Menus and Common Dialog Boxes

Chapter Objectives

- Create menus and submenus for program control
- Display and use the Windows common dialog boxes

Menus

- Menu Bar
  - Menus that drop down to display a list of menu items
  - Menu items are controls with properties and events
  - Each menu item has a Name and Text property and a Click event (similar to a button)
- Easy to create menus for a Windows form using Visual Studio's Menu Designer
  - Menus look and behave like standard Windows menus
**Separator Bars**

- Group menu items according to their purpose
- Draws a bar across the entire menu
- To create a separator bar, add a new menu item and click on its drop-down arrow
- Select Separator or type a single hyphen in the Menu Designer

**The Enabled Property**

- Menu items have Enabled property set to true by default
- Enabled menu items appear in black text and can be selected
- Disabled items are not available
- Enabled property can be set at design time or run time, in code

```csharp
ToolstripMenuItem.Enabled = false;
```

**The Checked Property**

- Menu items have Checked property set to false by default
- A check mark next to a menu item indicates the option is selected
- Checked property can be set at design time or in code

```csharp
ToolstripMenuItem.Checked = true;
```
**Toggling Check Marks On and Off**

- Set a menu item’s initial Checked property in the Properties window.
- To change the menu item’s state in code, set its Checked property to `true` or `false`.

```csharp
if (summaryToolStripMenuItem.Checked)
{
    // Uncheck the summary menu item.
    summaryToolStripMenuItem.Checked = false;
}
else
{
    // Check the summary menu item.
    summaryToolStripMenuItem.Checked = true;
}
```

**Setting Keyboard Shortcuts**

- Many users prefer to use keyboard shortcuts for selecting menu items.
- To set keyboard shortcuts:
  - Select the menu item in the designer.
  - In the Properties window, select the ShortcutKeys property.
  - Make a choice from the drop-down list.
  - Set ShowShortcutKeys property to `false` to prevent a shortcut from showing on the menu.

**Common Dialog Boxes**

- Predefined standard dialog boxes:
  - Specifying colors and fonts
  - Opening, saving and browsing for files
- Place the desired `common dialog` component from the Dialogs tab of the toolbox in the Component Tray.
  - Default names for the components are fine.
Displaying a Windows Common Dialog Box

- Display the dialog box at run time using the `ShowDialog` method.
- The `dialogObject` is the name of the common dialog component.
- Write code to show the dialog in the event handler for a menu item or button.

```csharp
colorDialog1.ShowDialog();
fontDialog1.ShowDialog();
```

Modal versus Modeless Windows

- A dialog box is said to be modal:
  - The box stays on top of the application and must be responded to.
  - Use the `ShowDialog` method to display a dialog box, which is a window displayed modally.
- A window that does not require response is said to be modeless:
  - Use the `Show` method to display a modeless window.

Using the Open File Dialog Box

- Color selected in the `Color` dialog box is stored in the `Color` property.
- Color property can be assigned to another object, such as a control:
  - Display dialog box with `ShowDialog` method.
  - User responds to dialog box.
  - Use the `Color` property.

```csharp
colorDialog1.ShowDialog();
titleLabel.ForeColor = colorDialog1.Color;
```
Using the Color Dialog Box (FYI)

- Color selected in the Color dialog box is stored in the Color property
- Color property can be assigned to another object, such as a control
  - Display dialog box with ShowDialog method
  - User responds to dialog box
  - Use the Color property
    ```csharp
colorDialog1.ShowDialog();
titleLabel.ForeColor = colorDialog1.Color;
```  

Using the Font Dialog Box (FYI)

- The Font common dialog box displays available fonts for the system display
- After the user makes a selection, the Font property can be assigned to other objects on the form  
  ```csharp
dialogBox1.ShowDialog();
titleLabel.Font = dialogBox1.Font;
```  

Setting Initial Values

- Before executing the ShowDialog method, assign the existing values of the object’s properties
- When the dialog box appears, the current values will be selected
- If the user presses Cancel, property setting for the objects will remain unchanged
  ```csharp
colorDialog1.Color = titleLabel.ForeColor;
colorDialog1.ShowDialog();
titleLabel.ForeColor = colorDialog1.Color;
```