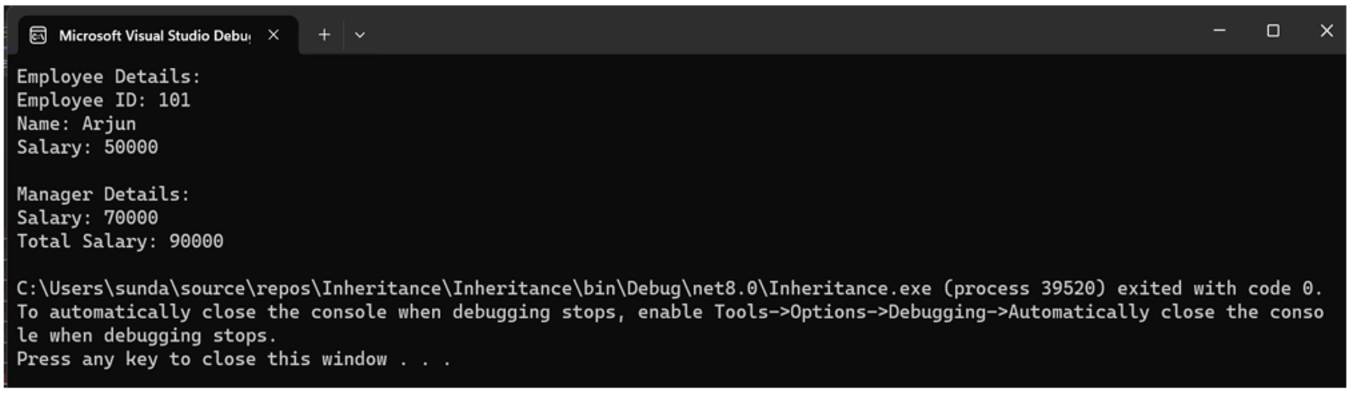
4.Write the programs to see how C# handles inheritance. Create a class Employee with 2 private data members: _empId and _name; 1 protected data member: _salary Add constructors: default and parameterized Add properties for the data members Add a method showDetails to display the values for a particular Employee. Write another class Manager with 1 private data member, _travelAllowance, which inherits from the Employee class. Add constructors: default and parameterized Add 2 methods: showSalary – which displays salary only showTotalSalary – which displays _salary + _travelAllowance In the Main method, create an object of Employee and Manager. For the employee object call the method, showDetails, and for the manager object call the methods, showSalary and showTotalSalary.

Code:

```
using System;
 // Base class Employee
class Employee
{
    // Private data members
    private int _empId;
    private string _name;

    // Protected data member
    protected double _salary;
     // Default constructor
    public Employee()
    {
        _empId = 0;
        _name = "Unknown";
        _salary = 0.0;
    }
     // Parameterized constructor
    public Employee(int empId, string name, double salary)
    {
        _empId = empId;
        _name = name;
        _salary = salary;
    }

    // Properties for data members
    public int EmpId
    {
        get { return _empId; }
        set { _empId = value; }
    }

    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }

    public double Salary
    {
        get { return _salary; }
        set { _salary = value; }
    }

    // Method to display employee details
    public void ShowDetails()
    {
        Console.WriteLine($"Employee ID: {_empId}");
        Console.WriteLine($"Name: {_name}");
        Console.WriteLine($"Salary: {_salary}");
    }
}
```

```
// Derived class Manager inheriting from Employee
class Manager : Employee
{
    // Private data member for Manager
    private double _travelAllowance;

    // Default constructor
    public Manager()
    {
        _travelAllowance = 0.0;
    }

    // Parameterized constructor
    public Manager(int empId, string name, double salary, double travelAllowance)
        : base(empId, name, salary)
    {
        _travelAllowance = travelAllowance;
    }

    // Method to display salary for Manager
    public void ShowSalary()
    {
        Console.WriteLine($"Salary: {_salary}");
    }

    // Method to display total salary for Manager
    public void ShowTotalSalary()
    {
        Console.WriteLine($"Total Salary: {_salary + _travelAllowance}");
    }
}

class Program
{
    static void Main()
    {
        // Create an object of Employee
        Employee employee = new Employee(101, "Arjun", 50000.0);

        // Call the method showDetails for the employee object
        Console.WriteLine("Employee Details:");
        employee.ShowDetails();
        Console.WriteLine();

        // Create an object of Manager
        Manager manager = new Manager(201, "Krishna", 70000.0, 20000.0);

        // Call the methods showSalary and showTotalSalary for the manager object
        Console.WriteLine("Manager Details:");
        manager.ShowSalary();
        manager.ShowTotalSalary();
    }
}
```



```
Employee Details:
Employee ID: 101
Name: Arjun
Salary: 50000

Manager Details:
Salary: 70000
Total Salary: 90000

C:\Users\sunda\source\repos\Inheritance\Inheritance\bin\Debug\net8.0\Inheritance.exe (process 39520) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

5. Write a program to use Try-catch block to handle exceptions generated while handling array and numbers.

Code:

