**What is a class ?**  
  
A class is the generic definition of what an object is . A Class describes all the attributes of object, as well as the methods that implements the behavior of member object. That means, class is a template of an object. Easy way to understand a class is to look at an example . In the class Employee given below, Name and Salary are the attributes of the class Person , Setter and Getter methods are used to store and fetch data from the variable

|  |  |
| --- | --- |
| **Questions :** | **What is Object Oriented Programming ?** |
| **Answers :** | It is a problem solving technique to develop software systems. It is a technique to think real world in terms of objects. Object maps the software model to real world concept. These objects have responsibilities and provide services to application or other objects. |
|  |  |
| **Questions : 2** | **What is a Class ?** |
| **Answers : 2** | A class describes all the attributes of objects, as well as the methods that implement the behavior of member objects. It is a comprehensive data type which represents a blue print of objects. It’s a template of object. |
|  |  |
| **Questions : 3** | **What is an Object ?** |
| **Answers : 3** | It is a basic unit of a system. An object is an entity that has attributes, behavior, and identity. Objects are members of a class. Attributes and behavior of an object are defined by the class definition. |
|  |  |
| **Questions : 4** | **What is the relation between Classes and Objects?** |
| **Answers : 4** | They look very much same but are not same. Class is a definition, while object is instance of the class created. Class is a blue print while objects are actual objects existing in real world. Example we have class CAR which has attributes and methods like Speed, Brakes, Type of Car etc.Class CAR is just a prototype, now we can create real time objects which can be used to provide functionality. Example we can create a Maruti car object with 100 km speed and urgent brakes. |
|  |  |
| **Questions : 5** | **What are different properties provided by Object-oriented systems ?** |
| **Answers : 5** | **Following are characteristics of Object Oriented System’s:-** **Abstraction** It allows complex real world to be represented in simplified manner. Example color is abstracted to RGB.By just making the combination of these three colors we can achieve any color in world. It’s a model of real world or concept. **Encapsulation** The process of hiding all the internal details of an object from the outside world. **Communication** Using messages when application wants to achieve certain task it can only be done using combination of objects. A single object can not do the entire task. Example if we want to make order processing form. We will use Customer object, Order object, Product object and Payment object to achieve this functionality. In short these objects should communicate with each other. This is achieved when objects send messages to each other. **Object lifetime** All objects have life time. Objects are created, initialized, necessary functionalities are done and later the object is destroyed. Every object have there own state and identity, which differ from instance to instance. |
|  |  |
| **Questions : 6** | **What is an Abstract class ?** |
| **Answers : 6** | Abstract class defines an abstract concept which can not be instantiated and comparing o interface it can have some implementation while interfaces can not. Below are some points for abstract class:-  =>We can not create object of abstract class it can only be inherited in a below class.  => Normally abstract classes have base implementation and then child classes derive from the abstract class to make the class concrete. |
|  |  |
| **Questions : 7** | **What are Abstract methods?** |
| **Answers : 7** | Abstract class can contain abstract methods. Abstract methods do not have implementation. Abstract methods should be implemented in the subclasses which inherit them. So if an abstract class has an abstract method class inheriting the abstract class should implement the method or else java compiler will through an error. In this way, an abstract class can define a complete programming interface thereby providing its subclasses with the method declarations for all of the methods necessary to implement that programming interface. Abstract methods are defined using "abstract" keyword. Below is a sample code snippet. abstract class pcdsGraphics { abstract void draw(); } Any class inheriting from "pcdsGraphics" class should implement the "draw" method or else the java compiler will throw an error. so if we do not implement a abstract method the program will not compile. |
|  |  |
| **Questions : 8** | **What is the difference between Abstract classes and Interfaces ?** |
| **Answers : 8** | Difference between Abstract class and Interface is as follows:-  Abstract class can only be inherited while interfaces can not be it has to be implemented.  Interface cannot implement any methods, whereas an abstract class can have implementation.  Class can implement many interfaces but can have only one super class.  Interface is not part of the class hierarchy while Abstract class comes in through inheritance.  Unrelated classes can implement the same interface. |
|  |  |
| **Questions : 9** | **What is difference between Static and Non-Static fields of a class ?** |
| **Answers : 9** | Non-Static values are also called as instance variables. Each object of the class has its own copy of Non-Static instance variables. So when a new object is created of the same class it will have completely its own copy of instance variables. While Static values have only one copy of instance variables and will be shared among all the objects of the class. |
|  |  |
| **Questions : 10** | **What are inner classes and what is the practical implementation of inner classes?** |
| **Answers : 10** | Inner classes are nested inside other class. They have access to outer class fields and methods even if the fields of outer class are defined as private. public class Pcds { class pcdsEmp { // inner class defines the required structure String first; String last; } // array of name objects clsName personArray[] = {new clsName(), new clsName(), new clsName()}; } Normally inner classes are used for data structures like one shown above or some kind of helper classes. |
|  |  |
| **Questions : 11** | **What is a constructor in class?** |
| **Answers : 11** | Constructor has the same name as the class in which it resides and looks from syntax point of view it looks similiar to a method. Constructor is automatically called immediately after the object is created, before the new operator completes. Constructors have no return type, not even void. This is because the implicit return type of a class' constructor is the class type itself. It is the constructor's job to initialize the internal state of an object so that the code creating an instance will have a fully initialized, usable object immediately. |
|  |  |
| **Questions : 12** | **Can constructors be parameterized?** |
| **Answers : 12** | Yes we can have parameterized constructor which can also be termed as constructor overloading. Below is a code snippet which shows two constructors for pcdsMaths class one with parameter and one with out. class pcdsMaths { double PI; // This is the constructor for the maths constant class. pcdsMaths() {PI = 3.14;} pcdsMaths(int pi) { PI = pi; } } |
|  |  |
| **Questions : 13** | **What is the use if instanceof keyword? and How do refer to a current instance of object?** |
| **Answers : 13** | "instanceof" keyword is used to check what is the type of object.  we can refer the current instance of object using "**this**" keyword. For instance if we have class which has **color** property we can refer the current object instance inside any of the method using "**this.color**". |
|  |  |
| **Questions : 14** | **what is Bootstrap, Extension and System Class loader? or Can you explain primordial class loader?** |
| **Answers : 14** | There three types of class loaders:- BootStrap Class loader also called as primordial class loader.  Extension Class loader.  System Class loader. Let’s now try to get the fundamentals of these class loaders.   **Bootstrap Class loader** Bootstrap class loader loads those classes those which are essential for JVM to function properly. Bootstrap class loader is responsible for loading all core java classes for instance java.lang.\*, java.io.\* etc. Bootstrap class loader finds these necessary classes from “jdk/ jre/lib/rt.jar”. Bootstrap class loader can not be instantiated from JAVA code and is implemented natively inside JVM.  **Extension Class loader** The extension class loader also termed as the standard extensions class loader is a child of the bootstrap class loader. Its primary responsibility is to load classes from the extension directories, normally located the “jre/lib/ext” directory. This provides the ability to simply drop in new extensions, such as various security extensions, without requiring modification to the user's class path.  **System Class loader** The system class loader also termed application class loader is the class loader responsible for loading code from the path specified by the CLASSPATH environment variable. It is also used to load an application’s entry point class that is the "static void main ()" method in a class. |
|  |  |
| **Questions : 15** | **what’s the main difference between ArrayList / HashMap and Vector / Hashtable?** |
| **Answers : 15** | Vector / HashTable are synchronized which means they are thread safe. Cost of thread safe is performance degradation. So if you are sure that you are not dealing with huge number of threads then you should use ArrayList / HashMap.But yes you can still synchronize List and Map’s using Collections provided methods :- List OurList = Collections.synchronizedList (OurList); Map OurMap = Collections.synchronizedMap (OurMap); |
|  |  |
| **Questions : 16** | **What are access modifiers?** |
| **Answers : 16** | Access modifiers decide whether a method or a data variable can be accessed by another method in another class or subclass. four types of access modifiers: **Public: -** Can be accessed by any other class anywhere. **Protected: -**Can be accessed by classes inside the package or by subclasses ( that means classes who inherit from this class). **Private -**Can be accessed only within the class. Even methods in subclasses in the same package do not have access. **Default -**(Its private access by default) accessible to classes in the same package but not by classes in other packages, even if these are subclasses. |
|  |  |
| **Questions : 17** | **Define exceptions ?** |
| **Answers : 17** | An exception is an abnormal condition that arises in a code sequence at run time. Basically there are four important keywords which form the main pillars of exception handling: try, catch, throw and finally. Code which you want to monitor for exception is contained in the try block. If any exception occurs in the try block its sent to the catch block which can handle this error in a more rational manner. To throw an exception manually you need to call use the throw keyword. If you want to put any clean up code use the finally block. The finally block is executed irrespective if there is an error or not. |
|  |  |
| **Questions : 18** | **What is serialization?How do we implement serialization actually?** |
| **Answers : 18** | Serialization is a process by which an object instance is converted in to stream of bytes. There are many useful stuff you can do when the object instance is converted in to stream of bytes for instance you can save the object in hard disk or send it across the network.  In order to implement serialization we need to use two classes from java.io package ObjectOutputStream and ObjectInputStream. ObjectOutputStream has a method called writeObject, while ObjectInputStream has a method called readObject. Using writeobject we can write and readObject can be used to read the object from the stream. Below are two code snippet which used the FileInputStream and FileOutputstream to read and write from harddisk. |
|  |  |

