long = 4405
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Control Structures (מבני בקרה)

בסעיף זה נכיר את מבני הבקרה השונים ב-Visual Basic, ובהם צורות שונות לרישום תנאים, מבני לולאה ומבנה רב-ברירתי (Case).

המבנה If...Then...Else

צורה ראשונה

```
If condition Then [statements] [Else elsestatements]
```

- condition הוא התנאי
- statements הן המשפטים שיתבצעו אם התנאי מתקיים (אם יש מספר משפטים, יש להפריד ביניהם באמצעות נקודתיים)
- elsestatements אלו הם המשפטים שיתבצעו אם התנאי לא מתקיים (אם יש מספר משפטים, יש להפריד ביניהם באמצעות נקודתיים)

דוגמה 1

```
If nGrade > 90 Then MsgBox "Very good!" Else MsgBox "OK"
```

דוגמה 2

```
If nAge >= 18 And nAge <= 21 Then Print "Army": Print "You are
Great"
```

צורה שנייה

```
If condition Then
   [statements]
[ElseIf condition-n Then
   [else if statements]...
[Else
   [else statements]]
End If
```

- בצורה זו ניתן ליצור מספר הסתעפויות אפשריות
- ניתן לרשום בלוק משפטים לביצוע עבור כל תנאי

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- אין חובה לכלול את חלקי ה-Elself או ה-Else אך יש להקפיד לרשום End If בסוף המבנה

דוגמה

```
If nAge < 13 Then
   Print "Child"
ElseIf nAge < 18 Then
   Print "Teenager"
ElseIf nAge < 22 Then
   Print "Soldier"
Else
   Print "Big-Boy"
End If
```

המבנה While...Wend

- תחביר המבנה הוא:

```
While condition
   [statements]
Wend
```

- condition הוא התנאי הנבדק
- statements הם המשפטים המבוצעים אם התנאי הוא True
- ***טיפ:*** עדיף להשתמש בלולאה Do...Loop כיוון שהיא גמישה יותר

דוגמה

```
Dim nCounter As Integer
nCounter = 1
While nCounter < 10
   Print "Now counter is equal to " & str(nCounter)
   nCounter = nCounter + 1
Wend
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

המבנה Do...Loop

צורה ראשונה

```
Do [{While | Until} condition]
   [statements]
   [Exit Do]
   [statements]
Loop
```

- condition הוא התנאי הנבדק
- While משמעותו – "כל עוד התנאי מתקיים"
- Until משמעותו – "עד אשר יתקיים התנאי"
- בצורה זו התנאי נבדק לפני הכניסה ללולאה, כך שאפשר שהלולאה לא תתבצע כלל
- ניתן לצאת בכל שלב מהלולאה ע"י Exit Do (מה שלא ניתן לעשות ב-While...Wend)

צורה שנייה

```
Do
   [statements]
   [Exit Do]
   [statements]
Loop [{While | Until} condition]
```

- השוני היחידי בין צורת רישום זו לקודמת הוא בכך שגוף הלולאה במקרה זה מתבצע פעם אחת לפחות (כיוון שהתנאי נבדק רק בסיום הלולאה)

דוגמה

```
Dim strBoard As String
Dim nColomns As Integer, nRows As Integer
Const cnMaxColomns As Integer = 10, cnMaxRows As Integer = 10
nRows = 1
Do While nRows <= cnMaxRows
   nColomns = 1
   Do While nColomns <= cnMaxColomns
      strBoard = strBoard & Format(nRows * nColomns, "000\ ")
      nColomns = nColomns + 1
   Loop
   strBoard = strBoard & vbCrLf
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   nRows = nRows + 1
Loop
MsgBox strBoard
```

- קטע קוד זה יוצר מחרוזת הכוללת רווחים ומעברי שורות, ואשר מאחסנת בתוכה את לוח הכפל
- שים לב לשימוש בפונקציה Format (תוכל לקבל עזרה אודות פונקציה זו במערכת העזרה של Visual Basic)
- קטע קוד זה משתמש בשתי לולאות Do...Loop מקוננות לשם ביצוע המשימה
- בסוף פירמוט המחרוזת, מוצגת המחרוזת ע"י הפונקציה MsgBox

המבנה For...Next

- תחביר המבנה:

```
For counter = start To end [Step step]
   [statements]
   [Exit For]
   [statements]
Next [counter]
```

- הלולאה For מאפשרת ביצוע נוח של בלוק פקודות מספר קבוע של פעמים
- counter = start זוהי פעולת האתחול של מונה הלולאה
- end הוא הערך הסופי של מונה הלולאה
- step זהו ערך הקפיצה של מונה הלולאה לאחר כל איטרציה (ניתן לקבוע ערך שלילי ל-Step כך שמונה הלולאה יקבל ערכים הולכים וקטנים)

דוגמה

```
Dim i As Integer
For i = 1 To 5
   Print "The number is: " & str(i)
Next
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

המבנה Select Case

- המבנה Select...Case הוא מבנה רב ברירות המאפשר בחירת בלוק לביצוע, ע"פ בדיקת ערכו של משתנה מסוים
- תחביר:

```
Select Case testexpression
[Case expressionlist-n
   [statements-n]] ...
[Case Else
   [elsestatements]]
End Select
```

- testexpression הוא הביטוי שערכו נבדק
- expressionlist זוהי רשימת ערכים, שאם המשתנה הנבחן שווה לאחד מהערכים שברשימה, אותו בלוק השייך ל-Case זה יתבצע
- elsestatements – יתבצע בכל מקרה אחר, כלומר אם לא נמצא ערך מתאים ברשימות הערכים השונות
- ניתן להשתמש בביטוי val1 To val2 כדי ליצור טווח ערכים
- ניתן להשתמש בביטוי Is comparison כדי להפעיל בדיקת ערך בתוך Select

דוגמה

```
Dim nAge As Integer
nAge = Val(InputBox("Enter an age"))
Select Case nAge
Case 1
   MsgBox "One year old"
Case 2
   MsgBox "Two years old"
Case 3, 4, 5, 6
   MsgBox "Well, growing!"
Case 7 To 13
   MsgBox "Not a child anymore!"
Case 14, 15, Is = 18
   MsgBox "A stupid teenager?"
Case Else
   MsgBox "Well, very big!"
End Select
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בקטע קוד זה מוצג Message שונה על-פי המספר שהזין המשתמש.
- תיאור הפונקציה InputBox מובא בסעיף הבא

הפונקציות Beep, InputBox ו-IIf

הרוטינה Beep

- הרוטינה Beep משמיעה צפצוף
- תחביר:

```
Beep
```

הפונקציה InputBox

- הפונקציה InputBox מציגה הודעה למשתמש, ומאפשרת לו להזין נתונים בתיבה מתאימה
- תחביר הפקודה:

```
InputBox(prompt[, title] [, default] [, xpos] [, ypos] [,
helpfile, context])
```

- Prompt הוא ההודעה למשתמש
- Title הוא כותרת ההודעה
- default הוא המחרוזת שתוצג עם הפעלת ה-InputBox
- xpos ו-ypos מגדירים את מיקום ה-InputBox במסך
- helpfile ו-context מאפשרים הפעלת מערכת עזרה בתגובה ללחיצה על F1
- הפונקציה מחזירה String כך שיהיה צורך בהמרות במקרה של בקשה להזנת מספר מצד המשתמש

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה

```
strGender = InputBox("Enter your gender", "Gender", "Male")
```

- בהפעלת שורת קוד זו, Visual Basic מציגה את החלון הבא:



הפונקציה IIf

- פונקציה זו מאפשרת בדיקת תנאי, והחזרת ערך אחד מתוך שניים נתונים בהתאם לבדיקה
- פונקציה זו דומה בפעולתה לאופרטור ? בשפת C
- תחביר:

```
IIf(expr, truepart, falsepart)
```

- expr הוא הביטוי המוערך
- הערך של truepart יוחזר ע"י הפונקציה אם התוצאה של expr היא True
- הערך של falsepart יוחזר ע"י הפונקציה אם התוצאה של expr היא False
- הפונקציה מפעילה בכל מקרה את הפקודות הרשומות ב-truepart וב-falsepart

דוגמה

```
Dim nAge As Integer, strAge As String

nAge = Val(InputBox("Enter an age", "Age"))
strAge = IIf(nAge = 5, "The age is five", "The age is not five")
MsgBox strAge
```

- קטע קוד זה מציג InputBox למשתמש וקולט ממנו מספר

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- הפונקציה IIf מכניסה למחרוזת strAge את הערך המתאים לפי בדיקת התנאי nAge = 5
- לבסוף מוצגת הודעה למשתמש

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגילים

1. צור תוכנית בשם ControlStructures המציגה למשתמש בקשה להזנת מספר בין 0 ל-2 באמצעות הפונקציה InputBox, ובהתאם למספר שהוזן יוצג MsgBox למשתמש. מקם על ה-Form שלושה CommandButtons וכתוב את הנדרש בשלוש צורות:
   א. תוך שימוש ב-If...Then...Else, בתגובה ל-Click על ה-CommandButton הראשון
   ב. תוך שימוש ב-Select...Case בתגובה ל-Click על ה-CommandButton השני
   ג. תוך שימוש בפונקציה IIf בתגובה ל-Click על ה-CommandButton השלישי
2. כתוב תוכנית בשם PrintSequence. מקם על טופס TextBox אחד ו-CommandButton. בעת לחיצה על ה-CommandButton יש להציג על Label שיונח על הטופס את רצף המספרים המתחיל ב-0 עד המספר שהוזן ע"י המשתמש לתוך ה-TextBox.
3. כתוב תוכנית בשם MultipleBoard. לחיצה על CommandButton שימוקם על הטופס תדפיס על גבי Label שימוקם בטופס את לוח הכפל בממדים 10X10 בתוספת כותרות.
4. כתוב תוכנית בשם ProcessInputs הקולטת מהמשתמש מספרים בעזרת InputBox עד אשר המשתמש מזין 0 או מחרוזת ריקה. התוכנית תציג על גבי הטופס את סכום המספרים, מכפלתם (אין לכפול כמובן ב-0 האחרון), ואת ממוצע המספרים.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 5:

הפקדים הסטנדרטיים II

בשיעור זה:

- תכיר את ה-Controls הבאים:
  - CheckBox (תיבת סימון)
  - OptionButton (תיבת אפשרות)
  - Frame (מסגרת)
  - ListBox (תיבת רשימה)
  - ComboBox (תיבה משולבת)
  - Line (קו)
  - Shape (צורה)
  - Timer (שעון)
- תכיר שיטה לניהול ממשק המשתמש באמצעות Timer

CheckBox (תיבת סימון)

- CheckBox שימושי כאשר יש צורך בבחירת אחת משתי אפשרויות: "כן" או "לא", "אמת" או "שקר", "מלא" או "ריק" וכו'
- ישנה אפשרות שלישית הנקראת Grayed. היא משמשת בעיקר לתיאור מצב לא מוגדר או חלקי (תוכל לעיין למשל בתוכנית ההוספה וההסרה של פריטי Windows מתוך ה-Control Panel, שם Grayed מסמל בחירה חלקית של התת קבוצה)
- המעבר בין שני המצבים הראשונים הוא אוטומטי. כדי להפוך את התיבה ל-Grayed יש לנהל זאת מתוך הקוד

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Properties של CheckBox

- **Picture-ו DownPicture ,DisabledPicture:** מגדירים תמונות למצבים שונים של ה-CheckBox במקרה שה-Style של ה-CheckBox מוגדר כ-Graphical
- **Value:** מכיל את אחד משלושה ערכים שבאוסף הקבועים CheckBoxConstants – vbChecked, vbUnchecked ו-vbGrayed

Events של CheckBox

- **Click**: מתרחש בעת לחיצת עכבר או מקש הרווח, המשנה את ערך ה-CheckBox

OptionButton (לחצן אפשרות)

- תוכל להשתמש ב-OptionButton במסגרת קבוצת אפשרויות שרק אחת מהן יכולה להיות מסומנת בזמן נתון

Properties של OptionButton

- **Value**: מכיל את ערך ה-OptionButton שהוא True או False

Frame (מסגרת)

- באמצעות Frame תוכל לקבץ מספר Controls תחת קבוצה אחת
- Frame הוא פקד מסוג Container
- שתי תכונות עיקריות מאפיינות Containers:
  - אם משנים את מיקומו של ה-Frame כל ה-Child Controls שבתוכו זזים יחד איתו
  - כאשר ה-Frame נמצא במצב Disabled כל ה-Child Controls שבו גם הם Disabled

Properties של Frame

- **Enabled**: הוא המאפיין היותר חשוב של ה-Frame, אם ערכו שווה ל-False כל ה-Controls שבו הופכים ללא זמינים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Methods של Frame

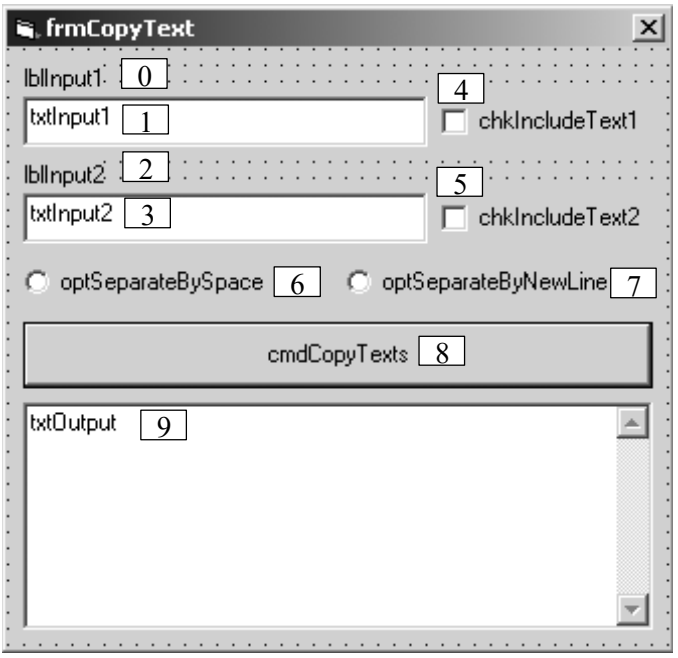
- **Move**: הוא Method חשוב ב-Frame, כיוון ששימוש נכון בו יכול לאפשר מיקום מהיר של קבוצת Controls

דוגמה 1: התוכנית CopyText

- בתוכנית זו תיצור טופס עם שני TextBoxes. לחיצה על CommandButton תעתיק את תוכני ה-TextBoxes לתיבה שלישית. ליד כל תיבת טקסט יופיע CheckBox שיקבע האם יש להעתיק את אותו הטקסט לפלט. בנוסף, יוצבו על הטופס שני OptionButtons הבוררים בין רווח (Space) לבין מעבר שורה (New line) כמפריד בין תיבות הטקסט.
- בתוכנית זו נבצע אתחול של ערכי ה-Controls השונים בשלב ה-Load של הטופס.

1. **צור פרויקט חדש מסוג Standard EXE, ושמור אותו:**
   - הקצה לפרויקט את השם CopyText ולטופס frmCopyText
   - שמור את הפרויקט
2. **הוסף Controls לטופס:**
   - הוסף את ה-Controls הבאים ומקם אותם ע"פ הציור שלפניך:
     - **שני Labels:** lblInput1 ו-lblInput2
     - **שני CheckBoxes:** chkIncludeText1 ו-chkIncludeText2
     - **שני OptionButtons:** optSeparateBySpace ו-optSeparateByNewLine
     - **CommandButton אחד:** cmdCopyTexts
     - **שלושה TextBoxes:** txtInput1, txtInput2 ו-txtOutput
   - מקם את ה-Controls כך שהטופס ייראה כמו בציור הבא
   - הקצה להם שמות כפי שמוצג בציור הבא
   - קבע את סדר מעבר הטאב כך שיהיה לפי סדר המספרים המוקפים במלבן בציור הבא

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il



3. הגדר את המאפיינים הבאים:

אובייקט	מאפיין	ערך
frmCopyText	BorderStyle	1-Fixed Single
	StartUpPosition	CenterScreen–2
cmdCopyTexts	Default	True
txtOutput	MultiLine	True

4. הוסף קוד לטופס:

- קוד זה מאתחל את הטקסט בכותרת הטופס ועל גבי ה-Controls השונים, בתגובה לאירוע Load
- בנוסף, קוד זה מאתחל את ערכי ה-Controls

```
Private Sub Form_Load()
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
    ' Initialize the form caption
    frmCopyText.Caption = "Copy Text"

    ' Initialize the labels caption
    lblInput1.Caption = "Input &1"
    lblInput2.Caption = "Input &2"

    ' Clear the text boxes
    txtInput1.Text = ""
    txtInput2.Text = ""
    txtOutput.Text = ""

    ' Initialize the check boxes
    chkIncludeText1.Caption = "Copy to output"
    chkIncludeText1.Value = vbChecked
    chkIncludeText2.Caption = "Copy to output"
    chkIncludeText2.Value = vbChecked

    ' Initialize the option buttons
    optSeparateByNewLine.Caption = "Separate by a &new-line"
    optSeparateBySpace.Caption = "Separate by a &space"
    optSeparateBySpace = True

    ' Initialize the command button
    cmdCopyTexts.Caption = "&Copy texts to output"
End Sub
```

5. הוסף קוד לטופס:

- הקוד הבא פועל בתגובה ללחיצה על הלחצן cmdCopyTexts

```
Private Sub cmdCopyTexts_Click()
    ' Copy first input text only if chkInputText1 is checked
    If chkIncludeText1.Value = vbChecked Then
        txtOutput.Text = txtOutput.Text & txtInput1.Text
    End If

    ' Copy space or new-line only if txtOutput is not empty
    If txtOutput.Text <> "" Then
        ' Determines which option button is checked and copy new-
line
        ' or space accordingly
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
      If optSeparateByNewLine.Value = True Then
         txtOutput.Text = txtOutput.Text & vbCrLf
      Else
         txtOutput.Text = txtOutput.Text & " "
      End If
   End If

   ' Copy second text only if chkInputText2 is checked
   If chkIncludeText2.Value = vbChecked Then
      txtOutput.Text = txtOutput.Text & txtInput2.Text
   End If

   ' Prepare the output text box to the next copy
   If txtOutput.Text <> "" Then
      txtOutput.Text = txtOutput.Text & vbCrLf
   End If

   ' Bring the focus back to input1
   txtInput1.SelStart = 0
   txtInput1.SelLength = Len(txtInput1.Text)
   txtInput1.SetFocus
End Sub
```

6. הרץ את התוכנית ובדוק את פעולתה

ListBox (תיבת רשימה)

- תיבת רשימה מאפשרת להציג למשתמש רשימת מחרוזות
- המשתמש יכול לגלול את רשימת הערכים כלפי מטה או מעלה
- האינדקסים של פריטי ה-ListBox נעים בין 0 ל-n-1, כאשר n הוא מספר הפריטים ברשימה.

Properties של ListBox

- **Columns**: קובע אם ה-ListBox יציג את התוכן בטור אחד עם גלילה אנכית (Columns = 0) או במספר טורים (Columns > 0) עם פס גלילה אופקי
- **IntegralHeight**: קובע אם ה-ListBox יתאים את גובהו כך שמחרוזות יוצגו בשלמות

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **ItemData**: מערך שלמים מסוג Long המאפשר לצרף מידע נוסף לפריטי ה-ListBox כך שלכל מחרוזת ניתן יהיה להצמיד ערך מספרי לשימוש פנימי כלשהו של התוכנית
- **List**: מערך מחרוזות המכיל בתוכו את תוכן ה-ListBox
- **ListCount**: מחזיר את מספר הפריטים ב-ListBox
- **ListIndex**: מחזיר את האינדקס של הפריט הנבחר (במקרה של ListBox ללא MultiSelect), או את הפריט שנמצא בפוקוס (בין אם הוא נבחר או לא) כאשר MultiSelect שווה ל1 או 2
- **MultiSelect**: קובע האם יהיה ניתן לבחור מספר פריטים ברשימה במקביל
- **NewIndex**: מחזיר את האינדקס של הפריט האחרון שנוסף לרשימה
- **SelCount**: מחזיר את מספר הפריטים הנבחרים ברשימה
- **Selected**: מאפיין זה מקבל כפרמטר אינדקס של פריט, ומחזיר ערך בוליאני הקובע האם הפריט נבחר או לא. מאפיין זה שימושי במקרה של MultiSelect. כמו כן, ניתן לקבוע בעזרת מאפיין זה האם פריט ייבחר או לא.
- **Sorted**: קובע אם האלמנטים ברשימה ימוינו באופן אוטומטי
- **Style**: קובע את סגנון ה-ListBox (רגיל או עם CheckBoxes)
- **TopIndex**: מאפשר לקבוע איזה פריט יוצג בראש הרשימה

Methods של ListBox

- **AddItem**: מאפשר להוסיף פריטים לרשימה בזמן ריצה
- **Clear**: מנקה את התוכן של ה-ListBox
- **Refresh**: מרענן את תצוגת ה-ListBox
- **RemoveItem**: מסיר פריט מהרשימה

Events של ListBox

- **ItemCheck**: מתרחש כאשר המשתמש מסמן V ב-CheckBox המוצמד לפריט רשימה (רלוונטי כאשר המאפיין Style שווה לcheckbox)
- **Scroll**: מתרחש כאשר המשתמש גולל את הרשימה או כאשר המתכנת גולל את הרשימה ע"י שימוש במאפיין TopIndex

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

ComboBox (תיבה משולבת)

- ComboBox הוא שילוב של TextBox ו-ListBox
- כל ה-Properties (מלבד Style), ה-Methods וה-Events של ComboBox Control זהים לאלו של TextBox וה-ListBox והתנהגותם זהה גם כן.
- Style הוא המאפיין היחיד ששונה ב-ComboBox

Properties של ComboBox

- **Style:** קובע את סגנון ה-ComboBox:
  - **Dropdown Combo**: כולל TextBox ורשימה נפתחת. המשתמש יכול לבחור ערך מהרשימה או להקליד את הערך בעצמו
  - **Simple Combo**: כולל TextBox ורשימה לא נפתחת. המשתמש יכול לבחור ערך מהרשימה או להקליד את הערך
  - **Dropdown List**: כולל רשימה נפתחת בלבד. המשתמש יכול לבחור ערך מהרשימה בלבד.

Line (קו) ו-Shape (צורה)

- Line ו-Shape הם פקדים המאפשרים להוסיף קווים שונים וצורות לטופס ללא שימוש בפונקציות גרפיקה לשם כך
- ל-Line ול-Shape אין Events

Properties של Line ו-Shape

- **Left, Top, Width ו-Height (ב-Shape):** מגדירים את גודל ומיקום המלבן החוסם את הצורה
- **X1, X2, Y1, Y2 (ב-Line)**: מגדירים את נקודות הקצה של הישר
- חשוב לשים לב למאפייני הצבע והרקע של Line ו-Shape כי מעבר לשימוש גרפי בצורות אלו, אין להם שימושים רבים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Timer (פקד שעון)

- Timer משמש לשיגור אירוע כל זמן נתון.
- Timer הוא מסוג הפקדים שלא נראים ב-Run Time (זמן ריצה)

Properties של Timer

- **Enabled:** קובע האם השעון פעיל וכתוצאה מכך משגר אירוע כל זמן נתון
- **Interval:** קובע את המרווח במילי-שניות בין שיגור אירוע אחד לשני

Events של Timer

- **Timer:** מתרחש כל Interval מילי-שניות (בתנאי שה-Timer נמצא במצב Enabled)

דוגמה 2 – StudentCourses

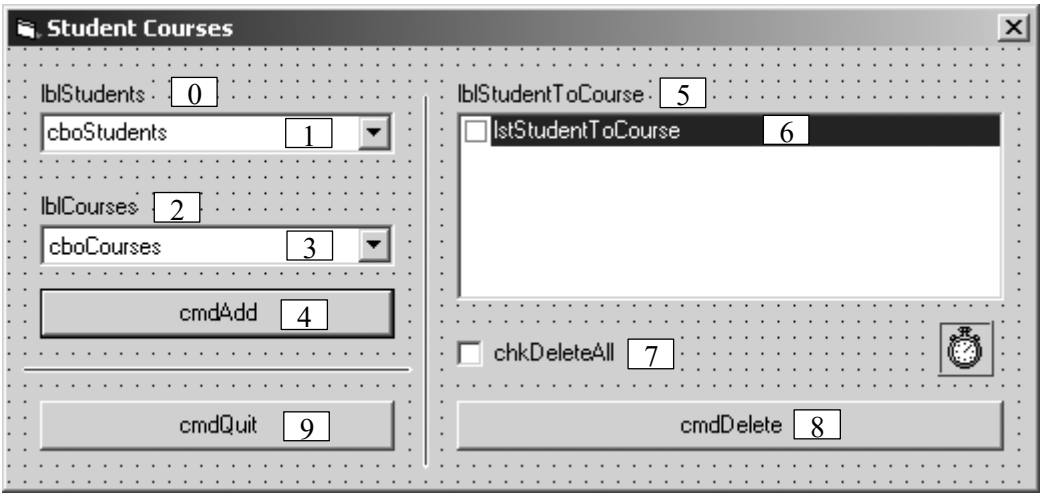
- בתוכנית זו נאפשר למשתמש לשבץ תלמידים לקורסים, ולהוסיף את השיבוצים ל-ListBox
- המשתמש יוכל גם להסיר שורות שהכניס או את כולן תוך שימוש ב-CommandButton שיוצב לשם כך
- בתוכנית זו נשתמש ב-Timer שינהל את ממשק המשתמש

1. **צור פרויקט חדש מסוג Standard EXE, ושמור אותו:**
   - הקצה לפרויקט את השם StudentCourses ולטופס frmStudentCourses
   - שמור את הפרויקט
2. **הוסף Controls לטופס:**
   - הוסף את ה-Controls הבאים ומקם אותם ע"פ הציור שלפניך:
     - **שלושה Labels:** lblStudents ,lblCourse ו-lblStudentToCourse
     - **שני ComboBoxes:** cboStudents ו-cboCourses
     - **שלושה CommandButtons:** cmdAdd ,cmdQuit ו-cmdDelete
     - **ListBox אחד:** lstStudentToCourse

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **CheckBox אחד:** chkDeleteAll
- **Timer אחד:** tmrUIUpdate

- מקם את ה-Controls כך שהטופס ייראה כמו בציור
- הקצה להם שמות כפי שמוצג בציור
- קבע את סדר מעבר הטאב כך שיהיה לפי סדר המספרים המוקפים במלבן בציור
- תוכל להשתמש בקווים כדי להפריד חלקים בטופס באופן ויזואלי



3. הגדר את המאפיינים הבאים:

אובייקט	מאפיין	ערך
frmStudentCourses	BorderStyle	1-Fixed Single
	StartUpPosition	2 - CenterScreen
lblStudents	Autosize	True
cboStudents	Style	0 - Dropdown combo
lblCourses	Autosize	True

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

cboCourses	Style	2 - Dropdown List
lblStudentToCourse	Autosize	True
lstStudentToCourse	Sorted	True
	Style	1 - Checkbox
cmdAdd	Default	True
cmdQuit	Cancel	True
tmrUIUpdate	Interval	20

4. **הוסף קוד לטופס:**

- הקוד הבא מצהיר על משתנים וקבועים:
- המשתנה m_nCheckCount נועד לספור את מספר הפריטים שיש להסיר מה-ListBox. בדיקת משתנה זה מקילה על שינוי מראה תיבת הסימון chkDeleteAll בתגובה לסימון או ביטול סימון של Check בתוך ה-ListBox
- במקרה של שינוי הסימון ב-chkDeleteAll כל הסימונים בתוך ה-lstStudentToCourse מתעדכנים בהתאם. כיוון שנוצרת תגובה מעגלית, בתוכנית זו נעשה שימוש במשתנה הגלובלי m_nCheckedSource כדי לאפשר בקוד התגובה לסימון שורה ב-lstStudentToCourse לבדוק האם מקור השינוי היה פעולת משתמש או פעולת השמה מתוך הקוד, שאז אין לבצע מאומה.
- הקבועים הופכים את התוכנית לקריאה יותר

```
Dim m_nCheckCount As Integer
Dim m_nCheckedSource As Integer
Const CHECK_SOURCE_USER As Integer = 0
Const CHECK_SOURCE_CODE As Integer = 1
```

5. **הוסף קוד לטופס:**

- הקוד הבא מאתחל את הפקדים השונים ואת המשתנים:

```
Private Sub Form_Load()
   ' Initializing lblStudets
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
    lblstudents.Caption = "&Students"
    ' Initializing cboStudents
    cboStudents.Clear
    cboStudents.AddItem "John Degweed"
    cboStudents.AddItem "Erick E"
    cboStudents.AddItem "Michel de Hey"
    cboStudents.AddItem "Paul Okenfold"
    cboStudents.AddItem "Dimmitri"
    cboStudents.AddItem "Tiesto"
    cboStudents.AddItem "Josh Wink"
    cboStudents.AddItem "Terry Francis"
    cboStudents.AddItem "Ferry Corsten"
    ' Initializing lblCourses
    lblCourses.Caption = "&Courses"
    ' Initializing cboCourses
    cboCourses.Clear
    cboCourses.AddItem "Advanced DJ Techniqs"
    cboCourses.AddItem "Using guitar loops in the Mix"
    cboCourses.AddItem "Synthisizer - Advance Topics"
    cboCourses.AddItem "Synthisizer - Fundamentals"
    cboCourses.AddItem "Extarcting Loops"
    ' Initializing cmdAdd
    cmdAdd.Caption = "&Add to list"
    ' Initializing cmdQuit
    cmdQuit.Caption = "&Quit"
    ' Initializing lblStudentsToCourses
    lblStudentToCourse.Caption = "Student to course"
    ' Initializing lstStudentToCourse
    lstStudentToCourse.Clear
    ' Initializing chkDeleteAll
    chkDeleteAll.Caption = "Delete &all rows"
    ' Initializing cmdDelete
    cmdDelete.Caption = "&Delete"
    ' Other initializations
    m_nCheckCount = 0
    m_nCheckedSource = CHECK_SOURCE_USER
End Sub
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

6. הוסף קוד לטופס:

- להלן קוד השגרות לתגובה בעת לחיצה על cmdAdd או לחיצה על cmdQuit
- לאחר הוספת השורה לרשימה, הפוקוס מוחזר ל-cboStudents להזנת ערך חדש

```
Private Sub cmdAdd_Click()
   Dim strRow As String
   strRow = cboStudents.Text & ": " & cboCourses.Text
   lstStudentToCourse.AddItem strRow
   cboStudents.SelStart = 0
   cboStudents.SelLength = Len(cboStudents.Text)
   cboStudents.SetFocus
End Sub

Private Sub cmdQuit_Click()
   Unload Me
End Sub
```

7. הוסף קוד לטופס:

- הקוד הבא מופעל בתגובה ללחיצה על cmdDelete
- אם יש להסיר את כל הפריטים מהרשימה השגרה משתמשת ב-Clear; אחרת, היא מבצעת איטרציה על פריטי הרשימה בכדי לבדוק אילו מהם מסומנים למחיקה
- בסיום, השגרה מאפסת את המונה m_nCheckCount ומבטלת את הסימון שליד chkDeleteAll

```
Private Sub cmdDelete_Click()
   Dim i As Integer
   ' Check whether to remove all items
   If chkDeleteAll.Value = vbChecked Then
      lstStudentToCourse.Clear ' Use Clear to remove all items
   Else
      ' Remove only the checked items
      i = 0
      Do While i <= (lstStudentToCourse.ListCount - 1)
         If lstStudentToCourse.Selected(i) = True Then
            lstStudentToCourse.RemoveItem (i)
         Else
            i = i + 1
         End If
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
      Loop
   End If
   ' Reset the "Delete all" - CheckBox and the
   ' Checked-Items counter (m_nCheckCount)
   chkDeleteAll.Value = vbUnchecked
   m_nCheckCount = 0
End Sub
```

8. הוסף קוד לטופס:

- הקוד הבא מופעל בתגובה לסימון שורה ברשימה lstStudentToCourse
- אם האירוע התרחש בגלל לחיצה על chkDeleteAll, אזי מתבצע דילוג על שאר הקוד כדי למנוע תגובה מעגלית
- הקוד בודק אם נוסף סימון או הוסר. בהתאם לכך הוא מעדכן את m_nCheckCount
- לאחר מכן נבדק ערכו של m_nCheckCount ובהתאם לערך זה מותאם הסימון ב-chkDeleteAll

```
Private Sub lstStudentToCourse_ItemCheck(Item As Integer)
   Dim nCheckedCount As Integer, i As Integer
   ' Process this code only if the Checked or Unchecked operation
   ' was invoked by the user and not the code inside chkDeleteAll
   If m_nCheckedSource = CHECK_SOURCE_USER Then
      m_nCheckCount = m_nCheckCount + _
         IIf(lstStudentToCourse.Selected(Item), 1, -1)
      m_nCheckedSource = CHECK_SOURCE_CODE
      If m_nCheckCount = 0 Then
         chkDeleteAll.Value = vbUnchecked
      ElseIf m_nCheckCount = lstStudentToCourse.ListCount Then
         chkDeleteAll.Value = vbChecked
      Else
         chkDeleteAll.Value = vbGrayed
      End If
      m_nCheckedSource = CHECK_SOURCE_USER
   End If
End Sub

Private Sub chkDeleteAll_Click()
   Dim i As Integer
   Dim bChecked As Boolean
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   If m_nCheckedSource = CHECK_SOURCE_CODE Then Exit Sub

   bChecked = (chkDeleteAll.Value = vbChecked)
   m_nCheckedSource = CHECK_SOURCE_CODE

   For i = 0 To lstStudentToCourse.ListCount - 1
      lstStudentToCourse.Selected(i) = bChecked
   Next i

   If bChecked Then
      m_nCheckCount = lstStudentToCourse.ListCount
   Else
      m_nCheckCount = 0
   End If

   m_nCheckedSource = CHECK_SOURCE_USER
End Sub
```

9. הוסף קוד לטופס:

- קטע הקוד הבא מופעל כל 20 מילי-שניות
- בקוד זה נבדק מצב מרכיבים שונים בטופס ובהתאם לכך מעודכן ממשק המשתמש
- הלחצן cmdAdd מאופשר רק אם קיים טקסט ב-cboStudents ו-cboCourses
- chkDeleteAll מאופשר רק אם ישנם פריטים ב-lstStudentToCourse
- הלחצן cmdDelete מאופשר רק אם m_nCheckCount גדול מ-0

```
Private Sub tmrUIUpdate_Timer()
   ' Enable the "Add to list" command only if there is text in
   ' cboStudents as well as in cboCourses
   cmdAdd.Enabled = cboCourses.Text <> "" And cboStudents.Text <>
""
   ' Enable the "Delete All" - CheckBox only if there are
   ' items in the list
   chkDeleteAll.Enabled = lstStudentToCourse.ListCount > 0
   ' Enable cmdDelete only if the number of the checked items
   ' is greater than 0
   cmdDelete.Enabled = m_nCheckCount > 0
End Sub
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

10. הפעל את התוכנית ובדוק את פעולתה:

- שים לב לעדכון ה-User Interface בתגובה לשינוי תכני הרשימות השונות
- שים לב ש-Delete מאופשר רק אם יש סימון כלשהו ברשימה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגילים

1. כתוב תוכנית בשם DayAndColor. מקם על טופס Frame ובתוכו שבעה OptionButtons עם הכותרות Sunday, Monday וכו'. כמו כן מקם על הטופס שלושה CheckBoxes עם הכותרות Red, Green ו-Blue. במקום ריק בטופס מקם Label. בעת בחירת אחד הימים ב-OptionButtons יש לשנות את הטקסט הכתוב ב-Label. בעת לחיצה על CheckBox של צבע, יש להוסיף את המרכיב שלו לצבע הטופס או להסיר מרכיב זה מצבע הטופס בהתאם לערך ה-CheckBox. כך למשל, אם צבע הטופס הוא אדום (RGB(255, 0, 0 לאחר לחיצה על ה-CheckBox הירוק יתוסף המרכיב שלו לצבע וכתוצאה מכך תקבל צבע צהוב ((RGB(255, 255, 0, לחיצה נוספת על Green תסיר את המרכיב והרקע ישוב להיות אדום.

2. כתוב תוכנית בשם FulfilListBox. בתוכנית הוסף על טופס את ה-Controls הבאים:
TextBox (עם Label מתאים ככותרת), ListBox ושלושה CommandButtons.
בעת לחיצה (Click) על ה-CommandButton הראשון יתוסף הטקסט שהוזן לתוך ה-TextBox ל-ListBox (ה-CommandButton הזה יהיה Default). אם לא קיים טקסט בתוך ה-TextBox, לא יתבצע מאומה. לאחר ההוספה יוחזר הפוקוס ל-TextBox ויסומן הקטע שבתוכו.
בעת לחיצה על ה-CommandButton השני יימחק הפריט המסומן ברשימה.
CommandButton שלישי ישמש ליציאה ויקבל את המאפיין Cancel.
כמו כן, בעת לחיצה כפולה (Double Click) על שורה בתוך ה-ListBox יוצג Message Box עם טקסט הפריט שנבחר.

3. צור תוכנית בשם SumListData.
בטופס מקם את ה-Controls הבאים:
שני TextBoxes עם Labels מתאימים שכותרת הראשון היא "Name" וכותרת השני היא "Additional Value".
שני CommandButtons שכותרת הראשון היא Insert וכותרת השני היא Sum.
ListBox אחד.
בעת לחיצה על Insert יוזן ל-ListBox המידע משני ה-TextBoxes. הערך שבתיבת

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Additional Value יישמר במאפיין ItemData של הפריט. אם המשתמש לא הזין דבר לשדה Name אזי לא תתבצע ההכנסה. אם רק השדה Additional Value ריק, ערך ברירת המחדל יהיה 1. לאחר ההוספה יוחזר הפוקוס לתיבה Name ויסומן בה הטקסט לשם החלפה.
בעת לחיצה על הלחצן Sum יחושב סכום כל מאפייני ה-ItemData הצמודים לפריטי הרשימה, ותוצג למשתמש הודעה מתאימה.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 6:

מרכיבי השפה II

בשיעור זה:

- תכיר את המילה With
- תלמד איך ליצור מערכים חד ממדיים, רב ממדיים ומערכים דינמיים
- תלמד ליצור Control Arrays (מערכים של פקדים)
- תלמד לכתוב רוטינות ופונקציות
- תכיר צורות העברה שונות של פרמטרים לרוטינה ופונקציה
- תבין מהם משתנים מקומיים, משתנים גלובליים ומשתנים סטטיים
- תבין מהי פרוצדורה סטטית
- תלמד ליצור פרוצדורה עם מספר פרמטרים משתנה
- תלמד לעבוד עם מודולי קוד (*.bas)
- תלמד כיצד להפעיל קוד מחלון ה-Immediate
- תלמד ליצור רשימות קבועים באמצעות המשפט Enum...End Enum
- תלמד להגדיר סוגי משתנים חדשים (User Defined Type)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

המילה WITH

- לעתים יש צורך להגדיר מספר גדול של מאפיינים של אובייקט
- כדי לקצר את זמן הכתיבה וכדי להפוך את הקוד מובן יותר אפשר להשתמש בתבנית הכתיבה הבאה:

```
With object
   .property1 = value1
   .property2 = value2
   ...
End With
```

ARRAYS (מערכים)

- מערך הוא רצף של משתנים מאותו הסוג הממוקמים ברצף בזיכרון
- ב-Visual Basic ניתן להצהיר על מערכים חד ממדיים, רב ממדיים, ומערכים דינמיים
- הפנייה לאיבר במערך ב-Visual Basic מתבצעת על-ידי רישום אינדקס האיבר בתוך סוגריים עגולים
- כאשר הפנייה היא לאיבר במערך רב ממדי, יש להפריד בין אינדקסים ברמות שונות באמצעות פסיקים
- הצהרה על מערך באמצעות המילים Public, Private, Dim או Static נעשית באותו אופן כמו הצהרה על משתנה רגיל.

One-Dimensional Arrays (מערכים חד ממדיים)

- הצהרה על מערך חד ממדי יכולה להתבצע בשתי צורות. הצורה הראשונה היא:

```
Dim arr_name(upper_index) As Type
```

- arr_name הוא שם המערך
- upper_index הוא האינדקס של האיבר האחרון במערך
- Type הוא סוג האיברים במערך
- הצורה השנייה היא:

```
Dim arr_name(lower_index To upper_index) As Type
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- השוני בצורה זו הוא בכך שאתה מגדיר את האינדקס התחתון ואת האינדקס העליון שיהוו את גבולות המערך
- מותר להשתמש גם במספרים שליליים עבור האינדקסים

המשפט Option Base

- אם לא מצוין האינדקס התחתון של המערך, ברירת המחדל היא 0
- כדי לשנות זאת ניתן להשתמש במשפט Option Base 1 בתחילת מודול הקוד, במקרה זה, האינדקס התחתון של מערכים שיוצהרו בצורה הראשונה יהיה 1
- ניתן להשתמש גם במשפט Option Base 0, אך זוהי ברירת המחדל

Multi-Dimensional Arrays (מערכים רב ממדיים)

- ההצהרה על מערכים רב ממדיים דומה למערכים חד ממדיים, אלא שיש להפריד בפסיקים בין הגדרת טווחי האינדקסים של כל ממד, באופן הבא:

```
Dim arr_name(lower_index1 To upper_index1, lower_index2 To
upper_index2,...) As Type
```

- טווחי האינדקסים של ממד במערך יכולים להיות מוצהרים כמו בצורה הראשונה שהוצגה עבור מערכים חד ממדיים. במקרה זה, האינדקס התחתון יהיה 0 אלא אם הוגדר Option Base 1 בתחילת הקובץ

Dynamic Arrays (מערכים דינמיים)

- כדי להצהיר על מערך דינמי, יש להצהיר עליו מלכתחילה ללא גודל מוגדר, באופן הבא:

```
Dim arr_name() As Type
```

- באמצעות המילה ReDim ניתן לשנות את גודל המערך במהלך ריצת התוכנית, באופן הבא:

```
ReDim [Preserve] (lower_index1 To upper_index1, ...) [As Type]
```

- ניתן לשנות את מספר הממדים במערך, את גודל כל ממד
- את סוג האיברים במערך ניתן לשנות רק אם המערך מאוחסן בתוך משתנה מסוג Variant
- אם משתמשים במילה Preserve גודלו משתנה תוך שמירת תוכנו הקודם

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בעת שימוש במילה Preserve לא ניתן לשנות את סוג האיברים במערך, כמו גם את גבולו התחתון של המערך
- בעת שימוש במילה Preserve בעת שינוי גודלו של מערך רב ממדי, ניתן לשנות רק את הגבול העליון של הממד האחרון במערך

אזהרה: השימוש במילה ReDim יכול לשמש גם כהצהרה על משתנה. משמעות הדבר היא שגם אם כתוב בראש המודול המשפט Option Explicit, והשתמשת ב-ReDim יחד עם שם משתנה שלא הוצהר בראש המודול, לא תקבל שגיאת קומפילציה ואותו מערך אכן ייוצר באופן מקומי.