```
using System;

class Program
{
    static void Main()
    {
        // Example arrays of numbers
        double[] array1 = { 10, 5, 2, 8, 4 };
        double[] array2 = { 2, 1, 0, 4, 2 };

        try
        {
            // Perform element-wise division and handle exceptions
            PerformElementwiseDivision(array1, array2);
        }
        catch (DivideByZeroException)
        {
            // Handle DivideByZeroException
            Console.WriteLine("Error: Cannot perform division, as there is an attempt to 
divide by zero.");
        }
        catch (IndexOutOfRangeException)
        {
            // Handle IndexOutOfRangeException
            Console.WriteLine("Error: Arrays must have the same length for element-wise 
division.");
        }
        catch (Exception)
        {
            // Handle other unexpected exceptions
            Console.WriteLine("An unexpected error occurred.");
        }
    }

    // Perform element-wise division of two arrays
    static void PerformElementwiseDivision(double[] array1, double[] array2)
    {
        // Check if arrays have the same length
        if (array1.Length != array2.Length)
        {
            // Throw an exception if arrays have different lengths
            throw new IndexOutOfRangeException();
        }

        // Iterate through each element of the arrays
        for (int i = 0; i < array1.Length; i++)
        {
            try
            {
                // Perform division and display the result
                double result = CheckDivision(array1[i], array2[i]);
                Console.WriteLine($"{array1[i]} / {array2[i]} = {result}");
            }
            catch (DivideByZeroException)
            {
                // Handle DivideByZeroException during the division
                Console.WriteLine($"Error at index {i}: Cannot divide by zero.");
            }
        }
    }

    // Check if divisor is zero before performing division
    static double CheckDivision(double dividend, double divisor)
    {
        if (divisor == 0)
        {
            // Throw DivideByZeroException if the divisor is zero
            throw new DivideByZeroException();
        }

        // Perform division and return the result
        return dividend / divisor;
    }
}
```

Output:

1:

```
10 / 2 = 5

5 / 1 = 5

Error at index 2: Cannot divide by zero.

8 / 4 = 2

4 / 2 = 2
```

2:

After changing the array length

```
double[] array1 = { 10, 5, 2, 8, 4 ,6};
 double[] array2 = { 2, 1, 0, 4, 2 };
```

```
Error: Arrays must have the same length for element-wise division.
```

6. Write a program to add and subtract 2 numbers with the help of delegates.
Code:

```
using System;

// Define the Calculation delegate
public delegate void Calculation(decimal val1, decimal val2, ref decimal result);

// Declare the namespace
namespace DelegateExample
{
    // Define the DelegateExample class
    public class DelegateExample
    {
        // Define Mycalc1 and Mycalc2 as instances of the Calculation delegate
        public Calculation Mycalc1;
        public Calculation Mycalc2;

        // Define the Add method for addition
        public static void Add(decimal add1, decimal add2, ref decimal result)
        {
            result = add1 + add2;
            Console.WriteLine("add {0} + {1} = {2}", add1, add2, result);
        }

        // Define the Sub method for subtraction
        public static void Sub(decimal sub1, decimal sub2, ref decimal result)
        {
            result = sub1 - sub2;
            Console.WriteLine("sub {0} - {1} = {2}", sub1, sub2, result);
        }
    }

    // Main program
    class Program
    {
        static void Main(string[] args)
        {
            decimal result = 0.0m;
            DelegateExample del = new DelegateExample();

            // Instantiate Mycalc1 and Mycalc2 with the Add and Sub methods
            del.Mycalc1 = new Calculation(DelegateExample.Add);
            del.Mycalc2 = new Calculation(DelegateExample.Sub);

            // Invoke the delegates
            del.Mycalc1(10.5m, 5.2m, ref result);
            del.Mycalc2(10.5m, 5.2m, ref result);
        }
    }
}
```

Output:

1:

add 10.5 + 5.2 = 15.7

sub 10.5 - 5.2 = 5.3

2:

