1. Write a program to find the factorial of given number using the concept of delegate

Factorial.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace factorial

{

public class Factorial

{

public int calculatefactorial(int n)

{

int fact = 1;

if( n < 0)

{

Console.WriteLine("Error! Factorial of a negative number does not exist.");

}

else

{

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

{

fact \*= i;

}

}

return fact;

}

}

}

Program.cs

using System;

using factorial;

namespace Lab2

{

internal class Program

{

public delegate int calcfactorial(int x);

static void Main(string[] args)

{

Factorial factorialcalc = new Factorial();

calcfactorial calculatefactorial = new calcfactorial(factorialcalc.calculatefactorial);

Console.WriteLine("Enter an integer:");

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

int result = calculatefactorial.Invoke(input);

Console.WriteLine($"The factorial of {input} is {result}");

}

}

}

1. Create a Class Quadratic having a, b, c, x1, and x2 as instance variables. Create a function name Input (no parameter and no return type) to take the user input for a, b, and c. Then create another function name double[ ] Calculate ( ) to calculate two roots and assign to variable x1 and x2 and return these two roots must return to main function. Create another class Imain having main () function to create an object of Quadratic class and invoke the function.

Quadratic.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Quadratic

{

public class quadratic

{

public double a, b, c, x1, x2;

public void Input()

{

Console.Write("Enter the vlaue of a:");

a = double.Parse(Console.ReadLine());

Console.Write("Enter the vlaue of b:");

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

Console.Write("Enter the vlaue of c:");

c = double.Parse(Console.ReadLine());

}

public double[] Calculate()

{

double discriminant = b \* b - 4 \* a \* c;

if (discriminant < 0)

{

Console.WriteLine(" No real roots exist.");

return null;

}

else if (discriminant == 0 )

{

x1 = -b / (2 \* a);

x2 = x1;

}

else

{

x1 = (-b + Math.Sqrt(discriminant)) / (2 \* a);

x2 = (-b - Math.Sqrt(discriminant)) / (2 \* a);

}

return new double[] { x1, x2 };

}

}

}

Program.cs

using System;

using Quadratic;

namespace Lab2

{

class Imain

{

public static void Main(string[] args)

{

quadratic quad = new quadratic();

quad.Input();

double[] roots = quad.Calculate();

if (roots != null)

{

Console.WriteLine("Root 1: " + roots[0]);

Console.WriteLine("Root 2: " + roots[1]);

}

}

}

}

1. Create a class Student having instance variable age and name and class also contains a function name void input( ) for user input age and name. Then create another class Imain and create an array of size 5 of Student then store the Student in array and print those records of array whose age is greater than or equal to 24.

Pseudocode:

class Student

{

int age;

void input( )

{

//input goes here

}

}

class Imain

{

Student[ ] lstStudent = new Student[ 5];

//Create 5 objects of Student and store in Array lstStudent

//print the roll and name of students stored in array whose ages is greater or equal to 24

}

Student.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace student

{

public class Student

{

public int age;

public string name;

public void Input()

{

Console.WriteLine("Enter student's age: ");

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

Console.WriteLine("Enter student's name: ");

name = Console.ReadLine();

}

}

}

Program.cs

using System;

using student;

namespace Lab2

{

class Imain

{

public static void Main(string[] args)

{

Student[] lstStudent = new Student[5];

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

{

Console.WriteLine($"Enter details for student {i + 1}:");

lstStudent[i] = new Student();

lstStudent[i].Input();

}

Console.WriteLine("\nStudents aged 24 or older:");

foreach (Student student in lstStudent)

{

if (student.age >= 24)

{

Console.WriteLine($"Name: {student.name}, Age: {student.age}");

}

}

}

}

}

1. Create a class Time with three integer member variables hr ,min,sec. The class also will contain the method Tim Sum(Time t1,Time t2) method that will return the sum of t1and t2 Create a class TimeDemo with main method that will create an object of Time and to invoke the Sum function and print the added time

Output :

t1—>5:40:40

t2—>4:40:50Time

Total Sum t3—>10:21:30

Time.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace time

{

public class Time

{

public int hr, min, sec;

public Time Sum(Time t1, Time t2 )

{

Time sum = new Time();

sum.sec = t1.sec + t2.sec;

sum.min = t1.min + t2.min + sum.sec/60;

sum.sec %= 60;

sum.hr = t1.hr + t2.hr + sum.min/60;

sum.min %= 60;

sum.hr %= 24;

return sum;

}

public override string ToString()

{

return $"{hr:D2}:{min:D2}:{sec:D2}";

}

}

}

Program.cs

using System;

using time;

namespace Lab2

{

class TimeDemo

{

static void Main(string[] args)

{

Time t1 = new Time { hr = 5, min = 40, sec = 40};

