FIT1013 - Week 8 Resources

User Forms

Week 8 Resources

Reference:

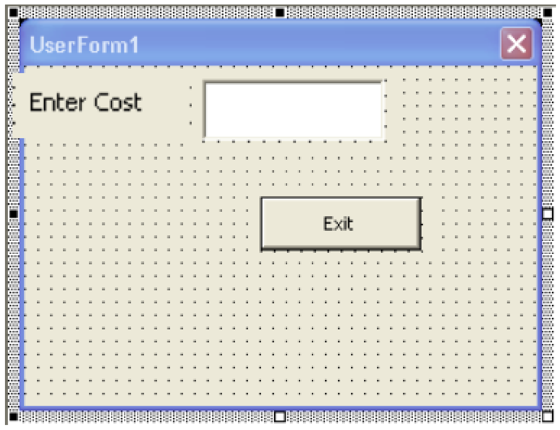
- https://msdn.microsoft.com/

1. Objectives

- Create a user form
- Add controls to a form
- Explain the use of text box, label, and command button controls
- Provide keyboard access to controls using accelerator keys
- Code a user form

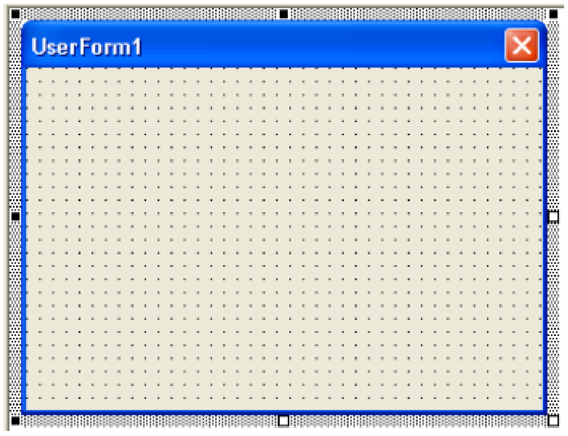
2. Creating Custom Dialog Boxes (User forms)

- You first add a form - the foundation of a dialog box, to the project, and then you add objects, called controls, to the form
- This form and its controls are what constitute a user form or dialog box



Design Standards for Dialog Boxes

- Before creating a custom dialog box, we will look at the Windows standards for dialog boxes:
    - When positioning the controls, be sure to maintain a consistent margin from the edge of the form; a margin of two or three dots is recommended
    - Because a dialog box is a window, it has a title bar and borders



Design Standards for Dialog Boxes

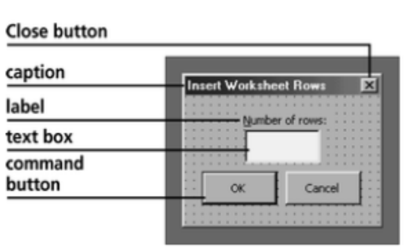
in, on, at, to, into, from, out of

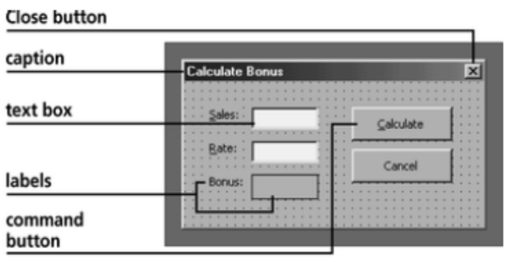
because, and, but, as, since, though, although

Caption:

- The dialog box's caption should be entered using **book title capitalisation**, which means you capitalise the first letter in each word, except for articles, conjunctions, and prepositions that do not occur at either the beginning or the end of the caption
- Examples: "a", "an", "the" and "and".

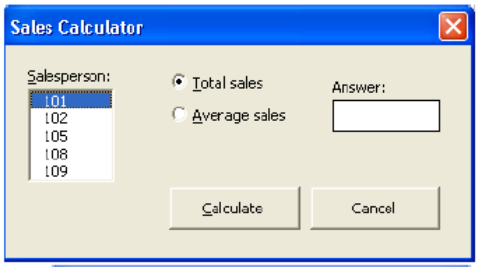
the, a, an , some, any

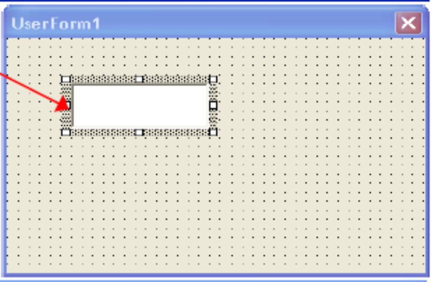


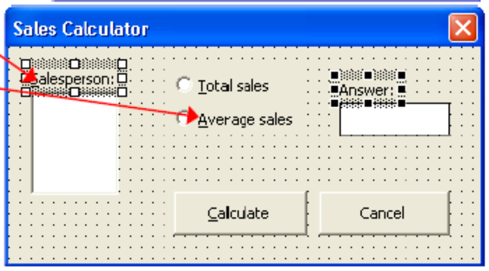


Dialog Box Controls

- You use a **text box control** to provide an area in the dialog box where data can be entered, edited, and displayed
- You use a **label control** to display text that you don't want the user to modify, such as text that identifies another control in the dialog box or text that represents the result of a calculation
- If a label control is used as an identifier for another control, its caption should be **no more than three words** in length and entered using **sentence capitalisation**, which means that you capitalise only the first letter in the first word and in any words that are customarily capitalised.

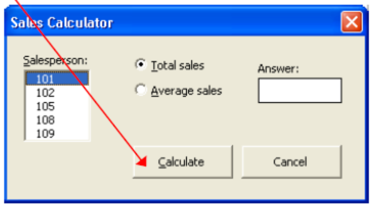






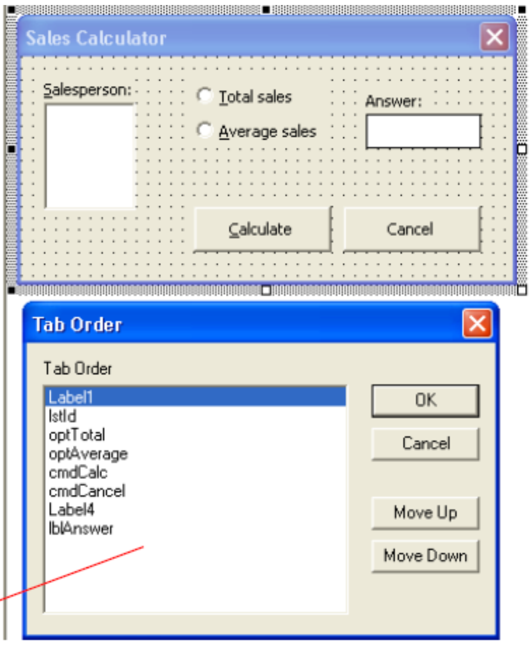
Dialog Box Controls

- You use a **command button** control to process one or more instructions when the user clicks the button
- A command button's caption should be no more than three words in length and entered using book title capitalisation
- Command buttons should be positioned either at the **bottom** or on the **right side** of the dialog box



Setting the Tab Order

- An **essential control** is one that can receive input from the user
  - E.g. text box, option button.
- The **tab order** is the order in which the focus moves from one **essential control** in a dialog box to the next **essential control** as you press the Tab key
- The **first** essential control in the tab order is typically located in the upper-left area of the dialog box

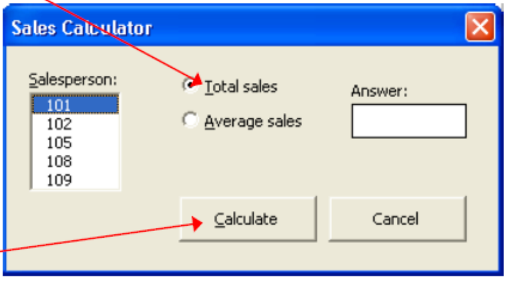


Label1
lstId
optTotal
optAverage
cmdCalc
cmdCancel
Label4
lblAnswer

Right click on form to view tab order

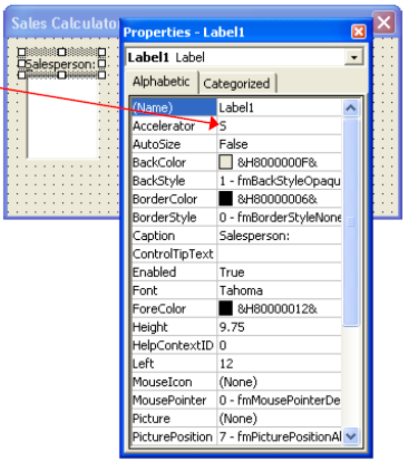
Providing Keyboard Access to a Control

- Providing keyboard access to the controls in a dialog box allows the user to work with the dialog box using the keyboard rather than the mouse
- The user may need to use the keyboard if his or her mouse becomes inoperative
- The user simply may prefer to use the keyboard if he or she is a fast typist



Assigning Accelerator Keys

- The underlined letter is called an **accelerator key** and it is used in combination with the **Alt key** as a shortcut for selecting a control
- In Excel, you use a control's Accelerator property to assign an accelerator key to the control
- **Note:** the accelerator key for a label control takes the focus to the next essential control in the tab order
- [Computer Inventory complete do loop Show.xls]()



Properties - Label1

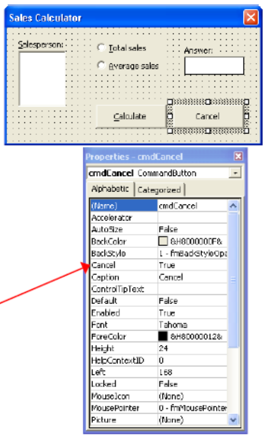
Label1 Label

Alphabetic | Categorized

(Name)	Label1
Accelerator	S
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - fmBackStyleOpaqu
BorderColor	■ &H80000006&
BorderStyle	0 - fmBorderStyleNone
Caption	Salesperson:
ControlTipText	
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	9.75
HelpContextID	0
Left	12
MouseIcon	(None)
MousePointer	0 - fmMousePointerDe
Picture	(None)
PicturePosition	7 - fmPicturePositionAl

Using the Default and Cancel Properties

- The **default button** is the one that is selected automatically when the user presses the Enter key, even when the button does not have the focus
- You make a command button the default button by setting its **Default** property to the **Boolean value True**
- The **cancel button** is the one that is selected automatically when the user presses the **Esc** key
- You make a command button the cancel button by setting its **Cancel** property to the Boolean value True