בדיקת גודל מערך בזמן ריצה באמצעות LBound ו-UBound

- מתעורר צורך לעתים קרובות לדעת מהו הגבול התחתון (האינדקס התחתון) של מערך, כמו גם גבולו העליון. צורך זה הופך לקריטי כשעוסקים בכתיבת קוד גנרי, כמו גם תוך עבודה עם מערכים דינמיים
- הפונקציה LBound (Lower Bound) מחזירה את האינדקס התחתון של מערך
- הפונקציה UBound (Upper Bound) מחזירה את האינדקס העליון של המערך

דוגמה 1

```
Dim m_nArr(9) As Integer
```

- פקודה זו מצהירה על מערך שהאינדקס הראשון שלו הוא 0 והאחרון הוא 9. סה"כ 10 איברים

דוגמה 2

```
Option Base 1
...
...
Dim m_nArr(9) As Integer
```

- שורות קוד אלו מצהירות על מערך שהאינדקס הראשון שלו הוא 1 והאחרון הוא 9. סה"כ 9 איברים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שים לב: המשפט Option Base יכול להופיע רק פעם אחת בתחילת המודול (לפני כל הפונקציות או ההצהרות על מערכים)

- משפט Option Base משפיע רק על מערכים שנוצרו בתוך המודול בו נרשם המשפט

דוגמה 3

```
Dim TicTacToe(1 To 3, 1 To 3) As Boolean
```

- בשורת קוד זו מוצהר מערך דו ממדי בשם TicTacToe, שהאינדקסים בשני הממדים שלו נעים בין 1 ל-3 (כולל)

דוגמה 4

- בדוגמה זו הנח ששורות הקוד מתבצעות בסדר כרונולוגי מהמשפט הראשון לאחרון.

```
Dim dyn_arr() As Integer
```

- בשורת קוד זו מוצהר מערך דינמי בשם dyn_arr מסוג Integer

```
ReDim dyn_arr(5 To 9)
```

- בשורת קוד זו קובעים מחדש את ממדי המערך dyn_arr, כך שיכיל ממד אחד שהאינדקסים שלו נעים בין 5 ל-9

```
ReDim dyn_arr(3 To 10)
```

- שורת קוד זו משנה שוב את ממדי המערך כך שיכיל ממד אחד שהאינדקסים שבו נעים בין 3 ל-10

```
ReDim Preserve dyn_arr(3 To 12)
```

- שורת קוד זו מגדירה שוב את ממדי המערך תוך שמירה על הערכים המאוחסנים בו. בשורה זו משתנה האינדקס ל-12
- ניסיון להריץ את שורת הקוד הבאה יגרום לשגיאה:

```
ReDim Preserve dyn_arr(-2 To 20)
```

- שורת קוד זו מנסה (אך לא יכולה) להשתמש במילה Preserve תוך שינוי האינדקס התחתון של המערך

```
ReDim dyn_arr(-2 To 7, 11)
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בשורת קוד זו משנים שוב את ממדי המערך כך שיכיל שני ממדים, כאשר האינדקסים בממד הראשון נעים בין –2 ל-7, והאינדקסים בממד השני נעים בין 0 (או 1 אם הוגדר Option Base 1 בראש המודול) ל-11
- ניסיון להריץ את שורת הקוד הבאה ייתקל בשגיאה:

```
ReDim dyn_arr(-2 To 7, 13) As Long
```

- לא ניתן לשנות סוג של מערך

דוגמה 5

- בדוגמה זו תראה כיצד ניתן לשנות את סוג המערך
- תחילה יש להצהיר על משתנה רגיל מסוג Variant:

```
Dim dyn_arr as Variant
```

- שים לב שזהו משתנה רגיל (אין סוגריים ריקים בהצהרה)
- לאחר מכן תוכל לשנות את גודל המערך ואת סוגו כל אימת שתרצה בכך:

```
ReDim dyn_arr(3 To 5, 6 To 7) As Long
```

- לא תוכל לשנות את סוג המערך אם השתמשת במילה Preserve

Control Array (מערכי פקדים)

- כפי שניתן ליצור מערכי משתנים מסוגים בסיסיים וסוגים המוגדרים ע"י המשתמש, כך ניתן ליצור Control Arrays (מערכי פקדים)
- לפקדים שניתן לשלב ב-Control Array קיים המאפיין Index
- ב-Control Array כל הפקדים חייבים להיות מאותו הסוג ובעלי אותו שם
- האינדקס של כל פקד במערך נקבע באמצעות המאפיין Index

גישה לפקד בודד מתוך Control Array

- כדי לגשת לפקד מסוים במערך יש לציין את האינדקס שלו באופן הבא:

```
ControlArrayName(desire_index).property = ...
```

או

```
ControlArrayName.Item(desire_index).property = ...
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

בדיקת גודל המערך וגבולותיו

- ניתן לקבוע את גודל ה-Control Array באמצעות הפונקציה Count, באופן הבא:

```
ControlArrayName.Count
```

- ניתן לקבוע את האינדקס התחתון ב-Control Array באמצעות הפונקציה LBound, באופן הבא:

```
ControlArrayName.LBound...
```

- ניתן לקבוע את גבולו העליון של ה-Control Array באמצעות הפונקציה UBound, באופן הבא:

```
ControlArrayName.UBound...
```

אירועים של פקדים השייכים ל-Control Array

- הפרמטר הראשון בשגרות אירוע של Control Array הוא האינדקס של הפקד במערך עבורו שוגרה ההודעה
- מבנה כללי של שגרת אירוע עבור ControlArray הוא כדלקמן:

```
Private Sub ControlArrayName_EventName(Index As Integer, .....all
other parameters in the event...)
```

שלבי יצירת Control Array

- ליצירת Control Array ישנן כמה דרכים. אתאר כאן שתיים:
- הדרך הראשונה והפשוטה היא ליצור פקד אחד, להגדיר לו את השם והמאפיינים הנדרשים ולאחר מכן להעתיק אותו (Ctrl+C) ולהדביק אותו (Ctrl+V). Visual Basic תשאל אותך אם אתה מעוניין ליצור מערך פקדים. אשר את הבקשה, והמערך ייווצר. דרך שנייה היא להגדיר באופן ידני את שמות הפקדים ואת האינדקס שלהם כך ששמות הפקדים יהיו זהים, ולכל אחד מהם יוקצה אינדקס ייחודי

יצירת פקדים בזמן ריצה

- כדי ליצור פקדים בזמן ריצה עליך ליצור Control Array שיש בו לפחות איבר אחד בזמן עיצוב התוכנית

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בזמן ריצת התוכנית תוכל להשתמש במתודה Load כדי לטעון פקדים נוספים באופן הבא:

```
Load arr_name(new_index)
```

- תקבל שגיאה אם תנסה ליצור פקד חדש עם אינדקס שכבר תפוס ע"י פקד אחר באותו המערך
- הקפד להגדיר את מאפייני הפקד מיד לאחר שנוצר
- כדי להסיר פקד בזמן ריצה, השתמש במתודה Unload באופן הבא:

```
Unload arr_name(index_to_remove)
```

- לא תוכל להסיר פקדים שנוצרו בזמן עיצוב, אלא רק פקדים שנוצרו באמצעות קריאה קודמת ל-Load

תרגיל 1 – התוכנית CommandBoard

- בדוגמה זו אמחיש לך כיצד ניתן ליצור פקדים בזמן ריצה
- בתוכנית זו נטען פקדים ב-Runtime ונמקם אותם על גבי הטופס הראשי של התוכנית
- לחיצה על אחד הפקדים תציג הודעה עם מספר הפקד, ותסיר אותו מהטופס (אלא אם מדובר בפקד הראשון, כלומר זה שמוקם על הטופס ב-Design time)

1. **צור תוכנית Standard EXE:**
   - הקצה לפרויקט את השם CommandBoard, ולטופס את השם frmCommandBoard
   - שמור את התוכנית
2. **הוסף פקד לטופס:**
   - הצב פקד מסוג CommandButton על גבי הטופס והקצה לו את השם cmdBoard

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

3. הגדר את המאפיינים הבאים:

אובייקט	מאפיין	ערך
frmCommandBoard	BorderStyle	1-Fixed Single
	StartUpPosition	2 - CenterScreen
	ScaleMode	0 - User
	ScaleLeft	0
	ScaleTop	0
	ScaleWidth	3
	ScaleHeight	3
cmdBoard	Index	0

4. הוסף קוד לטופס:

- לאירוע Load של הטופס, נוסיף פעולות אתחול ויצירה של ה-Control Array:

```
Private Sub Form_Load()
   Dim i As Integer

   ' Initialize the caption
   Me.Caption = "Command Board"

   ' Reposition the first control
   With cmdBoard(0)
      .Caption = "&0"
      .Left = 0
      .Top = 0
      .Width = 1
      .Height = 1
      .Visible = True
   End With
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
    ' Load the controls from 2 to 9 (th first is already on the
form)
    ' Then, reposition the loaded controls
    For i = 1 To 8
        Load cmdBoard(i)
        With cmdBoard(i)
            .Caption = "&" & Trim(Str(i))
            .Left = i Mod 3
            .Top = i \ 3
            .Width = 1
            .Height = 1
            .Visible = True
        End With
    Next i
End Sub
```

שים לב: לאחר טעינת כל פקד השתמשנו ב-With כדי להגדיר קבוצת מאפיינים שלו. צורת רישום כזו נוחה ומובנת

- יש להגדיר את המאפיין Visible ל-True כדי שהפקד הנטען יוצג על פני הטופס
- הכותרות של הפקדים הוגדרו לפי האינדקס שלהם לצורך הנוחות
- כשרוצים להכניס קו תחתון מתחת למספר (מאיץ מקלדת באמצעות ALT), משתמשים בתו המוכר & (אמפרסנד)
- הפונקציה Str המירה מספר למחרוזת מוסיפה רווח בתחילת המחרוזת כאשר המספר הוא חיובי, וסימן – (מינוס) כאשר המספר הוא שלילי. כדי להסיר רווח זה ניתן להשתמש בפונקציה Trim

5. הוסף קוד לטופס:

- הקוד הבא פועל בתגובה לאירוע Click של הפקדים במערך
- אם הפקד שנלחץ אינו הראשון (אינדקס 0 שאותו אסור להסיר באמצעות Unload), מוסר הפקד באמצעות Unload:

```
Private Sub cmdBoard_Click(Index As Integer)
    ' Remove the Command only if it isn't the first (index = 0)
command
    If Index <> 0 Then Unload cmdBoard(Index)
End Sub
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

6. **הרץ ובדוק את פעולת התוכנית**

פרוצדורות ב-Visual Basic

- כמו בכל שפת תכנות, גם ב-Visual Basic ניתן לכתוב פרוצדורות קוד
- פרוצדורות ב-Visual Basic מגיעות אלינו בארבעה טעמים:
  - Sub Routines – רוטינות
  - Functions - פונקציות
  - Properties – מאפיינים
  - Events - אירועים
- את השניים הראשונים תכיר בשיעור זה. יצירת מאפיינים ואירועים תכיר בשיעור העוסק במחלקות

רוטינות ופונקציות שלב I

- רוטינה ב-Visual Basic אינה מחזירה ערך, לעומת פונקציה המחזירה ערך
- לרוטינות ופונקציות ניתן להעביר פרמטרים בשתי צורות:
  - **By Value:** על ידי ערך
  - **By Reference:** על ידי ייחוס
- קיימים היבטים נוספים הנוגעים להעברת פרמטרים, נכיר אותם בהמשך
- העברת פרמטר By Value יוצרת העתק למשתנה והעתק זה נמצא בידי הרוטינה או הפונקציה
- העברת פרמטר By Reference גורמת למשתנה המקורי להיות נגיש על ידי הרוטינה או הפונקציה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תחביר רוטינות ופונקציות

- תחביר רוטינה הוא:

```
Public Sub SubName(parameters)
   ...
End Sub
```

- תחביר פונקציה הוא:

```
Public Function FunctionName(parameters) As Type
   ...
   FunctionName = ReturnValue
End Function
```

- SubName הוא שם הרוטינה ו-FunctionName הוא שם הפונקציה
- parameters הוא רשימת הפרמטרים המועברים לרוטינה או לפונקציה. בהמשך נראה כיצד מעבירים פרמטרים.
- Public הופך את הרוטינה או הפונקציה לציבורית, כלומר נגישה על ידי קוד ממודולי קוד אחרים בפרויקט
- ניתן להשתמש במילה Private במקום המילה Public, ואז הרוטינה או הפונקציה הופכת לפרטית באותו מודול קוד, ולא תהיה אליה גישה ממודולים אחרים
- Public היא הרשאת ברירת המחדל עבור פרוצדורות קוד
- Type הוא סוג הערך המוחזר על ידי הפונקציה. אם לא צוין במפורש, הסוג הוא Variant
- ReturnValue – כשרוצים להחזיר ערך מפונקציה יש לבצע השמה לשם הפונקציה

רשימת פרמטרים

- רישום הפרמטרים שהפרוצדורה מקבלת מתבצע באופן הבא:

```
[ByVal | ByRef] parameter1 As Type, [ByVal | ByRef] parameter2 As Type...
```

- parameter1 ו-parameter2 הם שמות הפרמטרים
- Type הוא סוג הערך של הפרמטר. אם לא צוין במפורש, סוג הפרמטר הוא Variant

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ByVal או ByRef קובעים את האופן שבו יועבר הפרמטר לפונקציה. האפשרויות הן:
  - **ByVal:** על ידי ערך
  - **ByRef:** על ידי ייחוס
- אם לא צוין במפורש, ברירת המחדל היא ByRef
- מערכים ניתן להעביר ByRef בלבד

יציאה מרוטינה או פונקציה

- ניתן לצאת בכל שלב מתוך רוטינה או פונקציה על ידי שימוש במשפטים:
  - **Exit Sub:** ליציאה מרוטינה
  - **Exit Function:** ליציאה מפונקציה

משתנים מקומיים, גלובליים וסטטיים

- ניתן להצהיר על משתנים בתוך פרוצדורות קוד באמצעות המילים Dim או Static בלבד
- הצהרה על משתנים בתוך פרוצדורה אינה יכולה להתבצע באמצעות המילים Public או Private
- משתנים המוצהרים בתוך פרוצדורת קוד נקראים משתנים מקומיים
- משתנים מקומיים זמינים אך ורק לקטע הקוד התחום בתוך הרוטינה או הפונקציה
- משתנים מקומיים מתחילים את חייהם עם תחילת הפרוצדורה ומשוחררים מהזיכרון עם סיום הפרוצדורה
- משתנים המוצהרים בראש מודול נקראים משתנים גלובליים
- משתנים גלובליים זמינים לכל הרוטינות והפרוצדורות באותו מודול
- משתנים גלובליים מתחילים את חייהם בתחילת חייו של מודול הקוד ומסיימים את חייהם בסיום חייו של מודול הקוד (במקרה של מודול *.bas מדובר בתחילת התוכנית ובסיומה. מודול *.bas יילמד בהמשך)
- משתנים המוצהרים בפרוצדורה באמצעות המילה Static נקראים משתנים סטטיים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- משתנים סטטיים נשמרים בזיכרון גם לאחר שהפרוצדורה מסיימת את פעולתה, וערכם נשמר עד להפעלה הבאה של הפרוצדורה. במילים אחרות, משתנים סטטיים שומרים על ערכם בין שתי קריאות סמוכות לפרוצדורה.

דוגמה 1

```
Public Sub Hello()
   MsgBox "Hello, world!"
End Sub
```

- פרוצדורה זו מציגה Message Box עם ההודעה "Hello, world!"
- שים לב שהיה ניתן להימנע מהשימוש במילה Public כיוון שזוהי ברירת המחדל

דוגמה 2

```
Public Sub DispSum(ByVal a As Integer, ByVal b As Integer)
   MsgBox Str(a + b)
End Sub
```

- רוטינה זו מציגה ב-Message Box את סכומם של שני מספרים המועברים אליה

דוגמה 3

```
Public Sub Swap(ByRef var1 As Variant, ByRef var2 As Variant)
   Dim varTmp As Variant
   varTmp = var1
   var1 = var2
   var2 = varTmp
End Sub
```

- הרוטינה Swap מחליפה בין התכנים של שני משתנים המועברים אליה
- כיוון שהפרמטרים מוצהרים כ-Variant, ניתן להפעיל רוטינה זו על כל הסוגים הבסיסיים של Visual Basic
- שים לב שניתן היה להימנע מהשימוש במילה ByRef כיוון שזוהי ברירת המחדל

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה 4

```
Public Function GiveMe5() As Integer
   GiveMe5 = 5
End Function
```

- פונקציה זו מחזירה 5 למי שקרא לה (Caller)

דוגמה 5

```
Public Function Factorial(ByVal n As Byte) As Long
   Dim lRes As Long
   lRes = 1
   Do While n > 0
      lRes = lRes * n
      n = n - 1
   Loop
   Factorial = lRes
End Function
```

- פונקציה זו מחשבת את העצרת של מספר המועבר אליה
- שים לב לשימוש בפרמטר n (הפרמטר המועבר לפונקציה). שינוי הפרמטר במהלך הפונקציה לא משפיע על המקור, כיוון שהפרמטר מועבר ByVal, כלומר בפונקציה נעשה שימוש בהעתק של המשתנה המועבר.

דוגמה 6

```
Public Function FindBigger(ByRef arr() As Integer) As Integer
   Dim l As Integer, u As Integer
   Dim bigger As Integer

   l = LBound(arr)
   u = UBound(arr)

   bigger = l

   Do While l <= u
      If arr(l) > arr(bigger) Then bigger = l
      l = l + 1
   Loop
   FindBigger = bigger
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
End Function
```

- פונקציה זו מקבלת מערך של Integers ומחזירה את האינדקס של האיבר הגדול במערך
- הפונקציה סורקת את המערך החל מאינדקס האיבר הראשון (LBound) עד לאינדקס האיבר האחרון (UBound)

דוגמה 7

```
Public Function SumCalls() As Integer
   Static nCalls As Integer
   nCalls = nCalls + 1
   SumCalls = nCalls
End Function
```

- בדוגמה זו SumCalls היא פונקציה המחזירה את מספר הקריאות שהתבצעו אליה עד כולל הקריאה האחרונה
- המשתנה nCalls מוצהר כסטטי, לכן הוא מאותחל ל-0 פעם אחת (בקריאה הראשונה), ובכל קריאה נוספת הוא שומר על ערכו הקודם

קריאות לרוטינות ופרוצדורות

```
[Call] name [argumentlist]
```

- ניתן לקרוא לרוטינות או פונקציות באמצעות המילה Call
- השימוש במילה Call הוא אופציונלי, ולרוב לא משתמשים בו
- אם נעשה שימוש במילה Call, יש לתחום בתוך סוגריים ארגומנטים המועברים לרוטינה או לפונקציה
- כאשר קוראים לרוטינה ללא שימוש במילה Call, אין לתחום בסוגריים את הארגומנטים
- אם משתמשים בערך המוחזר של פונקציה, יש לתחום בסוגריים את הארגומנטים המועברים לפונקציה

דוגמאות קריאה לפרוצדורות

```
Call Hello
```

- בשורת קוד זו נקראת הרוטינה Hello

```
Call Swap(a, b)
```

- בשורת קוד זו מתבצעת קריאה לרוטינה Swap, ומועברים אליה הפרמטרים a ו-b

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- באופן דומה ניתן היה לרשום:

```
Swap a, b
res = Factorial(8)
```

- בשורת קוד זו, מתבצעת קריאה לפונקציה Factorial, והערך המוחזר ממנה מושם למשתנה res

רוטינות ופרוצדורות שלב II

פרוצדורות סטטיות

- ניתן להפוך פרוצדורה לסטטית על ידי הוספת המילה Static לאחר המילה Public או Private ולפני שם הפרוצדורה, כלומר:

```
[Public | Private] Static ProcedureName(arglist) [As Type]
```

- פרוצדורה סטטית היא פרוצדורה שכל המשתנים שבה שומרים על ערכם בין הקריאות השונות לפרוצדורה

named-arguments

- בעת קריאה לפונקציה ניתן לציין לאיזה פרמטר פורמלי מיועד כל ארגומנט אקטואלי שמעבירים, תוך שימוש באופרטור := (נקודתיים ושווה)
- כך לדוגמה, אם מוצהרת הפונקציה הבאה:

```
Public Function Div(ByVal Num1 As Double, ByVal Num2 As Double)
As Double
```

- ונרצה לחשב את הביטוי $8 \div 2$, 3.1^2 נוכל לקרוא לפונקציה בשתי צורות:

```
res = Div(8, 2)
```

או

```
res = Div(Num2 := 2, Num1 := 8)
```

ParamArray

- ParamArray דומה לאופרטור ... (שלוש נקודות) בשפת C, בהקשר של פרמטרים לפונקציות

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ParamArray מאפשר להעביר מספר משתנה של פרמטרים לפרוצדורה
- ParamArray חייב להופיע בסוף רשימת הפרמטרים
- לפרוצדורה שהפרמטר האחרון שלה הוא מסוג ParamArray, ניתן לקרוא עם מספר משתנה של פרמטרים (חוץ מהפרמטרים שהם חובה כמובן)
- בתוך הפרוצדורה, הגישה לפרמטרים המועברים נעשית בצורה שבה ניגשים לאיברי מערך
- אי אפשר להשתמש במילים ByVal או ByRef יחד עם ParamArray
- ברשימת הפרמטרים לא יכולים להיות יחד ParamArray ופרמטרים אופציונליים (נלמד בסעיף הבא)
- תחביר השימוש ב-ParamArray הוא כדלקמן:

```
.....(arg1 As Type, arg2 As Type, ..., ParamArray
other_parameters ()).....
```

- מילת המפתח היא ParamArray
- other_parameters() זהו שם המערך (במקרה זה) כפי שיוכר בפרוצדורה
- בהמשך תראה דוגמה לשימוש ב-ParamArray

פרמטרים אופציונליים

- ניתן להגדיר פרמטרים מסוימים של פרוצדורה כאופציונליים, כלומר פרמטרים שאין חובה להעביר לפרוצדורה
- ניתן להשתמש בפונקציה IsMissing כדי לבחון אם פרמטר (מסוג Variant בלבד) הועבר לפונקציה
- ניתן להגדיר ערכי ברירת מחדל עבור פרמטרים אופציונליים
- כאשר משתמשים בפרמטר אופציונלי כל הפרמטרים שאחריו חייבים גם הם להיות אופציונליים
- התחביר לפרמטרים אופציונליים הוא כדלקמן:

```
...(...,Optional [ByVal | ByRef] arg As Type [= DefaultValue],...
```

- מילת המפתח היא Optional
- ניתן להמשיך ולהשתמש ב-ByRef או ByVal
- Type הוא סוג הפרמטר שלו מצפה הפרוצדורה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- DefaultValue הוא חלק רשות המגדיר את ערך הפרמטר במקרה שלא הועבר לפרוצדורה

יצירת שלד פרוצדורה באמצעות Add Procedure מהתפריט Tools

- ניתן ליצור תבנית פרוצדורות מכל הטעמים (רוטינות, פונקציות, מאפיינים ואירועים) באמצעות ה-Wizard: Add Procedure...
- לשם פתיחת ה-Wizard, עבור לחלון הקוד של אחד המודולים בתוכנית, ולאחר מכן בחר ב-Add Procedure... מתוך התפריט Tools.
- אפשרויות הבחירה הן: Sub, Function, Property ו-Event
- אפשרויות הגישה הן Public או Private
- אם תסמן את האפשרות All Local variables as Static ה-Wizard ייצור עבורך פרוצדורה סטטית

דוגמה 8

```
Public Function Mult(ByVal Num1 As Double, _
                     ByVal Num2 As Double, _
                     ParamArray AdditionalNums()) As Double
   Dim res As Double
   Dim l As Integer, u As Integer
   l = LBound(AdditionalNums)
   u = UBound(AdditionalNums)
   res = Num1 * Num2
   Do While l <= u
      res = res * AdditionalNums(l)
      l = l + 1
   Loop
   Mult = res
End Function
```

- לפונקציה זו ישנם שני פרמטרים שהם חובה: Num1 ו-Num2
- ניתן להעביר פרמטרים נוספים לפונקציה ואז תחושב המכפלה של כל המספרים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה 9

```
Public Function Div(Optional ByVal Num1 As Double = 1#, _
                    Optional ByVal Num2 As Double = 1#) As
Double
   If Num2 = 0# Then Exit Function ' Num2 must be non-zero
   Div = Num1 / Num2
End Function
```

- פונקציה זו מחזירה את מנת החלוקה של Num1 ב-Num2
- אם לא מועבר אחד מהמספרים, ערך ברירת המחדל שלו יהיה 1.0

מודולי קוד סטנדרטיים *.BAS

- מודולי הקוד ב-Visual Basic מתחלקים לשתי קבוצות עיקריות:
  - Class Modules
  - Standard Modules
- מודולי מחלקה (Class Modules) יילמדו בהמשך, טפסים לדוגמה הם סוג של מודול מחלקה
- מודול סטנדרטי יכול להכיל פונקציות ומשתנים גלובליים שיהיו זמינים לכל האובייקטים ביישום
- ניתן לכלול משתנים ופונקציות שיהיו זמינים לפונקציות שבמודול בלבד על ידי שימוש במילה Private
- כדי להוסיף מודול יש לבחור Add Module מתוך התפריט Project

הפעלת פונקציות מחלון ה-Immediate

- חלון ה-Immediate הוא אחד ממרכיבי מערכת ניפוי השגיאות (Debugger) של Visual Basic (נושא יילמד בשיעור הבא)
- חלון ה-Immediate, כשמו, מאפשר לבצע פעולות באופן מידי
- ניתן להריץ פונקציות ורוטינות באמצעות חלון ה-Immediate מבלי להריץ את התוכנית כולה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- כמובן שגם במהלך התוכנית ניתן לבחון ערכי משתנים ופונקציות בתוכנית, אך נושא זה יידון בפירוט בפרק הבא
- כדי להציג ערך המוחזר על ידי פונקציה בחלון ה-Immediate, יש להקדים את הסימן? (סימן שאלה) – כמובן שיש לתחום בסוגריים את הארגומנטים המועברים לפונקציה
- כדי להציג את חלון ה-Immediate ניתן לבחור Immediate window מהתפריט View, או לחילופין להפעיל את צירוף המקשים Ctrl+G

תרגיל 2 – הפעלת קוד מחלון ה-Immediate

- בתרגיל זה תכתוב במודול סטנדרטי קוד המוצג בדוגמאות הקודמות, ותריץ אותו דרך חלון ה-Immediate

1. **צור תוכנית Standard EXE:**
   - הסר מהתוכנית את הטופס:
     - לחץ לחיצה ימנית על שם הטופס ב-Project Explorer (אם אינך רואה את ה-Project Explorer בחר אותו מהתפריט View)
     - מהתפריט המוקפץ בחר ב-Remove Form1
   - הוסף מודול לפרויקט:
     - בחר באפשרות Add Module מהתפריט Project
     - בחר ב-Module ולחץ על Open
   - הקצה לפרויקט את השם ModuleAndImmediate ו-למודול הקצה את השם modProcedures
   - שמור את הפרויקט
2. **הצג את חלון ה-Immediate:**
   - ודא שחלון ה-Immediate מוצג. אם אינך רואה אותו, לחץ על Ctrl+G

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

3. **בחן את הרוטינה MsgBox מתוך חלון ה-Immediate:**

- הקלד בחלון ה-Immediate את הקוד הבא:

```
MsgBox "Hello, world!"
```

- ברגע שתלחץ על Enter חלון ה-Message Box יופיע
- נסה לשנות את הפרמטרים האחרים ברוטינה ובחן את השפעתם על אופן הצגת ה-Message Box

4. **בחן את הפונקציה Left של Visual Basic:**

- הקלד את שורת הקוד הבאה בחלון ה-Immediate:

```
?Left ("Hello, world!",4)
```

- שים לב לסימן השאלה בתחילת השורה, ולסוגריים המכילים את הארגומנטים
- הארגומנטים מטרתם להנחות את Visual Basic שברצוננו להשתמש בערך המוחזר מהפונקציה
- סימן השאלה מטרתו להנחות את Visual Basic להציג את הערך המוחזר בחלון ה-Immediate

5. **הוסף פונקציה ל-Module:**

- כתוב את הפונקציה Divide המוצגת להלן, בחלון ה-Module:

```
Public Function Divide(ByVal a As Double, _
                       Optional ByVal b As Double = 2#) As
Double
   Divide = a / b
End Function
```

6. **בחן את הפונקציה מחלון ה-Immediate:**

- הקלד את שורת הקוד הבאה בחלון ה-Immediate:

```
?Divide(3,6)
```

- אתה צריך לקבל את התוצאה 0.5 (האפס לא מוצג)
- הקלד את שורת הקוד הבאה בחלון ה-Immediate:

```
?Divide(b:=3, a:=6)
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בשורת קוד זו השתמשת ב-named arguments
- אתה צריך לקבל 2
- הקלד את שורת הקוד הבאה בחלון ה-Immediate:

```
?Divide(8)
```

- במקרה זה, חסר הארגומנט השני, לפיכך הפונקציה משתמשת בערך ברירת המחדל שקבעת לה
- אתה צריך לקבל 4
- הקלד את שורת הקוד הבאה בחלון ה-Immediate:

```
?Divide(7, 0)
```

- אתה אמור לקבל שגיאת חלוקה ב-0 על ידי סביבת הפיתוח

Enumerations (אוספי קבועים)

- בדומה למילת המפתח enum בשפת C, קיימת המילה Enum ב-Visual Basic
- Enum מאפשרת ליצור אוסף קבועים המאוגדים תחת שם אחד באופן הבא:

```
[Public | Private] Enum name
   membername [= constantexpression]
   membername [= constantexpression]
   ...
End Enum
```

- Public או Private קובעים לאילו חלקי קוד בתוכנית ה-Enum יהיה מוכר
- name הנו השם שבחרת ל-Enum
- membername הנו שם קבוע בתוך ה-Enum
- constantexpression הנו ערך קבוע שמוקצה עבור ה-membername
- אם לא מגדירים ערך קבוע עבור פריט בתוך Enum אז ערכו נקבע לפי הכלל הבא:
  - אם זהו הפריט הראשון ב-Enum ערכו יהיה 0
  - עבור כל פריט (שאינו הפריט הראשון) ערכו יהיה גדול ב-1 מערך הפריט הקודם
- לא ניתן לשנות את ערכו של פריט ב-Enum במהלך התוכנית

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- לאחר שהוגדר Enum, ניתן להשתמש בו לשם הצהרת משתנים, פרמטרים של פרוצדורות וערכים מוחזרים של פונקציות

דוגמאות

```
Public Enum AgeCategoriesConstants
   acChild
   acYoung
   acAdult
   acOld
End Enum
```

- ב-Enum זה מוצהרים ארבעה קבועים
- ערכו של acChild הוא 0, של acYoung הוא 1 וכן הלאה
- בהנחה שרוצים להציב את ערך הקבוע acChild במשתנה nAge, אפשר לבצע זאת בשתי צורות:

```
nAge = acChild
```

או

```
nAge = AgeCategoriesConstants.acChild
```

- הצורה השנייה נוחה כאשר יש התנגשות שמות עם קבוע אחר

User-Defined Types (סוגי נתונים מוגדרים על ידי המשתמש)

- ניתן ליצור סוגי נתונים מורכבים (כמו Struct בשפת C) באמצעות המילה Type
- התחביר הוא כדלקמן:

```
[Public | Private] Type varname
   elementname [(subscript)] As type
   elementname [(subscript)] As type
   ...
End Type
```

- Public או Private מגדירים את טווח ההכרה (Scope) של הסוג החדש
- varname הוא שם הסוג החדש שמוגדר על ידי Type
- elementname הוא שם אלמנט בתוך הסוג החדש

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- אלמנט יכול להיות גם מערך חד ממדי או רב ממדי, ואז יש להשתמש ב-subscript כדי להגדיר את ממדיו
- כל אחד מהאלמנטים בתוך הסוג החדש צריך להיות מסוג (type) מוכר באותו שלב
- כדי לגשת לאלמנט בתוך משתנה מהסוג החדש, יש להשתמש בסימן . (נקודה)

דוגמאות

```
Public Type Point
   x As Integer
   y As Integer
End Type
```

- בשורות קוד אלו הוגדר סוג חדש בשם Point שיש לו שני שדות: x ו-y

```
Public Type Rectangle
   ptTopLeft As Point
   ptBottomRight As Point
End Type
```

- בשורות קוד אלו הוגדר סוג חדש בשם Rectangle, ובו שני שדות: ptTopLeft ו-ptBottomRight מסוג Point שהוגדר קודם

```
Public Type Scribble
   ptArr(0 To 99) As Point
End Type
```

- בדוגמה זו הוגדר סוג חדש בשם Scribble (שרבוט) המכיל מערך מסוג Point בגודל 100

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 7:

הרצה וניפוי שגיאות

בשיעור זה:

- תכיר את אפשרויות ההרצה השונות
- תכיר את מרכיבי סביבת הפיתוח המאפשרים ניפוי שגיאות ב-Visual Basic
- תבין מהם שלושת המצבים: Design Time, Run Time ו-Break Mode
- תכיר את האובייקט Debug
- תכיר את המשפט On Error... המיועד לטיפול בשגיאות

חלונות תצוגה שונים לאבחון

- מתוך התפריט View ישנן ארבע תצוגות מעניינות, העשויות לסייע באבחון שגיאות בתוכנית:
  - **Immediate Window:** בנוסף לאפשרויות שהוזכרו בשיעור הקודם, ניתן להעביר לחלון ה-Immediate פלט מהאובייקט Debug (נדון בהמשך שיעור זה)
  - **Locals Window:** מאפשר לצפות במשתנים מקומיים בפרוצדורה
  - **Watch Window:** מאפשר לצפות במשתנים שנוספו לחלון ה-Watch (נדון בהמשך)
  - **Call Stack:** מציג את מצב הקריאות לפרוצדורות במחסנית. חלון זה מציג את שרשרת הקריאות לפרוצדורות עד לנקודה מסוימת בתוכנית.

שלושת המצבים של תוכנית

- **Design Time:** זהו השלב שבו התוכנית נמצאת בעיצוב, במילים אחרות, השלב שבו אתה מניח פקדים על טפסים, כותב קוד אך עדיין לא מבצע אותו קרוי Design Time

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Run Time:** זהו המצב שבו שורות הקוד שבתוכנית שלך מתחילות להתבצע
- **Break Mode:** זהו מצב שבו מושהית התוכנית (אינה מופסקת אלא מושהית)

התפריט Run (הרצה)

- ב-Visual Basic קיימות מספר אפשרויות להרצה
- האפשרויות השונות כלולות בתפריט Run
- הפעלת אפשרות להרצה מעבירה את התוכנית למצב Run Time

Start (התחל)

- Start היא צורת ההרצה השכיחה
- ניתן להפעיל את Start בשלושה אופנים:
  - לבחור Start מהתפריט Run
  - ללחוץ על הכפתור המתאים בסרגל הכלים
  - ללחוץ על המקש F5

שים לב: בעת הרצה עם Start לא מבוצע קימפול מלא של התוכנית

Start With Full Compile (התחל לאחר קימפול מלא)

- Start With Full Compile מבצע קימפול לפני ההרצה
- הרצה כזו מומלצת לשם גילוי שגיאות כבר בזמן הקימפול, למשל משתנים שלא מוצהרים
- ניתן להפעיל אפשרות זו מתפריט Run או על ידי צירוף המקשים Ctrl+F5

Break (עבור ל-Break Mode)

- Break מאפשר לבצע מעבר ממצב ריצה למצב Break
- אפשרות זו שימושית כשרוצים לעבור במצב מסוים לשלב Debug

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- אפשר להפעיל את Break בשלוש צורות:
  - בחירת Break מהתפריט Run
  - בחירת הכפתור המתאים בסרגל הכלים
  - הפעלת צירוף המקשים Ctrl+Break
- האפשרות האחרונה (צירוף המקשים Ctrl+Break) שימושית במיוחד כשנכנסים למצב שסרגלי הכלים והתפריטים לא נגישים: למשל בלולאות אינסופיות, או כאשר מופיע Message Box מודאלי

End (סיים)

- End מסיים את פעולת התוכנית
- ניתן להפעיל את End מהתפריט Run או מסרגל הכלים

Restart (התחל מחדש)

- Restart מאפשר להפעיל מחדש את התוכנית
- ניתן לבחור ב-Restart מהתפריט Run או להפעיל את צירוף המקשים Shift+F5

התפריט Debug (ניפוי שגיאות)

- התפריט Debug מאפשר לבצע כמעט כל דבר אפשרי בזמן ריצת התוכנית לשם ביצוע אבחון של התוכנית
- אפשרויות ה-Debug כלולות תחת תפריט Debug

Step Into (צעד לתוך)

- אפשרות זו יכולה לפעול במספר צורות:
  - אם התוכנית נמצאת ב-Design Time, התוכנית תתחיל לרוץ ותעבור מיד למצב של Break Mode כאשר יש שליטה על כל צעד וצעד בתוכנית
  - אם התוכנית עומדת להפעיל פרוצדורה הכתובה בתוכנית, תתבצע כניסה לתוך הפרוצדורה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ניתן להפעיל אפשרות זו מהתפריט Debug או באמצעות F8

Step Over (צעד מעבר)

- אפשרות זו פועלת כמו Step Into, אלא שבמקרה שהתוכנית קוראת לפרוצדורה, לא תתבצע כניסה לתוך הפרוצדורה
- ניתן להפעיל אפשרות זו מתוך התפריט Debug או על ידי הצירוף Shift+F8

Step Out (צא החוצה)

- מבצע יציאה מפרוצדורה פנימית לחיצונית
- ניתן להפעיל אפשרות זו מתוך התפריט Debug או על ידי Ctrl+Shift+F8

Run To Cursor (הרץ עד הסמן)

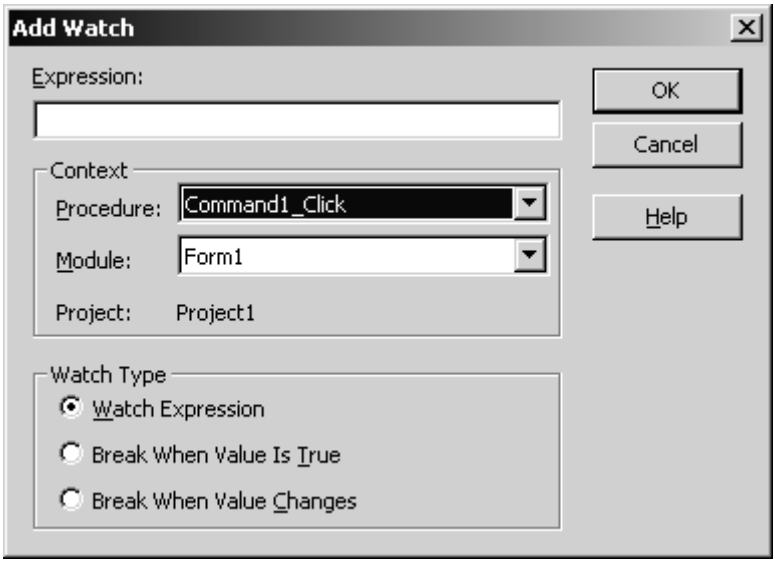
- אפשרות זו מריצה את התוכנית עד אשר שורת הביצוע מגיעה למיקומו של הסמן ואז התוכנית עוברת ל-Break Mode
- ניתן להפעיל אפשרות זו מתוך התפריט Debug או באמצעות הצירוף Ctrl+F8

Add Watch (הוספת תצפית)

- באמצעות Watch אפשר לעקוב אחר ערכו של משתנה במהלך ריצת התוכנית
- ניתן להוסיף Watch מהתפריט Debug

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- הבחירה ב-Add Watch פותחת את התיבה Add Watch המוצגת להלן:



- ב-Expression רושמים את הביטוי שאחריו רוצים לעקוב
- Procedure, Module ו-Project מגדירים את מיקומו של הביטוי מבין כל הפרויקטים, המודולים והפרוצדורות שנמצאים בעריכה
- Watch Type מגדיר את הפעולה שברצוננו לבצע:

  - **Watch Expression:** מציג את ערכו של הביטוי בכל נקודת זמן בתוכנית (אם הביטוי אינו קיים באותו שלב, תוצג לידו ההודעה Out Of Context)
  - **Break When Value Is True:** מעביר את התוכנית ל-Break Mode ברגע שהביטוי הוא True
  - **Break When Value Changes:** מעביר את התוכנית ל-Break Mode כאשר ערך הביטוי משתנה

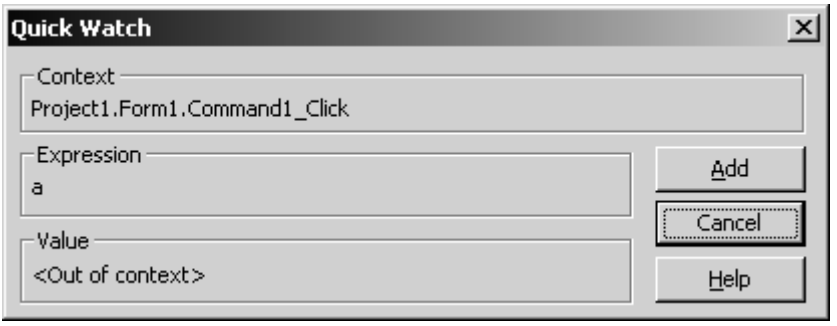
© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Edit Watch (ערוך תצפית)

- מאפשר לשנות הגדרות Watch
- תיבת הדו-שיח הנפתחת דומה מאוד לזו של Add Watch, אלא שגם קיימת האפשרות למחוק Watch
- כדי להפעיל אפשרות זו יש לבחור אותה מהתפריט Debug או להפעיל את צירוף המקשים Ctrl+W

Quick Watch (תצפית מהירה)

- אפשרות זו מאפשרת לבצע תצפית מיידית על ערכו של משתנה
- כדי להפעיל אפשרות זו יש:
  - לבחור ביטוי בחלון הקוד
  - להפעיל את האפשרות מהתפריט Debug או על ידי צירוף המקשים Shift+F9
- החלון הנפתח בתגובה הוא:



- **Context:** מתאר את מיקום הביטוי בתוך מרחב העבודה של Visual Basic, כלומר את הנתיב לפרוצדורה שבה ממוקם הביטוי הנבחן (שם פרויקט, שם מודול, שם פרוצדורה וכו')
- **Expression:** זהו הביטוי הנצפה
- **Value:** זהו ערכו של המשתנה. אם המשתנה לא קיים באותו רגע, תוצג ההודעה Out of context

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Toggle Breakpoint (הפוך מצב של נקודת עצירה)

- Breakpoint היא נקודת עצירה הממוקמת במשפט מסוים בתוכנית
- תוכנית רצה עוברת ל-Break Mode בעת הגעה ל-Breakpoint
- לחצן זה הוא דו מצבי, כלומר מוסיף או מנקה Breakpoint
- Breakpoint מוצג באמצעות נקודה אדומה בחלון הקוד
- כדי להוסיף או להסיר Breakpoint יש לבחור את האפשרות מהתפריט Debug, או מסרגל הכלים או על ידי המקש F9, או לחיצה על העיגול האדום המסמן Breakpoint בחלון הקוד

Clear All Breakpoints (הסר את כל נקודות העצירה)

- אפשרות זו מסירה את כל נקודות העצירה שמוקמו בקוד
- ניתן להפעיל אפשרות זו מהתפריט Debug או על ידי הצירוף Ctrl+Shift+F9

Set Next Statement (קבע את המשפט הבא לביצוע)

- Next Statement הנו הביטוי הבא שאמור להתבצע לאחר הביטוי הנוכחי
- במצב Break Mode, הביטוי הבא לביצוע מוצג באמצעות חץ צהוב
- בעת הרצה ב-Break Mode ניתן לשנות את הביטוי הבא שיבוצע בשתי צורות:
  - יש להציב את הסמן במשפט שאליו נרצה שהתוכנית תעבור בשלב הבא, ואז יש לבחור באפשרות Set Next Statement מתוך התפריט Debug או להפעיל את הצירוף Ctrl + F9
  - ניתן לגרור את החץ הצהוב אל השורה שרוצים שתהפוך למשפט הבא לביצוע
- יש לשים לב כי המשפט הבא לביצוע חייב להיות באותה הפרוצדורה

Show Next Statement (הצג את המשפט הבא לביצוע)

- אפשרות זו מעבירה את סמן הטקסט אל המשפט הבא לביצוע
- ניתן לבחור באפשרות זו מתוך התפריט Debug

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

האובייקט Debug

- האובייקט Debug הוא אובייקט של סביבת Visual Basic
- לאובייקט Debug יש שתי מתודות:
  - המתודה Print מוציאה פלט לחלון ה-Immediate
  - המתודה Assert בוחנת ביטוי בוליאני, ואם הוא False עוצרת את מהלך התוכנית באותה הנקודה
- הקריאות לאובייקט Debug פעילות בסביבת הפיתוח בלבד. בעת קימפול קובץ ההרצה באמצעות האפשרות Make מתפריט File, קריאות אלו אינן נכללות בקוד המתקמפל.

דוגמה 1

- נניח כי אנו רוצים לכתוב פונקציה בשם SumUpTo המקבלת מספר אי שלילי שלם ומחשבת את סכום כל המספרים מ-0 עד לאותו מספר (כולל)
- נוכל לוודא שהמספר שהועבר הוא אי שלילי באמצעות Debug.Assert
- נוכל להדפיס נתונים לחלון ה-Immediate באמצעות Debug.Print

```
Public Function SumUpTo(ByVal n As Integer) As Integer
   Dim res As Integer

   Debug.Print "The input for SumUpTo is : " & _
               Trim(Str(n)) ' Print to the immediate
   Debug.Assert n >= 0 ' Allow only non-negative arguments

   res = 0
   Do While n > 0
      res = res + n
      n = n - 1
   Loop
   SumUpTo = res
End Function
```

- הרצת קוד זה גורמת להדפסת הערך המועבר לפונקציה לחלון ה-Immediate
- במקרה כזה Debug.Assertאם מועבר ערך שלילי, ההרצה מושהית על ידי המשפט . כדי לדעת מאיזו פרוצדורה התבצעה קריאה לפונקציה זוCall Stackנוכל להיעזר ב-

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

כאמור, הקריאות לאובייקט Debug פועלות רק בסביבת הפיתוח, ואינן נכללות בקובץ ה-EXE המקומפל.

שים לב: Debug.Assert פועלת בסביבת הפיתוח בלבד ואינה מתבצעת בקובץ EXE מקומפל

הערה: מכיוון שהתוצאה מוחזרת כ-Integer, הדוגמה מתאימה לקלט עד 255. עבור 256 ומעלה תתרחש שגיאת Overflow.

טיפול בשגיאות בקוד התוכנית

- אובייקטים של Visual Basic מטפלים בשגיאות בסגנון ה-Exception Handling
- פונקציות המיובאות מ-DLL רגילים מחזירות ערך המצביע בדרך כלל על הצלחה או כישלון של הקריאה
- כאשר מפעילים פונקציות מ-DLL רגילים (שים לב שהכוונה ל-DLL שונים מאלו שתלמד לבנות בשיעורים אודות ActiveX) בודקים את הערך המוחזר כדי לזהות שגיאה
- כאשר מפעילים קוד סטנדרטי של Visual Basic יש ללכוד חריגים באמצעות המשפט On Error

שים לב: שגיאה שלא טופלה תגרום בדרך כלל להצגת הודעת שגיאת זמן ריצה ולהפסקת ביצוע התוכנית. לכן יש לטפל בשגיאות הצפויות

האובייקט Err

- בזמן התרחשות שגיאה, הנתונים אודות השגיאה מאוחסנים באובייקט Err
- בזמן לכידת שגיאה יש לבחון את האובייקט Err
- האובייקט Err מאפשר לעורר שגיאה ולהעבירה לקוד הקורא לפונקציה
- המאפיינים והמתודות של האובייקט Err הם:
  - **Number:** מספר השגיאה שהתרחשה
  - **Description:** תיאור מילולי של השגיאה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Source:** זוהי מחרוזת המכילה את המקור לשגיאה שהתרחשה, למשל שם המחלקה שיצרה את השגיאה (הנושא מחלקות נלמד בהמשך הספר)
- **LastDLLError:** מחזיר את קוד השגיאה האחרון שנוצר ב-DLL (רגיל) שהתבצעה אליו קריאה. ערך זה המצוי ב-LastDLLError שקול לקריאה GetLastError מתוך Windows API
- **HelpFile:** מכיל את הנתיב לקובץ עזרה הקשור לשגיאה שהתרחשה
- **HelpContext:** מכיל את זיהוי הנושא בקובץ העזרה
- **Clear:** מתודה זו מנקה את האובייקט Err. מומלץ לבצע זאת לאחר ששגיאה טופלה
- **Raise:** מתודה זו מאפשרת לעורר שגיאה. משתמשים במתודה זו בעיקר בתכנות רכיבים חדשים (הנושא נדון בשיעורים אודות ActiveX)

On Error...

