C#

1. TO READ 2 NOS. AND CALCULATE SUM

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace add2nos

{

public class Class1

{

public static void Main()

{

int iNum1, iNum2, iSum;

Console.Write("Enter 1st no:");

iNum1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter 2nd no:");

iNum2 = Convert.ToInt32(Console.ReadLine());

iSum = iNum1 + iNum2;

Console.WriteLine("Sum={0}", iSum);

Console.Read();

}

}

}

2.TO CHECK IF A NUMBER IS ODD OR EVEN

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace evenorodd

{

public class Class1

{

public static void Main()

{

int i;

Console.Write("Enter number");

i = Convert.ToInt32(Console.ReadLine());

if (i % 2 == 0)

{

Console.Write("Entered Number is an Even Number");

}

else

{

Console.Write("Entered Number is an Odd Number");

}

Console.Read();

}

}

}

3.DISPLAY FIRST n NATURAL NUMBERS USING FOR LOOP

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_1stn\_naturalnos\_for

{

public class Class1

{

public static void Main()

{

int i, n;

Console.Write("Enter value for n");

n = Convert.ToInt32(Console.ReadLine());

for (i = 1; i <= n; i++)

{

Console.WriteLine("{0}", i);

}

Console.Read();

}

}

4.LIST MULTIPLES OF 3 AND 5 IN RANGE OF 1 TO 100

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace multiplesrange100

{

public class Class1

{

public static void Main()

{

int a, b, i;

for (i = 1; i < 100; i++)

{

a = i % 3;

b = i % 5;

if (a == 0 || b == 0)

{

Console.Write("{0}\t", i);

}

}

Console.Read();

}

}

}

5. CHECK IF CHARACTER IS VOWEL OR CONSONANT USING SWITCH

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace vowelorconsonant\_switch

{

public class Class1

{

public static void Main()

{

char ch;

Console.Write("enter character");

ch = Convert.ToChar(Console.ReadLine());

ch = Char.ToLower(ch);

switch (ch)

{

case 'a':

Console.WriteLine("Vowel");

break;

case 'e':

Console.WriteLine("Vowel");

break;

case 'i':

Console.WriteLine("Vowel");

break;

case 'o':

Console.WriteLine("Vowel");

break;

case 'u':

Console.WriteLine("Vowel");

break;

default:

Console.WriteLine("Consonant");

break;

}

Console.Read();

}

}

}

6.READ ASSOCIATE NAME AND MARKS IN 5 SUBJECTS AND DISPLAY MESSAGE BASED ON SCORE

(80 AND>=DISTINCTION, 60-80:FIRST CLASS, 50-60:SECOND CLASS, <50:FAIL)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace marksheet

{

public class Class1

{

public static void Main()

{

int Num1, Num2, Num3, Num4, Num5, total;

float percentage;

string name;

Console.Write("Enter associate name");

name = Console.ReadLine();

Console.Write("Enter sub1 marks");

Num1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter sub2 marks");

Num2 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter sub3 marks");

Num3 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter sub4 marks");

Num4 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter sub5 marks");

Num5 = Convert.ToInt32(Console.ReadLine());

total = Num1 + Num2 + Num3 + Num4 + Num5;

percentage = total / 6;

Console.WriteLine("name:{0}", name);

if (percentage >= 80)

{

Console.WriteLine("DISTINCTION");

}

else if (percentage >= 60 && percentage < 80)

{

Console.WriteLine("FIRST CLASS");

}

else if (percentage > 50 && percentage < 60)

{

Console.WriteLine("SECOND CLASS");

}

else if (percentage < 50)

{

Console.WriteLine("FAIL");

}

Console.Read();

}

}

}

7.MENU DRIVEN SIMPLE CALCULATOR

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace menudrivencalculator

{

public class Class1

{

public static void Main()

{

int num1, num2, opt;

Console.WriteLine("A menu driven program for a simple calculator:\n");

do

{

Console.WriteLine("1-Addition.\n2-Substraction.\n3-Multiplication.\n4-Division.\n5-Exit.\n");

Console.WriteLine("\nInput your choice :");

opt = Convert.ToInt32(Console.ReadLine());

if (opt > 5)

{

Console.WriteLine("Input correct option\n");

break;

}

Console.WriteLine("Enter the first Integer :");

num1 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter the second Integer :");

num2 = Convert.ToInt32(Console.ReadLine());

switch (opt)

{

case 1:

Console.WriteLine("The Addition of {0} and {1} is: {2}\n", num1, num2, num1 + num2);

break;

case 2:

Console.WriteLine("The Substraction of {0} and {1} is: {2}\n", num1, num2, num1 - num2);

break;

case 3:

Console.WriteLine("The Multiplication of {0} and {1} is: {2}\n", num1, num2, num1 \* num2);

break;

case 4:

if (num2 == 0)

{

Console.WriteLine("The second integer is zero. Devide by zero.\n");

}

else

{

Console.WriteLine("The Division of {0} and {1} is : {2}\n", num1, num2, num1 / num2);

}

break;

case 5:

break;

default:

Console.WriteLine("Input correct option\n");

break;

}

} while (opt != 5);

Console.Read();

}

}

}

8. MATRIX ADDITION

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_2darray

{

class Program

{

public static void Main()

{

int i, j, m, n;

int[,] summatrix = new int[2, 2];

int[,] matrix1 = { { 1, 2 }, { 3, 4 } }; //TYPE1

int[,] matrix2 = new int[2, 2] { { 2, 2 }, { 4, 3 } }; //TYPE2

Console.WriteLine("\nno of rows? ");

m = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("\nno of columns? ");

n = Convert.ToInt32(Console.ReadLine());

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

matrix1[i, j] = Convert.ToInt16(Console.ReadLine());

}

}

Console.WriteLine("\nMatrix 1 : "); //type1

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

Console.Write(matrix1[i, j] + "\t");

}

Console.WriteLine();

}

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

matrix2[i, j] = Convert.ToInt16(Console.ReadLine());

}

}

Console.WriteLine("\nMatrix 2 : "); //type2

for (i = 0; i < matrix2.GetLength(0); i++)

