Access Specifiers

It tells how we can access elements

1. Private (default for class members)
2. Public (default for interface members)
3. Internal (default for class)
4. Protected (used in inheritance)
5. Protected Internal

Employee.cs file

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Employee

{

int id;

string name;

string dept;

string manager;

float salary;

void GetDetails()

{

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Department");

dept = Console.ReadLine();

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

void DisplayDetails()

{

Console.WriteLine("ID is " + id);

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

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

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

Console.WriteLine("Salary is " + salary);

}

}

}

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

Program.cs file

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Program

{

static void Main(string[] args)

{

Employee employee = new Employee();

}

}

}

Here this new keyword will do two things

1. It allocates memory from heap
2. It invokes constructor (default constructor)

**Constructor**

**Is a special method which is invoked automatically at time of object declaration**

**By default, within a class , we always have default constructor**

**Why do we need it?**

**It is used to initialize members of the object at time of object declaration**

**Features of constructor**

1. **Their name is same as class name**
2. **They are like member methods, but we don’t call them explicitly. They are called automatically at time of object declaration**
3. **They don’t have any return type**
4. **They follow polymorphism**

**Types of constructors**

**Parameterless OR they could be parameterized**

1. **Default constructor (ONE)**
2. **Static constructors (ONE)**
3. **Parameterized constructors**
4. **Private constructors**
5. **Copy Constructors**

**Employee class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Employee

{

int id;

string name;

string dept;

string manager;

float salary;

// default constructor

public Employee()

{

}

// parameretized

public Employee(string name , string dept)

{

this.name = name;

this.dept = dept;

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

// parameterized

public Employee(int id, string name, string dept, string manager)

{

this.name = name;

this.dept = dept;

this.id = id;

this.manager = manager;

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

// fully parameterized

public Employee(int id, string name, string dept, string manager , float salary)

{

this.name = name;

this.dept = dept;

this.id = id;

this.manager = manager;

this.salary = salary;

}

public void GetDetails()

{

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Department");

dept = Console.ReadLine();

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

public void DisplayDetails()

{

Console.WriteLine("ID is " + id);

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

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

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

Console.WriteLine("Salary is " + salary);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Program

{

static void Main(string[] args)

{

// it will invoke default

Employee employee1 = new Employee();

Console.WriteLine("Object which invoked def con");

; employee1.GetDetails();

employee1.DisplayDetails();

// it will invoke para const

Console.WriteLine("Object which invoked para con");

;

Employee employee2 = new Employee(name:"Ajay", dept: "HR");

employee2.DisplayDetails();

// it will invoke para const

Console.WriteLine("Object which invoked para con");

;

Employee employee3 = new Employee(2, "Paras", "Deepak", "HR");

employee3.DisplayDetails();

// it will invoke para const

Console.WriteLine("Object which invoked fully para con");

;

Employee employee4 = new Employee(3, "Paras", "Deepak", "HR", 90000);

employee4.DisplayDetails();

