Polymorphism

Course Code: CSC 3115 Course Title: Object Oriented Programming 2

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	05	**Week No:**	05	**Semester:**	
Lecturer:					

Topics

1. Polymorphism
2. Method Overriding
3. Method Overloading

Polymorphism

Polymorphism originates from the greek word *"polys"* which means many, and *"morphe"* which means forms, or shapes.

Polymorphism is possible when classes share a common interface. For example, a Cat class and a Dog class may share a common interface through inheriting the Animal class.

To understand Polymorphism, we must take a look at how we instantiate objects.

```
Cat myCat = new Cat();
```

Here we declare the reference type Cat, the variable name myCat and instantiate a new Cat object by calling new Cat();

Polymorphism

When classes share a common interface, we can redefine the reference type of the objects we are creating. So our myCat definition can be changed to:

```
Animal myCat = new Cat();
```

For example, we can now create an Array of Animals that can contains Dog objects, Cat objects and any other subclass of Animal.

```
Animal[] animals = new Animal[2];
     animals[0] = new Cat();
     animals[1] = new Dog();
```

Polymorphism is a very powerful concept that improves code extensibility and modularity.

Polymorphism

- **Polymorphism** is ability to **re-implement** inherited **method** in derived class
- To allow this base class must declare the polymorphic method as **virtual**
- e.g. public **virtual** void DrawWindow()
- When re-implementing polymorphic method in derived class this method declaration must be preceded with keyword **override**
- **The method name, parameter signature must remain the same!!! (access level and return type can be re-defined)**

Polymorphism - Abstract Classes

- Idea of an abstract class – to lay out common characteristics and behaviors of all derived classes
- To enforce implementation of a method of its base this method in a base class must be declared as abstract.
- Abstract **method** has no implementation: just name and signature
- If a class contains at least one abstract method it becomes abstract itself
- Abstract class can not be instantiated.
- If abstract class has a constructor – it must be concrete

Polymorphism

- **sealed keyword** prevents derivation
- **The root of all Types: Object**
- **All C# classes derive from System.Object (including value types e.g. int, long…)**
- NOTE 1: Object provides virtual methods that subclasses can override.
- e.g. ToString() method
- NOTE 2: Classes do not need to explicitly declare that they derive from Object; the inheritance is implicit
- NOTE 3: in the example project above try comment out the following override:

```
//public override string ToString()
//{
//    return val.ToString();
//}
```

- And you will see that output will be invoke base method which will change the output to:

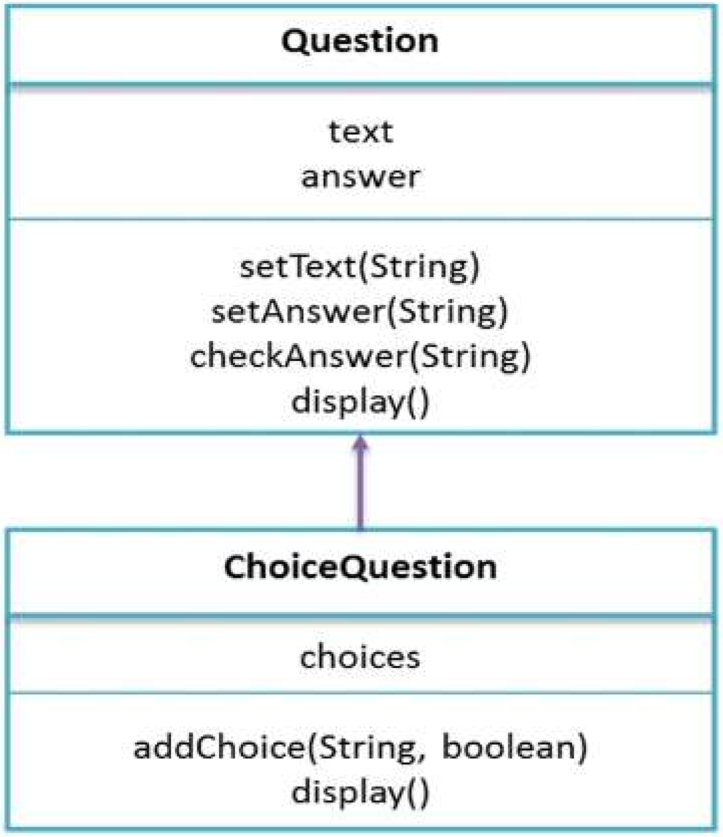
  InheritingFromObject.SomeClass

Polymorphism

- Implemented through virtual methods
- The method implementation to invoke is determined by the actual type of the object at “runtime”
- Polymorphism can be achieved through both “implementation” and “interface” inheritance (see interfaces)

Overriding

- When we create a subclass, we may want to extend the functionality of the inherited superclass methods.
- In order to override an inherited method, we must declare the same method signature in the subclass. Then inside the body of the method, we can provide our subclass specific functionality.



Overriding Example in C#

- If we declare a superclass method with the **virtual** keyword, we are saying that any subclasses can override this method. We then declare the method we are overriding in the subclass with the **override** keyword.

```
class Animal

{

    public virtual void Move()

    {

     Console.WriteLine("Moving.");

    }

}
```

```
class Rhino: Animal

{

    public override void Move()

    {

        Console.WriteLine("Charge!!!");

    }

}
```

```
Rhino myRhino = new Rhino();

myRhino.Move();
```

- C# will always calls the most specific method, i.e. the method that is lowest in the inheritance hierarchy. In this case the overridden Move() method will be called in the Rhino class and not the Animal class.

Overloading

- Overloading allows you to create multiple methods with the same name in the same scope however they differ in their method signatures.

```
public void Add(int number1, int number2)

public void Add(int number1, int number2, int number3)

public void Add(int number1, int number2, int number3,
int number4)
```

Sealed keyword

sealed keyword is used to define a class that cannot be inherited

```
public sealed class A
{
        ...

}
```

```
//Compiler error
public class B : A
{
        ...

}
```

Thank You

Books

- C# 4.0 The Complete Reference; Herbert Schildt; McGraw-Hill Osborne Media; 2010
- Head First C# by Andrew Stellman
- Fundamentals of Computer Programming with CSharp – Nakov v2013

References

MSDN Library; URL: http://msdn.microsoft.com/library

C# Language Specification; URL: http://download.microsoft.com/download/0/B/D/0BDA894F-2CCD-4C2C-B5A7-4EB1171962E5/CSharp%20Language%20Specixfication.doc

C# 4.0 The Complete Reference; Herbert Schildt; McGraw-Hill Osborne Media; 2010