USN	1	P	E			M	C	A		

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

INTERNAL ASSESSMENT TEST I

Date: 19/08/19	**Max Marks:** 40
Subject & Code: Programming Using C#.NET (17MCA52)	**Section:** V MCA
Name of Faculty: Mrs. Afshan Rehman	**Time:** 11:30-1:00 PM

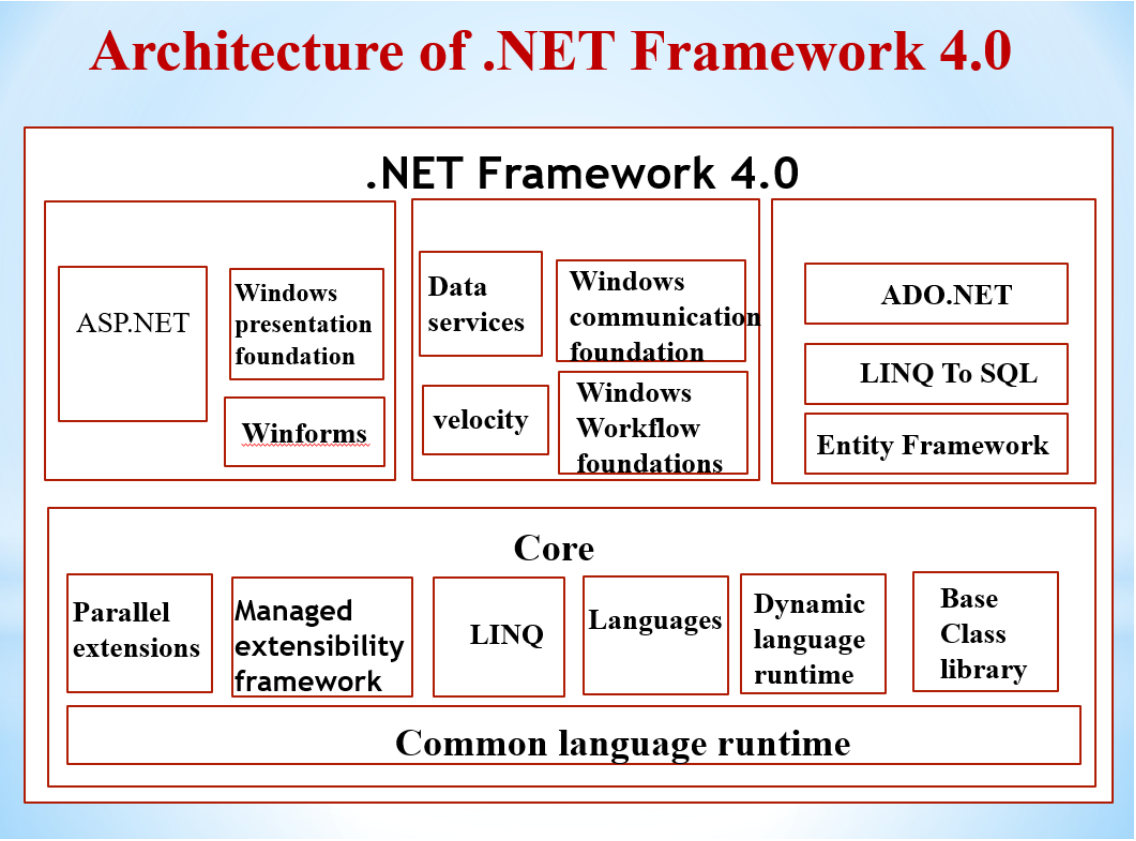
Note: ***Answer FIVE full questions. Select one question from each part.***

