Objectives (1 of 2)

- Use text boxes, group boxes, check boxes, radio buttons, and picture boxes effectively.
- Set the BorderStyle property to make controls appear flat or three-dimensional.
- Select multiple controls and move them, align them, and set common properties.
- Make your projects easy for the user to understand and operate by defining access keys, setting an accept and a cancel button, controlling the tab sequence, resetting the focus during program execution, and causing ToolTips to appear.

Objectives (2 of 2)

- Clear the contents of text boxes and labels.
- Change text color during program execution.
- Code multiple statements for one control using the With and End With statements.
- Concatenate (join) strings of text.
- Make a control visible or invisible at run time by setting its Visible property.
More Controls

• Click the plus and minus signs on the tabs to open and close each section.

• Forms can have several different types of controls such as text boxes, group boxes, check boxes, radio buttons, and picture boxes.

Text Box

• Allows for user input

  • Text property
    • What is displayed in text box
    • What user entered in text box

  • TextAlign property or HorizontalAlignment.Left
    • Controls alignment of text in the text box

Masked Text Box

• A specialized form of the Text Box

  • You can specify the format (the Mask property) of the data required of the user. For example, you can select a mask for a ZIP code, a date, a phone number, or a social security number.

  • At run time, the user cannot enter characters that do not conform to the mask.
Group Box

- Used as a container for other controls such as radio buttons and check boxes
- Improves readability of form by separating the controls into logical groups
- Example Names for Group Boxes
  - colorGroupBox
  - styleGroupBox

Check Box

- Allows the user to select or deselect one or more items in any group
- Checked property
  - Checked = True
  - Unchecked = False
- Use the Text property for the text you want to appear next to the box.
- Example names for Check Boxes
  - boldCheckBox
  - italicCheckBox

Radio Button

- User may select only one in any group.
- First create a group box and then create each radio button inside the group box.
- Checked property
  - Selected = True
  - Unselected = False
- Text property
  - What is displayed next to the radio button
- Example Names for Radio Buttons
  - redRadioButton
  - blueRadioButton
Picture Box

- Displays/contains an image
- Image property
  - Complete path and filename of graphic: it's a good idea to place the graphic into the folder with your project before it is assigned to a picture box.
  - Can set extension to .bmp, .gif, .jpg, .jpeg, .png, .ico, .emf, .wmf
- PictureBox controls have several useful properties such as the SizeMode property.
  - StretchImage causes picture to be resized to match the size of the control, or the
  - Visible property, which can be set to False to make the picture disappear

Borders and Styles

- Most controls can appear to be three-dimensional or flat.
- Labels, text boxes, and pictures boxes all have a BorderStyle property with different style choices including:
  - None
  - FixedSingle
  - Fixed3D properties that can be changed to the style of choosing

Drawing a Line

- A line can be drawn on a form by using the Label control.
- Lines can be used in logos or to divide the screen.
- To create the look of a line, set the following properties:
  - Text=blank
  - BorderStyle=none
  - BackColor=desired line color
  - Width and Height, located beneath the Size property as desired
- A line can also be drawn on a form using the graphics methods that are discussed in Chapter 13.
Selecting Multiple Controls (1 of 2)

- There are several methods of selecting multiple controls, which means controls can be moved as a group.
- Shift-Click or Ctrl-Click to select/deselect multiple controls.
- Use the mouse to drag a selection box around multiple controls.
- When wanting to select most of the controls on the form, use a combination of the two methods.
- To deselect all selected controls, click elsewhere on the form.

Selecting Multiple Controls (2 of 2)

Use the mouse to drag a selection box around multiple controls.

When multiple controls are selected, each has resizing handles (if resizable).

Working with Multiple Controls

What are some of the benefits of working with Multiple Controls?
- Drag a group of selected controls to move the entire group to a new location at one time.
- Set common properties.
- Use Format Menu or Layout Toolbar to
  - Align controls to each other.
  - Make controls the same size.
  - Modify the spacing between controls.
Designing the User Interface

- The design of the screen should be easy to understand and "comfortable" for the user. The best way to accomplish these goals is to follow industry standards for the color, size, and placement of controls.
- To the user, the Interface should be
  - Familiar
  - Comfortable
  - Organized—group controls
  - Sans Serif Fonts are best, not boldface or large
  - Gray is a neutral color that works well
  - Keyboard Accessible

