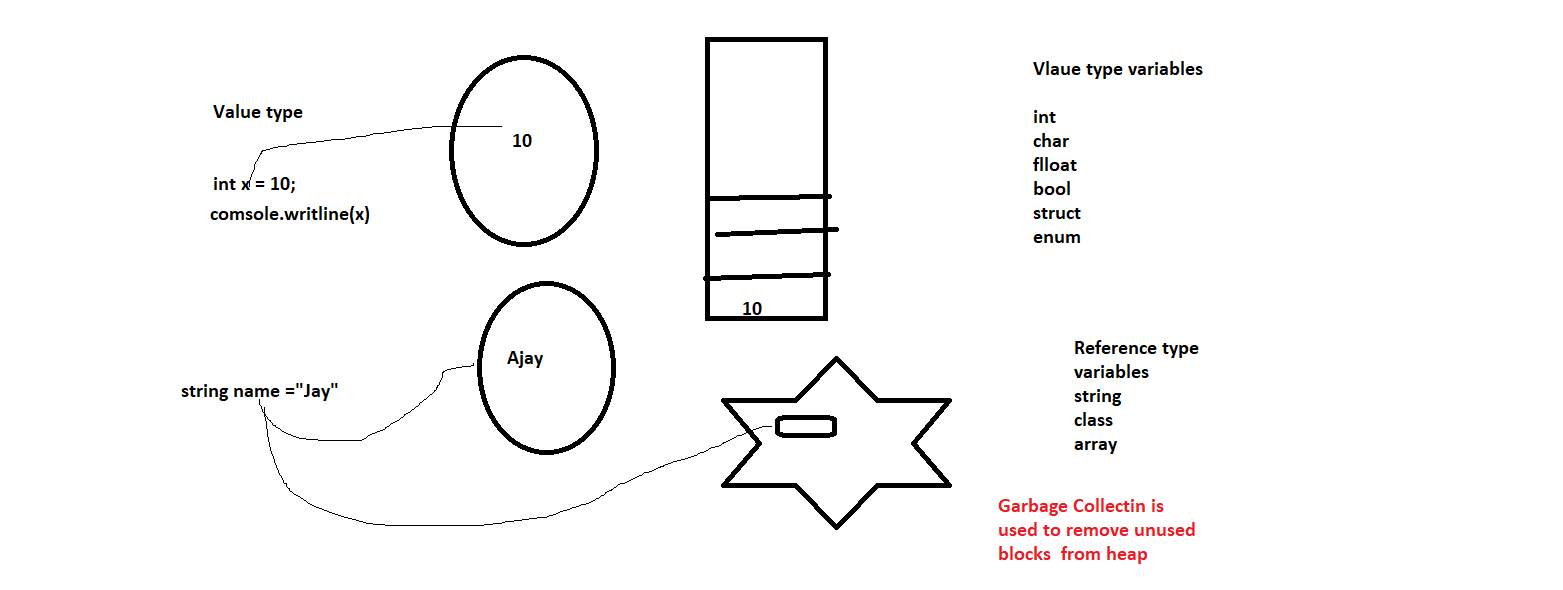
Struct class enum athers are user defined types