Properties - cmdCancel

cmdCancel CommandButton

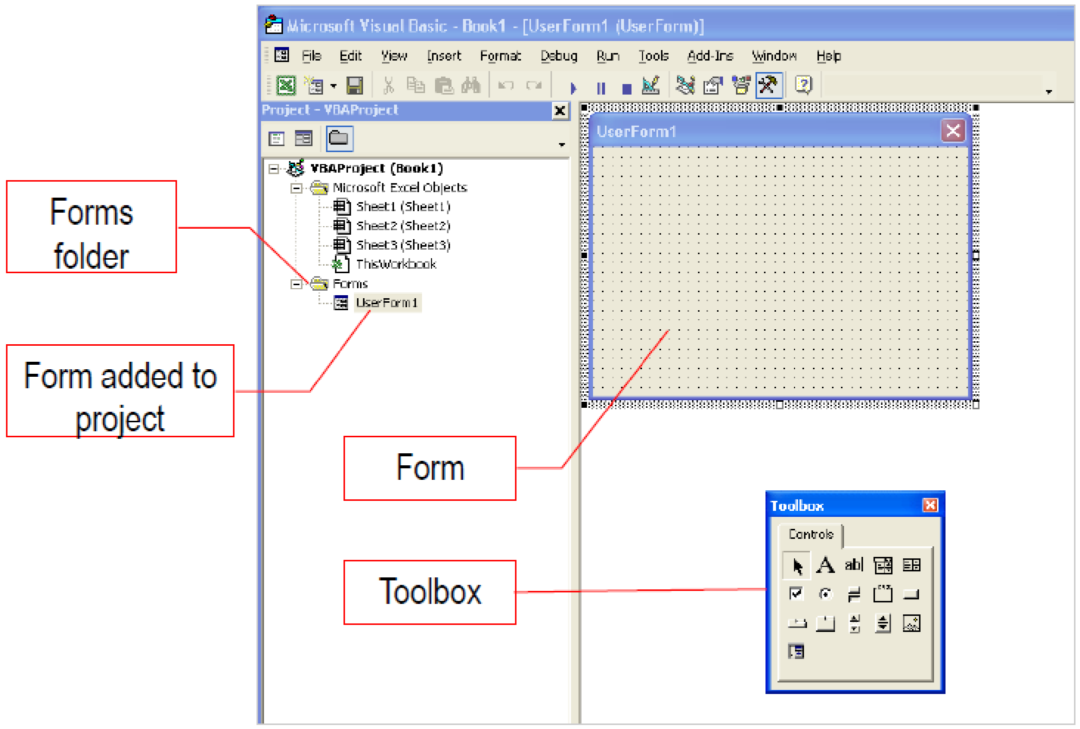
Alphabetic | Categorized

(Name)	cmdCancel
Accelerator	
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - fmBackStyleOpa
Cancel	True
Caption	Cancel
ControlTipText	
Default	False
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	24
HelpContextID	0
Left	168
Locked	False
MouseIcon	(None)
MousePointer	0 - fmMousePointer
Picture	(None)

Adding a Form to the Project

- Before you can create a custom dialog box, you first must add a form to the project
- The form will serve as the foundation of the dialog box
- To add a form to your project:
  - Open the VBE
  - Click Insert on the menu bar
  - Click UserForm
    - VBE adds a form to the project and also displays the toolbox

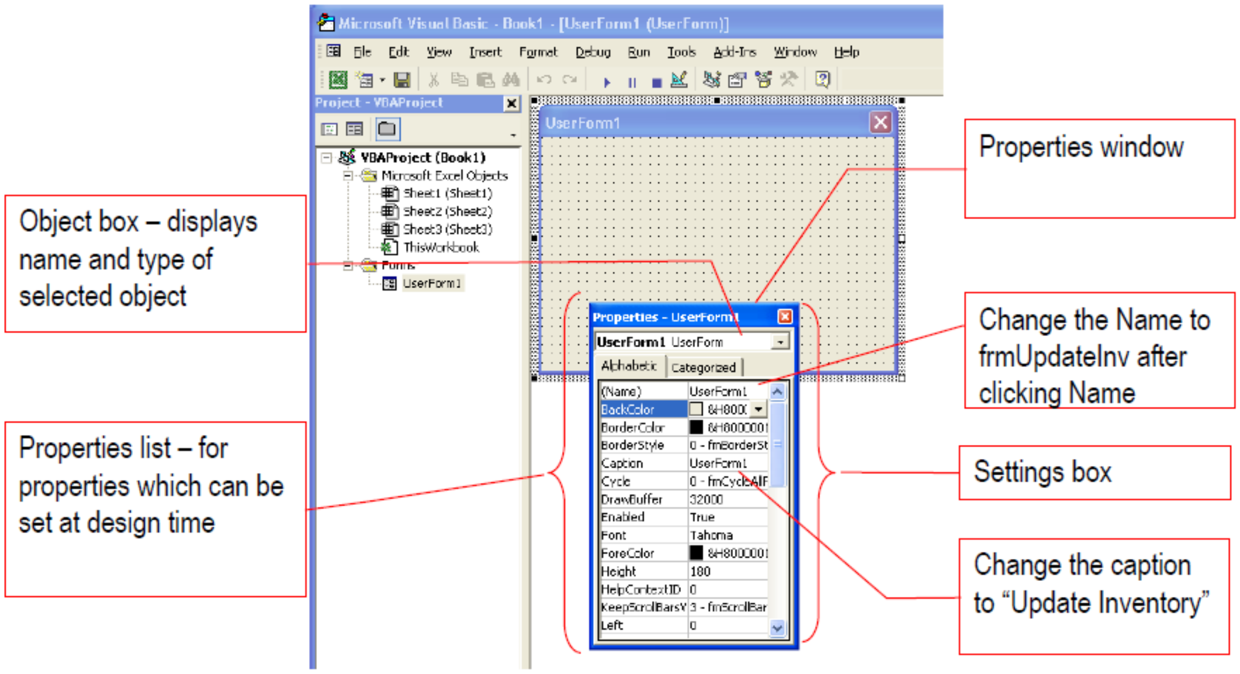
Form and Toolbox Window Shown in the Visual Basic Editor



Naming the Form

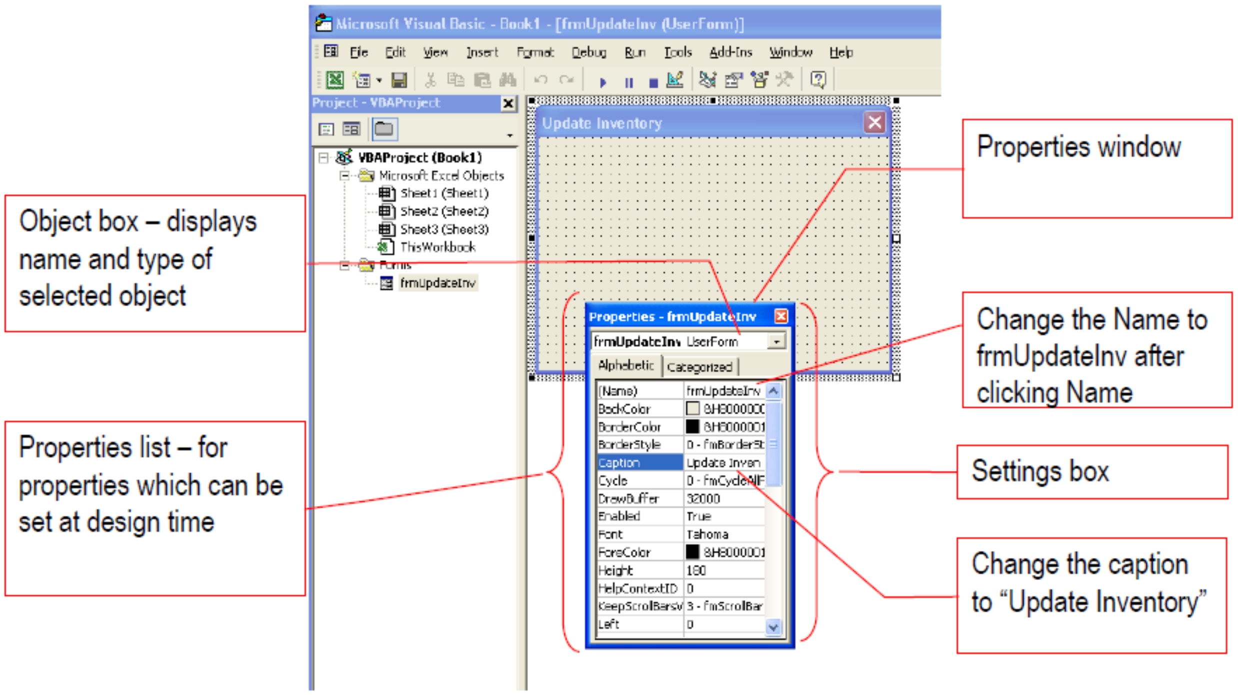
- Each form in a project must have a unique name
- The rules for naming forms are the same as the rules for naming variables
- The three-character ID used in form names is **frm**

The Properties Window


Microsoft Visual Basic - Book1 - [UserForm1 (UserForm)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - VBAProject
VBAProject (Book1)
Microsoft Excel Objects
Sheet1 (Sheet1)
Sheet2 (Sheet2)
Sheet3 (Sheet3)
ThisWorkbook
Forms
UserForm1
UserForm1
Properties - UserForm1
Alphabetic
Categorized
Properties window
Object box – displays name and type of selected object
Change the Name to frmUpdateInv after clicking Name
Properties list – for properties which can be set at design time
Settings box
Change the caption to “Update Inventory”

UserForm1 UserForm

(Name)	UserForm1
BackColor	&H800(
BorderColor	&H8000001
BorderStyle	0 - fmBorderSt
Caption	UserForm1
Cycle	0 - fmCycleAllF
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	&H8000001
Height	180
HelpContextID	0
KeepScrollBarsV	3 - fmScrollBar
Left	0

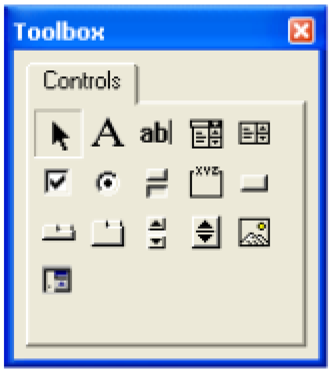

Microsoft Visual Basic - Book1 - [frmUpdateInv (UserForm)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - VBAProject
VBAProject (Book1)
Microsoft Excel Objects
Sheet1 (Sheet1)
Sheet2 (Sheet2)
Sheet3 (Sheet3)
ThisWorkbook
Forms
frmUpdateInv
Update Inventory
Properties - frmUpdateInv
Alphabetic
Categorized
Properties window
Object box – displays name and type of selected object
Change the Name to frmUpdateInv after clicking Name
Properties list – for properties which can be set at design time
Settings box
Change the caption to “Update Inventory”

frmUpdateInv UserForm

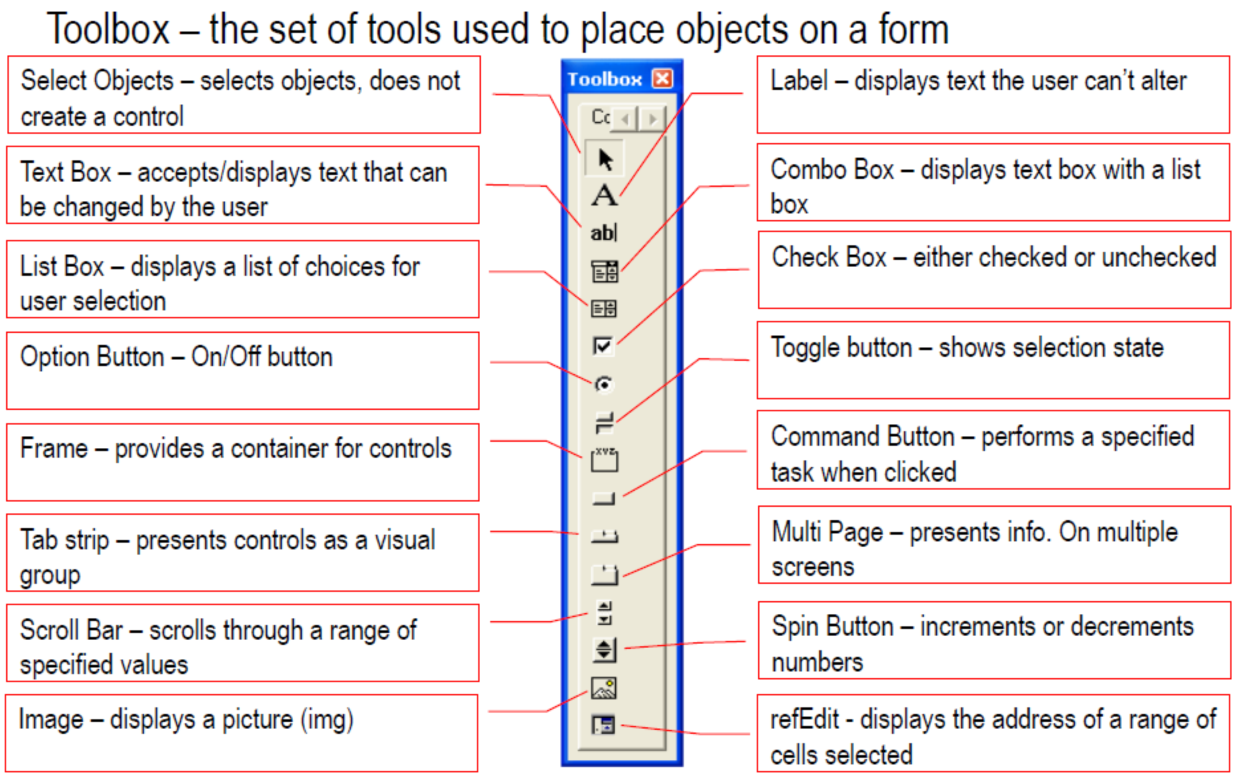
(Name)	frmUpdateInv
BackColor	&H800000C
BorderColor	&H8000001
BorderStyle	0 - fmBorderSt
Caption	Update Inven
Cycle	0 - fmCycleAllF
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	&H8000001
Height	180
HelpContextID	0
KeepScrollBarsV	3 - fmScrollBar
Left	0