```
del.Mycalc1(25.5m, 7.9m, ref result);
del.Mycalc2(67.6m, 8.4m, ref result);
```

add 25.5 + 7.9 = 33.4

sub 67.6 - 8.4 = 59.2

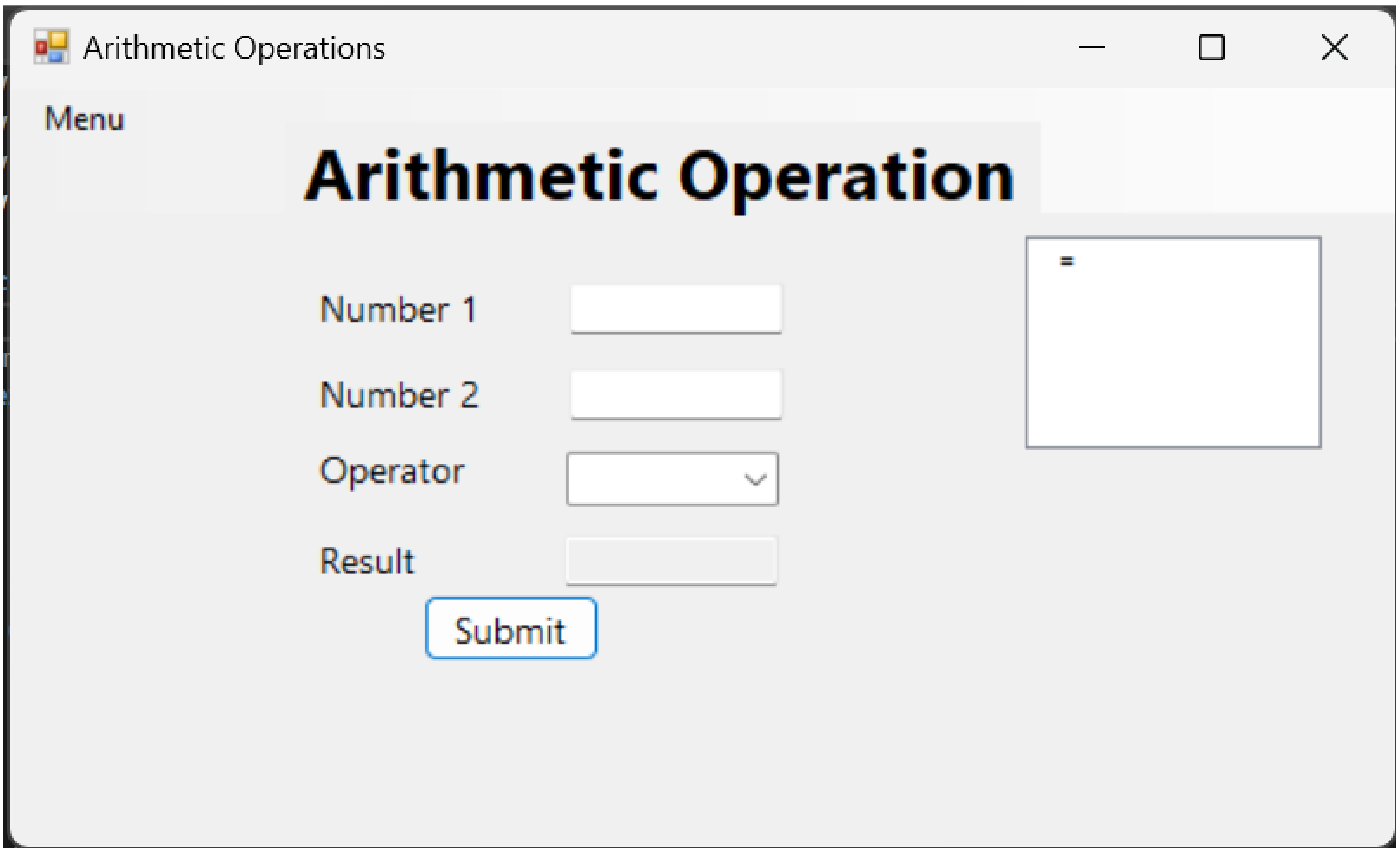
7. Build a Windows Form application that performs arithmetic operations on two numbers. Use Textbox controls to input and display the numbers, Label controls to describe each field, and Button controls to perform the arithmetic operations. Use a Combo Box control to select the operator (+, -, *, /). Use an Array class to store the history of the arithmetic operations performed. Add a menu to the form with options to clear the history and exit the application.

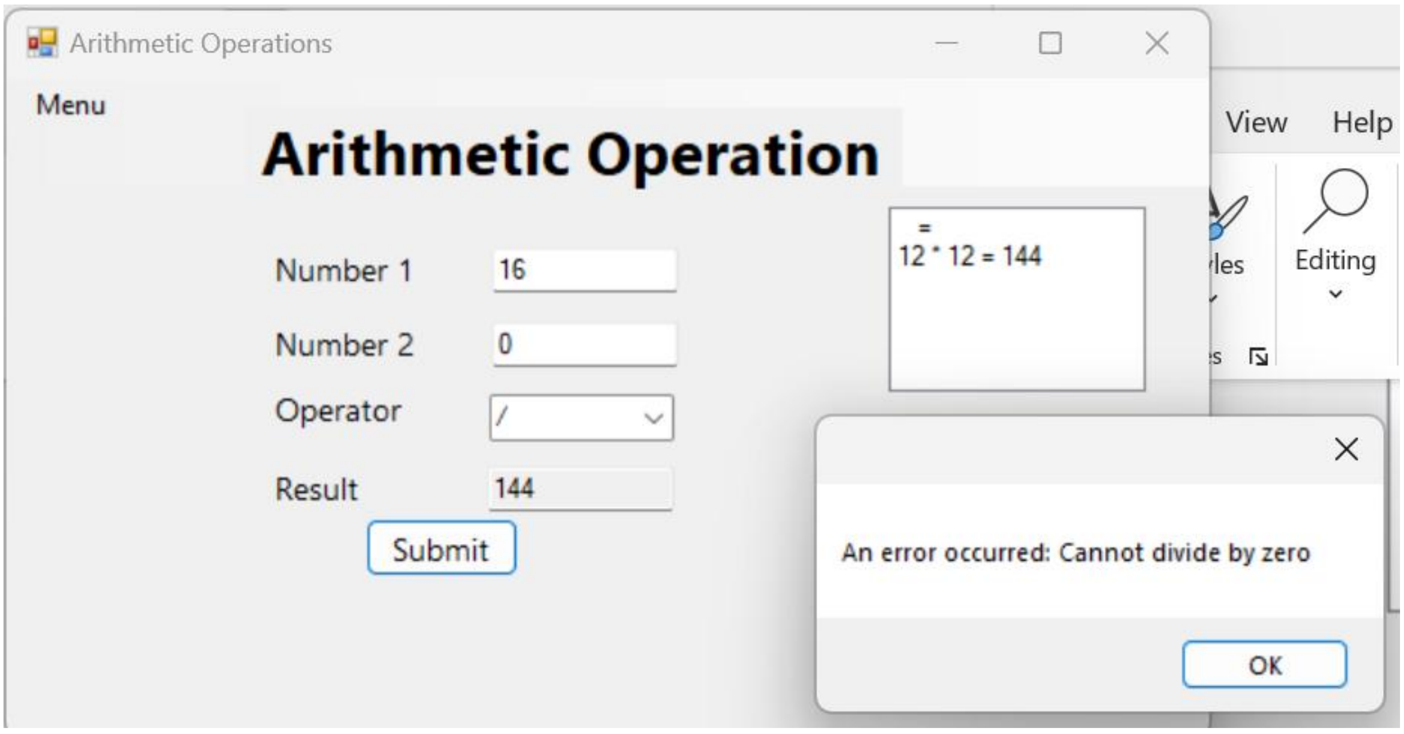
Code: Form1.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Arithmetic_Operation
{
    public partial class Form1 : Form
    {
        private readonly List<string> history;