	Part I	
Q1	Explain the components of .Net framework with the help of architecture diagram.	**8**
	OR	
Q2	a) What is an assembly? Explain each component of an assembly. b) What are the different uses of ADO.Net	**5** **3**
	Part II	
Q3	a) What is boxing and unboxing? Write a C# console application to demonstrate boxing and unboxing. b) Write a program in C# to read a sum of Jagged array and display the sum of all the elements of inner array.	**4** **4**
	OR	
Q4	Explain the different data types in C# with examples for each.	**8**
	Part III	
Q5	a) Explain partial classes and partial methods with examples for each. b) Write a code snippet to demonstrate Nested class.	**6** **2**
	OR	
Q6	Explain the different C# properties with examples for each.	**8**
	Part IV	
Q7	a) What are indexers? Write a program to access an array of objects. b) What are the different ways of accessing methods as class members. Write a program to illustrate "passing by output".	**4** **4**
	OR	
Q8	What is encapsulation? Write a C# program to explain accessor and mutator properties used in encapsulation.	**8**
	Part V	
Q9	What is Polymorphism? With an example explain Operation Overloading and Function Overloading.	**8**
	OR	
Q10	Explain the different types of inheritance in C# with examples for each.	**8**

USN

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

1) **Explain the components of .Net framework with the help of architecture diagram.**



Components of .NET Framework 4.0

Common language runtime (CLR):- It provides a runtime environment to run the code and provides various other services.

Services provided by CLR:-

1) Memory management
2) Exception handling
3) Debugging
4) Security

5) Thread execution
6) Code execution
7) Code safety
8) Verification and compilation

The code that works on CLR is managed code

Managed code:- code that is directly executed by the CLR.

The CLR compiles the code to **intermediate language(IL)** not the machine code.(you can select the compiler according to programming language of your code)

This IL along with metadata resides in an assembly.

The IL is compiled to native code

Native code is finally executed

Common type system (CTS)

- It provides certain guidelines for declaring, using and managing types at runtime.
- It is integral part of CLR for supporting cross language application development
- It supports object oriented model for implementation.
- CTS supports two data types:- value type and reference type.

Assemblies

Assemblies are packaged into units containing programs and libraries that can be either dynamic link library(DLL) file or executable(EXE) file

A single assembly can have more than one code file and one code file can have more than one assembly.

Assemblies are of two types:-

1) Static assembly:- contains interface, class and resource and stored in PE file on memory.
2) Dynamic assembly:- run directly form memory without being stored on disk.

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

Assembly content:- It is a logical unit that contains four elements:-

1) Manifest
2) Metadata
3) IL code
4) Resources like bitmap, JPEG files etc

.NET Framework class library

The framework class library is a huge library of reusable types meant to be used by managed codes.

It is object oriented library used in component based applications

It is made up of hierarchy of namespaces separated by dots, that contains classes, structures, interfaces and enumerations.

Example System.data, System.Data.SqlClient etc

All types should have a base class (System.Object)

Windows forms

It is a graphical representation of any application.

In window form you can add controls to the forms and raise events which is handled by event handlers in the application

ASP.NET and ASP.NET AJAX

ASP.NET is a web development model used to develop interactive and data driven web application over the internet.

It consist of textboxes, buttons, label to create HTML pages.

1) Better performance (cached copy)
2) Improved security (custom business logic for authentication for forms) (checks the identity of user against the user account information stored on server for windows)

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

AJAX also known as atlas, is used as an extension for ASP.NET, it has both client and server side functionality to develop the web application.

1)Asynchronous

2) Minimal transfer of data

3)Minimal processing on the web server

ADO.NET

It is a technology used for working with data and databases of all types.

Data sources like Sql database, oracle database, OLE DB, XML can be connected.

Windows workflow foundation

It's a Microsoft Technology that Provides a programming model (API) for workflow based applications on Windows.

It includes workflow runtime, activities(WriteLine), workflow designers, and rules engine.

Windows Presentation Foundation

Formally named as **"Avalon".** It provides base for building applications and a clear separation between user interface and the business logic.

It helps in building interface that includes document, media, 2D & 3D graphics, animations.

Windows Communication Foundation

Formally known as "indigo". It is a service oriented technology for developing applications in a distributed system that provides services to the client applications.

It provides web services, remoting, distributed transactions, and message queuing in an enhanced and better way.

Windows CardSpace

Formally known as InfoCard. It is client software that improves the safety of accessing resources and helps in sharing personal information on the internet.

It helps in reducing the problems of traditional online security mechanisms by relying on separate desktop and cryptographically strong authentication.

LINQ

It is one of the component of .Net Framework 4.0 that adds native data querying capabilities to .Net by using syntex similar to SQL.

