**Object Oriented Programming Langage Part 3**

**21. What are shared (VB.NET)/Static(C#) variables?**

Static members are not associated with a particular instance of any class, which can be invoked directly from the class level, rather than from its instance

Example

public static double PI = 3.1457;

**22. What is Nested Classes?**

Classes with in classes are called as Nested class.

Example

public class MyClassLevel\_1

{

public void Display()

{

Console.WriteLine("Level\_1");

}

public class MyClassLevel\_2

{

public void Display()

{

Console.WriteLine("Level\_2");

}

public class MyClassLevel\_3

{

public void Display()

{

Console.WriteLine("Level\_3");

}

}

}

}

Creating instance of the nested class

MyClassLevel\_1 L1 = new MyClassLevel\_1();

MyClassLevel\_1.MyClassLevel\_2 L2 = new MyClassLevel\_1.MyClassLevel\_2();

MyClassLevel\_1.MyClassLevel\_2.MyClassLevel\_3 L3 = new

MyClassLevel\_1.MyClassLevel\_2.MyClassLevel\_3();

L1.Display();

L2.Display();

L3.Display();

Output

Level\_1

Level\_2

Level\_3

**23. What are difference between Singleton and Static class?**

1. Singleton can extend classes and implement interfaces, while a static class cannot implement the interface.
2. Singleton can be initialized lazily or asynchronously while a static class is generally initialized when it is first loaded.
3. Singleton class can be extended and it's methods can be overridden.

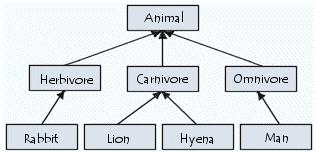
**24. Why Main () method is static?**

To ensure there is only one entry point to the application.

**25. What is mean by inheritance?**

Inheritance is one of the concepts of object-oriented programming, where a new class is created from an existing class. Inheritance class is often referred to as subclasses, comes from the fact that the subclass (the newly created class) contains the attributes and methods of the parent class. This can be used to create a highly specialized hierarchical class structure.

Example of Inheritance



class Circle

{

private double m\_radius;

public double Radius

{

get { return m\_radius; }

set { m\_radius = value; }

}

public double Diameter

{

get { return Radius \* 2; }

}

public double Circumference

{

get { return Diameter \* 3.14; }

}

public double Area

{

get { return Radius \* Radius \* 3.14; }

}

}

class Sphere : Circle

{

new public double Area

{

get { return 4 \* Radius \* Radius \* 3.14; }

}

public double Volume

{

get { return 4 \* 3.14 \* Radius \* Radius \* Radius / 3; }

}

}

**26. Can we inherit multiple classes?**

No, multiple inheritances are not supported in .Net. Because consider the provided example. Here there are two Parent class Paretn1 and Parent2. This is inherited by Child class, In *GetData* method, child call the parent class method *PrintData()*. In this case which method will be executed? It is very difficult for CLR to identify which method to call. It shows that we multiple inheritance create ambiguity to oops concept. In order to avoid this ambiguity we are going for multiple interface implementations.

public class Parent1

{

public string PrintData()

{

return "This is parent1";

}

}

public class Parent2

{

public string PrintData()

{

return "This is parent2";

}

}

public class Child1 : Parent1, Parent2

{

public string GetData()

{

return this.PrintData();

}

}

**27. What is mean by Shadowing?**

When the method is defined in base class are not override able and we need to provide different implementation for the same in derived class. In this kind of scenario we can use hide the base class implementation and provide new implementation using Shadows (VB.Net)/new(C#) keyword.

Example:

Public Class ParentClass

Public Sub Display()

Console.WriteLine("Parent class")

End Sub

End Class

Public Class ChildClass

Inherits ParentClass

Public Shadows Sub Display()

Console.WriteLine("Child class")

End Sub

End Class

Dim p As New ParentClass

Dim c As New ChildClass

Dim pc As ParentClass = New ChildClass

p.Display()

c.Display()

pc.Display()

**Output:**

Parent class

Child class

Parent class

**28. How a base class method is hidden?**

Using new keyword in the derived class, base class method can be hidden or suppressed. New implementation can be added to the derived class.

**29. What does the keyword virtual mean in the method definition?**

The method can be over-ridden.

**30. How method overriding different from overloading?**

If we are overriding the method, derived class method behavior is changed from the base class. In Overloading, method with same name by different signature is used.

**Example:**

{

public virtual void Display()

{

Console.WriteLine("ParentClass");

}

}

public class ChildClass : ParentClass

{

//Example for method override

public override void Display()

{

Console.WriteLine("ChildClass");

}

//Example for method overload

public void Display(string name)

{

Console.WriteLine(name);

}

//Example for method overload

public void Display(string name, string country)

{

Console.WriteLine("Name:"+name +"Country: "+ country );

}

}

ParentClass p = new ParentClass();

ChildClass c = new ChildClass();

ParentClass pc = new ChildClass();

p.Display();

c.Display();

pc.Display();

**OutPut:**

ParentClass

ChildClass

ChildClass

**31. Can you declare the override method static while the original method is non-static?**

No

**32. What is mean by Sealed Class?**

Class which cannot be inherited is called as sealed class. If we need to prevent a class from being inherited, use “Sealed” keyword. But sealed class can inherited from other classes.

**Example:**

public class MyBaseClass

{

public void Display()

{

Console.WriteLine("Base class");

}

}

//Compile Success: This class cannot be inherited

public sealed class MySealedClass:MyBaseClass

{

public void Display()

{

base.Display();

Console.WriteLine("Sealed class");

}

}

//Compilation Error: cannot derive from sealed type MySealedClass

public class MyChildClass : MySealedClass

{

}