Programming in Visual C# 2008

- Describe the process of visual program design and development
- Explain the term object-oriented programming
- Explain the concepts of classes, objects, properties, methods, and events
- List and describe the three steps for writing a C# program
- Describe the various files that make up a C# project
Writing Windows Applications

- Your C# programs will look and act like standard Windows programs
- Use tools to create:
  - Labels
  - Text boxes
  - Buttons
  - Radio buttons
  - Picture box
  - Check box
  - Menu Bar
  - Drop-down list
  - List box
  - Group box
Windows Graphical User Interface (GUI)

- Windows are called **forms**
- Elements are called **controls**
- Add controls to your forms using a toolbox
- The controls that you add look like and behave like standard Windows controls
Programming Languages—Procedural, Event Driven, and Object Oriented

- **Procedural**—Cobol, Fortran, Basic
  - Program specifies exact sequence of operations
- **Event Driven**—Early versions of Visual Basic
  - Contains some elements of object orientation but not all
- **Object Oriented Programming (OOP)**
  - User controls the sequence
  - Currently the most popular style of programming
The Object Model

- Objects have properties, methods and events
- An object is based on a class
  - Objects (noun or thing)
- Forms, Controls
  - Properties (adjectives)
    - Name, Color, Size, Location
  - Methods (verbs)
    - Close, Show, Clear
  - Events (occurs when user takes action)
    - Click, KeyPress, Scroll, Close window
  - Classes (template or blueprint)
    - Contain definition of all available properties, methods and events
OOP Terminology

- **Object.Property**
  - SalesForm.Text

- **Object.Method**
  - BillingForm.Show
  - exitButton.Show
Microsoft’s Visual Studio 2008

.NET 3.5 Framework

- Environment that allows objects from different languages to operate together, standardizes references to data and objects
- .NET languages all compile (translate) to Microsoft Intermediate Language (MSIL)
- MSIL runs in the Common Language Runtime (CLR)

- Programming Languages
  - Visual C#, Visual C++, Visual Basic

- C# Versions
Writing C# Programs – The Three-Step Process (Planning)

Planning

- Design the user interface
  - Sketch screens
  - Show forms and all controls
    - Name forms and objects on form
  - Consult with user
- Plan the properties
- Plan the C# code
  - Plan classes and methods
  - Write pseudocode
Writing C# Programs – The Three-Step Process (Programming)

- Define user interface
  - Create forms and controls
- Set the properties
  - Name each object and define its attributes
- Write the code
  - Define the actions of the program
C# Application Files

- **Solution file**
  - A C# application is called a solution and can consist of one or more projects
  - .sln extension

- **Project file**
  - Describes project and lists files included
  - .csproj extension

- **Form files**
  - .cs, .Designer.cs, .resx extensions

- The Visual Studio environment creates several more files
The Visual Studio Environment

● Integrated development environment (IDE)
  ➢ Includes various tools
    ▪ Form designer
    ▪ Editor for entering and modifying C# code
    ▪ Compiler
    ▪ Debugger
    ▪ Object Browser
    ▪ Help
Default Environment Settings

- Visual Studio 2008 allows selection of default IDE profile
  - Available with full version of Visual Studio
  - Choose Visual C# Development Settings
    - Text uses the Visual C# settings
    - Settings can be saved for more than one language
    - To switch between the settings for languages:
      - Select Tools/Import and Export Settings
      - Choose Reset all settings
The New Project Dialog

- Select *File/New Project*
  - New Project Dialog Box Opens

Select Visual C# Windows

Select the Windows Forms Application template

Enter the Project Name
The IDE Main Window

Each window can be moved, resized, closed, or customized
The Toolbars

- Toolbar buttons are shortcuts
- Each button’s command is also a menu selection
- Select View/Toolbars to display or hide a toolbar
- Three commonly used toolbars
  - Standard
  - Layout
  - Text Editor
Document Window

- Use tabs to switch between open documents
- Items in Document window
  - Form Designer
  - Code Editor
  - Project Designer
  - Database Designer
  - Object Browser
The Form Designer

- Design forms for user interface
- Change size of form using sizing handles or selection border
- A new default form is added when a new C# application is started
The Solution Explorer and Properties Windows

- **Solution Explorer window**
  - Holds filenames for project files
  - Lists classes referenced
  - Holds name of solution (.sln) file

- **Properties Window**
  - Used to set properties for objects in project
Toolbox

- Holds tools to place controls on form
- Tools vary depending on edition of C#
Design Time, Run Time, Debug Time

- **Design Time**
  - Design user interface (forms)
  - Write code
- **Run Time**
  - Testing project
  - Running project
- **Debug Time**
  - Run-time errors
  - Pause program execution
Writing Your First C# Project - 1

