**.NET ADVANTAGES:**

--provides object oriented vie wof windows:.NET framework encapsulates lots of functionalities into classes.

--Application security is built in --need not to worry about authentican and authorization tools.everything is built in

---deploying applic on .net framework is simple.

--versioning issue sare handled.

--all .net languages(u can write some piece of code in C#,VB) are interoperable(all can communicate with each other)

**Using system;//namespace declaration**

This line tells rest of code that we are going to make use of code that is present in “system” class.

Namespace is used to organize your code and is a collection of classes, interfaces, structs, enums and delegates.

Every console application shoudlhave main method which indicates the start and end of a rogram.

Types:

1.Value types:int,float,Boolean etc

2.Reference Types:Interfaces,classes

If you want to make value types as nullable use “?”

Ex:int? i=null;

To convert from one data type to another data typ:

As usual e have type cast as follows

Float f=1234.45f;

Int i=(int)f;

But what if float number is

Float f=12345678.4545f;

Type cast fails to convert this big num to int and it doensnt throw any exception.It just prints the lowest number of int which is -2,31,4…

For this purpose we have to use “Type convert”

Int i=convert.IntTo32(f);

What if you want to convert from string to int??

String name=”HH”

Int i=int.parse(name);

Parse throws na exception if it cannot parse the value

Tryparse returns Boolean indicating whetehr it is succeded or failed.

**Arrays:**

Int[] even=new int[3]//menaing create an integer array of size 3.

**Access modifier:**

1.Public: using public acess mofifier,the method can be called from anywhere.

After public define the return type of yiur nethod.

If you don’t want return anything

Public void methodName()

{

}

If you have statsic keyword for you method ,that’s is salled as static method

Else it is called instance method

Class Program

{

}

Public void EvenNumbers()

{

For (i=0;i<20;i++)

{

If(i%2==0)

Console.writeLine(i);

}

}

To invoke this method you have to create object of a class and using this object you can call the method as follows

Class Program

{

Program p=new Program();

p.EvenNumbers();

}

Public void EvenNumbers()

{

For (i=0;i<20;i++)

{

If(i%2==0)

Console.writeLine(i);

}

}

To invoke a static method,you have to use class name instead of object name as follows:

Class Program

{

Program.EvenNumbers();

}

Passing Parameters:

Class Program

{

Console.WriteLine(“pls eneter the number”);

Console.ReadLine(n);

Program.EvenNumbers(n);

}

Public static void EvenNumbers(int n)

{

For (i=0;i<n;i++)

{

If(i%2==0)

Console.writeLine(i);

}

}

Parameters by value:if you pass a parameter by value like this

Class Program

{

I=0;

Program.EvenNumbers(i);

consolewriteLine(i);

}

Public static void EvenNumbers(int j)

{

J=101;

}

When you pass value of I to j,initially j will be come 0

Then j is changed to 101.

U r printing I value but no where u have hanged value of j to I .

So i=0 is o/p

Parameters by reference:

For the same program,u define keyword ref which is called reference

Class Program

{

I=0;

Program.EvenNumbers(ref i);

consolewriteLine(i);

}

Public static void EvenNumbers(ref int j)

{

J=101;

}

Here the output is i=101 bcoz

Initially i=0

U r passing reference of I say’123’ to j

So j points to the memory location of 123 and it holds value=0

J changes the value it is referring to 101

So j’s memory location is 123 and value=101

Since ‘i’ is also pointing to memory location ‘123’ it also holds the value 101.

Output Parameters:

If you want to return more than one value from your method, you can do it using output parameters since method can return only one parameter.

Class Program{

Int add=0;

Int product=0;

Program.Total(10,20, out add,out product);

Console.WriteLine(“Sum={0},Prod={1}”,add,product);

}

Public static Total(int a,int b,out int sum,out int prod)

{

Sum=a+b;

Prod=a\*b;

}

In output parameters,u haave to define output parameters varaibles both in method and also while calling the method.

**Parameter Arrays:**

We use “Params” keyword.

Class Program{

Int[] numbers1=new int[3];

Numbers1[0]=1;

Numbers1[1]=2;

Numbers1[2]=3;

Program.ArrayParameters(numbers1);

}

Public static void ArrayParameters(params int[] numbers)

Console.writeline(“no.of elements in array is “+numbers.length);

Console.writeLine(“elements in the array are”);

Foreach (int i in numbers)

{

Console.log(i);

}

In the above program,while invoking the method you can even call the method without parameters.This doesn’t throw an error bcoz of usage of params keyword.

While giving 2 or 3 parameters ,parameters array parameter should be the last one.

Using params kewords,you can directly pass the values without initializing them as follows.

Program.ArrayParameters(1,2,3,4,5,6);

**Namespaces in C#:**

🡪Namespace helps you to organise your code.

🡪Help u in avoiding name clashes(ambiguity)

Ex:u have a proj called PROJECTA.

It has two teams A and B.

Classes developed by Team A should be present in PROJECTA.Team A namespace so that by looking at class name I can identify whch team has developed it.

//ly Classes developed by Team B should be present in PROJECTA.Team B namespace.

Thus how it organises to arrange the code.

Ex:Console.writeline(“hello);

Console class is coming from system namespace.