{

for (j = 0; j < matrix2.GetLength(1); j++)

{

Console.Write(matrix2[i, j] + "\t");

}

Console.WriteLine();

}

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

summatrix[i, j] = matrix1[i, j] + matrix2[i, j];

}

}

Console.WriteLine("\nsum of 2 matrices : ");

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

Console.Write(summatrix[i, j] + "\t");

}

Console.WriteLine();

}

Console.Read();

}

}

}

9. 1D CHAR ARRAY

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_1dchararray

{

class Program

{

static void Main(string[] args)

{

char[] grade = { 'A', 'B', 'C', 'D' };

Console.WriteLine("ARRAY ELEMENTS:");

foreach(char c in grade)

Console.WriteLine(c);

Console.WriteLine("NUMBER OF CHARACTERS IN ARRAY:");

Console.WriteLine(grade.Count());

Console.Read();

}

}

}

10. 2D ARRAY/ matrix addition

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_2darray

{

class Program

{

public static void Main()

{

int i, j;

int[,] summatrix = new int[2, 2];

int[,] matrix1 = { { 1, 2 }, { 3, 4} }; //TYPE1

int[,] matrix2 = new int[2, 2] { { 2, 2 }, { 4, 3 } }; //TYPE2

Console.WriteLine("\nMatrix 1 : ");

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

Console.Write(matrix1[i, j] + "\t");

}

Console.WriteLine();

}

Console.WriteLine("\nMatrix 2 : ");

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

Console.Write(matrix2[i, j] + "\t");

}

Console.WriteLine();

}

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

summatrix[i, j] = matrix1[i, j] + matrix2[i, j];

}

}

Console.WriteLine("\nsum of 2 matrices : ");

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

Console.Write(summatrix[i, j] + "\t");

}

Console.WriteLine();

}

Console.Read();

}

}

}

11. ARRAY(FUNTIONS)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace arrays

{

public class Program

{

public static void Main()

{

int[] mark; // declare numbers as an int array of any size

mark = new int[4]; // numbers is a 10-element array

mark[0] = 99;

mark[1] = 88;

mark[2] = 77;

mark[3] = 45;

Console.Write("Array1 elements:\n");

Console.WriteLine(mark[0]);

Console.WriteLine(mark[1]);

Console.WriteLine(mark[2]);

Console.WriteLine(mark[3]);

int[] mark1 = new int[2];

mark1[0] = 99;

mark1[1] = 88;

Console.Write("Array2 elements:\n");

Console.WriteLine(mark1[0]);

Console.WriteLine(mark1[1]);

int[] mark2 = { 55, 77, 22, 44, 98 };

Console.Write("Array3 elements:\n");

for (int i = 0; i < mark2.Length; i++)

{

Console.WriteLine("{0} ", mark2[i]);

}

Array.Sort(mark);

Console.Write("Sorted array is:\n");

for (int i = 0; i < mark.Length; i++)

{

Console.WriteLine("{0} ", mark[i]);

}

int pos, pos1;

pos = Array.BinarySearch(mark, 77);

pos1 = Array.BinarySearch(mark, 95);

Console.WriteLine("77 and 95 are found respectively as position {0}{1}", pos, pos1);

Array.Reverse(mark);

Console.Write("Reversed array1 is:\n");

for (int i = 0; i < mark.Length; i++)

{

Console.WriteLine("{0} ", mark[i]);

}

int[] markcopy = new int[4];

Array.Copy(mark, markcopy, 4);

Console.Write("copy of array1 is:\n");

for (int i = 0; i < markcopy.Length; i++)

{

Console.WriteLine("{0} ", markcopy[i]);

}

Array.Clear(mark1, 0, 2);

Console.Write("array2 after deleting 1st 2 elements is:\n");

for (int i = 0; i < mark1.Length; i++)

{

Console.WriteLine("{0} ", mark1[i]);

}

int[] clonemark2 = (int[])mark2.Clone();

Console.Write("clone of array3:\n");

for (int i = 0; i < mark2.Length; i++)

{

Console.WriteLine("{0} ", clonemark2[i]);

}

Console.Read();

}

}

}

12. MATRIX MULTIPLICATION

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace matrix\_multi

{

class Program

{

static void Main(string[] args)

{

int i, j,m,n;

Console.WriteLine("Enter the Number of Rows and Columns : ");

m = Convert.ToInt32(Console.ReadLine());

n = Convert.ToInt32(Console.ReadLine());

int[,] a = new int[m, n];

Console.WriteLine("Enter the First Matrix");

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

a[i, j] = int.Parse(Console.ReadLine());

}

}

Console.WriteLine("First matrix is:");

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

Console.Write(a[i, j] + "\t");

}

Console.WriteLine();

}

int[,] b = new int[m, n];

Console.WriteLine("Enter the Second Matrix");

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

b[i, j] = int.Parse(Console.ReadLine());

}

}

Console.WriteLine("Second Matrix is :");

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

Console.Write(b[i, j] + "\t");

}

Console.WriteLine();

}

Console.WriteLine("Matrix Multiplication is :");

int[,] c = new int[m, n];

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

c[i, j] = 0;

for (int k = 0; k < 2; k++)

{

c[i, j] += a[i, k] \* b[k, j];

}

}

}

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

Console.Write(c[i, j] + "\t");

}

Console.WriteLine();

}

Console.ReadKey();

}

}

}

13. METHODS- 1

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace methods\_1 //1 class- using 1 method

{

class Program

{ //PASSING PARAMETERS BY REFERENCE

int sum(int a,int b) //int sum(ref int a,ref int b)

{

int res = a + b;

return res;

}

static void Main(string[] args)

{

Program obj = new Program(); //int a=77, b=66;

int value = obj.sum(10, 10); //obj.sum(ref a,ref b);

Console.WriteLine("sum is :{0}", value);

Console.Read();

}

}

}

14. METHODS-2 /FIND MAX VAL

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace methods2 //2 classes- 1 method in class1 called from 2nd class

{

class Class1

{

public int FindMax()

{

int n1 = 55, n2 = 88;

int maxval;

if (n1>n2)

maxval=n1;

else

maxval=n2;

return maxval;