Example LINQ in C#

var query = from S in Students where s.Gender = "M" ;

Select s.name;

2 a) What is an assembly? Explain each component of an assembly.

Assemblies are packaged into units containing programs and libraries that can be either dynamic link library(DLL) file or executable(EXE) file

A single assembly can have more than one code file and one code file can have more than one assembly.

Assemblies are of two types:-

1) Static assembly:- contains interface, class and resource and stored in PE file on memory.
2) Dynamic assembly:- run directly form memory without being stored on disk.

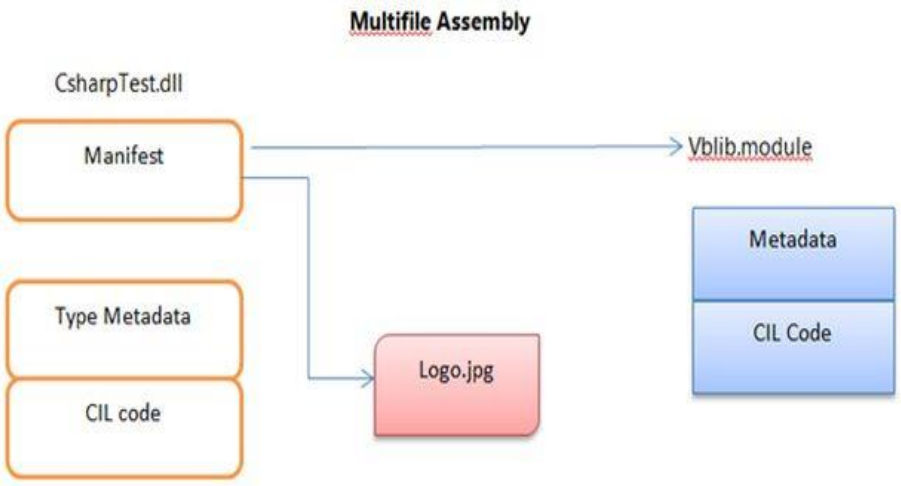
Assembly content:- It is a logical unit that contains four elements:-

1) Manifest
2) Metadata
3) IL code
4) Resources like bitmap, JPEG files etc

Assembly manifest:- It contains assembly metadata which contains assemblies version requirement and security identity. (Refer table 1.1 in book)

It can be stored in PE file with IL code or in a standalone PE file.

"Assemblies are smallest deployable units"



2 b) What are the different uses of ADO.Net

ADO.NET

It is a technology used for working with data and databases of all types.

Data sources like Sql database, oracle database, OLE DB, XML can be connected.

1) Disconnected data architecture
2) Cached data in datasets (dataset is cached copy of database independent of data source)
3) Scalability
4) Transfer of data in XML format
5) Interaction with the database through data commands.

USN	1	P	E			M	C	A		

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

PART – II

3a) What is boxing and unboxing? Write a C# console application to demonstrate boxing and unboxing.

When a value type is converted to object type, it is called **boxing** and on the other hand, when an object type is converted to a value type, it is called **unboxing**.

```
using System;
namespace Boxun {
   class Program {
      static void Main() {
         int i = 123;
         object o = i; // implicit boxing try
         {
         //int j = (short)o; // attempt to unbox but not possible int j =
         (int)o; // attempt to unbox
         System.Console.WriteLine("Unboxing OK.");
         Console.ReadLine(); } catch
         (System.InvalidCastException e) {
System.Console.WriteLine("{0} Error: Incorrectunboxing.", e.Message);
Console.ReadLine(); } } } }
```

b) Write a program in C# to read a sum of Jagged array and display the sum of all the elements of inner array.

In C#, jagged array is also known as "array of arrays" because its elements are arrays. The element size of jagged array can be different.