- Run Visual Studio
- Start a New Project
  - Select File/New/Project
    - Visual C# and Windows must be selected for Project types
    - Select Windows Forms Application for the template
  - Enter Hello World as the project name
  - Click the OK button
Writing Your First C# Project - 2

A project for creating an application with a Windows Forms user interface (NET Framework 3.5)

Name: HelloWorld
Location: C:\Users\UCB\Documents\Visual Studio 2008\Projects
Solution: Create new Solution
Solution Name: HelloWorld
Writing Your First C# Project - 3

- Set Up Your Environment
  - Click **Window/Reset Window Layout**, Click **Yes**
  - Point to toolbox icon
    - Pushpin icon toggles Auto Hide feature for toolbox
  - Optional
    - **Tools/Options**
    - **Options** dialog box
    - **Startup** under **Environment**
    - **At startup** list
      - *Show empty environment*
Writing Your First C# Project - 4

- Plan the project
  - Design the user interface
  - Plan the properties
  - Plan the code

(Planning the properties and code have been done for the first sample project)
● Define the User Interface
  ➢ Set Up the Form
    • Drag handle to resize form
  ➢ Place Controls on the Form

Double-click the Label tool
Writing Your First C# Project - 6

- Draw a button on the form
  - Click the button tool and draw with the crosshair pointer
- Create a second button
  - Double-click the Button tool in the toolbox
- Use snap lines to align all three controls
Writing Your First C# Project - 7

- Controls
  - Select, Delete, Move or Resize

<table>
<thead>
<tr>
<th>Action</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a control</td>
<td>Click on it</td>
</tr>
<tr>
<td>Delete a selected control</td>
<td>Select control and press the Delete key</td>
</tr>
<tr>
<td>Move a selected control</td>
<td>Point inside control (not on a handle) and drag to a new location</td>
</tr>
<tr>
<td>Resize a selected control</td>
<td>Point to a resizing handle and drag to resize</td>
</tr>
</tbody>
</table>
Writing Your First C# Project - 8

- Set Properties
  - Press F4 key if Properties window is not displayed
  - Click title bar of Properties window to make active window
  - Click on control to change its properties
  - Click Alphabetical button to sort properties
  - Select Name property
    - Enter Name for Control (used in code)
  - Select AutoSize property
    - Choose True or False
  - Select Text property
    - Change value for display on form
Lock the Controls

- When controls are in desired location, lock them in place
- Unlock controls by clicking again on *Lock Controls*

Right-click on the form and select Lock Controls

A button that is selected and locked
Writing Your First C# Project - 10

● Set Button Properties
  ➢ button1
    ▪ Name - displayButton
    ▪ Text - Display
  ➢ button2
    ▪ Name – exitButton
    ▪ Text – Exit
Writing Your First C# Project - 11

- Change Properties of the Form
  - Text property appears in the form’s title bar
  - Set StartPosition property to CenterScreen

Icon indicates that the form’s controls are locked

Form’s Text property appears in the title bar
Changing a Form’s Name

- You must change the name of the form’s file and the name of the class
- If you change the filename first, the IDE automatically changes the name of the class
  - The reverse is not true
HelloForm.cs

Name of the form's file

File name

Name of the form class
Write Code

• C# Events
  ➢ While a project is running the user can perform actions, each of which causes an event to occur
  ➢ C# will execute a method when code has been written for an event
  ➢ C# ignores events for which no methods are written
C# Event Handlers

- Write C# code in methods
  - Each begins with the words `private void`
  - Code in method is enclosed in braces `{ }`
- C# names the event-handling method as the object name, an underscore (_) and the name of the event
  - `displayButton_Click`
C# Code Statements

- Comment statements
  - Sometimes called *remarks*

- Ending a statement
  - Code statements are terminated by a semicolon (;)

- Assignment statements

- Executing a Method
  - Methods always have parentheses
  - Use *this* keyword to execute method of the current form
The Comment Statement

- Comments or *remarks* are for project documentation
- Explain purpose of the program
- Provide identifying information
- Non-executable statements within the code
- Automatically colored green by the editor
- Single-line comments begin with two slashes
  
  // Display a message to the user.

- Multiline comments begin with /* and end with */
  
  /* First line
     ...
   Last line */
Ending a Statement

- Most C# statements end with a semicolon (;)
- A statement can appear on multiple lines
- The semicolon determines the end of the statement
- Comments and a few other C# statements do not end with a semicolon
The Assignment Statement

- Assigns a value to a property or variable
- Operates from right to left
  - The value on the right side of the equal sign is assigned to the property on the left side of the equal sign
- Enclose text (a literal) in quotation marks (" ")

```csharp
messageLabel.Text = "Hello World";
```
Ending a Program by Executing a Method

- To execute a method of an object, write:
  
  Object.Method();

- Methods are always followed by parentheses
  
  ➢ Can help distinguish methods from properties

- To execute a method of the current form use the `this` keyword

  this.Close();