}

}

class Class2

{

static void Main(string[] args)

{

Class1 obj = new Class1();

int res = obj.FindMax();

Console.WriteLine("max is:{0}", res);

Console.Read();

}

}

}

15. METHODS 3 – USING OUTPUT PARAMETERS

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace methods3\_\_output\_parameters //using output parameters: return 2 values

{

class Program

{

public void getval(out int a)

{

int temp = 5;

a = temp;

}

static void Main(string[] args)

{

int x = 50;

Program obj = new Program();

Console.WriteLine("before calling getval value of x :{0}", x);

obj.getval(out x);

Console.WriteLine("after calling getval value of x :{0}", x);

Console.Read();

}

}

}

16. METHODS 4- USING PARAMS PARAMETER

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PARAMS\_PARAMETER

{

class Program

{

static void func1(params int[] arr1)

{

Console.WriteLine("length of arr1 : {0}", arr1.Length);

foreach (int i in arr1)

Console.WriteLine("{0}", i);

Console.WriteLine();

}

static void Main(string[] args)

{

int[] arr2 = { 4, 5, 3, 2 };

func1(arr2);

func1(2, 5, 3, 8, 9);

func1();

Console.Read();

}

}

}

17. JAGGED ARRAY

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace jagged\_arary

{

class Program

{

static void Main(string[] args)

{

int[][] jaggarr = new int[3][];

//method 1

jaggarr[0] = new int[3];

jaggarr[1] = new int[2];

jaggarr[2] = new int[2];

jaggarr[0]=new int[] {1,2,3};

jaggarr[1] = new int[] { 4,5 };

jaggarr[2] = new int[] { 6,7 };

//method 2

// int[][] jaggarr1 = new int[][]

// {

// new int[]{11,12,13},

// new int[]{14,15},

// new int[]{16,17}

// };

//method 3

//int[][] jaggarr2 =

//{

// new int[]{11,12,13},

// new int[]{14,15},

// new int[]{16,17}

// };

//multidimentional array

// int[][,] muljag = new int[2][,]

// {

// new int[,] {{1,2,3},{4,5,6}},

// new int[,] {{7,8,9},{10,11,12}},

// };

for (int i = 0; i < jaggarr.Length; i++)

{

System.Console.Write("Element({0}): ", i);

for (int j = 0; j < jaggarr[i].Length; j++)

{

System.Console.Write("{0}{1}", jaggarr[i][j], j == (jaggarr[i].Length - 1) ? "" : " ");

}

System.Console.WriteLine();

}

Console.WriteLine("length is:{0}",jaggarr.Length);

Console.Read();

}

}

}

18 INHERITANCE/ PROTECTED ACCESS SPECIFIER

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace protected\_derived\_inherit

{

class Shape

{

protected int length;

protected int breadth;

protected int side;

public void setlength(int l)

{

length=l;

}

public void setbreadth(int b)

{

breadth=b;

}

public void setside(int s)

{

side=s;

}

}

class rectangle: Shape

{

public int getArea()

{

return(length\*breadth);

}

}

class square : Shape

{

public int getArea2()

{

return(side\*4);

}

}

class Result

{

static void Main(string[] args)

{

rectangle obj1 = new rectangle();

obj1.setlength(10);

obj1.setbreadth(20);

int area1 = obj1.getArea();

Console.WriteLine("area of rectangle is {0}", area1);

square obj2 = new square();

obj2.setside(5);

int area2 = obj2.getArea2();

Console.WriteLine("area of square is {0}", area2);

Console.Read();

}

}

}

19 ACCESS SPECIFIER : PUBLIC PRIVATE

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace access\_specifiers

{

class Class1

{

public int sum(int a,int b)

{

int total;

total=a+b;

return total;

}

private int max(int a,int b)

{

int val;

if (a > b)

val=a;

else val=b;

return val;

}

}

class Claass2

{

static void Main(string[] args)

{

Class1 obj = new Class1();

int res=obj.sum(10, 20);

Console.WriteLine("result is:{0}", res);

int max = obj.max(10, 20);

if(max==null)

Console.WriteLine("due to protection level error raised");

Console.Read();

}

}

}

20. OBJECT ORIENTED PROGRAMING USING C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CLASSES

{

class Person

{

private string myName;

private int myAge;

private char mySex;

public Person() //default constructor

{

}

public Person(string name, int age,char sex) //parameterised constructor

{

this.myName = name;

this.myAge = age;

this.mySex = sex;

}

public string Name

{get //ACCESSORS

{

return myName;

}

set

{

myName = value;

}

}

public int Age

{

get //ACCESSORS

{

return myAge;

}

set

{

myAge = value;

}

}

public char Sex

{

get //ACCESSORS

{

return mySex;

}

set

{

mySex = value;

}

}

public override string ToString()

{

return "Name=" + Name + ",Age=" + Age + ",Sex=" + Sex;

}

public override bool Equals(object o)

{

if ((myName == ((Person)o).myName) && (myAge == ((Person)o).myAge) && (mySex == ((Person)o).mySex))

return true;

else return false;

}

public static void Main()

{

Person p1 = new Person();

p1.Name = "Joe";

p1.Age = 99;

p1.Sex = 'm';

Console.WriteLine("person Details: {0}", p1);

Person p2 = new Person();

p2.Name = "Jane";

p2.Age = 31;

p2.Sex = 'f';

Console.WriteLine("person Details: {0}", p2);

Person p3 = new Person();

p3.Name = "Jane";

p3.Age = 31;

p3.Sex = 'f';

Console.WriteLine("person Details: {0}", p3);

bool same = p2.Equals(p3);

Console.WriteLine("same person" + same);

Person p4 = new Person("Otto", 42, 'm');

Console.WriteLine("person details" + p4.myName);

Console.WriteLine("person details" + p4.mySex);

//arrays of persons

Person[] pmany = new Person[2];

pmany[0] = new Person("Carl", 23, 'm');

pmany[1] = new Person("Ola", 7, 'f');

Console.WriteLine("name of person[0]" + pmany[0]);

Console.WriteLine("name of person[0]" + pmany[0].myName);

