Chapter 11
Using Multiple Forms with Problem Domain Classes

Objectives
In this chapter, you will:
• Develop a GUI class that interacts with a PD class
• Simulate interaction with a database
• Develop a GUI class that interacts with multiple PD classes
• Navigate multiple forms in an integrated system
• Develop a GUI class that navigates a PD association relationship

Developing a GUI Class that Interacts with a PD Class

• AddCustomer
  – A GUI class
  – Used to
    • Input customer attribute values
    • Create instances of the Customer class

Developing a GUI Class that Interacts with a PD Class

• GUI form for adding a new customer contains:
  – A label
    • For the Bradshaw Marina logo
  – Three text boxes
    • txtName: to input customer name
    • txtAddress: to input customer address
    • txtPhone: to input customer phone number
  – Three labels
    • To identify the text boxes

Adding Variables to the Generated Code

• Variables needed to respond to events
  – Not generated by the visual programming process
  – Must be added to the source code through the Code Editor window
  – Often declared with class scope
Adding Variables to the Generated Code

- Variables needed for this example
  - A Customer reference
    - To create a new Customer instance from the data entered by the user
  - Three String variables
    - To contain the customer data that is retrieved from the text boxes

Handling Events

- Push buttons on the form
  - btnAdd_Click
    - Event: add a customer
  - btnClear_Click
    - Event: clear the form
  - btnClose_Click
    - Event: close the form

- Event handling methods
  - btnAdd.Click
  - btnClear.Click
  - btnClose.Click

Handling Events

- btnClear_Click method
  - Calls the ClearForm method, which
    - Stores an empty string in each of the text boxes
    - Invokes the Focus method for the txtName control
      - Focus method
        - Positions the cursor within the Name text box
        - Sets the input focus to this control
  - btnClose_Click method
    - Closes the form

Handling Events

- btnAdd_Click method
  - Retrieves the data from the text boxes
  - Validates values for name, address, and phone
    - if any of the values are missing, a message box prompts the user to enter all of the required data
    - if all three data items are entered
      - A Customer instance is created
      - A “Customer Added” message box is displayed
      - The form is cleared

Simulating Interaction with a Database

- FindCustomer class
  - Used to find a customer and then display related information
  - Simulates interaction with a database

Simulating Interaction with a Database

- The form to find a customer includes:
  - A customer list box
  - lstCustomer
  - Two text boxes
    - txtCustomerAddress
    - txtCustomerPhone
  - Three buttons
    - btnFind
    - btnUpdate
    - btnClose
  - A label for the Bradshaw Marina logo
    - lblLogo
Simulating Interaction with a Database

Creating an Array List of Customers

• FindCustomer simulates interaction with a database by
  – Creating six Customer instances
  – Populating an array list with Customer references

• ArrayList class
  – A member of the System.Collections namespace
  – Creates an array list, which
    • Works like an array
    • Is dynamic rather than fixed in size
  – Add method: appends an element to the end of an array list

Creating an Array List of Customers

• CreateCustomers method
  – Adds Customer instances to the array list
  – Invoked by the PopulateListBox method

• PopulateListBox method
  – Uses Customer instances to add customer names to the list box
  – Invoked from within the constructor of the FindCustomer class

Handling Events

• btnFind_Click method
  – Invoked when the user clicks the Find button
  – Purpose
    • Determine which customer name is selected on the list box
    • Retrieve and display that customer’s address and phone number

Handling Events

• btnUpdate_Click method
  – Invoked when the user clicks the Update button
  – Purpose
    • Allow the user to change a customer’s address and phone number

• btnClose_Click method
  – Invoked when the user clicks the Close button
  – Purpose
    • Closes the form

Developing a GUI Class That Interacts with Multiple PD Classes

• AddBoat class
  – A GUI class used to add a new boat to the Bradshaw system
  – Interact with the Boat, Sailboat, and Powerboat classes to create instances of each type of boat
Developing a GUI Class That Interacts with Multiple PD Classes

- **AddBoat GUI form**
  - **Upper portion**
    - Labels and text boxes to input the four attribute values for Boat

- **Middle portion**
  - Two radio buttons (radSailboat and radPowerboat)
  - Two panel instances (pnlSailboat and pnlPowerboat)
  - User must select either the Sailboat or the Powerboat radio button
    - If the Sailboat button is selected, pnlSailboat displays (shown in Figure 11-4)
    - If the Powerboat button is selected, pnlPowerboat displays (shown in Figure 11-5)

Handling Events

- **radSailboat_CheckedChanged method**
  - Determines whether the radSailboat button is checked; if so
    - Powerboat panel is hidden
    - Sailboat panel is displayed

- **radPowerboat_CheckedChanged method**
  - Determines whether the radPowerboat button is checked; if so
    - Sailboat panel is hidden
    - Powerboat panel is displayed
**Writing the ClearForm Method**

- **btnClear_Click event handling method**
  - Invokes the ClearForm method
- **ClearForm method**
  - Blanks out all of the text boxes by storing empty string in them
  - Sets the checked status of radio buttons used within the form back to their default values
  - Sets the focus to the Manufacturer text box

