Chapter 12

Basic Object-Oriented Program Concepts

Chapter Objectives

- Use object-oriented terminology correctly
  - Define polymorphism
    - What is overloading
  - Define encapsulation
  - Define abstraction and its relationship to the three-tiered model of programming
  - Define inheritance and its importance as an OO concept
    - Reusability

Object-Oriented Programming

- OOP is currently the most accepted style of programming
- C#, Java, and SmallTalk were designed to be object oriented (OO) from their inception
- Visual Basic and C++ have been modified to accommodate OOP
- As projects become more complex, using objects becomes increasingly important
Objects

- C# allows the creation of new object types by creating a class
  - Classes may have properties, methods, and events
- Many built-in choices for objects, such as all tools in the toolbox
  - The tool represents the class, such as button
  - The instance that you create from the class is the object, such as exitButton
- Defining a class creates a definition of what the object looks like and how it behaves
  - May create as many instances of the class as needed using the new keyword

Object-Oriented Terminology

- All object-oriented language have the following characteristics:
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Abstraction

Encapsulation

- Refers to the combination of the characteristics and behaviors of an object
  - One “package” or “capsule” that holds the definition of all properties, methods and events
  - Cannot make up new properties or tell the object to do anything it doesn’t already know how to do
- Can implement data hiding
  - An object can expose only those data elements and methods that it wishes to
    - public and private keywords determine access
Inheritance

- Ability to create a new class from an existing class
  - Add or modify class variables and methods in a new class that inherits from an existing class
- The purpose of inheritance is **reusability**
- Original class is called **base class**, **superclass**, or **parent class**
- Inherited class is called **subclass**, **derived class**, or **child class**
  - Inherited classes have an “is a” relationship with the base class

Polymorphism

- Methods that have identical names but different implementations depending on the specific object or arguments
  - Radio buttons, check boxes, and list boxes all have a **Select** method, which operates appropriately for its class
- **Overloading**—Several argument lists for calling the method
  - Example: MessageBox.Show method
- (FYI... **Overriding**—A method that has the same signature (name and parameter list) as its base class)
  - Method in subclass takes precedence, or overrides, the identically named method in the base class

Multitier Applications – Abstraction

- Each of the functions in a multitier application can be coded in a separate component and stored and run on different machines
- Most popular approach is a three-tier application
  - Presentation (user interface), Business (logic) and Data (retrieve and store in database) tiers
- Goal is to create components that can be combined and replaced
  - If one part of an application needs to change, other components do not need to be replaced
Multitier Applications – cont’d

- Common implementation of multitier application

<table>
<thead>
<tr>
<th>Presentation Tier</th>
<th>Business Tier</th>
<th>Data Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>Business Logic</td>
<td>Data Retrieval</td>
</tr>
<tr>
<td>Forms, controls, menus</td>
<td>Validation</td>
<td>Data storage</td>
</tr>
<tr>
<td></td>
<td>Calculations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Business rules</td>
<td></td>
</tr>
</tbody>
</table>