- המשפט On Error בא בשלושה טעמים:
  - On Error GoTo *line*
  - On Error Resume Next
  - On Error GoTo 0

On Error GoTo line

- משפט זה מעביר את הביצוע לשורה המצוינת במילה line כאשר שגיאה מתרחשת
- כדי לציין שורה יש להשתמש ב-Labels או במספרי שורות
  - כדי לציין Label יש לרשום שם חוקי עם נקודתיים בתחילת שורה
  - ניתן להשתמש במספרי שורות בקוד Visual Basic, אך זוהי שיטה ישנה המתאימה לימי הביניים. בכל אופן, מספר חייב להופיע בתחילת השורה
- חשוב למקם משפט יציאה מהפרוצדורה: Exit Sub או Exit Function לפני קוד הטיפול בשגיאות, על מנת שהוא לא יתבצע כאשר הקוד בוצע בהצלחה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה 2

```
Public Function Div(ByVal Num1 As Double, _
                    ByVal Num2 As Double) As Double
   On Error GoTo Err_Handler
   Div = Num1 / Num2
   Exit Function
Err_Handler:
   MsgBox "The following error occured: " & Err.Description
End Function
```

- פונקציה זו מחלקת שני מספרים
- אם תתרחש שגיאה (למשל שגיאת חלוקה ב-0) תועבר השליטה לקוד שאחרי התווית Err_Handler
- קוד הטיפול פשוט מציג הודעה עם תיאור השגיאה שהתרחשה
- שים לב שלפני התווית יש יציאה מפונקציה, כדי למנוע את ביצוע הקוד אם לא התרחשה שגיאה
- אין צורך לקרוא למתודה Err.Clear, כיוון שהאובייקט Err מתנקה אוטומטית בעת יציאה מפונקציה

On Error Resume Next

- משפט זה מעביר את הקוד לביצוע לשורה הבאה שלאחר השגיאה

On Error GoTo 0

- משפט זה מכבה (Turn Off) טיפול בשגיאות שהוגדר, כלומר אם התבצעה קריאה ל-*line* On Error GoTo או ל-On Error Resume Next הם יבוטלו
- טוב להשתמש במשפט זה אם רוצים לטפל באופן שונה בשגיאות המתרחשות במקומות שונים בקוד

המשפט Resume

- המשפט Resume מאפשר להמשיך את ביצוע התוכנית בשורה מסוימת לאחר טיפול בשגיאה שהתרחשה
- קיימות שלוש אפשרויות:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Resume:** מחזיר את הביצוע לשורה ממנה התרחשה השגיאה
- **Resume Next:** מעביר את הביצוע לשורה שאחרי השורה שבה התרחשה השגיאה
- **Resume *line*:** מעביר את הביצוע לשורה המצוינת על ידי תווית או מספר שורה (כמובן שהשורה חייבת להיות באותה פרוצדורה)

◄ לאחר קריאה ל-Resume האובייקט Err מתנקה אוטומטית

דוגמה 3

```
Public Function Div(ByVal Num1 As Double, _
                    ByVal Num2 As Double) As Double
   On Error GoTo Err_Handler
   Div = Num1 / Num2
   Debug.Print Err.Number, Err.Description
   Exit Function
Err_Handler:
   Num2 = 1#
   Resume
End Function
```

◄ דוגמה זו זהה לקודמת, אלא שכאן בקוד הטיפול משנים את ערכו של Num2 ל-1 ומחזירים את הביצוע לשורה שיצרה את השגיאה, כך נמנעים מחלוקה ב-0

הפונקציה Error

◄ הפונקציה Error מאפשרת לקבל תיאור של שגיאה לפי המספר שלה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 8:

Componentware Development - תכנות מבוסס רכיבים

בשיעור זה:

- תבין את ההבדלים בין Procedural Programming (תכנות פרוצדורלי) לבין Object Oriented Programming (תכנות מונחה עצמים OOP)
- תבין מהו Event Oriented Programming (תכנות מונחה אירועים)
- תבין את עקרונות COM
- תבין מהם COM Interfaces (ממשקי COM), ואת הצורך בהם
- תלמד אודות התמיכה של Visual Basic בתכנות מונחה עצמים, תכנות מונחה אירועים ותכנות רכיבים
- תהיה מוכן להרטיב את הרגליים בשיעור הבא העוסק בתכנות מונחה עצמים ב- Visual Basic

תכנות פרוצדורלי לעומת תכנות מונחה עצמים

מבדילים בין שני סוגים עיקריים של שפות תכנות:

- Procedural Languages
- Object Oriented Programming Languages

Procedural Programming

- בתכנות פרוצדורלי, הפונקציות והרוטינות הן מרכז הפעילות
- הגדרה של סוגי נתונים חדשים בתכנות פרוצדורלי אפשרית, אך במקביל יש לכתוב פרוצדורות המסוגלות לטפל בסוגים החדשים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- סוגי נתונים מוגדרים על ידי המשתמש יכולים להכיל בתוכם משתנים מכל הסוגים המוכרים עד לנקודה בה הם הוגדרו
- סוגי נתונים מוגדרים על ידי המשתמש אינם יכולים להכיל בתוכם פונקציות
- בתכנות פרוצדורלי, המשתנים והפרוצדורות חיים בנפרד
- בדרך כלל משתמשים ב-Modular Programming (תכנות מודולרי) כדי ליצור איזושהי רמה של הסתרת מידע. Modular Programming משמעותו היא שהקוד מרוכז במודולי קוד (המקבילים למודולים סטנדרטיים ב-Visual Basic), ובאופן זה ניתן להסתיר משתנים מסוימים שאין צורך שייראו מחוץ למודול. מודול הנו קובץ בודד של קוד.

דוגמה

- נמחיש את ההיבטים הללו של תכנות פרוצדורלי באמצעות יישום פשוט בשפת C וב-Visual Basic תוך שימוש במגבלות התכנות הפרוצדורלי
- נכתוב תוכנית המנהלת רשימה של 20 עובדים ומאפשרת את הפונקציונליות הבאה:
  - הוספת רשומת עובד (עד 20 עובדים, כאמור)
  - הדפסת פרטי העובדים

1. הגדרת סוג הנתונים:

- בשלב הראשון נגדיר סוג נתונים חדש בשם Worker שיכיל את השדות ID, Name, Address ו-Phone
- בשפת C היינו מגדירים זאת כך:

```
#include <stdio.h>
#include <stddef.h>
#include <tchar.h>

#define SIZE_NAME     20
#define SIZE_ADDRESS     100
#define SIZE_PHONE    20

typedef struct tagWorker
{
   long ID;
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   TCHAR Name[SIZE_NAME];
   TCHAR Address[SIZE_ADDRESS];
   TCHAR Phone[SIZE_PHONE];
} Worker;
```

- ב-Visual Basic נוכל להשתמש בהגדרה הבאה:

```
Public Const SIZE_NAME As Integer = 20
Public Const SIZE_ADDRESS As Integer = 100
Public Const SIZE_PHONE As Integer = 20

Public Type Worker
   ID As Long
   Name As String * SIZE_NAME
   Address As String * SIZE_ADDRESS
   Phone As String * SIZE_PHONE
End Type
```

2. כתיבת פרוצדורות:

- בשלב השני נרצה לכתוב פונקציה המזינה נתונים לרשומה מסוג Worker
- בשפת C היינו כותבים זאת כך:

```
void SetWorkerData(Worker *pWorker, long id, _TCHAR *name,
                   _TCHAR *address, _TCHAR *phone)
{
   pWorker->ID = id;
   _tcsncpy(pWorker->Name, name, SIZE_NAME - 1);
   _tcsncpy(pWorker->Address, address, SIZE_ADDRESS - 1);
   _tcsncpy(pWorker->Phone, phone, SIZE_PHONE - 1);
}
```

- ב-Visual Basic ניתן לרשום רוטינה דומה באופן הבא:

```
Public Sub SetWorkerData(ByRef pWorker As Worker, _
                         ByVal nID As Long, _
                         ByRef strName As String, _
                         ByRef strAddress As String, _
                         ByRef strPhone As String)
   pWorker.ID = nID
   pWorker.Name = strName
   pWorker.Address = strAddress
   pWorker.Phone = strPhone
End Sub
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- לבסוף נכתוב פונקציה המדפיסה את מערך העובדים
- בשפת C אפשר לכתוב אותה באופן הבא:

```
void PrintWorkerData(FILE *out, Worker *pWorker,
                     size_t nWorkers)
{
   size_t i;
   for(i = 0; i < nWorkers; i++)
   {
      _ftprintf(out,
             _T("ID: %ld\n"
             "Name: %s\n"
             "Address: %s\n"
             "Phone: %s\n\n"),
             (pWorker + i)->ID,
             (pWorker + i)->Name,
             (pWorker + i)->Address,
             (pWorker + i)->Phone);
   }
}
```

- וב-Visual Basic הרוטינה הבאה תהיה טובה מספיק:

```
Public Sub PrintWorkerData(ByRef out As TextBox, _
                           ByRef pWorker() As Worker, _
                           ByVal nWorkers As Integer)
   Dim i As Integer
   For i = 0 To nWorkers - 1
      out.Text = out.Text & _
                 "ID: " & Str(pWorker(i).ID) & vbCrLf & _
                 "Name: " & pWorker(i).Name & vbCrLf & _
                 "Address: " & pWorker(i).Address & vbCrLf & _
                 "Phone: " & pWorker(i).Phone & vbCrLf & vbCrLf
   Next
End Sub
```

3. כתיבת תוכנית דוגמה:

- ניתן להמשיך ולכתוב תוכנית המדגימה את השימוש בסוג הנתונים החדש, אך את זאת אני משאיר לך כתרגיל. בכל מקרה תוכל למצוא את תוכניות הדוגמה (ב-C וב-Visual Basic) בדיסק.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

מסקנה

- מעיון בקוד הדוגמה, קל להבחין שישנה הפרדה בין הפרוצדורות לבין הנתונים
- כדי להפעיל פונקציה על משתנה מסוג Worker יש להעביר אותו כפרמטר לפונקציה
- היה יותר נוח לו יכולנו לרשום:

```
pWorker(i).SetData(....)
```

בדיוק כפי שאנו רושמים כשאנחנו קוראים למשל למתודה Show של Form, אך זה כבר עניין ל-OOP

Object Oriented Programming

- בתכנות מונחה עצמים מגדירים סוגי נתונים חדשים באמצעות Class (מחלקה) ולא באמצעות Type (או Structבשפת C)
- במחלקות אורזים יחד משתנים ופרוצדורות. באופן כזה יוצרים יחידה לוגית אחת הכוללת משתנים ומתודות הפועלות על המשתנים הללו
- משתנים של מחלקה קרויים member variables, ומתודות של מחלקה קרויות member methods
- אובייקטים הם מופעים (Instances) של מחלקות. כלומר משתנים המוצהרים בתוכנית מסוג מחלקה מסוימת הנם אובייקטים של המחלקה
- כאשר יש להפעיל מתודה של אובייקט, אין צורך להעביר את האובייקט למתודה (כפי שראינו בדוגמה הקודמת). המתודה היא חלק פנימי של האובייקט כך שהפעלת המתודה נעשית באמצעות האובייקט עצמו
- בתכנות מונחה עצמים קיימים שלושה רעיונות מרכזיים:
  - Encapsulation (כמיסה)
  - Inheritance (הורשה)
  - Polymorphism (רב צורתיות)

Encapsulation (כמיסה)

- כמיסה היא היכולת של מחלקה להסתיר members מסוימים, ולחשוף אחרים
- ה-members החשופים של מחלקה קרויים ה-Interface (ממשק) של המחלקה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- רעיון הכמיסה הנו רעיון מרכזי ב-OOP כיוון שבעזרת כמיסה ניתן למנוע מקוד לא מורשה לגשת ל-members פרטיים באובייקט. כך לדוגמה, לא תרצה שיוזן מספר זיהוי שאינו חוקי ל-member של אובייקט המאחסן מספר תעודת זיהוי. כיוון שכך, מומלץ להפוך את כל ה-member variables של מחלקה לפרטיים (עד כמה שהדבר ניתן), ובמקביל, לאפשר גישה אליהם באמצעות member methods שייכתבו כך שלא יפרו את כללי הארגון (אלא אם אתה חתרן בלתי נלאה)
- מחלקה המעוניינת להסתיר members אך במקביל לאפשר גישה ל-members הללו למחלקות ספציפיות תוכל לעשות זאת על ידי הגדרת המחלקות הללו כ-Friends (כך נעשה הדבר בשפת C++)
- ב-Visual Basic הרעיון בבסיסו דומה, אך קיים שוני מסוים: ב-Visual Basic הגדרת members כ-Friends הופכת אותם לציבוריים עבור כל הקוד שבפרויקט (מחלקות ומודולים רגילים), אך לא נגישים לקוד מפרויקטים אחרים
- Friends פוגעים ברעיון הכמיסה, ולפיכך מומלץ שלא להשתמש בהם אלא במקרים מיוחדים מאוד

Inheritance (הורשה)

- הורשה היא היכולת של מחלקה לרשת תכונות ממחלקה אחרת
- המחלקה היורשת קרויה מחלקה נגזרת (Subclass), והמחלקה המורישה קרויה מחלקת בסיס (Superclass)
- הורשה מתבטאת בשני היבטים:
  - הורשת ממשק
  - הורשת מימוש
- **הורשת ממשק** מתבטאת בהורשת שמות המאפיינים והמתודות ממחלקת הבסיס למחלקה הנגזרת
- **הורשת מימוש** מתבטאת בהורשת מימוש המתודות והמאפיינים ממחלקת הבסיס למחלקה הנגזרת

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

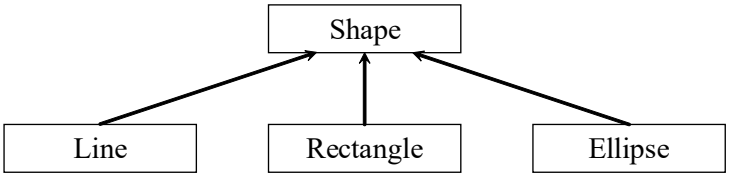
- הורשה היא אחד ממנגנוני היסוד של תכנות מונחה עצמים, ומאפשרת שימוש חוזר בקוד. משפות תכנות מונחות עצמים מלאות אנו מצפים שיתמכו בהורשת מימוש ובהורשת ממשקים.
- השימוש בהורשה צריך להתבצע בעת זיהוי יחסי אב-בן בין מחלקות. ניתן להיעזר בתבנית ClassX הוא סוג של ClassY כדי לזהות יחסי הורשה, כאשר ClassX היא המחלקה הנגזרת ו-ClassY היא מחלקת הבסיס. לדוגמה:
  - דוגמה נכונה: מלבן הוא צורה
  - דוגמה נכונה: מנהל הוא סוג של עובד
  - דוגמה לא נכונה: מכונית היא ארבע דלתות, מנוע, שלדה...
    נכון להגיד: מכונית מכילה ארבע דלתות, מנוע, שלדה...
    צריך להיזהר שלא ליפול למלכודת הטמונה בבלבול שבין יחסי הורשה (Inheritance) לבין יחסי הכלה (Containment)
- Multiple Inheritance (הורשה מרובה) היא היכולת לרשת ממספר מחלקות בו זמנית
- שפות מונחות עצמים תומכות בהורשה מרובה במידה כזו או אחרת. C++ למשל מאפשר הורשה מרובה מלאה, Java מאפשרת הורשה מלאה ממחלקה אחת והורשה של ממשקים נוספים. Visual Basic מאפשרת מימוש ממשקים בלבד (חיה מוזרה שדומה להורשת ממשקים)
- דוגמאות להורשה מרובה:
  - דוגמה נכונה: רכב אמפיבי הוא רכב יבשתי וגם רכב ימי
  - דוגמה לא נכונה: ComboBox הוא גם TextBox והוא גם ListBox
    נכון יהיה להגיד: ComboBox מכיל בתוכו TextBox ו-ListBox

Polymorphism (רב צורתיות)

- Polymorphism הוא היכולת של אובייקט ללבוש צורות שונות, או במילים אחרות לתפעל את אותו member באופנים שונים
- Polymorphism נובע מהשימוש בהורשה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

◄ קל יותר להבין מהו Polymorphism באמצעות דוגמה:
נניח שברצוננו ליצור תוכנית גרפית פשוטה המאפשרת ציור של שלוש צורות: Line (קו), Rectangle (מלבן) ו-Ellipse (אליפסה). ברור שנרצה לאחסן את הצורות באמצעות מבנה נתונים מסוים. הבעיה הראשונה שתתעורר היא שלא נוכל להשתמש במבנה נתונים הכולל מצביעים מסוג Line-ים, למשל, לאחסון Rectangle-ים או Ellipse-ות. כדי לפתור בעיה זו נגדיר את כל המחלקות הללו כנגזרות של מחלקה מופשטת בשם Shape. היררכיית המחלקות תיראה, אם כן, כמו בציור הבא (כאשר החץ משמעותו "יורש מ..."):



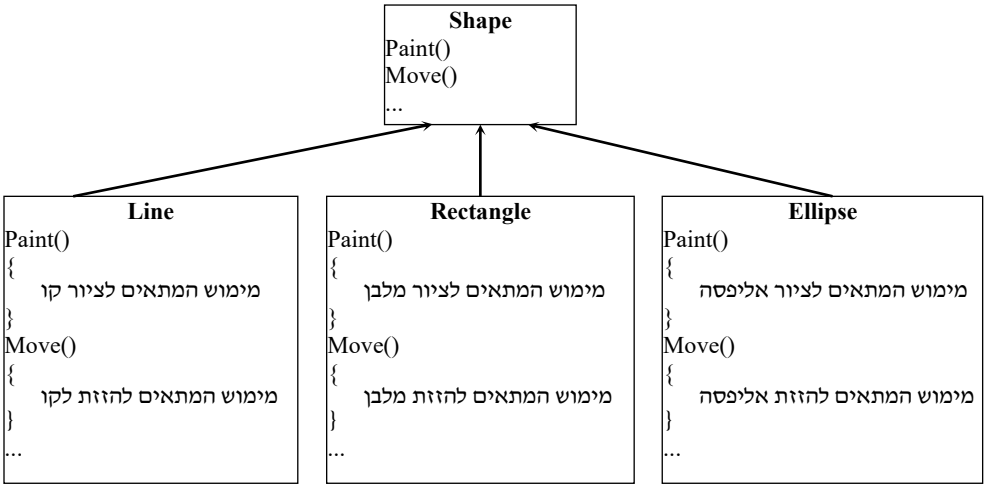
את הבעיה הראשונה פתרנו, כיוון שבשפות מונחות עצמים מצביע למחלקת בסיס (במקרה זה Shape) יכול להצביע לסוגים נגזרים (במקרה זה Line, Rectangle ו-Ellipse). כלומר סוג המצביע במבנה הנתונים יהיה מסוג Shape.

הבעיה השנייה היא כיצד נבצע סריקה על כל האובייקטים במבנה הנתונים לצורך ביצוע פעולה כלשהי (למשל ציור על המסך). הבעיה נובעת מכך שבפועל כל תא במבנה מצביע לסוג נתונים לא ידוע (Line, Rectangle או Ellipse). בעיה זו אינה בעיה כלל ועיקר, כיוון שנוכל להשתמש במצביע מחלקת בסיס כדי לבצע סריקה במבנה הנתונים ובאמצעות מצביע זה נפעיל את המתודה הרצויה. השאלה הנשאלת כעת היא איזו מתודה בדיוק תופעל, האם זו של המחלקה Shape ממנה מוגדרים המצביעים במבנה הנתונים, או אולי המתודה של האובייקט המוצבע בפועל, כאשר זה יכול להיות שונה בין תא לתא במבנה הנתונים. בהנחה שהמתודה תומכת בהפעלה פולימורפית, תופעל המתודה של האובייקט המוצבע בפועל ולא זו של סוג המצביע. זהו Polymorphism במיטבו.

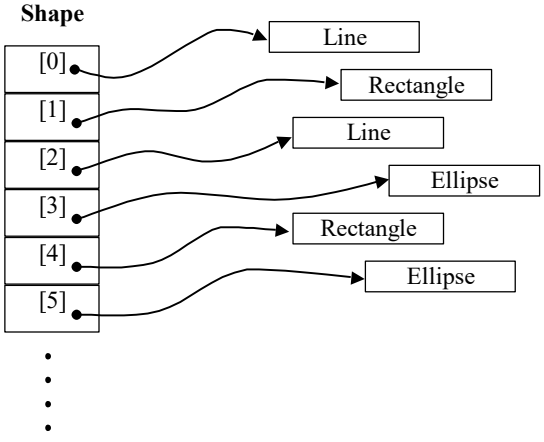
כדי להשלים את התמונה, ובכדי להפוך את העניין למוחשי יותר, נגדיר מתודות המשותפות לכל הצורות כחלק מהממשק של Shape. מתודות לדוגמה יהיו Paint,

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Move. מתודות אלו עוברות בירושה לכל אחת מהמחלקות הנגזרות (כלומר Line, Rectangle ו-Ellipse) ומקבלות בכל מחלקה משמעות שונה. כך למשל, Paint עבור Line פירושו שימוש בפונקציות מערכת היודעות לצייר קווים על גבי המסך. Paint עבור Rectangle ימומש באמצעות קריאה לפונקציה מערכת המתמחה בציור מלבנים.
היררכיית המחלקות אם כן תיראה כך:



כעת נניח שברשותנו מערך של Shape-ים שכל תא בו מצביע על צורה מסוג כלשהו:



© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

כעת נוכל להצהיר על משתנה מסוג Shape ולבצע סריקה על המערך. מתוך המצביע הזה נפעיל את המתודה Paint. מה שיופעל למעשה אלו הן המתודות של האובייקטים המוצבעים, ובאופן יותר ספציפי עבור האיבר עם האינדקס [0], תופעל המתודה Paint של Line, עבור האיבר עם האינדקס [1] תופעל המתודה Paint של Rectangle, וכן הלאה.

Event Oriented Programming (תכנות מונחה אירועים)

- בתכנות מונחה אירועים זרימת התוכנית מונחית על פי האירועים המתרחשים
- ב-Visual Basic למשל, מוכרים לך האירועים MouseDown, Click וכו'. קוד הנכתב בפרוצדורות האירועים הללו יופעל בתגובה לפעולות המשתמש.
- לכל אירוע יש שני צדדים: הצד המעורר את האירוע והצד המגיב לאירוע. לדוגמה, באירוע MouseMove הצד המעורר הוא מערכת ההפעלה, המזהה את תנועת העכבר, והצד המגיב הוא התוכנית שלך.
- אירועים אינם קשורים בהכרח לפעולות משתמש, למשל האירוע DataArrival של פקד Winsock מתרחש בכל פעם שמידע מגיע למחשב המקומי ממחשב אחר ברשת

Component Object Model (COM)

- COM מאפשר להתמודד טוב יותר עם הבעיות והאתגרים הבאים:

  - **פיתוח תוכניות מודולריות ברמה הבינרית:**
    כתיבת רכיבי קוד בשפות שונות מעלה בעיות של תאימות בינרית בין רכיבי הקוד. לעתים יש רצון וצורך לשפר פונקציונליות של תוכנה מסוימת. פעולה מהסוג הזה כרוכה בדרך כלל בהחלפת התוכנה כולה. לצורך ההדגמה, נניח שברשותך מעבד תמלילים מדהים שבדיקת האיות שלו זוועתית. היה נחמד לו יכולת לרכוש רכיב בדיקת איות מחברה צד ג' ולשלב אותו כחלק אינטגרלי של מעבד התמלילים.
  - **שימוש חוזר ברכיבים ותאימות גרסאות רכיבים:**
    יישומי תוכנה בסדר גודל בינוני ומעלה הופכים קשים לפיתוח ולתחזוק. לעתים תכופות מוצאים עצמם מתכנתים משתמשים בפונקציונליות שפותחה זה מכבר עבור תוכנה אחרת. הבעיה העיקרית היא, שמאוד מסוכן לשנות מחלקות, או להוסיף

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

פונקציונליות למחלקות בשל פגיעה לא צפויה במחלקות אחרות הנגזרות או משתמשות במחלקה הספציפית.

- **שקיפות בהפעלת רכיב שרת ללא קשר למיקומו ביחס לרכיב הלקוח:**
רכיבים הפועלים מול רכיבים אחרים, מושפעים ממיקום הרכיב ביחס אליהם, כלומר, האם הרכיב מצוי באותו מרחב כתובות (אותו Process), או במרחב כתובות שונה באותו מחשב (Process שונה באותו מחשב), או אולי במחשב אחר ברחבי הרשת. ברור שפניה לרכיב המצוי במחשב אחר ברחבי הרשת שונה בתכלית מפניה לרכיב המצוי תחת אותו מרחב כתובות, שכן פניה לרכיב ברשת כרוכה בהעברת מסרים בפרוטוקולים שונים ומשונים וכו'. היית מאושר לו יכולת להפעיל רכיב מרוחק בדיוק באותו האופן בו אתה מפעיל רכיב מקומי וכל משימת ההתקשרות דרך הרשת, או חציית מרחבי כתובות הייתה נופלת על כתפי מישהו אחר (ביל גייטס, למשל).

- COM מגדיר תקנים לפיתוח רכיבים: באמצעות ממשקים סטנדרטיים שרכיבי COM אמורים לתמוך בהם. אפשר להגיד ש-COM הוא הדבק שבין רכיבים
- כתיבת רכיבים בהתאם לספציפיקציה של COM מאפשרת שימוש חוזר ברמה בינרית, ללא תלות בשפות שבהן נכתבו הרכיבים. היתרון הנובע מכך הוא שניתן לפתח רכיב בשפה מסוימת המתאימה ביותר לפיתוח אותו רכיב ואז אפשר יהיה להשתמש בו בשפות שונות
- התמיכה הקיימת במערכת ההפעלה עבור COM מאפשרת לבצע את ההתחברות בין רכיבים. כדי להבהיר את העניין המעורפל הזה אשתמש באנלוגיה ובסוגריים ארשום את ההקבלה:
נניח שישנם שלושה אנשים איש א' (רכיב א), איש קשר (מערכת COM הבנויה כחלק אינטגרלי ממערכת ההפעלה) ואיש ב' (רכיב ב). בין איש א'(רכיב א') לאיש ב' (רכיב ב') נמצא קיר, כך שאף אחד מהם לא יודע האם השני קיים או לא. כדי לברר זאת שואל איש א' (רכיב א') את איש הקשר (מערכת COM) האם איש ב' (רכיב ב') נמצא והאם הוא יודע לדבר בשפה שלנו (COM) התשובה תהיה חיובית (נניח שהיא חיובית) ובשלב זה יקשר איש הקשר (מערכת COM) בין איש א' (רכיב א') לאיש ב' (רכיב ב') ויעזוב

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

אותם לנפשם. מכאן והלאה איש א' (רכיב א') ואיש ב' (רכיב ב') ידברו ביניהם – ימצאו או לא ימצאו עניין משותף (Interface) זהו עניינם.

- רכיב מאפשר גישה אליו על ידי חשיפת ממשק (Interface) שדרכו יוכלו לקוחות להתחבר אליו. נניח למשל שאתה מעוניין לפתח רכיב לביצוע חישובים מתמטיים מסובכים, ודאי תחשוף ממשק המאפשר ביצוע חישובים סטטיסטיים (ייתכן שתקרא לו IStatistics)
- ממשק COM מפורסם כחוזה המגדיר את הפעולות הזמינות. רכיבים שונים יכולים לממש אותו ממשק, ולקוחות יכולים להשתמש בו ללא תלות במימוש הפנימי. בכך מתאפשרת תחרות "הרכיב הטוב ביותר"
- COM אוסר על מפתחי רכיבים לשנות ממשק שכבר פורסם והופץ למשתמשים. שינוי מסוג זה עלול לגרום לקריסתן של תוכניות לקוח שהשתמשו ברכיב בגרסאותיו הקודמות. אם הממשק שיצרת דורש שינוי ושיפור, פשוט צור ממשק חדש שיכיל את הפונקציונליות הנדרשת, והשאר את הממשק הקודם ללא שינוי. שים לב שלא אמרתי שאינך יכול לשנות את המימוש הפנימי של הרכיבים, תוכל לעשות זאת כל אימת שתחפוץ בכך, על כל פנים זכור שהרכיבים שלך מתקשרים באמצעות הממשקים שיצרת עבורם לכן היזהר לפני ביצוע שינויים כלשהם.
- רכיבי COM נחלקים לשלושה סוגים עיקריים:

  - **In Process (תוך תהליכי)**: רכיב COM הפועל כחלק מתוכנית הלקוח (בתוך מרחב כתובות הזיכרון של תוכנית הלקוח)
  - **Out of Process (חוץ תהליכי)**: רכיב המופעל כתוכנית נפרדת, וכתוצאה מכך מוקצה עבורו מרחב כתובות נפרד.
  - **Remote (מרוחק)**: זהו רכיב COM הממוקם במחשב אחר ברחבי הרשת

- COM משתדל להסתיר ממך את פרטי ההתקשרות עם הרכיב, כך שפנייה לרכיב ברחבי הרשת תהיה זהה לפנייה לרכיב תוך תהליכי
- Distributed COM (DCOM) היא טכנולוגיה המאפשרת הפעלת רכיבים מרוחקים ברחבי הרשת, תוך הסתרת פרטי התקשורת בין המחשבים מקוד היישום
- ActiveX היא טכנולוגיה המבוססת על COM

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

התמיכה של Visual Basic ב-OOP ו-COM

- Visual Basic תומכת ב-Encapsulation וב-Polymorphism
- כיוון ש-Visual Basic אינה תומכת בהורשת מימוש, התמיכה שלה בתכנות מונחה עצמים חלקית בלבד.
- Visual Basic תומכת בהורשת ממשקים (יותר מדויק להגיד: מימוש ממשקים, אך נושא זה יידון בפירוט בשיעור הבא)
- כיוון ש-Visual Basic אינה תומכת בהורשת מימוש, לא יהיה נכון לקרוא לה שפה מונחית עצמים טהורה
- Visual Basic תומכת בתכנות מונחה אירועים. כלומר תוכל להגיב לאירועים, כמו גם לכתוב אירועים משלך לרכיבים חדשים שאתה מפתח

רכיבי ActiveX של Visual Basic

- ב-Visual Basic תוכל ליצור רכיבי ActiveX מהסוגים הבאים:
  - **ActiveX DLL:** רכיב קוד היכול להכיל גם אלמנטים של ממשק
  - **ActiveX EXE:** רכיבי COM הפועלים כתוכניות נפרדות, וניתן לתקשר איתם באמצעות COM.
  - **ActiveX Controls:** פקדים חדשים הנוצרים ומותאמים על-ידי המשתמש
- ב-Visual Basic יצירת הממשקים בהם תומך הרכיב נעשית באופן אוטומטי. עם זאת, תוכל ליצור ממשקים ריקים (ללא מימוש) באמצעות IDL (Interface Definition Language) ולממש אותם בשפות שונות
- Visual Basic מקלה מאוד על פיתוח רכיבים. בשפות אחרות (C++ למשל, שלא לדבר על C שמם לא תמצא את הידיים ואת הרגליים) תמצא את עצמך מתעסק עם ממשקים לסידור נתונים לפני שליחה IMarshal, רישום רכיבים לקבלת הודעות (IConnectionPointContainer), וכן STA (Single-Threaded Apartment) ו--MTA (Multi-Threaded Apartment).
- תשמח לשמוע שרכיבים שאתה מפתח ב-Visual Basic נגישים מסביבות אחרות התומכות ב-ActiveX (למשל MFC)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 9:

תכנות מונחה עצמים ב-Visual Basic

בשיעור זה:

- תלמד כיצד ליצור מחלקות ב-Visual Basic
- תלמד להוסיף פרוצדורות מכל הסוגים (Sub, Functions, Properties ו-Events) למחלקות
- תלמד מהו Interface (ממשק) וכיצד ניתן לממש Interface-ים ב-Visual Basic
- תלמד את הצורות השונות בהן ניתן להשתמש באובייקטים בתוכנית
- תבין מהו Reference Counting
- תבין את ההבדל בין Early Binding ל-Late Binding
- תלמד מהו Collection (אוסף), כיצד תוכל ליצור Collection-ים משלך וכיצד לסרוק Collection

Classes (מחלקות)

- ליצירת מחלקות ב-Visual Basic יש להשתמש ב-Class Module

Properties של Class Module

- **Name**: מגדיר את שם המחלקה
- **DataBindingBehavior**: קובע האם האובייקט יכול להיכרך סביב נתונים (נושא זה נדון בהמשך הספר)
- **DataSourceBehavior**: קובע האם האובייקט יכול לשמש כמקור נתונים (גם נושא זה נדון בהמשך הספר)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Events של Class Module

- **Initialize**: מתעורר בכל פעם שנוצר אובייקט חדש מסוג ה-Class שלך
- **Terminate**: מתעורר בכל פעם שאובייקט מסוג ה-Class שלך משתחרר מהזיכרון

הגדרת משתנים ב-Class

- ניתן להגדיר משתנים בתוך Class Module בדיוק כפי שמצהירים עליהם במודול רגיל
- הרשאות הגישה האפשריות עבור הצהרת משתנים:
  - **Private:** משתנה פרטי שאינו נגיש לפרוצדורות מחוץ למחלקה
  - **Public:** משתנה ציבורי שנגיש לפרוצדורות מחוץ למחלקה
- ברירת המחדל היא Private, כלומר הצהרה באמצעות Dim זהה להצהרה באמצעות Private
- מומלץ להסתיר את המשתנים באמצעות המילה Private ולאפשר גישה אליהם באמצעות Properties מתאימים בלבד. באופן כזה אתה שומר על עקרון ה-Encapsulation בתכנות מונחה עצמים

הוספת פרוצדורות ל-Class

- הרשאות הגישה האפשריות עבור פרוצדורות של Class הן:
  - **Private:** הפרוצדורה היא לשימוש פנימי בלבד של ה-Class
  - **Public:** הפרוצדורה נגישה מקוד חיצוני
  - **Friend:** הפרוצדורה נגישה מקוד חיצוני בתנאי שמדובר באותו פרויקט. עבור תוכניות לקוח מפרויקטים אחרים, פרוצדורת Friend תהיה בדיוק כמו Private
- בשיעור הבא תיצור ספריות רכיבים הניתנות לשימוש מתוכניות חיצוניות

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Properties (מאפיינים)

- Properties של Class הן פרוצדורות המאפשרות לקרוא או לקבוע מאפיינים של האובייקט, ובמידת הצורך לבצע פעולות נוספות בעת הגישה אליהם
- היתרון בשימוש במאפיינים Properties הוא בכך שניתן לאתחל משתנים באמצעות הסימן = (שווה) ולהפוך בכך את הקוד לקריא יותר
- Properties באים בשלוש צורות:
  - **Property Let:** מאפשר העברת ערכים ל-Class לשם ביצוע פעולות אתחול למשתנים מסוגים בסיסיים
  - **Property Set:** מאפשר העברת אובייקטים ל-Class לשם ביצוע פעולות אתחול לאובייקטים פנימיים
  - **Property Get:** מאפשר לאחזר ערך מהמחלקה (כלומר לבצע השמה מהמחלקה למשתנה הנמצא בחוץ)
- בדוגמאות שבהמשך תראה את השימוש ב-Properties

Sub (רוטינות) ו-Functions (פונקציות)

- ניתן להוסיף ל-Class רוטינות ופונקציות בדיוק כפי שהיית מוסיף למודולי הקוד שהכרת עד כה

Events (אירועים)

- בדיוק כפי שקיימים Events ל-CommandButton, Labels ואובייקטים ויזואליים (וגם לא ויזואליים), תוכל גם אתה להוסיף Events ל-Class שאתה יוצר
- כדי להוסיף Event יש להצהיר עליו בחלק ב-General של הקוד, כולל הפרמטרים שה-Event מקבל
- כדי לעורר (To Fire) Event יש להשתמש במילה RaiseEvent
- תוכל להעביר פרמטרים ByRef ולהשתמש בערך שלהם לאחר שהמשתמש ב-Class שלך סיים את הטיפול בהם, ובכך לאפשר למשתמש במחלקה להעביר מידע ל-Class

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שלך אודות האופן שבו הוא רוצה שה-Class שלך יתנהג (בדוגמאות שבהמשך עניין זה יובהר)

הצהרה על אובייקטים מסוג המחלקה

- כדי להבחין בין משתנים מסוגים בסיסיים לבין משתנים מסוגים של מחלקות משתמשים במילה אובייקט כדי לתאר מופע (Instance) של מחלקה
- כדי ליצור אובייקטים יש להשתמש במילה New
- כדי לשחרר אובייקט מהזיכרון (יותר מדויק יהיה להגיד להפחית את ה- Reference Counting שלו, אך נדון בכך בהמשך) משתמשים במילה Nothing
- השמת אובייקט מתבצעת באמצעות המילה Set
- עליך להבחין בין האובייקט עצמו (הנמצא בזיכרון) לבין המשתנה באמצעותו אתה ניגש לאובייקט. משתנה זה מאפשר לך להתייחס (Reference) לאובייקט
- אם לאובייקט מסוים יש אירועים ניתן להצהיר עליו תוך שימוש במילה WithEvents כדי להפוך את האירועים שלו לזמינים עבור הלקוח (עניין זה יובהר בדוגמאות שבהמשך)

דוגמה 1

```
Dim prs As New CPerson
```

- בדוגמה זו הוצהר משתנה ייחוס בשם prs מהמחלקה CPerson ואותחל באובייקט חדש בזיכרון באמצעות המילה New

דוגמה 2

```
Dim prs As CPerson
Set prs = New CPerson
```

- שתי הדרכים מאפשרות ליצור אובייקט מסוג CPerson, אך התנהגות האתחול שלהן שונה, כפי שיוסבר מיד
- קיימים הבדלים מהותיים באופן ההתנהגות של אובייקט המוצהר כמו בדוגמה 1 לבין אובייקט המוצהר כמו בדוגמה 2, כפי שמפורט להלן:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- אובייקט המוצהר כמו בדוגמה 1, אינו נוצר בפועל אלא רק כאשר אתה ניגש למאפיין או פונקציה שלו. אובייקט המוצהר כמו בדוגמה 2 נוצר בעת הקריאה ל-New
- אובייקט המוצהר כמו בדוגמה 1 הוא אובייקט בטוח יותר, כיוון שמובטח לך שאם תשחרר את האובייקט מהזיכרון באמצעות Nothing (כפי שתראה בהמשך) ותנסה לגשת אליו שוב מבלי לקרוא ל-New שוב, Visual Basic תיצור את האובייקט עבורך. כך אתה נמנע משגיאות. לעומת זאת, אם הצהרת על אובייקט כמו בדוגמה 2 ולא יצרת אותו בפועל באמצעות המילה New או ששחררת אותו, ותנסה לגשת אליו תקבל Runtime Error
- מההבדל הקודם נובע כי אובייקט המוצהר כמו בדוגמה 1 ירוץ לאט יותר כיוון ש-Visual Basic מוודאת עבורך שהאובייקט קיים בכל פעם שאתה מנסה לגשת לפרוצדורה של האובייקט

דוגמה 3

```
Dim WithEvents obj as CCalendar
```

- בדוגמה זו מוצהר אובייקט מסוג CCalendar. האירועים שהמחלקה CCalendar חושפת יהיו זמינים עבור יישום הלקוח
- לא ניתן להשתמש במילה WithEvents יחד עם המילה New בהצהרה. כלומר, המילה WithEvents זמינה רק עבור הצהרות כמו בדוגמה 2

דוגמה 4

```
Dim rc1 As CRect, rc2 As CRect
Set rc1 = New CRect
Set rc2 = rc1
```

- בדוגמה זו הוצהרו שני משתני ייחוס מסוג CRect. הראשון שבהם אותחל לאובייקט חדש מסוג CRect, ואילו השני אותחל כך שיתייחס לאותו אובייקט CRect ש-rc1 מתייחס אליו

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה 5

```
Set prs = Nothing
```

- פעולה זו מטרתה לשחרר את האובייקט ש-prs מתייחס אליו מהזיכרון (בסעיף הבא תראה שאין זה מדויק)

Reference Counting (ספירת התייחסויות)

- כל ה-Class-ים של Visual Basic הם למעשה אובייקטים של COM
- לכל אובייקט COM יש משתנה פנימי הסופר את מספר ההתייחסויות (Reference Count) אליו. בכל עת שמשתנה ייחוס נוסף מצביע לאובייקט, מונה זה גדל ב-1, ואילו כאשר משתנה מפסיק להתייחס לאובייקט (על ידי קריאה ל-Set...Nothing) מונה ההתייחסויות קטן ב-1
- אובייקט משתחרר מהזיכרון ברגע שה-Reference Count שלו מגיע ל-0
- ב-COM כל אובייקט מממש את ה-Interface ששמו IUnknown. זהו הממשק הבסיסי ביותר ב-COM
- IUnknown כולל שלוש מתודות:
  - **QueryInterface:** מתודה זו מאפשרת קבלת מצביע ל-Interface שהאובייקט מממש
  - **AddRef:** מתודה זו גורמת ל-Reference Count לגדול ב-1 (מתודה זו נקראת אוטומטית עבורך כאשר אתה מבצע השמה למשתנה ייחוס)
  - **Release:** מתודה זו מפחיתה את ה-Reference Count ב-1 (מתודה זו נקראת עבורך אוטומטית כאשר אתה מבטל התייחסות של משתנה לאובייקט מסוים)

דוגמה 6

```
Dim rc1 As CRect, rc2 As CRect
```

- בשורה זו הצהרנו על שני משתני ייחוס לאובייקט מסוג CRect

```
Set rc1 = New CRect
```

- בשורה זו יוצרים אובייקט מסוג CRect בזיכרון וגורמים ל-rc1 להתייחס אליו
- ה-Reference Count של האובייקט יהיה בשלב זה 1

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Set rc2 = rc1
```

- בשורה זו גם משתנה הייחוס rc2 מצביע לאותו אובייקט בזיכרון שנוצר בשורת הקוד הקודמת. לפיכך ה-Reference Count של האובייקט יגדל ב-1

```
Set rc1 = Nothing
```

- בשורת קוד זו מבטלים את התייחסות המשתנה rc1 לאובייקט בזיכרון, לפיכך ה-Reference Count יקטן ב-1 ויהיה שוב 1. כיוון שה-Reference Count עדיין אינו 0 האובייקט לא ישתחרר מהזיכרון

```
Set rc2 = Nothing
```

- בשורת קוד זו מבטלים את התייחסות המשתנה rc2 לאובייקט בזיכרון, לפיכך קטן ה-Reference Count ב-1. בשלב זה ה-reference Count מגיע ל-0 והאובייקט CRect שנוצר בשורת הקוד השנייה משוחרר מהזיכרון ומסיים את חייו.

תרגיל 1 – התוכנית ClassDemo

- בדוגמה זו ניצור מחלקה פשוטה באמצעותה נוכל לעקוב אחר אופן יצירת האובייקטים ושחרורם

1. **צור פרויקט חדש מסוג Standard EXE ושמור אותו:**
   - הקצה לפרויקט את השם ClassDemo ולטופס את השם frmClassDemo
   - שמור את הפרויקט
2. **הוסף Class Module לפרויקט ושמור אותו:**
   - הוסף לפרויקט Class Module תוך שימוש בתפריט Project
   - הקצה ל-Class את השם CDemo על ידי שינוי המאפיין Name של ה-Class
3. **הוסף קוד ל-CDemo:**

```
Public val As Integer
```

   - בחלון הקוד של CDemo בחר ב-Class מתוך התיבה Object (ה-Combo Box השמאלי המופיע בראש חלון הקוד) ומתוך התיבה הימנית בחר ב-Initialize
   - הוסף את הקוד הבא:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Private Sub Class_Initialize()
   val = 0
   MsgBox "Object is now loaded into the memory"
End Sub
```

- קטע קוד זה מאתחל את המשתנה val ברגע שנוצר אובייקט חדש, וכמו כן מציג הודעה באמצעות Message Box
- בחר ב-Terminate והוסף את הקוד הבא:

```
Private Sub Class_Terminate()
   MsgBox "Object is now unloaded from the memory!" & _
          vbCrLf & _
          "val = " & Str(val)
End Sub
```

- קטע קוד זה מציג הודעה בעת שהאובייקט נהרס

4. הוסף אחד עשר CommandButton-ים לטופס והגדר את המאפיינים הבאים:

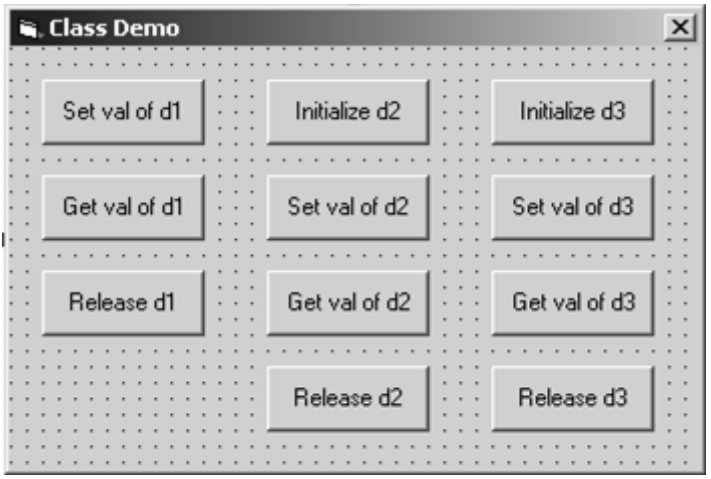
אובייקט	מאפיין	ערך
frmClassDemo	Caption	Class Demo
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdSetValOfD1
	Caption	Set val of d1
Command2	Name	cmdGetValOfD1
	Caption	Get val of d1
Command3	Name	cmdReleaseD1
	Caption	Release d1

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Command4	Name	cmdInitializeD2
	Caption	Initialize d2
Command5	Name	cmdSetValOfD2
	Caption	Set val of d2
Command6	Name	cmdGetValOfD2
	Caption	Get val of d2
Command7	Name	cmdReleaseD2
	Caption	Release d2
Command8	Name	cmdInitializeD3
	Caption	Initialize d3
Command9	Name	cmdSetValOfD3
	Caption	Set val of d3
Command10	Name	cmdGetValOfD3
	Caption	Get val of d3
Command11	Name	cmdReleaseD3
	Caption	Release d3

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- הטופס צריך להיראות כמו זה שבציור הבא:



5. הוסף את הקוד לטופס:

```
Option Explicit

Dim d1 As New CDemo
Dim d2 As CDemo
Dim d3 As CDemo
```

- בשורות הקוד הללו מוצהרים שלושה משתני ייחוס לאובייקטים מסוג CDemo
- ההצהרות מתבצעות בשתי הצורות שהוסברו קודם
- בהמשך (לאחר שנפעיל את התוכנית) תבין כיצד צורת ההצהרה משפיעה על אופן הפעולה של האובייקט

```
Private Sub cmdGetValOfD1_Click()
   MsgBox "d1.val = " & Str(d1.val)
End Sub

Private Sub cmdGetValOfD2_Click()
   MsgBox "d2.val = " & Str(d2.val)
End Sub

Private Sub cmdGetValOfD3_Click()
   MsgBox "d3.val = " & Str(d3.val)
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
End Sub

Private Sub cmdInitializeD2_Click()
   Set d2 = New CDemo
End Sub

Private Sub cmdInitializeD3_Click()
   Set d3 = d2
End Sub

Private Sub cmdReleaseD1_Click()
   Set d1 = Nothing
End Sub

Private Sub cmdReleaseD2_Click()
   Set d2 = Nothing
End Sub

Private Sub cmdReleaseD3_Click()
   Set d3 = Nothing
End Sub

Private Sub cmdSetValOfD1_Click()
   d1.val = 100
End Sub

Private Sub cmdSetValOfD2_Click()
   d2.val = 200
End Sub

Private Sub cmdSetValOfD3_Click()
   d3.val = 300
End Sub
```

6. בחינת התוכנית:

- כעת לאחר שהתוכנית מוכנה נריץ אותה יחד שלב אחר שלב ונבחן את מה שמתרחש וגם את מה שלא מתרחש
- לחץ על הכפתור Start או F5

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- שים לב, הצהרנו על d1 באמצעות New, ולמרות זאת, האירוע Initialize של CDemo לא התרחש עדיין
- לחץ על Set val of d1 או Get val of d1, שים לב שהאירוע Initializeהתרחש כעת כיוון שניסית לגשת לאובייקט
- כעת לחץ על Release d1 פעם אחת
- שים לב כי האובייקט התפרק ואתה מקבל Message Box מהאירוע Terminate של CDemo
- נסה ללחוץ שוב על Release d1, שום דבר לא מתרחש
- כעת לחץ שוב על Set val of d1 או Get val of d1, שוב Visual Basic יוצרת עבורך אובייקט והאירוע Initialize מתעורר שוב
- ***מסקנה:*** הקוד הניגש ל-d1 **בטוח** כיוון שמובטח שאם לא קיים אובייקט בזיכרון, ייווצר אחד כזה. אך בתמורה לכך משלמים בביצועים **מהירים פחות**
- כעת נסה ללחוץ על Set val of d2. אתה אמור לקבל שגיאה כיוון שאין אובייקט שעליו מצביע d2
- עצור את התוכנית והפעל אותה שוב
- לחץ על Initialize d2 כעת נוצר אובייקט עבור d2 והאירוע Initialize שלו מתרחש

שים לב: למרות שעדיין לא ניגשת למאפייני האובייקט הוא נוצר

- הפעל את Set val of d2 ולאחר מכן את Get val of d2, אתה אמור לקבל את הערך 200
- הפעל את Initialize d3. אם תעיין בקוד תבחין ש-d3 מושווה ל-d2, כלומר שניהם מצביעים כעת על אותו אובייקט
- לחץ על Get val of d3 אתה אמור לקבל 200. אכן זהו אותו אובייקט
- לחץ על Set val of d3 ולאחר מכן Get val of d2, תבחין בוודאי שהשתנה הערך ל-300, וניתן לקבל אותו הן מ-d2 והן מ-d3
- כעת לחץ על Release d2, שים לב שאירוע Terminateעדיין לא התרחש, לראייה תוכל ללחוץ על Get val of d3 ותראה כי האובייקט עדיין קיים בזיכרון. כלומר כעת ה-Reference Count של האובייקט הוא 1

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- לחץ על Release d3. ה-Reference Count של האובייקט יורד ל-0 וכתוצאה מכך האובייקט משתחרר מהזיכרון והאירוע Terminate שאתה מבחין בו מתרחש
- אם תנסה כעת לגשת לערך של האובייקט באמצעות d2 או d3 תקבל שגיאה כיוון שאין אובייקט בזיכרון
- כעת נבצע בדיקה אחרונה, הפסק את פעולת התוכנית והרץ אותה שוב
- לחץ על Initialize d2, הודעה בדבר האירוע Initialize צריכה להופיע. לחץ OK
- אנו יודעים כי כעת קיים בזיכרון אובייקט ש-d2 מתייחס אליו
- כעת לחץ שוב על Initialize d2, שים לב ששוב ייוצר אובייקט אשר d2 יהיה אמור להצביע עליו. לראיה אתה מקבל הודעה מאירוע Initialize של CDemo אך השאלה היא מה קורה עם האובייקט ש-d2 התייחס אליו קודם. לאובייקט זה הרי כבר אין מי שמתייחס, כלומר ה-Reference Count שלו יורד ל-0, לפיכך אתה מקבל הודעה בדבר פירוקו של האובייקט מהזיכרון
- אני ממליץ בפניך לבצע עוד מספר ניסויים כדי להבין כיצד נוצרים אובייקטים וכיצד הם מתפרקים. תוכל להשתמש גם בהרצה במצב Break Mode כדי לבחון את התוכנית Step By Step
- ***מסקנה:*** הקוד הניגש ל-d2 ו-d3 **מהיר** יותר, כיוון שאין בדיקה לגבי קיומו של אובייקט בזיכרון, אך קוד זה **פחות בטוח**

תרגיל 2 – התוכנית RectDemo

- בדוגמה זו ניצור Class בשם CRect
- Class זה יאחסן נתונים אודות אורך ורוחב של מלבן
- נוסיף ל-Class שגרות מאפיינים ומתודות לאתחול נתוני המלבן, חישוב היקף המלבן ושטחו

1. צור פרויקט חדש מסוג Standard EXE ושמור אותו:

- הקצה לפרויקט את השם RectDemo ולטופס את השם frmRectDemo
- שמור את הפרויקט

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

2. הוסף Class Module לפרויקט ושמור אותו:

- הוסף לפרויקט Class Module תוך שימוש בתפריט Project
- הקצה ל-Class את השם CRect על ידי שינוי המאפיין Name של ה-Class

3. הוסף קוד ל-CRect:

