Constructors : Used to initialize attributes of an object at compile time

Int x;

Int x =10;

class student

{

int rn;

string name;

string batch;

int marks;

}

Constructors : They are used to initialize attributes of an object at compile time

Features of a constructor

1. Their name is same as class name
2. They are like member methods , but we don’t call them , they are invoked automatically at time of object declaration.
3. They do not have any return type
4. They follow concept of polymorphism
5. By default , within a class, there is default constructor

Type of Constructors (5 types)

Parameterless / Parameterized

Default constructors : We don’t pass any parameter, There can be only 1 default constructor within a class

Parameterized Constructor > In which we pass parameters , There could be multiple Parameterized Constructors within a class

Copy constructor : Where we want to copy values of cone object to other object

Static constructor : which are used to initialize only static variables (Can be only one)

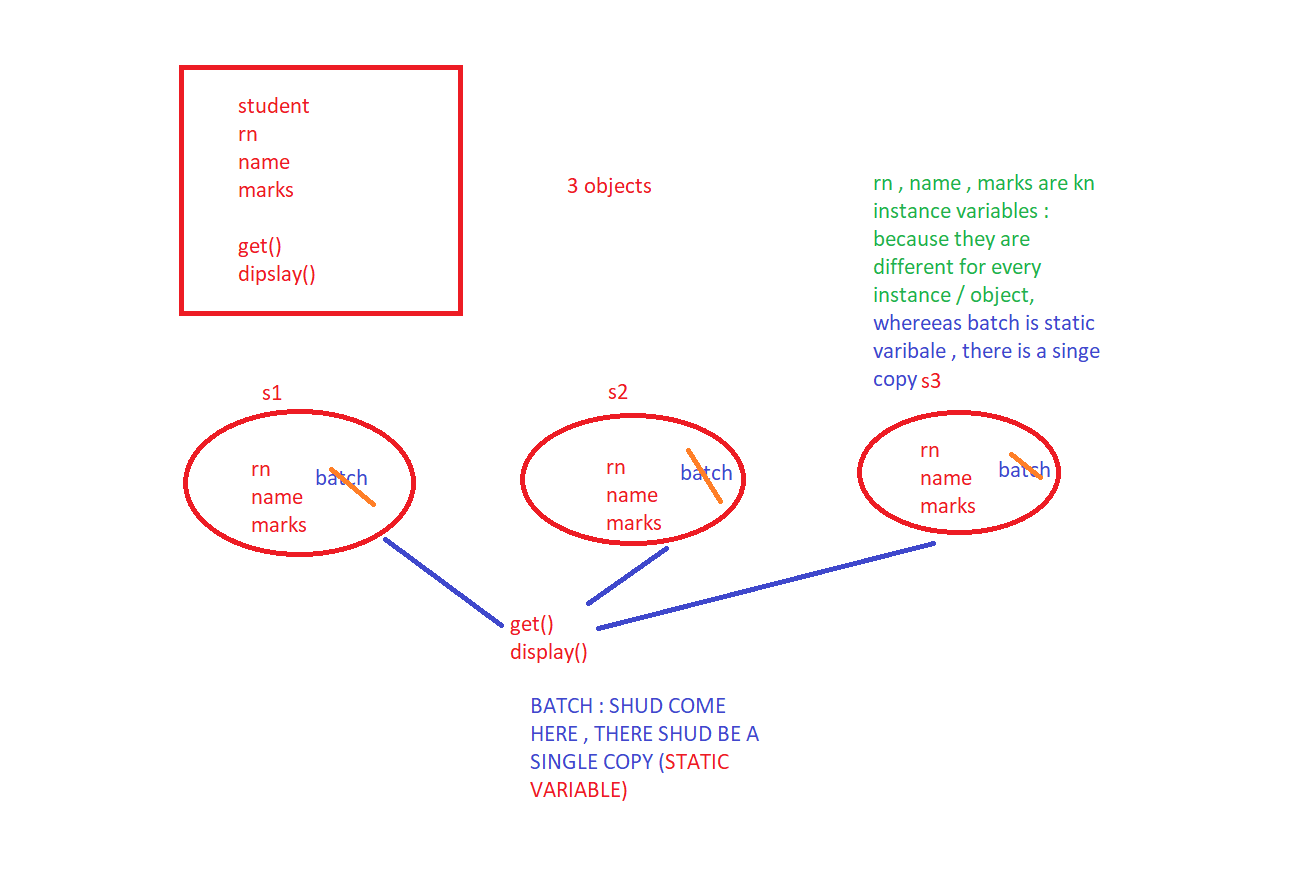
Private constructor :

By default, value type variable get their default values

Int gets 0

Boolean gets false

Reference type variables get by default null

Static variables : variables which are shared by all objects of a class, so there is a singe copy

using System;

using System.Collections;

using System.Collections.Generic;

class student

{

int rn;

string name;

string batch;

int marks;