There are two ways to sepia

1.Use the explicit name as follows

ProjectA.Team A.ClassA.print();

2.Or u can use “using” directive as follows

Using ProjectA.Team A;

ClassA.print();

Here you have two namespaces which have same class name .so,if you use above method it throws an error to you bcoz

Using ProjectA.Team A;

Using ProjectA.Team B;

ClassA.print();

classA.print();

If you use the stmnts like this,program doesn’t know which class to be rinted.so u can do one of the following things as follows:

1)Use fuly qualified names as shown below:

ProjectA.Team A .ClassA.print();

ProjectA.Team B.classA().print();

2)namespace aliases

Using print1=ProjectA.Team A;

Using print2=ProjectA.Team B;

Print1.classA.print();

Print2.classA.print();

Ex:

Using namespace system;

Class Program{

ProjectA.Team A.ClassA.print();

ProjectA.Team B.ClassA.print();

}

Namespace ProjectA

{

Namespace Team A

{

Class classA

{

Public static void Print()

{

Console.WriteLine(“This is Team A”);

}

}

}

}

Namespace ProjectA

{

Namespace Team B

{

Class classA

{

Public static void Print()

{

Console.WriteLine(“This is Team B”);

}

}

}

}

o/p:

This is Team A

This is Team B.

Namespace can be nested in two ways:

1.namespace ProjectA.Team A or

2)namespace ProjectA{

Namespace Team A{

}

}

**Class:**

Class consists of data and behaviour

Data is represented by it’s fields

And behaviour is represented by methods.

Classis a kind of template for objects

Class is a template from which objects can get their variables and methods.

**Why to create variables or method as static??**

Area of circle=pi\*r\*r

If you have 10 objects ,in memory for every object it will create a memory for pi and r values

Since pi is a common value which is used y all objects its better to make it as staic and then memory will create only one copy of pi and 10 different r values for 10 diff objects.

**Multiple inheritance is not possible.**

A sub class can extend from only one base class not from multiple base classes.

Ex: public class Triangle:polyon,A is not possible

Multilevel inheritance is possible.

Ex: public class triangle:polygon

{

}

Public class A:Triangle{

}

It means A will inherit properties from both Triangle and polygon.

In inheritance,when we instaniate a child class,automatically parent class is also instantiated and necessary actions will be performed.

But if parent class has two constructors how will u tell ur program to use the other constructor other than the default one

Public class parentclass{

Public parentclass()

{

Console,writeline(“this is default constructor”);

}

Public parentclass(String message)

{

Console.writeline(message);

}

}

Public derivedclass{

Public derivedclass: base(“this is the message I want my derived class to be printed”)

{

Console.writeline(“this is derived class”);

}

}

Public class program{

PUBLCI STATIC VOID MAIN(){

Derivedclass dc=new derivedclass();

}

}

**Method hiding:**

Making the derived class method get executed instead of base class method is called method hiding.

**Diff b/n Method overriding and Method Hiding:**

In method overriding, you will use base class variable which points to derived class object as follows

Baseclass b=new derivedclass();

Here since u have used virual,override methods in respective classes, it will override the baseclasmethod and prints the derved clas sstmnts.

In method hiding if you use base class variable which points to derived class object as follows

Baseclass b=new derivedclass();

Though the derived class is hiding base class method usng new keyword,it still prints the base class method stmnts bcoz here u r using baseclass variable.

**Method overload:**

Metods having same name but either diff no.fo argumenst or diff type of aguments.

Or based on kidn of parameters(like input parameters, output parametrs)

**Properties:**

Usually we use encapsulation (getter and setter methods) to protect fields

In C# we use properties to encapsulate and protect fields

Diff form of representation for get and set methods.

**Interface:**

Just like classes interface also has properties, methods, events or delegates just can be implemented but there will be no implementation of it in interface.

Interface can’t contain member variables.

**Abstract classes vs Interfaces:**

1.Interfaces can’t have fields where as abstract classes can have member fields.

2.Interface can inherit only from other interface but not from abstract class whereas abstract class an inherit from both interaces and other abstract class.

3.abstract class members can have access modifiers where as interface class members cant have acess modifiers.

4.Classes can inherit from multiple interfaces but not from multiple abstract classes.

**Class vs Structs:**

1.A struct is a value type and class is a reference type.

There are several value types in .net

Ex: built in types like int,float,Boolean types,enum

Reference types like : classes,interfaces,delegates etc.

2.Structs are stored on stack whereas classes are stored on heap.

Physical memory of a computer is divided by .net into something called stack and heap

Value types like int i=10,int j=20 are stod on stack.

Reference type like class have two things:

Customer c1=new customer()

C1.id=101;

C1.name=”mark”;

1.reference variable c1 which is stored on stack

2.actual customer object id, name are stored on heap.

3.value types are destroyed immediately after the scope is lost whereas for reference types, only the ref variable Is destroyed after the scope is lost. object will still be preented in heap and is later destroyed by garbage collector.

4.When you copy one struct into other ,it creates copies of the struct and hnage in one struct will not affect the copies.

When you copy a class from another class,both class reference variables point to same object and operations on one class reference variable will affect the values held by other reference variables.

5.Structs can’t have destructors hereas classes can have destructors.

6.Structs can’t have parameter less constructors whereas classes can have parameter less constructor.

Delegates:

A delegate is a type safe functional pointer.

A delegate is created which will point to the function to invoke the function.

I have you confirmed to speak with Brad Best (director), Jude Bowyer (Manager), Roger Finnigan (senior Consultant) and James Bailey (senior consultant). Below is the bridge number for you to dial into.  I will be away on vacation. If you have any questions or if you need to reschedule please contact lynn. I have cc’d her on this email

[1-866-943-2930](tel:1-866-943-2930)

Conference ID

5044720