**PRACTICAL # 09**

**OBJECT:**

Encapsulation and Inheritance

**THEORY:**

C# is Object-Oriented Programming language. The core concepts of object oriented programming include Encapsulation, Inheritance and Polymorphism.

**Encapsulation** is the hiding of data. It combines the members data and member methods into a single unit. In C#, encapsulation is achieved using classes which combines member variables and member methods. Encapsulation prevents alteration of data accidently. By binding the class fields with methods and properties, we can protect the data from accidental corruption. Encapsulation is achieved with access modifiers. An access modifier defines the scope and visibility of a class member variable. C# supports the following access modifiers.

• Public: allows a class to expose its member variables or member methods to other functions and objects. Any public member can be accessed from outside the class.

• Private: allows a class to hide its member variables or member methods from other functions and objects. Only functions of the same class can access its private members. Even an instance of a class cannot access its private members.

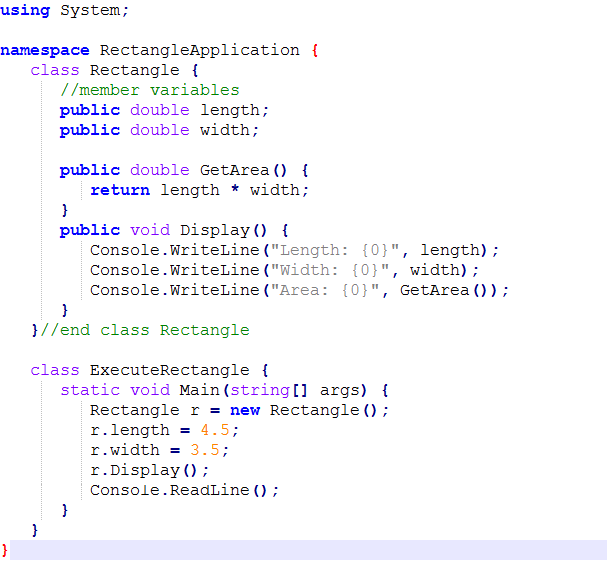
• Protected: allows a child class to access the member variables or member methods of its base class. This way it helps in implementing inheritance.

• Internal: allows a class to expose its member variables and member functions to other functions and objects in the current assembly. In other words, any member with internal access methods can be accessed from any class or method defined within the application in which the member is defined.

• Protected internal: allows a class to hide its member variables and member functions from other class objects and functions, except a child class within the same application.

**Program:**

The programs below demonstrate these access modifiers.



Output:

*Length: 4.5*

*Width: 3.5*

*Area: 15.75*

In the code above, the member variables are defined as public. Thus such variables can be modified outside of the class as shown in the ExecuteRectangle class. Thus any member variable can be modified accidentally. To solve this problem, we can encapsulate the member variables as private and set public properties or methods to avoid accidental modification and provide controlled access.



Output:

*Enter Length:*

*4.4*

*Enter Width:*

*3.3*

*Length: 4.4*

*Width: 3.3*

*Area: 14.52*

In the code above, the member variables are defined as private. So these variables cannot be modified outside of the class as shown in the ExecuteRectangle class. These private member variables can only be accessed using member methods.

**Inheritance:**

Inheritance is a one of the primary concept of object oriented programming (OOP) and it is used to inherit the properties from one class (base) to another (child) class. The inheritance will enable us to create a new class by inheriting the properties from other classes to reuse, extend and modify the behavior of other class members based on our requirements. The class whose members are inherited is called a base (parent) class and the class that inherits the members of base (parent) class is called a derived (child) class.

C# Inheritance Syntax

Following is the syntax of implementing an inheritance to define a derived class that inherit the properties of base class in c# programming language.

*<access\_modifier> class <base\_class\_name>*

*{*

*// Base class Implementation*

*}*

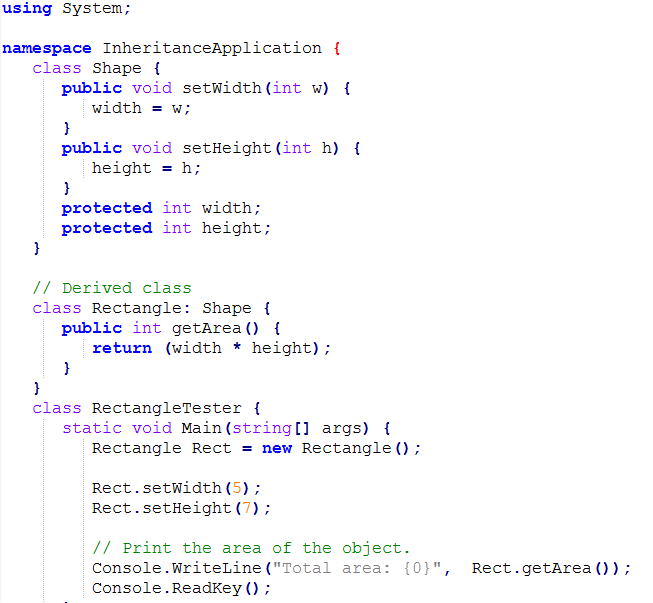
*<access\_modifier> class <derived\_class\_name> : <base\_class\_name>*

*{*

*// Derived class implementation*

*}*

Consider a base class shape and child class rectangle



Output:

Total area: 35

**ACTIVITIES**

**Activity 1**

Define a class named building with fields length, width, number of rooms, and number of floors. Make these fields private and provide public methods to get and set their values.

**REVIEW QUESTIONS**

1. What is encapsulation?
2. How is encapsulation implemented in C#?
3. What is inheritance?
4. What is multiple levels of inheritance?
5. Is a private member of base class accessible in the child class?