Console.WriteLine("sex of person[1]" + pmany[1].mySex);

Console.Read();

}

}

}

21. FUNCTION OVERLOADING POLYMORPHISM

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace funtion\_overloading\_poly

{

class Program

{

void print(int i)

{

Console.WriteLine("value is {0}", i);

}

void print(double d)

{

Console.WriteLine("value is {0}", d);

}

void print(string s)

{

Console.WriteLine("value is {0}", s);

}

static void Main(string[] args)

{

Program obj = new Program();

obj.print(10);

obj.print(76.55);

obj.print("hello");

Console.Read();

}

}

}

22 OPERATOE OVERLOADING : POLY

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OPERATOR\_OVERLOADING\_POLY

{

class Box

{

private double length;

private double breadth;

private double height;

public double volume()

{

return (length\*breadth\*height);

}

public void setlength(double l)

{

length=l;

}

public void setbreadth(double b)

{

breadth=b;

}

public void setheight(double h)

{

height=h;

}

//overload + operatatoe for 2 objects

public static Box operator +(Box b,Box c)

{

Box box = new Box();

box.length=b.length+c.length;

box.breadth=b.breadth+c.breadth;

box.height=b.height+c.height;

return box;

}

}

class Test

{

static void Main(string[] args)

{

Box box1 = new Box();

Box box2 = new Box();

Box box3 = new Box();

double volume =0.0;

box1.setlength(10.56);

box1.setbreadth(13.44);

box1.setheight(15.77);

volume = box1.volume();

Console.WriteLine("volume box1 {0}", volume);

box2.setlength(12.56);

box2.setbreadth(14.44);

box2.setheight(16.77);

volume = box2.volume();

Console.WriteLine("volume box2 {0}", volume);

box3 = box1 + box2;

volume = box3.volume();

Console.WriteLine("volume box3 {0}", volume);

Console.ReadKey();

}

}

}

23. DYNAMIC POLYMORPHISM- ABSTRACT CLASS- OVERRIDE

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DYNAMIC\_POLY

{

abstract class Shape

{

public abstract int area();

}

class Rectangle : Shape

{

private int length;

private int width;

public Rectangle(int a = 0, int b = 0)

{

length = a;

width = b;

}

public override int area()

{

Console.WriteLine("Rectangle class area :");

return (width \* length);

}

}

class RectangleTester

{

static void Main(string[] args)

{

Rectangle r = new Rectangle(10, 7);

double a = r.area();

Console.WriteLine("Area: {0}", a);

Console.ReadKey();

}

}

}

24. POLYMORPHISM – CLASSROOM

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace poly

{

class Vehicle

{

private int weight;

private int topSpeed;

private double price;

public Vehicle()

{

}

public Vehicle(int wt, int speed, double vPrice)

{

weight=wt;

topSpeed=speed;

price=vPrice;

}

public int getWeight()

{

return weight;

}

public int gettopSpeed()

{

return topSpeed;

}

public double getPrice()

{

return price;

}

public virtual void print()

{

Console.WriteLine("weight {0} kg", weight);

Console.WriteLine("topspeed {0} km/hr", topSpeed);

Console.WriteLine("price {0} dollar", price);

}

}

class Car: Vehicle{

private int numberCylinders;

private int horsepower;

private int displacement;

public Car()

{

}

public Car(int wt,int speed, double price,int numCylinders,int horsepowers, int displacemt) : base(wt,speed,price)

{

numberCylinders=numCylinders;

horsepower=horsepowers;

displacement=displacemt;

}

public int getNumberCylinders()

{

return numberCylinders;

}

public int getHorsePower()

{

return horsepower;

}

public int getDisplacement()

{

return displacement;

}

public override void print()

{

base.print();

Console.WriteLine("cylinders {0}", numberCylinders);

Console.WriteLine("horsepower {0}", horsepower);

Console.WriteLine("displacement {0}", displacement);

}

}

class myCar

{

static void Main(string[] args)

{

Vehicle obj = new Vehicle(15000, 120, 30000.00);

Console.WriteLine("A vehicle");

obj.print();

Car objcar = new Car(3500, 100, 12000, 6, 120, 300);

Console.WriteLine("A car");

objcar.print();

Console.WriteLine(" ");

Console.ReadKey();

}

}

}

**25. INHERITANCE- CLASSROOM**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace inheritance

{

class DrawingObject

{

public virtual void Draw()

{

Console.WriteLine("Drawing object.....");

}

}

public class Line: DrawingObject

{

public override void Draw()

{

Console.WriteLine("drawing a line....");

}

}

public class Circle: DrawingObject

{

public override void Draw()

{

Console.WriteLine("drawing a circle....");

}

}

public class Square: DrawingObject

{

public override void Draw()

{

Console.WriteLine("drawing a square....");

}

}

public class DrawDemo

{

public static int Main()

{

DrawingObject[] obj = new DrawingObject[4];

obj[0]=new Line();

obj[1]=new Circle();

obj[2]=new Square();

obj[3]=new DrawingObject();

foreach(DrawingObject o in obj)

{

o.Draw();

}

return 0;

}

}

}

26. INTERFACE – CLASROOM

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace @interface

{

interface IErase

{

void Erase();

}

abstract class Shape

{

protected int NoOfPoints;

public Shape()

{

Console.WriteLine("Constructor of base class");

}

public virtual void M1()

{

Console.WriteLine("Implemented M1()");

}

public abstract void Draw();

public abstract void Move();

public abstract int Area(int a);

}

sealed class Square : Shape, IErase

{

public void Erase()

{

Console.WriteLine("erase the shape");

}

public Square(int nopt)

{

this.NoOfPoints = nopt;

}

public override void Draw()

{

Console.WriteLine("drawing a square with" + this.NoOfPoints.ToString());

}

public override void Move()

{

Console.WriteLine("moving a square");

}

public override int Area(int s)

{

return s\*s;

//return 8;

}

}

sealed class Pentagon : Shape, IErase

{

public void Erase()

{

Console.WriteLine("erase the shape");

}

public Pentagon(int nopt)

{

this.NoOfPoints = nopt;

}

public override void Draw()