**What is an Object?**  
  
Object is an instance of a class, it contains real values instead of variables. For example, lets create an instance of class Employee called “Jhon”.

**Employee Jhon = new Employee();**

Now we can access all the methods in the class “Employee” via object “Jhon” as shown below.  
  
**Jhon.setName(“XYZ”);**  
  
**3) What are the Access Modifiers in C# ?**  
  
Different Access Modifier are - Public, Private, Protected, Internal, Protected Internal

**Public** – When a method or attribute is defined as Public , It can be accessed from any code in project. For example in the above Class “Employee” , getName(), setName() etc are public.

**Private** - When a method or attribute is defined as Private , It can be accessed by any code within the containing type only. For example in the above Class “Employee” , attributes name and salary can be accessed with in the Class Employee Only. If an attribute or class are defined without access modifiers , its default access modifier will be private.

**Protected** - When an attribute and methods are defined as protected, it can be accessed by any method in inherited classes and any method within the same class. The protected access modifier cannot be applied to class and interfaces. Methods and fields in a interface cannot be declared protected.  
  
**Internal** – If an attribute or method is defined as Internal , Access is restricted to classes within the current project assembly  
Protected Internal – If an attribute or method is defined as Protected Internal , Access is restricted to classes within the current project assembly and types derived from the containing class.   
  
**4) Define Static Members in C#** ?  
  
If an attributes value had to be same across all the instances of the same class , static keyword is used. For example if the Minimum salary should be set for all employees in the employee class, use the following code

**private static double MinSalary = 30000;**

To access a private or public attributes or methods in a class, at first an object of the class should be created . Then using that instance of class , attributes or methods can be accessed. To access a static variable, we don't want to create an instance of the class having the static varibale . We can directly refer that static variable as shown below.  
  
**double var = Employee.MinSalary ;**  
  
**5) Define Reference Types in C#** ?  
  
Let us explain this with an example . For the code given below,

Employee emp1 ;  
Employee emp2 = new Employee();  
emp1 = emp2;

Here emp2 has an object instance of Employee Class . But emp1 object is set as emp2. What this means is that object emp2 is refereed in emp1 and not that emp2 is copied into emp1. When a change is made in emp2 object, corresponding changes can be seen in emp1 object.