Declaration of Jagged array

```
int[][] arr = new int[2][];
```

```
using System;
using System.Collections.Generic;
using System.Linq; using System.Text;
using System.Threading.Tasks;
namespace Prog4 {
```


```
class Program {
static void Main(string[] args) {
int[][] myJag = new int[3][];
for(int i=0; i<myJag.Length;i++) {
myJag[i] = new int[i + 3];
  }
for(int i=0;i<3;i++) {
Console.WriteLine("Enter {1} Elements of row {0}", i, myJag[i].Length);
for(int j=0;j<myJag[i].Length;j++) {
myJag[i][j] = int.Parse(Console.ReadLine());
}
Console.WriteLine();
 }
int sum = 0;
for(int i=0;i<3;i++) {
for(int j=0;j<myJag[i].Length;j++)
{
sum += myJag[i][j]; } }
Console.WriteLine("Sum is\n" + sum);
Console.ReadLine(); } } }
```

4) Explain the different data types in C# with examples for each.

Effective use of data types will help use memory considerably

Different Data Types are: int, char, float

They are categorized into three simple types

a) Value type

b) Reference type

c) Pointer type

Value type are assigned value directly and reference type must be interfaced through method and functions

Value types are further divided into following categories:-

a) Struct type

b) Enum type

Struct type are special form of classes having properties of value types.

Value types are stored on the stack

It capsulate small group of related variables.

```
enstruct Books {
  public string title;
  public string author;
  public string subject;
  public int book_id;
};
```

- Enum: An enumeration is a set of named integer constants.
- An enumerated type is declared using the **enum** keyword.
- C# enumerations are value data type. In other words, enumeration contains its own values and cannot inherit or cannot pass inheritance.
- The general syntax for declaring an enumeration is −

enum <enum_name>

{

enumeration list

};

Ex: enum Days { Sun, Mon, tue, Wed, thu, Fri, Sat };

Reference Type refers to some object(memory location) whereas value type contains discrete entities.

- The reference types do not contain the actual data stored in a variable, but they contain a reference to the variables.

- In other words, they refer to a memory location. Using multiple variables, the reference types can refer to a memory location.
- If the data in the memory location is changed by one of the variables, the other variable automatically reflects this change in value.
- Example of **built-in** reference types are: **object**, **dynamic,** and **string**.

Object Type

The Object Type is the ultimate **base class** for all data types in C# Common Type System (CTS).

Object is an alias **for System.Object class**.

Ex: **int num = 23; // 23 will assigned to num**

object obj = num;

Dynamic types are similar to object types except that type checking for object type variables takes place at compile time, whereas that for the dynamic type variables takes place at run time.

Syntax for declaring a dynamic type is −

dynamic <variable_name> = value;

For example,

dynamic d = 20;

The **String Type** allows you to assign any string values to a variable.

String str = "Tutorials Point";

Pointer and reference type represent memory address but difference is that references are tracked by the garbage collector but pointers are not.

Ex: int* iptr;

PART – III

5 a) Explain partial classes and partial methods with examples for each.

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

Partial Classes

Each class in C# resides in a separate physical file with a .cs extension.

C# provides the ability to have a single class implementation in multiple .cs files using the partial modifier keyword.

The partial modifier can be applied to a class, method, interface or structure.

For example, the following MyPartialClass splits into two files, PartialClassFile1.cs and PartialClassFile2.cs:

Example: PartialClassFile1.cs

```
public partial class MyPartialClass
{
public MyPartialClass()
{ }
public void Method1(int val)
{
Console.WriteLine(val);
}
}
```

Example: PartialClassFile2.cs

```
public partial class MyPartialClass
{
public void Method2(int val)
{
Console.WriteLine(val);
```

} }

Example: Partial class

```
public class MyPartialClass
{
public MyPartialClass()
{ }

public void Method1(int val)
{
Console.WriteLine(val);
}
public void Method2(int val)
{
Console.WriteLine(val);
} }
```

Partial Method

A partial class or struct may contain partial methods.

A partial method must be declared in one of the partial classes.

A partial method may or may not have an implementation. If the partial method doesn't have an implementation in any part then the compiler will not generate that method in the final class.

Partial method must have a void return type. Partial methods are private by default.

b) Write a code snippet to demonstrate Nested class.

Nested classes are classes within classes. The nested class acts as the member of the parent class in which it is defined.

