Namespace : Collection of logically related classes

Namepspace N1

{

Class A

{

}

}

Namepspace N2

{

Class A

{

}

}

}

-------------------------------------------------------------

namespace Day3Demos

{

class Program

{

static void Main(string[] args)

{

Day3Demos2.Program p = new Day3Demos2.Program();

Day3Demos.Program p1 = new Program();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Day3Demos2

{

class Program

{

}

}

OOPS

Class > User Defined Type

Object > Instance of a class

Int x;

Student student;

Encapsulation > Hiding the details which are not needed by the user

Abstraction > Showing the details which are needed by the user

How do we achieve this > By using Access Specifiers

* Private (By default all class members are private), which means that they can be accessed within the class
* Public : The members are accessible outside the class , through objects
* Protected (This is used in Inheritance only)
* Internal (By default class is internal) That the members are accessible within the Assembly
* Protected internal Inheritance

Syntax for class

Class <classname>

{

Members;

Variables;

Properties

Methods

Indexer;

}

Student student = new Student();

New keyword will do 2 things

1. It allocates memory from heap
2. It initializes all variables of class by calling Constructor(default constructor)

class Student

{

int rn;

string name;

string batchCode;

int marks;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Convert.ToByte(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch Code");

batchCode = Console.ReadLine();

Console.WriteLine("Enter Marks");

marks = Convert.ToByte(Console.ReadLine());

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is " + rn);

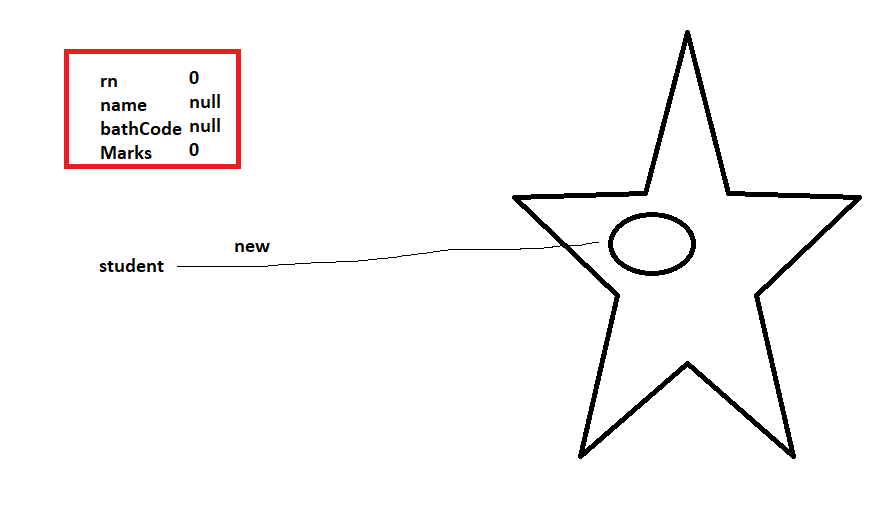
Console.WriteLine("Name is " + name);

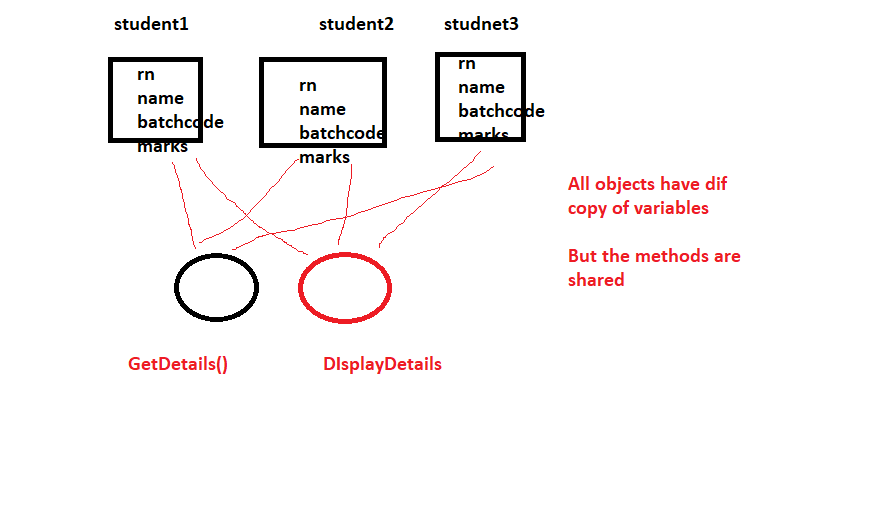
Console.WriteLine("Batch Code is " + batchCode);