}

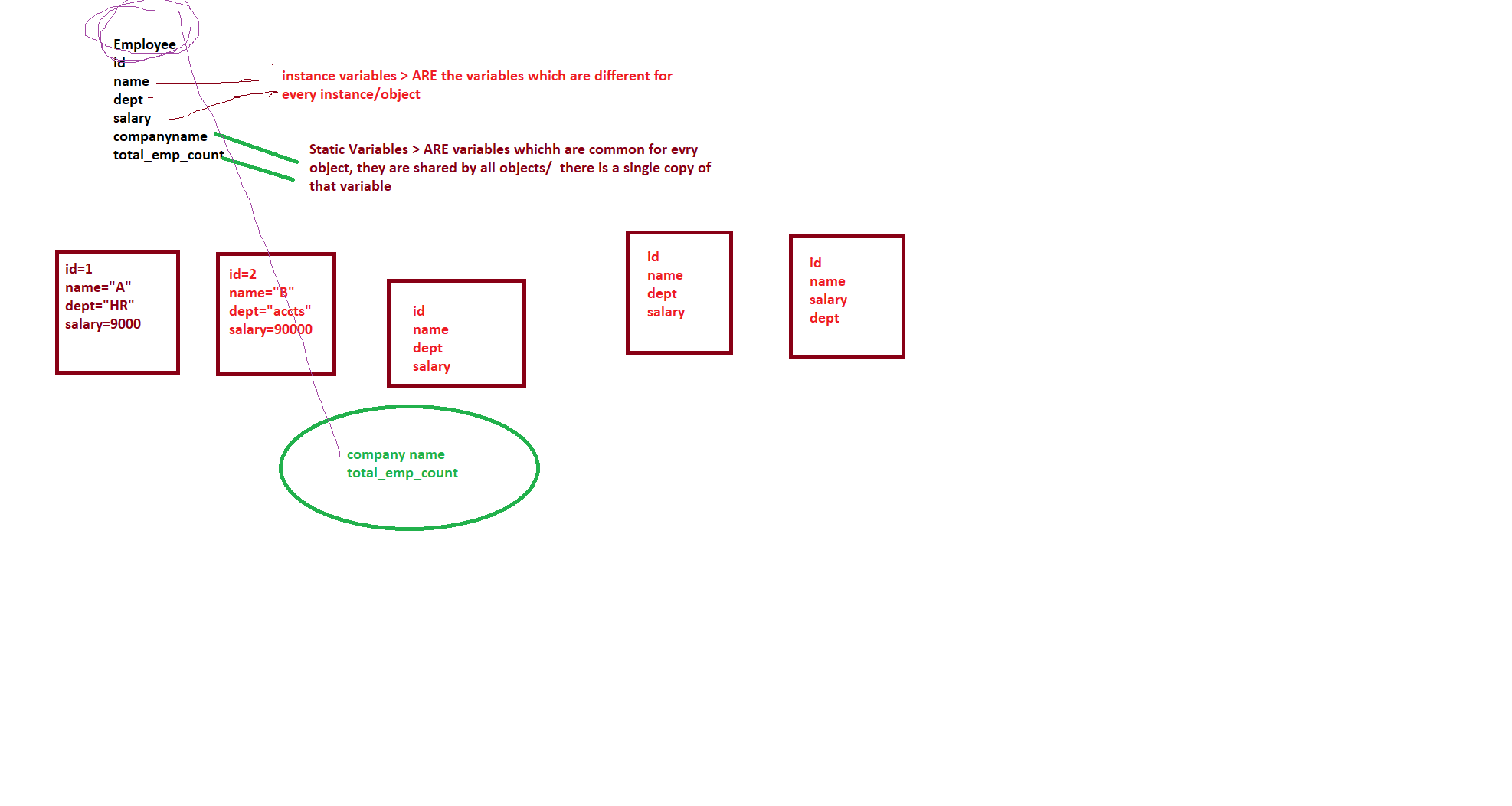
}

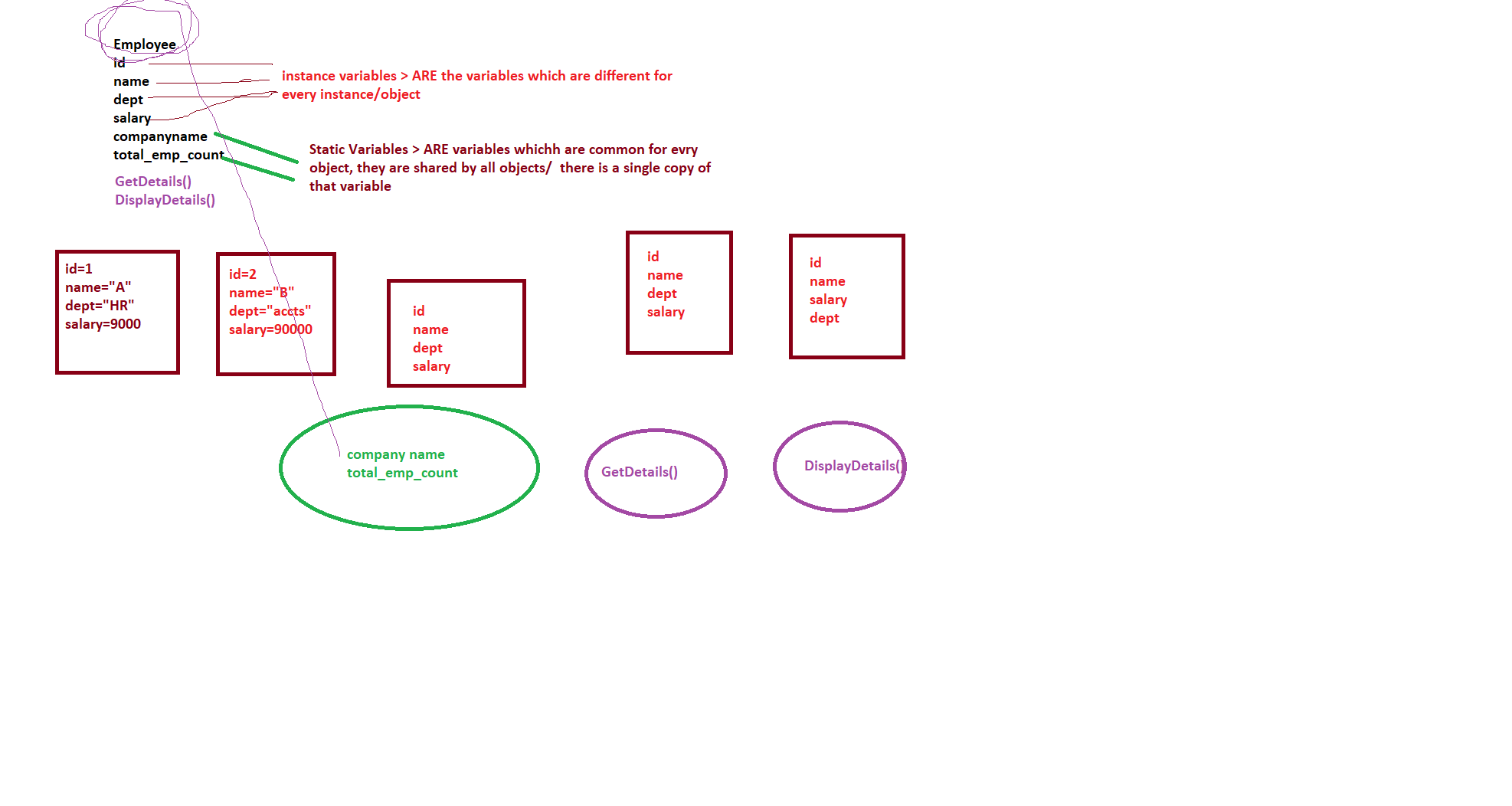
}

Static Variables

Static variables are the variable which are shared by all the objects of a class

There is a single copy of that variable





class Employee

{

// instance varibles from id to salary

int id;

string name;

string dept;

string manager;

float salary;

// static variable

static string companyname;

How do we give value or access static variable

1. You give value while declaring it

// static variable

static string companyname ="CTS";

1. You make it public & then we can access it outside the class

public static string companyname ="CTS";

Employee.companyname = "CTS";

1. Use Static function

Static function is a function which can only access static variables

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Employee

{

// instance varibles from id to salary

int id;

string name;

string dept;

string manager;

float salary;

// static variable

public static string companyname;

// default constructor

public Employee()

{

}

// parameretized

public Employee(string name , string dept)

{

this.name = name;

this.dept = dept;

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

// parameterized

public Employee(int id, string name, string dept, string manager)

{

this.name = name;

this.dept = dept;

this.id = id;

this.manager = manager;

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

// fully parameterized

public Employee(int id, string name, string dept, string manager , float salary)

{

this.name = name;

this.dept = dept;

this.id = id;

this.manager = manager;

this.salary = salary;

}

public void GetDetails()

{

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Department");

dept = Console.ReadLine();

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

public void DisplayDetails()

{

//Console.WriteLine("Company Name is " + companyname);

Console.WriteLine("ID is " + id);

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

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

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

Console.WriteLine("Salary is " + salary);

}

public static void DisplayCompmayName()

{

Console.WriteLine("COMPANY NAME IS : "+ companyname);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Program

{

static void Main(string[] args)

{

Employee.companyname = "CTS";

Employee.DisplayCompmayName();

// Console.WriteLine(Employee.companyname);

// it will invoke default

Employee employee1 = new Employee();

Console.WriteLine("Object which invoked def con");

; employee1.GetDetails();

employee1.DisplayDetails();

// it will invoke para const

Console.WriteLine("Object which invoked para con");

;

Employee employee2 = new Employee(name:"Ajay", dept: "HR");

employee2.DisplayDetails();

// it will invoke para const

Console.WriteLine("Object which invoked para con");

;

Employee employee3 = new Employee(2, "Paras", "Deepak", "HR");

employee3.DisplayDetails();

// it will invoke para const

Console.WriteLine("Object which invoked fully para con");

;