Value type & reference type variables



Structures are used to group logically related variables

Struct Vs class

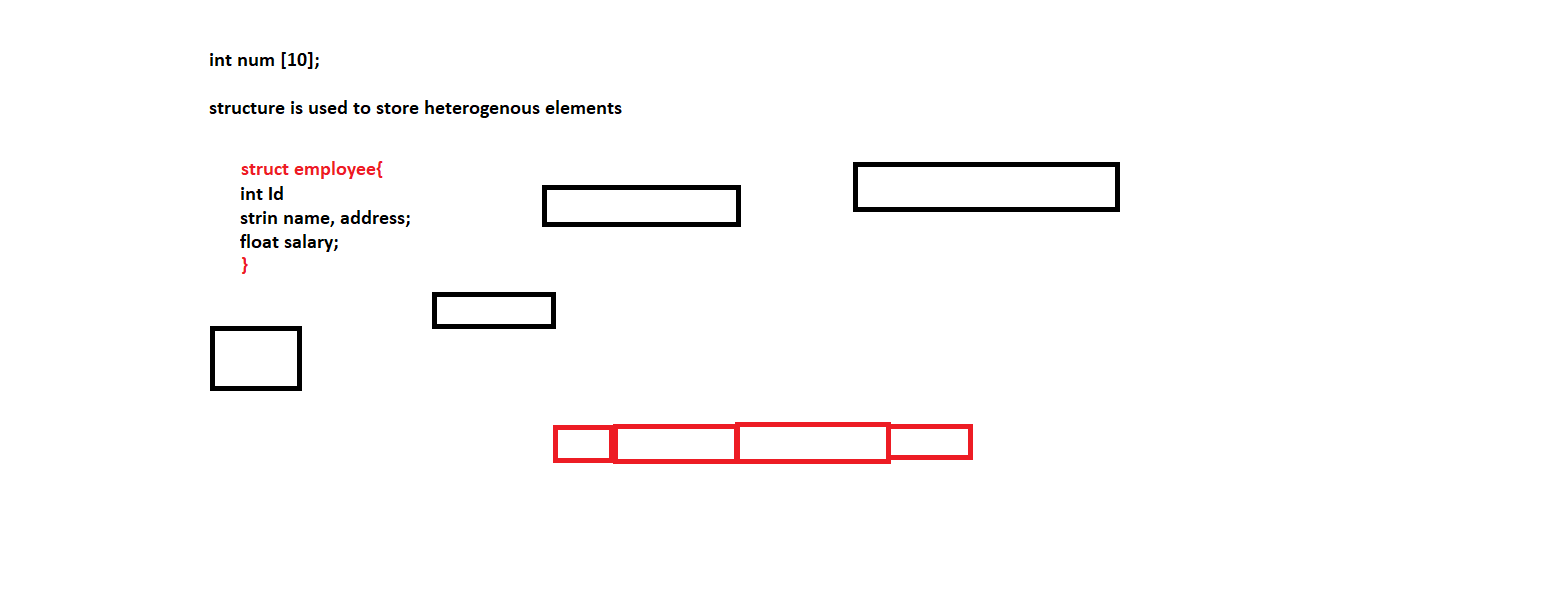
1.Structure is value type variable

Class is reference type variable

2. In structures, we cannot add def constructor explicitly

Itcan only one fully parameterized constructor

3. structures don’t support inheritance



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp7

{

struct Address

{

public string hno, city, state, pincode;

}

struct record

{

int id;

string name;

Address address;

float salary;

public record(int id, string name, string address, float salary)

{ this.id = id;

this.name = name;

this.address.hno = address;

this.salary = salary;

}

public void get() { }

public void show() { }

}

class Employee

{

int id;

string name;

Address address;

static void Main()

{

record rec = new record();

rec.get();

}

}

}

In C#, an enum (or enumeration type) is used **to assign constant names to a group of numeric integer values**. It makes constant values more readable, for example, WeekDays.

We shud use it when we have some limited options.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp7

{

enum Choice { Addition=1 , Subtraction=3, Divsion=6, Product};

class EnumDemo

{

static void Main()

{

int x,y, ch;

Console.WriteLine("ENter No1");

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

Console.WriteLine("ENter No2");

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

Console.WriteLine("ENter choice");

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

switch(ch)

{

case (int)Choice.Addition:

{

Console.WriteLine(x+y);

break;

}

case (int)Choice.Subtraction:

{

Console.WriteLine(x - y);

break;

}

case (int)Choice.Divsion:

{

Console.WriteLine(x / y);

break;

}

case (int)Choice.Product:

{

Console.WriteLine(x \* y);

break;

}

default: Console.WriteLine("Invalid choice");

break;

}

}

}

}

Interface is like a syntactical contract in which we declare all the methods

Rules

{ Rule 1: clear1 paper

Rule 2: clear1 paper

Rule 3: clear1 paper

Rule 4: clear1 paper

Rule 5: clear1 paper

}

Abstract Classes

We have to calculate area of different figures

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp7

{ class Rectangle

{

int l, w , area;

public void get()

{

Console.WriteLine("Enter Length");

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

Console.WriteLine("Enter Width");

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

}

public void CalculareArea()

{

area = l \* w;

}

public void DispalyArea()

{

Console.WriteLine("Area is " + area);

}

}

class Square

{

int s, area;

public void get()

{

Console.WriteLine("Enter side");

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

}

public void CalculareArea()

{

area = s \* s;

}

public void DispalyArea()

{

Console.WriteLine("Area is " + area);

}

}

class Traingle

{

int b , h, area;

public void get()

{

Console.WriteLine("Enter base");

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

Console.WriteLine("Enter height");

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

}

public void CalculareArea()

{

area = 1 / 2 \* (b \* h);

}

public void DispalyArea()

{

Console.WriteLine("Area is " + area);