{

Console.WriteLine("drawing a Pentagon of points" + NoOfPoints.ToString());

}

public override void Move()

{

Console.WriteLine("moving a Pentagon");

}

public override int Area(int s)

{

return (Convert.ToInt32(2.6 \* s));

// return 10;

}

}

class Program

{

static void Main(string[] args)

{

Square objSquare = new Square(4);

Pentagon objPentagon=new Pentagon(5);

objSquare.Draw();

objSquare.Move();

objSquare.Erase();

Console.WriteLine("Area {0}",objSquare.Area(4));

objPentagon.Draw();

objPentagon.Move();

objSquare.Erase();

Console.WriteLine("Area {0}",objPentagon.Area(5));

Console.ReadKey();

}

}

}

27 PROPERTY - CLASSROOM

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace property

{

class Program

{

int age;

Program()

{ }

Program(int age)

{

this.age = age;

}

public int Age

{

get

{

return age;

}

set

{

if (value > 60)

{

age = 0;

Console.WriteLine("please enter a valid age<60");

}

else

age = value;

}

}

public override string ToString()

{

return "Age=" + Age;

}

static void Main(string[] args)

{

Program obj = new Program();

Console.WriteLine("Enter Person Age");

obj.Age = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Person details {0}", obj);

Console.ReadKey();

}

}

}

28 STATIC CLASS- CLASS ROOM

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace staticclass

{

static class staticClass

{

//int n1=11;

static int n1 = 14;

public static int sum(int m1, int m2)

{

return m1 + m2;

}

static void Main(string[] args)

{

//staticClass obj=new staticClass();

Console.WriteLine(n1);

Console.WriteLine(staticClass.sum(12, 23));

Console.ReadKey();

}

}

}

29- SWITCH WITH GOTO AND RETURN

using System;

public class Program

{

public int Calculate()

{

int x, y;

label:

Console.WriteLine("enter 1st num");

x = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter 2nd num");

y = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter 1-addition,2-substraction");

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

switch (option)

{

case 1:

return (x + y);

case 2:

return (x - y);

default:

Console.WriteLine("invalid");

goto label;

}

}

public static void Main()

{

Program p1 = new Program();

int result = p1.Calculate();

Console.WriteLine("Required result is={0}", result);

Console.Read();

}

}

30. PRIVATE CONSTRUCTOR

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PRIVATE\_CONSTRUCTOR

{

public class Counter

{

private Counter() {}

public static int currentCount;

public static int incrementCounter()

{

return ++currentCount;

}

}

class Test

{

static void Main(string[] args)

{

Counter.currentCount = 100;

Counter.incrementCounter();

Console.WriteLine("new count is {0}", Counter.currentCount);

Console.ReadKey();

}

}

}

31.COPY CONSTRUCTOR

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace COPY\_CONSTRUCTOR

{

class Sample

{

public string param1, param2;

public Sample(string x, string y)

{

param1 = x;

param2 = y;

}

public Sample(Sample obj) // Copy Constructor

{

param1 = obj.param1;

param2 = obj.param2;

}

}

class Program

{

static void Main(string[] args)

{

Sample obj = new Sample("Welcome", "c#"); // Create instance to class Sample

Sample obj1 = new Sample(obj); // Here obj details will copied to obj1

Console.WriteLine(obj1.param1 + " to " + obj1.param2);

Console.ReadLine();

}

}

}

32. INTERFACES: EG 1- NO DEFINITION

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace interfaces\_eg\_1\_no\_defn

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("hello");

Console.ReadKey();

}

}

interface abc

{

}

}

33: INTERFACES: EG 2- ITERFACE WITH FIELD DECLARED

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace int\_eg\_2\_declare\_field

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("hello");

}

}

interface abc

{

int x;

set;

get;

}

}

//error interface cannot contain fields

34- INTERFACES: EG 3- INTERFACE WITH METHOD & ITS DEFINITION

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace int\_eg\_3\_method\_defn

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("hello");

}

}

interface abc

{

void xyz()

{

Console.WriteLine("in xyz");

}

}

}

//error interface members cannot have definitions

35- INTERFACES: EG 4: INTERFACE WITH METHOD DECLARATION ONLY

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace int\_eg\_4\_method\_declaration

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("hello");

Console.ReadKey();

}

}

interface abc

{

void xyz();

}

}

36- INTERFACES: EG 5- INTERFACE CLASS WITHOUT MEHOD DEFINITION

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace int\_eg\_5\_class\_and\_int

{

class Demo: abc

{

static void Main()

{

Console.WriteLine("hi");

Console.ReadKey();

}

}

interface abc

{

void xyz();

}

}

//error if class implements interface, it should have definition for

//methods declared in interface

37- INTERFACES : EG 6- USING STATIC METHOD IN INTERFACE

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace int\_eg\_5\_class\_and\_int

{

class Demo : abc

{

public void xyz()

{

Console.WriteLine("in xyz");

}

static void Main()

{

Demo obj = new Demo();

obj.xyz();

Console.WriteLine("hi");

Console.ReadKey();

}

}

interface abc

{

static void xyz();

}

}

//error static modifier is not valid

38- INTERFACE EG 7: INTERFACE USING PUBLIC MODIFIER

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace int\_eg\_5\_class\_and\_int

{

class Demo : Iabc

{

static void Main()

{

Console.WriteLine("hi");

Console.ReadKey();

}

public void xyz()

{

Console.WriteLine("in xyz");

}

}

interface Iabc

{

public void xyz();

}

}

//error public modifier is not valid

39. ACCESS SPECIFIERS

* PART A- ACCESS SPECIFIER PROJECT – 3 CLASSES
* PART B- EXTERNALNAMESPACE PROJECT – 1 CLASS
* PART C- VB DOT NET PROJECT

PART A

CLASS 1

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AccessModifiers

{

public class Class1

{

public int x = 999;

}

}

CLASS 2

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AccessModifiers

{

class Class2

{

}

}

CLASS 3:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

// User defined namespaces

using externalNamespace;

namespace AccessModifiers

{

class Program : baseclass (assignment://externalNamespace.Program)

