Class student

{

Int rn;

String name;

Static String batch;

Void Get() {}

Void Calculate() {}

Void Display(){}

}

Static variables are not associated with the objects. They are associated with class

STATIC : NO OBJECT NEEDED

How to access static variables

2 ways

1. Make the static variables public , then we can access it outside the class
2. Use Static Methods

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoProject

{

class Employee

{

int Empid;

public string name;

// Static varibales

public static string dept="Accounts";

public static String Manager="Ajay";

double BasicSalary;

int Exp;

double DA, HRA, PF, NetSalary;

public void GetEmployeeDetails()

{

Console.WriteLine("Enter Employee ID");

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

Console.WriteLine("Enter Name of the employee");

name = Console.ReadLine();

//Console.WriteLine("Enter Department name");

//dept = Console.ReadLine();

//Console.WriteLine("Enter Manager name");

//Manager = Console.ReadLine();

Console.WriteLine("Enter Basic Salary");

BasicSalary = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Experience of the employee");

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

}

public void CalculateNetsalary()

{

if (Exp <= 5)

{

DA = (1.9 \* BasicSalary) / 100;

HRA = (2.0 \* BasicSalary) / 100;

PF = 1200;

}

else if (Exp > 5 && Exp <= 7)

{

DA = (4.1 \* BasicSalary) / 100;

HRA = (3.8 \* BasicSalary) / 100;

PF = 1800;

}

else if (Exp > 7 && Exp <= 10)

{

DA = (7 \* BasicSalary) / 100;

HRA = (6.5 \* BasicSalary) / 100;

PF = 4100;

}

else if (Exp > 10)

{

DA = (10 \* BasicSalary) / 100;

HRA = (8.5 \* BasicSalary) / 100;

PF = 6200;

}

NetSalary = (BasicSalary + DA + HRA) - PF;

}

public void DisplayEmployeeDetails()

{

Console.WriteLine("Employee ID : " + Empid);

Console.WriteLine("Employee Name : " + name);

//Console.WriteLine("Department is " + dept);

//Console.WriteLine("Manager is " + Manager);

Console.WriteLine("Basic Salary is" + BasicSalary);

Console.WriteLine("DA is" + DA);

Console.WriteLine("HRA is" + HRA);

Console.WriteLine("PF is" + PF);

Console.WriteLine("Net salary is" + NetSalary);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoProject

{

class Program

{

static void Main()

{

Employee.dept = "Accounts";

Employee.Manager = "Deepak";

Console.Write("Department is " + Employee.dept);

Console.WriteLine("\t Manager is " + Employee.Manager);

Employee employee1 = new Employee();

employee1.GetEmployeeDetails();

employee1.CalculateNetsalary();

employee1.DisplayEmployeeDetails();

Employee employee2 = new Employee();

employee2.GetEmployeeDetails();

employee2.CalculateNetsalary();

employee2.DisplayEmployeeDetails();

}

}

}

Static Method : These are the methods which can access only static members

Static Main(){}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoProject

{

class Employee

{

int Empid;

public string name;

// Static varibales

static string dept;

static String Manager;

double BasicSalary;

int Exp;

double DA, HRA, PF, NetSalary;

public static void GetManagerDetails()

{

Manager = "Deepak";

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

}

public static void GetDepartmentDetails()

{

dept = "Accounts";

Console.WriteLine("Department Name is " + dept);

}

public void GetEmployeeDetails()

{

Console.WriteLine("Enter Employee ID");

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

Console.WriteLine("Enter Name of the employee");

name = Console.ReadLine();

//Console.WriteLine("Enter Department name");

//dept = Console.ReadLine();

//Console.WriteLine("Enter Manager name");

//Manager = Console.ReadLine();

Console.WriteLine("Enter Basic Salary");

BasicSalary = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Experience of the employee");

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

}

public void CalculateNetsalary()

{

if (Exp <= 5)

{

DA = (1.9 \* BasicSalary) / 100;

HRA = (2.0 \* BasicSalary) / 100;

PF = 1200;

}

else if (Exp > 5 && Exp <= 7)

{

DA = (4.1 \* BasicSalary) / 100;

HRA = (3.8 \* BasicSalary) / 100;

PF = 1800;

}

else if (Exp > 7 && Exp <= 10)

{

DA = (7 \* BasicSalary) / 100;

HRA = (6.5 \* BasicSalary) / 100;