```
Option Explicit

Private m_dLength As Double
Private m_dWidth As Double
```

- בשורות הקוד הללו אנו מצהירים על שני משתנים, האחד לאחסון אורך המלבן והשני לאחסון רוחב המלבן
- משתנים אלו מוצהרים כ-Private ולפיכך לא ניתן לגשת אליהם ישירות

```
Private Sub Class_Initialize()
   m_dLength = 0#
   m_dWidth = 0#
End Sub
```

- באירוע Initialize נהוג לבצע אתחולים של משתנים ואובייקטים השייכים ל-Class

```
Public Property Let Length(ByVal dLength As Double)
   Debug.Print "Inside Property Let Length"
   If dLength >= 0# Then
      m_dLength = dLength
   End If
End Property
```

- המאפיין Length מאפשר הכנסת ערך למשתנה m_dLength
- נעשית בדיקה כדי לוודא שלא מוכנס ערך לא חוקי (מספר שלילי)
- במקרה שמוכנס ערך חוקי, מעודכן ערכו של m_dLength בהתאם
- כמו כן הוספתי שורת קוד המדפיסה לחלון ה-Immediate כאשר הביצוע של התוכנית עובר לפרוצדורה זו

```
Public Property Get Length() As Double
   Debug.Print "Inside Property Get Length"
   Length = m_dLength
End Property
```

- Property Get מאפשר ללקוח לקרוא את הערך שב-Length

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- גם באירוע זה נכתוב לחלון ה-Immediate כדי לבחון את התרחשות התוכנית

```
Public Property Let Width(dWidth As Double)
   Debug.Print "Inside Property Let Width"
   If dWidth >= 0# Then
      m_dWidth = dWidth
   End If
End Property

Public Property Get Width() As Double
   Debug.Print "Inside Property Get Width"
   Width = m_dWidth
End Property
```

- באופן דומה יוצרים מאפיינים עבור Width

```
Public Function Perimeter() As Double
   Perimeter = 2# * (m_dLength + m_dWidth)
End Function

Public Function Surface() As Double
   Surface = m_dLength * m_dWidth
End Function
```

- שתי הפונקציות Perimeter (היקף) ו-Surface (שטח) מחשבות את היקף ושטח המלבן בהתאמה

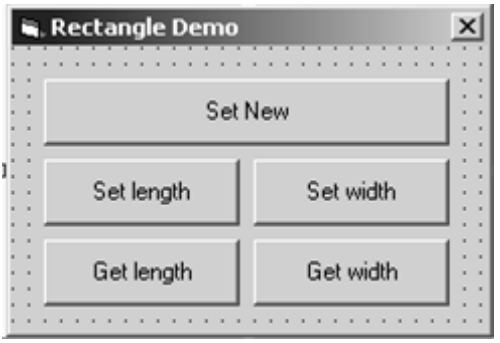
4. הוסף חמישה CommandButton-ים לטופס והגדר את המאפיינים הבאים:

אובייקט	מאפיין	ערך
fmRectDemo	Caption	Rectangle Demo
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdSetNew
	Caption	Set New

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Command2	Name	cmdSetLength
	Caption	Set length
Command3	Name	cmdGetLength
	Caption	Get length
Command4	Name	cmdSetWidth
	Caption	Set width
Command5	Name	cmdGetWidth
	Caption	Get width

- הטופס צריך להיראות כמו בציור הבא:



5. הוסף קוד לטופס:

```
Option Explicit

Dim rc As CRect

Private Sub cmdSetNew_Click()
   Set rc = New CRect
End Sub
```

- בשורת קוד זו יוצרים את האובייקט בזיכרון

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Private Sub cmdGetLength_Click()
   MsgBox "Length = " & Str(rc.Length)
End Sub

Private Sub cmdGetWidth_Click()
   MsgBox "Width = " & Str(rc.Width)
End Sub

Private Sub cmdSetLength_Click()
   rc.Length = Val(InputBox("Enter the length:"))
End Sub

Private Sub cmdSetWidth_Click()
   rc.Width = Val(InputBox("Enter the width:"))
End Sub
```

- ארבע הפרוצדורות הנ"ל מאתחלות את האובייקט rc ומציגות את הערכים
- שים לב לנוחות שבשימוש במאפיינים. אתה יכול להשתמש בסימן = כדי לבצע השמה, ופרוצדורת המאפיין תופעל עבורך, כך שאפשר לרשום rc.Length = 90 במקום לרשום rc.Length(90)

6. בחינת התוכנית:

- התוכנית הנ"ל מציגה מידע בחלון ה-Immediate כך שאם חלון זה אינו מוצג, הצג אותו על ידי בחירתו מהתפריט View
- הרץ את התוכנית
- לחץ על Set New כדי ליצור אובייקט מסוג CRect
- לחץ על Set length והכנס ערך כלשהו. שים לב שבחלון ה-Immediate מוצגת הודעה מתוך הפרוצדורה Property Let Length
- באותו אופן נסה את שאר הלחצנים והבחן בפלט שבחלון ה-Immediate
- אני ממליץ בפניך להריץ את התוכנית Step By Step במצב Break Mode (השתמש ב-F8 לשם כך), ולעקוב אחר ביצוע התוכנית

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגיל 3 – התוכנית EventDemo

- בדוגמה זו ניצור Class שיכלול גם Events ונשתמש ב-Events הללו מתוך תוכנית הלקוח
- תוכנית זו אינה קשה, אם כי ייתכן שזרימת התוכנית תהיה מוזרה לך מעט. אני ממליץ לך להריץ את התוכנית Step By Step כדי לעקוב מקרוב אחר אופן ההתנהלות שלה

1. **צור פרויקט חדש מסוג Standard EXE ושמור אותו:**
   - הקצה לפרויקט את השם EventDemo ולטופס את השם frmEventDemo
2. **הוסף Class Module לפרויקט ושמור אותו:**
   - הוסף לפרויקט Class Module תוך שימוש בתפריט Project
   - הקצה ל-Class את השם CPerson על ידי שינוי המאפיין Name של ה-Class
3. **הוסף קוד ל-CPerson:**

```
Option Explicit

Private m_strName As String
Private m_nAge As Integer
```

- בשורות קוד אלו מוצהרים שני משתנים שאובייקט מסוג CPerson יכיל בתוכו

```
Public Event Changed()
Public Event WillReset(ByRef bCancel As Boolean)
```

- שתי השורות הנ"ל מצהירות על שני Events שאובייקט מסוג CPerson יכול לעורר
- את האירוע Changed נעורר בכל פעם שערכו של אחד מהמשתנים הפנימיים (m_strName או m_nAge) ישונה
- האירוע WillReset יתעורר בכל פעם שנקראת השגרה Reset (רשומה בהמשך)
- תפקיד השגרה Reset הוא לאפס את המשתנים m_strName ו-m_nAge
- באירוע WillReset אנו מעבירים משתנה bCancel באמצעות ייחוס (ByRef). לפיכך, אם המשתמש באובייקט החליט לבטל את פעולת ה-Reset הוא יוכל פשוט לשנות את ערכו של bCancel ל-True (בהמשך כשנכתוב תוכנית המשתמשת ב-CPerson דבר זה יובהר)

```
Private Sub Class_Initialize()
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   m_strName = ""
   m_nAge = 0
End Sub
```

- באירוע Initialize אנו מאפסים את m_nAge ומאתחלים את m_strName למחרוזת ריקה

```
Public Property Let Name(strName As String)
   m_strName = strName
   RaiseEvent Changed
End Property
```

- מאפיין זה משנה את תוכן m_strName ומיד מעורר את האירוע Changed על ידי שימוש במילת המפתח RaiseEvent

```
Public Property Get Name() As String
   Name = m_strName
End Property

Public Property Let Age(ByVal nAge As Integer)
   If nAge >= 0 Then
      m_nAge = nAge
      RaiseEvent Changed
   End If
End Property
```

- מאפיין זה משנה את ערכו של m_nAge בתנאי שהמספר המועבר הוא אי-שלילי
- לאחר שינוי הערך, מעוררים את האירוע Changed

```
Public Property Get Age() As Integer
   Age = m_nAge
End Property

Public Sub Reset()
   Dim bCnl As Boolean
   bCnl = False
   RaiseEvent WillReset(bCnl)
   If bCnl = False Then
      m_strName = ""
      m_nAge = 0
      RaiseEvent Changed
   End If
End Sub
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- תפקיד המתודה Reset הוא לאפס את המשתנים m_strName ו-m_nAge
- תחילה המתודה Reset מעוררת את האירוע WillReset ומעבירה את הפרמטר bCnl (המאותחל ל-False)
- אם המשתמש שלוכד את האירוע החליט לבטל את פעולת ה-Reset, כל שעליו לעשות הוא לשנות את ערכו של bCnl ל-True
- לאחר שלוכד האירוע סיים את הטיפול באירוע, נבחן ערכו של bCnl. אם הוא עדיין False, אזי מאפסים את ערכי המשתנים m_strName ו-m_nAge ומעוררים את האירוע Changed. אם ערכו שונה ל-True לא מבצעים מאומה

4. הוסף שלושה CommandButton-ים לטופס והגדר את המאפיינים הבאים:

אובייקט	מאפיין	ערך
fmEventDemo	Caption	Event Demo
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdSetAge
	Caption	Set Age
Command2	Name	cmdSetName
	Caption	Set Name
Command3	Name	cmdReset
	Caption	Reset
Command4	Name	cmdSetWidth
	Caption	Set width

- הטופס צריך להיראות כמו בציור שלפניך:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il



5. הוסף קוד לטופס:

```
Option Explicit

Dim WithEvents prs As CPerson
```

- שורת קוד זו מצהירה על משתנה ייחוס לאובייקט מסוג CPerson
- נעשה שימוש במילה WithEvents כדי שהאירועים של האובייקט יילכדו בתוך הטופס

```
Private Sub Form_Load()
   Set prs = New CPerson
End Sub
```

- ברגע שהטופס נוצר יוצרים את האובייקט

```
Private Sub Form_Unload(Cancel As Integer)
   Set prs = Nothing
End Sub
```

- באירוע Unload משחררים את האובייקט מהזיכרון

```
Private Sub cmdReset_Click()
   prs.Reset
End Sub
```

- בעת לחיצה על Reset קוראים למתודה Reset של prs

```
Private Sub cmdSetAge_Click()
   prs.Age = Val(InputBox("Enter the age"))
End Sub
```

- בתגובה ללחיצה על Set Age מאתחלים את Age לפי הערך המוזן על ידי המשתמש ב-InputBox

```
Private Sub cmdSetName_Click()
   prs.Name = InputBox("Enter the name")
End Sub
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בתגובה ללחיצה על SetName מאתחלים את Name לפי המחרוזת המוזנת על ידי המשתמש ב-InputBox

```
Private Sub prs_Changed()
   MsgBox "prs has changed. The new values are:" & vbCrLf & _
          "Name = " & prs.Name & vbCrLf & _
          "Age = " & Trim(Str(prs.Age))
End Sub
```

- אירוע זה מתעורר (Fired) על-ידי prs בכל פעם שערך מסוים בתוך האובייקט שונה. בנקודה זו אנו מציגים באמצעות Message Box את הערכים העדכניים של prs

```
Private Sub prs_WillReset(bCancel As Boolean)
   If MsgBox("Do you want to cancel the reset?", vbYesNo) = vbYes
Then
      bCancel = True
   End If
End Sub
```

- אירוע זה מתעורר על ידי prs בכל פעם שמנסים לאפס את prs. בנקודה זו אנו מציגים הודעה המאפשרת למשתמש להחליט האם לבטל את פעולת האיפוס או לא

6. בחינת התוכנית:

- הרץ את התוכנית ובחן את פעולתה
- כאמור, אני ממליץ בפניך להריץ את התוכנית Step By Step באמצעות F8 כדי לבצע מעקב מדויק אחר זרימת התוכנית

מימוש Interface

- Interface הוא החלק הציבורי (Public) של Class מסוים – כלומר החלק הנגיש של המחלקה
- ב-COM נהוג לכתוב ממשקים באמצעות שפת IDL (Interface Definition Language) ולקמפל את הקוד באמצעות הקומפיילר MIDL
- אם אתה מכיר את שפת IDL תוכל להשתמש בה כדי ליצור ממשקים ולממש אותם בכל שפה שתרצה (בתנאי ששפה זו תומכת COM כמובן)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ב-Visual Basic נוכל ליצור Interface-ים פשוט על ידי יצירת Class ללא משתנים וללא מימושים כלשהם לפונקציות. למעשה לכל Class ב-Visual Basic יש ממשק משלו
- Class המעוניין לממש Interface צריך להשתמש במילה Implements
- מימוש Classים הנו מימוש ה-Interface-ים של אותם Class-ים
- Class המממש Interface חייב לממש את כל המתודות והמאפיינים שלו
- אם לא תממש פרוצדורה מסוימת של Interface תקבל שגיאת קומפילציה
- אם הצהרת על משתנים Public ב-Class שאת ה-Interface שלו אתה מממש ב-Class אחר, תקבל עבור כל משתנה כזה פרוצדורות Property Get ו-Property Let
- באמצעות מימוש Interface-ים ניתן ליצור היררכיה של Class-ים וכן ליצור רכיבי תוכנה שיתאימו לתוכנה מוכנה מצד שלישי
- כאשר אובייקט מממש מספר Interface-ים, ניתן להשתמש במשתני ייחוס מכל סוגי ה-Interface-ים הממומשים כדי להתייחס לאובייקט
- בשיעור הבא תלמד כיצד ליצור Add-In ל-Visual Basic על ידי מימוש ממשק מתאים

אבחון תכונות אובייקט בזמן ריצה

- ניתן לבדוק אם אובייקט תומך ב-Interface מסוים ע"י שימוש במילה TypeOf...Is:
- ניתן לבדוק מהו סוג האובייקט המוצבע בפועל על ידי שימוש בפונקציה TypeName. פונקציה זו מחזירה מחרוזת הזהה לשם ה-Class שאליו שייך האובייקט (הכוונה כאן היא לאובייקט המוצבע בפועל ולא לסוג המצביע)
- ניתן לבדוק האם אובייקט אותחל או שהוא שווה ל-Nothing ע"י שימוש במילים Is... Nothing...

דוגמאות

```
If TypeOf shape is IArchive Then
    ...
End If
```

- בשורות קוד אלו נבדק האם shape מממש את ה-Interface ששמו IArchive, אם כן נכנסים לתוך בלוק הביצוע של ה-If

```
MsgBox TypeName(obj)
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- שורת קוד זו מציגה את שם המחלקה אליה שייך האובייקט המוצבע ע"י obj

```
If obj Is Nothing Then Exit Sub
```

- שורת קוד זו גורמת ליציאה מרוטינה אם obj אינו מאותחל (כלומר הוא Nothing)
- שים לב שכדי לשלול את התנאי עליך לרשום Not בתחילת התנאי. כלומר:

```
If Not TypeOf shape is IArchive Then...
```

וכן:

```
If Not obj Is Nothing Then Exit Sub
```

תרגיל 4 – התוכנית ImplementsDemo

- בדוגמה זו ניצור Class עם מספר פרוצדורות בעלות מימוש ריק כלומר Interface, ונממש אותו ב-Class אחר

1. צור פרויקט חדש מסוג Standard EXE ושמור אותו:

- הקצה לפרויקט את השם ImplementsDemo ולטופס את השם frmImplementsDemo
- שמור את הפרויקט

2. הוסף שלושה Class Module-ים לפרויקט ושמור אותם:

- הקצה ל-Class-ים את השמות: IShape, CCircle ו-CRect
- שמור את ה-Class Module-ים

3. הוסף קוד ל-IShape:

```
Option Explicit

Public Function Perimeter() As Double
   ' Empty inside interface
End Function

Public Function Surface() As Double
   ' Empty inside interface
End Function
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בשורות קוד אלו מוצהרות שתי פונקציות, Perimeter ו-Surface
- שים לב כי אין מימוש לפונקציות (בתוך כל פונקציה רשומה הערה!)

4. הוסף קוד ל-CCircle:

```
Option Explicit

Implements IShape
```

- בשורת הקוד הזו אנו מבקשים לממש את ה-Interface של IShape
- תשים לב כי ב-Combo Box השמאלי שבחלון הקוד (Object) מופיע כעת IShape

```
Public m_dRadius As Double
```

- בשורת קוד זו מצהירים על משתנה שיאחסן את רדיוס המעגל
- שים לב כי לצורך ההדגמה המשתנה מוצהר כ-Public אך ברור לך שכדאי להצהיר עליו כ-Private וליצור עבורו פרוצדורות Property Get ו-Property Let מתאימות

```
Private Function IShape_Perimeter() As Double
   IShape_Perimeter = 2 * 3.14 * m_dRadius
End Function
```

- המימוש של Perimeter עבור Circle הוא כמובן $P = 2\pi r$

```
Private Function IShape_Surface() As Double
   IShape_Surface = 3.14 * m_dRadius * m_dRadius
End Function
```

- המימוש של Surface עבור Circle הוא $S = \pi r^2$

5. הוסף קוד ל-CRect:

```
Option Explicit

Implements IShape

Public m_dLength As Double
Public m_dWidth As Double

Private Function IShape_Perimeter() As Double
   IShape_Perimeter = 2 * (m_dLength + m_dWidth)
End Function
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Private Function IShape_Surface() As Double
   IShape_Surface = m_dLength * m_dWidth
End Function
```

- כמו ב-CCircle גם ב-CRect ממומש ה-Interface של IShape, תוך ביצוע החישובים המתאימים למלבן

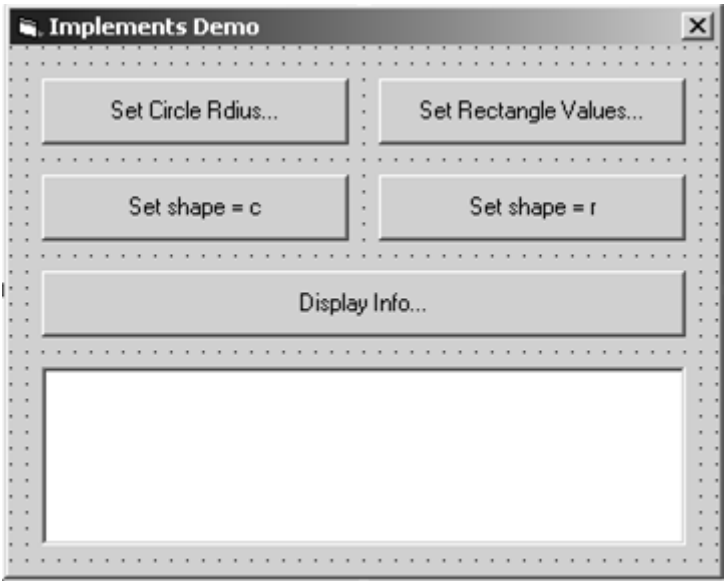
6. הוסף לטופס חמישה CommandButton-ים ו-TextBox אחד, והגדר את המאפיינים הבאים:

אובייקט	מאפיין	ערך
frmImplementsDemo	Caption	Implements Demo
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdSetCircle
	Caption	Set Circle Radius...
Command2	Name	cmdSetShapeIsCircle
	Caption	Set shape = c
Command3	Name	cmdSetRectangleValues
	Caption	Set Rectangle Values...
Command4	Name	cmdSetShapeIsRect
	Caption	Set shape = r
Command5	Name	cmdDisplayInfo
	Caption	Display Info...
Text1	Name	txtInfo

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Text	[ריק]
MultiLine	True

- הטופס אמור להיראות כמו בציור שלפניך:



7. הוסף קוד לטופס:

```
Option Explicit

Dim c As New CCircle
Dim r As New CRect
Dim shape As IShape
```

- השורות הנ"ל מצהירות על אובייקט מסוג CRect, אובייקט מסוג CCircle ומשתנה ייחוס ל-IShape

```
Private Sub cmdSetCircle_Click()
   c.m_dRadius = Val(InputBox("Enter the radius of the circle:"))
End Sub

Private Sub cmdSetRectangleValues_Click()
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   r.m_dLength = Val(InputBox("Enter the length of the
recangle:"))
   r.m_dWidth = Val(InputBox("Enter the width of the
rectangle:"))
End Sub
```

- שורות הקוד הנ"ל מאפשרות אתחול של המעגל c והמלבן r

```
Private Sub cmdSetShapeIsCircle_Click()
   Set shape = c
End Sub

Private Sub cmdSetShapeIsRect_Click()
   Set shape = r
End Sub
```

- בשורות הקוד הללו קובעים לאיזה מבין הצורות (c או r) יצביע המשתנה shape
- שים לב כי אין כל בעיה ש-shape יצביע על אובייקט מסוג CCircle או מסוג CRect כיוון שאובייקטים אלו ממשים את IShape
- עם זאת – shape לא יוכל להפעיל פונקציות שאינן שייכות ל-IShape (נדון בכך בסעיף הבא "Early Binding לעומת Late Binding")

```
Private Sub cmdDisplayInfo_Click()
   If shape Is Nothing Then Exit Sub
   txtInfo.Text = TypeName(shape) & "..." & vbCrLf
   txtInfo.Text = txtInfo.Text & String(8, " ") & _
                  "Perimeter = " & Str(shape.Perimeter()) &
vbCrLf
   txtInfo.Text = txtInfo.Text & String(8, " ") & _
                  "Surface = " & Str(shape.Surface()) & vbCrLf
End Sub
```

- בתגובה ללחיצה על Display Info... מתבצעות שורות הקוד הנ"ל
- תחילה נבדק שהמשתנה shape מאותחל. אם shape אינו מאותחל יוצאים מהרוטינה
- לאחר מכן מציגים את שם סוג האובייקט המוצבע ע"י shape. עושים זאת באמצעות הפונקציה TypeName
- מוסיפים קריאות ל-Perimeter ו-Surface כדי להציג את היקף ושטח הצורה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

8. **בחינת התוכנית:**

- הרץ את התוכנית ובדוק את אופן פעולתה

Early Binding לעומת Late Binding

- ב-Early Binding נקבעים בזמן הקימפול הממשק והאופן שבו מפעילים את המתודות, בדרך כלל באמצעות מיקומן בטבלת הפונקציות (VTable), ללא צורך בחיפוש שם המתודה בזמן ריצה.
- קישור מסוג זה קרוי VTBL Binding והוא המהיר ביותר ב-COM
- בדרך כלל אתה משתמש בסוג זה של קישור בתוכניות Visual Basic
- Late Binding היא האפשרות לגשת למאפיין או מתודה של אובייקט שאין ודאות בזמן הפיתוח שהם נתמכים ע"י האובייקט. משמעות הדבר היא שבזמן ריצת התוכנית יהיה על התוכנית לגלות אם באותו אובייקט קיימת המתודה או קיים המאפיין שאליו רוצים לגשת
- פונקציונליות זו אפשרית על ידי שימוש בממשק IDispatch מ-COM או ע"י שימוש ב-Type Library וזיהויו בזמן ריצה
- כל האובייקטים הנוצרים ב-Visual Basic מממשים אוטומטית את הממשק IDispatch.
- הממשק IDispatch מאפשר לגשת למאפיינים ולמתודות של אובייקטים באמצעות Late Binding. ב-Visual Basic זה נעשה תוך שימוש במשתנה ייחוס מסוג Object.

> **שים לב:** כל אובייקט COM חייב לממש את הממשק IUnknown, שהוא הממשק הבסיסי של COM. לעומת זאת, המימוש של IDispatch, המאפשר Late Binding, אינו חובה. בעוד שאובייקטים הנוצרים ב-Visual Basic מממשים את IDispatch באופן אוטומטי, לא כל אובייקט COM תומך בממשק זה.

- Late Binding הרבה יותר גמיש מבחינת אפשרויות התכנות מאשר Early Binding. עם זאת, Late Binding איטי ביותר ביחס ל-Early Binding

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בעת קריאה למתודה או מאפיין בזמן ריצה תוך שימוש ב-Object מתבצעות מספר הכנות לפני שהפונקציה מופעלת. פעולות אלה בודקות את קיומה של המתודה או קיומו של המאפיין הנקרא. ניקח לדוגמה את המשפטים:

```
Dim obj As Object
...
...
obj.SetValues("Today is your day", 57)
```

- שלבי הביצוע הם:
  - הכנת כל המחרוזות הטקסטואליות שיש לזהות: במקרה זה המחרוזת שיש לזהות היא SetValues – כלומר יש לבדוק האם האובייקט שאליו מצביע obj אכן כולל פונקציה כזו
  - קריאה ל-IDispatch::GetIDsOfNames(). קריאה זו מטרתה לאחזר את מספרי הזיהוי (DISPIDs) של שם המתודה והפרמטרים שיש להעביר לה: במקרה שלנו יוחזרו מספרי זיהוי עבור השם SetValues ועבור הפרמטרים שיש להעביר לה
  - כל מספרי הזיהוי (DISPIDs) יוחזרו
  - קריאה ל-IDispatch::Invoke(). קריאה זו מטרתה להפעיל (סוף סוף) את המתודה לאחר שזיהינו שאכן היא נתמכת, וכל הפרמטרים שלה הוכנו.
  - הערך המוחזר של הפונקציה (HRESULT) מוחזר מ-IDispatch::Invoke() לקוד הקורא
- תחשוב על כך שהפעולות הללו מתבצעות עבור כל קריאה למתודה, כאשר היא מתבצעת באמצעות IDispatch. ללא ספק זוהי תקורה (Overhead) גבוהה למדי עבור קריאה בודדת. תחשוב כמה זמן היית ממתין לתשובה לו הקטע הנ"ל היה מתבצע בין שני Process-ים נפרדים או בין שני מחשבים ברחבי הרשת
- בדוגמה הבאה תבחן במושגים של זמן את ההבדלים בין Late Binding ל-Early Binding

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגיל 5 – התוכנית Analyzer

1. **צור פרויקט חדש מסוג Standard EXE ושמור אותו:**
   - הקצה לפרויקט את השם Analyzer ולטופס את השם frmAnalyzer
   - שמור את הפרויקט
2. **הוסף Class Module לפרויקט:**
   - הקצה את השם CTest ל-Class Module
   - שמור את הפרויקט שוב
3. **הוסף קוד ל-CTest:**

```
Option Explicit

Private m_lVal As Long

Public Property Let Val(lVal As Long)
   m_lVal = lVal
End Property
```

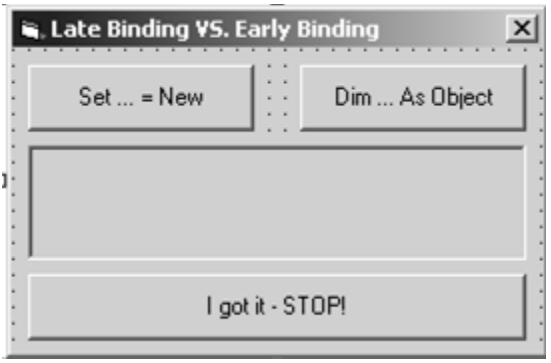
   - זוהי מחלקה פשוטה עם מאפיין אחד המאפשר להזין ערך למשתנה Val
4. **הוסף לטופס שלושה CommandButton-ים ו-TextBox אחד והגדר את המאפיינים הבאים:**

אובייקט	מאפיין	ערך
frmAnalyzer	Caption	Late Binding VS. Early Binding
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdSetNew

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

	Caption	Set ... = New
Command2	Name	cmdUseObject
	Caption	Dim ... As Object
Command3	Name	cmdStop
	Caption	I got it - STOP!
Text1	Name	txtTimer
	Background	Button Face
	Locked	True
	Text	[ריק]
	MultiLine	True

- הטופס אמור להיראות כמו בציור שלפניך:



5. הוסף קוד לטופס:

```
Option Explicit

Private Sub cmdSetNew_Click()
   Dim cls As CTest
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   Dim lVal As Long, i As Integer
   Dim sTime As Single
   Set cls = New CTest
   sTime = Timer
   For i = 1 To 1000
      txtTimer.Text = "As CTest... " & vbCrLf & _
                      "i = " & Str(i) & vbCrLf & _
                      "Time elapsed:" & Str(Timer - sTime) & "
sec"
      DoEvents
      For lVal = 1 To 10000
         cls.Val = lVal
      Next
   Next
End Sub
```

- בקטע קוד זה מריצים שתי לולאות מקוננות
- בתוך הלולאה הפנימית ניגשים למאפיין של האובייקט cls
- לפני הכניסה ללולאה החיצונית נמדד הזמן ולפני כל איטרציה של הלולאה הפנימית מוצג ההפרש בשניות מאז הכניסה ללולאה החיצונית
- בפעולת ההצגה מוצגים מספר האיטרציה והזמן שחלף, לצד הכיתוב As CTest

```
Private Sub cmdUseObject_Click()
   Dim cls As Object
   Dim lVal As Long, i As Integer
   Dim sTime As Single
   Set cls = New CTest
   sTime = Timer
   For i = 1 To 1000
      txtTimer.Text = "As Object..." & vbCrLf & _
                      "i = " & Str(i) & vbCrLf & _
                      "Time elapsed:" & Str(Timer - sTime) & "
sec"
      DoEvents
      For lVal = 1 To 10000
         cls.Val = lVal
      Next
   Next
End Sub
```

- קטע קוד זה זהה לקודם, אלא שכאן מוצהר האובייקט כ-Object

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- במקרה זה מופעלת הגישה למאפיין Val באמצעות Late Binding

```
Private Sub cmdStop_Click()
   Unload Me
End Sub

Private Sub Form_Unload(Cancel As Integer)
   End
End Sub
```

6. בחינת התוכנית:

- הרץ את התוכנית והשווה את הביצועים של Early Binding לעומת Late Binding
- תסכים איתי שהתוצאות מרשימות. אצלי במחשב (Pentium III 733 MHz) השימוש ב-Early Binding (שימוש במשתנה ייחוס מסוג CTest) הסתכם ב-4.41 שניות, ואילו הפעלה באמצעות Late Binding (שימוש במשתנה ייחוס מסוג Object) הסתכמה ב-77.21 שניות.

הגדרת תכונות של פרוצדורות ב-Interface

- בעת יצירת Class ב-Visual Basic נוצר עבורו באופן אוטומטי Interface
- לפרוצדורות ב-Interface ישנן הגדרות ותכונות נוספות
- הגדרה מותאמת אישית של תכונות אובייקטים ב-Visual Basic מהווה לעתים תכופות מטלה שחובה לבצע אותה על מנת לגרום לאובייקט לפעול בצורה תקינה
- בסעיף זה תלמד כיצד ניתן לצפות בתכונות ה-Interface, וכיצד ניתן להגדיר את התכונות הללו ב-Visual Basic
- חלק מהתכונות יילמדו בשיעורים הבאים

צפייה בתכונות באמצעות Object Browser

- כדי לצפות בפרוצדורות והתכונות שלהן ניתן להשתמש ב-Object Browser

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגיל 6 – צפייה באובייקטים באמצעות ה-Object Browser

◄ תרגיל זה נטול קוד. מטרת התרגיל היא להכיר לך את חלון ה-Object Browser

1. **פתח את ה-Object Browser:**
   - מתוך התפריט View בחר ב-Object Browser. לחילופין, לחץ על F2
2. **בחר אובייקט לבחינה:**
   - לצורך התרגול (וגם בגלל שהנושא הבא הוא Collections) נבחר לצפות באובייקט Collection
   - מתוך הרשימה Classes בחר ב-Collection (אם אינך מוצא את Collection, ודא שב-Combo Box העליון בחרת ב-<All Libraries> או ב-VBA)
3. **בחן את Collection:**
   - כפי שאתה רואה, בפנל הימני מוצגים המאפיינים והמתודות של Collection
   - סמן את אחת הפרוצדורות. תבחין מיד כי בתחתית החלון מוצגת אינפורמציה אודות הפרוצדורה הנבחרת. אינפורמציה זו כוללת את סוג הפרוצדורה (Sub, Function, Property או Event), הסבר קצר, ולינק ל-Class המכיל (במקרה זה VBA.Collection)
4. **זהה את פרוצדורת ברירת המחדל:**
   - שים לב כי ליד Item ישנו עיגול כחול. זהו סימן לכך שהפונקציה Item היא פרוצדורת ברירת המחדל עבור Collection
   - פרוצדורת ברירת מחדל מופעלת באופן אוטומטי אם לא ציינת במפורש שום פרוצדורה להפעלה
5. **הצג פרוצדורות נסתרות:**
   - לחץ לחיצה ימנית במקום כלשהו ובחר ב-Show Hidden Members
   - תבחין כי לרשימת הפרוצדורות של Collection נוסף חבר חדש ושמו _NewEnum
   - על משמעות _NewEnum תלמד בסעיף Collections

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

צפייה באמצעות היישום OLE VIEW

- לא כל התכונות של פרוצדורה מוצגות ב-Object Browser
- כדי לצפות בכל הפרטים אודות Interface מסוים, אפשר להשתמש ביישום OLE View. יישום זה מותקן יחד עם חבילת Visual Studio. תוכל לפתוח אותו מתוך תת התפריט Microsoft Visual Studio 6.0 Tools תחת Microsoft Visual Studio 6.0

תרגיל 7 – OLE View לבעלי לב חזק בלבד

- אם אתה בעל לב חזק, בטח תרצה לבחון את Collection גם מתוך OLE View

1. **פתח את OLE View:**
   - OLE View נמצא במסלול הבא (אלא אם התקנת את Visual Studio במסלול שונה מאשר ברירת המחדל):

   Start-->Programs--> Microsoft Visual Studio 6.0 --> Microsoft Visual Studio 6.0 Tools --> OLE View

2. **פתח את ה-Type Library:**
   - מהתפריט File בחר ב-View TypeLib...
   - עבור לספריית VB98 תחת ספריית ההתקנה של Visual Studio (בדרך כלל זהו המסלול:
     C:\Program Files\Microsoft Visual Studio\VB98\)
   - בחר בקובץ VBA6.DLL. האובייקט Collection נמצא בספריית VBA.
   - OLE View ישתהה מעט בזמן שהוא קורא את נתוני הספרייה

3. **בחר ב-Interface של האובייקט Collection:**
   - בתחתית הפנל השמאלי לחץ על לחצן הפלוס (+) שליד הפריט interface _Collection
   - תוכל להבחין שהמתודות Item, Add, Count, Remove ו-NewEnum_ מופיעות

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- מעיון בפרוצדורות תבחין כי לכל פרוצדורה יש מספר זיהוי (id) הרשום בהגדרת ה-Interface
- אם תעיין במתודה NewEnum_ תבחין בכך שמספר הזיהוי שלה הוא 0xFFFFFFFC (שהוא 4- בבסיס עשרוני) פרט זה ישמש אותנו כשניצור Collection מותאם אישית

4. **עיין ב-Interface-ים ממומשים על-ידי Collection:**

- כפי שזכור לך מהדיון על Reference Counting ועל Late Binding, אמרתי אז ש-IUnknown הוא ה-Interface הבסיסי ביותר ב-COM ואילו IDispatch מאפשר Late Binding עבור רכיבי ActiveX
- אם תלחץ על Inherited Interfaces תחת הענף הנוכחי, תגלה את IDispatch חבוי עם המתודות המוכרות לנו מהדיון אודות Late Binding (GetIDsOfNames, Invoke ועוד.)
- אם תלחץ שוב על Inherited Interfaces שתחת IDispatch תגלה את IUnknown עם שלוש המתודות שלו
  (QueryInterface, AddRef ו-Release). ואכן זהו ה-Interface הבסיסי ביותר.
  IUnknown אינו נורש מאף Interface אחר

5. **אם אתה מרגיש שאתה צריך קפה – זה הזמן!**

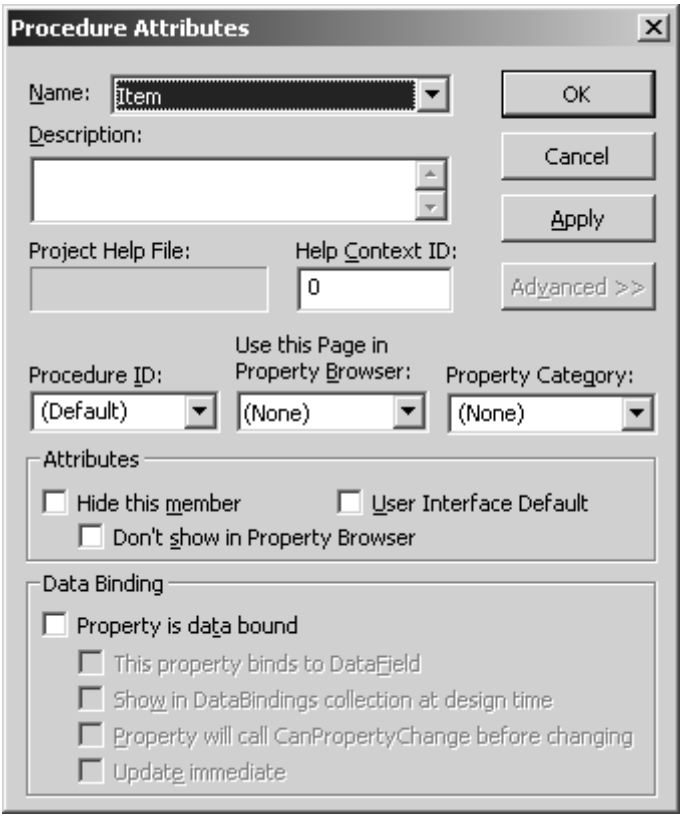
הגדרת תכונות

- עד כה ראינו איפה צופים בתכונות של Interface
- תכונות פרוצדורה של מחלקה נקבעות בשני מקומות חשובים:

- הגדרת הפרוצדורה כפי שמופיעה בקוד קובעת:
  - מהו סוג הפרוצדורה (Sub, Function, Property Get, Property Let או Event)
  - מהם הפרמטרים המועברים לפרוצדורה (אם בכלל)
  - מהו הערך המוחזר מהפונקציה
- תיבת הדו-שיח ה-Procedure Attributes:
  - תיבת הדו-שיח ה-Procedure Attributes נמצאת תחת התפריט Tools

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- תיבת דו-שיח זו מאפשרת לשלוט במאפיינים רבים שאת חלקם נסקור רק בשיעורים הבאים



- **Name:** שם הפרוצדורה
- **Description:** תיאור קצר של הפרוצדורה
- **Project Help:** שם קובץ העזרה של הפרויקט (ניתן להגדרה מתוך חלון ה-Project Properties)
- **Help Context ID:** מגדיר את מספר הקשר העזרה בתוך קובץ העזרה
- **Procedure ID:** (מזהה שגרה): לכל שגרה ב-Interface יש מספר זיהוי. מספר זה משפיע על אופן התנהגות השגרה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Use This Page In Property Browser:** מגדיר את הדף שייפתח מבין דפי המאפיינים עבור מאפיין זה
- **Property Category:** הקטגוריה אליה שייכת הפרוצדורה (שים לב כי בחלון המאפיינים יש אפשרות להציג מאפיינים לפי קטגוריות)
- **Hide This Member:** מסתיר את הפרוצדורה ממשתמשי המחלקה
- **Don't Show In Property Browser:** לא מציג את המאפיין בחלון המאפיינים
- **User Interface Default:** האם הפרוצדורה תהיה פרוצדורת ברירת המחדל בחלון המאפיינים
- **Data Binding:** האם הפרוצדורה יכולה להיות קשורה לנתונים חיצוניים. נושא זה יידון בהמשך הספר

- בהמשך שיעור זה תשתמש בתיבת דו-שיח זו כדי להגדיר Procedure ID (מזהה שגרה) עבור פרוצדורה שתכתוב
- ביתר ההגדרות שצוינו נשתמש בשיעורים הבאים

Collections (אוספים)

- Collection הוא מבנה נתונים דינמי
- Collection הוא Class מוכן של Visual Basic, כלומר, מבנה שניתן להוסיף ולהסיר ממנו פריטים במהלך ריצת התוכנית
- לפריטים באוסף יש אינדקסים הרצים מ-1 עד Count (כאשר Count הוא גודל ה-Collection)
- Collection יכול להכיל פריטים מסוגים שונים, לרבות אובייקטים וערכים פשוטים כגון מספרים ומחרוזות.
- כיוון שניתן להפנות לכל סוגי האובייקטים ב-Visual Basic באמצעות Object, ניתן להוסיף את כולם ל-Collection.
- אם אתה מוסיף אובייקטים מסוגים שונים ל-Collection תוכל להשתמש בזיהוי סוג האובייקט בזמן ריצה כדי לבצע פעולות האופייניות רק לסוג מסוים
- ניתן לבצע סריקה על אובייקטים ב-Collection בשתי צורות:
  - לולאת For רגילה הרצה מ-1 עד Count

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- לולאת For Each...In...Next. סוג זה של לולאה נוסף לשפה כדי להקל את העבודה עם אוספים

תחביר

- תחביר ההצהרה על Collection הוא:

```
Dim coll As New Collection
```

- אפשר להצהיר ללא המילה New ואז ליצור את ה-Collection מאוחר יותר

- ל-Collection יש את המאפיינים והמתודות הבאות:

- **Add:** מוסיף פריט ל-Collection.

