conditional and looping statements

1. Conditional Branching  
2. Conditional Looping

conditional Branching

1. IF statement  
2. Switch Statement

If(a>b)  
{}

If(a>b)  
{}  
else  
{}

If(a>b)-> True or False  
{  
If()  
{  
}  
}  
else if(b>c)  
{}

2. Switch statement

Switch(Expression)-> 4/2  
{  
Case <value1>: Case 1  
<statements>;  
Break;

Case <value2>: Case 2  
<statements>;  
Break;

Default:  
<statements>;  
Break;  
}

II-> Condititional Loops

1. for loop  
2. While Loop  
3. Do While  
4. Foreach loop

For Loop:

3 things to consider in For Loop

For(Initialiser;Condition;Iteration)  
{

}

for(int i=0;i<10;i++)  
{  
<statements>  
}

2. While Loop  
While(condition)  
{

}

int i=10;  
Whlie(i/0) -> condition-> Boolean- True of False  
{  
<stements>;  
i--;  
}

3. Do While Loop

Do  
{  
<statements>;  
}  
While(condition)

Garbage Collection and Garbage Collector  
Revolves Around Memory

Managed Code-> CLR- CLS and CTS complaint  
allocation and De allocation ->

CLR-> Function-> GC.Collect()-> this Should not be executed

int a=0  
float b=1.0

Allocate and deallocate

Exception Handling

try  
Catch  
finally  
throw

int i=10;  
While(i/0)  
{  
throw (System.DividebyException)  
}  
-> Exception-> Yellow page.

try  
{  
int i=10;  
While(i/0)  
{  
}

}

Catch(Exception ex)  
{  
Log the Exception into text file and support team   
}: NOT going to see the Yellow page.

finally  
{  
some Statements tobe executed mandatorily if the Exception occurs or NOT  
}

Types of Exceptions-> System.Exception

1. System.IO.IOExeption  
2. System.IndexOutOfRangeException  
3. System.ArrayTypeMismatchException  
4. System.NullReferenceException  
5. System.DivideByZeroException  
6. System.InvalidCastException  
7. System.OutOfMemoryException  
8. System.StackOverflowException

Classes and Objects  
BluePrint of Assosiated artefacts  
Logical grouping of Elements  
Logical grouping of components  
Logical grouping of entities

A Class can contain  
Constructors/Destructors  
Variables/Datatypes  
Constants  
Properties  
Methods  
Structs

Instantiation of the Class ->Object

Types of Classes  
1. Abstract class  
2. Concrete Class  
3. Static Class  
4. Sealed class

Method signature-> <Access modifier><Return Type> Method Name(Zero of More parameters)

public int Addition() -> Method signature-> <Access modifier><Return Type> Method Name  
{  
//Method body-> Implementation of a method  
}

Abstract Class-> Which contains ZERO or more Abstract members/methods

Abstract Method-> A Method which contains only the Signature and does not contain the implementation

Abstract Cannot be instantiated-> Inherited in Derived class and used after that

3. Static Class-> It can contain only Static Methods/ variables/elements

I cannot Create an instance of Static class-> Which can be used without creating an instance  
It can contain only Static Members

4. Sealed class-> Inheritance concept

Base Class and some properties and Methods, Elements  
Derived Class Inherit Base Class with a symbol :

Sealed class cannot be inheritited-> Base class will loose the inheritance property

Interface->

Access is public by default

Interface contains only the Method Signatures and NO implementations  
It has to be Implemented  
all the members of an interface are public by default

Difference between Abstract Class and An interface:

1. Abstract calss can contain Abstract methods (Methods with just sig) and NO body  
as well as concrete methods (Methods with sig + body (implemtation)

Interface cannot contain any Concrete methods  
Just method sig

2. Abstract class i can define and assign the access modifier   
Memer of iterface are public by default

When do you go for Abstract and , when do you go for Interface?

Addition Method-> Implemntation of Addition method ois Diff in US and IN and Its same across all other countries

Abstract Class->

Addition\_>  
Implementation also (Default (other countries)  
LEt IN and US Override my addition