{

static void Main(string[] args)

{

int num = 100;

Console.WriteLine(num);

Class1 obj = new Class1();

Console.WriteLine("hello");

Console.WriteLine(n1);

externalNamespace.Program obj1 = new externalNamespace.Program();

obj1.display();

//assign Console.WriteLine("externalNamespace.Program.m");

//assign Program probj = new Program();

//assign probj.display();

//assign Console.WriteLine(externalNamespace.Program.m);

Console.WriteLine(obj.x);

Console.ReadKey();

}

}

class baseclass

{

public static int n1 = 10;

}

}

PART 2:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace externalNamespace

{

public class Program

{

public static int m=26;

static void Main(string[] args)

{

}

public void display() //PROTECTED INTERNAL (ASSIGNMENT)

{

Console.WriteLine("Display of external namespace ");

}

}

}

PART 3:

Module Module1

Public i As Integer = 1

Sub Main()

End Sub

End Module

40. STRUCT

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace @struct

{

public struct CoOrds

{

public int x, y;

public CoOrds(int p1, int p2)

{

x = p1;

y = p2;

}

}

public class Class1

{

static void Main()

{

// Initialize:

CoOrds coords1 = new CoOrds();

CoOrds coords2 = new CoOrds(10, 10);

// Display results:

Console.Write("CoOrds 1: ");

Console.WriteLine("x = {0}, y = {1}", coords1.x, coords1.y);

Console.Write("CoOrds 2: ");

Console.WriteLine("x = {0}, y = {1}", coords2.x, coords2.y);

// Keep the console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

}

41. STRING ARRAY

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_1dchararray

{

class Class2

{

static void Main(String[] args)

{

int i;

string[] name = new string[5];

Console.WriteLine("ENTER NAMES");

for (i = 0; i < name.Length; i++)

{

name[i]=Console.ReadLine();

}

Console.WriteLine("NAMES LIST1");

for (i = 0; i < name.Length; i++)

{

Console.WriteLine(name[i]);

}

string[] names = { "najma", "ankit", "adarsh" };

Console.WriteLine("NAMES LIST2");

for (i = 0; i < names.Length; i++)

{

Console.WriteLine(names[i]);

}

Console.Read();

}

}

}

42. STRING AND STRING BUILDER

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace string\_and\_strbuilder

{

class Program

{

static void Main(string[] args)

{

string s = "Carl";

StringBuilder b1 = new StringBuilder(s.Length + 12);

StringBuilder b5 = new StringBuilder();

Console.WriteLine(b5.MaxCapacity);

Console.WriteLine(b1.Capacity);

b1.Append("Carl");

b1.Append("uli");

b1.AppendLine();

Console.WriteLine(b1);

b1.Remove(3, 2);

Console.WriteLine(b1);

StringBuilder b2 = new StringBuilder("A.C");

b2.Insert(2, "B.");

Console.WriteLine(b2);

b2.Replace(',',':');

Console.WriteLine(b2);

StringBuilder b3 = new StringBuilder("stringbuilder");

b3.Remove(4, 4);

Console.WriteLine(b3);

Console.ReadKey();

}

}

}

43.STATIC CONSTRUCTOR

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace staticcons

{

class Program

{

static void Main(string[] args)

{

staticConstructor obj1 = new staticConstructor();

staticConstructor obj2 = new staticConstructor();

Console.ReadKey();

}

class staticConstructor

{

static int locationViewer;

static staticConstructor()

{

Console.WriteLine("static constructor called");

locationViewer = 10;

}

public staticConstructor()

{

Console.WriteLine("constructor called");

}

}

}

}

44. SEALED CLASS

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SEALED\_CLASS

{

class Num1

{

public int x;

public int y;

public int max(int a,int b)

{

int maxval=a > b ? a : b;

return maxval;

}

}

sealed class Num2:Num1

{

//protected int y;

public int min(int p,int q)

{

x = p; y = q;

int minval=x<y?x:y;

return minval;

}

//protected void show();

}

/\* class demo : Num2

{

public void show()

{

Console.WriteLine("data {0}", y); // error cannot derive from sealed class

}

}\*/

class result

{

static void Main()

{

Num2 obj = new Num2();

int a = obj.max(50, 60);

int b = obj.min(23,66);

//obj.show();

Console.WriteLine("max,min {0} {1}", a, b);

Console.ReadKey();

}

}

}

45. USE OF PROTECTED ( INHERITANCE)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace protected\_derived\_inherit

{

class Shape

{

protected int length;

protected int breadth;

protected int side;

public void setlength(int l)

{

length=l;

}

public void setbreadth(int b)

{

breadth=b;

}

public void setside(int s)

{

side=s;

}

}

class rectangle: Shape

{

public int getArea()

{

return(length\*breadth);

}

}

class square : Shape

{

public int getArea2()

{

return(side\*4);

}

}

class Result

{

static void Main(string[] args)

{

rectangle obj1 = new rectangle();

obj1.setlength(10);

obj1.setbreadth(20);

int area1 = obj1.getArea();

Console.WriteLine("area of rectangle is {0}", area1);

square obj2 = new square();

obj2.setside(5);

int area2 = obj2.getArea2();

Console.WriteLine("area of square is {0}", area2);

Console.Read();

}

}

}

46: INHERITANCE : VEHICLE , SIR’S EXAMPLE

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace poly

{

class Vehicle

{

private int weight;

private int topSpeed;

private double price;

public Vehicle()

{

}

public Vehicle(int wt, int speed, double vPrice)

{

weight=wt;

topSpeed=speed;

price=vPrice;

}

public int getWeight()

{

return weight;

}

public int gettopSpeed()

{

return topSpeed;

}

public double getPrice()

{

return price;

}

public virtual void print()

{

Console.WriteLine("weight {0} kg", weight);

Console.WriteLine("topspeed {0} km/hr", topSpeed);

Console.WriteLine("price {0} dollar", price);

}

}

class Car: Vehicle{

private int numberCylinders;

private int horsepower;

private int displacement;

public Car()

{

}

public Car(int wt,int speed, double price,int numCylinders,int horsepowers, int displacemt) : base(wt,speed,price)