  - תחביר:

```
coll.Add Item, [Key], [Before], [After]
```

  - **Item:** זהו הפריט שיש להוסיף ל-Collection
  - **Key:** מחרוזת ייחודית שניתן להצמיד לפריט. מאוחר יותר ניתן יהיה לאתר את הפריט באמצעות מחרוזת זו
  - **Before:** מוסיף את הפריט לפני הפריט שהאינדקס שלו או המפתח (Key) שלו מצוין ב-Before. לא ניתן לציין Beforeיחד עם After
  - **After:** מוסיף את הפריט אחרי הפריט שהאינדקס שלו או המפתח (Key) שלו מצוין ב-After. לא ניתן להשתמש ב-After יחד עם Before

- **Remove:** מסיר פריט מה-Collection:

  - תחביר:

```
coll.Remove Index
```

  - **Index:** מציין את האינדקס או המפתח (Key) של הפריט שיש להסיר

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Item:** מחזיר מצביע לפריט מסוים ב-Collection
  - תחביר:

```
coll.Item (Index)
```

  - **Index:** מציין את האינדקס או המפתח (Key) של הפריט שיש להחזיר
  - Item הוא מאפיין ברירת המחדל. לפיכך ניתן לפנות לפריט גם באופן הבא:

```
coll(Index)
```

- **Count:** מחזיר את מספר הפריטים ב-Collection
  - תחביר:

```
coll.Count
```

הלולאה For each ..In .. Next

- לולאה זו מיועדת לסריקת אובייקטים מתוך אוסף
- תחביר הלולאה הוא כדלקמן:

```
For Each obj In Coll
   obj.Method1...
   obj.Method2...
Next
```

- בכל איטרציה של הלולאה obj מתייחס לאובייקט הנוכחי ב-Coll

תרגיל 8 – התוכנית ShapeColl1

- התוכנית ShapeColl1 תאפשר למשתמש להוסיף מלבנים או עיגולים
- האובייקטים שייוצרו יצורפו ל-Collection שיוצהר לשם כך
- לחיצה על List Items תציג את כל הפריטים הנמצאים באוסף
- סריקת האוסף נעשית באמצעות הלולאה For Each ... In ... Next

1. **צור פרויקט חדש מסוג Standard EXE ושמור אותו:**

- הקצה לפרויקט את השם ShapeColl1 ולטופס את השם frmTest
- שמור את הפרויקט

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

2. הוסף שני Class Module-ים לפרויקט:

- הקצה את השמות CCircle ו-CRect למחלקות
- שמור את הפרויקט שוב

3. הוסף את הקוד הבא ל-CCircle:

```
Option Explicit

Private Const PI As Double = 3.14

Private m_dRadius As Double

Public Property Get Radius() As Double
   Radius = m_dRadius
End Property

Public Property Let Radius(dRad As Double)
   If dRad >= 0 Then m_dRadius = dRad
End Property
Public Function Surface() As Double
   Surface = PI * m_dRadius * m_dRadius
End Function

Public Function Perimeter() As Double
   Perimeter = 2 * PI * m_dRadius
End Function
```

4. הוסף את הקוד הבא ל-CRect:

```
Option Explicit

Private m_dLength As Double
Private m_dWidth As Double

Public Property Get Length() As Double
   Length = m_dLength
End Property

Public Property Let Length(dLen As Double)
   If dLen >= 0 Then m_dLength = dLen
End Property
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Public Property Get Width() As Double
   Width = m_dWidth
End Property

Public Property Let Width(dWid As Double)
   If dWid >= 0 Then m_dWidth = dWid
End Property

Public Function Surface() As Double
   Surface = m_dLength * m_dWidth
End Function

Public Function Perimeter() As Double
   Perimeter = 2 * (m_dLength + m_dWidth)
End Function
```

5. **הוסף ארבעה CommandButton-ים ו-TextBox אחד והגדר את המאפיינים הבאים:**

אובייקט	מאפיין	ערך
frmTest	Caption	Shape Collection Demo 1
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdAddRect
	Caption	Add Rectangle
Command2	Name	cmdAddCircle
	Caption	Add Circle
Command3	Name	cmdListItemsMethod1
	Caption	List items - Method 1
Command4	Name	cmdListItemsMethod2

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

	Caption	List items - Method 2
Text1	Name	txtColl
	Background	ButtonFace
	Locked	True
	Text	[ריק]
	MultiLine	True

6. הוסף את הקוד הבא לטופס:

```
Option Explicit

Public shpColl As Collection
```

- בשורת קוד זו מוצהר מצביע ל-Collection בשם shpColl

```
Private Sub Form_Load()
   Set shpColl = New Collection
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Set shpColl = Nothing
End Sub
```

- הקוד הנ"ל יוצר אובייקט Collection חדש עם טעינת הטופס, ומשחרר את ההתייחסות ל-Collection עם התפרקות הטופס

```
Private Sub cmdAddCircle_Click()
   Dim cr As CCircle
   Set cr = New CCircle
   cr.Radius = Val(InputBox("Enter the radius for the circle:"))
   shpColl.Add cr
   Set cr = Nothing
End Sub

Private Sub cmdAddRect_Click()
   Dim rc As CRect
   Set rc = New CRect
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   rc.Length = Val(InputBox("Enter the length of the
rectangle:"))
   rc.Width = Val(InputBox("Enter the width of the rectangle:"))
   shpColl.Add rc
   Set rc = Nothing
End Sub
```

- בשורות הקוד הנ"ל יוצרים עיגול או מלבן, בהתאם ללחיצה של המשתמש
- מאתחלים את האובייקט שנוצר על ידי שימוש בפונקציה InputBox
- מוסיפים את האובייקט ל-shpColl

```
Private Sub cmdListItemsMethod1_Click()
   Dim obj As Object
   txtColl.Text = ""
   On Error Resume Next
   For Each obj In shpColl
      txtColl.Text = txtColl.Text & TypeName(obj) & vbCrLf
      txtColl.Text = txtColl.Text & String(8, " ") & _
                  "Length = " & obj.Length & vbCrLf
      txtColl.Text = txtColl.Text & String(8, " ") & _
                  "Width = " & obj.Width & vbCrLf
      txtColl.Text = txtColl.Text & String(8, " ") & _
                  "Radius = " & obj.Radius & vbCrLf
      txtColl.Text = txtColl.Text & String(8, " ") & _
                  "Surface = " & obj.Surface & vbCrLf
      txtColl.Text = txtColl.Text & String(8, " ") & _
                  "Perimeter = " & obj.Perimeter & vbCrLf
   Next
End Sub
```

- בשורות קוד אלו מבצעים סריקה של האובייקטים ב-shpColl
- עבור כל אובייקט מציגים (יותר מדויק להגיד "מנסים להציג") את Length, Width, Radius, Surface ו-Perimeter
- כיוון שלאובייקטים מסוג CRect אין את המאפיין Radius, ולאובייקטים מסוג CCircle אין את המאפיינים Length ו-Width, קוראים ל-On Error Resume Next כדי להעביר את הביצוע לשורה הבאה אם מתרחשת שגיאה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- שיטה זו משתמשת ב-Late Binding כדי להציג פרטים, וכפי שראית בתרגיל הקודם, ביצועי Late Binding הם גרועים. חשוב כמה זמן הייתה נמשכת פעולת הצגת הפרטים אם נניח שהאוסף מכיל 10000 צורות!

```
Private Sub cmdListItemsMethod2_Click()
   Dim obj As Object
   Dim rc As CRect
   Dim cr As CCircle
   txtColl.Text = ""
   On Error Resume Next
   For Each obj In shpColl
      If TypeOf obj Is CRect Then
         Set rc = obj
         txtColl.Text = txtColl.Text & "CRect" & vbCrLf
         txtColl.Text = txtColl.Text & String(8, " ") & _
               "Length = " & rc.Length & vbCrLf
         txtColl.Text = txtColl.Text & String(8, " ") & _
               "Width = " & rc.Width & vbCrLf
         txtColl.Text = txtColl.Text & String(8, " ") & _
               "Surface = " & rc.Surface & vbCrLf
         txtColl.Text = txtColl.Text & String(8, " ") & _
               "Perimeter = " & rc.Perimeter & vbCrLf
      Else
         Set cr = obj
         txtColl.Text = txtColl.Text & "CCircle" & vbCrLf
         txtColl.Text = txtColl.Text & String(8, " ") & _
               "Radius = " & cr.Radius & vbCrLf
         txtColl.Text = txtColl.Text & String(8, " ") & _
               "Surface = " & cr.Surface & vbCrLf
         txtColl.Text = txtColl.Text & String(8, " ") & _
               "Perimeter = " & cr.Perimeter & vbCrLf
      End If
   Next
End Sub
```

- גם כאן מבצעים סריקה על האובייקטים ב-shpColl
- לעומת השיטה הקודמת, כאן נבחן סוג האובייקט ולאחר מכן הוא מושם למשתנה מסוג CRect או מסוג CCircle. כיוון שכך, הגישה לאובייקטים נעשית דרך ה- Type Library שלהם וכתוצאה מקבלים ביצועים הרבה יותר טובים.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

7. **בחינת התוכנית:**
   - הרץ את התוכנית ובדוק את פעולתה

מחלקות Collection מותאמות אישית

- כפי שראית, Collection מקבל את הסוג Object
- ניתן ליצור מחלקות אוסף מותאמות, כך שמחלקות אלו יקבלו נתונים מסוג מסוים או מנגזרותיו
- היתרון ב-Collection מותאם אישית הוא בכך שהאובייקטים המוחזרים ממנו הם מסוג מסוים. לפיכך ניתן להשתמש ב-Early Binding
- משתמשים ב-Collection-ים גם כדי ליצור היררכיית עצמים
- היררכיית ADO (שתכיר בהמשך הספר) היא דוגמה לצירוף של Class-ים ו-Collection-ים היוצרים היררכיה אחת
- בתרגיל הבא תיצור Collection משלך. בכל מקרה דע שתוכל להשתמש ב- VB Class Builder (אתה מציג אותו ע"י בחירת Add Class Module מתפריט Project) כדי ליצור היררכיית עצמים ואוספים
- שים לב שבאוספים הנוצרים ע"י VB Class Builder המתודה Add יוצרת את האובייקטים. בתרגיל שלפנינו המתודה Add מקבלת את האובייקט בדיוק כמו ב-Collection רגיל
- הלולאה For Each...In...Next המוכרת לך, פועלת על Collection כיוון שהוא מממש פונקציה בשם NewEnum_ (כפי שראית כשצפינו ב-Collection ב-Object Browser)
- המתודה NewEnum מאפשרת ללולאה לאתר רשומות ב-Collection. למעשה Enumerator עוקב אחר המיקום של הלולאה ב-Collection
- כדי שהלולאה For Each...In...Next תפעל ב-Collection שלך, עליך להוסיף פונקציה דומה
- כפי שראינו ב-OLE View, ה-id של המתודה NewEnum_ הוא (4-). לפיכך, גם למתודה שאתה תכתוב ה-id חייב להיות (4-). תוכל להגדיר זאת באמצעות תיבת הדו-שיח Procedure Attributes (תוכל להבין מכך שהשם שתיתן לפונקציה שלך הוא לא חשוב, מה שחשוב הוא שמספר הזיהוי שלה יהיה (4-))

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- כיוון שלא ניתן לכתוב Collection יעילים ב-Visual Basic, משתמשים בטכניקה הנקראת Object Containment and Delegation כדי ליצור Collections משלנו:
  - **Containment:** פירושה **הכלה**. ובאופן יותר מעשי, האובייקט שאנו יוצרים מכיל אובייקט אחר
  - **Delegation:** פירושה **האצלה**. כלומר, בעת קריאה לפונקציה של האובייקט שאנו יוצרים, אנו מעבירים את הביצוע (העבודה השחורה) לאובייקט הפנימי
- בתרגיל הבא נשתמש בעקרון הנ"ל כדי ליצור Collection משלנו לפי השלבים הבאים:
  - צור מחלקה והצהר על משתנה Collection פרטי למחלקה
  - הוסף למחלקה פרוצדורות Add, Remove, Item, Count ו-NewEnum
  - כדי לאפשר ללולאה For Each ... In ... Next, עליך להוסיף מתודה בשם NewEnum (תוכל לתת לה גם שם אחר, אך עדיף לתת לה שם זה)
  - ממש את הפרוצדורות של המחלקה שלך, ע"י קריאה לפרוצדורה המקבילה ב-Collection שבתוך המחלקה
  - מימוש NewEnum יפעיל את הפונקציה NewEnum_ ב-Collection הפנימי במחלקה
  - מתוך תיבת הדו-שיח Procedure Attributes הגדר את מספר הזיהוי של המתודה NewEnum למספר –4
    והגדר את הפונקציה Item כ-Default

תרגיל 9 – התוכנית ShapeColl2

- בתוכנית הבאה תיצור היררכיה של שלושה Class-ים רגילים: IShape, CRect ו-CCircle, כאשר CRect ו-CCircle ממשים את הממשק של IShape
- תיצור גם Collection המקבל את הסוג IShape ולפיכך גם תומך ב-CRect ו-CCircle

1. צור פרויקט חדש מסוג Standard EXE ושמור אותו:

- הקצה לפרויקט את השם ShapeColl2 ולטופס את השם frmTest
- שמור את הפרויקט

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

2. **הוסף לפרויקט ארבעה Class Module-ים:**
   - הקצה ל-Class-ים את השמות IShape, CRect, CCircle ו-Shapes
   - שמור את הפרויקט שוב
3. **הוסף קוד ל-IShape:**
   - IShape מגדיר ממשק בלבד, לפיכך הפונקציות שלו ריקות ממימוש

```
Option Explicit

Public Function Surface() As Double
   ' Empty inside interface
End Function
```

   - הפונקציה Surface מחזירה את שטח הצורה

```
Public Function Perimeter() As Double
   ' Empty inside interface
End Function
```

   - הפונקציה Perimeter מחזירה את היקף הצורה

```
Public Function FormatDetails() As String
   ' Empty inside interface
End Function
```

   - הפונקציה FormatDetails מחזירה מחרוזת עם נתונים אודות:
     - סוג הצורה
     - פרמטרים שונים של הצורה
     - היקף הצורה
     - שטח צורה
   - ***זכור:*** הממשק הנ"ל ממומש ב-CRect וב-CCircle, כך שהמשמעות שתינתן לכל אחת מהפונקציות הנ"ל תהיה שונה בכל אחת מהמחלקות הללו
4. **הוסף קוד למחלקה CCircle:**

```
Option Explicit

Private Const PI As Double = 3.14
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Implements IShape

Private m_dRadius As Double

Public Property Get Radius() As Double
   Radius = m_dRadius
End Property

Public Property Let Radius(dRad As Double)
   If dRad >= 0 Then m_dRadius = dRad
End Property

Private Function IShape_Perimeter() As Double
   IShape_Perimeter = 2 * PI * m_dRadius
End Function
```

- מימוש הפונקציה IShape_Perimeter בתוך המעגל $P = 2\pi r$

```
Private Function IShape_Surface() As Double
   IShape_Surface = PI * m_dRadius * m_dRadius
End Function
```

- מימוש הפונקציה IShape_Surface בתוך המעגל $S = \pi r^2$

```
Private Function IShape_FormatDetails() As String
   Dim strRes As String
   strRes = "CCircle" & vbCrLf
   strRes = strRes & String(8, " ") & "Radius = " & _
            Trim(Str(Me.Radius)) & vbCrLf
   strRes = strRes & String(8, " ") & "Perimeter = " & _
            Trim(Str(IShape_Perimeter())) & vbCrLf
   strRes = strRes & String(8, " ") & "Surface = " & _
            Trim(Str(IShape_Surface())) & vbCrLf
   IShape_FormatDetails = strRes
End Function
```

- המימוש של FormatDetails במעגל מחזיר מחרוזת המכילה:
  - שם המחלקה (CCircle)
  - רדיוס המעגל
  - היקף המעגל

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- שטח המעגל

5. הוסף קוד למחלקה CRect:

```
Option Explicit

Implements IShape

Private m_dLength As Double
Private m_dWidth As Double

Public Property Get Length() As Double
   Length = m_dLength
End Property

Public Property Let Length(dLen As Double)
   If dLen >= 0 Then m_dLength = dLen
End Property

Public Property Get Width() As Double
   Width = m_dWidth
End Property

Public Property Let Width(dWid As Double)
   If dWid >= 0 Then m_dWidth = dWid
End Property

Private Function IShape_Perimeter() As Double
   IShape_Perimeter = 2 * (m_dLength + m_dWidth)
End Function

Private Function IShape_Surface() As Double
   IShape_Surface = m_dLength * m_dWidth
End Function

Private Function IShape_FormatDetails() As String
   Dim strRes As String
   strRes = "CRect" & vbCrLf
   strRes = strRes & String(8, " ") & "Length = " & _
            Trim(Str(Me.Length)) & vbCrLf
   strRes = strRes & String(8, " ") & "Width = " & _
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
            Trim(Str(Me.Width)) & vbCrLf
   strRes = strRes & String(8, " ") & "Perimeter = " & _
            Trim(Str(IShape_Perimeter())) & vbCrLf
   strRes = strRes & String(8, " ") & "Surface = " & _
            Trim(Str(IShape_Surface())) & vbCrLf
   IShape_FormatDetails = strRes
End Function
```

- המימוש של FormatDetails במלבן מחזיר מחרוזת המכילה:
  - שם המחלקה (CRect)
  - אורך המלבן
  - רוחב המלבן
  - היקף המלבן
  - שטח המלבן

6. הוסף קוד ל-Shapes:

```
Option Explicit

Private m_coll As Collection

Private Sub Class_Initialize()
   Set m_coll = New Collection
End Sub

Private Sub Class_Terminate()
   Set m_coll = Nothing
End Sub
```

- בשורות הקוד הנ"ל מוצהר אובייקט Collection שיכיל בפועל את האובייקטים
- בעת אתחול האובייקט (Initialize) יוצרים Collection חדש
- בעת שחרור האובייקט (Terminate) משחררים את ה-Collection שהוצהר
- זהו למעשה קטע ה-Containment במחלקה. כלומר, המחלקה שלנו (Shapes) מכילה אובייקט מסוג Collection

```
Public Sub Add(Item As IShape, Optional Key As String = "")
   If Len(Key) > 0 Then
      m_coll.Add Item, Key
   Else
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
      m_coll.Add Item
   End If
End Sub
```

- הפונקציה Add מוסיפה אובייקט ל-Collection
- הפעולה מתבצעת ע"י הפעלת Add מתוך ה-Collection שהוצהר במחלקה
- העברת הפעולה מהמחלקה שלנו לאובייקט הפנימי m_coll, היא למעשה Delegation (האצלה)
- באופן דומה ננהג בפרוצדורות הבאות

```
Public Sub Remove(Index As Variant)
   m_coll.Remove Index
End Sub

Public Function Item(Index As Variant) As IShape
   Set Item = m_coll.Item(Index)
End Function

Public Function Count() As Long
   Count = m_coll.Count
End Function

Public Property Get NewEnum() As IUnknown
   Set NewEnum = m_coll.[_NewEnum]
End Property
```

- בפונקציה NewEnum העברנו את הפעולה ל-_NewEnum של האובייקט m_coll
- כיוון שאסור להשתמש בקו תחתון בתחילת שם פרוצדורה או משתנה ב- Visual Basic אנו עוטפים את המילה בסוגריים מרובעים

7. הגדר את תכונות הפרוצדורות במחלקה Shapes:

- הישאר בחלון הקוד של Shapes
- מתוך התפריט Tools בחר ב-...Procedure Attributes
- מתוך הרשימה Name בחר ב-Item
- לחץ על הלחצן Advanced – תיבת הדו-שיח תתרחב ותכלול אפשרויות נוספות
- מתוך הרשימה Procedure ID בחר ב-Default
- כך הגדרת את הפרוצדורה Item כפרוצדורת ברירת המחדל של Shapes

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בשדה Name בחר ב-NewEnum
- בתיבה Procedure ID רשום (-4)
- ב-Attributes סמן V ליד Hide This Member
- באופן זה הגדרת את המתודה NewEnum כמתודה נסתרת (לא היית חייב, אבל מומלץ להסתיר ממשתמש המחלקה את מה שהוא לא אמור להשתמש בו). כמו כן נתת ל-NewEnum את מזהה השגרה (-4) שלו מצפה הלולאה For Each...In...Next
- כעת הוסף תיאור בשדה Description לכל אחת מהפרוצדורות של Shapes

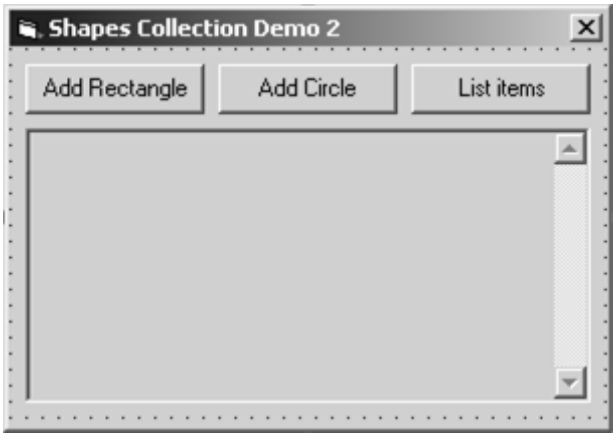
8. **הוסף שלושה CommandButton-ים ו-TextBox אחד לטופס, והגדר את המאפיינים הבאים:**

אובייקט	מאפיין	ערך
frmTest	Caption	Shapes Collection Demo 2
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdAddRect
	Caption	Add Rectangle
Command2	Name	cmdAddCircle
	Caption	Add Circle
Command3	Name	cmdListItems
	Caption	List items
Text1	Name	txtColl
	Background	Button Face
	Locked	True

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

ScrollBars	2 - Vertical
Text	[ריק]
MultiLine	True

- הטופס אמור להיראות דומה לזה שבציור:



9. הוסף קוד לטופס:

```
Option Explicit

Public shpColl As Shapes

Private Sub Form_Load()
   Set shpColl = New Shapes
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Set shpColl = Nothing
End Sub
```

- בשורות הנ"ל מוצהר אובייקט Collection מסוג Shapes
- אובייקט זה נוצר בעת טעינת הטופס ומשוחרר עם פריקת הטופס

```
Private Sub cmdAddCircle_Click()
   Dim cr As CCircle
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
   Set cr = New CCircle
   cr.Radius = Val(InputBox("Enter the radius for the circle:"))
   shpColl.Add cr
   Set cr = Nothing
End Sub

Private Sub cmdAddRect_Click()
   Dim rc As CRect
   Set rc = New CRect
   rc.Length = Val(InputBox("Enter the length of the
rectangle:"))
   rc.Width = Val(InputBox("Enter the width of the rectangle:"))
   shpColl.Add rc
   Set rc = Nothing
End Sub
```

- בדומה לתרגיל הקודם, נוסף מלבן או מעגל בהתאם לבחירת המשתמש

```
Private Sub cmdListItems_Click()
   Dim shp As IShape
   txtColl.Text = ""
   For Each shp In shpColl
      txtColl.Text = txtColl.Text & shp.FormatDetails
   Next
End Sub
```

- בקטע קוד זה מתבצעת סריקה של האובייקטים ב-Collection והפונקציה FormatDetails נקראת כדי להציג את נתוני הצורה
- בשונה מהתרגיל הקודם, כאן אנו יודעים מראש שאנו ניגשים לאובייקטים מסוג IShape, ולפיכך אנו קוראים מיד לפונקציה FormatDetails. אנו מרוויחים מהירות גבוהה המתאימה ל-Early Binding
- בנוסף לרווחים הביצועיים, הקוד כאן הרבה יותר קריא

10. בחינה ראשונה של התוכנית – הרצה:

- הרץ את התוכנית ובחן את פעולה
- אם התקבלו שגיאות בלולאה For Each...In...Next, בדוק אם הגדרת את תכונות הפרוצדורה בשלב 7

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

11. בחינה שנייה של התוכנית - צפייה המחלקות באמצעות Object Browser:

- פתח את חלון ה-Object Browser מתוך התפריט View
- מתוך רשימת ה-Classes מצא את המחלקה Shapes (אם אינך רואה אותה, ודא שב-ComboBox העליון בחרת <All Libraries> או ShapeColl2)
- בפנל הימני אתה אמור לראות את הפרוצדורות של Shapes
- שים לב שפרוצדורות מוסתרות גם מוצגות בפנל הימני. כשתיצור DLL (בשיעור הבא) פרוצדורות אלו יוסתרו.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 10:

ActiveX DLL

בשיעור זה:

- תבין יותר את המושגים Process, Address Space, Thread ו-Component
- תלמד ליצור רכיבי ActiveX מסוג ActiveX DLL
- תלמד כיצד להשתמש ברכיבים שנכתבו כ-ActiveX DLL מתוך תוכנית אחרת
- תכיר את המושג Instancing
- תבין כיצד Windows טוענת קובץ DLL לתוך תוכנית
- תלמד לטפל בשגיאות בתוך רכיבים שתכתוב

מושגים

בסעיף זה תכיר מספר מושגים שיוכלו לסייע לך לנצל טוב יותר את מערכת Windows, להבין מהו Component, ולהכיר טוב יותר את ההבדל בין סוגי ה-Component-ות שניתן לפתח ב-Visual Basic

- **Instance (מופע) של תוכנית:** ניתן להריץ תוכנית מספר פעמים. כל הרצה נפרדת של תוכנית נקראת Instance של התוכנית. כך למשל, אם הרצת את Microsoft Word שלוש פעמים, כל הרצה כזו מהווה Instance של Microsoft Word
- **Process (תהליך):** אוסף המשאבים הדרושים להרצת Instance של תוכנית נקרא Process. ה-Process מנהל את המשאבים הקשורים בתוכנית שרצה. משאבים לדוגמה הם: הקצאת זיכרון לתוכנית, יצירת Thread-ים (יוסבר בהמשך), טעינת רכיבי DLL וכו'. ברגע שתוכנית מסתיימת, מערכת ההפעלה פורקת את ה-Process וכל המשאבים הקשורים אליו.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Thread (מטלת ביצוע):** Thread הינו רצף של פקודות שעל המחשב לבצע. Thread הינו משאב הצורך בפועל זמן מעבד. ב-Process אחד יכולים להיות מספר Thread-ים שרצים בו זמנית. כך למשל, התוכנית Microsoft Word מאפשרת לך לערוך ולהדפיס בו זמנית על ידי הרצת Thread נפרד עבור פעולת ההדפסה. בעת טעינת Process מערכת ההפעלה יוצרת באופן אוטומטי Thread עבור ה-Process שהורץ. Thread זה נקרא ה-Thread הראשי של התוכנית.
- **Address Space (מרחב כתובות):** Address Space הוא מרחב הכתובות הווירטואליות המוקצה לכל Process במערכת ההפעלה. במערכת ההפעלה Windows מסוג 32 ביט, לכל Process יש מרחב כתובות של 4GB (4 ג'יגה) משלו, כאשר בתצורת ברירת המחדל 2GB ממנו זמינים לקוד הפועל במצב User Mode, ו-2GB הנותרים שמורים למערכת ההפעלה (Kernel Mode). תוכל להסיק מכך שאם תוכנית מנסה לגשת לכתובת מסוימת ותוכנית אחרת מנסה לגשת לאותה הכתובת, כל תוכנית תיגש בפועל לנתונים ייחודיים משלה – כלומר יש הגנה על מרחב הכתובות מפני Process-ים אחרים. תשאל את עצמך, בוודאי, איך ייתכן שלכל Process ישנם 4G כתובות זיכרון, והרי זיכרון ה-RAM הוא מצומצם מאוד (בדרך כלל בין 32M ל-256M ברוב המחשבים). יתרה מזאת, אם נניח שישנם 50 Process-ים שרצים במקביל במערכת, כי אז אנו מדברים על 200G זיכרון (50x4G=200G). חסוך מעצמך את הטרדה הזו, בכך שתדע (מבלי שניכנס לפרטים טכניים) כי מערכת ההפעלה עושה שימוש חסכוני בכתובות על ידי שימוש במספר טריקים, כגון שימוש ב-Page-ים (דפי זיכרון), Page Tables (טבלאות דפים) ועוד. היבט נוסף שחשוב לשים לב אליו, בעוד שלכל Process יש Address Space משלו, Thread-ים הרצים באותו ה-Process משתתפים באותו Address Space. תוכל להסיק מכך ש-Thread-ים הרצים תחת אותו Process יכולים לשתף ביניהם מידע גלובלי.
- **EXE (Executable File – קובץ ריצה):** קובץ EXE הוא קובץ תוכנית שעם הרצתו נוצר עבורו Instance חדש, וכתוצאה מכך נוצר Process במערכת. ל-Process שנוצר מוקצים מיד Address Space, Main Thread ועוד.
- **DLL (Dynamic Link Library – ספריית קישור דינמי):** קובץ DLL הוא קובץ שיכול לייצא משאבים (למשל Icon-ים, מחרוזות ועוד), כמו גם פונקציות.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- משמעות המושג לייצא היא שמשאבים אלו נועדו לשימושן של תוכניות אחרות, המייבאות אותם. DLL אינו פועל בפני עצמו, אלא נטען לשימושן של תוכניות אחרות. קובץ DLL הנטען על ידי תוכנית, נטען לתוך ה-Address Space של ה-Process.

- **Component (רכיב):** Component (כפי שהוסבר בפרק 8) הוא רכיב בינרי. Component יכול להיות מאוחסן בקובץ EXE, DLL ועוד. נבדיל בין מספר מושגים הקשורים ב- Component-ות:
  - **Server Component (רכיב שרת):** זהו Component המספק שירותים. כך למשל ניתן לכתוב Component מיוחד, המספק שירותי חישובים מתמטיים ועוד.
  - **Client Component (רכיב לקוח):** זהו רכיב המנצל שירותים של רכיב אחר.

שים לב: Client Component יכול להיות גם Server Component. כך למשל ייתכן ש-Component מסוים מספק שירותים מסוימים ומבחינה זו הוא Server, אך במקביל מנצל שירותים של Component אחר כדי להשלים את המשימה, ומבחינה זו הוא Client.

- **In Process Component (רכיב תוך תהליכי):** זהו Component הנטען אל תוך Address Space של Process קיים. חשוב למשל על Control מיוחד המציג לוח שנה, וממוקם על טופס בתוכנית שלך. רכיב זה הינו חלק מהתוכנית והוא נטען לתוך התוכנית.
- **Out Of Process Component (רכיב חוץ תהליכי):** זהו רכיב הרץ כתוכנית עצמאית ומספק שירותים לתוכניות אחרות. חשוב למשל על התוכנית שלך המריצה את Microsoft Word כדי לנצל את שירותי האיות (Spelling) של Microsoft Word. במקרה זה ה-Server Component הנו התוכנית Word הרצה ב-Process נפרד (Word היא תוכנית EXE ולכן היא רצה כתוכנית עצמאית)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

יצירת ActiveX DLL

- יש לבחור את הפרויקט ActiveX DLL
- יש להבחין בין DLL-ים שנוצרים ב-Visual Basic לבין DLL-ים רגילים. DLL-ים הנוצרים ב-Visual Basic הם מסוג ActiveX ולפיכך מהווים ספריות של רכיבים. ב-DLL מסוג זה אין גישה ישירה לפונקציה מסוימת, אלא קיימת אפשרות ליצור רכיב. לאחר שהרכיב נוצר ניתן לפנות למתודות שלו. כלומר DLL-ים של ActiveX מייצאים רכיבים ולא פונקציות. DLL-ים רגילים של Win32 מייצאים פונקציות.
- כל רכיב ב-DLL נכתב ב-Class Module נפרד

יצירת אובייקטים

יישום לקוח המעוניין להשתמש ברכיבים מספרייה חיצונית יכול לבצע זאת במספר דרכים:

- הוספת הפניה ב-Project References, ולאחר מכן יצירת אובייקט באמצעות המילה New.
- יצירת אובייקט והשמתו במשתנה מסוג Object באמצעות הפונקציה CreateObject (במקרה כזה יהיה שימוש ב-Late Binding. כדי להימנע מכך, ניתן ליצור משתנה מסוג האובייקט, ולהוסיף את ההפניה המתאימה ב-Project References)
- באמצעות הפונקציה CreateObject ניתן ליצור אובייקטים גם במחשבים מרוחקים. פשוט יש להעביר את שם המחשב ברשת כפרמטר שני לפונקציה לפי התחביר הבא:

```
CreateObject(component_name, computer_name)
```

הצגת ממשק משתמש

- רכיב DLL יכול ליצור ממשק משתמש למשל על ידי קריאה לפונקציה MsgBox
- ניתן למנוע הצגת ממשק משתמש על ידי הרכיב אם מסמנים V ליד האפשרות Unattended Execution בכרטיסיית General בחלון ה-Project Properties
- אם רכיב שהוגדר עבור Unattended Execution מנסה להציג ממשק משתמש, תירשם הודעה בקובץ ה-Log של המערכת (ב-Windows 2000 ניתן לצפות בהודעה מתוך חלון ה-Event Viewer תחת הקטגוריה Application Log. היישום Application Viewer נמצא תחת Administrative Tools)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

ניהול קבוצת פרויקטים במקביל

- בעת פיתוח רכיב יש לבחון אותו במקביל לפיתוח
- ניתן ליצור קבוצת פרויקטים ב-Visual Basic ולנהל אותם במקביל
- Visual Basic יוצרת קובץ פרויקט (*.vbg) העוקב אחר מיקום הפרויקטים ומנהל אותם
- אני ממליץ בפניך ליצור ספרייה עבור קבוצת הפרויקטים. תחת ספרייה זו צור תתי ספריות עבור כל פרויקט בקבוצת הפרויקטים. את הקבצים של כל פרויקט שמור בספרייה השייכת לפרויקט, ואת קובץ הפרויקט שמור תחת ספריית הקבוצה

תרגיל 1 – ספריית הרכיבים SecurityLib

- בתרגיל זה ניצור רכיב DLL המאפשר הצגת מסך Login, ובדיקת שם משתמש וסיסמה
- הפונקציה Login תציג את מסך ה-Login. אם הסיסמה שהוכנסה תקינה, הפונקציה תחזיר True, אחרת יוחזר False
- בדוגמה זו אנו מניחים שהמשתמש היחיד הוא "superuser" ושסיסמתו היא "1234". ברור, עם זאת, שביישום אמיתי תכתוב משהו קצת פחות טיפשי

1. **צור קבוצת פרויקטים ActiveX DLL ו-Standard EXE:**
   - פתח את Visual Basic יחד עם פרויקט מסוג ActiveX DLL
   - שנה את שם הפרויקט ל-SecurityLib
   - בתוך הפרויקט ישנו Class Module. שנה את שמו ל-CLogin
   - הוסף פרויקט Standard EXE על ידי בחירת האפשרות Add Project מהתפריט File
   - שנה את שם הפרויקט ל-TestLib ואת שם הטופס שבו ל-frmTestLib
2. **שמור את קבוצת הפרויקטים:**
   - בעת לחיצה על Save ייפתח חלון השמירה. צור ספרייה בשם Security ותחת ספרייה זו צור שתי ספריות בשם SecurityLib ו-TestLib
   - את קבצי פרויקט ה-ActiveX DLL שמור תחת הספרייה SecurityLib ואת קובצי פרויקט ה-Standard EXE שמור תחת הספרייה TestLib

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- לאחר שתשמור את הפרויקטים, Visual Basic תבקש ממך לשמור את קובץ קבוצת הפרויקטים. שנה את שם הקובץ ל-grpSecurity.vbg ושמור אותו תחת הספרייה Security

3. הוסף טופס לספריית הרכיבים:

- בחלון ה-Project Explorer בחר בפרויקט SecurityLib (אם אינך רואה את חלון ה-Project Explorer תוכל לבחור אותו מתוך התפריט View או על ידי הצירוף Ctrl+R)
- הוסף טופס על ידי בחירת האפשרות Add Form מתוך התפריט Project
- שנה את שם הטופס ל-frmLogin
- שמור את הטופס תחת הספרייה השייכת לפרויקט SecurityLib (\Security\SecurityLib\)

4. עצב את הטופס frmLogin:

- הוסף שני TextBox-ים, שני CommandButton-ים ושני Label-ים לטופס, והגדר את המאפיינים הבאים:

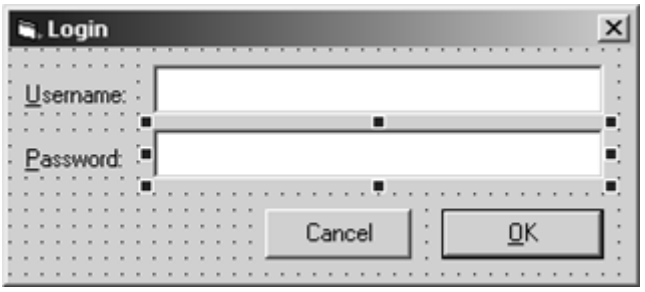
אובייקט	מאפיין	ערך
frmLogin	Caption	Login
	BorderStyle	3- Fixed Dialog
	StartUpPosition	1-CenterOwner
Label1	Name	lblUsername
	Caption	&Username
	TabIndex	0
Label2	Name	lblPassword

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

	Caption	&Password
	TabIndex	2
Text1	Name	txtUsername
	Text	[ריק]
	TabIndex	1
Text2	Name	txtPassword
	MaxLength	8
	PasswordChar	*
	Text	[ריק]
	TabIndex	3
Command1	Name	cmdOK
	Caption	&OK
	Default	True
Command2	Name	cmdCancel
	Caption	&Cancel
	Cancel	True

- הטופס אמור להיראות בדומה לציור שלפניך:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il



5. הוסף קוד לטופס frmLogin:

```
Option Explicit

Private m_strUsername As String
Private m_strPassword As String
```

- שני המשתנים הללו יכילו את שם המשתמש והסיסמה לאחר שהטופס יוסר

```
Private Sub cmdOK_Click()
   m_strUsername = txtUsername.Text
   m_strPassword = txtPassword.Text
   Unload Me
End Sub
```

- בעת לחיצה על OK מעודכנים ערכי המשתנים m_strUsername ו-m_strPassword והטופס מפורק על ידי המתודה Unload

שים לב: למרות שהטופס מפורק, המשתנים m_strUsername ו-m_strPassword עדיין קיימים בזיכרון. ברגע שהטופס ישוחרר מהזיכרון על-ידי Set...= Nothing יפורקו גם המשתנים הללו.

```
Private Sub cmdCancel_Click()
   Unload Me
End Sub
```

- בעת לחיצה על Cancel הטופס מפורק מבלי לעדכן את m_strUsername ו-m_strPassword

```
Public Property Get Username() As String
   Username = m_strUsername
End Property
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Public Property Get Password() As String
   Password = m_strPassword
End Property
```

- שתי פרוצדורות ה-Property Get הללו מאפשרות לאחזר את ערכי המשתנים m_strUsername ו-m_strPassword.

6. הוסף קוד למחלקה CLogin:

```
Option Explicit

Public Function ShowLogin() As Boolean
   Dim loginForm As New frmLogin
   loginForm.Show vbModal
   If LCase(loginForm.Username) = "superuser" And _
      loginForm.Password = "1234" Then
      ShowLogin = True
   Else
      ShowLogin = False
   End If
   Set loginForm = Nothing
End Function
```

- הפונקציה ShowLogin יוצרת Instance של הטופס frmLogin ומציגה אותו למשתמש במצב מודאלי
- לאחר שהמשתמש לחץ על OK או Cancel נבדקים המאפיינים Username ו-Password של loginForm. אם הם שווים ל-"superuser" ו-"1234" בהתאמה, אזי מוחזר מהפונקציה הערך True, אחרת יוחזר False

7. עבור לפרויקט TestLib:

- לחץ על שם הפרויקט TestLib בחלון ה-Project Explorer

8. עצב את הטופס frmTestLib:

- הוסף לטופס frmTestLib שני CommandButton-ים והגדר את המאפיינים הבאים:

אובייקט	מאפיין	ערך

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

frmTestLib	Caption	Test Login
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdUseNew
	Caption	Use "Set ...New"
Command2	Name	cmdUseCreate
	Caption	Use "CreateObject (...)"

- הטופס אמור להיראות דומה לזה שבציור שלפניך:



9. הוסף Reference לפרויקט SecurityLib:

- מתוך התפריט Project בחר באפשרות References
- סמן V ליד הספרייה SecurityLib

שים לב: ללא הפניה זו לא תוכל ליצור אובייקטים או משתנים מהסוג CLogin בתוך יישום הלקוח (במקרה הזה התוכנית TestLib) תוך שימוש במילה New. עם זאת, תוכל ליצור אובייקטים באמצעות הפונקציה CreateObject תוך ציון שם הספרייה ושם הרכיב בספרייה, כלומר CreateObject("SecurityLib.CLogin")

10. הוסף קוד לטופס frmTestLib:

```
Option Explicit

Private Sub cmdUseNew_Click()
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
    Dim sec As New SecurityLib.CLogin
    If sec.ShowLogin() Then
       MsgBox "You are succesfull logged in!"
    Else
       MsgBox "Login Failed!"
    End If
    Set sec = Nothing
End Sub
```

- בתגובה ללחיצה על cmdUseNew יוצרים מופע של המחלקה CLogin מתוך הספרייה SecurityLib
- לאחר מכן מפעילים את הפונקציה ShowLogin ומציגים הודעה למשתמש באמצעות Message Box על פי התוצאה המוחזרת מ-ShowLogin

```
Private Sub cmdUseCreate_Click()
    Dim sec As Object
    Set sec = CreateObject("SecurityLib.CLogin")
    If sec.ShowLogin() Then
       MsgBox "You are succesfull logged in!"
    Else
       MsgBox "Login Failed!"
    End If
    Set sec = Nothing
End Sub
```

- פונקציה זו זהה לקודמת אלא שהפעם האובייקט נוצר באמצעות CreateObject ומושם לתוך משתנה מסוג Object

11. קבע את ה-Start Up Project:

- לפני שתנסה להריץ את הפרויקט עליך לקבוע איזה פרויקט יופעל על ידי Visual Basic
- בחלון ה-Project Explorer לחץ לחיצה ימנית על הפרויקט TestLib. בתגובה יוצג תפריט מוקפץ שהאפשרות הראשונה בו היא Set As Start Up. בחר באפשרות זו.
- שים לב ש-Visual Basic הדגישה את שם הפרויקט

12. בחינה ראשונה של הפרויקט:

- הרץ את התוכנית באמצעות F5 ובחן את פעולתה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

13. בחינה שנייה של הפרויקט:

- בטל את ה-Reference לספרייה SecurityLib על ידי הסרת סימן ה-V מתיבת הדו-שיח References. (לפתיחת תיבת דו-שיח זו בחר ב-References מתוך התפריט Project)
- נסה להריץ את הפרויקט. שים לב כי לחיצה על הלחצן "Set...New" Use תגרום לשגיאה, ואילו שימוש ב-"CreateObject (...)" Use ימשיך לפעול

14. שמור את הפרויקטים:

- החזר את ה-Reference שביטלת
- שמור את הפרויקטים
- אם אתה לא הולך לישון, השאר את הפרויקטים פתוחים. בתרגיל הבא נעשה שימוש בקבוצת פרויקטים זו

INSTANTIATION (יצירת מופעים)

- לכל Class Module ישנו מאפיין Instancing המגדיר כיצד ניתן ליצור אובייקטים מסוג המחלקה
- המאפיין Instancing משפיע על אופן יצירת האובייקטים על ידי תוכנית חיצונית המשתמשת בספריית הרכיבים. מאפיין זה אינו משפיע על אופן יצירת הרכיבים מתוך הפרויקט עצמו.
- Instancing עבור Class בפרויקט מסוג ActiveX DLL מאפשר את הערכים הבאים:

- **Private:** ה-Class הוא פרטי של הפרויקט. יישומים חיצוניים אינם יכולים ליצור אובייקטים מסוג המחלקה. יתרה מכך, יישומים חיצוניים אינם יכולים להצהיר אפילו על משתנים מסוג המחלקה. כלומר, ה-Class-ים הללו פרטיים באופן מוחלט בפרויקט ואינם מוכרים כלל ליישומים חיצוניים.
- **PublicNotCreatable:** ה-Class יהיה מוכר ליישומים חיצוניים באופן כזה שניתן להצהיר על משתני ייחוס מסוג האובייקט הזה. עם זאת, יישומים חיצוניים אינם יכולים ליצור בעצמם אובייקטים מסוג המחלקה.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **MultiUse:** מאפשר ליישומי לקוח ליצור מופעים רבים של המחלקה. לכל מופע עותק נפרד של המשתנים הפנימיים שלו, ואילו כל המופעים באותו תהליך לקוח משתמשים בעותק משותף של קוד המחלקה.
- **GlobalMultiUse:** זהה ל-MultiUse אלא שהיישום החיצוני אינו חייב ליצור עותק של האובייקט. כלומר באופן אוטומטי נוצר אובייקט כזה על ידי ספריית הרכיבים. כמו כן, היישום יכול לפנות לפונקציות של המחלקה ללא הקדמת שם אובייקט. אם לדוגמה קיימת מחלקה בשם CStatistics הכוללת את הפונקציה StdDev, אפשר להשתמש בפונקציה פשוט על ידי רישום StdDev(...) כאילו היה מדובר בפונקציה גלובלית.

שים לב: בכל פרויקט ActiveX חייב להיות לפחות Class אחד הניתן ליצירה

תרגיל 2 – שינוי המאפיין Instancing במחלקה CLogin

בתרגיל זה ננסה את האפשרות GlobalMultiUse במקום MultiUse

1. **פתח את קבוצת הפרויקטים grpSecurity.vbg:**
   - אם סגרת את קבוצת הפרויקטים grpSecurity תוכל לפתוח אותה מתוך הסייר על ידי לחיצה כפולה על שם הקובץ grpSecurity.vbg
2. **שנה את המאפיין Instancing:**
   - שנה את המאפיין Instancing של המחלקה CLogin ל-GlobalMultiUse
   - שמור את הפרויקט
3. **בחינה ראשונה:**
   - הרץ את הפרויקט
   - ודא שהכל פועל כרגיל
4. **בחינה שנייה:**
   - הוסף CommandButton נוסף לטופס. שנה את שמו ל-cmdUseGlobal ואת הכותרת שלו ל-"Use Global"

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- הוסף את הקוד הבא בתגובה ללחיצה על cmdUseGlobal:

```
Private Sub cmdUseGlobal_Click()
   If ShowLogin() Then
      MsgBox "You are succesfull logged in!"
   Else
      MsgBox "Login Failed!"
   End If
End Sub
```

- שים לב כי הפונקציה ShowLogin מופעלת כאילו הייתה פונקציה גלובלית. דבר זה מתאפשר הודות להגדרת CLogin כ-GlobalMultiUse. במקרה זה Visual Basic יוצרת אוטומטית מופע מסוג CLogin בעת הפנייה הראשונה לאחד מחברי המחלקה. כמו כן שים לב שאתה יכול לרשום SecurityLib.ShowLogin כאילו מדובר בפונקציה גלובלית של הספרייה

5. שמור את קבוצת הפרויקטים:

- מתוך התפריט File בחר ב-Save
- אם עדיין לא נרדמת אז תישאר ער עוד קצת. בתרגיל הבא נסיים את העבודה על קבוצת הפרויקטים הזו.

הגדרת כתובות בסיס לרכיבי DLL

- רכיב DLL (כפי שהוסבר בתחילת השיעור) נטען לתוך ה-Address Space של תוכנית הלקוח
- בעת קימפול הרכיב, קריאות לפונקציות מתפרשות לפי כתובות הפונקציות. לדוגמה, אם נניח שפונקציה מסוימת נמצאת במרחק (Offset) &H00000100 מתחילת ה-DLL ואותו DLL קומפל מתוך הנחה שייטען לכתובת &H11000000 בתוכנית הלקוח (זוהי כתובת הבסיס), אזי כתובת הפונקציה לאחר הקימפול תהיה &H11000100 (כתובת בסיס + כתובת היסט)
- כאשר תוכנית משתמשת במספר DLL-ים סביר מאוד ששני DLL-ים או יותר ינסו להיטען לאותה כתובת בסיס

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- אם DLL מנסה להיטען לכתובת בסיס שנתפסה כבר על ידי DLL אחר, מערכת ההפעלה Windows מקצה כתובת בסיס ייחודית עבור הרכיב כדי למנוע התנגשויות. בנוסף, מערכת ההפעלה משנה את הכתובות כך שיתאימו לכתובת הבסיס החדשה. פעולה זו נקראת Rebasing. שינוי כתובות הבסיס כרוך בזמן טעינה ארוך יותר.
- ניתן להגדיר את כתובת הבסיס שלפיה יתבצע קימפול הרכיב מתוך הכרטיסייה Compile בתיבת הדו-שיח Project Properties
- כתובת ברירת המחדל היא &H11000000 ומומלץ לשנות אותה, כיוון שסביר להניח שישנם לא מעט רכיבים שכתובת זו מוקצת עבורם
- בעת הגדרת כתובת בסיס זכור את הכללים הבאים:
  - על כתובת הבסיס להיות כפולה של 64K (בייצוג הקסדצימלי: &H10000), כלומר ארבע הספרות האחרונות של כתובת הבסיס יהיו 0000.
  - Address Space של Process כולל 4G כתובות. עם זאת, הכתובות הבאות שמורות עבור מרכיבים שונים של Windows:
    - ב-Windows NT/2000 וכן ב-Windows 98, רצף הכתובות בין 2G (הכתובת &H80000000) לבין 4G (הכתובת &HFFFFFFFF) שמור עבור Windows
    - ב-Windows98 רצף הכתובות שבין &H00000000 ל-&H00400000 (כלומר 4M הזיכרון התחתון) שמור גם הוא ל-Windows
    - ב-Windows NT/2000 64KB התחתונים של הזיכרון (כלומר הרצף &H00000000-&H00010000) שמור למערכת ההפעלה, וכן 64K מתחילים ברצף &H7FFF0000 עד &H80000000
  - אני ממליץ לשנות את כתובת הבסיס כך שתהיה כפולה של 64K בין הכתובת &H40000000 לבין &H80000000 (כלומר בין 1G ל-2G)

תרגיל 3 – שינוי כתובת הבסיס לספרייה SecurityLib

בתרגיל זה נשנה את כתובת הבסיס לטעינת ה-DLL

1. **פתח את קבוצת הפרויקטים שכתבנו בתרגילים הקודמים**

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

2. **בחר בפרויקט SecurityLib:**
   - מתוך קבוצת הפרויקטים המוצגים בחלון ה-Project Explorer בחר ב-SecurityLib
3. **שנה את כתובת הבסיס:**
   - בחר באפשרות SecurityLib Properties... מתוך התפריט Project
   - עבור לכרטיסייה Compile
   - בשדה DLL Base Address המופיע בתחתית תיבת הדו-שיח שנה את הכתובת לערך &H80000000 ולחץ OK. אתה אמור לקבל הודעה כי כתובת זו אינה חוקית. נסה את הכתובות הבאות:
     - &H00000000 כתובת 0 אינה חוקית
     - &H40001000 כתובת שאינה כפולה של 64K אינה חוקית (בכתובות שהן כפולה של 64K ארבע הספרות הימניות הן 0)
     - &H40010000 כתובת חוקית
4. **שמור את הפרויקט:**
   - שמור את הפרויקט וסגור אותו

טיפול בשגיאות ברכיב

- בעת יצירת רכיב שים לב שעל הרכיב שלך לטפל בכל השגיאות העלולות להתרחש במהלך החיים שלו
- שגיאה המתרחשת ברכיב ואינה זוכה לטיפול תגרום לקריסת תוכנית הלקוח (חשוב איך יגיבו המשתמשים של הרכיב שלך ברגע שהם יגלו שאתה הוא הבאג שלהם!!!)
- תוכל לטפל בשגיאות הן ע"י פתרון הבעיה בתוך הרכיב או ע"י שיגור שגיאה (מותאמת אישית או המקורית) החוצה אל תוכנית הלקוח
- כדי להקצות מספר לשגיאה מותאמת אישית עליך להוסיף מספר בין 513 ל-65535 לקבוע vbObjectError
- תוכל לעורר את השגיאה באמצעות המתודה Raise של האובייקט Err

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- אם בתוך הרכיב שלך מתבצע טיפול בשגיאות ואתה רוצה לעורר שגיאה ללקוח, קרא תחילה ל-On Error GoTo 0, כדי לבטל את מטפל השגיאות הפעיל

אזהרה: אל תנסה לעורר שגיאה ללקוח מתוך אירוע השייך לאובייקט. לשגיאה כזו אין מקום לכידה בתוך תוכנית הלקוח, ולפיכך היא תקרוס.