**Define Property in C# ?**  
  
Properties are a type of class member , that are exposed to outside world as a pair of Methods.For example for the static field Minsalary, we will Create a property.

private double minimumSalary ;  
public static double MinSalary  
{  
get  
{  
return minSalary;  
}  
set  
{  
minSalary = value;  
}  
}

So when we execute the following lines code

double minSal = Employee. MinSalary;

get Method will get triggered and value in minimumSalary field will be returned.When we execute ,

Employee. MinSalary = 3000;

set Method will get triggered and value will be stored in minimumSalary field .  
  
  
**7) Define Overloading in C# ?**  
  
When methods are created with same name , but with different signature its called overloading.   
For example , WriteLine method in console class is an example for overloading. In first instance , it takes one variable. In the second instance , “WriteLine” method takes two variable.  
  
Console.WriteLine(x);  
Console.WriteLine("The message is {0}", Message);  
Different type of overloading in C# are  
  
1)Constructor overloading  
2)Function overloading.  
3)Operator overloading  
  
  
**8) Define Constructor Overloading in C# .net ?**  
  
In Constructor overloading, n number of constructors can be created for same class. But the signatures of each constructor should vary. For example

public class Employee   
{  
public Employee()   
{ }  
public Employee(String Name)   
{ }  
}

**9) Define Function Overloading in C# .net ?**  
  
In Function overloading, n number of functions can be created for same class. But the signatures of each function should vary. For example

public class Employee   
{  
public void Employee()   
{ }  
public void Employee(String Name)   
{ }  
}

**10) Define Operator Overloading in C# .net ?**  
  
We had seen function overloading in the previous example.For operator Overloading , we will have look at the example below. We define a class rectangle with two operator overloading methods.

class Rectangle  
{  
private int Height;  
private int Width;  
public Rectangle(int w,int h)  
{  
Width=w;  
Height=h;  
}  
}  
public static bool operator >(Rectangle a,Rectangle b)  
{  
return a.Height > b.Height ;  
}  
  
public static bool operator <(Rectangle a,Rectangle b)  
{  
return a.Height < b.Height ;  
}  
}

Let us call the operator overloaded functions from the method below. When first if condition is triggered, first overloaded function in the rectangle class will be triggered. When second if condition is triggered, second overloaded function in the rectangle class will be triggered.

public static void Main()  
{  
Rectangle obj1 =new Rectangle();  
Rectangle obj2 =new Rectangle();  
if(obj1 > obj2){  
Console.WriteLine("Rectangle1 is greater than Rectangle2");  
}  
  
if(obj1 < obj2)  
{  
Console.WriteLine("Rectangle1 is less than Rectangle2");  
}  
}

**) Define Data Encapsulation in c# ?**  
  
Data Encapsulation is defined as the process of hiding the important fields from the end user. In the above example , we had used getters and setters to set value for MinSalary. Idea behind this is that , private field “minimumSalary” is an important part of our classes. So if we give a third party code to have complete control over the field without any validation it can adversely affect the functionality. This is in line with OOPS Concept that an external user should know about the what an object does. How it does it should be decided by the program.  
  
So if a user set a negative value for MinSalary , we can put a validation in set method to avoid negative values as shown below  
  
set  
{  
if(value > 0)  
{  
minSalary = value;  
}  
}  
  
**12) Define Inheritance in C#?**  
  
In object-oriented programming (OOP), inheritance is a way to reuse code of existing objects. In inheritance there will be two classes - base class and derived classes . A class can inherit attributes and methods from existing class called base class or parent class. The class which inherits from base class is called derived classes or child class. For more clarity on this topic , let us have a look at 2 classes shown below. Here Class Car is Base Class and Class Ford is derived class.  
  
class Car  
{  
public Car()  
{  
Console.WriteLine("Base Class Car");  
}  
  
public void DriveType()  
{  
Console.WriteLine("Right Hand Drive");  
}  
}  
  
class Ford : Car  
{  
public Ford()  
{  
Console.WriteLine("Derived Class Ford");  
}  
  
public void Price()  
{  
Console.WriteLine("Ford Price : 100K $");  
}  
}  
  
When we execute following lines of code ,  
  
Ford CarFord = new Ford();  
CarFord.DriveType();  
CarFord.Price();  
  
Output Generated is given below.  
  
Base Class Car  
Derived Class Ford  
Right Hand Drive  
Ford Price : 100K $  
  
What this means is that , all the methods and attributes of Base Class car is available in Derived Class Ford. When an object of class Ford is created , constructors of the Base and Derived class get invoked. Even though there is no method called DriveType() in Class Ford, we are able to invoke the method because of inheriting Base Class methods to derived class.   
  
**13) Define Multiple Inheritance in C# ?**  
  
In C# , derived classes can inherit from only one base class. If you want inherit multiple base classes, use interface.  
  
**14) What is Polymorphism in C#?**  
  
Ability of a programming language to process objects in different ways depending on their data type or class is known as Polymorphism.  
  
There are two types of polymorphism  
1) Compile time polymorphism. Best example is Overloading  
2) Runtime polymorphism. Best example is Overriding  
  
**15) Define Virtual Keyword in C#?**  
  
When we want to give permission to derived class to override a method in base class, Virtual keyword is used . For example lets us look at the classes Car and Ford as shown below.  
  
class Car  
{  
public Car()  
{  
Console.WriteLine("Base Class Car");  
}  
  
public virtual void DriveType()  
{  
Console.WriteLine("Right Hand Drive");  
}  
}  
  
class Ford : Car  
{  
public Ford()  
{  
Console.WriteLine("Derived Class Ford");  
}  
  
public void Price()  
{  
Console.WriteLine("Ford Price : 100K $");  
}  
  
public override void DriveType()  
{  
Console.WriteLine("Right Hand ");  
}  
}  
  
When following lines of code get executed   
  
Car CarFord = new Car();  
CarFord.DriveType();  
CarFord = new Ford();  
CarFord.DriveType();  
  
Output is as given below.  
  