A nested class has the advantage of accessing all the members of its outer class.

```
class A
{
public int _v1;

public class B
        {
           public int _v2;
        }
}
class Program
{
static void Main() {
A a = new A();
a._v1++;
A.B ab = new A.B();
ab._v2++; } }
```

6) Explain the different C# properties with examples for each.

c# Properties

It is a member of class or structure that provides a flexible mechanism to read, write, or compute the value of a private field.

Why need properties?

If the members of a class are private then how another class in C# will be able to read, write, or compute the value that field.

If the members of the class are public then another class may misuse that member.

Properties provide the opportunity to protect a field in a class by reading and writing to it through the property. C# properties enable this type of protection while also letting you access the property just like it was a field.

Creating Read-Only Properties

Properties can be made read-only. This is accomplished by having only a *get* accessor in the property implementation.

Get Accessor: It specifies that the value of a field can access publicly. It returns a single value and it specifies the read-only property.

```
using System;

public class Student {

    // Declare counter field as cnt
    private static int cnt;

    // to define constructor
    public Student()
    {

        // increment the counter
        // using constructor
        cnt++;
    }

    // Declare counter property
    public static int Counter
```

```
    {

        // read-only property
        get
        {
            return cnt;
        }
    }
}

class StudentTest {

    // Main Method
    public static void Main(string[] args)
    {


        // create three instances of
        // Student class it call constructor
        // three times which increase the counter
        Student s1 = new Student();
        Student s2 = new Student();
        Student s3 = new Student();

        // s1.Counter = 10;
        // Compile Time Error: Can't set value of
        // Counter because it is read only.

        Console.WriteLine("Total No of Student: " + Student.Counter);

        // Program Give Warning
        // The variable `s1' is assigned but its value is never used
    }
}
```

Creating a Write-Only Property

You can assign values to, but not read from, a write-only property. A write-only property only has a *set* accessor.

Set Accessor: It will specify the assignment of a value to a private field in a property. It returns a single value and it specifies the write-only property.

Example:

```
using System;

public class Student {

    // Declare name field
    private string name = "GeeksforGeeks";

    // Declare name property
    public string Name
    {

       get
        {
            return name;
        }

        set
        {
            name = value;
        }


 }
}

class TestStudent {

    // Main Method
    public static void Main(string[] args)
    {
        Student s = new Student();

        // calls set accessor of the property Name,
        // and pass "GFG" as value of the
        // standard field 'value'.
        s.Name = "GFG";

        // displays GFG, Calls the get accessor
        // of the property Name.
```

```
        Console.WriteLine("Name: " + s.Name);
    }
}
```

Static properties

You can declare a static property with a static keyword followed by the property name.

A static property can be used to access only the static members of the class.

USN	1	P	E			M	C	A		

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

PART – IV

7 a) What are indexers? Write a program to access an array of objects.

Indexers

Indexers allow our class to be used just like an array. Or we can say we can index the object using [ ] brackets just like arrays.

Indexers are special type of properties which add functionality to a class to be indexed.

Any valid access specifiers can be used and similarly any return type.Indexers are created using this keyword.

```
using System;

// class declaration

class IndexerCreation

{

    // class members

    private string[] val = new string[3];     // Indexer declaration Here pubic is the modifier string is the return type of  Indexer and "this" is the keyword having parameters list

    public string this[int index]

    {


        // get Accessor retrieving the values  stored in val[] array of strings

        get

        {    return val[index];

        }


   // set Accessor setting the value at passed index of val
```


```
set
        {

            // value keyword is used to define the value being assigned by the set indexer.
            val[index] = value;
        } }}

// Driver Class
class main {
// Main Method
    public static void Main() {

        // creating an object of parent class which
        // acts as primary address for using Indexer
        IndexerCreation ic = new IndexerCreation();

        // Inserting values in ic[]
        // Here we are using the object
        // of class as an array
        ic[0] = "C";
        ic[1] = "CPP";
        ic[2] = "CSHARP";
```


```
Console.Write("Printing values stored in objects used as arrays\n");


    // printing values

    Console.WriteLine("First value = {0}", ic[0]);

    Console.WriteLine("Second value = {0}", ic[1]);

    Console.WriteLine("Third value = {0}", ic[2]);

    }

}
```

7 b) What are the different ways of accessing methods as class members. Write a program to illustrate "passing by output".