        public Form1()
        {
            InitializeComponent();
            history = new List<string>();
            InitializeUI();
        }

        private void InitializeUI()
        {
            comboBox1.Items.Add("+");
            comboBox1.Items.Add("-");
            comboBox1.Items.Add("*");
            comboBox1.Items.Add("/");

            // Button for calculation
            button1.Click += Button1_Click;


            clearToolStripMenuItem.Click += ClearMenuItem_Click;
            exitToolStripMenuItem.Click += ExitMenuItem_Click;
            // Add the operation to the history
            string operation = $"{textBox1.Text} {comboBox1.SelectedItem} {textBox2.Text} =
{textBox3.Text}";
            history.Add(operation);

            // Update the history list
            UpdateHistoryListBox();
        }


        private void Button1_Click(object sender, EventArgs e)
        {
            try
            {
                // Get the inputs
                decimal num1 = decimal.Parse(textBox1.Text);
                decimal num2 = decimal.Parse(textBox2.Text);
                string selectedOperator = comboBox1.SelectedItem.ToString();

                // Perform arithmetic operation
                decimal result = PerformArithmeticOperation(num1, num2, selectedOperator);

                // Display the result
                textBox3.Text = $"{result}";

                // Add the operation to the history
                string operation = $"{num1} {selectedOperator} {num2} = {result}";
                history.Add(operation);

                // Update the history list
                UpdateHistoryListBox();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"An error occurred: {ex.Message}");