Time t2 = new Time { hr = 4, min = 40, sec = 50 };

Time t3 = new Time();

t3 = t3.Sum(t1, t2);

Console.WriteLine($"t1 --> {t1}");

Console.WriteLine($"t2 --> {t2}");

Console.WriteLine($"Total Sum t3 --> {t3}");

}

}

}

1. Write a program to read a file from input.txt then copy the content of input.txt to output.txt.

E.g

**//input.txt**

Hello and Welcome

It is the first content

of the text file from first file

**output.txt:**

Hello and Welcome

It is the first content

of the text file from first file

Readncopy.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace readncopy

{

public class Readncopy

{

public static string Readfile(string filePath)

{

try

{

if (File.Exists(filePath))

{

return File.ReadAllText(filePath);

}

else

{

Console.WriteLine("File not found. Creating new file.");

File.WriteAllText(filePath, "Hello and Welcome\r\nIt is the first content \r\nof the text file from first file\r\n");

return null;

}

}

catch (Exception ex)

{

Console.WriteLine("Error reading file: " + ex.Message);

return null;

}

}

public static void CopyFile( string content, string newDestination)

{

try

{

if (content != null)

{

File.WriteAllText(newDestination, content);

Console.WriteLine("File Copied");

}

else

{

Console.WriteLine("Content is null. Cannot copy.");

}

}

catch(Exception e)

{

Console.WriteLine("Error copying file:" + e.Message);

}

}

}

}

Program.cs

using System;

using readncopy;

namespace Lab2

{

class fileoperation

{

static void Main(String[] args)

{

string inputFilePath = @"D:\Shreyan Bista\6th Sem\NCC\Lab2\Lab2\bin\Debug\net6.0\input.txt";

string outputFilePath = @"D:\Shreyan Bista\6th Sem\NCC\Lab2\Lab2\bin\Debug\net6.0\output.txt";

string content = Readncopy.Readfile(inputFilePath);

if (content != null)

{

Readncopy.CopyFile(content,outputFilePath);

}

else

{

Console.WriteLine("File reading failed. Can't copy.");

}

}

}

}

1. Write a program to read a file from Input.txt then display those words which ends with the character ‘g’.

E.g

**//Input.txt**

C# is a OOP Programming Language

Mr Hari Gurung loves a programming

**Output**

Programming

Gurung

Programming

Dispg.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dispg

{

public class Displayg

{

public static void Displaytheg(string filePath)

{

try

{

string[] words = File.ReadAllText(filePath).Split(new char[] { ' ', '\n', '\r', '\t' }, StringSplitOptions.RemoveEmptyEntries);

Console.WriteLine("Words ending with 'g':");

foreach (string word in words)

{

if (word.EndsWith("g", StringComparison.OrdinalIgnoreCase))

{

Console.WriteLine(word);

}

}

}

catch (Exception e)

{

Console.WriteLine($"An error occured: {e.Message}");

}

}

}

}

Program.cs

using System;

using Dispg;

namespace Lab2

{

class Program

{

static void Main(string[] args)

{

string filePath = "Input.txt";

Displayg.Displaytheg(filePath);

}

}

}

1. Create an interface called ICalculator which has methods int add(int x ,int y) and int diff(int x,int y) to perform addition and subtraction of numbers passed as arguments. Then define a class that implements interface ICalculator.

ICalculator.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CalculatorInterface

{

public interface ICalculator

{

int Add(int x, int y);

int Diff(int x, int y);

}

public class Calculator : ICalculator

{

public int Add(int x, int y)

{

return x + y;

}

public int Diff(int x, int y)

{

return x - y;

}

}

}

Program.cs

using CalculatorInterface;

using System;

namespace Lab2

{

class Program

{

static void Main(string[] args)

{

Calculator calc = new Calculator();

int resultAdd = calc.Add(6, 10);

int resultDiff = calc.Diff(22, 18);

Console.WriteLine("Sum : " + resultAdd);

Console.WriteLine("Difference: " +resultDiff);

}

}

}

1. Create an interface Shape with method get() and display().Create two classes Rectangle and Square which implements this interface.defines the member variable of these classes as per requirement in class itself.Create some instances of Rectangle and Square classes and demonstrate interface implementation by classes.