Base Class Car  
Right Hand Drive  
Base Class Car  
Derived Class Ford  
Right Hand

**) Define overriding in c# ?**  
  
To overide a base class method which is defined as virtual , Overide keyword is used. In the above example , method DriveType is overrided in derived class.  
  
**17) Define Method Hiding in C# ?**  
  
If the derived class doesn't want to use methods in base class , derived class can implement the same method in derived class with same signature. For example in the classes given below, DriveType() is implemented in the derived class with same signature. This is called Method Hiding.  
  
class Car  
{  
public void DriveType()  
{  
Console.WriteLine("Right Hand Drive");  
}  
}  
  
class Ford : Car  
{  
public void DriveType()  
{  
Console.WriteLine("Right Hand ");  
}  
}  
  
**18) Define Abstract Class in C#?**  
  
If we don't want a class object to be created define the class as abstract. An abstract class can have abstract and non abstract classes. If a method in abstract id defined as abstract , it must be implemented in derived class. For example , in the classes given below , method DriveType is defined as abstract.   
  
abstract class Car  
{  
public Car()  
{  
Console.WriteLine("Base Class Car");  
}  
  
public abstract void DriveType();  
}  
  
class Ford : Car  
{  
public void DriveType()  
{  
Console.WriteLine("Right Hand ");  
}  
}  
  
Method DriveType get implemented in derived class.  
  
**19) Define Sealed Classes in c# ?**  
  
If a class is defined as Sealed it cannot be inherited in derived class. Example is given below.  
  
public sealed class Car  
{  
public Car()  
{  
Console.WriteLine("Base Class Car");  
}  
  
public void DriveType()  
{  
Console.WriteLine("Right Hand ");  
}  
}  
  
**20) Define Interfaces in C# ?**  
  
An interface is similar to a class with method signatures. There wont be any implementation of the methods in Interface. Classes which implements interface should have implementation of methods defined in abstract class.  
  
interface Breaks  
{  
void BreakType();  
}  
  
interface Wheels  
{  
void WheelType();  
}   
  
class Ford : Breaks, Wheels  
{  
public Ford()  
{  
Console.WriteLine("Derived Class Ford");  
}  
  
public void Price()  
{  
Console.WriteLine("Ford Price : 100K $");  
}  
  
public void BreakType()  
{  
Console.WriteLine("Power Break");  
}  
  
public void WheelType()  
{  
Console.WriteLine("Bridgestone");  
}  
}  
  
  
**21) Define Constructor in C# ?**  
  
Constructor is a special method that get invoked / called automatically whenever an object of a given class gets instantiated.  
  
For example in our class car constructor is defined as shown below  
  
public Car()  
{  
Console.WriteLine("Base Class Car");  
}  
  
When ever an instance of class car is created from the same class or its derived class(Except Few Scenarios) Constructor get called and sequence of code written in the constructor get executed.  
  
**22) Define Destructor in C# ?**  
  
Destructor is a special method that get invoked / called automatically whenever an object of a given class gets destroyed. Main idea behind using destructor is to free the memory used by the object.

|  |  |
| --- | --- |
| 1. | Which of the following statements is correct about the C#.NET code snippet given below?  class Student s1, s2; // Here 'Student' is a user-defined class.  s1 = new Student();  s2 = new Student(); |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | Contents of *s1* and *s2* will be exactly same. | | [B.](javascript:%20void%200;) | The two objects will get created on the stack. | | [C.](javascript:%20void%200;) | Contents of the two objects created will be exactly same. | | [D.](javascript:%20void%200;) | The two objects will always be created in adjacent memory locations. | | [E.](javascript:%20void%200;) | We should use *delete()* to delete the two objects from memory. |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **C**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-102).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-102) |

|  |  |
| --- | --- |
| 2. | Which of the following statements is correct about the C#.NET code snippet given below?  class Sample  {  private int i;  public Single j;  private void DisplayData()  {  Console.WriteLine(i + " " + j);  }  public void ShowData()  {  Console.WriteLine(i + " " + j);  }  } |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | *j* cannot be declared as *public*. | | [B.](javascript:%20void%200;) | *DisplayData()* cannot be declared as *private*. | | [C.](javascript:%20void%200;) | *DisplayData()* cannot access *j*. | | [D.](javascript:%20void%200;) | *ShowData()* cannot access to *i*. | | [E.](javascript:%20void%200;) | There is no error in this class. |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **E**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-105).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-105) |

|  |  |
| --- | --- |
| 3. | Which of the following statements are correct?   1. Instance members of a *class* can be accessed only through an object of that *class*. 2. A *class* can contain only instance data and instance member *function*. 3. All objects created from a *class* will occupy equal number of bytes in memory. 4. A *class* can contain Friend functions. 5. A *class* is a blueprint or a template according to which objects are created. |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | 1, 3, 5 | | [B.](javascript:%20void%200;) | 2, 4 | | [C.](javascript:%20void%200;) | 3, 5 | | [D.](javascript:%20void%200;) | 2, 4, 5 | | [E.](javascript:%20void%200;) | None of these |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **A**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-104).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-104) |

|  |  |
| --- | --- |
| 4. | Which of the following statements is correct? |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | Procedural Programming paradigm is different than structured programming paradigm. | | [B.](javascript:%20void%200;) | Object Oriented Programming paradigm stresses on dividing the logic into smaller parts and writing procedures for each part. | | [C.](javascript:%20void%200;) | Classes and objects are corner stones of structured programming paradigm. | | [D.](javascript:%20void%200;) | Object Oriented Programming paradigm gives equal importance to data and the procedures that work on the data. | | [E.](javascript:%20void%200;) | C#.NET is a structured programming language. |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **D**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-101).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-101) |