}

}

class Calaculate\_Area

{

}

}

THIS IS NOT CORRECT APPROCAH , BECAUSE WE SAW THAT SOMANY METHPDS ARE OMMON

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp7

{

abstract class Shape

{

int side1, side2, area;

public abstract void get();

public abstract void CalcutateArea();

public void DispalyArea()

{

Console.WriteLine("Area is " + area);

}

}

class Rectangle : Shape

{

int l, w , area;

public override void get()

{

Console.WriteLine("Enter Length");

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

Console.WriteLine("Enter Width");

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

}

public override void CalcutateArea()

{

area = l \* w;

}

}

class Square : Shape

{

int s, area;

public override void get()

{

Console.WriteLine("Enter side");

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

}

public override void CalcutateArea()

{

area = s \* s;

}

}

class Traingle : Shape

{

int b , h, area;

public override void get()

{

Console.WriteLine("Enter base");

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

Console.WriteLine("Enter height");

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

}

public override void CalcutateArea()

{

area = 1 / 2 \* (b \* h);

}

}

class Calaculate\_Area

{

static void Main()

{

Rectangle rectangle = new Rectangle();

rectangle.get();

rectangle.CalcutateArea();

rectangle.DispalyArea();

}

}

}

Interface

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp71

{

interface Shape

{

void get();

void CalcutateArea();

void DispalyArea();

}

class Rectangle : Shape

{

int l, w, area;

public void get()

{

Console.WriteLine("Enter Length");

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

Console.WriteLine("Enter Width");

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

}

public void CalcutateArea()

{

area = l \* w;

}

public void DispalyArea()

{

Console.WriteLine("Area is " + area);

}

}

class Square : Shape

{

int s, area;

public void get()

{

Console.WriteLine("Enter side");

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

}

public void CalcutateArea()

{

area = s \* s;

}

public void DispalyArea()

{

Console.WriteLine("Area is " + area);

}

}

class Traingle : Shape

{

int b, h, area;

public void get()

{

Console.WriteLine("Enter base");

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

Console.WriteLine("Enter height");

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

}

public void CalcutateArea()

{

area = 1 / 2 \* (b \* h);

}

public void DispalyArea()

{

Console.WriteLine("Area is " + area);

}

}

class Calaculate\_Area

{

static void Main()

{

Rectangle rectangle = new Rectangle();

rectangle.get();

rectangle.CalcutateArea();

rectangle.DispalyArea();

}

}

}

Method Overriding > Redefining method of base class in child class

It happens only in Inheritance

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp72

{

class Student

{

int rn;

string name;

public void Get()

{

Console.WriteLine("ENter RollNo");

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

Console.WriteLine("Enter Name");

name = Console.ReadLine();

}

public void Show()

{

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

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

}

}

class ParttimeStudent : Student

{

string batchCode;

public void GetBatchCode()

{

Console.WriteLine("Enter Batch Code");

batchCode = Console.ReadLine();

}

public void ShowBatchCode()

{

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

}

}

class FulltimeStudent : Student

{

string yearofjoining;

public void Getyearofjoining()

{

Console.WriteLine("Enter Year of Joining");

yearofjoining = Console.ReadLine();

}

public void Showyearofjoining()

{

Console.WriteLine("Year of Joining is " + yearofjoining);

}

}

class Class4

{

static void Main()

{

Student student = new Student();

student.Get();

student.Show();

ParttimeStudent part = new ParttimeStudent();

part.Get();

part.GetBatchCode();

part.Show();

part.ShowBatchCode();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp72

{

class Student

{

int rn;

string name;

public void Get()

{

Console.WriteLine("ENter RollNo");

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

Console.WriteLine("Enter Name");

name = Console.ReadLine();

}

public void Show()

{

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

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

}

}

class ParttimeStudent : Student

{

string batchCode;

public void Get()

{

base.Get();

Console.WriteLine("Enter Batch Code");

batchCode = Console.ReadLine();

}

public void Show()

{

base.Show();

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

}

}

class FulltimeStudent : Student

{

string yearofjoining;

public void Get()

{

base.Get();

Console.WriteLine("Enter Year of Joining");

yearofjoining = Console.ReadLine();

}

public void Show()

{

base.Show();

Console.WriteLine("Year of Joining is " + yearofjoining);

}

}

class Class4

{

static void Main()

{

Student student = new Student();

student.Get();

student.Show();

ParttimeStudent part = new ParttimeStudent();

part.Get();

//part.GetBatchCode();

part.Show();

//part.ShowBatchCode();

}

}

}