תרגיל 4: ספריית הרכיבים DynamicArray

- בדוגמה זו ניצור ספריית רכיבים בשם DynamicArray שתכיל רכיב אחד בשם CDynamicArray
- בשונה מהתרגיל הקודם, בתרגיל זה נקמפל את ה-DLL ונטען אותו לתוכנית שנייה.
- רכיב זה ישמש כמערך דינמי המשנה את גודלו באופן אוטומטי כאשר מכניסים איבר באינדקס מסוים שאינו מוקצה באותו שלב במערך
- האינדקסים במערך רצים מ-0 עד (n – 1), כאשר n הוא גודל המערך
- ניסיון לגשת לאינדקס שלילי יגרום לשגיאה
- ניסיון לקריאה (Property Get) של איבר באינדקס שעדיין לא הוקצה גורם לשגיאה
- שים לב כי הרכיב מטפל בכל השגיאות, ע"י פתרון שלהם (כמו שינוי ממדי המערך) או העברה שלהם החוצה, אל רכיב הלקוח
- בשיעור הקודם הכרנו את אפשרות הגדרת התכונות עבור פרוצדורות של מחלקה. בתרגיל זה נשתמש באפשרות זו.

1. **צור פרויקט חדש מסוג ActiveX DLL, ושמור אותו:**
   - הקצה לפרויקט את השם DynamicArray ולמחלקה CDynamicArray
   - שמור את הפרויקט
2. **הוסף משתנים למחלקה:**
   - המחלקה תכיל מערך דינמי פנימי

```
Private m_anValues() As Integer
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

3. הוסף את המאפיין Count:

- המאפיין Count מחזיר את גודל המערך (אחד מעל האינדקס האחרון במערך) תוך שימוש בפונקציות LBound ו-UBound
- אם לא הוקצו איברים למערך תתרחש שגיאה. לוכדים שגיאה זו כדי להחזיר גודל מערך 0

```
Public Property Get Count() As Integer
   On Error GoTo Err_Handler
   Count = UBound(m_anValues) - LBound(m_anValues) + 1
   Exit Property
Err_Handler:
   Count = 0
End Property
```

4. הוסף את השגרה Clear:

- השגרה Clear משחררת את הזיכרון שהוקצה עבור המערך תוך שימוש במילה השמורה Erase

```
Public Sub Clear()
   Erase m_anValues
End Sub
```

5. הוסף את המאפיין Item:

- המאפיין Item מקבל אינדקס במערך ומאפשר לאחזר או לקבוע את ערכו
- כאשר מעבירים מספר פרמטרים ל-Property Let הפרמטר האחרון יהווה את סוג המאפיין
- יש להקפיד שכל הפרמטרים (מלבד האחרון ב-Property Let) שב-Property Get וב-Property Let יהיו זהים בסוגם ובשמם

```
Public Property Get Item(ByVal n As Integer) As Integer
   On Error GoTo Err_Handler
   Item = m_anValues(n)
   Exit Property
Err_Handler:
   Err.Raise Err.Number
End Property
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Public Property Let Item(ByVal n As Integer, ByVal value As
Integer)
   On Error GoTo Err_Handler
   m_anValues(n) = value
   Exit Property
Err_Handler:
   If n >= 0 Then
      ReDim Preserve m_anValues(n)
      Resume
   Else
      Err.Raise Err.Number
   End If
End Property
```

6. הוסף תיאורים לממשק המחלקה:

- בעת שימוש באובייקטים ניתן לבחון את מאפייניהם תוך שימוש ב-Object Browser (ניתן לפתוח את ה-Object Browser באמצעות F2 או מתוך תפריט View)
- מתוך התפריט Tools בחר ב-Procedure Attributes
- מתוך התיבה Name בחר ב-Clear והקלד את התיאור הבא בשדה Description:

```
Free the memory occupied by the array.
```

- באופן דומה הוסף את התיאור הבא למאפיין Count:

```
Returns the number of items in the array.
```

- ואת התיאור הבא למאפיין Item:

```
Sets/Returns the value of an item in the array.
```

- אל תסגור את חלון ה-Procedure Attributes. תזדקק לו בשלב הבא.

7. הגדר את Item כמאפיין ברירת מחדל:

- להזכירך, מאפיין ברירת מחדל הוא המאפיין שיופעל במקרה שלא צוין במפורש שם מאפיין (לדוגמה, המאפיין Text הוא מאפיין ברירת המחדל של TextBox)
- בחר במאפיין Item בתיבה Name ולחץ על הלחצן Advanced
- כעת, כדי להפוך את Item למאפיין ברירת מחדל, ב-Procedure ID בחר ב-(Default)

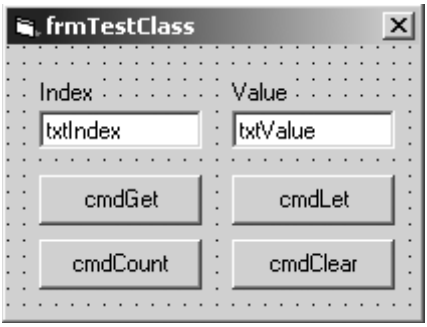
© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

8. **בצע קומפילציה לספריית הרכיבים:**
   - מתוך התפריט File בחר ב-Make DynamicArray.dll...
   - הספרייה תקומפל ו-Visual Basic תרשום את הרכיב באופן אוטומטי במערכת
   - אם התגלו שגיאות, תקן אותן וקמפל מחדש את הפרויקט
9. **צור תוכנית לקוח לבחינת ספריית הרכיבים:**
   - הפעל את Visual Basic מחדש ופתח פרויקט Standard EXE
   - הקצה לפרויקט את השם TestDArray ולטופס שבו את השם frmTestDArray
   - שמור את הפרויקט
10. **הוסף הפנייה לספריית הרכיבים:**
    - מתוך התפריט Project בחר בפריט References
    - סמן V ליד הספרייה DynamicArray (שים לב שלא היית צריך לבצע רישום של הרכיב, כי Visual Basic ביצעה זאת עבורך בעת קימפול הספרייה בשלב 8)

שים לב: אם תרצה לערוך ולאחר מכן לקמפל את ה-DLL כאשר יש הפניה אליו מתוך הפרויקט TestDArray, יהיה עליך להסיר את ההפניה כדי שהקימפול יצליח.

11. **בחינה ראשונה: ה-Object Browser:**
    - פתח את ה-Object Browser מתוך התפריט View או ע"י שימוש בקיצור F2
    - בחר ב-CDynamicArray ועיין במאפיינים והמתודות המוצגים בפנל הימני של החלון
    - שים לב שתיאור המאפיינים והמתודות מופיע בתחתית החלון של ה- Object Browser
    - שים לב כי ליד סמל המאפיין Item מופיע עיגול כחול קטן המעיד כי זהו מאפיין ברירת המחדל
    - סגור את חלון ה-Object Browser
12. **הוסף Controls לטופס:**
    - הוסף Controls לטופס כפי שמוצג בציור הבא:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il



13. הוסף קוד לטופס:

- הקוד הבא מצהיר על אובייקט מסוג CDynamicArray

```
Private m_DArray As DynamicArray.CDynamicArray
```

14. הוסף קוד לטופס:

- הקוד הבא מאתחל את ממשק המשתמש ויוצר Instance של CDynamicArray:

```
Private Sub Form_Load()
    ' Initializations:
    Me.Caption = "Test Dynamic Array"
    txtIndex.Text = ""
    txtValue.Text = ""
    cmdGet.Caption = "&Get"
    cmdLet.Caption = "&Let"
    cmdCount.Caption = "&Count"
    cmdClear.Caption = "Clea&r"

    ' Create the dynamic-array class
    Set m_DArray = New CDynamicArray
End Sub
```

15. הוסף קוד לטופס:

- הקוד הבא פועל בתגובה ללחיצה על אחד הלחצנים
- שים לב שאין חובה לציין במפורש את המאפיין Item בעת השימוש בו, כיוון שזהו מאפיין ברירת המחדל. כלומר הרישום m_DArray(1) = 100 כמוהו כרישום m_DArray.Item(1) = 100

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Private Sub cmdClear_Click()
   m_DArray.Clear
End Sub

Private Sub cmdCount_Click()
   MsgBox "The array contains " & Str(m_DArray.Count) & " items"
End Sub

Private Sub cmdGet_Click()
   txtValue.Text = Str(m_DArray(Val(txtIndex.Text)))
End Sub

Private Sub cmdLet_Click()
   m_DArray(Val(txtIndex.Text)) = Val(txtValue.Text)
End Sub
```

16. בחינה שנייה: הרצת הפרויקט:

- הרץ את התוכנית ובצע את הפעולות הבאות (לפי הסדר)
- לחץ על Count: אתה צריך לקבל 0
- לחץ על Get: אתה אמור לקבל שגיאה. זוהי השגיאה שהרכיב מעורר כשמנסים לקרוא פריט שאינו קיים (עיין בקוד הרכיב לפרטים נוספים)
- הרץ שוב את הפרויקט והזן ערך למערך. בשדה Index בחר באינדקס המתאים ובשדה Value בחר בערך עבור אותו פריט. לאחר מכן לחץ Let. באופן זה הזן מספר ערכים
- השתמש ב-Get כדי לאחזר את הערכים הללו
- השתמש ב-Count כדי לקבל את מספר האיברים החדש במערך
- השתמש ב-Clear כדי לרוקן את המערך
- נסה להזין ערך למערך באינדקס שלילי כלשהו, אתה אמור לקבל שגיאה שהרכיב מעורר (עיין בקוד הרכיב לפרטים נוספים)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 11:

כתיבת Stored Procedures ב-SQL Server

בשיעור זה:

- תבין מהם Stored Procedures ותלמד כיצד להוסיף Stored Procedure ל-Database של SQL Server
- תלמד לכתוב Stored Procedures המקבלות פרמטרים
- תכיר מבני בקרה שונים ב-Transact SQL

Stored Procedures

- Stored Procedure היא פרוצדורה המאוחסנת בתוך ה-Database
- Stored Procedure יכולה לקבל פרמטרים ולפיכך יכולה לשמש לבקשות שונות
- Stored Procedure נשמרת במסד הנתונים. בעת הפעלתה SQL Server יכול ליצור תוכנית ביצוע (Execution Plan) ולשמור אותה בזיכרון לשימוש חוזר. במקרים מסוימים נדרש קימפול מחדש.
- כדי להבין את היתרון שבהפעלת Stored Procedure מתוך תוכנית לקוח לעומת בניית משפט SQL ושליחתו לביצוע ל-Database על ידי תוכנית לקוח, נבחן את תהליך הביצוע בשני המקרים על ידי ה-Database:

תהליך הביצוע של משפט SQL הנשלח על ידי לקוח

- נניח כי תוכנית לקוח יוצרת את משפט ה-SQL הבא ושולחת אותו לביצוע ב-Database:

```
SELECT *
FROM Employees
WHERE EmployeeID = 10
```

- בתוך ה-Database:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

1. מתבצע ניתוח סינטקטי (תחבירי) של המשפט
2. קיום הטבלה Employees נבדק. כמו כן נבדקות הרשאות הגישה לטבלה זו על ידי המשתמש המפעיל את משפט ה-SQL
3. נוצר Execution Plan עבור משפט ה-SQL

(Execution Plan היא תוכנית הביצוע של משפט ה-SQL הכוללת הנחיות כיצד יש לסרוק טבלאות כדי לאתר את הרשומות, האם יש להשתמש באינדקסים, בצירוף טבלאות, כיצד יש למזג את הטבלאות וכו')

4. משפט ה-SQL מופעל והתוצאות מוחזרות למשתמש

- נניח שאותו משפט SQL נשלח שוב על ידי תוכנית הלקוח
- SQL Server שומר בזיכרון תוכניות ביצוע שניתן להשתמש בהן שוב, וכן דפי נתונים שנקראו מהדיסק.
- אם תוכנית הביצוע נמצאת בזיכרון SQL Server, יכול להשתמש בה שוב. אם גם דפי הנתונים הדרושים כבר נמצאים בזיכרון, ניתן לחסוך חלק מקריאות הדיסק. אם תוכנית הביצוע אינה זמינה עוד, ייתכן שיהיה צורך ליצור אותה מחדש.
- אם המשפט אינו נמצא בין המשפטים האחרונים שבוצעו (זה יקרה אם עבר זמן רב מאז הרצת המשפט, כי אז אזורי הזיכרון הללו פינו את מקומם למשפטים חדשים. ככה זה בשיטת LRU) כל השלבים שתוארו קודם יבוצעו שוב
- נניח שהמשפט הבא נשלח על ידי תוכנית הלקוח:

```
SELECT *
FROM Employees
WHERE EmployeeID = 18
```

- לכאורה נראה כי שני משפטי ה-SQL דומים, ורק התנאי שונה, ובכל זאת אלו הם שני משפטי SQL שונים לגמרי וכל התהליך שבוצע קודם יבוצע שוב עד להחזרת הרשומות לתוכנית הלקוח. מדוע? כיוון שטקסט השאילתה שונה, ייתכן שתידרש תוכנית ביצוע נפרדת.

הערה: לעתים SQL Server מסוגל להתגבר על הבעיה הזו על ידי ביצוע פרמטריזציה פנימית לשאילתה.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תהליך הביצוע של בקשה להפעלת Stored Procedure מתוך תוכנית לקוח

- נניח כי ב-Database מאוחסנת פרוצדורה בשם GetEmployee המקבלת כפרמטר מספר העובד (EmployeeID)
- תוכנית הלקוח שולחת בקשה לקבל את פרטי העובד שמספרו 10
- בתוך ה-Database:

1. נבדקת התאמת הפרמטרים להגדרות הפרוצדורה. בדיקות נוספות לתקינות ערכיהם יש לממש בקוד הפרוצדורה לפי הצורך.
2. מופעלת הפרוצדורה. אם תוכנית הביצוע שלה כבר נמצאת בזיכרון, SQL Server יכול להשתמש בה שוב, ללא צורך בבניית תוכנית ביצוע (Execution Plan) חדשה. במקרים מסוימים עשוי להידרש קימפול מחדש.

- כעת נניח כי תוכנית הלקוח מבקשת את פרטי העובד שמספרו 18. בקשה כזו לא תיחשב כבקשה שונה מהקודמת כיוון שמספר העובד מועבר כפרמטר לפרוצדורה

כתיבת Stored Procedures ב-SQL Server

- פרוצדורות ב-SQL Server נכתבות ב-Transact SQL
- Transact SQL זוהי ההרחבה של שפת SQL הסטנדרטית במערכת SQL Server (בדיוק כפי ש-PL SQL היא ההרחבה של שפת SQL במערכת Oracle)

פרוצדורה פשוטה ללא פרמטרים

```
CREATE PROCEDURE proc_name
AS
   statements...
GO
```

דוגמה 1

- הוספת Stored Procedure ל-Northwind
- (לקוח מתוך Microsoft SQL Server 7 Administrator Companion)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- הפרוצדורה LateShipments מחזירה את תאריך המשלוח הנדרש (RequiredDate), תאריך המשלוח בפועל (ShippedDate) ושם חברת המשלוחים (CompanyName) מהטבלה Shippers) עבור הזמנות שנשלחו לאחר תאריך המשלוח הנדרש
- דוגמה זו לקוחה מהספר Microsoft SQL Server 7.0 Administrator Companion

```
CREATE PROCEDURE LateShipments
AS
   SELECT RequiredDate, ShippedDate, Shippers.CompanyName
   FROM Orders, Shippers
   WHERE ShippedDate > RequiredDate AND Orders.ShipVia =
Shippers.ShipperID
GO
```

- הפעלת הפרוצדורה ב-Query Analyzer:
- אם זהו המשפט הראשון שמופעל מתוך סדרת משפטים, ניתן לרשום:

```
LateShipments
```

- אם זהו אינו המשפט הראשון יש לרשום:

```
EXECUTE LateShipments
```

או

```
EXEC LateShipments
```

פרוצדורה המקבלת פרמטרים

```
CREATE PROCEDURE proce_name @parameter1 type, @parameter2 type...
AS
   statements...
GO
```

שים לב: GO אינו משפט Transact-SQL, אלא מפריד אצוות (Batches) המוכר ל- Query Analyzer ולכלים נוספים.

דוגמה 2

- דוגמה זו זהה לדוגמה הקודמת, אלא שכאן מועבר שם חברת המשלוח כפרמטר ומוצגות רק השורות של חברה זו
- דוגמה זו לקוחה מהספר Microsoft SQL Server 7.0 Administrator Companion

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
CREATE PROCEDURE LateShipmentsByShipperName @shipperName char(40)
AS
   SELECT RequiredDate, ShippedDate, Shippers.CompanyName
   FROM Orders, Shippers
   WHERE  ShippedDate > RequiredDate
          AND Orders.ShipVia = Shippers.ShipperID
          AND Shippers.CompanyName = @shipperName
GO
```

◄ הפעלת הפרוצדורה מתוך Query Analyzer:

```
EXECUTE LateShipmentsByShipperName "Speedy Express"
```

הגדרת ערך ברירת מחדל לפרמטרים

```
CREATE PROCEDURE proce_name @parameter1 type [= deft_val],
@parameter2 type [= def_val]...
AS
   statements...
GO
```

דוגמה 3

◄ דוגמה זו מציבה בפרוצדורה הקודמת ערך ברירת מחדל עבור שם חברת המשלוחים

◄ דוגמה זו לקוחה מהספר Microsoft SQL Server 7.0 Administrator Companion

```
CREATE PROCEDURE LateShipmentsByShipperName @shipperName char(40)
= 'United Package'
AS
   SELECT RequiredDate, ShippedDate, Shippers.CompanyName
   FROM Orders, Shippers
   WHERE  ShippedDate > RequiredDate
          AND Orders.ShipVia = Shippers.ShipperID
          AND Shippers.CompanyName = @shipperName
GO
```

◄ הפעלת הפרוצדורה מתוך Query Analyzer:

```
EXECUTE LateShipmentsByShipperName
EXECUTE LateShipmentsByShipperName 'Speedy Express'
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

החזרת ערך מפרוצדורה באמצעות משתנה

```
CREATE PROCEDURE proce_name @parameter1 type [= deft_val]
OUTPUT,...
AS
   statements...
GO
```

דוגמה 4

- בדוגמה הבאה ניצור פרוצדורה בשם GetUnitPrice
- פרוצדורה זו מחזירה באמצעות המשתנה unit_price את המחיר ליחידה של מוצר שמספר הזיהוי שלו נתון ב-prod_id
- דוגמה זו לקוחה מהספר Microsoft SQL Server 7.0 Administrator Companion

```
CREATE PROCEDURE GetUnitPrice @prod_id INT, @unit_price MONEY
OUTPUT
AS
   SELECT @unit_price = UnitPrice
   FROM Products
   WHERE ProductID = @prod_id
GO
```

- הפעלת הפרוצדורה מתוך Query Analyzer:
  - יש להצהיר (באמצעות DECLARE) על משתנה שיאוחסן בו הערך המוחזר מהפרוצדורה
  - את הפרמטר יש להעביר תוך שימוש במילה OUTPUT
  - ניתן להדפיס ערך באמצעות הפונקציה PRINT
  - אם רוצים להמיר סוג נתונים לסוג אחר, ניתן להשתמש בפונקציה CONVERT

```
DECLARE @price MONEY
EXEC GetUnitPrice 77, @price output
PRINT CONVERT(VARCHAR(8), @price(
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה 5

- ניתן להשתמש במשתנה הן כדי להעביר ערך לפרוצדורה והן כדי לקבל ערך מוחזר מהפרוצדורה
- בדוגמה זו ניצור פרוצדורה בשם GetBoss המחזירה באמצעות emp_id את מספר הזיהוי של המנהל של עובד שמספר הזיהוי שלו מועבר באמצעות emp_id
- בטבלת Employees (ב-Northwind) ישנו שדה בשם ReportsTo המזהה את מספר המנהל
- אם מדובר במנהל הראשי, מוחזר המספר 0
- שים לב שניתן להשתמש במשפט תנאי IF...BEGIN...END
- כדי להציב ערך למשתנה יש להשתמש במילה SET לפני ההשמה

```
CREATE PROCEDURE GetBoss @emp_id INT OUTPUT
AS
   SELECT @emp_id = ReportsTo
   FROM Employees
   WHERE EmployeeID = @emp_id

   IF @emp_id IS NULL
   BEGIN
      SET @emp_id = 0
   END
GO
```

- הפעלת הפרוצדורה מתוך ה-Query Analyzer:

```
DECLARE @emp INT
SET @emp = 8
EXEC GetBoss @emp OUTPUT
PRINT CONVERT(VARCHAR(4), @emp)
```

שימוש במשתנים מקומיים

- ניתן להצהיר על משתנים מקומיים בפרוצדורה באמצעות DECLARE

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה 6

- הפרוצדורה הבאה מכניסה 5 שורות חדשות לטבלת PupilsTBL (שניצור בהמשך)
- בטבלה זו שני שדות: pupil_id ו-pupil_name
- בפרוצדורה נעשה שימוש בלולאה WHILE...BEGIN...END

```
CREATE PROCEDURE Insert5Rows @low_val NUMERIC
AS
   DECLARE @loop_counter NUMERIC
   SET @loop_counter = 5
   WHILE @loop_counter > 0
   BEGIN
      INSERT INTO PupilsTBL VALUES (@low_val, 'new_pupil')
      SET @low_val = @low_val + 1
      SET @loop_counter = @loop_counter - 1
   END
GO
```

- הרצה מתוך Query Analyzer:
  - תחילה ניצור את הטבלה באמצעות המשפט CREATE TABLE...
  - לאחר מכן נריץ את הפרוצדורה
  - לבסוף נציג את התוצאות באמצעות המשפט SELECT

```
CREATE TABLE PupilsTBL (pupil_id NUMERIC(4), pupil_name
NVARCHAR(20))
EXEC Insert5Rows 8
SELECT * FROM PupilsTBL
```

יציאה מפרוצדורה באמצעות RETURN

- ניתן לצאת מפרוצדורה באופן מיידי על ידי שימוש במילה RETURN
- ניתן להחזיר באמצעות RETURN ערך מסוג שלם (INT) לתוכנית הקוראת

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

דוגמה 7

- בדוגמה זו ניצור פרוצדורה בשם GetPriceCategory המקבלת מספר מוצר ומחזירה את הערכים הבאים:
  - 1- אם הועבר NULL לפרוצדורה (זהו יהיה גם ערך ברירת המחדל)
  - 0 אם המוצר אינו קיים בטבלה Products
  - 1 אם מחירו של המוצר קטן מ-20 דולר
  - 2 אם מחירו של המוצר קטן מ-30 דולר
  - 3 אם מחירו של המוצר קטן מ-50 דולר
  - 99 בכל מקרה אחר
- לצורך הדוגמה נשתמש במשפט CASE...WHEN...THEN...END

```
CREATE PROCEDURE GetPriceCategory @prod_id INT = NULL
AS
   DECLARE @price INT

   IF @prod_id IS NULL
      RETURN -1

   SELECT @price = UnitPrice FROM Products WHERE ProductID =
@prod_id

   RETURN
      CASE
         WHEN @price IS NULL THEN 0
         WHEN @price < 20 THEN 1
         WHEN @price < 30 THEN 2
         WHEN @price < 50 THEN 3
         ELSE 99
      END
GO
```

- הרצה באמצעות Query Analyzer:

```
DECLARE @cat_id INT
EXEC @cat_id = GetPriceCategory 38
PRINT @cat_id
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 12:

ActiveX Data Objects

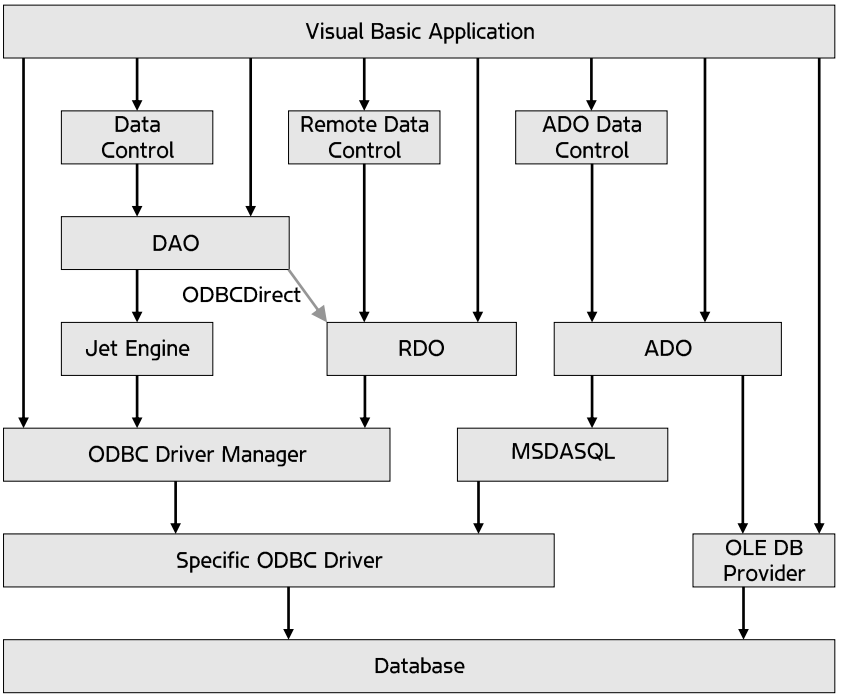
בשיעור זה:

- אסקור בפניך את אפשרויות החיבור השונות מהיישום שלך ב-Visual Basic למקורות נתונים
- תכיר את מודל האובייקטים של הספרייה ADODB
- תכיר את המאפיינים, השיטות והאירועים של האובייקטים השונים של ADO

סקירה של טכנולוגיות הגישה לנתונים

- קיימות שיטות רבות לגישה לנתונים, החל מטכנולוגיות מבוססות ממשקי קריאות API, וכלה באובייקטים ויזואליים נוחים המתממשקים למסדי נתונים
- Visual Basic תומכת באפשרויות רבות המאפשרות גישה לנתונים
- בשיעור זה נסקור אוסף הרכיבים הללו, ונראה את אפשרויות החיבור השונות
- הגרף הבא מתאר את היררכיית האובייקטים, הספריות ורכיבי הקוד השונים המפרידים בין יישום Visual Basic לבין שכבת ה-Data

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il



ODBC (Open Database Connectivity)

- ODBC מספק ממשק API לקישור עם Databases
- ODBC מהיר מאוד כיוון שהוא מבוסס CLI (Call Level Interface), משתמש המפעיל פונקציות ODBC API מנצל תקורת (Overhead) משאבים נמוכה יחסית, לעומת תקורת קוד גבוהה
- ODBC, החל מגרסה 3, תומך ב-Connection Pooling. כיוון שהתחברות לשרת Database נחשבת פעולה כבדה, ODBC מנהל חיבורים פתוחים, כך שייתכן שבקוד שלך מתבצעת פתיחה וסגירה של חיבורים ל-Databases אך בפועל אותו חיבור ב-ODBC משרת את כל החיבורים שלך
- בעת התחברות, נבחר ה-Driver הספציפי המותאם לעבודה עם ה-Database הספציפי

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- המידע אודות ה-Database, מיקומו, אופן ההתחברות אליו, שם המשתמש והסיסמה המשמשים בהתחברות אליו, שם ה-Driver המשמש לתקשורת עמו ועוד נשמרים ב-DSN (Data Source Name)
- ניתן ליצור DSN מתוך ה-Control Panel, או מתוך הקוד של התוכנית
- קיימים שלושה סוגים של DSN:
  - **User DSN**: DSN הזמין לאותו משתמש בלבד
  - **System DSN**: DSN הזמין לכל המשתמשים במערכת, כולל שירותי NT (IIS ממומש כשירות NT ולפיכך יש לשתף Databases שאמורים להשתמש בשירות זה)
  - **File DSN**: DSN המאפשר לשמור את ההגדרות בקובץ (עם הסיומת dsn.*), כך שמשתמשים ברחבי הרשת יכולים לשתף אותו ביניהם, בתנאי שיש להם Drivers מתאימים

הערה: בשני הסוגים הראשונים המידע מאוחסן ב-Registry

כרטיסיות ODBC ב-Control Panel

הערה: ב-Win2k יש להיכנס ל-Administration Tools ומשם לפתוח Data Sources (ODBC)

- **User DSN**: כרטיסייה ליצירת DSN של משתמש ספציפי
- **System DSN**: הכרטיסייה ליצירת DSN של מערכת
- **File DSN**: הכרטיסייה ליצירת DSN הנשמר בקובץ
- **Drivers**: מציג את רשימת ה-Drivers המותקנים ומידע נוסף לגביהם
- **Tracing**: מאפשר לבצע מעקב אחר פעולות ה-ODBC, כולל הפעלה משולבת יחד עם Visual Studio Analyzer המאפשר ביצוע אבחון ברשת של פעולות היישום
- **Connection Pooling**: מאפשר לקבוע את הגדרות ה-Connection Pooling עבור ה-Drivers הספציפיים השונים של ODBC

DAO (Data Access Object)

- DAO הוא ממשק תכנות מונחה עצמים הפועל על גבי מנוע Jet (מנוע Jet הוא מנוע ה-Databases היוצר ומנהל קובצי *.mdb המוכרים מ-Access)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- DAO עוצב והותאם במיוחד לעבודה מול Databases של Access
- ניתן לגשת לטבלאות מכל Database שקיים עבורו ODBC Driver, באמצעות שיטת הטבלאות המקושרות של Access
- חסרון השימוש במנוע Jet הוא בכך שכולו נטען לזיכרון, גם אם עובדים עם Database מסוג שונה מ-*.mdb
- DAO אינו חושף את כל היכולות של ODBC API
- DAO לא פותחה רבות מאז גרסת 3.5, וככל הנראה Microsoft רואה את העתיד כשייך ל-ADO

DC (Data Control)

- Data Control הוא פקד סטנדרטי של Visual Basic
- זהו ממשק ויזואלי המאפשר חיבור ל-Database
- Data Control מהווה ממשק ויזואלי (לא כ"כ מתוחכם) ל-DAO
- ניתן לכרוך פקדים (to bind) למקורות מידע המסופקים ע"י ה-Data Control
- כריכת פקדים היא פעולה שהופכת את היישום לדו שכבתי (שכבת תוכנית המשתמש ושכבת ה-Database), ולפיכך קיימים בכך חסרונות עיצוב רבים

RDO (Remote Data Object)

- RDO משלב את הפשטות של DAO עם הכוח של ODBC
- RDO מדלג על מנוע Jet, מה שמשפר את הביצועים באופן ניכר
- RDO עוצב במיוחד לעבודה מול ODBC API
- הגרסה האחרונה של RDO היא 2.0 (והיא די ישנה), כך שניתן להסיק ש-Microsoft דוחפת אותנו "בגסות" לכיוון ה-ADO

RDC (Remote Data Control)

- Remote Data Control הוא פקד ויזואלי הדומה בתפקודו ל-Data Control הסטנדרטי של Visual Basic
- כל היתרונות והחסרונות של הפקד Data Control קיימים גם ב-Remote Data Control

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ההבדל בין Data Control ל-Remote Data Control הוא בשכבת הביניים. עבור DC זהו אובייקט DAO ואילו עבור RDC זהו אובייקט RDO

ODBCDirect

- ODBCDirect הוא מצב פעולה של DAO הטוען את רכיב RDO במקום מנוע Jet
- כדי להפעיל אובייקט DAO במצב ODBCDirect יש לרשום:

```
DBEngine.DefaultType = dbUseODBC
```

- במקרה של מצב פעולה של ODBCDirect, ניתן לנצל את יתרונות ODBC ואת פשטות הקוד של DAO

OLE DB

- OLE DB מגדיר ממשקי COM סטנדרטיים שעל כל OLE DB Provider לתמוך בהם, כדי שצרכני (Consumers) מידע יוכלו להתקשר אליו באמצעות ממשקים אלו
- התחברות ישירה יכולה להתבצע באמצעות קוד בשפת C המבוסס על קריאות COM API. התחברות בדרך זו מתוך Visual Basic כמעט אינה אפשרית. לשם כך קיימים רכיבי ADO, שאותם נכיר בפרק זה.
- יתרונה הגדול של טכנולוגיה זו הוא בתמיכה בסוגי מידע שונים ולאו דווקא ב-Database-ים רגילים
- באמצעות OLE DB ניתן לגשת למידע שבכל קובץ נתונים, מסוג כלשהו, כולל מסמך Excel, הודעת דואר ועוד, ובתנאי שקיים עבורו OLE DB Provider מתאים
- השכבה היחידה שחוצצת בין ה-Database לבין היישום היא OLE DB Provider. כיוון שברכיב זה קיימת תמיכה בממשקי COM הדרושים, אין צורך בשכבה מפרידה כמו ב-ODBC
- הנתונים שאליהם ניתן לגשת ב-OLE DB הם כל הנתונים תומכי COM ולא רק נתונים סטנדרטיים של SQL

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

ADO (ActiveX Data Objects)

- ADO הוא ממשק עילי ל-OLE DB
- כיאה לממשקי COM, ADO תומך ב-OLE DB Provider המציעים ממשקים נוספים המרחיבים את יכולת פעילותו של ADO
- הפונקציונליות של ADO אינה ממוקמת ב-DLL יחיד, אלא במספר DLL-ים. לכן אין צורך להפיץ את כל
  ה-DLL-ים למשתמש הקצה אם אין שימוש בפונקציונליות של כולם
- ניתן להשתמש ב-ADO מתוך קוד HTML ו-ASP
- תת מערכת של ADO בשם Remote Data Services מאפשר שליחת רשומות ל-Client ברחבי הרשת, ואפילו הפעלת COM Components אצל ה-Client

MSDASQL

- MSDASQL הוא אחד מה-Provider-ים של OLE DB
- הייחוד ב-MSDASQL הוא בכך שהוא מהווה Provider מתאם בין OLE DB ל-ODBC
- הצורך ב-MSDASQL קיים, כיוון שישנם DBMS-ים שעדיין לא פותח עבורם OLE DB Provider מתאים

ADODC (ADO Data Control)

- ADODC הינו אובייקט ויזואלי הדומה בעיקרו ל-DC ול-RDC, אם כי ADODC מציג מספר שיפורים לעומת DC ו-RDC
- ADODC משתמש ב-ADO (OLE DB) כשכבת הביניים להתקשרות מול מקור הנתונים

תרגיל 1 – NwindDSN

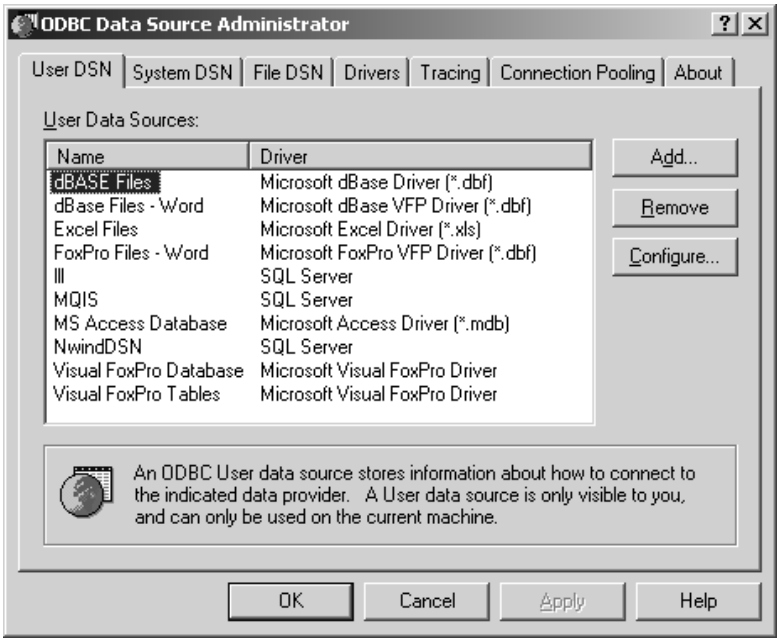
בדוגמה זו ניצור DSN באמצעות לוח הבקרה של Windows

1. **פתח את חלון ODBC Data Source Administrator:**
   - פתח את ה-Control Panel
   - ב-Windows 2000, מתוך ה-Control Panel בחר ב-Administrative Tools
   - בחר ב-ODBC (או Data Sources (ODBC) ב-Windows 2000)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

2. **הכר את הכרטיסיות השונות ביישום ODBC Data Source Administrator:**

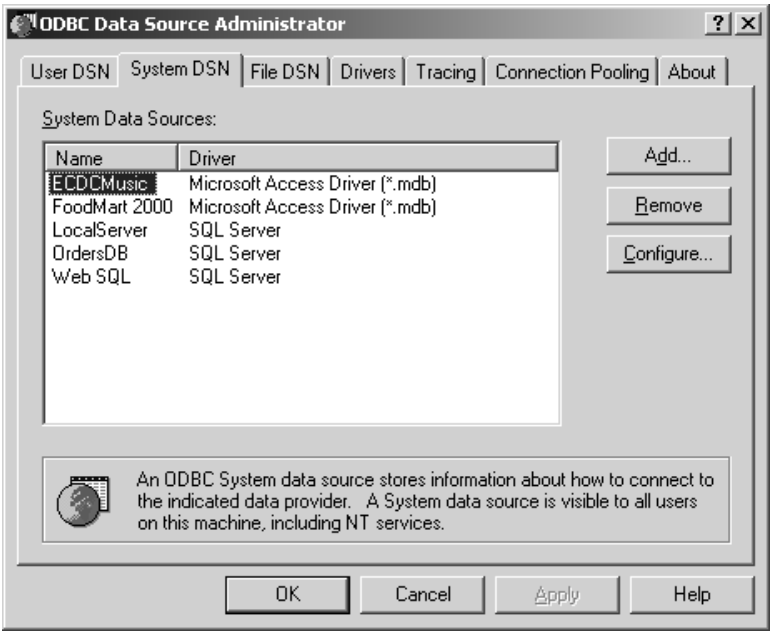
- לפני שניגש להגדרת ה-DSN בתרגיל זה, נכיר את הכרטיסיות השונות
  - **הכרטיסייה User DSN:**



בכרטיסייה User DSN אתה יכול להוסיף (Add...) DSN חדש השייך למשתמש הנוכחי בלבד במערכת (User DSN), ולהגדיר (Configure...) את המאפיינים שלו, או להסיר User DSN קיים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

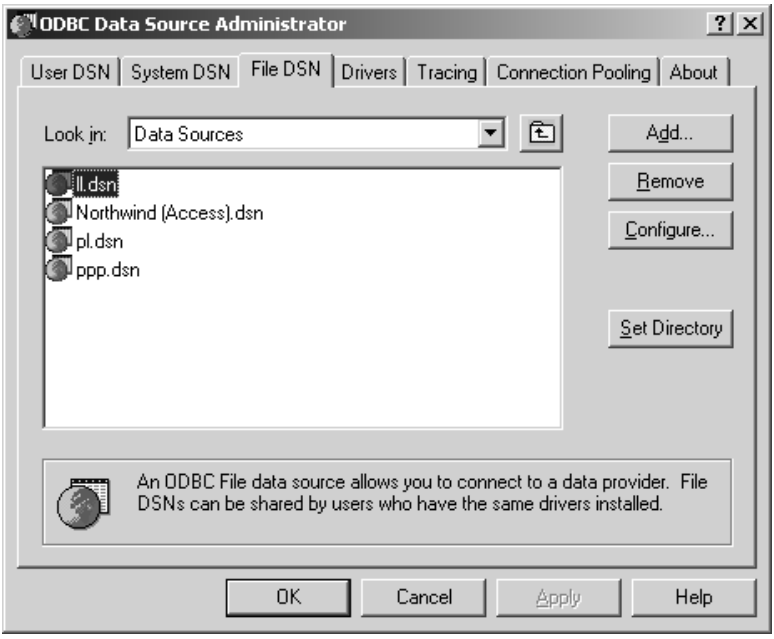
- **הכרטיסייה System DSN:**



בכרטיסייה System DSN ניתן לבצע את אותן הפעולות כמו ב-User DSN, אלא שה-DSN-ים הנוצרים ומוגדרים בכרטיסייה זו יהיו זמינים לכל משתמשי המערכת (System DSN)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

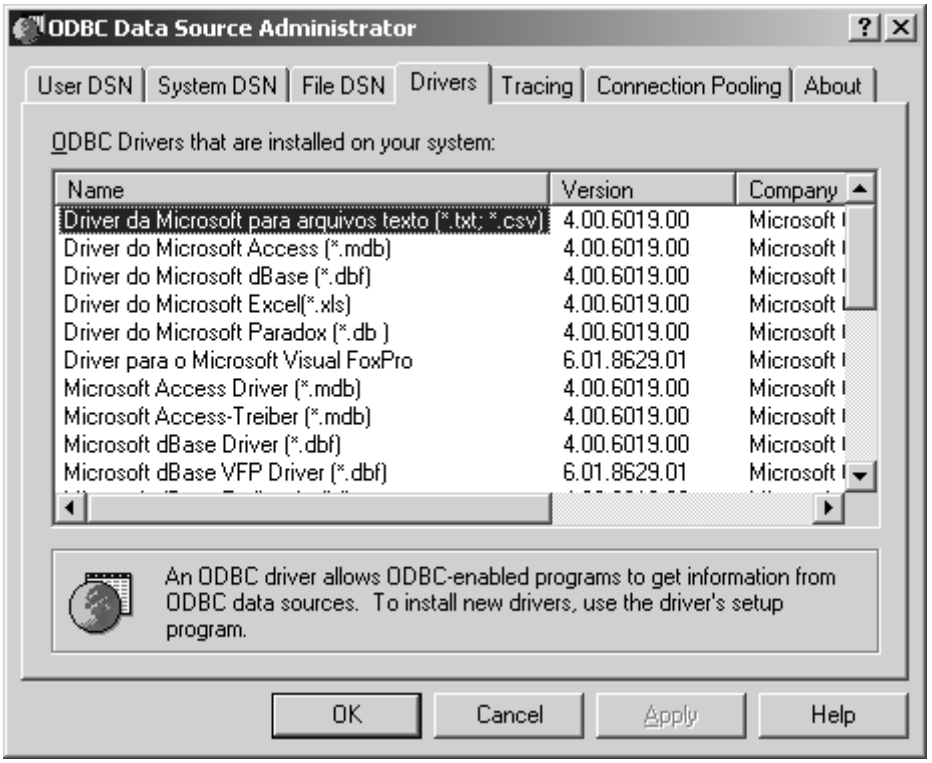
- **הכרטיסייה File DSN:**



- בכרטיסייה File DSN תוכל ליצור DSN-ים חדשים ולהגדירם, כמו גם להסיר DSN-ים קיימים. ה-DSN-ים הנוצרים בכרטיסייה זו נשמרים בקבצים (File DSN)
- קובצי File DSN נשמרים בדרך כלל בספרייה C:\Program Files\Common Files\ODBC\Data Sources. אם תרצה לשנות את מיקום קובץ ה-DSN שאתה יוצר יהיה עליך ללחוץ על Set Directory לאחר המעבר לספרייה המתאימה

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

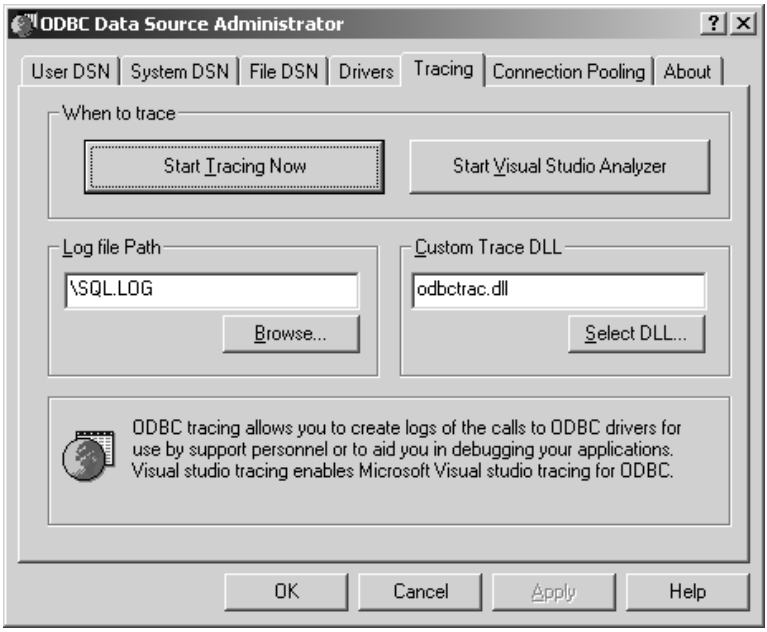
- **הכרטיסייה Drivers:**



- בכרטיסייה Drivers מוצגים ה-Driver-ים השונים של ODBC המותקנים במערכת והפרטים הבאים עבור כל Driver:
  - ✓ **Version:** מספר הגרסה של ה-Driver
  - ✓ **Company:** שם החברה המייצרת
  - ✓ **File:** שם קובץ ה-DLL של ה-Driver
  - ✓ **Date:** התאריך האחרון שבו שונה הקובץ

