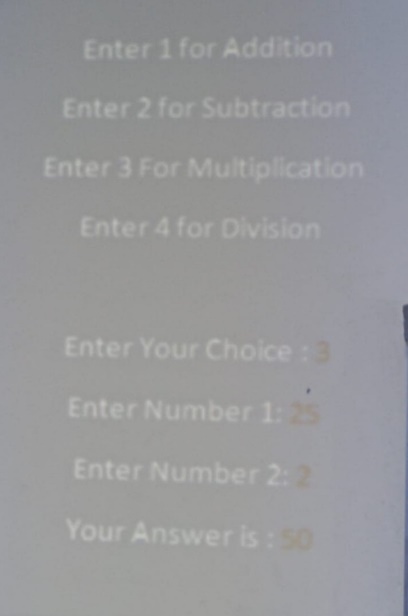
**PMK NIMNANJALEE**

**26994**

**Lab 05**

**Question 03.**

****

**Create the above mentioned console application and display it to the user. If user need to do an Addition user need to insert 1 as the choice. For subtraction it should be 2 etc.**

**Your program should contain a separate class call “CalculateValues” and inside the class you should add *four methods* which perform *four arithmetic operations*. All the methods should take two parameters which are user inserted numbers.**

**And at the end of the method return the answer out of the method.**

**In main class if user want to do an addition call only the addition method in separate class.**

**If user want to do a subtraction call only the subtraction method in separate class. ETC.**

**And display the final answer as shown in the figure 01.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LS05

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Choose an operation:");

Console.WriteLine("1. Addition");

Console.WriteLine("2. Subtraction");

Console.WriteLine("3. Multiplication");

Console.WriteLine("4. Division");

Console.Write("Enter your choice: ");

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

Console.Write("Enter the first number: ");

double num1 = double.Parse(Console.ReadLine());

Console.Write("Enter the second number: ");

double num2 = double.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Console.WriteLine("Result: " + CalculateValues.Addition(num1, num2));

break;

case 2:

Console.WriteLine("Result: " + CalculateValues.Subtraction(num1, num2));

break;

case 3:

Console.WriteLine("Result: " + CalculateValues.Multiplication(num1, num2));

break;

case 4:

Console.WriteLine("Result: " + CalculateValues.Division(num1, num2));

break;

default:

Console.WriteLine("Invalid choice!");

break;

}

Console.ReadLine();

}

}

public static class CalculateValues

{

public static double Addition(double num1, double num2)

{

return num1 + num2;

}

public static double Subtraction(double num1, double num2)

{

return num1 - num2;

}

public static double Multiplication(double num1, double num2)

{

return num1 \* num2;

}

public static double Division(double num1, double num2)

{

if (num2 == 0)

{

throw new DivideByZeroException();

}

return num1 / num2;

}

}

}

**A screenshot of a computer program

Description automatically generated**

**Question 04.**

**Add a separate class file to Console application program and create a method call *private void sayHello().***

**Inside the method display hello world.**

**In main class create object and try to access the sayHello() method by using the class object.**

**Can you access the method? Explain why?**

**SeparateClass.cs:**

**using System;**

**namespace MyNamespace**

**{**

**public class SeparateClass**

**{**

**private void SayHello()**

**{**

**Console.WriteLine("Hello, World!");**

**}**

**}**

**}**

**Program.cs:**

**using System;**

**namespace MyNamespace**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**SeparateClass separateObject = new SeparateClass();**

**Console.WriteLine("You cannot access the private method 'SayHello()' from outside the class.");**

**Console.ReadLine();**

**}**

**}**

**}**

**The method SayHello() is declared as private in the SeparateClass. As a result, it cannot be accessed from outside the class, including from the Program class. Private access modifiers restrict access to class members to within the class itself. The purpose of using private is to encapsulate implementation details and control access to methods, ensuring data integrity and avoiding unintended interference from external code. This concept enforces the principle of information hiding, promoting better code organization and maintenance. By making SayHello() private, we limit its accessibility, and it can only be called and used within the confines of the SeparateClass.**

**Question 05.**

**Declare a Single dimensional array with 10 elements. Input the values to the array and find the followings,**

* **Minimum value.**
* **Maximum value.**
* **Average value.**
* **Reverse order of values.**

**Hint – use a method which in separate class. And call the method from main the method.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LS05

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[10];

Console.WriteLine("Enter 10 integers to populate the array:");

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

{

arr[i] = int.Parse(Console.ReadLine());

}

ArrayStats stats = new ArrayStats(arr);

Console.WriteLine("Minimum value: {0}", stats.GetMinValue());

Console.WriteLine("Maximum value: {0}", stats.GetMaxValue());

Console.WriteLine("Average value: {0}", stats.GetAverageValue());

Console.WriteLine("Array in reverse order:");

foreach (int num in stats.GetReversedArray())

{

Console.Write("{0} ", num);

}

Console.ReadLine();

}

}

public class ArrayStats

{

private int[] arr;

public ArrayStats(int[] arr)

{

this.arr = arr;

}

public int GetMinValue()

{

int min = arr[0];

for (int i = 1; i < arr.Length; i++)

{

if (arr[i] < min)

{

min = arr[i];

}

}

return min;

}

public int GetMaxValue()

{

int max = arr[0];

for (int i = 1; i < arr.Length; i++)

{

if (arr[i] > max)

{

max = arr[i];

}

}

return max;

}

public double GetAverageValue()

{

double sum = 0;

foreach (int num in arr)

{

sum += num;

}

return sum / arr.Length;

}

public int[] GetReversedArray()

{

int[] reversedArr = new int[arr.Length];

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

{

reversedArr[arr.Length - 1 - i] = arr[i];

}

return reversedArr;

}

}

}

A screenshot of a computer program

Description automatically generated