// 9. Demonstration of error handling

using System;

namespace test

{

public class Program

{

// Main Method

static public void Main()

{

int a = 10, b = 0, c;

try

{

c = a / b;

}

catch (Exception e)

{

Console.WriteLine(e);

}

finally

{

Console.WriteLine("finally : exception raises or not but i will be executed");

}

Console.WriteLine("out of exception scope");

Console.ReadKey();

}

}

}

// 10. Simple calculator using switch statement

using System;

namespace Calculator

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter the action to be performed");

Console.WriteLine("Press 1 for Addition");

Console.WriteLine("Press 2 for Subtraction");

Console.WriteLine("Press 3 for Multiplication");

Console.WriteLine("Press 4 for Division \n");

int action = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter 1st input");

int opr1 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter 2nd input");

int opr2 = Convert.ToInt32(Console.ReadLine());

int result = 0;

switch (action)

{

case 1:

{

result = opr1 + opr2;

break;

}

case 2:

{

result = opr1 - opr2;

break;

}

case 3:

{

result = opr1 \* opr2;

break;

}

case 4:

{

result = opr1 / opr2;

break;

}

default:

Console.WriteLine("Wrong action!! try again");

break;

}

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

Console.ReadKey();

}

}

}

//11. Demonstration use of Virtual and Overide key word.

using System;

namespace VirtualOveride

{

class Program

{

static void Main(string[] args)

{

// Base class, method draw is virtual

Shape objShape = new Shape();

objShape.draw();

// Derived class, method draw is overidden

Square objSquare = new Square();

objSquare.draw();

Console.ReadKey();

}

}

// Base class

class Shape

{

public virtual void draw()

{

Console.WriteLine("From base class : Shape");

}

}

// Derived class

class Square : Shape

{

public override void draw()

{

Console.WriteLine("From derived class, Method(draw) is overidden : Square");

}

}

}

// 13. Abstract class and Abstract method

using System;

namespace AbstractClsMthd

{

class Program

{

static void Main(string[] args)

{

int a, b;

Console.WriteLine("Enter first and second values");

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

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

// Creating an object

MathsOprtion objMathsOprtion = new MathsOprtion();

Console.WriteLine("sum = {0}", objMathsOprtion.addition(a, b));

Console.WriteLine("substraction = {0}", objMathsOprtion.substraction(a, b));

Console.ReadKey();

}

}

// Abstract class having prototype.

abstract class Arithmatic

{

// Abstract method : doen't have implementation

public abstract int addition(int a, int b);

// Method have implementation

public int substraction(int a, int b)

{

return a - b;

}

}

// Class is implemented.

class MathsOprtion : Arithmatic

{

public override int addition(int a, int b)

{

return a + b;

}

}

}

// 14. Illustrate different properties

using System;

namespace Properties

{

class Program

{

static void Main(string[] args)

{

Student objStd = new Student();

// Accesing read only property i.e College Name

Console.WriteLine("College Name = {0}", objStd.CollegeName);

// setting property. Write property

objStd.Remarks = "Welcome";

//Console.WriteLine("College Name = {0}", objStd.Remarks);

// setting and accessing (Write and Read ) property

objStd.Name = "Kiran";

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

Console.ReadKey();

}

}

class Student

{

string collegeName="ABC", name, remarks;

// Read only property

public string CollegeName

{

get

{

return collegeName;

}

}

// Write only property

public string Remarks

{

set

{

remarks = value;

}

}

// Write and Read property

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

}

}