- `this.Close()` is used to end a program in the event-handling method for an Exit button or menu choice
Code Event-Handling Methods - 1

- Double-click the *Display* button
- The Editor window opens
- The header line for the method is in place
- The insertion point appears inside the opening and closing braces

```csharp
private void displayButton_Click(object sender, EventArgs e)
{
    //
}
```
Code Event-Handling Methods - 2

Editor tab

Form Designer tab

private void displayButton_Click(object sender, EventArgs e)
{
    // Display the Hello World message.
    messageLabel.Text = "Hello World";
}

private void exitButton_Click(object sender, EventArgs e)
{
    // Exit the project.
    this.Close();
}
Run the Project - 1

- Use one of three techniques to run
  - Open the **Debug** menu / **Start Debugging**
  - Press the **Start Debugging** toolbar button
  - Press F5, the keyboard shortcut

IDE Title bar indicates program is in run time

Editor and Form Designer locked tabs

Form for the running application

CSC 330 Object-Oriented Programming
Run the Project - 2

Click the Display button

"Hello World" message appears in the label

Click the Exit button to end the project and return to Design time
Save Your Work

- **File menu/Save All**
  - Saves the current form, project and solution files
- **Click Save All toolbar button to quickly save all files**
- **File menu/Close Solution**
  - If file has not been saved since last change, prompt to save appears
Open the Project

Three ways to open a project

- File menu/Open Project
  - Browse to find .sln file (small “9” as part of file’s icon)
- Choose project from File/Recent Projects menu
- Start Page (if available)
  - View/Other Windows/Start Page
Modify the Project

- Modify the project by making changes to the design (including properties) and/or the code (including comments)
- Double-click on a button to open the editor
  - Editor will open in corresponding click event code
Print the Code

- Print
  - Open Editor window
  - File/Print

- View Event Handlers
  - Select a button control
  - Click Events button (lightning bolt) on Properties window
  - Double-click on the event name
    - Code Editor window opens inside template for event handler
Automatically Generated Code

- **Using Statements**
  - Provide references to classes from the language library
    - `System.Windows.Forms`
      - Allows program to refer to all Windows controls

- **Namespace Statement**
  - .NET Framework requires every program component to have a namespace

- **Class Statement**
  - A new class can inherit from another class
Finding and Fixing Errors

- Syntax errors
  - Breaking the rules of the language
  - Editor finds most, compiler reports remainder
    - Editor identifies syntax errors with a red squiggly line

- Run-time errors or exceptions
  - Program halts due to statements that cannot execute
    - For example, impossible arithmetic operations

- Logic errors
  - Program runs but produces incorrect results
  - Carefully proof output to make sure it is correct
    - Computations, text and spacing
Project Debugging

- Program errors are bugs
- Editor highlights syntax and run-time errors
- Programmer must locate logic errors
- C# allows edit-and-continue
  - Fix error and press F5 or Debug/Continue
- Clean Compile
  - No errors during the compile process
Modifying an Event Handler -1

- Deleting an Event Handler
  - Select form or control
  - Click *Events* button in Properties window
  - Select and delete name of the method

Select the event
Delete the method name
Modifying an Event Handler - 2

- Renaming a Control
  - Change the name of the control in the design window
  - Switch to Editor window
  - Right-click name of event-handling method
  - Select Refractor/Rename
  - Enter new name in Rename dialog box
  - Click Apply in Preview Changes dialog box
Naming Rules and Conventions for Objects

- Follow C# rules for naming objects, methods and variables
- Good programmers follow naming rules and conventions
- Most programming shops have standards that must be followed
- When programmers consistently follow standards, programs are easier to read and to maintain, by both the original programmer and by others
The Naming Rules

- Name must begin with a letter or an underscore
- Name can contain letters, digits, and underscores
- Name cannot include a space, punctuation mark or be a reserved word
- Names are case sensitive
  - exitbutton, ExitButton and exitButton refer to three different objects
The Naming Conventions

- Follow standard naming conventions
- Use camel casing
  - Begin name with lowercase character
  - Capitalize each additional word in name
- Use meaningful names, indicate purpose of object
- Append full name of control’s class
  - Examples
    - messageLabel, exitButton, discountRateLabel
The Naming Conventions - 2

- Do not keep C# default names
  - Exception is labels that never change during execution

- Use pascal casing to name forms and classes
  - Capitalize first letter of name and first letter of all other words in name
  - Examples
    - HelloForm, MainForm, AboutForm
Visual Studio Help

- Installing and Running MSDN
  - Can be run from a hard drive or online

- Viewing Help Topics
  - Help window opens on top of IDE window
  - Filter by C# and either Express or Professional Edition

- Context-Sensitive Help
  - Select a C# object (i.e. form or control)
  - Place insertion point in a word in the editor
    - Press F1