3. Using the Toolbox Window to Add a Control to the Form

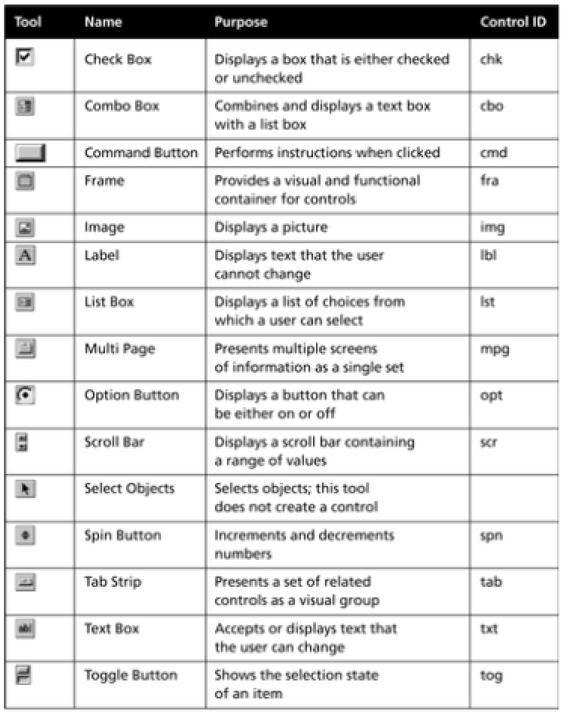
- The Toolbox window, also referred to simply as the toolbox, contains the set of tools you use to place objects, called controls, on the form
- You can add a control to a form simply by dragging the appropriate tool to the desired location on the form



Toolbox Window


Toolbox – the set of tools used to place objects on a form
Toolbox
Cc
Select Objects – selects objects, does not create a control
Text Box – accepts/displays text that can be changed by the user
List Box – displays a list of choices for user selection
Option Button – On/Off button
Frame – provides a container for controls
Tab strip – presents controls as a visual group
Scroll Bar – scrolls through a range of specified values
Image – displays a picture (img)
Label – displays text the user can't alter
Combo Box – displays text box with a list box
Check Box – either checked or unchecked
Toggle button – shows selection state
Command Button – performs a specified task when clicked
Multi Page – presents info. On multiple screens
Spin Button – increments or decrements numbers
refEdit - displays the address of a range of cells selected

Basic Tools included in the Toolbox



Tool	Name	Purpose	Control ID
	Check Box	Displays a box that is either checked or unchecked	chk
	Combo Box	Combines and displays a text box with a list box	cbo
	Command Button	Performs instructions when clicked	cmd
	Frame	Provides a visual and functional container for controls	fra
	Image	Displays a picture	img
	Label	Displays text that the user cannot change	lbl
	List Box	Displays a list of choices from which a user can select	lst
	Multi Page	Presents multiple screens of information as a single set	mpg
	Option Button	Displays a button that can be either on or off	opt
	Scroll Bar	Displays a scroll bar containing a range of values	scr
	Select Objects	Selects objects; this tool does not create a control	
	Spin Button	Increments and decrements numbers	spn
	Tab Strip	Presents a set of related controls as a visual group	tab
	Text Box	Accepts or displays text that the user can change	txt
	Toggle Button	Shows the selection state of an item	tog

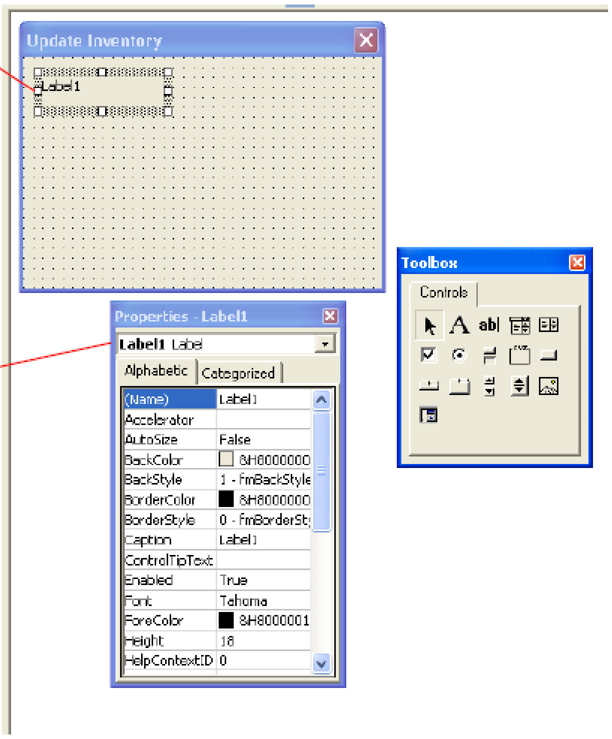
Basic tools included in the toolbox

Note: 3 character ID used to name the controls

Label Control

Label control – default size

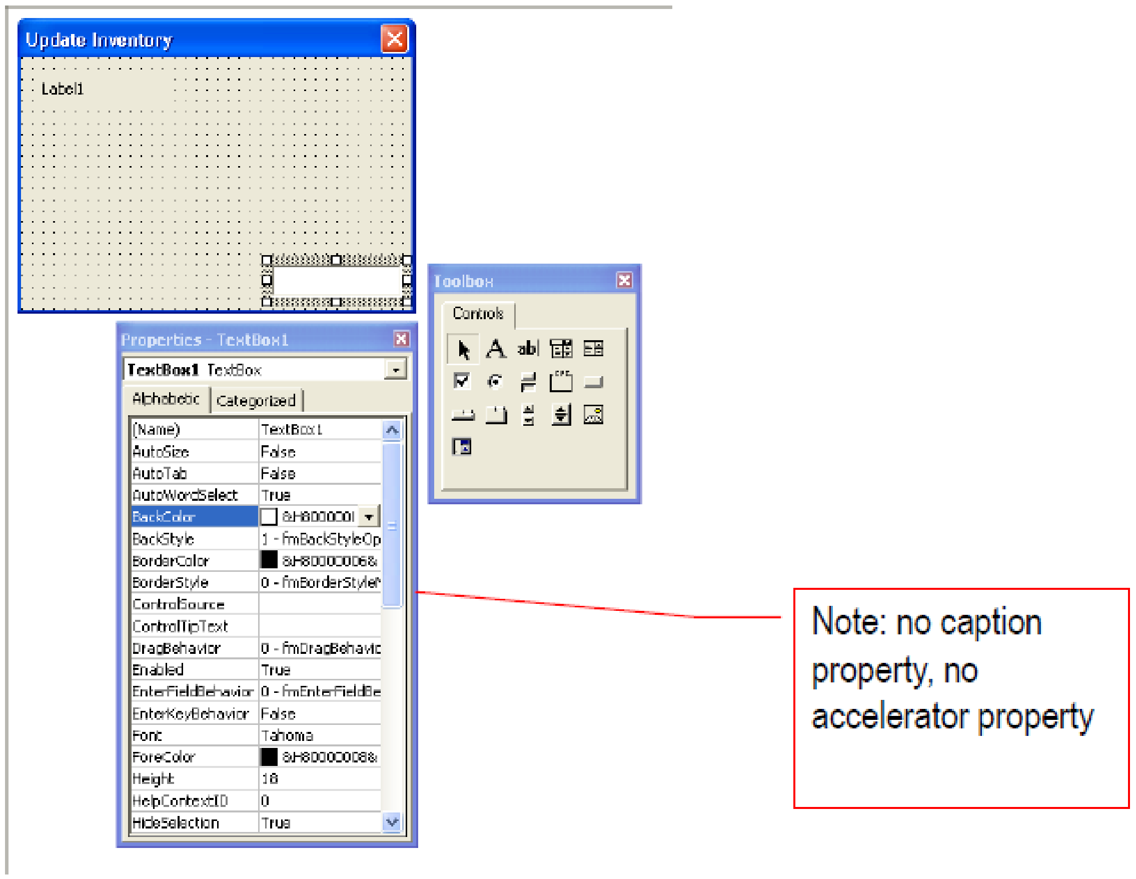
Label control – note default name and caption



Label1 Label

(Name)	Label1
Accelerator	
AutoSize	False
BackColor	&H8000000
BackStyle	1 - fmBackStyle
BorderColor	&H8000000
BorderStyle	0 - fmBorderSt
Caption	Label1
ControlTipText	
Enabled	True
Font	Tahoma
ForeColor	&H8000001
Height	18
HelpContextID	0

Default-size Text box Control

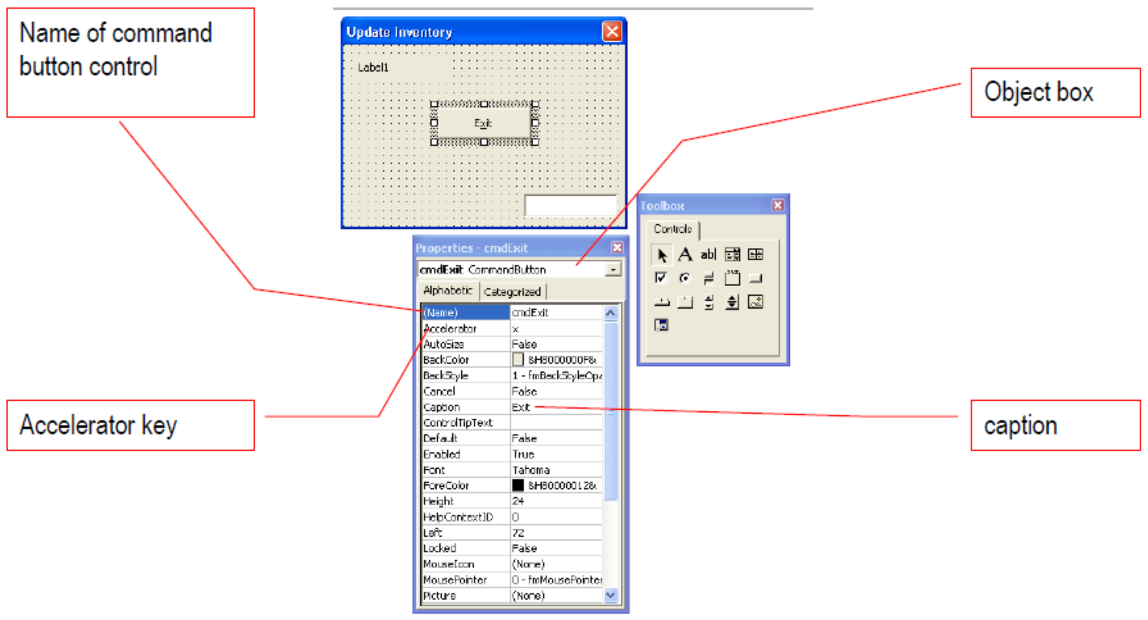


TextBox1 TextBox

(Name)	TextBox1
AutoSize	False
AutoTab	False
AutoWordSelect	True
BackColor	☐ &H800000
BackStyle	1 - fmBackStyleOp
BorderColor	■ &H80000006&
BorderStyle	0 - fmBorderStyle
ControlSource	
ControlTipText	
DragBehavior	0 - fmDragBehavio
Enabled	True
EnterFieldBehavior	0 - fmEnterFieldBe
EnterKeyBehavior	False
Font	Tahoma
ForeColor	■ &H80000008&
Height	18
HelpContextID	0
HideSelection	True

