

Constructors

These are methods which are called at the time of object declaration & they are used to initialize objects

Features of Constructors :

1. Their name is same as class name
2. They don’t have any return type
3. They follow concept of Polymorphism
4. They are like methods but they are not called explicitly. They are called automatically at time of object declaration

Type of Constructors

1. Default Constructors (Parameterless) (Only 1)
2. Parameterized Const (They contain para) (Multiple)
3. Static (One)
4. Copy
5. Private

Contstructors are called at what point

When we write new()

Student student = new Student();

Student student = new Student(1,”Ajay”);

Arrays & Collections

Int[] number = new int[10];

**Limitation in Array s:**

Fixed Size, Memory could be wasted

Memory need should be known at compile time

Insertion/ Deletion is time consuming

Can not delete elements

**Advantage of Arrays : All elements are of same type**

Collections > Dynamic

ArrayList > Insertion/Deletion can be done anywhere

Stack > LIFO . insertion & deletion at end

Queue

HashSet

ArrayList list = new ArrayList();

List.Add(10);

List.InsertAt(0,100);

List.Remove(1);

**Advantages of using Collections**

No Fixed Size, Memory will not be wasted

Memory need should not be known at compile time, decided at run time

Insertion/ Deletion is easier

We can delete elements

Stack stack = new Stack();

Stack.Push(10);

Stack.Push(20);

Stack.Push(30);

Stack.Push(40);

Stack.Pop(); > 40

Queue queue = new Queue();

Queue.Enque(1);

Queue.Enque(2);

Queue.Enque(3);

Queue.Enque(4);

Queue.DeQue(); > 1

Foreach(int x in stack)

Console.Write(x);

HashSet hs = new HashSet();

Hs[“Ajay”] = 10;

Hs[“Deepk”] = 20;

Consolw.Write(hs[“Deepak”]); > 20

**Limitation in Collections> All the elements of dif types**

ArrayList list = new ArrayList();

List.Add(10);

List.Add(“Ajay”);

List.InsertAt(0,100);

List.Remove(1);

Foreach(var x in list)

Console.Write(x);

// Generic Collection > Taka advantage of both Arrays & Colections

**All elements are of same type**

No Fixed Size, Memory will not be wasted

Memory need should not be known at compile time, decided at run time

Insertion/ Deletion is easier

List<int> list = new List<int>();

List.Add(1);

Stack<string> stack = new Stack<string>();

Stack.Push(“Ajay”);

Dictionary<int,int> hs = new Dictionary<int,int>();

Hs[1] = 20;

ADO.Net

Entity Framework

BE

Database

FE

Console

Web

Windows

FE BE

**TOMORROW :**

ADO.Net > WAP in Console & connect with Database using

1. Ado.Net
2. Thru Entity Framework (Code First Approach)

Table Student > Rn Name Batch Marks

Polymorphism

Abstract Class

Inheritance

Compile Time & Run Time Polym