            }
        }

        private decimal PerformArithmeticOperation(decimal num1, decimal num2, string
operation)
        {
            switch (operation)
            {
                case "+":
                    return num1 + num2;
                case "-":
                    return num1 - num2;
                case "*":
                    return num1 * num2;
                case "/":
                    if (num2 != 0)
                        return num1 / num2;
                    else
                        throw new ArgumentException("Cannot divide by zero");
                default:
                    throw new ArgumentException("Invalid operation");
            }
        }

        private void ClearMenuItem_Click(object sender, EventArgs e)
        {
            // Clear the history
            history.Clear();
            UpdateHistoryListBox();
        }

            private void ExitMenuItem_Click(object sender, EventArgs e)
            {
                // Exit the application
                Application.Exit();
            }

            private void UpdateHistoryListBox()
        {
            listBox1.Items.Clear();
            listBox1.Items.AddRange(history.ToArray());
        }

        }
    }
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Arithmetic_Operation
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

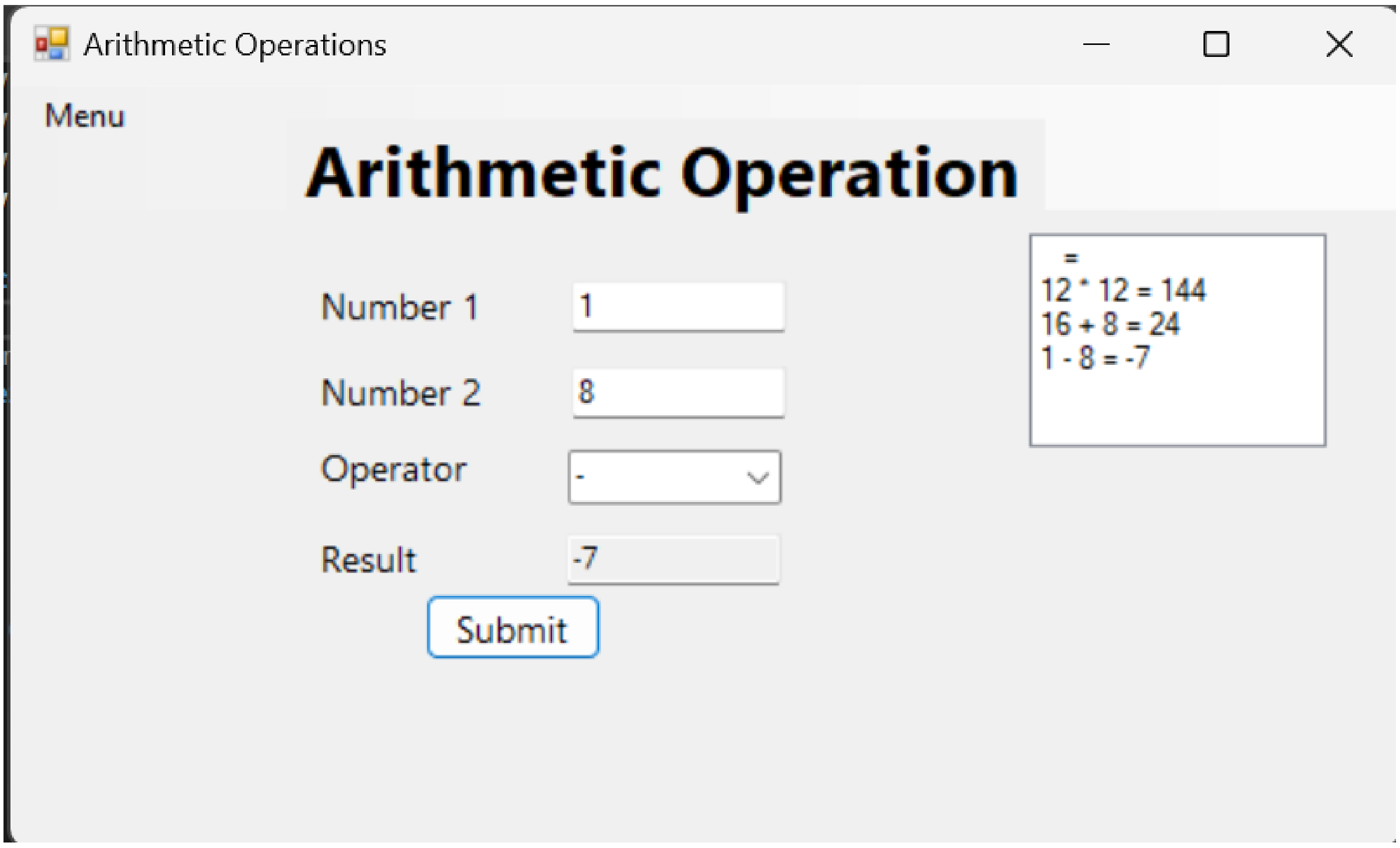




Number 1 16
Number 2 0
Operator /
Result 144

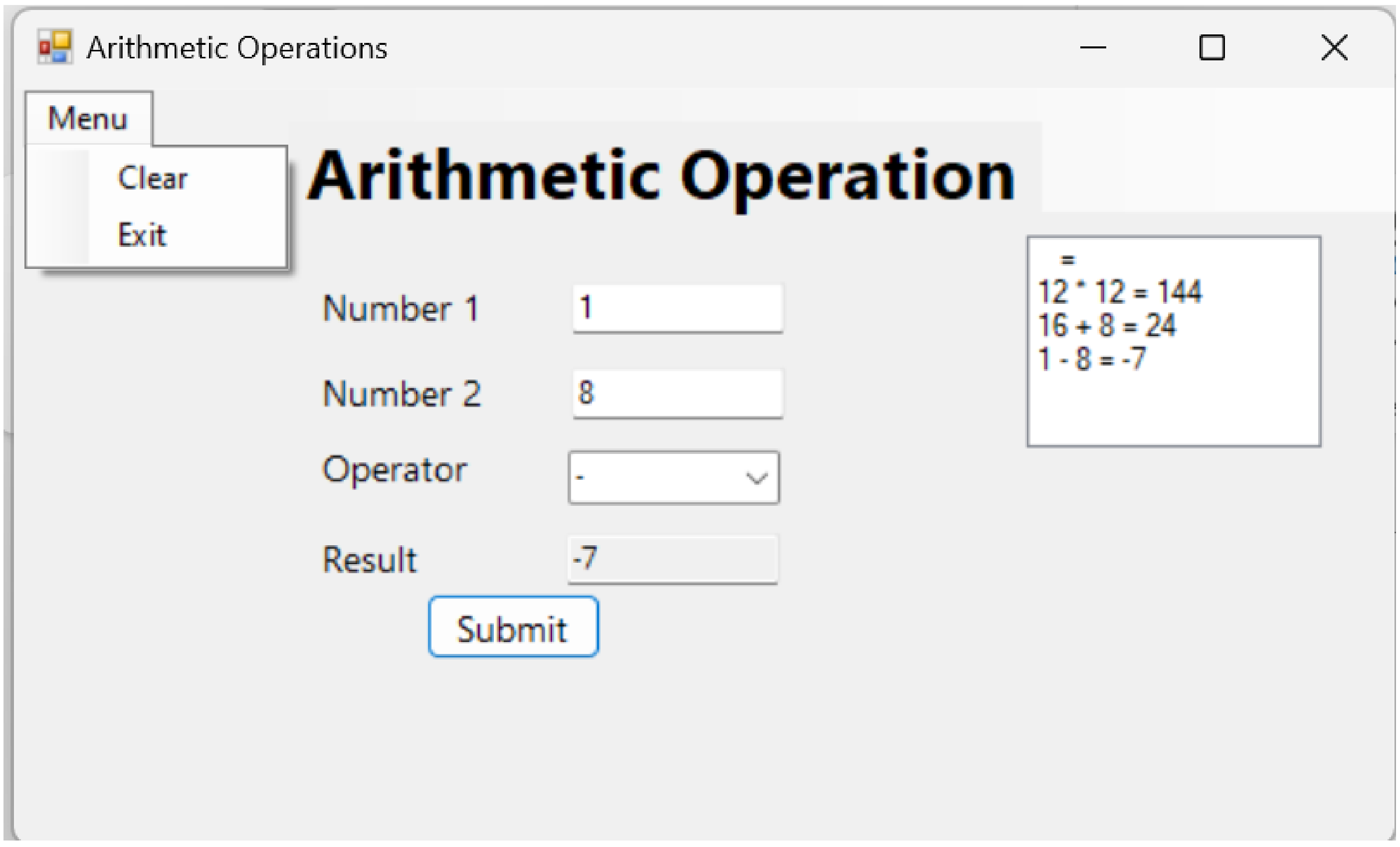
=
12 * 12 = 144

An error occurred: Cannot divide by zero



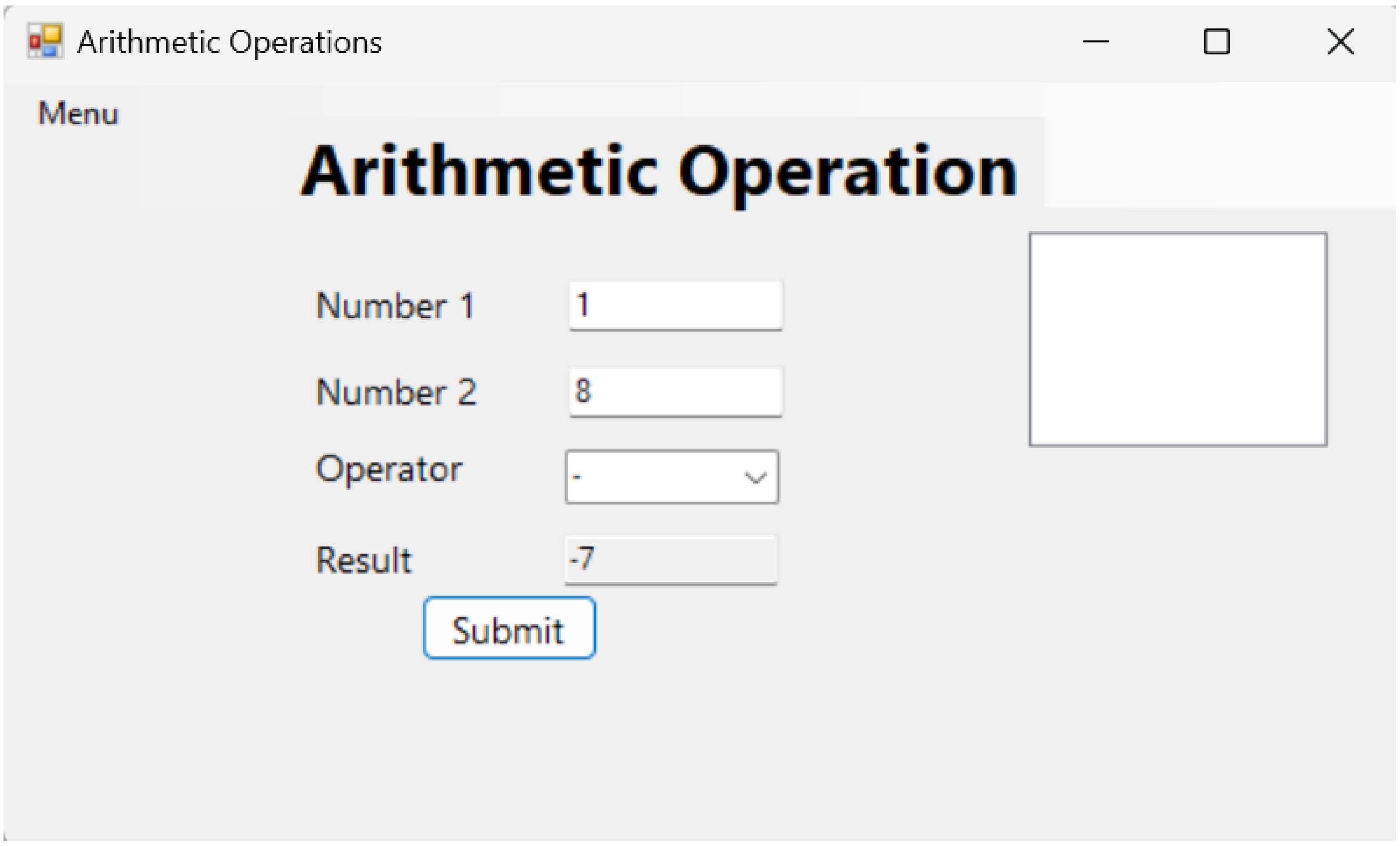
Number 1: 1
Number 2: 8
Operator: -
Result: -7