Note: no caption property, no accelerator property

Adding Command button control

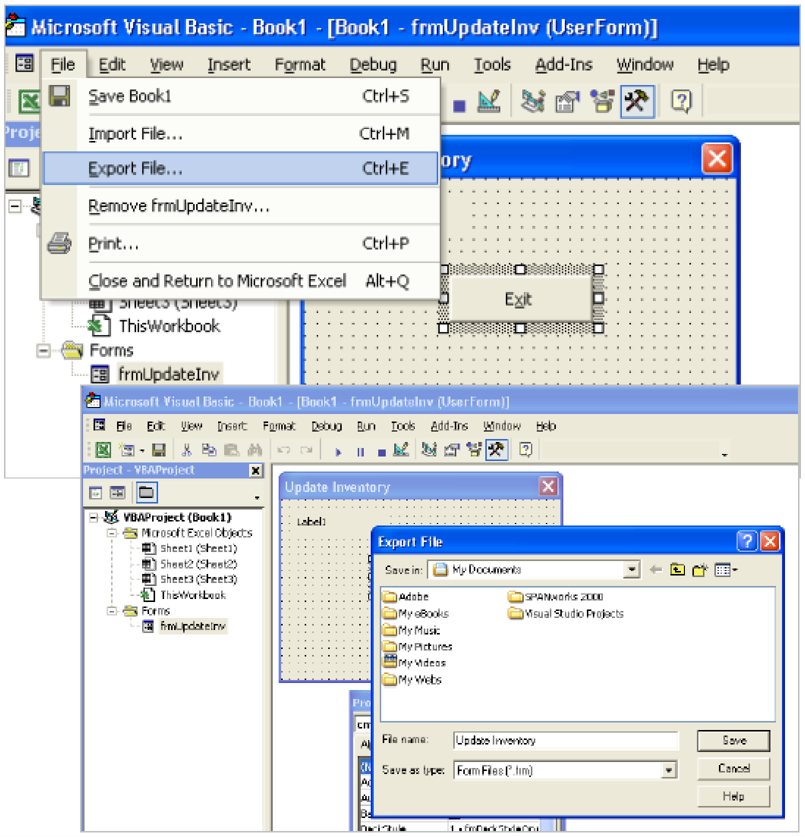


cmdExit CommandButton

(Name)	cmdExit
Accelerator	x
AutoSize	False
BackColor	☐ &H8000000F&
BackStyle	1 - fmBackStyleOp
Cancel	False
Caption	Exit
ControlTipText	
Default	False
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	24
HelpContextID	0
Left	72
Locked	False
MouseIcon	(None)
MousePointer	0 - fmMousePointe
Picture	(None)

Saving a Form

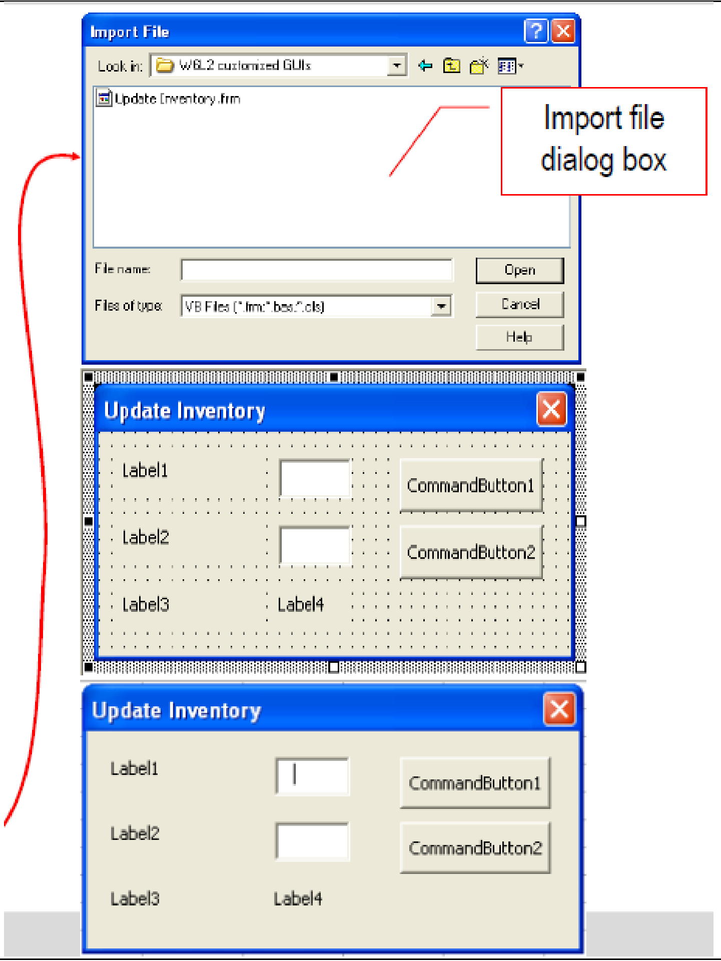
- You can save a form in the usual way – save button
- The process of saving a form to a file on a disk is referred to as exporting
- After a form has been exported, you can add the form to one or more projects later



4. Removing and Adding an Existing Form

- You can remove an existing form from a project by right-clicking the form's name in the Project Explorer window and then clicking Remove <formname> on the shortcut menu

- You can add an existing form to a project, a process referred to as **importing**, by right-clicking the Project Explorer window and then clicking Import File on the shortcut menu.



5. Displaying and Removing a User Form

- You use the form's **Show method** to bring the custom dialog box into the computer's memory and then display it on the screen, and you use the **Unload statement** to remove the dialog box from both the screen and memory

- The syntax of the Show method is
  formName.Show

- The syntax of the Unload statement is
  Unload formName

Removes an object from memory

You use the form's **Show method** to bring the custom dialog box into the computer's memory and then display it on the screen, and you use the **Unload** statement to remove the dialog box from both the screen and memory

The workbook open event:

```
Private Sub Workbook_Open()
   frmUpdateInv.Show
End Sub
```

The workbook close event:

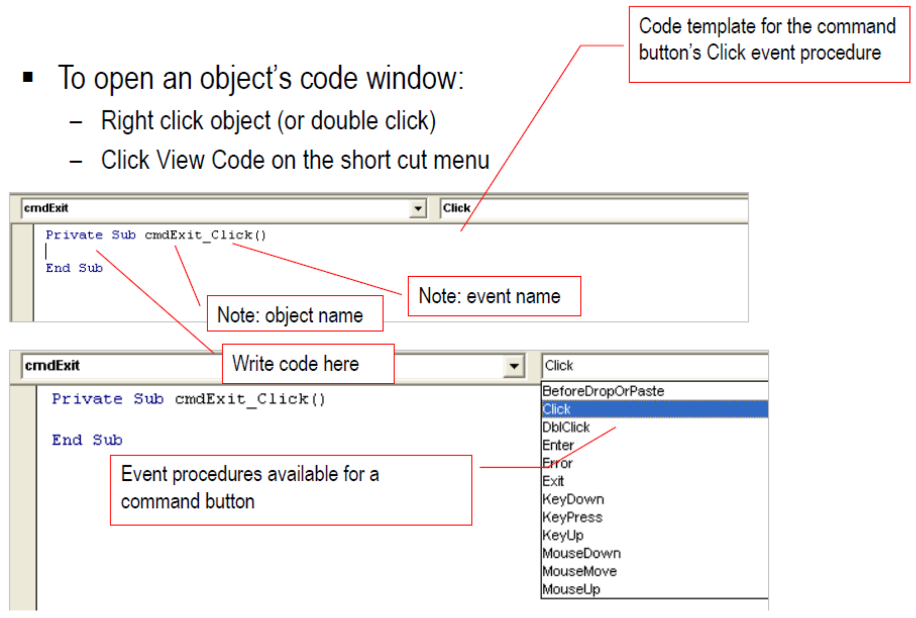
```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
   Unload frmUpdateInv
End Sub
```

Computer Inventory complete do loop Show.xls
(select ThisWorkbook)

6. Coding a User Form

- Actions performed by the user—such as clicking, double-clicking, and scrolling—are called **events**
- You tell an object how to respond to an event by writing an **event procedure**
- **Event procedures** are blocks of instructions that perform a task
- **Event procedures** run in response to an event rather than in response to running a macro
- Every form has its own set of event procedures (e.g. Activate, Click, Double Click)
- Every object on a form also has its own set of event procedures (e.g. Click, Double Click….)

- To open an object's code window:
  - Right click object (or double click)
  - Click View Code on the short cut menu



```
Private Sub cmdExit_Click()

End Sub
```

Example 6.1: Coding the Click event procedure for a command button

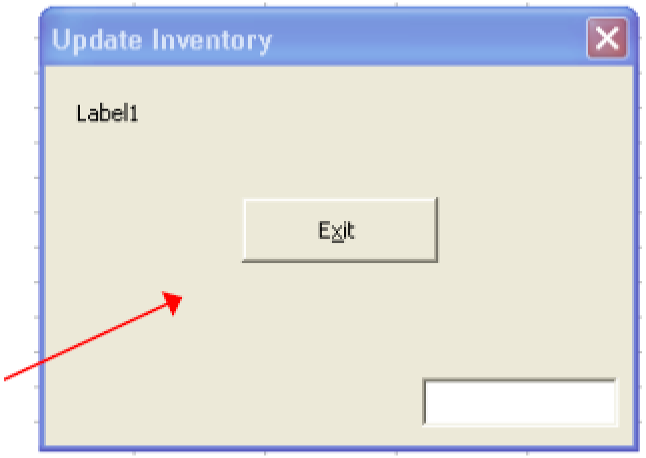
- Code to enter:

```
Unload frmUpdateInv
```

- Procedure becomes:

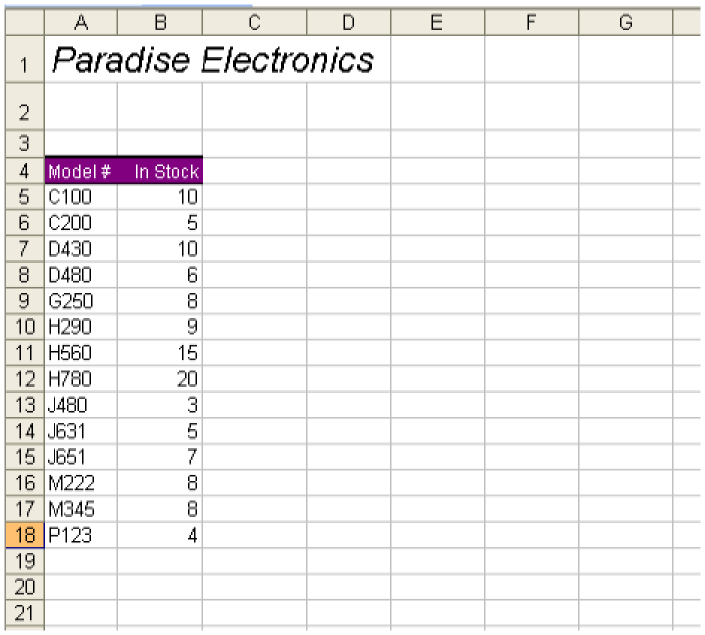
```
Private Sub cmdExit_Click()
  Unload frmUpdateInv
End Sub
```

