

## 目 录

一、 实验内容.....	1
二、 类图及代码结构.....	2
三、 代码实现.....	6
四、 运行效果.....	25
五、 实验心得.....	35

## 一、实验内容

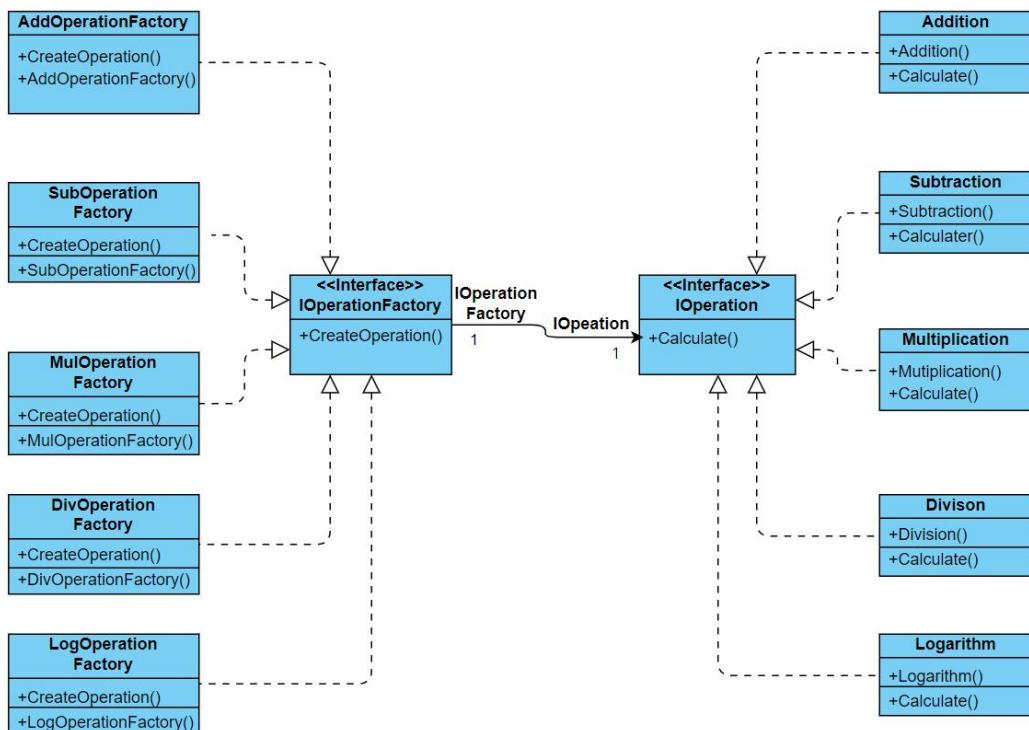
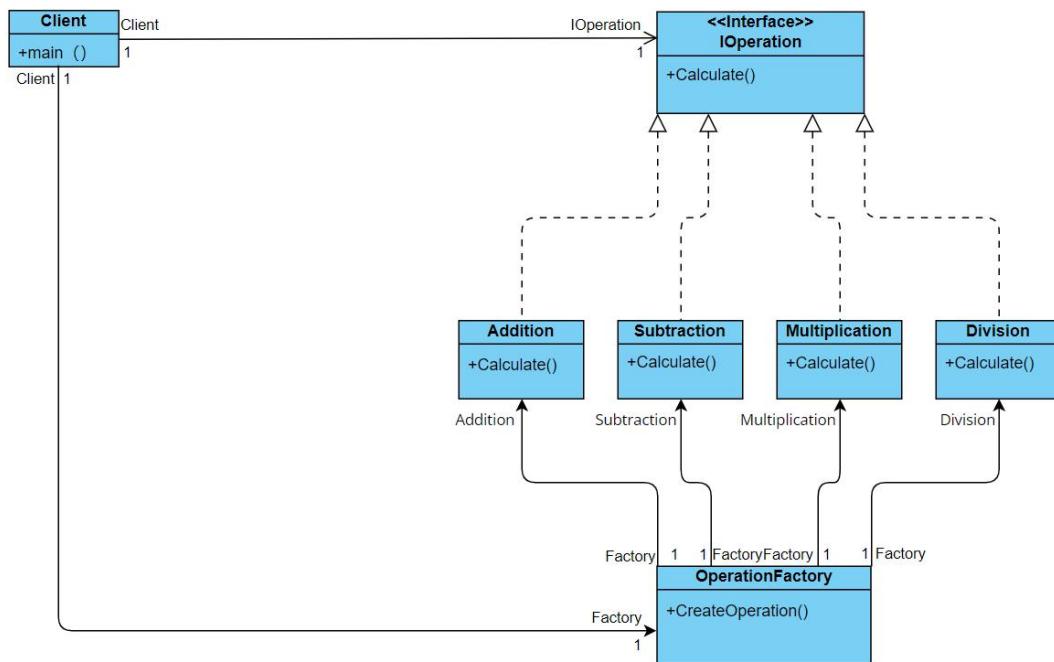
基于作业 1，完成如下任务

