Notes for Chapter 4

The Object Concept

One of the underlying assumptions of the object-oriented methodology is that a number of entities exist that can be identified and described in terms of their current state and behaviors. By abstracting out the attributes (data) and the behaviors (processes on the data), you can create a class to serve as a template from which many objects of that type can be instantiated. Begin by determining what objects are needed in the solution. From that, determine what data characteristics will belong to the objects and what kind of behaviors the data will need to perform.

Private Member Data

When you define a class and determine what data members it should have, you declare instance variables or fields that represent the state of an object. These fields are declared inside the class, but not inside any specific method. They become visible to all members of the class, including all of the method members. Normally the data, instance variables, are defined to have a private access. This enables the class to protect the data and allow access to the data only through the class member methods.

Constructor

Constructors are special methods used to create instances of a class. Constructors differ from other methods in that they never return a value, but the keyword void is not included and the constructors use the same identifier as the class name. A public access modifier is always associated with constructors. Constructors are often overloaded.

When you write a class, you do not have to write a constructor; one is automatically created for you, called the default constructor. For the one automatically created for you, it has no functionality, other than the fact that it can be used to create an instance of the class. Default values are assigned to each member data field if the default constructor is used to create an object of the class type. No parameters are included with this one. It has no body, simply open and closed curly braces.

To add full functionality to your classes, you normally write one or more constructors for all classes in which objects will be instantiated. If you write even one constructor, you lose the default that is created automatically.

Writing your own Instance Methods

You do not use the static keyword in the heading. In order to call an instance method, an object must be instantiated and associated with the method.

Accessor

Accessors are special methods, also called getters, that access the value of a data member without changing it. Accessors are used because instance variables, data members of a class, are normally defined using a private modifier. The accessors are defined with public access. Many bodies of accessor methods consist of single return statements. A standard naming convention is to add “Get” to the front of the instance variable identifier.

Mutators

Mutators are special methods, also called setters, that change the current state or value of an object member’s data. A mutator normally is written to include one parameter that represents the new value a data member should have. Often the body of the method consists of a single assignment statement. A standard naming convention is to add “Set” to the front of the instance variable identifier.

Other Instance Methods

It is through instance methods that private data is manipulated. Instance method members of a class can directly access private data members of the same class.

Property

Properties looks like data fields but do not directly represent a storage location. They are more closely aligned to methods. They provide a way to set or get private member

data without defining separate getter and setter methods. A standard naming convention is to use the same name as the instance variable or field but start the identifier with an uppercase character.

It is not necessary to include both a set and get for every field. The body of the get must return a value of the property type. The body of the set portion in the property is similar to a mutator. Mutators normally have one argument representing the value to be assigned to the instance variable. The set portion of the property uses an implicit parameter, called value. If you include only the get, the property is considered a read-only property

Auto Implemented Property

You do not have to include return or set statements when you create auto-implemented properties. You simply write get; and/or set; as part of the property definition. If you choose to create auto implemented properties, do not define separate private data members to tie to the property. An anonymous backing field is automatically created.

Auto Properties Initializers

Auto Properties Initializers were introduced with C# 6.0 Like auto-implemented properties, you do not efine separate private data members to tie to the property. An anonymous backing field is automatically created. Simply add an equal symbol and a compatible value to be used for initialization.

ToString( ) Method

All user-defined classes inherit four methods (ToString( ), Equals( ), GetType( ), and GetHashCode( )) from the object class. The ToString( ) method is called automatically by several methods, including the Write( ) and WriteLine( ) methods. You can also invoke or call the ToString( ) method directly. To override the ToString( ) method, the keyword override is added to the heading immediately preceding the return type.

Calling Instance Methods

Calling the Constructor

Normally in an object-oriented development environment, the first method called is the constructor to create an object instance of the class. The keyword new is used as an operator to call constructor methods. Calls using the default constructor place default values in the memory locations. Table 4-3 shows the default values.

Calling Accessor and Mutator Methods

If the method is being called from another class that has instantiated an object, the method name is preceded by the object name and a dot. If the method is returning a value, as accessors do, there must be a place for the value to be returned.

Calling Other Instance Methods

If the method is being called from within the class, no reference to the class or object is needed. If the method needs arguments, those are sent as was done with static methods.

Testing Your New Class

A different class is needed for testing and using your class. This class will include a Main( ) method. Objects need to be instantiated of the other class and as many of the methods and properties invoked as possible.

PROGRAMMING EXAMPLE: REALESTATEINVESTMENT

This example demonstrates the use of methods in a program. Both static and instance methods are used. It begins by showing a problem specification that details the problem definition. The focus is placed on designing an object-oriented solution using two classes. The application written determines what the cash flow is for a real estate investment used as a rental.

Data members for the real estate class include the year the home was built, purchase price, street address, monthly income from rent, and monthly expense characteristics. The class has a method to determine the monthly cash flow as well as constructor

methods. The properties area is defined as opposed to accessor and mutator methods. In the second class, objects of the real estate property class are instantiated using the constructor. A static class method is defined to allow the user to input the expenses.

The output consists of a display showing the yearly income, expenses, property address, and the earnings for the month. A prototype for the output is shown with the example. Structured English (pseudocode) class diagrams are shown with the example.