- Running the code:



Example 6.2.1: Updating the Inventory Worksheet

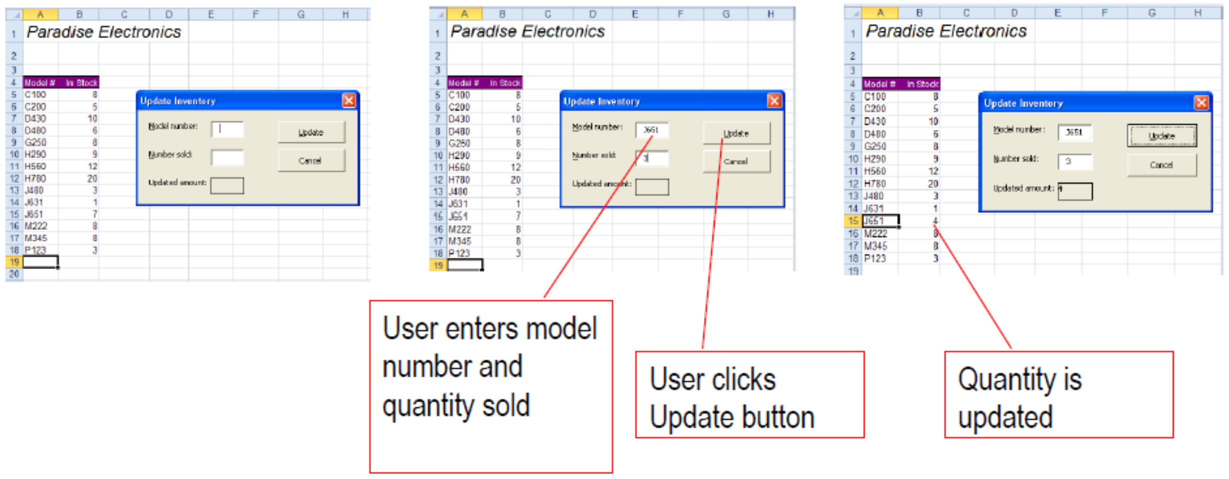
This exercise involves creating a macro that uses a custom dialog box to **update the inventory amounts** (by subtracting the numbers sold from the number in stock)



	A	B	C	D	E	F	G
1	*Paradise Electronics*						
2							
3							
4	Model #	In Stock					
5	C100	10					
6	C200	5					
7	D430	10					
8	D480	6					
9	G250	8					
10	H290	9					
11	H560	15					
12	H780	20					
13	J480	3					
14	J631	5					
15	J651	7					
16	M222	8					
17	M345	8					
18	P123	4					
19							
20							
21							

Example 6.2.2: command button click event procedure

Paradise Electronics:



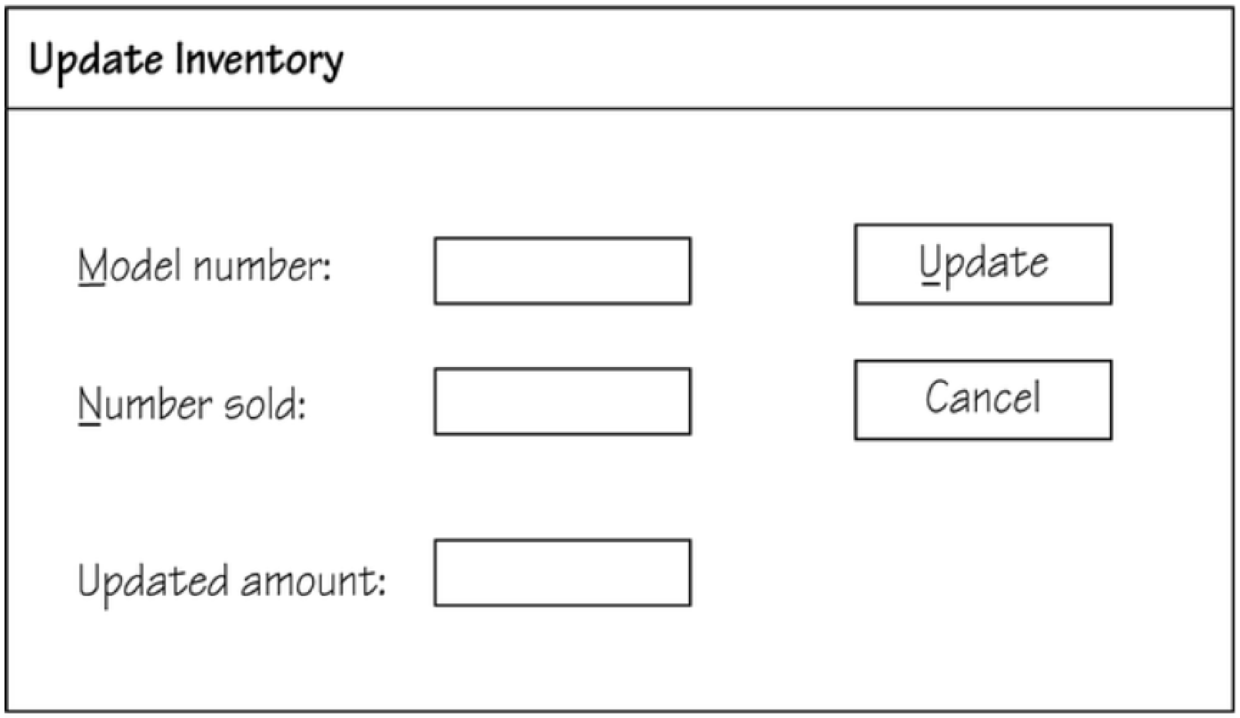
Paradise Electronics

Model #	In Stock
C100	8
C200	5
D430	10
D480	6
G250	8
H290	9
H560	12
H780	20
J480	3
J631	1
J651	7
M222	8
M345	8
P123	3

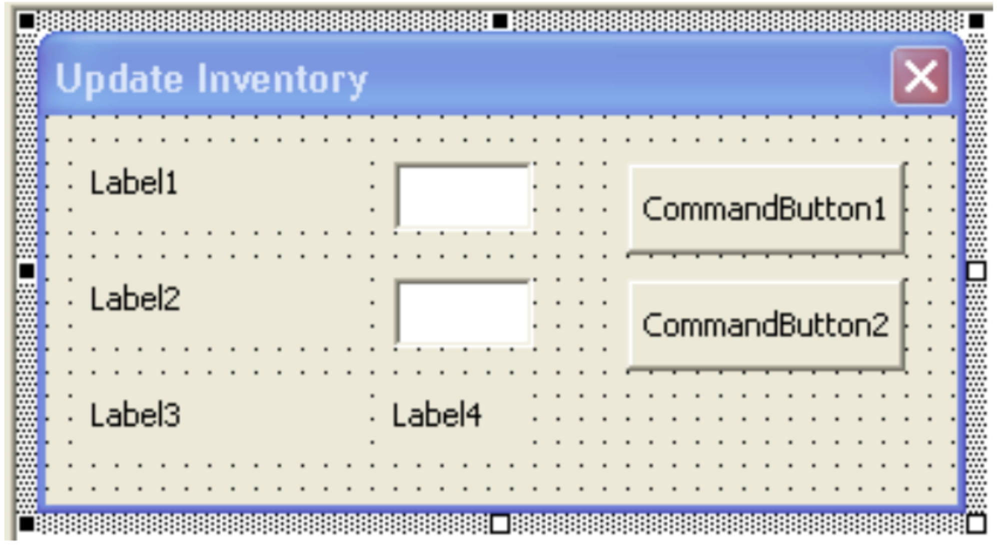
Model number: J651
Number sold: 3
Updated amount: 4

Model #	In Stock
C100	8
C200	5
D430	10
D480	6
G250	8
H290	9
H560	12
H780	20
J480	3
J631	1
J651	4
M222	8
M345	8
P123	3

Sketch of the Custom Dialog Box



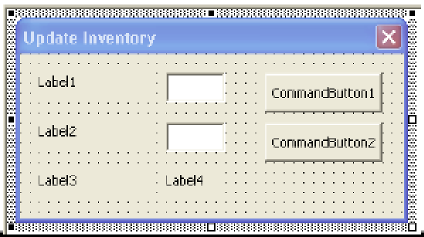
The Partially Completed frmUpdateInv Form



Setting the Name Property

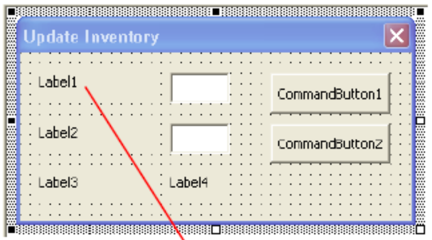
- The form and any controls that will be either coded or referred to in code should have their default name changed to a more meaningful one
- The form's name has been changed from **UserForm1** to **frmUpdateInv**; you now need to change the appropriate control names
- You will **not need** to change the names of the three identifying labels (Label1, Label2, and Label3), because those controls will not be coded or referred to in code

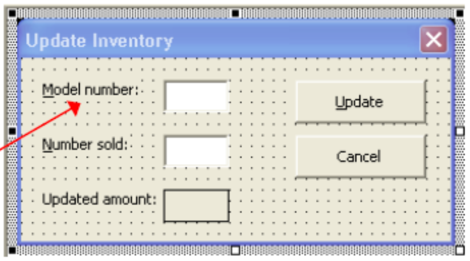
Controls Included in the Update Inventory Dialog Box



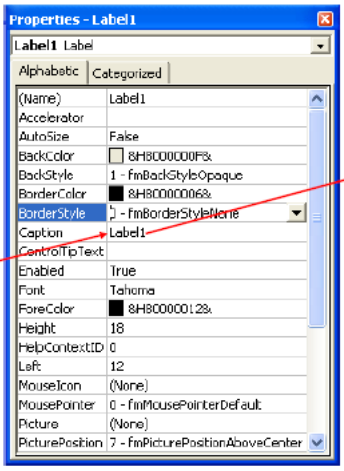
Default Name	Status	New Name
CommandButton1	Coded	cmdUpdate
CommandButton2	Coded	cmdCancel
Label1	Not coded or referred to in code	
Label2	Not coded or referred to in code	
Label3	Not coded or referred to in code	
Label4	Referred to in code	lblUpdated
TextBox1	Referred to in code	txtModel
TextBox2	Referred to in code	txtNumSold

Setting the Caption Property





- Label controls and command buttons have a **Caption property** that controls the text appearing inside the control
- When a label or command button is added to the form, its **default name** is assigned to the Caption property. These require updating.
- [Computer Inventory complete.xlsm]()