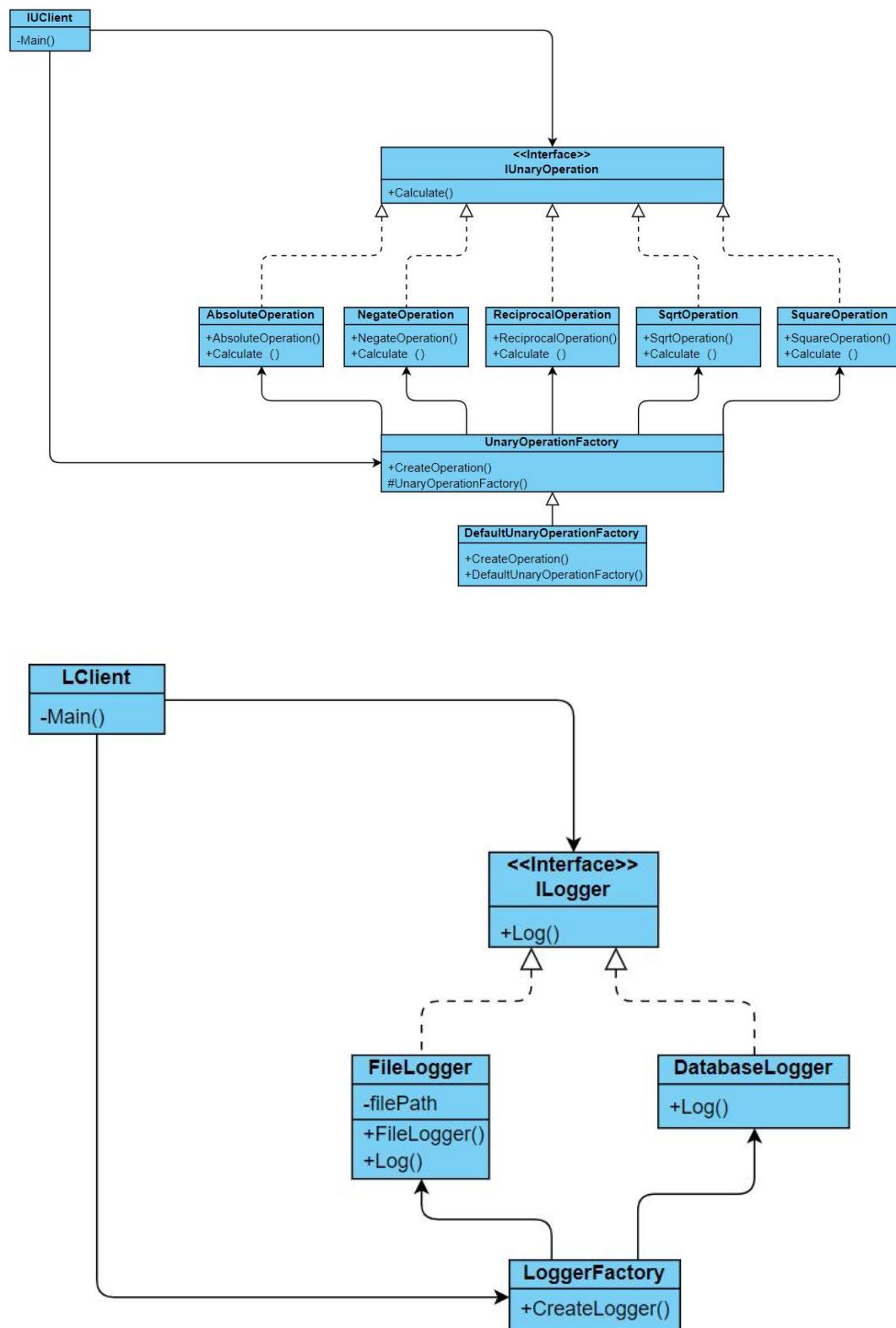
- (1) 在一元计算和二元计算接口中重载 calculate 方法，实现浮点型数据的加减乘除；
- (2) 对于整数除法和浮点型数的除法，考虑分母为零的情况
- (3) 基于 AAA 原则对于一元计算模块、二元计算模块和日志读写模块进行单元测试
- (4) 参考 windows 自带计算器实现计算器的 UI 界面，在 UI 界面中使用之前实现的一元计算机和二元计算库  
    实现计算逻辑