PF = 4100;

}

else if (Exp > 10)

{

DA = (10 \* BasicSalary) / 100;

HRA = (8.5 \* BasicSalary) / 100;

PF = 6200;

}

NetSalary = (BasicSalary + DA + HRA) - PF;

}

public void DisplayEmployeeDetails()

{

Console.WriteLine("Employee ID : " + Empid);

Console.WriteLine("Employee Name : " + name);

//Console.WriteLine("Department is " + dept);

//Console.WriteLine("Manager is " + Manager);

Console.WriteLine("Basic Salary is" + BasicSalary);

Console.WriteLine("DA is" + DA);

Console.WriteLine("HRA is" + HRA);

Console.WriteLine("PF is" + PF);

Console.WriteLine("Net salary is" + NetSalary);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoProject

{

class Program

{

static void Main()

{

//Employee.dept = "Accounts";

//Employee.Manager = "Deepak";

//Console.Write("Department is " + Employee.dept);

//Console.WriteLine("\t Manager is " + Employee.Manager);

Employee.GetManagerDetails();

Employee.GetDepartmentDetails();

Employee employee1 = new Employee();

employee1.GetEmployeeDetails();

employee1.CalculateNetsalary();

employee1.DisplayEmployeeDetails();

Employee employee2 = new Employee();

employee2.GetEmployeeDetails();

employee2.CalculateNetsalary();

employee2.DisplayEmployeeDetails();

}

}

}

using System;

class Program

{

static void add(int x , int y)

{

Console.WriteLine(x+y);

}

static void Main()

{

add(10, 90);

}

}

Instance Variables : address , state , phoneno

Static variables : batch , training , trainer , timings, cohort code, course

Readonly variables : rn , dob , name (associated at object level) ,

we can assign value to this variable either while declaring it or in a constructor

Constant variables : companyname (associated at class level), when you declare, it is mandatory to assign a value

Constructor : It is used to initialize variables of objects at compile time

Features of constructor

1. Their name is same as class name
2. They are just like member methods, But we do not call them explicitly
3. They are invoked or called automatically when objects are declared
4. They do not have return type
5. They follow polymorphism (Method Overloading)

Types of constructors

1. Default constructor : These are parameter less , They are by default present in the class, but if we create other constructors, then it will become mandatory to declare it also

**(ONLY 1 IN A CLASS)**

1. Parameterized constructors : Here we pass parameters

**(COULD BE MULTIPLE IN A CLASS)**

1. Static constructor : They are used to initialize only static variables.

They are also parameter less, we cannot add any access specifier

**(ONLY 1 IN A CLASS) (IT IS ALWAYS INVOKED FIRST)**

1. Copy constructor: They are used to copy values of one object to other
2. Private constructor: When we don’t add any access specifier

Int x;

Int x= 20;

Console.Write(“Enter value of x”);

Class student

{

Int rn;

String name;

Void Get(){}

Void Display() {}

Public student() {

}

Public student(int rn)

{Entere yiur Name

}

Student s1= new Student();

S1.Get();

S1.Display();

Student s2 = new student(12);

S2.Get();

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoProject

{

class Student

{

int rn;

string name;

string batch;

int marks;

static string course;

public void GetStudentDetails()

{

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 DisplayStudentDetails()

{

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

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

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

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

}

// Default constructor

public Student() { }

// Parameterized constructor

public Student(int x , string y)

{

rn = x;

name = y;

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

// Parameterized constructor

public Student(int x)

{

rn = x;

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

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

}

// Parameterized constructor

public Student(int x, string y, string z, int a)

{

rn = x;

name = y;

batch = z;

marks = a;

}

// Static constructor

static Student()

{

course = "Dot Net";

}

// Copy constructor

public Student(Student obj)

{

Console.WriteLine("Enter RollNo");

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

name = obj.name;

batch = obj.batch;

marks = obj.marks;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoProject

{

class Program

{

static void Main()

{

Student student1 = new Student();

//student1.GetStudentDetails();

//student1.DisplayStudentDetails();

Student student2 = new Student(101,"Doris");

//student2.GetStudentDetails();

//student1.DisplayStudentDetails();

Student student3 = new Student(102);

//student3.DisplayStudentDetails();

Student student4 = new Student(103, "Gagan", "B001", 90);

//student4.DisplayStudentDetails();

Student student5 = new Student(student4);

}

}

}