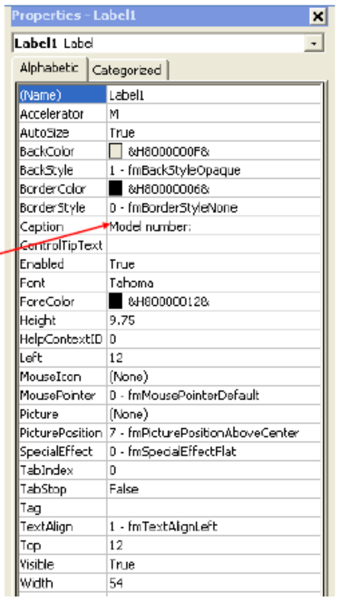


Properties - Label1

Label1 Label

Alphabetic | Categorized

(Name)	Label1
Accelerator	
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - fmBackStyleOpaque
BorderColor	&H80000006&
BorderStyle	0 - fmBorderStyleNone
Caption	Label1
ControlTipText	
Enabled	True
Font	Tahoma
ForeColor	&H80000012&
Height	18
HelpContextID	0
Left	12
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
Picture	(None)
PicturePosition	7 - fmPicturePositionAboveCenter



Properties - Label1

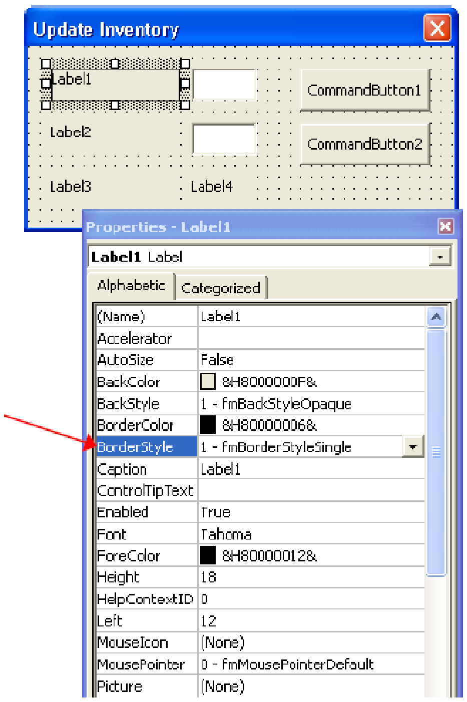
Label1 Label

Alphabetic | Categorized

(Name)	Label1
Accelerator	M
AutoSize	True
BackColor	&H8000000F&
BackStyle	1 - fmBackStyleOpaque
BorderColor	&H80000006&
BorderStyle	0 - fmBorderStyleNone
Caption	Model number:
ControlTipText	
Enabled	True
Font	Tahoma
ForeColor	&H80000012&
Height	9.75
HelpContextID	0
Left	12
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
Picture	(None)
PicturePosition	7 - fmPicturePositionAboveCenter
SpecialEffect	0 - fmSpecialEffectFlat
TabIndex	0
TabStop	False
Tag	
TextAlign	1 - fmTextAlignLeft
Top	12
Visible	True
Width	54

Setting the BorderStyle Property

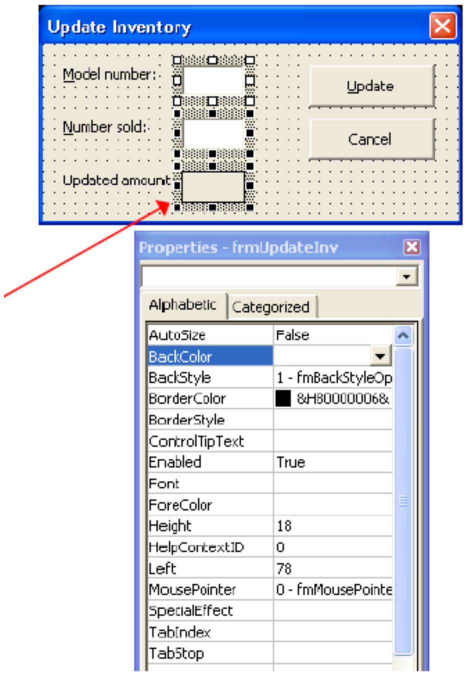
- Many objects have a **BorderStyle property** that determines the style of the object's border
- Label controls, for example, have a BorderStyle property that can be set to either 0 (fmBorderStyleNone) or 1 (fmBorderStyleSingle)
- The 0 - fmBorderStyleNone setting displays the label control without a border, while the 1 (fmBorderStyleSingle) setting displays the label control with a thin line around its border
- Many controls also have an AutoSize property, which does just what its name implies



(Name)	Label1
Accelerator	
AutoSize	False
BackColor	⬜ &H8000000F&
BackStyle	1 - fmBackStyleOpaque
BorderColor	⬛ &H80000006&
BorderStyle	1 - fmBorderStyleSingle
Caption	Label1
ControlTipText	
Enabled	True
Font	Tahoma
ForeColor	⬛ &H80000012&
Height	18
HelpContextID	0
Left	12
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
Picture	(None)

Changing the AutoSize Property for More Than One Control at a Time

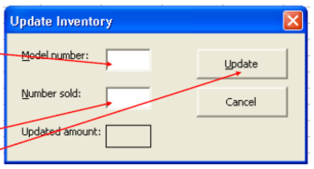
- You can set the **AutoSize** property for the three identifying labels individually, or you can change the property for the three controls at the same time
- Before you can change a property for a group of controls, you need to select the controls



AutoSize	False
BackColor	
BackStyle	1 - fmBackStyleOp
BorderColor	⬛ &H80000006&
BorderStyle	
ControlTipText	
Enabled	True
Font	
ForeColor	
Height	18
HelpContextID	0
Left	78
MousePointer	0 - fmMousePointe
SpecialEffect	
TabIndex	
TabStop	

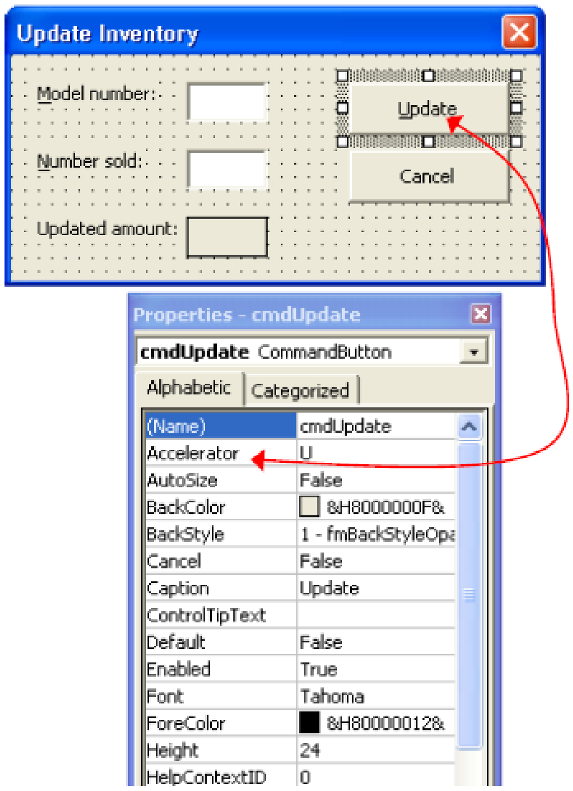
7. Providing Keyboard Access to Essential Controls

- You should provide keyboard access to each essential control on the form
- Use accelerator keys to provide keyboard access to the text boxes and to the Update command button



The Update button and Text box keyboard access

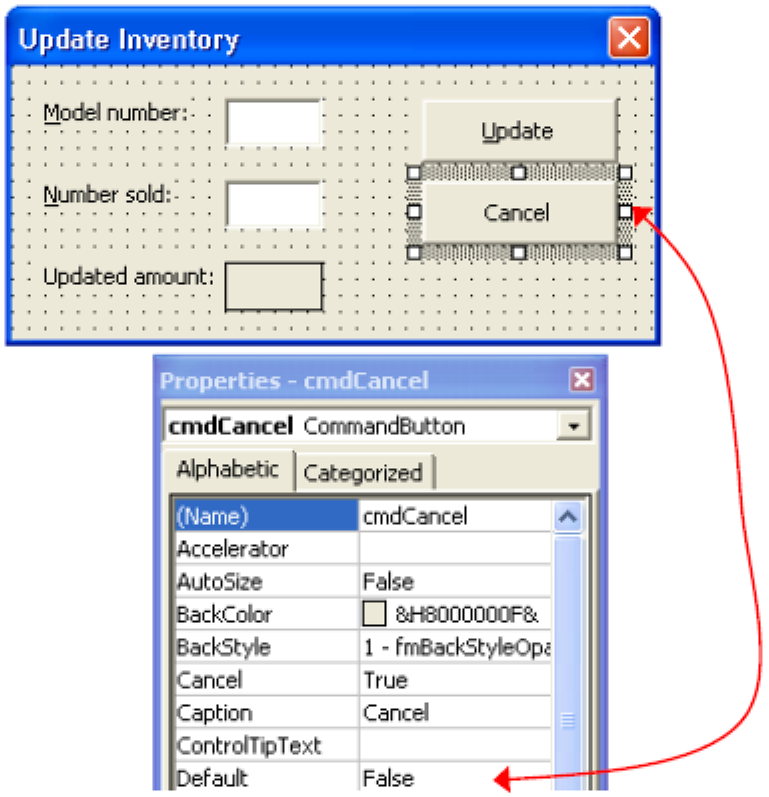
- Give the Update button an accelerator button of U
- Ditto for other essential controls….
- NB: text boxes don't have an accelerator key or a caption property
- To provide keyboard access for text boxes
  - Assign an accelerator key to the identifying label
  - Give the label control a TabIndex property immediately before the text box TabIndex property



(Name)	cmdUpdate
Accelerator	U
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - fmBackStyleOpa
Cancel	False
Caption	Update
ControlTipText	
Default	False
Enabled	True
Font	Tahoma
ForeColor	&H80000012&
Height	24
HelpContextID	0

The Cancel button

- Designate the Cancel button as the cancel button:



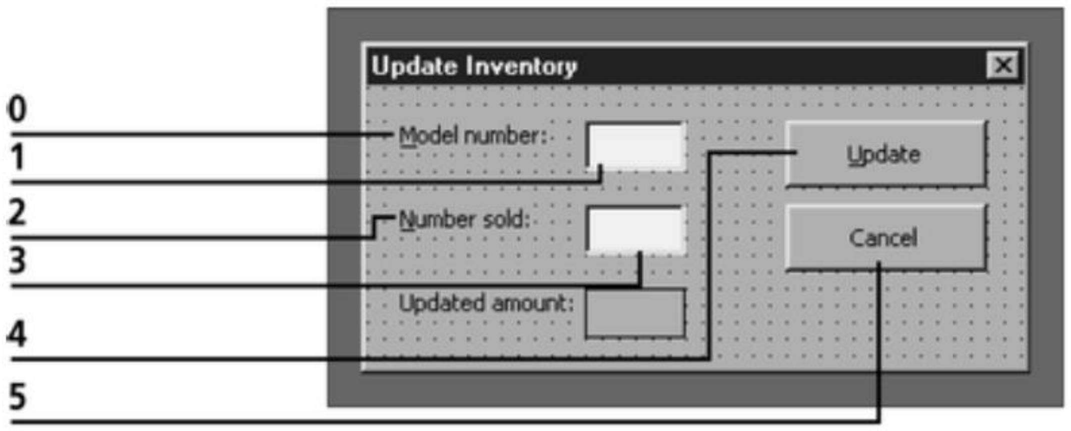
cmdCancel CommandButton

(Name)	cmdCancel
Accelerator	
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - fmBackStyleOpa
Cancel	True
Caption	Cancel
ControlTipText	
Default	False