|  |  |
| --- | --- |
| 5. | Which of the following is the correct way to create an object of the *class Sample*?   1. *Sample s = new Sample();* 2. *Sample s;* 3. *Sample s; s = new Sample();* 4. *s = new Sample();* |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | 1, 3 | | [B.](javascript:%20void%200;) | 2, 4 | | [C.](javascript:%20void%200;) | 1, 2, 3 | | [D.](javascript:%20void%200;) | 4, 5 | | [E.](javascript:%20void%200;) | None of these |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **A**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-108)   |  | | --- | | Which of the following will be the correct output for the C#.NET program given below?  namespace IndiabixConsoleApplication  {  class Sample  {  int i;  Single j;  public void SetData(int i, Single j)  {  i = i;  j = j;  }  public void Display()  {  Console.WriteLine(i + " " + j);  }  }  class MyProgram  {  static void Main(string[ ] args)  {  Sample s1 = new Sample();  s1.SetData(10, 5.4f);  s1.Display();  }  }  } | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | 0 0 | | [B.](javascript:%20void%200;) | 10 5.4 | | [C.](javascript:%20void%200;) | 10 5.400000 | | [D.](javascript:%20void%200;) | 10 5 | | [E.](javascript:%20void%200;) | None of the above |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **A**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-110).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-110) |  |  |  | | --- | --- | | 7. | The *this* reference gets created when a member function (non-shared) of a class is called. | | |  |  |  |  | | --- | --- | --- | --- | | [A.](javascript:%20void%200;) | True | [B.](javascript:%20void%200;) | False |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **A**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-115).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-115) |  |  |  | | --- | --- | | 8. | Which of the following statements are correct?   1. Data members ofa class are by default *public*. 2. Data members of a class are by default *private*. 3. Member functions ofa class are by default *public*. 4. A private function of a class can access a *public* function within the same class. 5. Member function of a class are by default *private*. | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | 1, 3, 5 | | [B.](javascript:%20void%200;) | 1, 4 | | [C.](javascript:%20void%200;) | 2, 4, 5 | | [D.](javascript:%20void%200;) | 1, 2, 3 | | [E.](javascript:%20void%200;) | None of these |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **C**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-109).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-109) |  |  |  | | --- | --- | | 9. | Which of the following statements is correct about the C#.NET code snippet given below?  namespace IndiabixConsoleApplication  {  class Sample  {  public int index;  public int[] arr = new int[10];    public void fun(int i, int val)  {  arr[i] = val;  }  }    class MyProgram  {  static void Main(string[] args)  {  Sample s = new Sample();  s.index = 20;  Sample.fun(1, 5);  s.fun(1, 5);  }  }  } | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | *s.index = 20* will report an error since index is *public*. | | [B.](javascript:%20void%200;) | The call *s.fun(1, 5)* will work correctly. | | [C.](javascript:%20void%200;) | *Sample.fun(1, 5)* will set a value *5* in *arr[ 1 ]*. | | [D.](javascript:%20void%200;) | The call *Sample.fun(1, 5)* cannot work since *fun()* is not a shared function. | | [E.](javascript:%20void%200;) | *arr* being a data member, we cannot declare it as *public*. |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **B**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-107).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-107) |  |  |  | | --- | --- | | 10. | Which of the following statements are correct about the C#.NET code snippet given below?  sample c;  c = new sample();   1. It will create an object called sample. 2. It will create a nameless object of the type sample. 3. It will create an object of the type sample on the stack. 4. It will create a reference *c* on the stack and an object of the type sample on the heap. 5. It will create an object of the type sample either on the heap or on the stack depending on the size of the object. | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | 1, 3 | | [B.](javascript:%20void%200;) | 2, 4 | | [C.](javascript:%20void%200;) | 3, 5 | | [D.](javascript:%20void%200;) | 4, 5 | | [E.](javascript:%20void%200;) | None of these |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **B**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-103). |  |  |  | | --- | --- | |  | Which of the following statements is correct about the C#.NET code snippet given below?  int i;  int j = new int();  i = 10;  j = 20;  String str;  str = i.ToString();  str = j.ToString(); | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | This is a perfectly workable code snippet. | | [B.](javascript:%20void%200;) | Since *int* is a primitive, we cannot use *new* with it. | | [C.](javascript:%20void%200;) | Since an *int* is a primitive, we cannot call the method *ToString()* using it. | | [D.](javascript:%20void%200;) | *i* will get created on stack, whereas *j* will get created on heap. | | [E.](javascript:%20void%200;) | Both *i* and *j* will get created on heap. |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **A**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-100).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-100) |  |  |  | | --- | --- | | 12. | Which of the following statements are correct about the *this* reference?   1. *this* reference can be modified in the instance member function of a class. 2. Static functions of a class never receive the *this* reference. 3. Instance member functions of a class always receive a *this* reference. 4. *this* reference continues to exist even after control returns from an instance member function. 5. While calling an instance member function we are not required to pass the *this* reference explicitly. | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | 1, 4 | | [B.](javascript:%20void%200;) | 2, 3, 5 | | [C.](javascript:%20void%200;) | 3, 4 | | [D.](javascript:%20void%200;) | 2, 5 | | [E.](javascript:%20void%200;) | None of these |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **B**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-114).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-114) |  |  |  | | --- | --- | | 13. | Which of the following will be the correct output for the C#.NET program given below?  namespace IndiabixConsoleApplication  {  class Sample  {  int i;  Single j;  public void SetData(int i, Single j)  {  this.i = i;  this.j = j;  }  public void Display()  {  Console.WriteLine(i + " " + j);  }  }  class MyProgram  {  static void Main(string[ ] args)  {  Sample s1 = new Sample();  s1.SetData(36, 5.4f);  s1.Display();  }  }  } | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | 0 0.0 | | [B.](javascript:%20void%200;) | 36 5.4 | | [C.](javascript:%20void%200;) | 36 5.400000 | | [D.](javascript:%20void%200;) | 36 5 | | [E.](javascript:%20void%200;) | None of the above |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **B**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-111).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-111) |  |  |  | | --- | --- | | 14. | Which of the following statements are correct about objects of a user-defined class called *Sample*?   