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

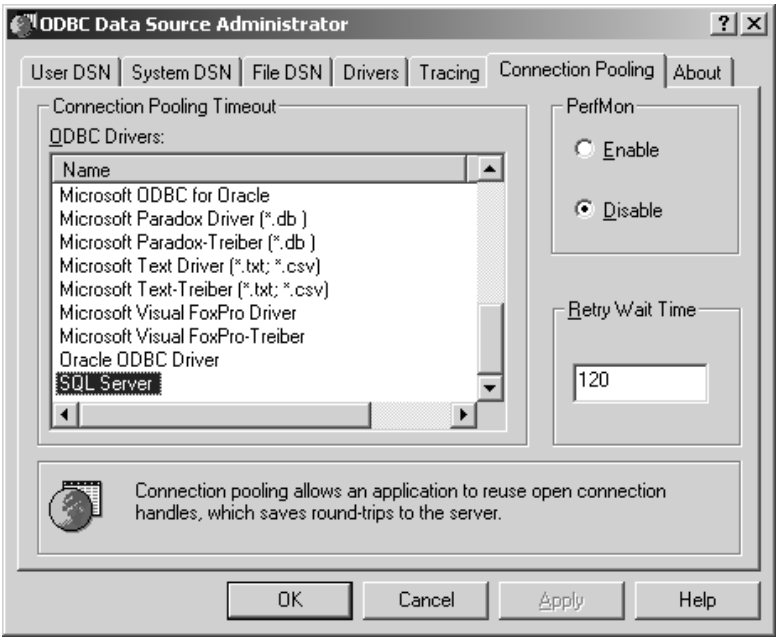
- **הכרטיסייה Tracing:**



- בכרטיסייה Tracing תוכל להפעיל מעקב אחר פעולות ODBC שיירשמו בקובץ ששמו נתון בשדה Log file Path. אפשרות זו יכולה לסייע כשעובדים באופן ישיר או עקיף (באמצעות ADO למשל) יחד עם ODBC. תוכל גם להפעיל את Visual Studio Analyzer כדי לאבחן פעולות של היישום שלך ברחבי הרשת. בשדה Custom Trace DLL תוכל לבחור DLL אחר שינהל את פעולות המעקב, אם קיים אחד כזה ברשותך

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

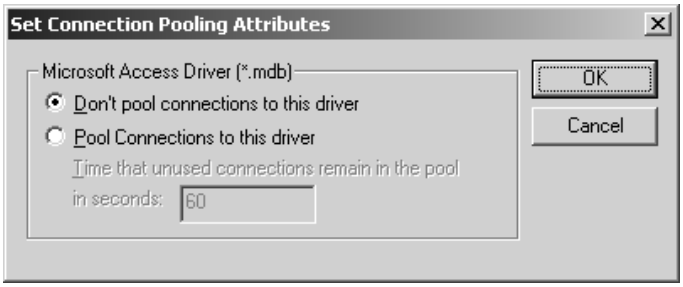
- **הכרטיסייה Connection Pooling:**



- בכרטיסייה Connection Pooling תוכל להגדיר את ההגדרות הבאות:

✓ **Connection PoolingTimeout:** בשדה זה אתה מגדיר את משך הזמן שבו יישארו Connection-ים בזיכרון לאחר סגירתם. המטרה בהשארת Connection-ים בזיכרון ה-Pooling היא לחסוך את זמן ההתחברות בעת יצירת חיבורים חדשים על ידי אותה תוכנית אם קיימים חיבורים פנויים המתאימים לדרישות החיבור החדש של התוכנית. כדי לשנות הגדרה זו עבור Driver מסוים, עליך להקליק על שמו ולהתאים את ההגדרות בתיבת הדו-שיח המתקבלת (מוצגת להלן):

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

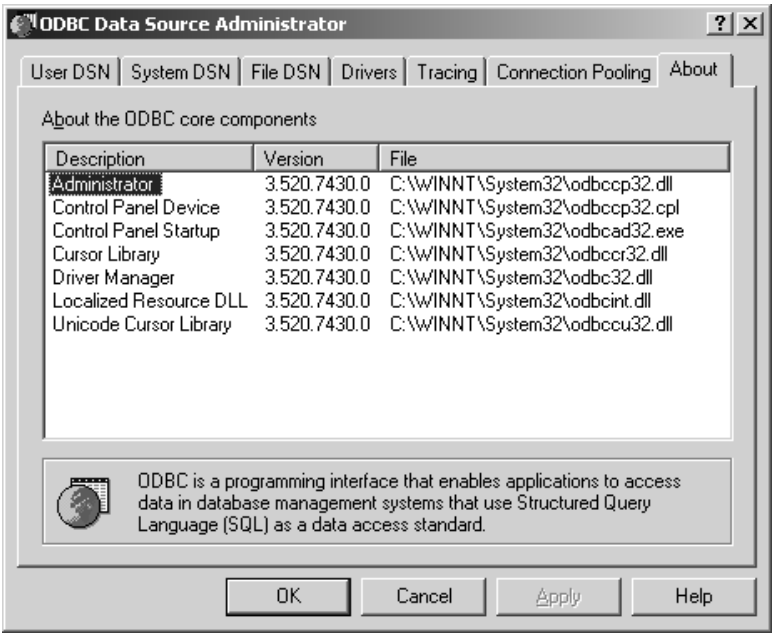


- ✓ **PerfMon (Performance Monitoring):** סימון אפשרות זו מאפשר מעקב אחר ביצועי ה-Connection Pooling של כל ה-Driver-ים
- ✓ **Retry Wait Time:** כאשר ניסיון התחברות למערכת ניהול Database-ים נכשל, מתבצע ניסיון נוסף. משך הזמן בשניות בין ניסיון אחד למשנהו מוגדר בשדה זה

- מומלץ שלא לשנות את ההגדרות של ה-Connection Pooling אלא אם אתה מודע למשמעות השינויים שאתה מבצע

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **הכרטיסייה About:**



- בכרטיסייה About תוכל לצפות במידע אודות קובצי הליבה (Core) המנהלים את תת המערכת ODBC תחת מערכת ההפעלה

3. הוסף DSN:

- ודא שהכרטיסייה UserDSN נבחרה
- לחץ על Add

4. בחר ב-Driver המתאים:

- מתוך רשימת ה-Driver-ים המוצעים בחר ב-SQL Server
- לחץ על Finish

5. הקצה שם ל-DSN:

- בשדה Name הקצה את השם NwindDSN

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

6. **הקצה תיאור ל-DSN:**
   - בשדה Description רשום DSN עבור Northwind Database
7. **קבע את מיקום ה-Server ברשת:**
   - בשדה Server בחר מהרשימה הנפתחת או הקלד ידנית את שם המחשב ברשת שבו נמצאת מערכת ה-SQL Server שאליה תרצה להתחבר באמצעות DSN זה (אם מערכת ה-SQL Server מותקנת במחשב המקומי ושמו אינו מוצג ברשימה הנפתחת, תוכל להשתמש גם בכתובת ה-IP: 127.0.0.1)
8. **בחר את שם המשתמש והסיסמה שבאמצעותם תתחבר ל-SQL Server:**
   - אם מערכת ה-SQL Server שאליה אתה מתחבר מאפשרת כניסה באמצעות שם המשתמש והסיסמה של Windows NT/2000, אזי תוכל לבחור באפשרות With Windows NT Authentication...
   - תוכל לבחור באפשרות With SQL Server Authentication... ובתחתית החלון הקלד את שם המשתמש והסיסמה באופן מפורש
   - בדוגמה זו הקלד sa עבור שם המשתמש והשאר את הסיסמה ריקה (בעת התקנת SQL Server נוסף באופן אוטומטי מנהל מערכת בשם sa ללא סיסמה, אם שינית את הסיסמה התאם את השדה Password בהתאם)
   - לחץ על Next
9. **בחר ב-Database אליו תרצה להתחבר:**
   - סמן V ב-:Change the default database to ומתוך הרשימה הנפתחת בחר Northwind
   - לחץ על Next
10. **סיים את הגדרת ה-DSN:**
    - השאר את ההגדרות במסך האחרון ללא שינוי ולחץ על Finish

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

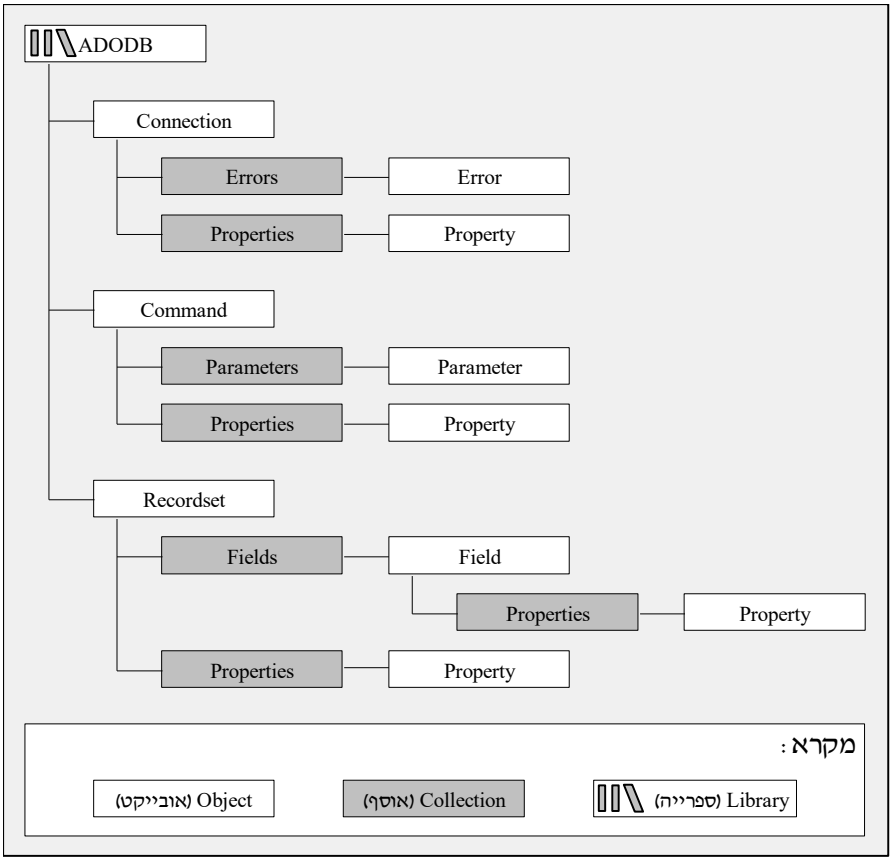
ADO (ActiveX Data Objects)

- כאמור, ADO הוא ממשק עילי ל-OLE DB
- בסעיף זה נסקור את מודל העצמים של ADO ונלמד להשתמש בכל אחד מהאובייקטים הללו
- כדי לכלול את ספריית ADO בפרויקט Visual Basic בצע את השלבים הבאים:

1. פתח את החלון References על ידי בחירת האפשרות References מהתפריט Project
2. הוסף הפנייה לספרייה *Microsoft ActiveX Data Objects 2.7*. ייתכן שבמחשב שלך הגרסה 2.7 לא תופיעה, או שתופיעה גרסה מתקדמת יותר. בכל מקרה מומלץ לבחור בגרסה המעודכנת ביותר של הספרייה.

- התרשים הבא מציג את היררכיית ה-Object-ים וה-Collection-ים שבספרייה ADODB (הספרייה ADODB היא הספרייה המכילה את האובייקטים של ADO):

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il



- בהמשך השיעור נתעכב על כל אחד מהאובייקטים והאוספים הללו

Data Link

- בדומה ל-DSN, Data Link מהווה את אוסף הפרטים הנחוצים לשם פניה למקור נתונים על ידי אובייקט ADO
- Data Link הוא קובץ טקסט הכולל פרטי מידע אודות שם ה-Provider באמצעותו ניגשים למקור הנתונים, שם מקור הנתונים ומיקומו ועוד.
- בעת בחירת Provider משתנים הפרטים הנוספים הנדרשים לפי סוג ה-Provider. כך למשל אם בחרת ב-MSDASQL (זהו ה-Provider עבור מקור נתונים של ODBC) תידרש

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

להזין את שם ה-DSN ולא יהיה צורך בציון מקור הנתונים כיוון שהוא כלול כבר בתוך ה-DSN. אם, לעומת זאת, בחרת ב-SQLOLEDB (זהו Provider עבור SQL Server) תידרש להזין את שם המחשב עליו מצוי ה-SQL Server ושם ה-Database אליו ברצונך לגשת.

יצירת קובץ Data Link

- תחת Windows 98 או Windows NT:

1. פתח את הסייר ועבור לספרייה שברצונך ליצור בה את הקובץ
2. לחץ לחיצה ימנית על העכבר במקום ריק בספריה ובחר באפשרות New
3. בחר Microsoft Data Link
4. הקצה שם לקובץ

- תחת Windows 2000 (פועל גם ב-Windows 98 ו-Windows NT):

1. פתח את הסייר ועבור לספרייה שברצונך ליצור בה את הקובץ
2. לחץ לחיצה ימנית על העכבר במקום ריק בספריה ובחר ב-New
3. בחר באפשרות Text Document
4. הקצה לקובץ שם כרצונך, ושנה את הסיומת שלו ל-udl. (אם אינך רואה את סיומת הקובץ בחר באפשרות Folder Options מתוך התפריט Tools ובכרטיסייה View נקה את הסימון ליד Hide file Extensions for known file types ולחץ על OK)
5. אל תתייחס לאזהרה שתתקבל כתוצאה משינוי הסיומת

- לאחר שיצרת את הקובץ הקלק עליו והתאם את ההגדרות בתיבת הדו-שיח Data Link Properties
- בתרגילים הבאים ובסעיף העוסק באובייקט Connection, תכיר היטב את פרטי המידע הנחוצים לשם הגדרת Data Link ואת משמעותם

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

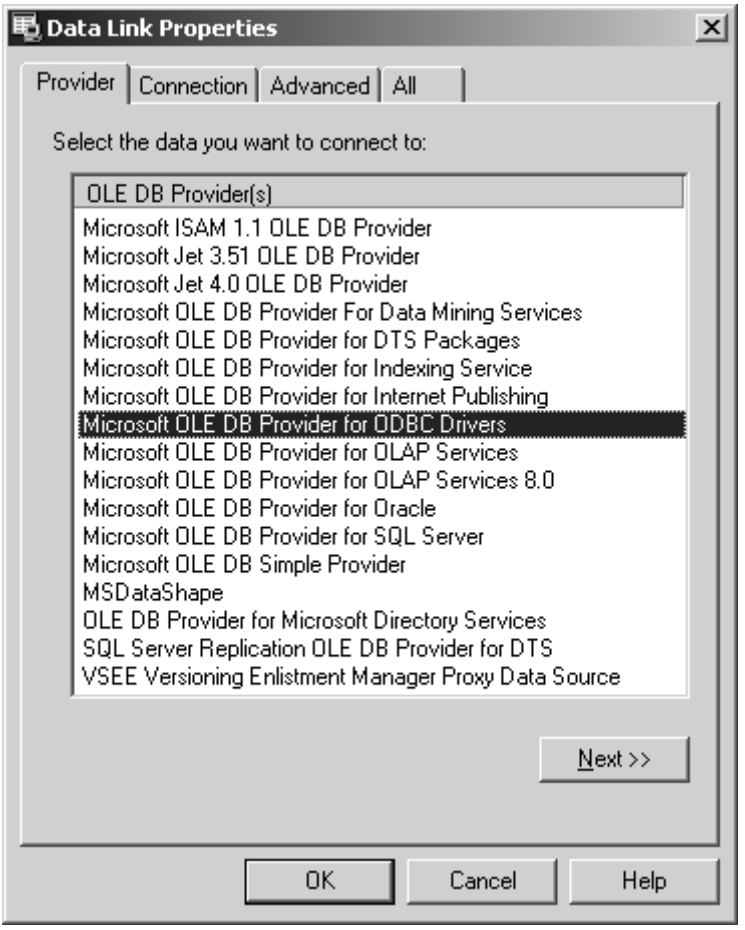
תרגיל 2: NWind.udl

בתרגיל זה ניצור קובץ Data Link שיכיל את הנתונים הדרושים לשם פניה ל-Nwind.mdb.

1. **צור קובץ Data Link ריק:**
   - צור קובץ עם הסיומת udl. כפי שהוסבר בסעיף הקודם
   - הקצה לקובץ את השם NWind.udl
2. **פתח את תיבת הדו-שיח Data Link Properties והכר את הכרטיסיות שבה:**
   - הקלק על הקובץ. תיבת הדו-שיח Data Link Properties אמורה להיפתח
   - לפני שנתאים את ההגדרות, נבחן את הכרטיסיות השונות של תיבת הדו-שיח Data Link

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **הכרטיסייה Provider:**



- בכרטיסייה Provider אתה בוחר את ה-OLE DB Provider לפי סוג מקור הנתונים שאליו אתה מתכוון לגשת:

✓ עבור Database-ים של Access בחר ב- Microsoft Jet 3.51 OLE DB Provider או גרסה 4.0 שהיא מעודכנת יותר

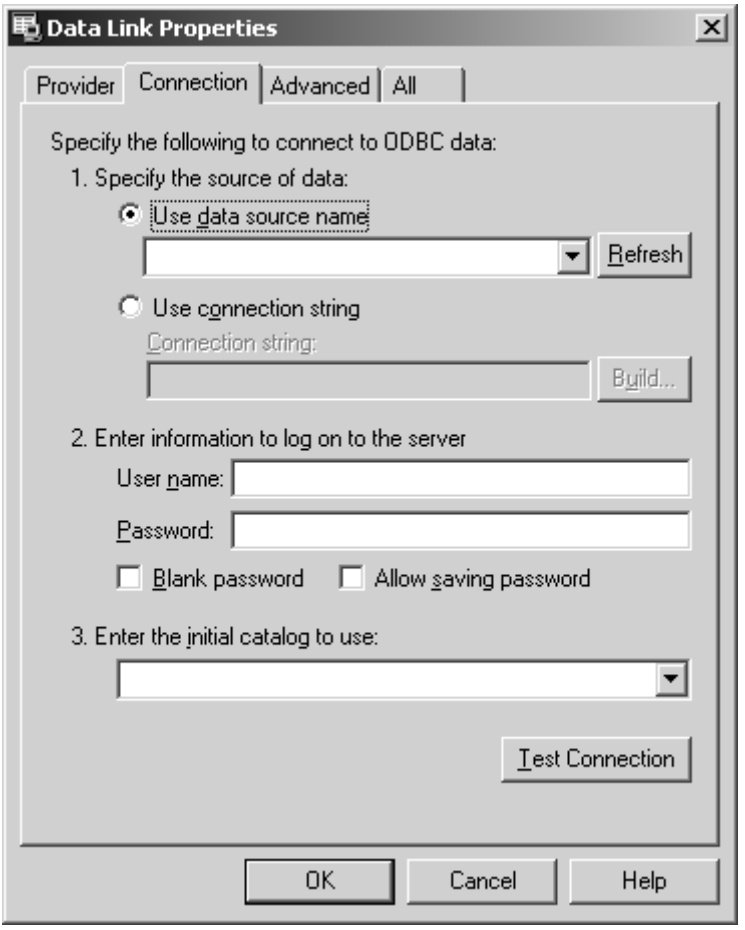
✓ עבור Database-ים של SQL Server בחר ב- Microsoft OLE DB Provider for SQL Server

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ✓ עבור Database-ים של Oracle בחר ב- Microsoft OLE DB Provider for Oracle או ב-Provider אחר המתאים ל-Oracle אם קיים כזה במחשב שלך
- ✓ עבור גישה ל-Database שלא קיים עבורו OLE DB Provider אך קיים עבורו Driver של ODBC, בחר ב- Microsoft OLE DB Provider for ODBC Drivers. שים לב ש-Provider זה מסומן כברירת מחדל אם לא בחרת Provider אחר

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

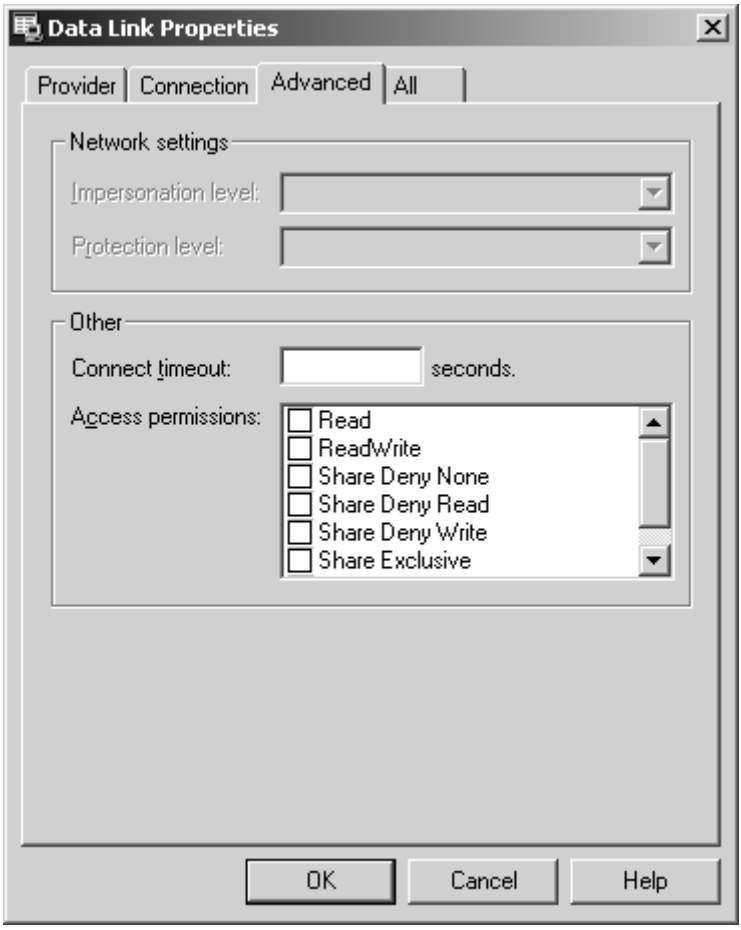
- **הכרטיסייה Connection:**



- הכרטיסייה Connection מאפשרת להגדיר את מיקום ה-Database, שם המשתמש והסיסמה וכו'.
- הפרטים בתוך כרטיסייה זו משתנים בהתאם ל-Provider שנבחר בכרטיסייה הראשונה
- את פרטי הכרטיסייה הזו תכיר בהמשך תרגיל זה ובשני התרגילים הבאים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **הכרטיסייה Advanced:**

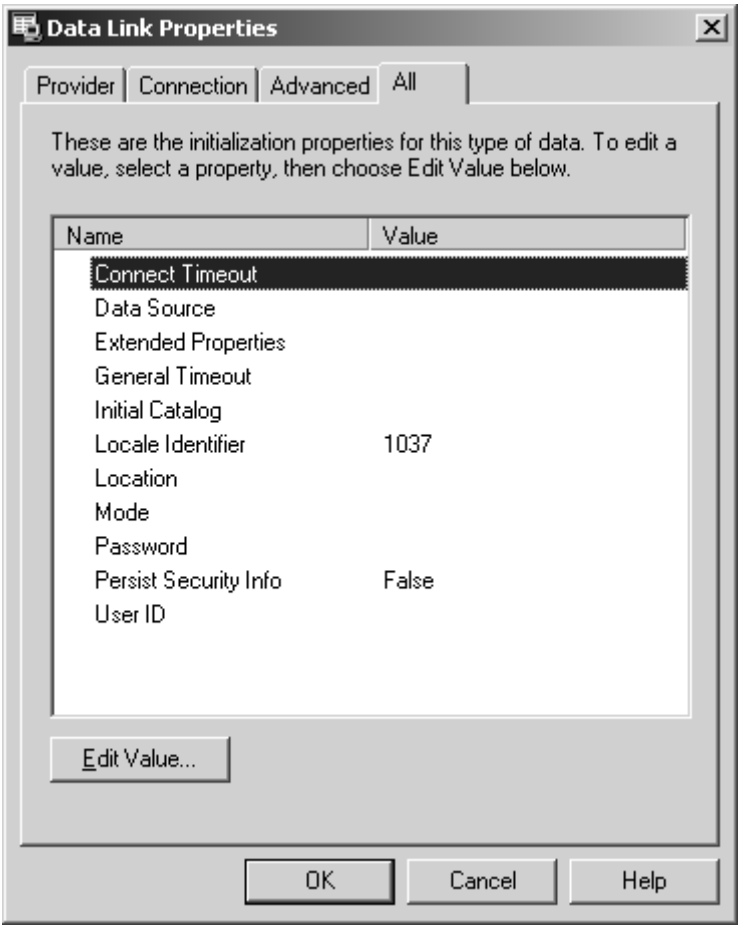


- Connect Timeout מגדיר את משך הזמן בשניות שבו יתבצע ניסיון התחברות למקור הנתונים. אם החיבור לא הצליח בפרק זמן זה תתרחש שגיאה
- Access Permissions מגדיר את הרשאות הגישה ל-Database. בסעיף העוסק במאפיין Mode של האובייקט Connection (בהמשך) תמצא הסבר אודות האפשרויות השונות

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- Impersonation Level ו-Protection Level מאפשרים הגדרות שונות עבור חיבור למקורות נתונים ברחבי הרשת המבוססים על שיטות קריאה שונות מאשר RPC (Remote Procedure Call). ברוב המקרים הגדרות אלו אינן רלוונטיות

  - **הכרטיסייה All:**



- בכרטיסייה All מוצגים כל המאפיינים של ה-Provider שבחרת בכרטיסייה הראשונה, בין שהגדרת להם ערכים ובין שלא הגדרת להם ערכים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- המאפיינים המוצגים בכרטיסייה זו שונים עבור Provider-ים שונים
- תוכל לסמן מאפיין מסוים ולהגדיר את ערכו באמצעות לחיצה על הכפתור Edit Value...
- בדרך כלל לא תצטרך להגדיר מאפיינים שאינם סטנדרטיים, אך אם בכל זאת תרצה להבין את משמעותו של מאפיין מסוים, יהיה עליך לעיין בתיעוד של ה-Provider המתאים

3. **הגדר את מאפייני ה-Data Link:**

- בכרטיסייה Provider:
  - בחר ב-Microsoft Jet 4.0 OLE DB Provider (אם ברשותך גרסה מעודכנת יותר מ-4.0 בחר בה)
- בכרטיסייה Connection:
  - בסעיף 1 (Select or enter a database name) הזן את המסלול של NWIND.MDB (קובץ זה נמצא תחת ספריית ההתקנה של Visual Basic, במחשב שלי המסלול הוא :
    C:\Program Files\Microsoft Visual Studio\VB98\NWIND.MDB, אם בחרת בספריית התקנה שונה בעת התקנת Visual Studio התאם את המסלול בהתאם)
  - לחץ על Test Connection כדי לוודא שההתחברות ל-Database מצליחה
- בכרטיסייה Advanced:
  - השאר את הגדרת ה-Access permissions כמו שהיא
- בכרטיסייה All:
  - תוכל לעיין בכרטיסייה All כדי לצפות בהגדרות שקבעת עד כה. תוכל גם לשנות הגדרות בכרטיסייה זו, או להגדיר ערכי מאפיינים שאינם מופיעים בכרטיסיות הקודמות

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

4. **אשר את ההגדרות:**
   - לחץ על OK וסגור את תיבת הדו-שיח
5. **עיין בקובץ NWind.udl:**
   - פתח את הקובץ NWind.udl באמצעות ה-Notepad ועיין במבנה הקובץ ובאופן שבו נשמרו ההגדרות השונות כפי שקבעת בכרטיסיות השונות

תרגיל 3: Pubs.udl

◄ בתרגיל זה ניצור קובץ Data Link שיכיל את פרטי המידע הדרושים לשם פניה ל-Database בשם Pubs הנמצא ב-SQL Server

1. **צור קובץ Data Link ריק:**
   - צור קובץ Data Link ריק כפי שביצעת בתרגיל הקודם והקצה לו את השם Pubs.udl
2. **הגדר את מאפייני ה-Data Link:**
   - בכרטיסייה Provider:
     - בחר ב-Microsoft OLE DB Provider for SQL Server
   - בכרטיסייה Connection:
     - בסעיף 1 (Select or enter a server name) בחר בשם המחשב בו נמצאת תוכנת ה-SQL Server שלך (אם התוכנה מותקנת על אותו המחשב בו ממוקם קובץ ה-Data Link ואינך יודע מה שמו של המחשב ברשת, תוכל להזין בשדה זה את כתובת ה-IP 127.0.0.1. כתובת זו היא כתובת ה-loopback אשר מצביעה חזרה על המחשב המקומי)
     - בסעיף 2 (Enter information to log on to the server) קיימות שתי אפשרויות:
       - **Use Windows NT Integrated security:** אפשרות זו מתאימה אם התקנת ה-SQL Server אליה אתה מתחבר מאפשרת התחברות אוטומטית

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

באמצעות שם המשתמש והסיסמה של מערכת ההפעלה Windows NT שעליה היא מותקנת

- **Use a specific username and password:** באפשרות זו אתה מזין את שם המשתמש והסיסמה ידנית. בעת התקנת SQL Server נוצר באופן אוטומטי username ששמו הוא sa בעל סיסמה ריקה. אם לא שינית את ההגדרות ב-SQL Server הזן sa לשדה username והשאר את הסיסמה ריקה. אם הסיסמה ריקה הוסף סימן V ליד האפשרות Blank password. אם אתה מזין סיסמה שאתה מעוניין שתישמר בקובץ ה-Data Link עליך לסמן V ליד האפשרות Allow Saving Password

- בסעיף 3 עליך לבחור את ה-Database אליו תרצה להתחבר. באפשרותך לציין את ה-Database בשתי צורות:

  - לסמן V ליד האפשרות Select the database on the server, ולבחור ב-Pubs מתוך רשימת ה-Database-ים
  - לסמן V ליד האפשרות Attach a database file as a database name, ולבחור את השם Pubs בשדה הראשון, ולהזין את מיקום הקובץ pubs.mdf בשדה השני (ממקם קובץ זה תחת תת הספרייה Data שתחת ספריית ההתקנה של SQL Server, בדרך כלל זהו C:\Program Files\Microsoft SQL Server\Data\Pubs.MDF)

- בשונה מ-Access יישומים "רציניים" לניהול Database-ים, למשל SQL Server, Oracle, DB2 ועוד, שומרים את הנתונים ופרטי המידע הדרושים עבור ניהולו השוטף של Database במספר קבצים. במקרה של SQL Server קיימים שלושה סוגי קבצים:

  - קובץ עם הסיומת **.MDF**: זהו קובץ הנתונים הראשי של ה-Database
  - קבצים עם הסיומת **.NDF**: אלו הם קובצי נתונים משניים של ה-Database
  - קבצים עם הסיומת **.LDF**: אלו הם קובצי רישום (Log Files) המשמשים לפעילותו השוטפת של ה-Database

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- תוכל לבדוק אם ההתחברות ל-Database מצליחה על ידי לחיצה על Test Connection

- בכרטיסייה Advanced:

- בשדה Connect timeout הקצה 30 שניות לניסיון ההתחברות
- שים לב שה-Provider עבור SQL Server אינו תומך באפשרות Access permissions

- בכרטיסייה All:

- תוכל לעיין בערכי המאפיינים שהזנת
- תוכל גם לשנות את ערכי המאפיינים, או להזין ערכים למאפיינים שאינם מופיעים בכרטיסיות Connection ו-Advanced (אני ממליץ בפניך לא לשנות ערכי מאפיינים שאינך יודע את משמעותם)

3. **אשר את ההגדרות:**

- לחץ על OK וסגור את תיבת הדו-שיח

4. **עיין בקובץ Pubs.udl:**

- פתח את הקובץ Pubs.udl, שזה עתה יצרת, באמצעות היישום Notepad, ובחן את תוכנו ואת מבנהו

תרגיל 4: NWindDSN.udl

בתרגיל זה ניצור קובץ Data Link שיכיל את פרטי המידע הדרושים לשם פניה ל-Database בשם Northwind המצוי ב-SQL Server תוך שימוש ב-DSN שיצרנו בתרגיל 1 בשם NWindDSN

1. **צור קובץ Data Link ריק:**

- צור קובץ Data Link ריק והקצה לו את השם NWindDSN.udl כפי שביצעת בשני התרגילים הקודמים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

2. **הגדר את מאפייני ה-Data Link:**
   - בכרטיסייה Provider:
     - בחר ב-Microsoft OLE DB Provider for ODBC Drivers (זוהי אפשרות ברירת המחדל)
   - בכרטיסייה Connection:
     - בסעיף 1 (Use data source name) בחר מהרשימה הנפתחת ב-NwindDSN (זהו ה-User DSN שיצרנו בתרגיל 1). תוכל גם ליצור ConnectionString אך נושא זה יילמד בהמשך השיעור.
     - בסעיף 2 (Enter information to log on to the server) הקצה שם משתמש וסיסמה מתאימים לשם התחברות ל-Database
     - לחץ על Test Connection על מנת לוודא שההתחברות מצליחה
   - בכרטיסייה Advanced:
     - תוכל לשנות את הגדרות ה-Connect timeout והגדרות ה-Access permissions (האפשרויות השונות של ה-Access permissions יילמדו בהמשך, כאשר תכיר את המאפיין Mode של האובייקט Connection). לצורך תרגיל זה אני ממליץ בפניך להשאיר את ההגדרות כמו שהן
3. **אשר את ההגדרות:**
   - אשר את ההגדרות על ידי לחיצה על OK
4. **עיין בקובץ NWindDSN.udl:**
   - באמצעות ה-Notepad פתח את הקובץ NWindDSN.udl כדי לעיין במבנהו ותוכנו של הקובץ

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

האובייקט Connection

- אובייקט Connection שולט על התקשרות אחת למקור נתונים
- אובייקט Connection משמש כמכולה לטרנזקציות מול מקור הנתונים

מאפיינים, מתודות ואירועים להכנה וביצוע התחברות למקור נתונים

הערה: לא כל המאפיינים נתמכים על ידי כל ה-Provider-ים השונים. למשל המאפיין Mode נתמך על ידי ה-Provider של Access (Microsoft.Jet.OLEDB) אך אינו נתמך על ידי ה-Provider של SQL Server (SQLOLEDB). אינך צריך לזכור את כל המאפיינים הנתמכים על ידי Provider מסוים כיוון שיצירת ה-ConnectionString הכוללת את המאפיינים נעשית בדרך כלל על ידי Wizard.

- **Provider:** מגדיר את שם ה-OLE DB Provider המשמש לביצוע ההתחברות. ברירת המחדל היא MSDASQL המאפשר התחברות באמצעות Driver של ODBC. ניתן להגדיר מאפיין זה כחלק מהמאפיינים בתוך ה-ConnectionString כפי שיוסבר להלן, או כאובייקט Property. לדוגמה כדי להגדיר את ה-Provider כ-SQLOLEDB.1 (Provider עבור SQL Server) ניתן להשתמש בכל אחת מהצורות הבאות:

```
Dim conn As New ADODB.Connection
...
...
conn.Provider = "SQLOLEDB.1"
```

או

```
conn.Properties("Provider") = "SQLOLEDB.1"
```

או

```
conn.ConnectionString = "Provider=SQLOLEDB.1;..."
```

- **ConnectionString:** מחרוזת המכילה את מאפייני החיבור השונים. המחרוזת היא במבנה של *"PropertyName1=Value; PropertyName2=Value;..."* כאשר PropertyName הוא שם של מאפיין ו-Value הוא הערך המושם לאותו מאפיין. ניתן להגדיר את מאפייני החיבור גם באמצעות האוסף Properties כפי שהודגם קודם. קיימים מאפיינים רבים עבור חיבור והם משתנים בין Provider-ים שונים. נציין מספר מאפיינים נפוצים:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Data Source:** שם מקור הנתונים או שם של DSN הקיים במערכת במקרה של התחברות למקור נתוני ODBC
- **DSN:** שם של DSN הקיים במערכת. מאפיין זה יכול להחליף את Data Source במקרה של התחברות למקור נתונים של ODBC
- **File Name:** שם של קובץ המכיל את נתוני ההתחברות. קובץ זה יכול להיות מסוג DSN עבור חיבור למקור נתונים של ODBC או שם של קובץ מסוג UDL
- **Initial Catalog:** השם של ה-Default Database
- **Database:** שם ה-Database. ניתן להשתמש במאפיין זה בעת התחברות למקור נתונים של ODBC
- **Password:** סיסמת המשתמש באמצעותו מתבצע החיבור
- **PWD:** סיסמת המשתמש באמצעותו מתבצע החיבור. ניתן להשתמש במאפיין זה בעת התחברות למקור נתונים של ODBC
- **Persist Security Info:** קובע האם פרטי אימות רגישים, כגון סיסמה, יישארו זמינים דרך פרטי החיבור לאחר פתיחתו. בדרך כלל מומלץ להשאיר את ערכו False.
- **Provider:** כפי שהוסבר קודם, זהו שם ה-OLE DB Provider באמצעותו מתבצעת ההתחברות
- **User ID:** שם המשתמש באמצעותו מתבצעת ההתחברות
- **UID:** שם המשתמש באמצעותו מתבצעת ההתחברות. ניתן להשתמש במאפיין זה בעת התחברות למקור נתונים של ODBC

- **Default Database:** מחזיר את שם ה-Default Database. מאפיין זה זמין רק לאחר שההתחברות בוצעה, ואז הוא לקריאה בלבד
- **ConnectionTimeout:** מגדיר את משך הזמן (בשניות) המקסימלי שאובייקט ה-Connection ימתין עד להשלמת ההתחברות למקור הנתונים. אם ההתחברות לא הסתיימה תוך פרק הזמן המוגדר, תתעורר שגיאה מתאימה. ערך ברירת המחדל למאפיין זה הוא 15 שניות
- **CommandTimeout:** מגדיר את משך הזמן (בשניות) המקסימלי שאובייקט ה-Connection ימתין עד להשלמת פקודה מול מקור הנתונים. אם הפקודה לא הושלמה תוך פרק הזמן המוגדר תתעורר שגיאה מתאימה. ערך ברירת המחדל למאפיין זה הוא

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

30 שניות. מאפיין זה משפיע על הפקודות המורצות באופן ישיר באמצעות Connection זה, ואינו משפיע על פקודות המורצות מאובייקט Command המשתמש באובייקט זה. לאובייקט Command ישנו מאפיין CommandTimeout משלו

- **CursorLocation:** קובע את מיקום ה-Cursor. מאפיין זה יכול לקבל אחד מהערכים הבאים:

> ***הערה:*** Cursor הינו אובייקט המכיל רשומה אחת או בלוק של רשומות מתוך אוסף הרשומות המוחזר על ידי שאילתה ממקור הנתונים

  - **adUseServer:** ה-Cursor נוצר במחשב ה-Server
  - **adUseClient:** ה-Cursor נוצר במחשב ה-Client

> **שים לב:** ניתן ליצור את כל סוגי ה-Cursor-ים ב-Server, ואילו במחשב ה-Client ניתן ליצור Cursor מסוג adOpenStatic בלבד. הסבר אודות סוגי ה-Cursor-ים תמצא בהמשך השיעור.

- **Mode:** מאפיין זה מגדיר את הרשאות ה-Connection. מאפיין זה יכול לקבל אחד או יותר מהערכים הבאים:

  - **adModeRead:** מאפשר קריאה ממקור הנתונים
  - **adModeWrite:** מאפשר כתיבה למקור הנתונים
  - **adModeReadWrite:** מאפשר קריאה ממקור הנתונים כמו גם כתיבה למקור הנתונים
  - **adModeShareDenyRead:** מונע מלקוחות אחרים להתחבר לאותו מקור נתונים לשם קריאה
  - **adModeShareDenyWrite:** מונע מלקוחות אחרים להתחבר לאותו מקור נתונים לשם כתיבה
  - **adModeShareExclusive:** מונע מלקוחות אחרים התחברות לאותו מקור נתונים בהרשאה כלשהי
  - **adModeShareDenyNone:** מאפשר ללקוחות אחרים להתחבר למקור הנתונים עם הרשאות כלשהן

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

יצירת Connection String

◄ תוכל להגדיר את ה-ConnectionString במספר דרכים:

1. לקודד בעצמך מאפיין זה – הגדרה באופן כזה קשה למדי כיוון שקשה לזכור את רשימת המאפיינים הארוכה הקיימים עבור מאפיין זה. תוכל להשתמש בטריק קטן: צור קובץ Data Link ולאחר מכן העתק את ההגדרות שבתוך הקובץ (תוכל לצפות בהן באמצעות ה-Notepad, כפי שביצעת בתרגילים הקודמים) למאפיין ConnectionString שבקוד התוכנית שלך
2. ליצור קובץ Data Link ולהגדיר אותו באמצעות המאפיין File Name ב- Connection String. כך למשל, כדי להתחבר למקור הנתונים שהוגדר באמצעות קובץ ה- Data Link מתרגיל 3 (Pubs.udl), ניתן להגדיר את ה-Connection String באופן הבא:

```
Dim conn As New ADODB.Connection
...
...
   conn.ConnectionString = "File Name=...\Pubs.udl"
```

כאשר יש לציין כמובן את המסלול המלא של הקובץ

שיטה זו מומלצת מאוד, כיוון שתוכל לשנות את הגדרת מקור הנתונים בקובץ ה-Data Link מבלי לשנות את קוד המקור של התוכנית או לקמפל את התוכנית מחדש

3. תוכל לבנות את המאפיין ConnectionString באמצעות האשפים השונים. בתיבת הדו-שיח המגדירה את מקור הנתונים עבור פקד ADODC קיים כפתור Build ליד השדה Connection String. לחיצה על כפתור זה מפעילה אשף המאפשר לך להגדיר את מקור הנתונים. לאחר סיום פעולת האשף, נוצר עבורך Connection String. תוכל להעתיק Connection String זה לתוך הקוד שלך. באחת הדוגמאות הבאות תשתמש בטכניקה זו.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

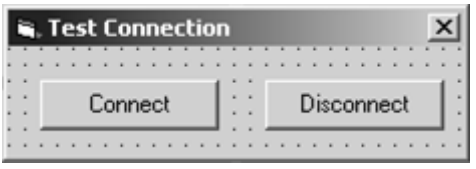
תרגיל 5 – התוכנית SimpleConnection

בתרגיל זה ניצור חיבור בין תוכנית Visual Basic לבין מקור נתונים באמצעות ADO תוך שימוש ב-DSN מוכן

1. **צור פרויקט Standard EXE ושמור אותו:**
   - הקצה לפרויקט את השם SimpleConnection ולטופס את השם frmTestConnection
2. **הוסף שני CommandButtons לטופס והגדר את המאפיינים הבאים:**

אובייקט	מאפיין	ערך
frmTestConnection	Caption	Test Connection
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdConnect
	Caption	Connect
	Enabled	True
Command2	Name	cmdDisconnect
	Caption	Disconnect
	Enabled	False

   - הטופס צריך להיראות דומה לזה שבציור:



© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

3. הוסף הפנייה לספריית ADO:

- מתוך תפריט Project בחר באפשרות References
- מתוך רשימת ספריות הרכיבים בחר בספרייה Microsoft ActiveX Data Objects 2.7 Library (ייתכן שבמחשב שלך מותקנת גרסה מוקדמת יותר או מאוחרת יותר של הספרייה. בחר בגרסה הכי מתקדמת)

4. הוסף קוד לטופס:

```
Option Explicit

Dim conn As ADODB.Connection
```

- בחלק הגלובלי של הטופס הוצהר אובייקט Connection של ADO

```
Private Sub Form_Load()
   Set conn = New ADODB.Connection
End Sub
```

- בעת טעינת הטופס נוצר אובייקט באמצעות המילה New

```
Private Sub cmdConnect_Click()
   On Error GoTo ConnectionError

   conn.ConnectionString = "DSN=NwindDSN;User ID=sa"
   conn.Open

   If conn.State = adStateOpen Then
      MsgBox "Connection successful"
      cmdConnect.Enabled = False
      cmdDisconnect.Enabled = True
   Else
      MsgBox "Connection Failed"
   End If

   Exit Sub

ConnectionError:
   MsgBox "Connection Failed: " & Err.Description
End Sub
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בעת לחיצה על Connect יוצרים Connection String עם הפניה ל-DSN שיצרנו בתרגיל הקודם וכפרמטר נוסף מעבירים את שם המשתמש
- לאחר קריאה לפרוצדורה Open בודקים את מצב החיבור. adStateOpen מצביע על חיבור מוצלח

```
Private Sub cmdDisconnect_Click()
   conn.Close
   If conn.State = adStateClosed Then
      MsgBox "Disconnected"
      cmdConnect.Enabled = True
      cmdDisconnect.Enabled = False
   End If
End Sub
```

- בעת לחיצה על Disconnect קוראים לשיטה Close כדי לנתק את החיבור ל-Database

```
Private Sub Form_Unload(Cancel As Integer)
   If conn.State = adStateOpen Then
      conn.Close
   End If
   Set conn = Nothing
End Sub
```

- בעת הסרת הטופס מנתקים את החיבור (אם הוא עדיין לא נותק) ומשחררים את אובייקט ה-ADO
- אם לא קוראים באופן מפורש ל-Close, פעולה זו תתבצע באופן אוטומטי על ידי האובייקט ADO בעת שחרורו מהזיכרון

5. בחינת הפרויקט:

- הרץ את התוכנית ובדוק את פעולתה

Event-ים של אובייקט Connection

האובייקטים השונים של ADO חושפים Event-ים שונים. לחלק מבין ה-Event-ים ישנו מבנה קבוע:

- **Will*xxxx***: אירוע זה מתרחש לפני שהפעולה *xxxx* מתרחשת

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **xxxxComplete:** אירוע זה מתרחש לאחר השלמת הפעולה *xxxx*
- דוגמאות לצמדים כאלה של אירועים הם WillConnect ו-ConnectComplete, WillExecute ו-ExecuteComplete וכו'
- בעת כניסה לאירוע Will*xxxx* מועבר פרמטר adStatus שיכול להכיל את הערכים הבאים:
  - **adStatusOK:** אין שום שגיאות
  - **adStatusErrorsOccurred:** התרחשה שגיאה כלשהי
  - **adStatusCantDeny:** לא התרחשה שגיאה, אך לא ניתן לבטל את הפעולה
- בעת כניסה לאירוע ***Willxxxx*** תוכל לשנות את ערכו של adStatus לאחד הערכים הבאים:
  - **adStatusCancel:** מורה ל-ADO לבטל את הפעולה (כאמור, לא ניתן לבצע זאת אם הערך הקודם של adStatus היה adStatusCantDeny)
  - **adStatusUnwantedEvent:** מורה ל-ADO לחדול משליחת האירוע

תרגיל 6 – התוכנית ConnectWithEvents

בתרגיל זה ניצור חיבור בין תוכנית Visual Basic ל-SQL Server באמצעות ADO וספק OLE DB מתאים. כמו כן, נשתמש באירועים של האובייקט Connection.

1. **הכנה:**
   - שלבים 1-3 בתרגיל זה, זהים לתרגיל הקודם (לתוכנית זו הקצה את השם ConnectWithEvents)
2. **צור Connection String:**
   - בתרגיל זה ניצור Connection String שישתמש ב- Microsoft OLEDB Provider for SQL Server
   - יצירת Connection String באמצעות הגדרה ידנית של מאפייני החיבור היא משימה מפרכת. לפיכך נשתמש באופן זמני באובייקט ADODC כדי ליצור את המחרוזת באמצעות Wizard:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- מהתפריט Project בחר ב-Components והוסף את הרכיב Microsoft ADO Data Control 6.0 (OLEDB) לפרויקט
- הוסף אובייקט ADODC לטופס
- לחץ לחיצה ימנית על הפקד שהוספת ובחר ב-Properties. חלון המאפיינים של ADODC אמור להיפתח
- בכרטיסייה General בחר ב-Use Connection String ולאחר מכן לחץ על הלחצן ...Build הממוקם מימין לתיבת הטקסט. ייפתח חלון שכותרתו Data Link Properties
- מתוך רשימת ה-OLEDB Providers בחר ב- Microsoft OLEDB Provider for SQL Server ולאחר מכן לחץ על Next
- בסעיף 1 בכרטיסייה Connection בחר ב-Server שבו ממוקמת מערכת SQL Server שאליה אתה מתחבר
- בסעיף 2 הזן שם משתמש וסיסמה או שתבחר ב- Use Windows NT Integrated Security אם מערכת ה-SQL שאליה אתה מתחבר מאפשרת זאת. תוכל להזין את שם המשתמש sa בצירוף סיסמה ריקה אם במערכת ה- SQL Server לא שונו ההגדרות
- בסעיף 3 בחר מתוך רשימת ה-Database-ים את Northwind
- תוכל ללחוץ על Test Connection כדי לבחון האם ה-Connection שהגדרת פועל
- לחץ על OK. כעת חזרת לכרטיסייה General
- שים לב שבתיבת הטקסט מופיעה מחרוזת. העתק מחרוזת זו ע"י סימונה ולחיצה על Ctrl+C

סגור את חלון ה-Property Pages והסר את אובייקט ה-ADODC מהטופס. תוכל להסיר גם את ההפניה ל-Microsoft ADO Data Control 6.0 (OLEDB) שהוספת קודם.

3. הוסף את הקוד הבא לטופס:

```
Option Explicit

Dim WithEvents conn As ADODB.Connection
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בשורה זו מוצהר אובייקט Connection תוך שימוש במילה WithEvents כדי שגם האירועים שלו יילכדו

```
Private Sub Form_Load()
   Set conn = New ADODB.Connection
End Sub
```

- בעת טעינת הטופס יוצרים את האובייקט

```
Private Sub cmdConnect_Click()
   On Error Resume Next
   conn.ConnectionString = "Provider=SQLOLEDB.1;" & _
                  "Persist Security Info=False;" & _
                  "User ID=sa;Initial Catalog=Northwind;" & _
                  "Data Source=WIN2K"
   Err.Clear
   conn.Open
   If Err.Number <> 0 Then
      MsgBox "Error No. " & Trim(Str(Err.Number)) & ":" & vbCrLf
& _
            Err.Description
      Err.Clear
      conn.Errors.Clear
   End If
End Sub
```

- בעת לחיצה על Connect מגדירים את ה-Connection String (**שים לב:** זוהי המחרוזת שהעתקת בשלב 2 מחלון ה-Property Pages של האובייקט ADODC שהוספת באופן זמני לטופס)
- לאחר הפעלת הפרוצדורה Open נבדק האובייקט Err כדי לבחון האם ההתחברות הצליחה, בוטלה או נכשלה מסיבה כלשהי

```
Private Sub conn_WillConnect(ConnectionString As String, _
                             UserID As String, _
                             Password As String, _
                             Options As Long, _
                             adStatus As
ADODB.EventStatusEnum, _
                             ByVal pConnection As
ADODB.Connection)
   If adStatus = adStatusOK Then
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
      If MsgBox("Do you want to cancel the operation?", _
         vbYesNo Or vbQuestion Or vbDefaultButton2) = vbYes Then
            adStatus = adStatusCancel
      End If
   End If
End Sub
```

- באירוע WillConnect מציגים למשתמש שאלה המאפשרת לו לבטל את פעולת ההתחברות
- שים לב ששאלה זו מוצגת רק אם ערך adStatus שווה ל-adStatusOK.

adStatusOK משמעותו היא שלא התרחשו שגיאות וכן ניתן לבטל את הפעולה

```
Private Sub conn_ConnectComplete(ByVal pError As ADODB.Error, _
                                 adStatus As
ADODB.EventStatusEnum, _
                                 ByVal pConnection As
ADODB.Connection)
   If conn.State = adStateOpen Then
      MsgBox "Connection successful"
      cmdConnect.Enabled = False
      cmdDisconnect.Enabled = True
   End If
End Sub
```

- ב-ConnectComplete בודקים אם ההתחברות הצליחה. אם כן מציגים הודעה למשתמש
- אם ההתחברות נכשלה, תיווצר שגיאה שתילכד ותטופל בשגרת cmdConnect_Click.