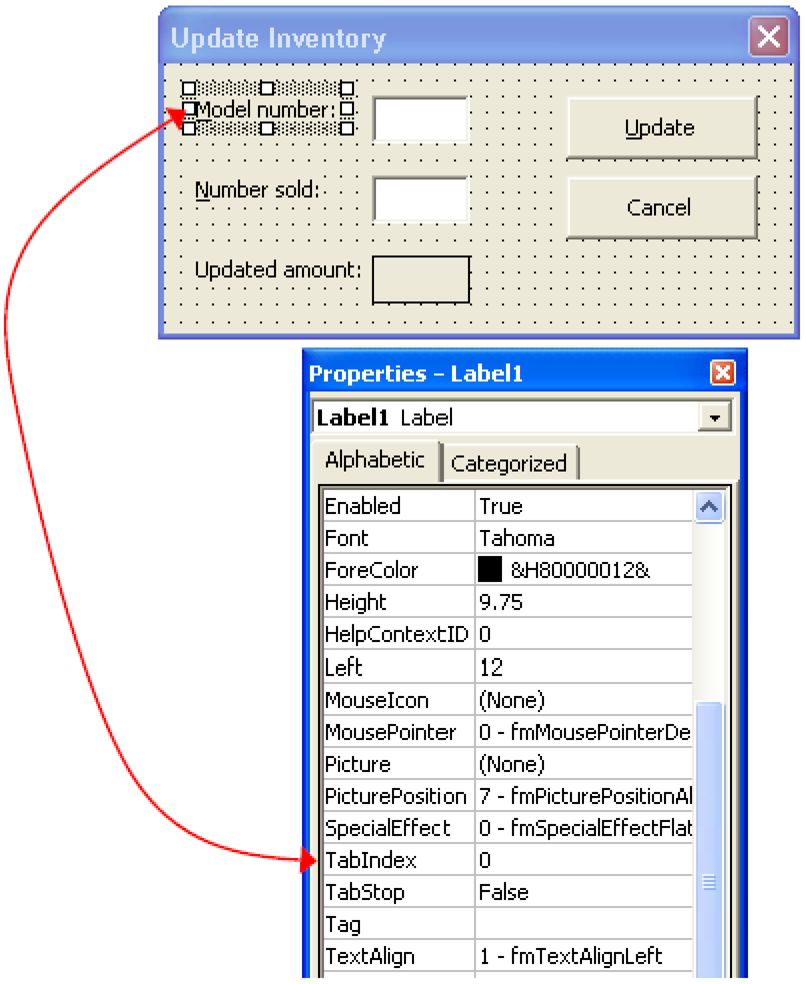
8. Setting the Tab Order

- The **tab order** is determined by the **TabIndex** property of the controls included in the dialog box
- When you add a control to a form, the control's **TabIndex** property is set to a number that represents the order in which the control was added to the form
- The control whose **TabIndex** value is 0 will receive the focus first, because it is the first control in the tab order
- Before you can set the **TabIndex** property of the controls, you need to determine where each essential control should fall in the tab order

TabIndex Values for Essential Controls and Their Identifying Labels



TabIndex Values for Model Number



Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	9.75
HelpContextID	0
Left	12
MouseIcon	(None)
MousePointer	0 - fmMousePointerDe
Picture	(None)
PicturePosition	7 - fmPicturePositionAl
SpecialEffect	0 - fmSpecialEffectFlat
TabIndex	0
TabStop	False
Tag	
TextAlign	1 - fmTextAlignLeft

Coding the Controls in the Update Inventory Dialog Box

- The first control to code is the **Cancel button**, which should remove the form from both the screen and the computer's memory when the user selects the button

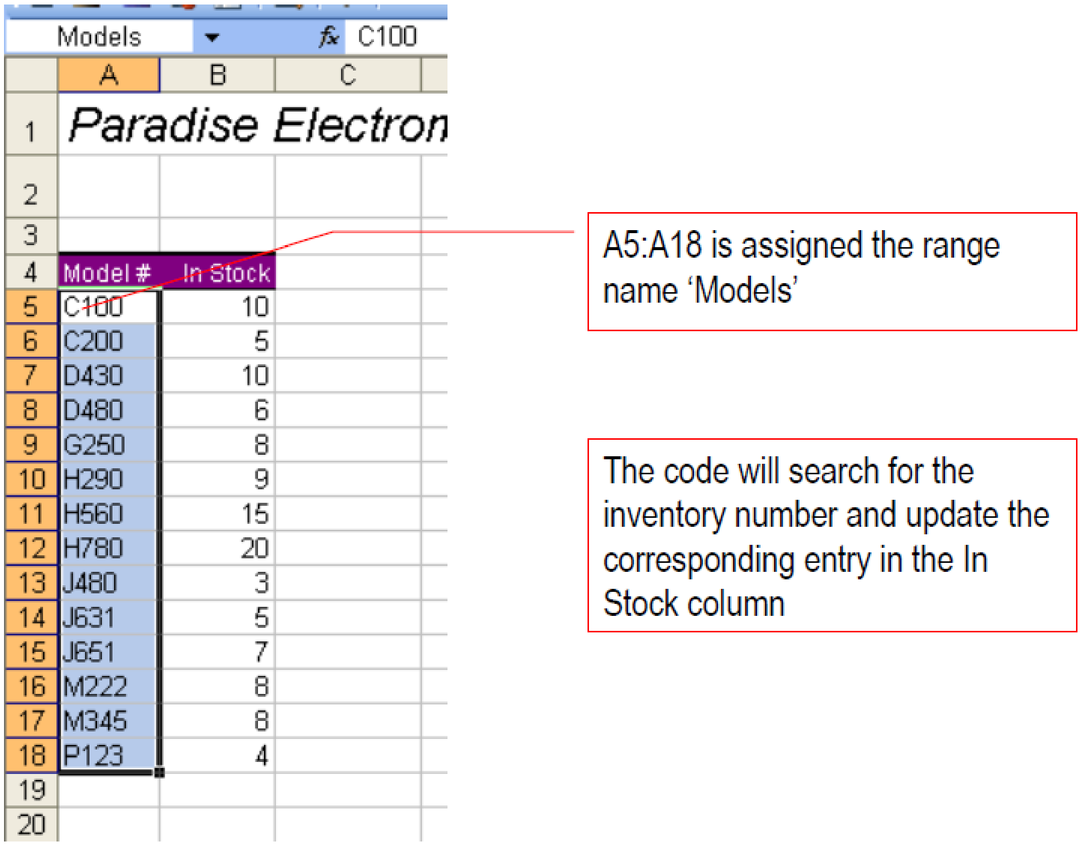
- The next control to code is the **Update command button**, which the user can select either by clicking it or by pressing the Enter key when the button has the focus

Code for the Cancel button

- Code (same as previously):
- The syntax of the Unload statement is
  Unload formName

```
Private Sub cmdCancel_Click()
   Unload frmUpdateInv
End Sub
```

The Inventory Worksheet in the Computer Inventory Workbook – code for Update button



Models | C100

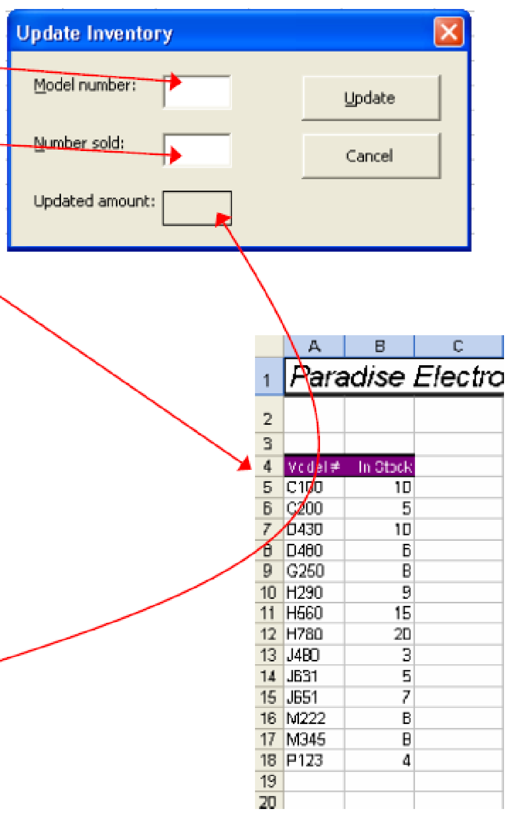
	A	B	C
1	*Paradise Electron*		
2			
3			
4	Model #	In Stock	
5	C100	10	
6	C200	5	
7	D430	10	
8	D480	6	
9	G250	8	
10	H290	9	
11	H560	15	
12	H780	20	
13	J480	3	
14	J631	5	
15	J651	7	
16	M222	8	
17	M345	8	
18	P123	4	
19			
20			

A5:A18 is assigned the range name 'Models'

The code will search for the inventory number and update the corresponding entry in the In Stock column

Pseudocode for the Update Button’s Click Event Procedure

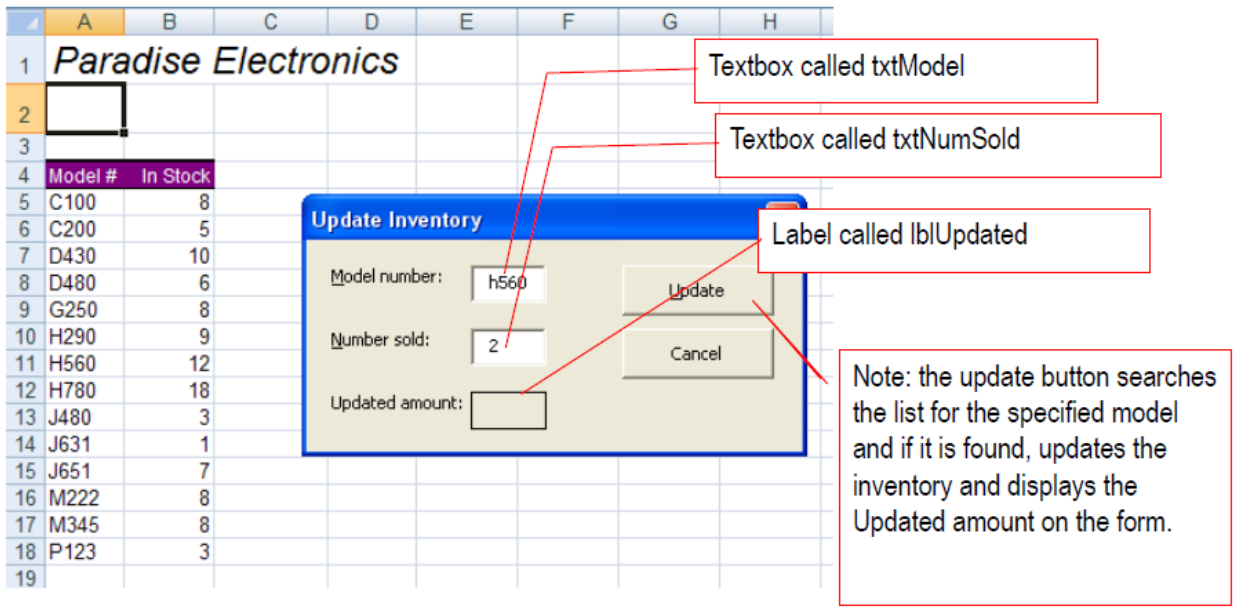
```
1.  Assign the contents of the txtModel control, in
    uppercase letters, to a string variable named strModel
2.  Assign the contents of the txtNumSold control, treated
    as a number, to an integer variable intNumSold
3.  Repeat the following for each cell in the models
    range:
a.  If the model number stored in the current cell is equal
    to the model number stored in the strModel variable,
    then
        1)  Calculate the updated inventory amount by
            subtracting the contents of the intNumSold
            variable from the model’s current inventory
            amount, which is contained in the cell located to
            the immediate right of the current cell in the
            worksheet. Assign the result to an integer
            variable named intUpdated.
        2)  Assign the contents of the intUpdated variable
            both to the lblUpdated control in the dialog box
            and to the cell located to the immediate right of
            the current cell in the worksheet.
        3)  Exit the loop.
```



Model #	In Stock
C100	10
C200	5
D430	10
D480	6
G250	8
H290	9
H560	15
H780	20
J480	3
J631	5
J651	7
M222	8
M345	8
P123	4

Click event procedure for Update Command button

- Form control names



Model #	In Stock
C100	8
C200	5
D430	10
D480	6
G250	8
H290	9
H560	12
H780	18
J480	3
J631	1
J651	7
M222	8
M345	8
P123	3

Note: the update button searches the list for the specified model and if it is found, updates the inventory and displays the Updated amount on the form.