ShapeInterface.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShapesInterface

{

public interface IShape

{

void Get();

void Display();

}

public class Rectangle : IShape

{

private double length;

private double width;

public Rectangle (double length, double width)

{

this.length = length;

this.width = width;

}

public void Get()

{

Console.WriteLine("Given Length of the rectangle is " + length);

Console.WriteLine("Given Width of the rectangle is " + width);

}

public void Display()

{

Console.WriteLine("Rectangle\_\_\_\_");

Console.WriteLine("Length: " + length);

Console.WriteLine("Width: " + width);

}

}

public class Square : IShape

{

private double side;

public Square(double side)

{

this.side = side;

}

public void Get()

{

Console.WriteLine("Given side of the square is " + side);

}

public void Display()

{

Console.WriteLine("Square\_\_\_\_");

Console.WriteLine("Side: " + side);

}

}

Program.cs

using System;

using ShapesInterface;

namespace Lab2

{

class Program

{

static void Main(string[] args)

{

Rectangle rectangle = new Rectangle(6, 8);

Square square = new Square(12);

rectangle.Get();

rectangle.Display();

Console.WriteLine();

square.Get();

square.Display();

}

}

}

1. Create an abstract class called Vehicle with an abstract method startEngine() and a non-abstract method stopEngine(). Derive a concrete class Airplane from Vehicle, having instance variables such as model, manufacturer, and maxCapacity. The Airplane class should implement the abstract startEngine() method with appropriate logic for starting an aircraft engine. Additionally, provide concrete implementations for methods like takeOff() and land().

In your main class, create an instance of the Airplane class and demonstrate the usage of the inherited methods (startEngine(), stopEngine(), takeOff(), and land()) along with accessing the instance variables.

Vehicle.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace vehicle

{

public abstract class Vehicle

{

public abstract void StartEngine();

public void StopEngine()

{

Console.WriteLine("Engine Stopped");

}

}

public class Airplane: Vehicle

{

private string model;

private string manufacturer;

private int maxcapacity;

public Airplane (string model, string manufacturer, int maxcapacity)

{

this.model = model;

this.manufacturer = manufacturer;

this.maxcapacity = maxcapacity;

}

public override void StartEngine()

{

Console.WriteLine("Starting airplane engine.");

}

public void Takeoff()

{

Console.WriteLine("Airplane taking off.");

}

public void Land()

{

Console.WriteLine("Airlplane Landing.");

}

public void DisplayDetails()

{

Console.WriteLine($"Model: {model}");

Console.WriteLine($"Manufacturer: {manufacturer}");

Console.WriteLine($"Maximum Capacity: {maxcapacity}");

}

}

}

Program.cs

using System;

using vehicle;

namespace Lab2

{

class Program

{

static void Main(string[] args)

{

Airplane airplane = new Airplane ("Zeppelin LZ 1", "Luftschiffbau Zeppelin", 5);

Console.WriteLine("Airplane Details:");

airplane.DisplayDetails();

Console.WriteLine();

Console.WriteLine("Status");

airplane.StartEngine();

airplane.Takeoff();

airplane.Land();

airplane.StopEngine();

}

}

1. Create a class Employee which contains the properties like ID, Name, Address and Salary. Create another class Imain then print all the employees whose salary is greater than 40000 using the concept of Linq Query.

Employeeinfo.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Employeeinfo

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public string Address { get; set; }

public decimal Salary { get; set; }

}

}

Program.cs

using System;

using System.Net.Sockets;

using Employeeinfo;

namespace Lab2

{

public class Imain

{

static void Main(String[] args)

{

List<Employee> employees = new List<Employee>

{

new Employee {Id = 1,Name = "Ramesh", Address = "Thimi", Salary = 50000 },

new Employee {Id = 1,Name = "Suresh", Address = "Chyamashingh", Salary = 60000 },

new Employee {Id = 1,Name = "Kaman", Address = "Kamalbinayak", Salary = 35000 },

new Employee {Id = 1,Name = "Logan", Address = "Metropolis", Salary = 45000 },

new Employee {Id = 1,Name = "Slade", Address = "Gotham", Salary = 55000 }

};

var highpaid = from emp in employees where emp.Salary > 40000 select emp;

Console.WriteLine("Employees with Salary > 40000:");

foreach(var emp in highpaid)

{

Console.WriteLine($"ID: {emp.Id}\t Name: {emp.Name}\t Address: {emp.Address}\t Salary: {emp.Salary}");

}

}

}

}

1. Create an array of Integer of size n then enter the element in array and find the sum of odd numbers of array elements using the concept of Lambda Expression.

Oddsum.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SumOfOdd

{

public class Array

{

public static int OddSum(int[] numbers)

{

int sumOfodd = numbers.Where(num => num % 2 != 0).Sum();

return sumOfodd;

}

}

}

Program.cs

using System;

using System.Net.Sockets;

using SumOfOdd;

namespace Lab2

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter the size of the array:");

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

int[] numbers = new int[n];

Console.WriteLine("Enter the elements of the array:");

for (int i = 0; i < n; i ++ )

{

numbers[i] = Convert.ToInt32(Console.ReadLine());

}

int sumOfodd = SumOfOdd.Array.OddSum(numbers);

Console.WriteLine($"Sum of odd numbers in the array: {sumOfodd}");

}

}

}