שים לב: בעת הטיפול באירוע ConnectComplete אין להסתמך על Err.Number לבדיקת כישלון ההתחברות, משום שהשגיאה של הקריאה ל-conn.Open עשויה להתעדכן בו רק לאחר החזרה מהאירוע. עם זאת, ניתן לבדוק כבר בתוך האירוע את הפרמטר adStatus, ובמקרה של כישלון לקבל את פרטי השגיאה באמצעות pError. בדוגמה זו הטיפול בשגיאה מתבצע בשגרת cmdConnect_Click, לאחר החזרה מ-conn.Open.

```
Private Sub cmdDisconnect_Click()
   conn.Close
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
End Sub
```

- בעת לחיצה על Disconnect נסגר החיבור על ידי Close

```
Private Sub conn_Disconnect(adStatus As ADODB.EventStatusEnum, _
                            ByVal pConnection As
ADODB.Connection)
   If conn.State = adStateClosed Then
      MsgBox "Disconnected"
      cmdConnect.Enabled = True
      cmdDisconnect.Enabled = False
   End If
End Sub
```

- אם פעולת הניתוק הצליחה וערכו של conn.State שווה ל-adStateClosed מציגים הודעה מתאימה למשתמש

```
Private Sub Form_Unload(Cancel As Integer)
   If conn.State = adStateOpen Then
      conn.Close
   End If
   Set conn = Nothing
End Sub
```

- בעת הסרת הטופס מנתקים את החיבור אם לא נותק עד כה

4. בחינת הפרויקט:

- הרץ את הפרויקט ובחן את פעולתו
  - אני ממליץ בפניך להריץ את התוכנית Step By Step באמצעות F8 כדי לבחון אותה מקרוב

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

האובייקט Recordset

תרגיל 7 – התוכנית DisplayEmployees

בתרגיל זה ניצור חיבור ל-Northwind ונציג את תוכן הטבלה Employees

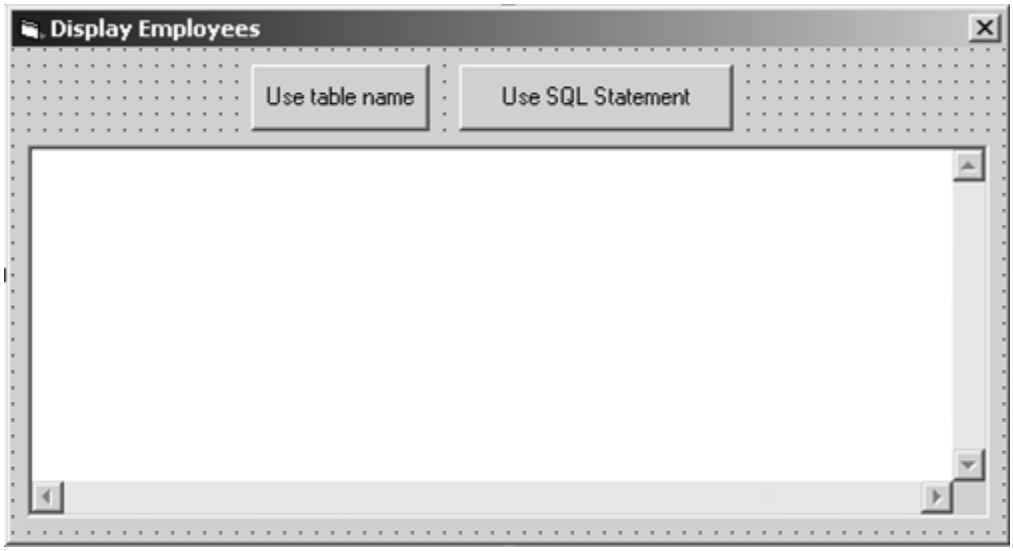
1. **צור פרויקט Standard EXE ושמור אותו:**
   - הקצה לפרויקט את השם DisplayEmployees ולטופס את השם frmDisplayEmployees
   - שמור את הפרויקט
2. **הוסף לפרויקט Reference לספריית ADO:**
   - בחר באפשרות References מתוך התפריט Project
   - סמן V ליד הספרייה Microsoft ActiveX Data Objects 2.7 Library (אם יש לך גרסה מוקדמת יותר או מאוחרת יותר, סמן V ליד הספרייה המעודכנת ביותר)
3. **הוסף שני CommandButton-ים ו-TextBox אחד לטופס, והגדר את המאפיינים הבאים:**

אובייקט	מאפיין	ערך
frmDisplayEmployees	Caption	Test Connection
	BorderStyle	3- Fixed Dialog
	StartUpPosition	2-CenterScreen
Command1	Name	cmdUseTableName
	Caption	Use table name
Command2	Name	cmdUseSQLStatement
	Caption	Use SQL statment

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Text1	Name	txtDisplay
	Multiline	True
	ScrollBars	3 - Both

- הטופס צריך להיראות דומה לזה שבציור:



4. הוסף קוד לטופס:

```
Option Explicit

Dim conn As ADODB.Connection
Dim rcs As ADODB.Recordset
```

- בשתי השורות הנ"ל מצהירים על שני אובייקטים, conn עבור התחברות ל-Northwind ו-rcs עבור אחזור של אוסף רשומות (Recordset)

```
Private Sub Form_Load()
   On Error GoTo ConnectionError

   Set conn = New ADODB.Connection
   conn.Open "Provider=SQLOLEDB.1;" & _
      "Persist Security Info=False;" & _
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
        "User ID=sa;" & _
        "Initial Catalog=Northwind;" & _
        "Data Source=WIN2K"

    Set rcs = New ADODB.Recordset
    Set rcs.ActiveConnection = conn
    Exit Sub

ConnectionError:
    MsgBox "Error: " & Err.Description
    cmdUseTableName.Enabled = False
    cmdUseSQLStatement.Enabled = False
End Sub
```

- באירוע Form_Load פותחים חיבור ל-Northwind באמצעות הפרוצדורה Open

שים לב

1. ניתן להעביר את ה-Connection String ל-Conn לפני הפעלת Open או כפרמטר ל-Open. בדוגמאות הקודמות העברנו את ה-Connection String לפני הפעלת Open
2. את ה-Connection String צור באמצעות פקד ADODC כפי שעשית בתרגיל הקודם

- לאחר יצירת החיבור, יוצרים Recordset חדש באמצעות המילה New, והופכים את החיבור conn לחיבור הפעיל (ActiveConnection) עבור האובייקט rcs

```
Private Sub cmdUseTableName_Click()
    If rcs.State = adStateOpen Then
        rcs.Close
    End If
    rcs.Source = "Employees"
    rcs.Open , , , , adCmdTable
    txtDisplay.Text = ""
    DisplayRecordset rcs
End Sub
```

- בעת לחיצה על Use table name מפעילים את הפרוצדורה Open של rcs כדי לאחזר את רשומות הטבלה Employees

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שים לב

1. ניתן להעביר את ה-Source String ל-rcs בעת הפעלת המתודה Open (כפי שתראה בשגרה הבאה)
2. את סוג הפקודה (בקשת טבלה – adCmdTable, משפט SQL – adCmdText, וכו') אין חובה להעביר. במקרה כזה, יהיה הפסד ביצועי קל כיוון שה- OLE Provider יהיה חייב להחליט באיזה סוג פקודה מדובר
3. ניתן גם להגדיר את מקור הנתונים של ה-Recordset באמצעות אובייקט מסוג ADODB.Command, על ידי השמת האובייקט במאפיין rcs.Source לפני פתיחת ה-Recordset.

- לאחר שהמשפט בוצע, מוצגות הרשומות באמצעות הפרוצדורה DisplayRecordset שנכתוב בהמשך

```
Private Sub cmdUseSQLStatement_Click()
   If rcs.State = adStateOpen Then
      rcs.Close
   End If
   rcs.Open "SELECT * FROM Employees", , , , adCmdText
   txtDisplay.Text = ""
   DisplayRecordset rcs
End Sub
```

- בדומה ל-Use table name, גם ב-Use SQL Statement נציג את תוכן הטבלה Employees, אלא שנשתמש במשפט SQL במקום שם טבלה
- שים לב שסוג הפקודה שצוין הוא adCmdText המציין כי מדובר במשפט SQL

```
Private Sub DisplayRecordset(rs As ADODB.Recordset)
   Do Until rs.EOF
      DisplaySingleRecord rs.Fields
      txtDisplay.Text = txtDisplay.Text & vbCrLf
      rs.MoveNext
   Loop
End Sub
```

- ברוטינה DisplayRecordset מציגים את תוכן הטבלה באמצעות לולאה הסורקת את אוסף הרשומות עד שמגיעים ל-EOF המציין את סיום אוסף הרשומות

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בעת פתיחת ה-Recordset הוא ממוקם אוטומטית ברשומה הראשונה, אם קיימות רשומות. אם האוסף ריק, ערכו של EOF יהיה True והלולאה לא תתבצע.
- עבור כל רשומה מעבירים את אוסף השדות שלה למתודה DisplaySingleRecord (מוגדרת בהמשך) המשמשת להצגת הרשומה הנוכחית באוסף הרשומות

```
Private Sub DisplaySingleRecord(flds As ADODB.Fields)
   Dim fld As ADODB.Field
   Dim sValue As String

   For Each fld In flds
      If IsNull(fld.Value) Then
         sValue = "<Null>"
      Else
         Select Case fld.Type
            Case adBinary, adVarBinary, adLongVarBinary
               sValue = "<Binary data>"
            Case Else
               sValue = CStr(fld.Value)
         End Select
      End If

      txtDisplay.Text = txtDisplay.Text & _
      fld.Name & " = " & sValue & _
      String(8, " ")
   Next
End Sub
```

- עבור כל רשומה, סורקים את אוסף השדות שבה, תוך הצגת שמות השדות ותוכנם
- אם ערך שדה מסוים הוא Null מציגים את המחרוזת <Null>

שים לב: לא כל ערך שדה מתאים להצגה כמחרוזת. בדוגמה אנו בודקים את סוג השדה באמצעות המאפיין fld.Type. אם מדובר בשדה בינארי (adBinary, adVarBinary או adLongVarBinary), מוצג הכיתוב <Binary data> במקום תוכנו.

```
Private Sub Form_Unload(Cancel As Integer)
   Set rcs = Nothing
   Set conn = Nothing
End Sub
```

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- באירוע Form_Unload משחררים את ה-Connection ואת ה-Recordset

5. בחינה והרצת הפרויקט:

- הרץ את התוכנית ובדוק את פעולתה

המאפיין CursorType

- **adOpenForwardOnly:** יוצר Cursor הנע קדימה בלבד. ב-Cursor מסוג זה תוכל להשתמש במתודות MoveNext, MoveFirst או Move עם פרמטר חיובי. לא תוכל לדלג לרשומה האחרונה באמצעות MoveLast או לחזור לרשומות קודמות באמצעות MovePrevious או Move עם פרמטר שלילי. Cursor מסוג Forward Only יכול להיות בצד ה-Server בלבד. ל-Cursor זה הביצועים הטובים ביותר.
- **adOpenDynamic:** יוצר Cursor המאפשר תנועה חופשית ב-Recordset. Cursor מסוג זה מעודכן תמיד בנתונים העדכניים ביותר מה-Database. בעת מעבר לרשומה מסוימת, מובאת הרשומה המעודכנת מה-Database. Cursor מסוג זה יכול להיות ממוקם בצד ה-Server בלבד. ל-Cursor זה הביצועים הגרועים ביותר.
- **adOpenKeyset:** בדומה ל-adOpenDynamic, Cursor מסוג זה מציג גם הוא את הנתונים העדכניים ביותר. עם זאת, Cursor מסוג Keyset אינו מציג רשומות חדשות שנוספו על-ידי משתמשים אחרים. Cursor מסוג Keyset יכול להיות במחשב ה-Server בלבד.
- **adOpenStatic:** Cursor מסוג זה יוצר העתק של הנתונים המאוחזרים מה-Database. ניתן לנוע בחופשיות בין הרשומות שב-Cursor סטטי. שינויים הנעשים על ידי משתמשים אחרים ברשומות לא יעודכנו ב-Cursor מסוג זה. ביצועיו של Cursor סטטי הם טובים למרות שמשך הטעינה הראשון הוא ממושך יחסית. מומלץ לעבוד עם Cursor כזה כאשר מספר הרשומות המאוחזרות קטן יחסית. Cursor סטטי הוא ה-Cursor היחיד הניתן ליצירה בצד ה-Client

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תנועה ב-Recordset

- **MoveFirst:** הופך את הרשומה הראשונה ב-Recordset לרשומה הנוכחית
- **MoveLast:** הופך את הרשומה האחרונה ב-Recordset לרשומה הנוכחית
- **MoveNext:** הופך את הרשומה הבאה לרשומה הנוכחית
- **MovePrevious:** הופך את הרשומה הקודמת לרשומה הנוכחית
- **Move(n):** הופך את הרשומה הנמצאת במרחק n מהרשומה הנוכחית, לרשומה הנוכחית. (n יכול להיות גם מספר שלילי)

Cursors (סמנים)

- Cursor הוא מנגנון המאפשר גישה לרשומות של Recordset וניווט ביניהן. אופן פעולתו תלוי במאפיינים CursorLocation, CursorType ו-LockType.
- ניתן לשלוט על שלושה מאפיינים של Cursor:
  - CursorLocation (מיקום ה-Cursor)
  - CursorType (סוג ה-Cursor)
  - LockType (סוג הנעילה)
- ניתן להגדיר מאפיינים אלו, כל עוד לא נפתח ה-Recordset באמצעות הפרוצדורה Open
- המאפיין CursorType הוסבר קודם לכן. להלן נכיר את שני המאפיינים הנוספים.

המאפיין CursorLocation

- המאפיין CursorLocation קובע היכן ינוהל ה-Cursor. קיימות שתי אפשרויות עיקריות:
  - **adUseServer:** ה-Cursor מנוהל באמצעות ספק הנתונים (Provider). זוהי ברירת המחדל.
  - **adUseClient:** ה-Cursor מנוהל בצד הלקוח באמצעות מנגנון Cursor מקומי של ADO. מצב זה מאפשר שימוש ביכולות מסוימות שאינן נתמכות בהכרח על ידי ה-Provider.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- בחירת המיקום משפיעה על היכולות הזמינות של ה-Cursor ועל אופן העבודה עם הנתונים.

המאפיין LockType

- **adLockReadOnly:** מאפשר קריאת נתונים בלבד, ללא אפשרות לעדכן רשומות באמצעות ה-Recordset. זוהי ברירת המחדל.
- **adLockPessimistic:** נעילה פסימית. נועדה למנוע ממשתמשים אחרים לשנות רשומה בזמן עריכתה, בדרך כלל באמצעות נעילתה עם תחילת העריכה.
- **adLockOptimistic:** נעילה אופטימית. הרשומה אינה ננעלת למשך כל זמן העריכה, והעדכון מתבצע בעת הפעלת המתודה Update.
- **adLockBatchOptimistic:** מאפשר לצבור שינויים במספר רשומות ולשלוח אותם למקור הנתונים כקבוצה באמצעות המתודה UpdateBatch.

> **שים לב:** לא כל ספקי הנתונים תומכים בכל סוגי הנעילה. כאשר סוג מסוים אינו נתמך, ADO עשויה להשתמש בסוג נעילה אחר.

מאפיינים נוספים

- **MaxRecords:** קובע את מספר הרשומות המקסימלי שיאוחזר מה-Database
- **CacheSize:** קובע את מספר הרשומות ש-ADO מאחזר בכל פעם מספק הנתונים ושומר בזיכרון המקומי. הגדלת ערכו עשויה להפחית את מספר הפניות לספק הנתונים ולשפר ביצועים, אך צורכת יותר זיכרון. ערך ברירת המחדל הוא 1.

האובייקט Command

- האובייקט Command מאפשר להגדיר ולהפעיל פקודות מול מקור הנתונים, כגון משפטי SQL או Stored Procedures.
  - **ActiveConnection:** מגדיר את החיבור שבאמצעותו תופעל הפקודה.
  - **CommandText:** מכיל את משפט ה-SQL או את שם ה-Stored Procedure.
  - **CommandType:** מציין את סוג הפקודה, כגון adCmdText או adCmdStoredProc.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- **Parameters:** אוסף הפרמטרים של הפקודה. כל פרמטר מיוצג באמצעות אובייקט Parameter, המאפשר להגדיר את סוג הפרמטר, כיוון העברת הנתונים והערך שלו.

◄ לאחר הגדרת הפקודה והפרמטרים שלה, ניתן להפעילה באמצעות המתודה Execute. אם הפקודה מחזירה רשומות, ניתן לקבל אותן באמצעות Recordset.

ADO Data control

◄ ADODC הוא Control המבוסס על אובייקט מסוג ADO
◄ ADODC ניתן לקשר לאוסף רשומות מ-Database בדרכים שונות
◄ ניתן להגדיר את ADODC כספק נתונים עבור פקד שהוא צרכן נתונים
◄ רוב הפקדים הסטנדרטיים תומכים ב-Data Binding, כלומר יכולים לשמש כצרכני נתונים
◄ כדי שניתן יהיה להשתמש ב- ADODCיש להגדיר אליו הפניה ב-Project Components

האובייקט Recordset

◄ הוספת רשומה חדשה

```
myadodc.Recordset.AddNew
```

◄ מחיקת רשומה

```
myadodc.Recordset.Delete
```

◄ התחלת טרנזקציה (באמצעות גישה ל-ActiveConnection מתוך ה-Recordset)

```
myadodc.Recordset.ActiveConnection.BeginTrans
```

◄ סיום תוך שמירה של הטרנזקציה

```
myadodc.Recordset.ActiveConnection.CommitTrans
```

◄ סיום ואי שמירה של הטרנזקציה

```
myadodc.Recordset.ActiveConnection.RollbackTrans
```

◄ לפי בדיקה שלי אין תמיכה ב-Transaction Nested
◄ ADODC מבצע Commit אוטומטית בסוף, אלא אם קראת ל-BeginTrans

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Properties של ADODC

- **Align:** מגדיר כיצד יושר הפקד על פני הטופס
- **BOFAction:** מגדיר מה יבוצע בעת ניסיון לחרוג מתחילת אוסף הרשומות (BOF - Beginning Of File):
  - **adDoMoveFirst:** גורם להחזרה לרשומה הראשונה
  - **adStayBOF:** נשאר ב-BOF
- **EOFAction:** מגדיר מה יבצע ה-ADODC בעת ניסיון לחרוג מתחום אוסף הרשומות (End Of File - EOF)
  - **adDoMoveLast:** מחזיר לרשומה האחרונה
  - **adStayEOF:** נשאר ב-EOF
  - **adDoAddNew:** גורם להוספת רשומה חדשה
- **ConnectionString:** מחרוזת המכילה את כל הפרטים הדרושים לשם התחברות ל-Database
- **Password-ו Username:** כוללים את שם המשתמש והסיסמה, שבאמצעותם מתחברים ל-Database
- **ConnectionTimeout:** מגדיר את משך הזמן שלאחריו ייפסק הניסיון להתחבר אל ה-Database
- **Mode:** מגדיר את אוסף הפעולות המותר מול ה-Database
- **RecordSource:** מגדיר את מקור הרשומות עבור ה-ADODC. קיימות ארבע אפשרויות:
  - **Table:** טבלה או שאילתה
  - **StoredProcedure:** שגרה מאוחסנת ב-Database
  - **Text:** משפט SQL
- **CommandType:** מגדיר את סוג ה-RecordSource
- **CommandTimeout:** מגדיר את משך הזמן שלאחריו ייפסק ניסיון להפעיל פקודה מול ה-Database
- **Caption:** כותרת הפקד

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Methods של ADODC

- **Refresh**: מרענן את התצוגה (חשוב לאחר ביצוע שינויים כגון Delete או CommitTrans או ביטול שינויים באמצעות RollbackTrans)

Events של ADODC

- **WillChangeRecord**: מתרחש רגע לפני שרשומה מעודכנת. זהו המקום לבצע בדיקת ערכים ולאשר או לבטל את השינוי
- **Error**: מתרחש כאשר מתרחשת שגיאה כלשהי. ניתן לבטל את ההודעה המוצגת ע"י שינוי ערכו של fCancelDisplay המועבר להודעה – לערך True

עיצוב הנתונים

- ניתן להשתמש במספר דרכים לעיצוב הנתונים:
  - המאפיין DataFormat של Bound Controls
  - קוד בדיקה מותאם אישית – כרוך בעבודה רבה
  - שימוש ב-StdDataFormat Objects

הערות לגבי Data Bound Controls

- כאשר משתמשים ב-Data Bound Label Control יש להגדיר את המאפיין UseMnemonics לערך False כדי שתווי אמפרסנד לא יוצגו כקו תחתי.
- CheckBox מקבל ערך vbChecked עבור ערך שונה מ-0 ב-Database, vbUnchecked לערך 0, ו-vbGrayed ל-Null
- יש לקשר CheckBox אך ורק לערך בוליאני, כיוון שיש לו שני מצבים שהמשתמש שולט בהם - True או False
- תמונות מ-Access יש לקשר ל-control OLE
- תמונות מ-SQL Server ניתן לקשר ל-Picture Box
- OptionButton אינו Data Bound Control, ולכן לא ניתן לקשר אותו ישירות לשדה ב-Database. בכל אופן, ניתן לבצע טריקים כדי לפתור את הבעיה בדרך עקיפה, וגם כמובן ליצור ActiveX מתאים

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ניתן לשנות בזמן ריצה את ה-DataSource ל-ADODC אחר. (לא קיימת אפשרות כזו ב-Data וב-RDO Data Control)

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

שיעור 13:

COM+

בשיעור זה:

- תכיר את טכנולוגיית COM+ ואת יתרונותיה בניהול טרנזקציות.
- תלמד ליצור רכיבי ActiveX DLL הפועלים תחת COM+.
- תיצור רכיב המשתתף בטרנזקציה ותבדוק את פעולתו במקרה של הצלחה וכישלון.

אודות COM+ וניהול טרנזקציות

- COM+ היא טכנולוגיה של Microsoft לניהול רכיבי תוכנה ולמתן שירותים כגון ניהול טרנזקציות ואבטחה. היא התפתחה מטכנולוגיית MTS (Microsoft Transaction Server), ושולבה ב-Windows 2000.
- טרנזקציה היא קבוצת פעולות שיש לבצע כיחידה אחת: או שכל הפעולות מצליחות, או שכולן מתבטלות.
- כאשר כל הפעולות הצליחו, ניתן לבצע Commit, כלומר לאשר את השינויים. אם פעולה נכשלה, מתבצע Rollback, כלומר ביטול השינויים שבוצעו במסגרת הטרנזקציה.
- טרנזקציה מבוזרת היא טרנזקציה שבה משתתפים כמה מקורות נתונים או משאבים, העשויים להימצא גם במחשבים שונים.
- COM+ נעזרת בשירות MSDTC (Microsoft Distributed Transaction Coordinator) לתיאום הטרנזקציות. רק משאבים התומכים בהשתתפות בטרנזקציה יכולים להיכלל בה. למשל, SQL Server או תורי הודעות טרנזקציוניים של MSMQ.

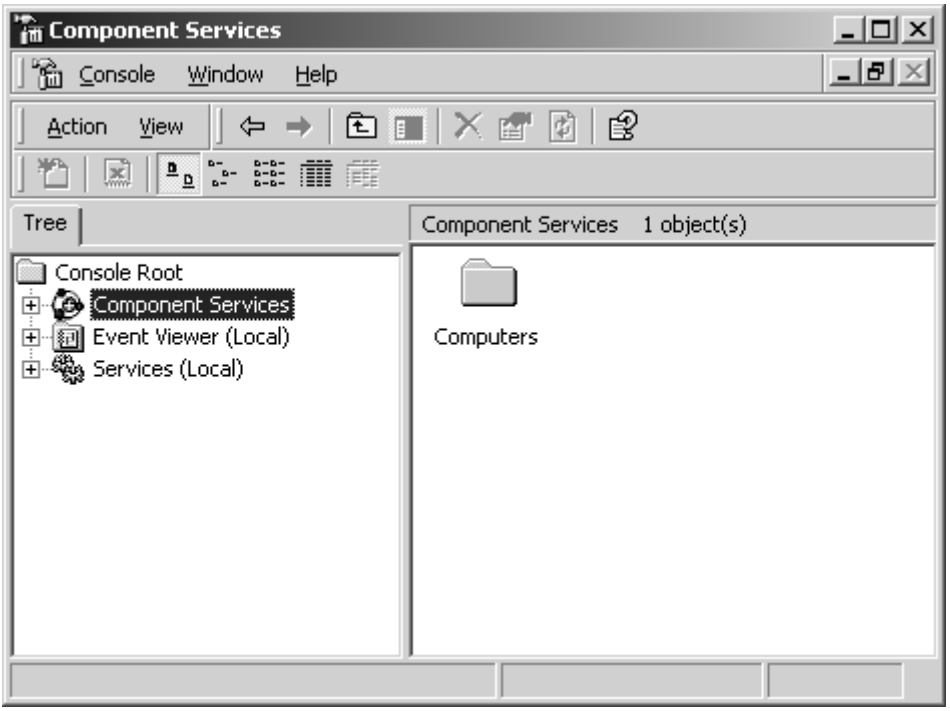
© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגיל 1: COM+Demo

- בתרגיל זה ניצור אפליקציית +COM ריקה (כלומר ללא רכיבים)
- בהמשך ניצור רכיב באמצעות Visual Basic, נוסיף אותו לאפליקציה שיצרנו ונפעיל אותו מתוך תוכנית לקוח.

1. פתח את החלון Component Services:

- פתח את ה-Control Panel ומשם עבור ל-Administrative Tools
- בחר ב-Component Services. החלון הבא אמור להיפתח:

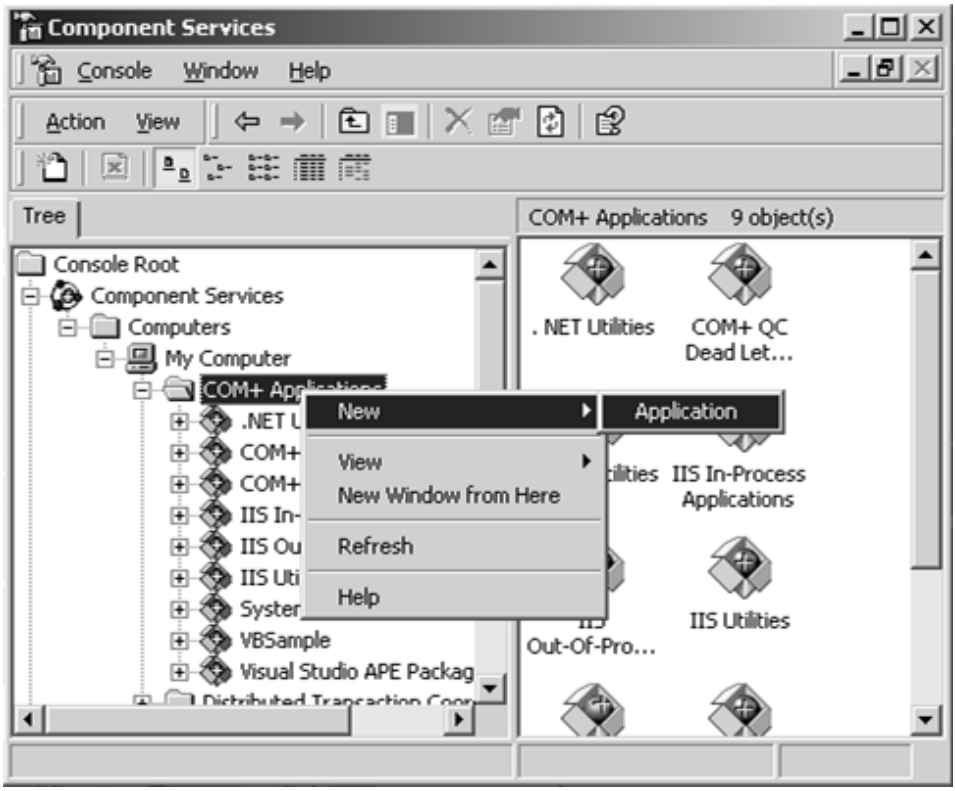


2. הוסף אפליקציית +COM ריקה:

- בפנל השמאלי הרחב את הענף Component Services, ולאחר מכן את הענף Computers.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

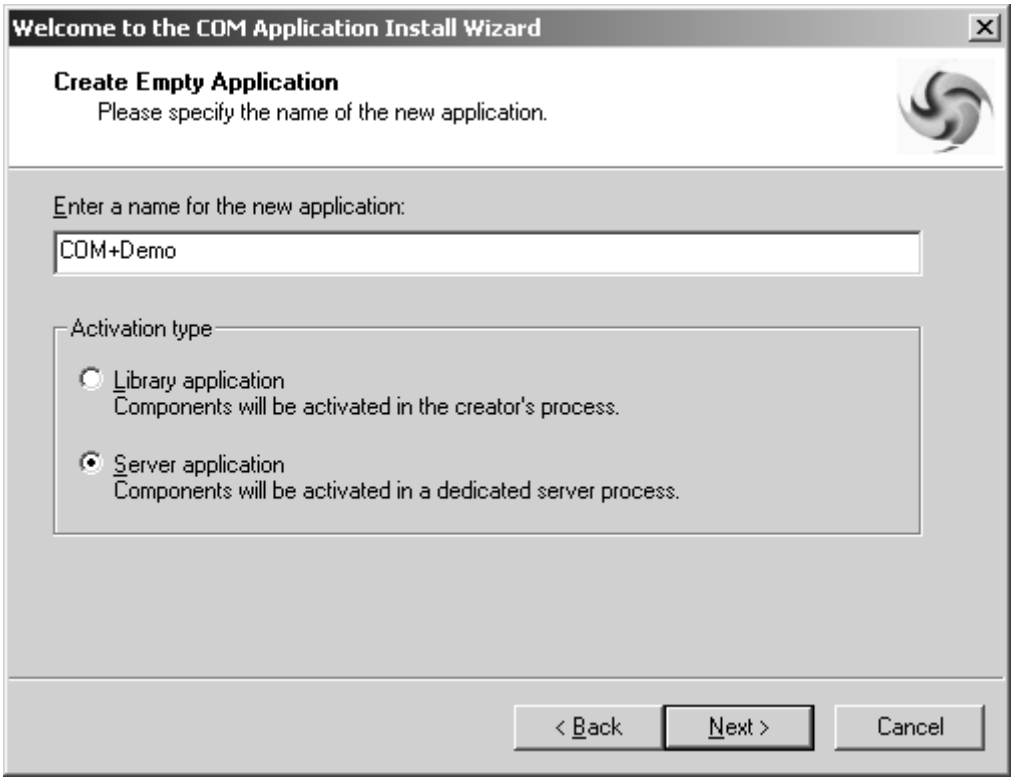
- הרחב את My Computer ובחר ב-COM+ Applications. בחלון הימני יוצגו אפליקציות ה-COM+ המותקנות במחשב.
- לחץ לחיצה ימנית על COM+ Applications ובחר ב-New ואחר כך ב-Application



- ייפתח Wizard המשמש ליצירת אפליקציית COM+:
  - במסך הראשון של ה-Wizard לחץ על Next
  - במסך השני של ה-Wizard קיימות שתי אפשרויות. האפשרות הראשונה, להתקין אפליקציה מוכנה על ידי בחירת קובץ התקנה, והאפשרות השנייה, ליצור אפליקציה חדשה. בחר באפשרות השנייה Create an empty application
  - במסך השלישי של ה-Wizard עליך להקצות שם לאפליקציה ולבחור את אופן ההפעלה של האפליקציה. קיימים שני סוגי הפעלה לאפליקציית COM+:

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

Library Application ו-Server Application. Library Application מציין כי האפליקציה תרוץ כמו DLL בתוך ה-Process שיצר רכיב מהאפליקציה. Server Application מציין כי האפליקציה תרוץ בתוך Process משל עצמה. הגדר את המאפיינים כפי שמתואר בציור שלפניך:



- בחר באפשרות Server application, כדי שהרכיבים יפעלו בתהליך (Process) נפרד.
- במסך הרביעי של ה-Wizard עליך לקבוע תחת איזה משתמש NT האפליקציה תרוץ בעת טעינתה. תוכל לבחור שם משתמש מסוים ולהזין את הסיסמה, או לבחור Interactive user שמשמעותו היא כי האפליקציה רצה תחת חשבון המשתמש המחובר למערכת

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- לצורך התרגיל בחר ב-Interactive user. האפליקציה תפעל תחת חשבון המשתמש המחובר למחשב.
- במסך האחרון של ה-Wizard לחץ על Finish

יצירת רכיבים המותאמים לפעולה תחת COM+

המאפיין MTSTransactionMode

- המאפיין MTSTransactionMode קובע כיצד אובייקט שנוצר תחת COM+ משתתף בטרנזקציות.
- המאפיין יכול לקבל את הערכים הבאים:

- **NotAnMTSObject:** המחלקה אינה מבקשת שירותי טרנזקציות באמצעות מאפיין זה (Disabled).
- **NoTransactions:** האובייקט אינו משתתף בטרנזקציה.
- **RequiresTransaction:** האובייקט משתתף בטרנזקציה קיימת, ואם אין כזו, COM+ יוצרת טרנזקציה חדשה.
- **UsesTransaction:** האובייקט משתתף בטרנזקציה של האובייקט הקורא, אם קיימת כזו. אחרת הוא פועל ללא טרנזקציה.
- **RequiresNewTransaction**: COM+ יוצרת עבור האובייקט טרנזקציה חדשה ונפרדת, גם אם האובייקט הקורא כבר משתתף בטרנזקציה.

האובייקט ObjectContext

- ObjectContext הוא אובייקט המאפשר לרכיב לקבל מידע על סביבת COM+ שבה הוא פועל, ולהשפיע על ניהול הטרנזקציה שלו.
- הפונקציה GetObjectContext מחזירה את אובייקט ה-ObjectContext של הרכיב הנוכחי.
- המתודה SetComplete מודיעה שהרכיב סיים בהצלחה את עבודתו וניתן להשלים את הטרנזקציה. השלמת הטרנזקציה תלויה גם בשאר המשתתפים בה.
- המתודה SetAbort מודיעה שיש לבטל את הטרנזקציה. במקרה כזה, השינויים שבוצעו באמצעות המשאבים המשתתפים בה יבוטלו.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

תרגיל 2: MTSSampleDLL.dll

- בתרגיל זה ניצור רכיב DLL אשר יתמוך ב-COM+
- לרכיב שניצור נוסיף מתודה בשם Transfer, המדמה העברה של 10 יחידות מחשבון אחד לחשבון אחר.
- המתודה תבצע שתי פעולות עדכון: הפחתה מחשבון אחד והוספה לחשבון האחר.
- נוסיף אפשרות לדמות כישלון, כדי לבדוק ששתי פעולות העדכון מתבטלות יחד.
- לצורך פשטות הדוגמה, נציג טיפול שגיאות מינימלי בלבד, ולא נעסוק בניהול מפורט של משאבים או באבחון שגיאות.

1. **צור פרויקט חדש מסוג ActiveX DLL:**
   - הקצה לפרויקט את השם MTSSampleDLL
   - הקצה ל-Class בפרויקט את השם MTSObj
   - בחלון המאפיינים של המחלקה ודא שהמאפיין Instancing מוגדר כ-MultiUse - 5, כדי שתוכנית לקוח תוכל ליצור מופעים של המחלקה.
   - שמור את הפרויקט
2. **התאם את הגדרות הפרויקט:**
   - מתוך התפריט Project בחר ב-MTSSampleDLL Properties..., כך שיוצג חלון המאפיינים של הפרויקט
   - בכרטיסייה General סמן Unattended Execution, מכיוון שהרכיב מיועד לפעול ללא ממשק משתמש.
   - בכרטיסייה General ודא כי ב-Threading Model נבחרה האפשרות Apartment Threaded
3. **הוסף הפניות לספריות רכיבים בפרויקט:**
   - כיוון שהפרויקט משתמש ברכיבי ADO וברכיבי COM+ כדי לגשת למקורות הנתונים יש להוסיף הפניות מתאימות בחלון ה-References
   - בחר באפשרות References... מתוך התפריט Project

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- סמן V ליד הספרייה Microsoft ActiveX Data Objects 2.7 או גרסה אחרת (בחר את הגרסה המעודכנת ביותר במחשב שלך)
- סמן V ליד הפריט COM+ Services Type Library

4. הגדר את המאפיין MTSTransactionMode:

- האפשרויות השונות של המאפיין MTSTransactionMode הוסברו קודם
- בחר באפשרות RequiresTransaction - 2
- כך אנו מבטיחים שהאובייקט שלנו ירוץ בתוך טרנזקציה. אם האובייקט נקרא על ידי אובייקט אחר שיש לו טרנזקציה, האובייקט שניצור ישתתף בטרנזקציה של אובייקט האב. אם אין לאובייקט האב טרנזקציה משלו, תיווצר טרנזקציה חדשה עבור האובייקט

5. הוסף קוד למחלקה:

- במודול המחלקה MTSObj הוסף את הקוד הבא:

```
Option Explicit

Public Sub Transfer(ByVal Fail As Boolean)
   Dim cn As ADODB.Connection

   On Error GoTo Failed

   Set cn = New ADODB.Connection
   cn.Open "Provider=SQLOLEDB;Data Source=(local);" & _
      "Initial Catalog=COMPlusDemo;Integrated Security=SSPI;"


   cn.Execute "UPDATE Accounts SET Balance=Balance-10 WHERE Id=1"
   cn.Execute "UPDATE Accounts SET Balance=Balance+10 WHERE Id=2"

   If Fail Then Err.Raise vbObjectError + 1000, , "Test failure"

   cn.Close
   GetObjectContext.SetComplete
   Exit Sub
```


© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

```
Failed:
   GetObjectContext.SetAbort
   Err.Raise vbObjectError + 1001, , "Transaction aborted"
End Sub
```

- המתודה פותחת חיבור ל-SQL Server ומבצעת שתי פעולות עדכון באמצעות ADO.
- מכיוון שהרכיב פועל תחת +COM והחיבור משתמש בספק OLE DB מתאים, החיבור מצטרף אוטומטית לטרנזקציה של הרכיב.
- אם Fail הוא True, אנו מדמים שגיאה לאחר שתי פעולות העדכון.
- אם לא התרחשה שגיאה, מתבצעת קריאה ל-SetComplete. אם התרחשה שגיאה, מתבצעת קריאה ל-SetAbort כדי לבטל את הטרנזקציה.

6. קמפל את הרכיב:

- מתוך התפריט File בחר ב-Make MTSSampleDLL.dll...
- תקן שגיאות במידת הצורך

תרגיל 3: הוספת MTSSampleDLL.dll ל-+COM

1. פתח את האפליקציה COM+Demo שיצרת קודם:

- פתח את Component Services, כפי שלמדת בתרגיל 1.
- הרחב את Component Services --> Computers --> My Computer --> +COM Applications.
- הרחב את האפליקציה COM+Demo ובחר בענף Components.

2. הוסף את הרכיב לאפליקציה:

- לחץ לחיצה ימנית על Components, בחר ב-New ולאחר מכן ב-Component.
- בחלון ה-Wizard שייפתח לחץ על Next, ולאחר מכן בחר Install new components.
- לחץ על Add, בחר בקובץ MTSSampleDLL.dll שקימפלת קודם ולחץ על Open.
- לחץ על Next ולאחר מכן על Finish כדי להשלים את ההתקנה.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

3. **בדוק את הגדרות הרכיב:**
   - ודא שהרכיב MTSSampleDLL.MTSObj מופיע בענף Components של האפליקציה.
   - לחץ לחיצה ימנית על הרכיב ובחר Properties.
   - בכרטיסייה Transactions ודא ש-Transaction support מוגדר כ-Required.

שים לב: אם תשנה את הרכיב ותקמפל אותו מחדש לאחר התקנתו, ייתכן שיהיה עליך להתקינו מחדש ב-COM+.

תרגיל 4: בחינת פעולת MTSSampleDLL.dll תחת COM+

1. **הכנת מסד הנתונים**
   - ודא ש-SQL Server מותקן ופועל במחשב המקומי, וששירות MSDTC זמין.
   - ודא שלחשבון המשתמש שתחתיו פועלת אפליקציית COM+ יש הרשאת גישה ועדכון למסד הנתונים באמצעות Windows Authentication.
   - פתח את SQL Query Analyzer וצור מסד נתונים וטבלת בדיקה באמצעות הקוד הבא:

```
CREATE DATABASE COMPlusDemo
GO

USE COMPlusDemo
GO

CREATE TABLE Accounts (
   Id int NOT NULL PRIMARY KEY,
   Balance int NOT NULL
)
GO

INSERT INTO Accounts VALUES (1, 100)
INSERT INTO Accounts VALUES (2, 100)
GO
```

   - הטבלה מכילה שני חשבונות, ובכל אחד מהם יתרה התחלתית של 100.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

2. יצירת תוכנית הלקוח

- צור פרויקט חדש מסוג Standard EXE.
- הוסף לטופס פקד CheckBox בשם chkFail.
- הוסף פקד CommandButton בשם cmdRun.
- באירוע Click של הכפתור הוסף את הקוד הבא:

```
Private Sub cmdRun_Click()
   Dim obj As Object

   On Error GoTo Failed

   Set obj = CreateObject("MTSSampleDLL.MTSObj")
   Call obj.Transfer(chkFail.Value = vbChecked)

   MsgBox "Transaction completed"
   Exit Sub

Failed:
   MsgBox Err.Description
End Sub
```

3. בדיקת הצלחת הטרנזקציה

- הרץ את התוכנית כש-chkFail אינו מסומן.
- לחץ על הכפתור cmdRun.
- ודא שמופיעה ההודעה Transaction completed.
- חזור ל-SQL Query Analyzer ובצע:

```
SELECT Id, Balance FROM COMPlusDemo.dbo.Accounts ORDER BY Id
```

- ודא שהיתרות הן 90 בחשבון הראשון ו-110 בחשבון השני.

4. בדיקת ביטול הטרנזקציה:

- סמן את chkFail ולחץ שוב על cmdRun.
- הפעם תתרחש שגיאה יזומה, והמתודה תקרא ל-SetAbort.
- ודא שמופיעה ההודעה Transaction aborted.
- בצע שוב את משפט ה-SELECT הקודם.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il

- ודא שהיתרות נותרו 90 ו-110. שתי פעולות העדכון של ההרצה השנייה בוטלו.

- **לסיכום:** הרכיב שיצרנו משתתף בטרנזקציה המנוהלת על ידי COM+. במקרה של הצלחה שתי הפעולות נשמרות, ובמקרה של כישלון שתיהן מתבטלות. בדוגמה השתמשנו במסד נתונים אחד, אך COM+ מסוגלת לתאם טרנזקציות גם בין מספר משאבים מתאימים.

יכולות נוספות של COM+

- **Roles (תפקידים):** COM+ מאפשרת להגדיר תפקידים ולהקצות להם משתמשים. כך ניתן לקבוע אילו משתמשים רשאים להפעיל רכיבים ומתודות מסוימות.
- **IsCallerInRole:** מתודה של ObjectContext המאפשרת לבדוק אם המשתמש הקורא לרכיב שייך לתפקיד מסוים.
- **ConstructString:** מחרוזת אתחול שניתן להגדיר במאפייני הרכיב ב-COM+. רכיב המממש את הממשק IObjectConstruct יכול לקבל אותה בעת יצירתו. למשל, לצורך העברת הגדרות התחברות למסד נתונים.
- **Queuing:** באמצעות MSMQ ניתן להפעיל רכיבי COM+ גם באופן אסינכרוני, באמצעות תור הודעות. הבקשה נשמרת בתור ומטופלת מאוחר יותר, כאשר השירות זמין.

© 2001 - כל הזכויות שמורות לאילן אמויאל
אין להעתיק ללא רשות בכתב מאת המחבר
kita.co.il