1. All objects of *Sample* class will always have exactly same data. 2. Objects of *Sample* class may have same or different data. 3. Whether objects of *Sample* class will have same or different data depends upon a Project Setting made in Visual Studio.NET. 4. Conceptually, each object of *Sample* class will have instance data and instance member functions of the *Sample* class. 5. All objects of *Sample* class will share one copy of member functions. | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | 1, 3 | | [B.](javascript:%20void%200;) | 2, 4 | | [C.](javascript:%20void%200;) | 4, 5 | | [D.](javascript:%20void%200;) | 3, 5 | | [E.](javascript:%20void%200;) | None of these |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **C**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-113).  [Workspace](javascript:%20void%200;)    [Report Errors](javascript:%20void%200;)    [View Answer](javascript:%20void%200;) [Workspace](javascript:%20void%200;) [Report](javascript:%20void%200;) [Discuss in Forum](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-113) |  |  |  | | --- | --- | | 15. | Which of the following statements are correct about the C#.NET code snippet given below?  namespace IndiabixConsoleApplication  {  class Sample  {  int i, j;  public void SetData(int ii, int jj)  {  this.i = ii;  this.j = jj  }  }  class MyProgram  {  static void Main(string[ ] args)  {  Sample s1 = new Sample();  s1.SetData(10, 2);  Sample s2 = new Sample();  s2.SetData(5, 10);  }  }  } | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | The code will not compile since we cannot explicitly use *this*. | | [B.](javascript:%20void%200;) | Using *this* in this program is necessary to properly set the values in the object. | | [C.](javascript:%20void%200;) | The call to *SetData()* is wrong since we have not explicitly passed the *this* reference to it. | | [D.](javascript:%20void%200;) | The definition of *SetData()* is wrong since we have not explicitly collected the *this* reference. | | [E.](javascript:%20void%200;) | Contents of *this* will be different during each call to *SetData()*. |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **E**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-112). | | Which of the following statements is correct about classes and objects in C#.NET? | | | |  |  | | --- | --- | | [A.](javascript:%20void%200;) | Class is a value type. | | [B.](javascript:%20void%200;) | Since objects are typically big in size, they are created on the stack. | | [C.](javascript:%20void%200;) | Objects of smaller size are created on the heap. | | [D.](javascript:%20void%200;) | Smaller objects that get created on the stack can be given names. | | [E.](javascript:%20void%200;) | Objects are always nameless. |     [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **E**  **Explanation:**  No answer description available for this question. [**Let us discuss**](http://www.indiabix.com/c-sharp-programming/classes-and-objects/discussion-106). | |   Class:   It is a collection of objects.  Object:  It is a real time entity.  An object can be considered a "thing" that can perform a set of related activities. The set of activities that the object performs defines the object's behavior. For example, the hand can grip something or a Student (object) can give the name or address. In pure OOP terms an object is an instance of a class    The above template describe about object Student  Class is composed of three things name, attributes, and operations  public class student  {  }  student objstudent=new student ();  According to the above sample we can say that Student object, named objstudent, has created out of the student class.  In real world you will often find many individual objects all of the same kind. As an example, there may be thousands of other bicycles in existence, all of the same make and model. Each bicycle has built from the same blueprint. In object-oriented terms, we say that the bicycle is an instance of the class of objects known as bicycles. In the software world, though you may not have realized it, you have already used classes. For example, the Textbox control, you always used, is made out of the Textbox class, which defines its appearance and capabilities. Each time you drag a Textbox control, you are actually creating a new instance of the Textbox class.  Encapsulation:  Encapsulation is a process of binding the data members and member functions into a single unit.  Example for encapsulation is class. A class can contain data structures and methods.  Consider the following class  public class Aperture  {  public Aperture ()  {  }  protected double height;  protected double width;  protected double thickness;  public double get volume()  {  Double volume=height \* width \* thickness;  if (volume<0)  return 0;  return volume;  }  }  In this example we encapsulate some data such as height, width, thickness and method Get Volume. Other methods or objects can interact with this object through methods that have public access modifier  Abstraction:  Abstraction is a process of hiding the implementation details and displaying the essential features.  Example1: A Laptop consists of many things such as processor, motherboard, RAM, keyboard, LCD screen, wireless antenna, web camera, usb ports, battery, speakers etc. To use it, you don't need to know how internally LCD screens, keyboard, web camera, battery, wireless antenna, speaker’s works.  You just need to know how to operate the laptop by switching it on. Think about if you would have to call to the engineer who knows all internal details of the laptop before operating it. This would have highly expensive as well as not easy to use everywhere by everyone.  So here the Laptop is an object that is designed to hide its complexity.  How to abstract: - By using Access Specifiers  .Net has five access Specifiers  Public -- Accessible outside the class through object reference.  Private -- Accessible inside the class only through member functions.  Protected -- Just like private but Accessible in derived classes also through member  functions.  Internal -- Visible inside the assembly. Accessible through objects.  Protected Internal -- Visible inside the assembly through objects and in derived classes outside the assembly through member functions.  Let’s try to understand by a practical example:-  public class Class1      {          int  i;                                         //No Access specifier means private          public  int j;                                        // Public          protected int k;                             //Protected data          internal int m;                        // Internal means visible inside assembly          protected internal int n;                   //inside assembly as well as to derived classes outside assembly          static int x;                                 // This is also private          public static int y;                       //Static means shared across objects          [DllImport("MyDll.dll")]          public static extern int MyFoo();       //extern means declared in this assembly defined in some other assembly          public void myFoo2()          {              //Within a class if you create an object of same class then you can access all data members through object reference even private data too              Class1 obj = new Class1();              obj.i =10;   //Error can’t access private data through object.But here it is accessible.:)              obj.j =10;              obj.k=10;              obj.m=10;              obj.n=10;         //     obj.s =10;  //Errror Static data can be accessed by class names only              Class1.x = 10;           //   obj.y = 10; //Errror Static data can be accessed by class names only              Class1.y = 10;          }      }  Now lets try to copy the same code inside Main method and try to compile  [STAThread]          static void Main()          {             //Access specifiers comes into picture only when you create object of class outside the class              Class1 obj = new Class1();         //     obj.i =10; //Error can’t access private data through object.              