// default constructor

public student() {}

// parameterized constructor

public student(string name)

{

name = name;

Console.WriteLine("Enter RollNo");

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

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

// parameterized constructor

public student(int rn , string name)

{

name = name;

rn = rn;

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

// parameterized constructor (fully parameterized)

public student(int rn, string name, int marks, string batch)

{

name = name;

rn = rn;

batch = batch;

marks = marks;

}

// copy constructor

public student(student obj)

{

Console.WriteLine("Enter RollNo");

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

name = obj.name;

marks = obj.marks;

batch = obj.batch;

}

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

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

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

public void DisplayDetails()

{

Console.WriteLine("Roll No is " + rn);

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

Console.WriteLine("Batch is " +batch);

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

}

}

class Program

{

static void Main()

{

student s = new student();

s.GetDetails();

s.DisplayDetails();

student s2 = new student(name:"Jayant");

s2.DisplayDetails();

student s3 = new student(rn:11, name:"Mounika");

s3.DisplayDetails();

student s4 = new student(marks:12, name:"Neelavani", rn:90,batch: "B001");

s4.DisplayDetails();

student s5 = new student(s4); //opy constructor

}

}

We can access static variables in 2 ways

1. Make it public , then we can access it like : student.batch=”B001”;
2. Use Static methods , static method is a method which can access only static members

using System;

using System.Collections;

using System.Collections.Generic;

class student

{

int rn;

string name;

public static string batch;

int marks;

public static void GetBatchDetails()

{

batch = "B001";

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

}

// default constructor

public student() {}

// parameterized constructor

public student(string name)

{

this.name = name;

Console.WriteLine("Enter RollNo");

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

//Console.WriteLine("Enter Batch");

//batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

// parameterized constructor

public student(int rn , string name)

{

this.name = name;

this.rn = rn;

//Console.WriteLine("Enter Batch");

//batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

// parameterized constructor (fully parameterized)

public student(int rn, string name, int marks, string batch)

{

this.name = name;

this.rn = rn;

//batch = batch;

this.marks = marks;

}

// copy constructor

public student(student obj)

{

Console.WriteLine("Enter RollNo");

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

name = obj.name;

marks = obj.marks;

// batch = obj.batch;

}

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

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

Console.WriteLine("Enter Name");

name = Console.ReadLine();

//Console.WriteLine("Enter Batch");

//batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

public void DisplayDetails()

{

Console.WriteLine("Roll No is " + rn);

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

Console.WriteLine("Batch is " +batch);

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

}

}

class Program

{

static void Main()

{

// student.batch = "B001";

student.GetBatchDetails();

student s = new student();

s.GetDetails();

s.DisplayDetails();

student s2 = new student(name:"Jayant");

s2.DisplayDetails();

student s3 = new student(rn:11, name:"Mounika");

s3.DisplayDetails();

student s4 = new student(marks:12, name:"Neelavani", rn:90,batch: "B001");

s4.DisplayDetails();

student s5 = new student(s4); //copy constructor

}

}

Static constructor is always called / invoked first before any other constructor

using System;

using System.Collections;

using System.Collections.Generic;

class student

{

int rn;

string name;

public static string batch;

int marks;

public static void GetBatchDetails()

{

// batch = "B001";

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

}

// default constructor

public student() {}

// parameterized constructor

public student(string name)

{

this.name = name;

Console.WriteLine("Enter RollNo");

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

//Console.WriteLine("Enter Batch");

//batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

// parameterized constructor

public student(int rn , string name)

{

this.name = name;

this.rn = rn;

//Console.WriteLine("Enter Batch");

//batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

// parameterized constructor (fully parameterized)

public student(int rn, string name, int marks, string batch)

{

this.name = name;

this.rn = rn;

//batch = batch;

this.marks = marks;

}

// copy constructor

public student(student obj)

{

Console.WriteLine("Enter RollNo");

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

name = obj.name;

marks = obj.marks;

// batch = obj.batch;

}

static student()

{

batch = "B001";

}

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

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

Console.WriteLine("Enter Name");

name = Console.ReadLine();

//Console.WriteLine("Enter Batch");

//batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

public void DisplayDetails()

{

Console.WriteLine("Roll No is " + rn);

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

Console.WriteLine("Batch is " +batch);

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

}

}

class Program

{

static void Main()

{

// student.batch = "B001";

//student.GetBatchDetails();

student s = new student();

s.GetDetails();

s.DisplayDetails();

student s2 = new student(name:"Jayant");

s2.DisplayDetails();

student s3 = new student(rn:11, name:"Mounika");

s3.DisplayDetails();

student s4 = new student(marks:12, name:"Neelavani", rn:90,batch: "B001");

s4.DisplayDetails();

student s5 = new student(s4); //copy constructor

}

}