**Writing the btnAdd_Click Method**

- **btnAdd_Click method**
  - Retrieves the manufacturer, length, year, and registration information from text boxes
  - Performs simple data validation of the information
  - Invokes either AddSailboat or AddPowerboat, depending on which radio button is selected

**Writing the AddSailboat method**

- Receives the registration, length, manufacturer, and year into parameter variables
- Retrieves the number of sails and keel depth from the text boxes
- Displays an error message box if the data is not valid
- Converts the values for sails and keel depth to numeric data

**AddSailboat method (Continued)**

- Determines which of the three motor type radio buttons is selected
  - The appropriate value is placed into motorType
  - A new Sailboat instance is created
  - A "Sailboat Added" message is displayed in a message box

**Writing the AddPowerboat method**

- Similar to AddSailboat, except:
  - Number of engines is retrieved
  - There are only two radio buttons to test:
    - radGasoline
    - radDiesel

**Navigating Multiple Forms in an Integrated System: Simulating a Data Access Class**

- **OO systems**
  - Employ a three-tier design consisting of
    - GUI classes: provide a graphical interface for the input and display of data
    - PD classes: model the business entities and processes
    - DA classes: provide data storage and retrieval services
  - Advantage of three-tier design
    - Classes in each tier can be independent of those in another
Simulating a Data Access Class

• DA methods
  – Invoked by GUI classes which are unaware of their implementation
  – Can later be used with a real database management system without changing the GUI classes

• CustomerData class
  – Contains DA methods
  – Uses an array list to simulate the database

Simulating a Data Access Class

• DA methods of the CustomerData class:
  – Initialize
    • Performs initialization tasks
  – GetAll
    • Retrieves references to all instances of the Customer class that are stored in the database
  – AddNew
    • Stores a reference to a new Customer instance into the database
  – Update
    • Replaces a reference to a particular Customer instance with a reference to a new Customer instance

Integrating Multiple Forms

• Forms of the Integrated system
  – Main menu
    • Appears when the system starts
  – AddCustomer
  – FindCustomer

• Users navigate to other forms by clicking buttons on the main menu

Designing the Main Menu

• Main menu form
  – Integrates the forms
  – Serves as the startup object for the entire project
  – Contains three buttons
    • btnAdd
    • btnFind
    • btnClose

Integrating Multiple Forms

• GUI sequence to find and add a customer

Designing the Main Menu

• Main menu methods
  – btnClose_Click method
    • Closes the form
  – btnAdd_Click method
    • Hides the main menu form
    • Instantiates the AddCustomer GUI form
    • Displays the AddCustomer GUI form by invoking the ShowDialog method
Designing the Main Menu

- Main menu methods (Continued)
  - btnFind_Click method
    - Hides the main menu form
    - Instantiates the FindCustomer GUI form
    - Displays the FindCustomer GUI form by invoking the ShowDialog method
  - Initialize method
    - Invoked from within the constructor of the MainMenu class

Developing a GUI class that Navigates a PD Association Relationship

- Customer and Boat have a one-to-one association
  - Customer has a reference attribute for Boat
  - Boat has a reference attribute for Customer
  - The Boat instance method AssignBoatToCustomer populates both attributes

Developing a GUI class that Navigates a PD Association Relationship

- Customer and Boat have a one-to-one association
  - Customer has a reference attribute for Boat
  - Boat has a reference attribute for Customer
  - The Boat instance method AssignBoatToCustomer populates both attributes

Understanding the CustomerAndBoatData Class

- CustomerAndBoatData class
  - Simulates a database that contains both customer and boat information
  - Initialize method invokes two procedures
    - InitializeCustomer
      - Creates six Customer instances
    - InitializeBoat:
      - Creates six Boat instances: three sailboats and three powerboats
      - Invokes the AssignBoatToCustomer method of the Boat class to link each boat to a customer

Understanding the CustomerAndBoatData Class

- CustomerAndBoatData class (Continued)
  - GetAll method
    - Returns a reference to an array list containing the Customer references
  - Update methods
    - Modifies the attributes of a particular customer in the array list

Understanding the CustomerAndBoatData Class

- CustomerAndBoatData class (Continued)
  - AddNewBoat method
    - Adds a Customer reference to the customer array list
    - Adds a reference to the customer’s boat to the boat array list
Designing the GUI Sequence

• GUI sequence to add a new customer and boat
  – In main menu, click the Add Customer and Boat button
  – AddCustomer form is displayed
  – Enter the customer’s name, address, and phone number
  – Click the Add Boat button
  – AddBoat form is displayed
  – Enter the boat information
  – Click the Add Customer and Boat button
  – Message “Customer and Boat Added” appears

Finding a Customer and Boat

• Customers and their boats are linked with an association relationship
• FindCustomer class
  – Used to display the customer and boat information together

Summary

• ArrayList class is similar to an array but is dynamic rather than fixed in size
• You can simulate interaction with a database by using one or more array lists
• You can use panels and radio buttons to dynamically change the appearance of a form in response to user-generated events
• A main menu has push buttons that launch other GUIs to accomplish various tasks

Summary

• A GUI class can be designed to add customers and their boats together by linking the AddCustomer GUI to the AddBoat GUI
• A single GUI class can be used to find a customer, display customer data, and display the customer’s boat information using the association relationship between the Customer and Boat classes