obj.j =10;        //      obj.k=10;     //Error can’t access protected data through object.              obj.m=10;              obj.n=10;         //     obj.s =10;  //Errror Static data can be accessed by class names only              Class1.x = 10;  //Error can’t access private data outside class           //   obj.y = 10; //Errror Static data can be accessed by class names only              Class1.y = 10;          }  What if Main is inside another assembly  [STAThread]          static void Main()          {             //Access specifiers comes into picture only when you create object of class outside the class              Class1 obj = new Class1();         //     obj.i =10; //Error can’t access private data through object.              obj.j =10;        //      obj.k=10;     //Error can’t access protected data through object.       //     obj.m=10; // Error can’t access internal data outside assembly      //      obj.n=10; // Error can’t access internal data outside assembly         //     obj.s =10;  //Errror Static data can be accessed by class names only              Class1.x = 10;  //Error can’t access private data outside class           //   obj.y = 10; //Errror Static data can be accessed by class names only              Class1.y = 10;          }  In object-oriented software, complexity is managed by using abstraction.  Abstraction is a process that involves identifying the critical behavior of an object and eliminating irrelevant and complex details.  Inheritance:  Inheritance is a process of deriving the new class from already existing class  C# is a complete object oriented programming language. Inheritance is one of the primary concepts of object-oriented programming. It allows you to reuse existing code. Through effective use of inheritance, you can save lot of time in your programming and also reduce errors, which in turn will increase the quality of work and productivity. A simple example to understand inheritance in C#.  Using System;  Public class BaseClass  {      Public BaseClass ()      {          Console.WriteLine ("Base Class Constructor executed");      }        Public void Write ()      {          Console.WriteLine ("Write method in Base Class executed");      }  }    Public class ChildClass: BaseClass  {        Public ChildClass ()      {          Console.WriteLine("Child Class Constructor executed");      }        Public static void Main ()      {          ChildClass CC = new ChildClass ();          CC.Write ();      }  }  In the Main () method in ChildClass we create an instance of childclass. Then we call the write () method. If you observe the ChildClass does not have a write() method in it. This write () method has been inherited from the parent BaseClass.  The output of the above program is   Output:    Base Class Constructor executed   Child Class Constructor executed   Write method in Base Class executed  this output proves that when we create an instance of a child class, the base class constructor will automatically be called before the child class constructor. So in general Base classes are automatically instantiated before derived classes.  In C# the syntax for specifying BaseClass and ChildClass relationship is shown below. The base class is specified by adding a colon, ":", after the derived class identifier and then specifying the base class name.  Syntax:  class ChildClassName: BaseClass               {                    //Body               }  C# supports single class inheritance only. What this means is, your class can inherit from only one base class at a time. In the code snippet below, class C is trying to inherit from Class A and B at the same time. This is not allowed in C#. This will lead to a compile time  error: Class 'C' cannot have multiple base classes: 'A' and 'B'.  public class A  {  }  public class B  {  }  public class C : A, B  {  }  In C# Multi-Level inheritance is possible. Code snippet below demonstrates mlti-level inheritance. Class B is derived from Class A. Class C is derived from Class B. So class C, will have access to all members present in both Class A and Class B. As a result of multi-level inheritance Class has access to A\_Method(),B\_Method() and C\_Method().   Note: Classes can inherit from multiple interfaces at the same time. Interview Question: How can you implement multiple inheritance in C#? Ans : Using Interfaces. We will talk about interfaces in our later article.  Using System;  Public class A  {      Public void A\_Method ()      {          Console.WriteLine ("Class A Method Called");      }  }  Public class B: A  {      Public void B\_Method ()      {          Console.WriteLine ("Class A Method Called");      }  }  Public class C: B  {      Public void C\_Method ()      {          Console.WriteLine ("Class A Method Called");      }        Public static void Main ()      {          C C1 = new C ();          C1.A\_Method ();          C1.B\_Method ();          C1.C\_Method ();      }  }  When you derive a class from a base class, the derived class will inherit all members of the base class except constructors. In the code snippet below class B will inherit both M1 and M2 from Class A, but you cannot access M2 because of the private access modifier. Class members declared with a private access modifier can be accessed only with in the class. We will talk about access modifiers in our later article.   Common Interview Question: Are private class members inherited to the derived class?  Ans: Yes, the private members are also inherited in the derived class but we will not be able to access them. Trying to access a private base class member in the derived class will report a compile time error.  Using System;  Public class A  {  Public void M1 ()  {  }  Private void M2 ()  {  }  }  Public class B: A  {  Public static void Main ()  {  B B1 = new B ();  B1.M1 ();  //Error, Cannot access private member M2  //B1.M2 ();  }  }  Method Hiding and Inheritance We will look at an example of how to hide a method in C#. The Parent class has a write () method which is available to the child class. In the child class I have created a new write () method. So, now if I create an instance of child class and call the write () method, the child class write () method will be called. The child class is hiding the base class write () method. This is called method hiding.   If we want to call the parent class write () method, we would have to type cast the child object to Parent type and then call the write () method as shown in the code snippet below.  Using System;  Public class Parent  {      Public void Write ()      {          Console.WriteLine ("Parent Class write method");      }  }    Public class Child: Parent  {      Public new void Write ()      {          Console.WriteLine ("Child Class write method");      }        Public static void Main ()      {          Child C1 = new Child ();          C1.Write ();          //Type caste C1 to be of type Parent and call Write () method          ((Parent) C1).Write ();      }  }  Polymorphism:  When a message can be processed in different ways is called polymorphism. Polymorphism means many forms.    