Keyboard Access Keys

- Many people prefer to use the keyboard, rather than a mouse, for most operations.
- Also referred to as Hot Keys
- Underlined Letter
- User presses Alt + underlined letter.
- Use Windows-Standard Keys
- Defined using Text property

Accept & Cancel Buttons

- Accept Button
  - Identified visually on Form by its darker outline
  - Responds to Enter key
  - Form's AcceptButton property
- Cancel Button
  - Form's CancelButton property
  - An example of a Cancel button is the Esc key.
Setting the Tab Order for Controls (1 of 2)

- One control on a Form always has the focus.
- Not all control types can receive the focus.
  - Text boxes, buttons — focus
  - Picture boxes and labels — no focus
- **TabStop property** is applicable only for controls that are capable of receiving the focus.
  - Designates whether a control is allowed to receive the focus; set to True or False
- **TabIndex property** determines the order the focus moves as the Tab key is pressed.

Setting the Tab Order for Controls (2 of 2)

- Users should be able to use the Tab key to move the focus through a form in an organized manner; top to bottom, left to right.
- **TabIndex property**
  - Number in tab sequence
  - 0 for first control to receive the focus when the form loads
- Use View Menu, Tab Order to set.

Setting the TabIndex Property

- With the Design window active, select View / Tab Order or click the Tab Order button on the Layout toolbar.
- Click on each control in sequence to set TabIndex property of controls automatically.
Setting the Form’s Screen Location

- You can set the form’s screen position by setting theStartPosition property of the form.
- To center a form on the user’s screen, set the StartPosition property to CenterScreen.

Creating ToolTips

- Small label that is displayed when user pauses mouse pointer over a control
  - Add a ToolTip Control to Form.
  - The new control appears in a new pane, Component Tray, which opens at the bottom of the Form Designer.
  - After you add the component to your form, each of the form’s controls has a new property.
  - Select ToolTip on ToolTip1 property of each control and add Tool Tip comments.
Clearing Text Boxes & Labels

- Set Text property equal to an Empty String.
  - Empty String is 2 quotation marks with no space between them ("").
- Use the Clear Method of a Text Box or set Text property to String.Empty.

```csharp
nameTextBox.Text = ""
messagelabel.Text = ""
dataTextBox.Clear();
messagelabel.Text = String.Empty
```

Resetting the Focus

- Places the Insertion Point in a Text Box
- Use the Focus Method.

Statement to set the insertion point in the text box called NameTextBox is as follows:

```csharp
nameTextBox.Focus = null
```

Setting the Checked Property of Radio Buttons and Check Boxes

- Selects/Deselects Check Box or Radio Button at design or run time
- Set Checked property.
  - True = Checked, selected
  - False = Unchecked, deselected

In code, assign True to its Checked property.

```csharp
redRadioButton.Checked = true
displayCheckBox.Checked = false
```
Setting Visibility at Run Time

- Make label invisible.

```vbnet
messageLabel.Visible = False
```

Changing the Color of Text

- Change ForeColor and BackColor Properties.
- Use VB Color Constants from the Color Class.

```vbnet
Color.AliceBlue
Color.AntiqueWhite
Color.Blue
```

- Type the keyword Color and a period in the editor to view a full color list.

```vbnet
nameTextBox.ForeColor = Color.Red
messageLabel.ForeColor = Color.White
```

Changing Multiple Properties of a Control

- There are times when you will want to change several properties of a single control.
- Previous to version 4, the entire name (Object.Property) for each statement had to be written, such as:
  ```vbnet
  titleTextBox.Visible = True
  titleTextBox.ForeColor = Color.White
  titleTextBox.Focus()
  ```
- Now, VB provides With and End With statements.
With and End With

- Change several properties at once in Code.
- Will run more efficiently

```
With title TextBox
  .Visible=True
  .ForeColor=Color.White
  .Focus()
End With
```

Concatenation

- Think of concatenation as "tacking" text strings together.
- Use an ampersand (&) preceded and followed by a space between the two strings.

Example:

```
MessageLabel.Text="Your name is: " & NameTextBox.Text
NameAndAddressLabel.Text=NameTextBox.Text & " 
AddressTextBox.Text
```

Continuing Long Program Lines

- For long lines of code, it is more readable to continue them on the next line.
- At the end of the line use a Line Continuation Character (a Space, an Underscore and press Enter).

```
GreetingsLabel.Text="Greetings " & NameTextBox.Text & ". " & 
"You have been selected to win a free prize. " & 
"Just send us &100 for postage and handling. "
```