Delegates : Pointer in C

We don’t any pointer in C#

Function Pointers in C#

Delegates : Function pointer, which means that they point to functions at run time

To use delegate , we declare it

// Declare Delegate

delegate int mydelegate (int x, int y);

Here delegate is a keyword , mydelegate is Delegate Name

It means that this delegate (mydelegate) can point to any function

Which takes 2 integers and returns 1 integer

static int add (int x, int y)

{

return x + y;

}

We want that this delegate should point to add function

mydelegate del = new mydelegate(add);

It means that del variable points to add function

using System;

namespace ConsoleApp6

{

// Declare Delegate

delegate int mydelegate (int x, int y);

class Program

{

static int add (int x, int y)

{

return x + y;

}

static int subtract(int x, int y)

{

return x - y;

}

static int product(int x, int y)

{

return x \* y;

}

static int divide(int x, int y)

{

return x / y;

}

static void Main(string[] args)

{

// Initialize delegate variable

mydelegate del = new mydelegate(add);

Console.WriteLine(del(20,10));

}

}

}

Uses of Delegates

1. Events
2. Anonymous Methods
3. Lambda Expressions
4. Call Back Methods

Delegates are of 2 types

1.Single Casting > A Delegate points to one function

2.MultiCasting > A Delegate points more than 1 function in a chain

Single Casting

using System;

namespace ConsoleApp6

{

// Declare Delegate

delegate int mydelegate (int x, int y);

class Program

{

static int add (int x, int y)

{

return x + y;

}

static int subtract(int x, int y)

{

return x - y;

}

static int product(int x, int y)

{

return x \* y;

}

static int divide(int x, int y)

{

return x / y;

}

static void Main(string[] args)

{

// Initialize delegate variable

mydelegate del = new mydelegate(add);

Console.WriteLine(del(20,10));

del = new mydelegate(subtract);

Console.WriteLine(del(20, 10));

}

}

}

MultiCast Delegate

using System;

namespace ConsoleApp6

{

// Declare Delegate

delegate void mydelegate (int x, int y);

class Program

{

static void add (int x, int y)

{

Console.WriteLine("Sum is " + (x + y));

}

static void subtract(int x, int y)

{

Console.WriteLine("Difference is " + (x - y));

}

static void product(int x, int y)

{

Console.WriteLine("Product is " + (x \* y));

}

static void divide(int x, int y)

{

Console.WriteLine("Quotient is " + (x / y));

}

static void Main(string[] args)

{

// Initialize delegate variable

mydelegate del = new mydelegate(add);

del += new mydelegate(subtract);

del += new mydelegate(product);

del += new mydelegate(divide);

del(20, 10);

}

}

}

+= is used for adding a method to that chain

del -= new mydelegate(add);

del(100, 10);