{

numberCylinders=numCylinders;

horsepower=horsepowers;

displacement=displacemt;

}

public int getNumberCylinders()

{

return numberCylinders;

}

public int getHorsePower()

{

return horsepower;

}

public int getDisplacement()

{

return displacement;

}

public override void print()

{

base.print();

Console.WriteLine("cylinders {0}", numberCylinders);

Console.WriteLine("horsepower {0}", horsepower);

Console.WriteLine("displacement {0}", displacement);

}

}

class myCar

{

static void Main(string[] args)

{

Vehicle obj = new Vehicle(15000, 120, 30000.00);

Console.WriteLine("A vehicle");

obj.print();

Car objcar = new Car(3500, 100, 12000, 6, 120, 300);

Console.WriteLine("A car");

objcar.print();

Console.WriteLine(" ");

Console.ReadKey();

}

}

}

47: GENERICS 2

/\* it is much more flexible and removes the overhead inolved in boxing and unboxing operations that we discussed earlier

\* as a generic type is assigned a specific type only at runtime. the type checks done only at the compile time itself.

\* hence the generic stack in this example works much faster compared to its non generic counterpart.

\* the following is implementation of generic stack using c# generics. \*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace generics\_2

{

public class CustomStack<S>

{

const int size = 10;

private S[] register;

public int count=0;

public void CustomStacK()

{

register=new S[size];

}

public void Push(S x)

{

if(count<size)

register[count++]=x;

}

public S Pop()

{

return register[--count];

}

}

public class Test

{

static void Main(string[] args)

{

CustomStack<int> intStack = new CustomStack<int>();

intStack.Push(10);

int i=intStack.Pop();

Console.WriteLine(i);

CustomStack<double> floatStack = new CustomStack<double>();

floatStack.Push(12.25);

double f = floatStack.Pop();

Console.WriteLine(f);

CustomStack<char> charStack = new CustomStack<char>();

charStack.Push('A');

char c = charStack.Pop();

Console.WriteLine(c);

}

}

}

48: GENERIC: NON GENERIC LISTING

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace generics

{

//non generics listing

public class CustomStack

{

const int size= 10;

private object[] register; //array of type object(holds any datatype)

private int count=0;

private CustomStack()

{

register=new object[size];

}

public void Push(object x)

{

if( count <size)

register[count++]=x;

}

public object Pop()

{

return register[--count];

}

static void Main(string[] args)

{

CustomStack intStack = new CustomStack();

intStack.Push(10);

int i = (int)intStack.Pop();

Console.WriteLine(i);

Console.ReadKey();

}

}

}

49 FOREACH

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace @foreach

{

public class Class1

{

static void Main(string[] args)

{

int[] fibarray = new int[] { 0, 1, 1, 2, 3, 5, 8, 13 };

foreach (int element in fibarray)

{

System.Console.WriteLine(element);

}

System.Console.WriteLine();

// You can maintain a count of the elements in the collection.

int count = 0;

foreach (int element in fibarray)

{

count += 1;

System.Console.WriteLine("Element #{0}: {1}", count, element);

}

System.Console.WriteLine("Number of elements in the array: {0}", count);

Console.Read();

}

}

}

50 ENUMERATION

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace enumeration

{

public class Class1

{

enum LoginType

{

Admin = 1, Employer, Guest

}

static void Main()

{

int utype;

Console.WriteLine("Enter the user type as 1 or 2 or 3");

utype = Convert.ToInt32(Console.ReadLine());

LoginType testType = (LoginType)utype;

switch (testType)

{

case LoginType.Admin:

Console.WriteLine("You are an Administrator");

break;

case LoginType.Employer:

Console.WriteLine("You are an Employer");

break;

case LoginType.Guest:

Console.WriteLine("You are a Guest");

break;

default:

Console.WriteLine("You dont have access to login");

break;

}

Console.Read();

}

}

}

51 ABSTRACT CLASS N INTERFACE

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace abstract\_class

{

abstract class shapeClass

{

public abstract int Area();

}

class Square: shapeClass

{

int side = 0;

public Square(int n)

{

side = n;

}

public override int Area()

{

return side \* side;

}

static void Main()

{

Square sq = new Square(8);

Console.WriteLine("area : {0}", sq.Area());

Console.ReadKey();

}

interface I

{

void M();

}

abstract class C : I

{

public abstract void M();

}

}

}

52 STATIC CONSTRUCTOR

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace sttaic\_constructor

{

public class Bus

{

// Static variable used by all Bus instances.

// Represents the time the first bus of the day starts its route.

protected static readonly DateTime globalStartTime;

// Property for the number of each bus.

protected int RouteNumber { get; set; }

// Static constructor to initialize the static variable.

// It is invoked before the first instance constructor is run.

static Bus()

{

globalStartTime = DateTime.Now;

// The following statement produces the first line of output,

// and the line occurs only once.

Console.WriteLine("Static constructor sets global start time to {0}",

globalStartTime.ToLongTimeString());

}

// Instance constructor.

public Bus(int routeNum)

{

RouteNumber = routeNum;

Console.WriteLine("Bus #{0} is created.", RouteNumber);

}

// Instance method.

public void Drive()

{

TimeSpan elapsedTime = DateTime.Now - globalStartTime;

// For demonstration purposes we treat milliseconds as minutes to simulate

// actual bus times. Do not do this in your actual bus schedule program!

Console.WriteLine("{0} is starting its route {1:N2} minutes after global start time {2}.",

this.RouteNumber,

elapsedTime.TotalMilliseconds,

globalStartTime.ToShortTimeString());

}

}

class TestBus

{

static void Main()

{

// The creation of this instance activates the static constructor.

Bus bus1 = new Bus(71);

// Create a second bus.

Bus bus2 = new Bus(72);

// Send bus1 on its way.

bus1.Drive();

// Wait for bus2 to warm up.

System.Threading.Thread.Sleep(25);

// Send bus2 on its way.

bus2.Drive();

// Keep the console window open in debug mode.