Console.WriteLine("Marks are " + marks);

}

}



What is an object ? Instance of a class

class Student

{

int rn; // Instance Variables

string name;

string batchCode; // shud be static variable

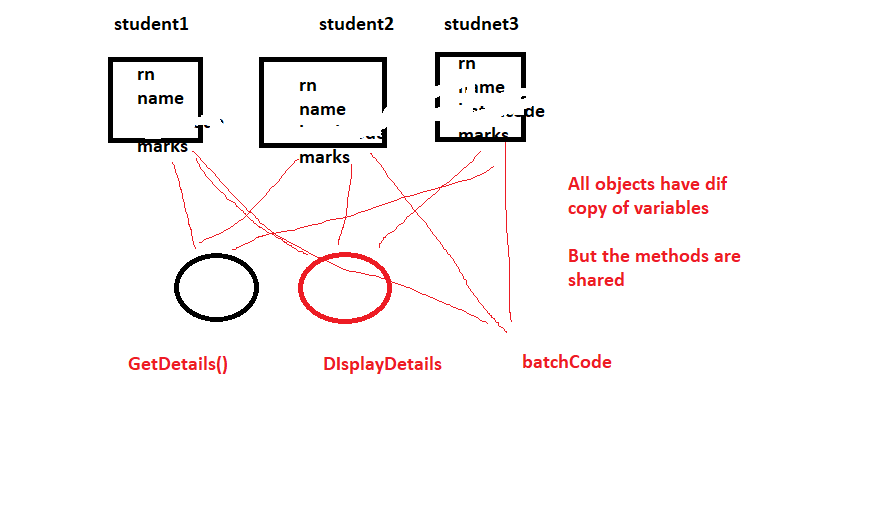
int marks;}

Student student = new Student();

**Instance Variables >** These are the variables which are different for every object

Rn , name , marks

**Static Variables >** These are the variables which are shared by all objects. So there is a single copy of this variable



How do you give static variable value

1. You assign it a value while declaring it

static string batchCode="B001";

1. Use Static Methods > Static Methods are methods which can access only static members

Static Variable > There is a single copy of this variable. It is shared by all objects. We can change its value anytime. It is Accessed at Class level

3. **Constant Variable >** Are variables whose value once assigned cannot be changed

It is Accessed at Class level. When you declare it, we have to give it a value at that time only

4.**ReadOnly variables >** Are variable attached to the objects, but once you assign value, it cannot be changed. We can assign value to these variblaes at 2 point,

1. While declaring
2. In a constructor

Loops

For()

Foreach()

For(int i=1;i<10;i++)

{

}

Foreach loop can be used to iterate through a collection or an array

foreach(datatype <rangeVariable> in <array/collection>)

{

Console.WriteLine(rangeVariable);

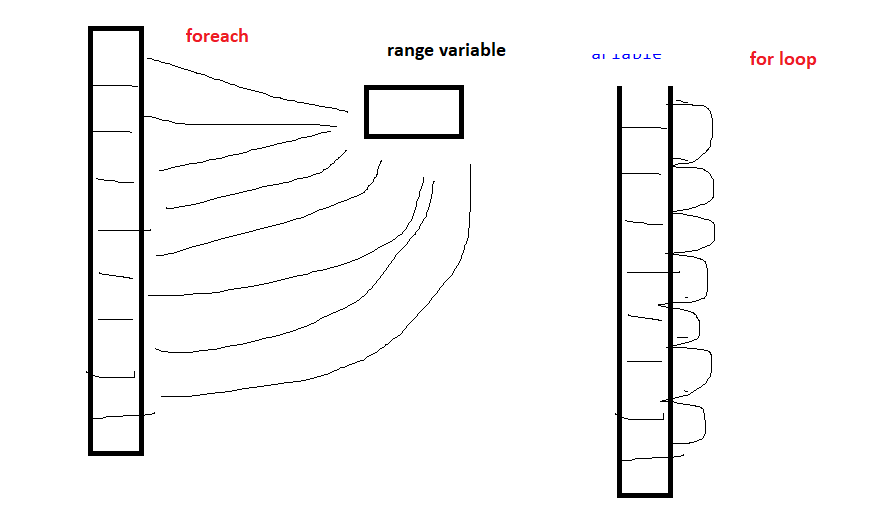
}

foreach(int temp in num)

{

Console.WriteLine(temp);

}



Foearch loop is slower as compared to or loop

We cannot change values in foreach loop(Its only to get / fetch values from Collection / Array)

Boxing / Unboxing is also there which again makes this loop slower