The different ways are:

- Pass by Value
- Pass by Reference
- Pass by Output

Passing parameters by output

A return statement can be used for returning only one value from a function. However, using **output parameters**, you can return two values from a function.

Output parameters are similar to reference parameters, except that they transfer data out of the method rather than into it.

```
namespace CalculatorApplication

{

class NumberManipulator
```

```
{

public void getValue(out int x )
        {
                int temp = 5;
                x = temp;
        }
static void Main(string[ ] args)


{
                NumberManipulator n = new NumberManipulator();
               int a = 100;
               Console.WriteLine("Before method call, value of a :
               {0}", a);
               n.getValue(out a);
               Console.WriteLine("After method call, value of a :
               {0}", a);
               Console.ReadLine();
        }
 } }
```

8) What is encapsulation? Write a C# program to explain accessor and mutator properties used in encapsulation.

It is the process of hiding the irrelevant information and showing only the relevant information of a specific object to a user.

It is the process of wrapping up data and members of a class.

It minimizes data corruption thereby providing security.

It increases the maintainability of code.

Encapsulation using assessors and mutators

- To provide security to our code we make our variable private.
- To access them we define two methods. Set ( ) method called as mutator, sets the value of the private variable.
- Get ( ) method called accessor, returns the value of the private variable.

Encapsulation using properties

By using get and set keywords and defining properties.

```
using System;

public class DemoEncap {

   // private variables declared
   // these can only be accessed by
   // public methods of class
   private String studentName;
   private int studentAge;

   // using accessors to get and
   // set the value of studentName
   public String Name
   {

      get
      {
```

```
        return studentName;
      }

      set
{
        studentName = value;
      }

   }

   // using accessors to get and
   // set the value of studentAge
   public int Age
   {

      get
      {
        return studentAge;
      }



  set
      {
        studentAge = value;
      }

   }


}

class GFG {

   // Main Method
   static public void Main()
   {

      // creating object
      DemoEncap obj = new DemoEncap();

      // calls set accessor of the property Name,
      // and pass "Ankita" as value of the
```

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

```
// standard field 'value'
    obj.Name = "Ankita";

    // calls set accessor of the property Age,
    // and pass "21" as value of the
    // standard field 'value'
    obj.Age = 21;
```

PART – V

9) What is Polymorphism? With an example explain Operation Overloading and Function Overloading.

The word **polymorphism** means having many forms. In object-oriented programming paradigm, polymorphism is often expressed as 'one interface, multiple functions'.

Function overloading

You can have multiple definitions for the same function name in the same scope.

The definition of the function must differ from each other by the types and/or the number of arguments in the argument list.

You cannot overload function declarations that differ only by return type.

Example :-

```
class Printdata

{

void print(int i)

{

Console.WriteLine("Printing int: {0}", i );

}

void print(double f)

{
```

```
Console.WriteLine("Printing float: {0}" , f);

}

 void print(string s)

 {


Console.WriteLine("Printing string: {0}", s);

 }

static void Main(string[] args)

{

Printdata p = new Printdata();

// Call print to print integer

p.print(5);

// Call print to print float

p.print(500.263);

// Call print to print string

p.print("Hello C++");

Console.ReadKey(); } }
```

Operator overloading

You can redefine or overload most of the built-in operators available in C#.

Programmer can use operators with user-defined types as well.

Overloaded operators are functions with special names the keyword **operator** followed by the symbol for the operator being defined.

Similar to any other function, an overloaded operator has a return type and a parameter list.

```
public static Box operator + (Box b, Box c)
{
Box box = new Box();

box.length = b.length + c.length;
box.breadth = b.breadth + c.breadth;
box.height = b.height + c.height;
return box;
}
```

The above function implements the addition operator (+) for a user-defined class Box. It adds the attributes of two Box objects and returns the resultant Box object.

Over loadable Operators

1	**+, -, !, ~, ++, --** These unary operators take one operand and can be overloaded.
2	**+, -, *, /, %** These binary operators take one operand and can be overloaded.
3	**==, !=, <, >, <=, >=** The comparison operators can be overloaded.