System.Console.WriteLine("Press any key to exit.");

System.Console.ReadKey();

}

}

}

53. GENERIC COLLECTION

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace generic\_prac

{

public class Customer

{

public Customer(int id, string name)

{

ID = id;

Name = name;

}

private int m\_id;

public int ID

{

get { return m\_id; }

set { m\_id = value; }

}

private string m\_name;

public string Name

{

get { return m\_name; }

set { m\_name = value; }

}

}

class Program

{

static void Main(string[] args)

{

List<int> myInts = new List<int>();

myInts.Add(1);

myInts.Add(2);

myInts.Add(3);

for (int i = 0; i < myInts.Count; i++)

{

Console.WriteLine("MyInts: {0}", myInts[i]);

}

Dictionary<int, Customer> customers = new Dictionary<int, Customer>();

Customer cust1 = new Customer(1, "Cust 1");

Customer cust2 = new Customer(2, "Cust 2");

Customer cust3 = new Customer(3, "Cust 3");

customers.Add(cust1.ID, cust1);

customers.Add(cust2.ID, cust2);

customers.Add(cust3.ID, cust3);

foreach (KeyValuePair<int, Customer> custKeyVal in customers)

{

Console.WriteLine(

"Customer ID: {0}, Name: {1}",

custKeyVal.Key,

custKeyVal.Value.Name);

}

Console.ReadKey();

}

}

}

54. REFLECTIONS

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Reflection;

namespace reflections

{

class MyMemberInfo

{

public static int Main()

{

string testclass= "System.Reflection.PropertyInfo";

Console.WriteLine("\nfollowing is member info for class {0}", testclass);

Type mytype = Type.GetType(testclass);

MemberInfo[] MyMemberInfoArray = mytype.GetMembers();

Console.WriteLine("there are {0} members in {1}", MyMemberInfoArray.GetLength(0),mytype.FullName);

for(int counter=0;

counter<MyMemberInfoArray.GetLength(0);counter++)

{

Console.WriteLine("{0},{1} member type {2}", counter, MyMemberInfoArray[counter].Name,MyMemberInfoArray.MemberType.ToString());

}

retutn 0;

}

}

}

55 DELEGATE

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace @delegate

{

class Program

{

delegate void SimpleDelegate(string s);

static void Main(string[] args)

{

SimpleDelegate del = new SimpleDelegate(displayMessage);

del("welcome to simple delegates");

}

static void displayMessage(string s)

{

Console.WriteLine(s);

}

}

}

56.MULTICAST DELEGATES

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace multicast\_delegates

{

class Program

{

delegate void MyDelegate(string m, int a, int b);

static void Main(string[] args)

{

MyDelegate md = new MyDelegate(FirstMethod);

md += new MyDelegate(SecondMethod);

md("MESSAGE A", 4, 5);

md("MESSAGE B", 7, 11);

}

static void FirstMethod(string s1, int x1, int y1)

{

Console.WriteLine("1st method" + s1);

int sum1 = x1 + y1;

Console.WriteLine("sum1=" + sum1);

}

static void SecondMethod(string s2, int x2, int y2)

{

Console.WriteLine("2nd method" + s2);

int diff = x2 - y2;

Console.WriteLine("diff=" + diff);

}

}

}

57.STACK

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace stack

{

class Program

{

static void Main(string[] args)

{

Stack st = new Stack();

st.Push("panka");

st.Push(1);

st.Push(10.5);

st.Push(true);

st.Push('A');

Console.WriteLine("count:{0}", st.Count);

Console.WriteLine();

Console.WriteLine("element in stack");

foreach (object obj in st)

Console.WriteLine(obj);

Console.WriteLine();

Console.WriteLine("topmost element:{0}", st.Peek());

Console.WriteLine();

object topelement = st.Pop();

Console.WriteLine("removing element of stack ={0}\n Now top element = {1}", topelement, st.Peek());

if (st.Contains("pankaj"))

Console.WriteLine("found");

else

Console.WriteLine("not found");

Object[] ob = st.ToArray();

foreach (object obj in ob)

Console.WriteLine(obj);

st.Clear();

Console.WriteLine();

Console.WriteLine("count: {0}", st.Count);

Console.ReadKey();

}

}

}

58.QUEUE

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace queue

{

class Program

{

static void Main(string[] args)

{

Queue q = new Queue();

q.Enqueue("panka");

q.Enqueue(1);

q.Enqueue(10.5);

q.Enqueue(true);

q.Enqueue('A');

Console.WriteLine("count:{0}", q.Count);

Console.WriteLine();

Console.WriteLine("element in queue");

foreach (object obj in q)

Console.WriteLine(obj);

Console.WriteLine();

Console.WriteLine("topmost element:{0}", q.Peek());

Console.WriteLine();

object topelement = q.Dequeue();

Console.WriteLine("removing element of stack ={0}\n Now top element = {1}", topelement, q.Peek());

if (q.Contains("pankaj"))

Console.WriteLine("found");

else

Console.WriteLine("not found");

Object[] ob = q.ToArray();

foreach (object obj in ob)

Console.WriteLine(obj);

q.TrimToSize();

q.Clear();

Console.WriteLine();

Console.WriteLine("count: {0}", q.Count);

Console.ReadKey();

}

}

}

59. SORTED LIST

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace sortedlist

{

class Program

{

static void Main(string[] args)

{

SortedList country = new SortedList();

country["india"] = 1;

country["china"] = 2;

country["russia"] = 3;

country["brazil"] = 4;

foreach(DictionaryEntry element in country)

{

string name = (string)element.Key;

int ranking = (int)element.Value;

Console.WriteLine("name{0},ranking{1}", name, ranking);

}

}

}

}

60.HASH TABLE

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace hashtable

{

class Program

{

static void Main(string[] args)

{

Hashtable country = new Hashtable();

country["india"] = 1;

country["china"] = 2;

country["russia"] = 3;

country["brazil"] = 4;

country[100] = 1;

country[101] = 2;

country[102] = 3;

country[103] = 4;

country[103] = 4;

foreach(DictionaryEntry element in country)

{

string name = (string)element.Key;

int ranking = (int)element.Value;

Console.WriteLine("name:{0}, ramk:{1}", name, ranking);

}

}

}

}