-= To Remove the method from the chain of methods

User Defined Types : Type defined by the user

Class : Reference Type

Struct

Enum

Struct & enum are value type variables

Arrays : To used to store data of same type

Structure : To used to store data of different type

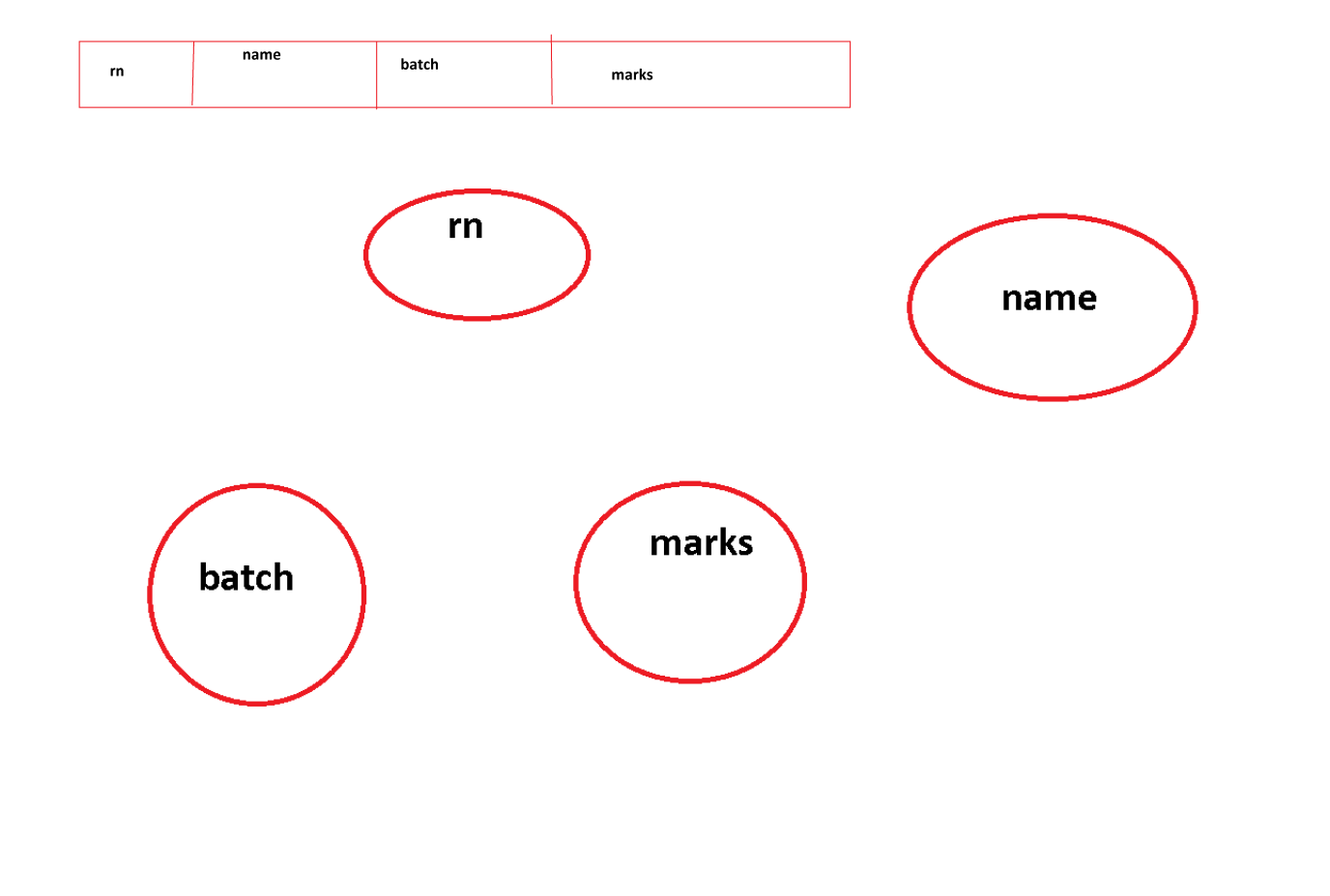
Struct records

{Int rn;

String name;

String batch

Int marks;}



DIffferences in Class & Structure

1.Class is a reference type variable

Struct is a value type variable

2. Structure have default constructor , but it does not allow to add default constructor explicitly.

We can only add 1 fully parameterized constructor

3. struct does not support Inheritance

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp6

{

struct record

{

int rn;

string name;

string batch;

int marks;

//public student() { }

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

{

this.rn = rn;

this.name = name;

this.batch = batch;

this.marks = marks;

}

public void Get() { }

public void Display() { }

}

class structDemo

{

static void Main()

{

record rec = new record();

rec.Get();

rec.Display();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp6

{

struct Address

{

public string houseno;

public string city, state, pincode;

}

struct Dob

{

public int dd, mm, yy;

}

class Student

{

int rn;

string name;

string batch;

int marks;

Address address;

Dob dob;

public void Get() {

Console.WriteLine("Enter RollNo");

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

Console.WriteLine("Enter Address");

Console.WriteLine("Enter House No");

address.houseno = Console.ReadLine();

Console.WriteLine("Enter City");

address.city = Console.ReadLine();

Console.WriteLine("Enter Date of Birth");

dob.dd = Convert.ToByte(Console.ReadLine());

}

public void Display() { }

}

class structDemo

{

static void Main()

{

Student rec = new Student();

rec.Get();

rec.Display();

}

}

}