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

10. Explain the different types of inheritance in C# with examples for each.

Inheritance is of four types:-

a) **Single inheritance :-** one base class and one derived class

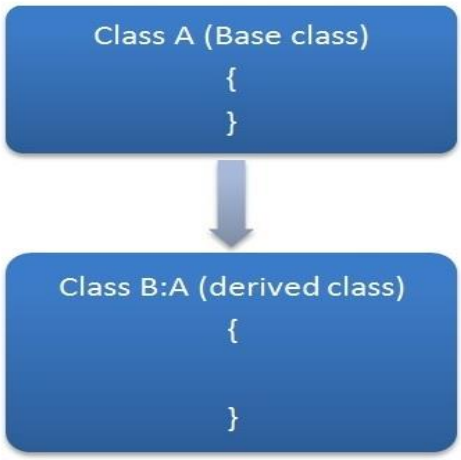
b) **Hierarchal inheritance :-** one base class and multiple derived class

c) **Multilevel inheritance :-** child class is derived from a class which in turn is derived from another class.

d) **Multiple inheritance :-** one child class is derived from multiple base class. (C# doesn't support for classes)

Single inheritance

It is the type of inheritance in which there is one base class and one derived class.



For example:

```
public class Accountcreditinfo
{
   public string Credit()
```

{

return "balance is credited";

USN	1	P	E			M	C	A		

}}

public class debitinfo : Accountcreditinfo

{

public string debit()

{

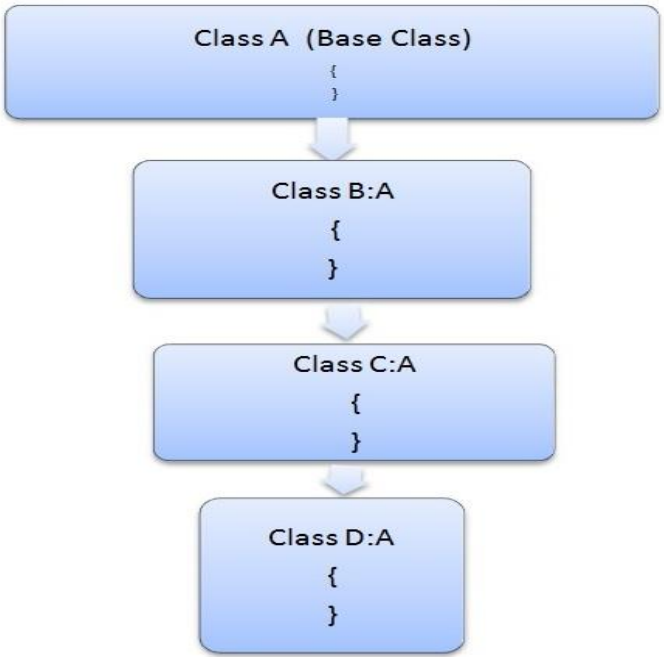
Credit();

return "balance is debited";

}}

Hierarchical inheritance

This is the type of inheritance in which there are multiple classes derived from one base class. This type of inheritance is used when there is a requirement of one class feature that is needed in multiple classes.



USN	1	P	E			M	C	A		

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications

For example:

```
class A
{
   public string msg()
   {return "this is A class Method";
   }}
class B : A
{public string info()
   {msg();
      return "this is B class Method";
   }
   class C : A
```

```
{ public string getinfo()
    { msg();
        return "this is B class Method";
    } }}
```

Multilevel inheritance

When one class is derived from another derived class then this type of inheritance is called multilevel inheritance.

For example:

```
public class Person
{
public string persondet()
```



```
 {
      return "this is the person class";}}
public class Bird : Person
{public string birddet()
   {   persondet();
       return "this is the birddet Class";
   }}
public class Animal : Bird
{    public string animaldet()
      {    persondet();
           birddet();
           return "this is the Animal Class";  }}
```

Multiple inheritance using Interfaces

C# does not support multiple inheritances of classes.

To overcome this problem we can use interfaces, we will see more about interfaces in detail.

USN	1	P	E			M	C	A		

PESIT Bangalore South Campus
Hosur Road, 1km before Electronic City, Bengaluru -560100
Department of Master of Computer Applications