Employee employee4 = new Employee(3, "Paras", "Deepak", "HR", 90000);

employee4.DisplayDetails();

}

}

}

Static Constructor (One)

It I used to initialize only static variables

It can be only one within a class

We cannot use any access specifier with this constructor

And it should be always parameterless

If a class contains def , para , static const . which constructor will be invoked first

**ANS : STATIC CONSTRUCTOR**

Copy constructor : It is used to copy values of one object to other

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Employee

{

// instance varibles from id to salary

int id;

string name;

string dept;

string manager;

float salary;

// static variable

static string companyname;

**// default constructor**

public Employee()

{

}

**// parameretized**

public Employee(string name , string dept)

{

this.name = name;

this.dept = dept;

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

**// parameretized**

public Employee(int id, string name, string dept, string manager)

{

this.name = name;

this.dept = dept;

this.id = id;

this.manager = manager;

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

**// fully parameretized**

public Employee(int id, string name, string dept, string manager , float salary)

{

this.name = name;

this.dept = dept;

this.id = id;

this.manager = manager;

this.salary = salary;

}

**//static constructor**

static Employee()

{

companyname = "CTS";

}

**// copy constructor**

public Employee(Employee employee)

{

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

this.name = employee.name;

this.dept = employee.dept;

this.manager = employee.manager;

this.salary = employee.salary;

}

public void GetDetails()

{

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Department");

dept = Console.ReadLine();

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

public void DisplayDetails()

{

//Console.WriteLine("Company Name is " + companyname);

Console.WriteLine("ID is " + id);

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

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

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

Console.WriteLine("Salary is " + salary);

}

public static void DisplayCompmayName()

{

// companyname = "CTS";

Console.WriteLine("COMPANY NAME IS : "+ companyname);

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Program

{

static void Main(string[] args)

{

**// The first cons that will be invoked is static constr**

//Employee.companyname = "CTS";

Employee.DisplayCompmayName();

// Console.WriteLine(Employee.companyname);

**// it will invoke default**

Employee employee1 = new Employee();

Console.WriteLine("Object which invoked def con");

; employee1.GetDetails();

employee1.DisplayDetails();

**// it will invoke para const**

Console.WriteLine("Object which invoked para con");

;

Employee employee2 = new Employee(name:"Ajay", dept: "HR");

employee2.DisplayDetails();

**// it will invoke para const**

Console.WriteLine("Object which invoked para con");

;

Employee employee3 = new Employee(2, "Paras", "Deepak", "HR");

employee3.DisplayDetails();

**// it will invoke para const**

Console.WriteLine("Object which invoked fully para con");

;

Employee employee4 = new Employee(3, "Paras", "Deepak", "HR", 90000);

employee4.DisplayDetails();

**// It invokes copy constr**

Employee employee5 = new Employee(employee4);

employee5.DisplayDetails();

}

}

}

**// Polymorphism : One name different forms**

Add()

Add(int x, int y)

Add(float x, int y)

Employee()

Employee(int rn , string name)

Employee(Employee emp)