Polymorphism is one of the fundamental concepts of OOP.    Polymorphism provides following features:  It allows you to invoke methods of derived class through base class reference during runtime.  It has the ability for classes to provide different implementations of methods that are called through the same name.  Polymorphism is of two types:  Compile time polymorphism/Overloading  Runtime polymorphism/Overriding  Compile Time Polymorphism    Compile time polymorphism is method and operators overloading. It is also called early binding.    In method overloading method performs the different task at the different input parameters.    Runtime Time Polymorphism    Runtime time polymorphism is done using inheritance and virtual functions. Method overriding is called runtime polymorphism. It is also called late binding.    When overriding a method, you change the behavior of the method for the derived class.  Overloading a method simply involves having another method with the same prototype.    Caution: Don't confused method overloading with method overriding, they are different, unrelated concepts. But they sound similar.    Method overloading has nothing to do with inheritance or virtual methods.    Following are examples of methods having different overloads:    void area(int side);  void area(int l, int b);  void area(float radius);    Practical example of Method Overloading (Compile Time Polymorphism)    using System;    namespace method\_overloading  {      class Program      {          public class Print          {                public void display(string name)              {                  Console.WriteLine ("Your name is : " + name);              }                public void display(int age, float marks)              {                  Console.WriteLine ("Your age is : " + age);                  Console.WriteLine ("Your marks are :" + marks);              }          }            static void Main(string[] args)          {                Print obj = new Print ();              obj.display ("George");              obj.display (34, 76.50f);              Console.ReadLine ();          }      }  }  Note: In the code if you observe display method is called two times. Display method will work according to the number of parameters and type of parameters.  When and why to use method overloading    Use method overloading in situation where you want a class to be able to do something, but there is more than one possibility for what information is supplied to the method that carries out the task.    You should consider overloading a method when you for some reason need a couple of methods that take different parameters, but conceptually do the same thing.    Method overloading showing many forms.    using System;    namespace method\_overloading\_polymorphism  {      Class Program      {          Public class Shape          {              Public void Area (float r)              {                  float a = (float)3.14 \* r;                  // here we have used function overload with 1 parameter.                  Console.WriteLine ("Area of a circle: {0}",a);              }                Public void Area(float l, float b)              {                  float x = (float)l\* b;                  // here we have used function overload with 2 parameters.                  Console.WriteLine ("Area of a rectangle: {0}",x);                }                public void Area(float a, float b, float c)              {                  float s = (float)(a\*b\*c)/2;                  // here we have used function overload with 3 parameters.                  Console.WriteLine ("Area of a circle: {0}", s);              }          }            Static void Main (string[] args)          {              Shape ob = new Shape ();              ob.Area(2.0f);              ob.Area(20.0f,30.0f);              ob.Area(2.0f,3.0f,4.0f);              Console.ReadLine ();          }      }  }    Things to keep in mind while method overloading    If you use overload for method, there are couple of restrictions that the compiler imposes.    The rule is that overloads must be different in their signature, which means the name and the number and type of parameters.    There is no limit to how many overload of a method you can have. You simply declare them in a class, just as if they were different methods that happened to have the same name.  Method Overriding:  Whereas Overriding means changing the functionality of a method without changing the signature. We can override a function in base class by creating a similar function in derived class. This is done by using virtual/override keywords.  Base class method has to be marked with virtual keyword and we can override it in derived class using override keyword.  Derived class method will completely overrides base class method i.e. when we refer base class object created by casting derived class object a method in derived class will be called.  Example:  // Base class public class BaseClass { public virtual void Method1() { Console.Write("Base Class Method"); } } // Derived class public class DerivedClass : BaseClass { public override void Method1() { Console.Write("Derived Class Method"); } } // Using base and derived class public class Sample { public void TestMethod() { // calling the overriden method DerivedClass objDC = new DerivedClass();  objDC.Method1();  // calling the baesd class method BaseClass objBC = (BaseClass)objDC;  objDC.Method1(); } }  Output ---------------------  Derived Class Method  Derived Class Method  Constructors and Destructors:  Classes have complicated internal structures, including data and functions, object initialization and cleanup for classes is much more complicated than it is for simple data structures. Constructors and destructors are special member functions of classes that are used to construct and destroy class objects. Construction may involve memory allocation and initialization for objects. Destruction may involve cleanup and deallocation of memory for objects.  Constructors and destructors do not have return types nor can they return values.  References and pointers cannot be used on constructors and destructors because their addresses cannot be taken.  Constructors cannot be declared with the keyword virtual.  Constructors and destructors cannot be declared const, or volatile.  Unions cannot contain class objects that have constructors or destructors.  Constructors and destructors obey the same access rules as member functions. For example, if you declare a constructor with protected access, only derived classes and friends can use it to create class objects.  The compiler automatically calls constructors when defining class objects and calls destructors when class objects go out of scope. A constructor does not allocate memory for the class object it’s this pointer refers to, but may allocate storage for more objects than its class object refers to. If memory allocation is required for objects, constructors can explicitly call the new operator. During cleanup, a destructor may release objects allocated by the corresponding constructor. To release objects, use the delete operator.  Example of Constructor  class C  {         private int x;         private int y;         public C (int i, int j)         {                   x = i;                   y = j;         }         public void display ()         {                 Console.WriteLine(x + "i+" + y);         }  }  Example of Destructor  class D  {          public D ()          {              // constructor          }          ~D ()          {             // Destructor          }  } |