Chapter 6
Multiform Projects

Chapter Objectives

- Include multiple forms in an application
- Create a new instance of a form's class and show the new form
- Use the `Show`, `ShowDialog`, and `Hide` methods to display and hide forms
- Declare variables with the correct scope and access level for multiform projects
- Create new properties of a form and pass data values from one form to another

Using Multiple Forms

- Projects can appear more professional if different windows are used for different types of information
  - Example: Display summary information on a well designed form instead of in a message box
- The first form a project displays is called the startup form
- A project can have as many forms as you wish
Creating New Forms – 1

- Select **Add Windows Form** from the **Project** menu
  - Select from many installed templates in the **Add New Item** dialog box
  - Choose **Windows Forms** in **Categories** list
  - Choose **Windows Form** as the template
  - Enter a name for the new form and click **Add**

Creating New Forms – 2

- After adding a new form, it appears on the screen and is added to the Solution Explorer window
- While in design time, switch between forms
  - In the Solution Explorer window
    - Select a form name, click **View Designer** or **View Code** button
  - Use tabs at top of the Document window
    - If too many tabs to display all of the forms, click the **Active Files** button to drop down a list and make a selection
- Each form is a separate class with separate files

Creating New Forms – 3

- Click on tabs to switch between forms
- Drop down the list of active files and make a selection
### Adding Existing Form Files to a Project

- Forms may be used in more than one project
  - Use a form created for one project in a new project
- A form is saved as three separate files
  - Extensions: .cs, .designer.cs, and .resx
- To add an existing form to a project, use **Project/Add Existing Item**
  - Select only the FormName.cs file
    - All three files for that form will be automatically copied into the project folder

### Removing Forms from a Project

- Select the name of the file to be removed in the Solution Explorer and do one of the following:
  - Press the Delete key
  - Right-click the file name and choose:
    - Delete
    - Choose **Exclude From Project**
      - Removes the form from the project but does not delete the files

### Creating a New Instance of a Form

- Each form in a project is a class which can be used to create a new object
  - Similar to controls placed on forms
- Create a new instance of a form before it can be displayed
**Showing a Form**

- **Show method** displays a form as *modeless*
  - Both forms are open and a user can navigate from one form to the other.
- **ShowDialog method** displays a form as *modal*
  - A user must respond to the form in some way
  - No other program code can execute until the user responds to, hides, or closes a modal form.

**The Show and ShowDialog Methods**

- **General Form**
  ```csharp
  FormObjectName.Show();
  FormObjectName.ShowDialog();
  ```

- **Examples**
  ```csharp
  SummaryForm aSummaryForm = new SummaryForm();
  aSummaryForm.Show();
  SummaryForm aSummaryForm = new SummaryForm();
  aSummaryForm.ShowDialog();
  ```
  
  - The code is generally placed in a menu item or button's click event handler.

**Hiding or Closing a Form**

- Close a form using the *Close method*
  ```csharp
  aSummaryForm.Close();
  ```

- The *Close* method behaves differently for a modeless form (using the *Show* method) compared to a modal form (using the *ShowDialog* method)
  - Modeless—Close destroys the form instance and removes it from memory
  - Modal—Close only hides the form instance
    - if the same instance is displayed again, any data from the previous ShowDialog will still be there

- Using the form's *Hide* method sets the form object's Visible property to False and keeps the form object in memory ready to be redisplayed.
Responding to Form Events

- Two primary events to write code for
  - `FormName.Load`
    - Occurs only the first time a form is loaded into memory
  - `FormName.Activated`
    - Occurs after the Load event, just as control is passed to a form
    - Occurs each time the form is shown
    - A good location to place initializing steps or set the focus in a particular place on the form

The Sequence of Form Events – Opening

- Load – Occurs the first time a form is displayed
- Activated – Occurs each time the form is shown
  - Place initialization or SetFocus events here
- Paint – Occurs each time any portion of the form is redrawn
  - Happens each time a change is made or the form is moved or uncovered

The Sequence of Form Events – Closing

- Deactivate – Occurs when the form is no longer the active form
  - When a user clicks another window or the form is about to be hidden or closed
- FormClosing – Occurs as the form is about to close
- FormClosed – Occurs after the form is closed
### Variables and Constants in Multiform Projects

- Private variables and constants within a form (or other class) cannot be seen by code in other classes
  - Private is the default access level
- To declare variables as Public makes them available to all other classes
  - Public variables present security problems and violate the principles of object oriented programming (OOP)
- The correct approach for passing variables from one form object to another is to set up properties of the form’s class
  - Use a property method to pass values from one form to another

### Running Your Program Outside the IDE

- The .exe file is in the project’s bin\Debug folder
- It can be moved to another computer, placed on the system desktop, or used as a shortcut, like any other application
- To copy the .exe file to another computer, make sure it has the correct version of the Microsoft .NET Framework
  - The framework can be downloaded for free from the Microsoft Web site
- You can change the program’s icon in the Project Designer (Project/ProjectName Properties)
  - Application tab, click Browse button next to Icon and browse to another file with an .ico extension
  - Recompile after setting a new icon