Variables required

Variable	Data type
strModel	String
intNumSold	Integer
intUpdated	Integer
wksInventory	Worksheet
rngCell	range

Variables used by Update button's click event procedure.

Start of Update Button's Click Event Procedure

```
Private Sub cmdUpdate_Click()
    'declare variables and assign address to Worksheet variable
    Dim strModel As String, intNumSold As Integer, intUpdated As Integer
    Dim wksInventory As Worksheet,
     Dim rngCell As Range
    Set wksInventory = _
        Application.Workbooks("computer
     inventory.xls").Worksheets("inventory")
.......
End Sub
```

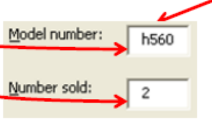
Click event procedure for Update Command button

Example – updating an inventory Computer Inventory complete For Each version.xlsm

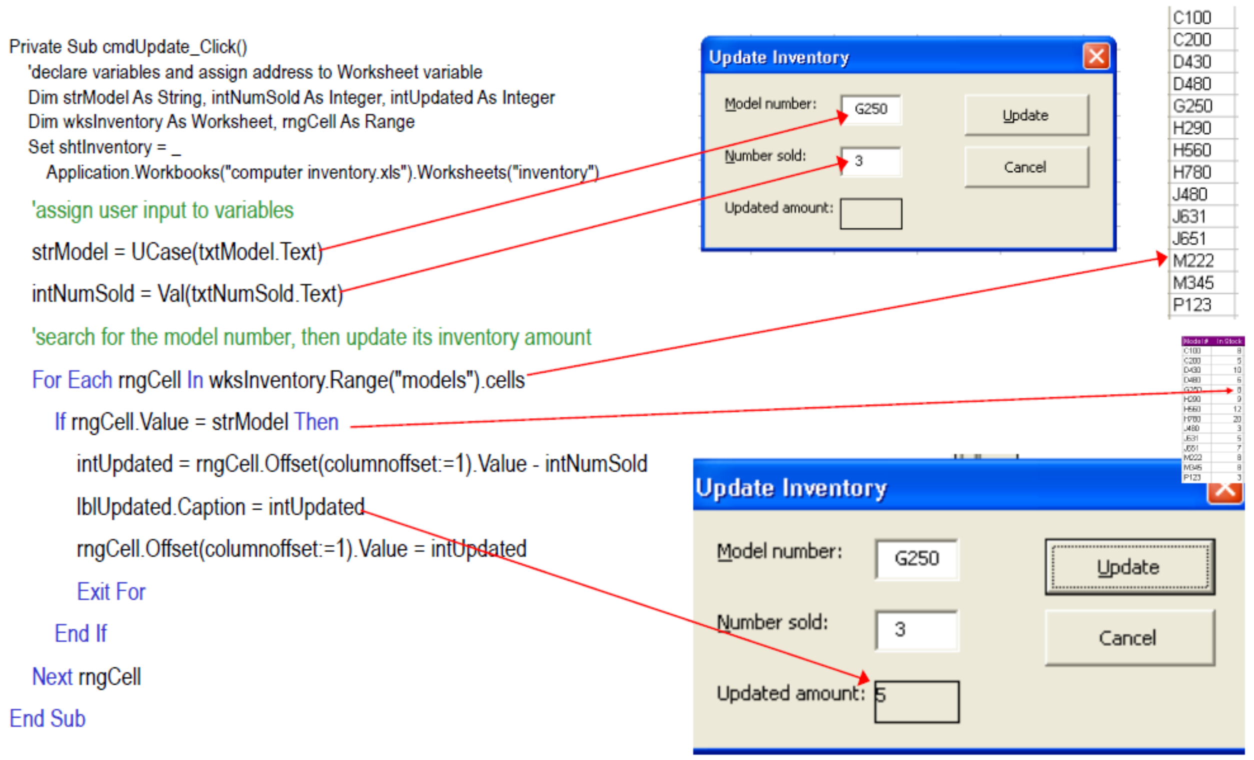
```
Private Sub cmdUpdate_Click()
    'declare variables and assign address to Worksheet variable
    Dim strModel As String
    Dim intNumSold As Integer
    Dim intUpdated As Integer
    Dim wksInventory As Worksheet
    Set wksInventory = _
       Application.Workbooks("computer inventory complete do loop
     version.xlsm").Worksheets("inventory")
    'assign user input to variables
    strModel = UCase(txtModel.Text)
    intNumSold = Val(txtNumSold.Text)
```

Declaring and assigning variables

Capturing user input and assigning to variables



Completed Click Event Procedure for the cmdUpdate Control



```
Private Sub cmdUpdate_Click()
    'declare variables and assign address to Worksheet variable
    Dim strModel As String, intNumSold As Integer, intUpdated As Integer
    Dim wksInventory As Worksheet, rngCell As Range
    Set shtInventory = _
        Application.Workbooks("computer inventory.xls").Worksheets("inventory")
    'assign user input to variables
    strModel = UCase(txtModel.Text)
    intNumSold = Val(txtNumSold.Text)
    'search for the model number, then update its inventory amount
    For Each rngCell In wksInventory.Range("models").cells
        If rngCell.Value = strModel Then
            intUpdated = rngCell.Offset(columnoffset:=1).Value - intNumSold
            lblUpdated.Caption = intUpdated
            rngCell.Offset(columnoffset:=1).Value = intUpdated
            Exit For
        End If
    Next rngCell
End Sub
```

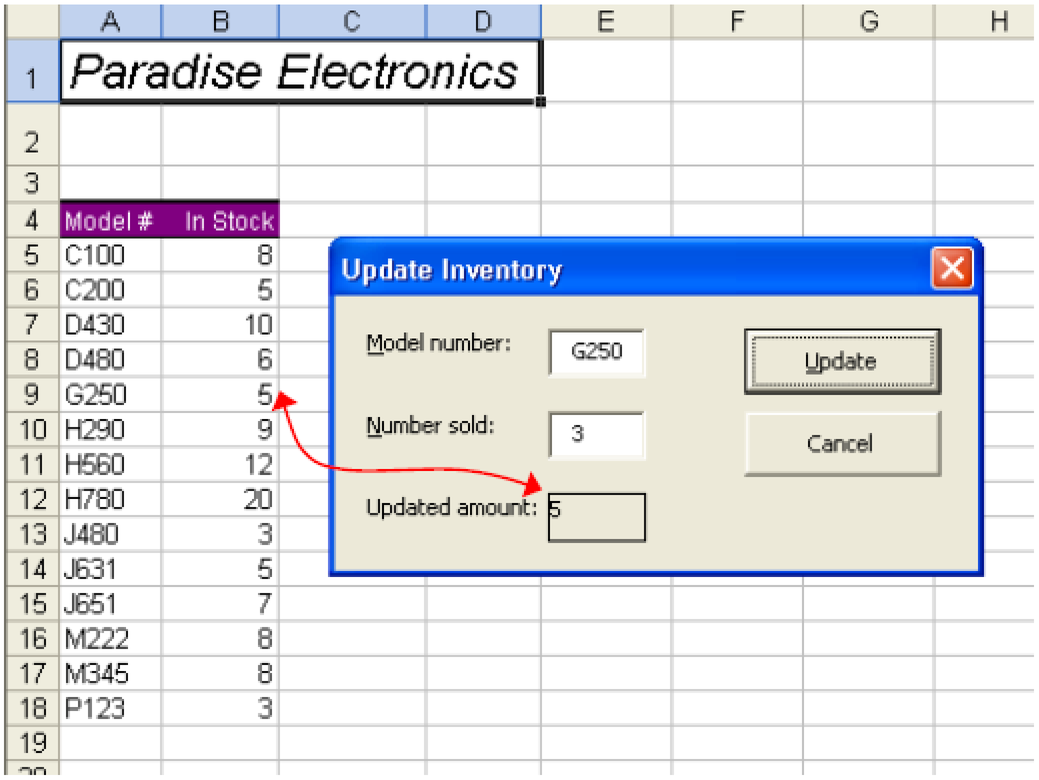
Update Inventory

Model number: G250

Number sold: 3

Updated amount: 5

Updated Inventory Amount Shown in the Dialog Box and in the Worksheet



Paradise Electronics

Model #	In Stock
C100	8
C200	5
D430	10
D480	6
G250	5
H290	9
H560	12
H780	20
J480	3
J631	5
J651	7
M222	8
M345	8
P123	3

Update Inventory

Model number: G250

Number sold: 3

Updated amount: 5

Alternate Click event procedure for Update Command button – using a Do Loop

- Example – updating an inventory [Computer Inventory complete do loop version.xls]

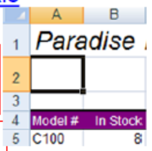
Code continued:

```
  'search for the model number, then update its inventory amou
  wksInventory.Range("model").Select
  Do Until IsEmpty(ActiveCell)
     If ActiveCell.Value = strModel Then
        intUpdated = ActiveCell.Offset(columnoffset:=1).Value - intNumSold
        lblUpdated.Caption = intUpdated
        ActiveCell.Offset(columnoffset:=1).Value = intUpdated
        Exit Do
     End If
     ActiveCell.Offset(1, 0).Select
  Loop
End Sub
```

"model" refers to cell A4

Runs through the list of models

If the model is found, the details are updated



Summary

- To create **a user form** or **custom dialog box**:
- Add a form to the project, then add controls to the form:
    - Click Insert on the menu bar, and then click UserForm
    - Align the controls wherever possible to minimize the number of different margins on the form
- To follow the Windows standards for controls:
    - Use a **label** control to display text that you don't want the user to modify
    - Use a **text box** control to provide an area in the dialog box where data can be entered
    - Use a **command button** control to process one or more instructions as soon as the button is clicked
- Position the **command button** either at the **bottom** or on the **right** side of the dialog box
- Group related command buttons together by positioning them close to each other in the dialog box
- Provide keyboard access to the essential controls in the dialog box using **accelerator** keys
- To select an appropriate accelerator key for a control:
    - Use the first letter of the control's caption, unless another letter provides a more meaningful association
- To specify a command button as the default button:
    - Set the command button's **Default** property to **True**
- To specify a command button as the **cancel** button:
    - Set the command button's Cancel property to **True**
- Set an appropriate tab order
- To change the properties of an object:
    - Use the Properties window
- To have a procedure display a custom dialog box on the screen:
    - Use the Show method, whose syntax is: **formName.Show**

- To have a procedure remove a form from both the screen and the computer's memory:
  - Use the Unload statement, whose syntax is **Unload formName**
- To have an object respond to an event in a particular way:
  - Enter VBA instructions in the appropriate event procedure for the object

9. Practice and Apply

- Understanding how to create a user form
- Understanding how to Add controls to a form
- Be able to explain the use of text box, label, and command button controls
- Understanding how to provide keyboard access to controls using accelerator keys
- Understanding how to code a user form
- Complete Tutorial 8 Exercises