```
 =
12 * 12 = 144
16 + 8 = 24
1 - 8 = -7
```



Number 1: 1
Number 2: 8
Operator: -
Result: -7

```
 =
12 * 12 = 144
16 + 8 = 24
1 - 8 = -7
```



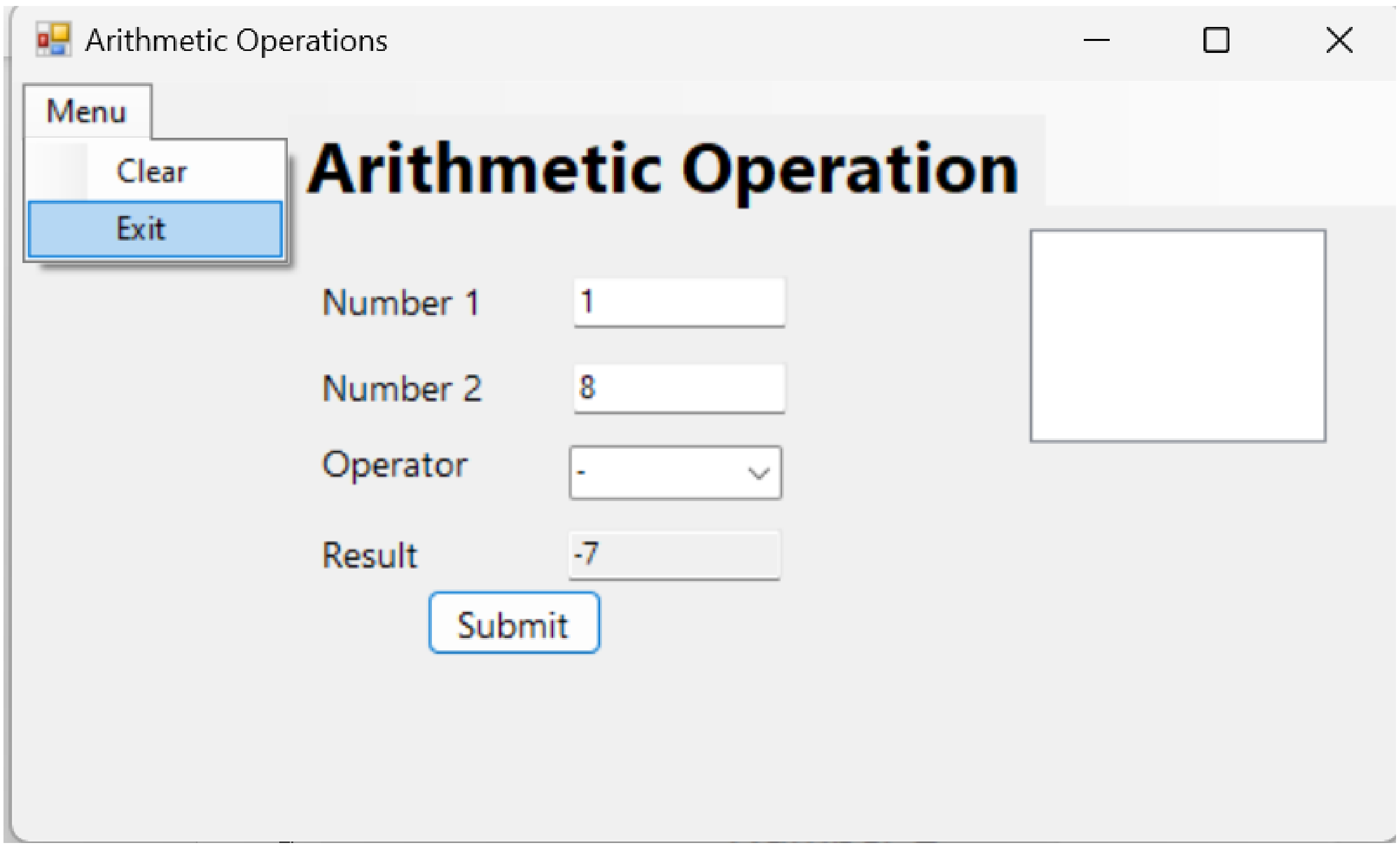
Arithmetic Operation

Number 1: 1

Number 2: 8

Operator: -

Result: -7



Arithmetic Operation

Number 1: 1

Number 2: 8

Operator: -

Result: -7