Polymorphism is of 2 types

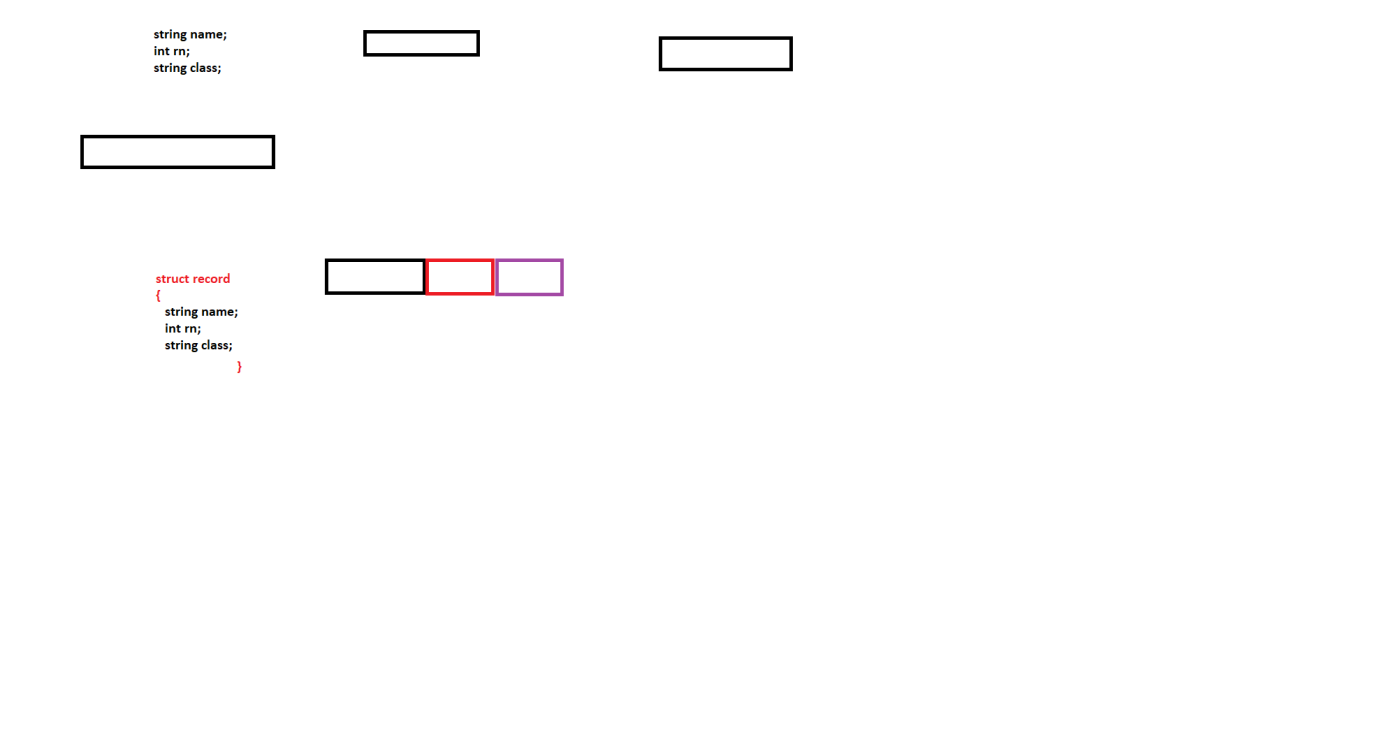
1. Compile Time : It is achieved by using Method Overloading
2. Run Time

Structure

Is also a user defined type

Structure is value type variables

Why so we need structure



Differences in struct & class

1.Struct are value type variables

Class is reference type variable

2.In Struct, we can only 2 constr , default & fully parameterized , but it does not allow to add def cons explicitly

1. Structures can not be inherited

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

struct record

{

int rn;

string name;

string classno;

public void GetDetails() { }

public void DisplayDetails() { }

}

}

To Access it

// Structure variable

record record = new record();

record.GetDetails();

record.DisplayDetails();

We declared Address variable

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

**struct Address**

**{**

**public string streetno;**

**string city;**

**string state;**

**string pincode;**

**}**

class Employee

{

// instance varibles from id to salary

int id;

string name;

string dept;

string manager;

float salary;

**Address address;**

// static variable

static string companyname;

// default constructor

public Employee()

{

}

// parameretized

public Employee(string name , string dept)

{

this.name = name;

this.dept = dept;

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

// parameterized

public Employee(int id, string name, string dept, string manager)

{

this.name = name;

this.dept = dept;

this.id = id;

this.manager = manager;

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

}

// fully parameterized

public Employee(int id, string name, string dept, string manager , float salary)

{

this.name = name;

this.dept = dept;

this.id = id;

this.manager = manager;

this.salary = salary;

}

static Employee()

{

companyname = "CTS";

}

public void GetDetails()

{

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Department");

dept = Console.ReadLine();

Console.WriteLine("Enter Manager");

manager = Console.ReadLine();

Console.WriteLine("Enter Salary");

salary = float.Parse(Console.ReadLine());

**Console.WriteLine("ENter Address");**

**Console.WriteLine("Nter Street No");**

**address.streetno = Console.ReadLine();**

}

public void DisplayDetails()

{

//Console.WriteLine("Company Name is " + companyname);

Console.WriteLine("ID is " + id);

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

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

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

Console.WriteLine("Salary is " + salary);

}

public Employee(Employee employee)

{

Console.WriteLine("Enter Id");

id = int.Parse(Console.ReadLine());

this.name = employee.name;

this.dept = employee.dept;

this.manager = employee.manager;

this.salary = employee.salary;

}

public static void DisplayCompmayName()

{

// companyname = "CTS";

Console.WriteLine("COMPANY NAME IS : "+ companyname);

}

}

}