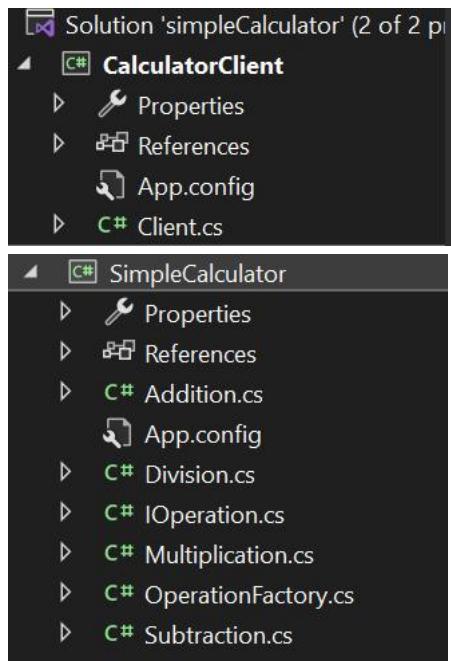
## 二、类图及代码结构

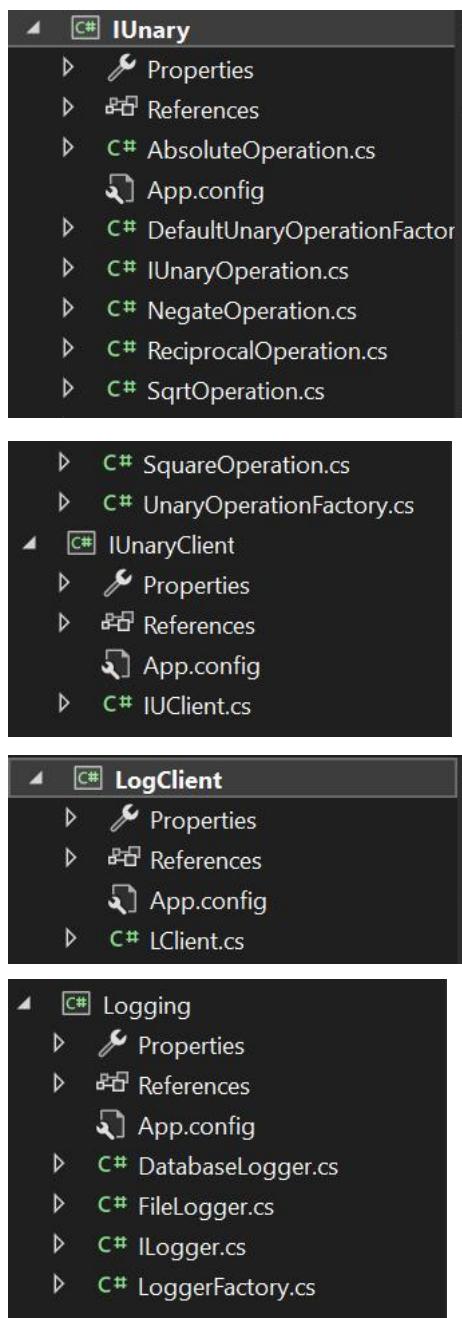
### 1. 类图





## 2. 代码结构





### 三、代码实现

(1) 在一元计算和二元计算接口中重载 calculate 方法，实现浮点型数据的加减乘除；

一元：

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace IUnary
8  {
9      /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Create an interface for Unary Operation
13      */
14     public interface IUnaryOperation
15     {
16         int CalculateInt(int num);
17         double CalculateDouble(double num);
18     }
19 }
20

```

Solution Explorer shows files: IUnary, Properties, References, AbsoluteOperation.cs, App.config, DefaultUnaryOperationFactory.cs, IUnaryOperation.cs, NegateOperation.cs, ReciprocalOperation.cs, SqrtOperation.cs, SquareOperation.cs, UnaryOperationFactory.cs.

Properties for IUnaryOperation.cs:

- Advanced: Build Action: Compile, Copy to Output Direct: Do not copy, Custom Tool, Custom Tool Namespace:
- Misc: File Name: IUnaryOperation.cs, Full Path: D:\VS\softwareDesign\simpleCalculator\IUnary\IUnaryOperation.cs

Output window shows:

```

Show output from: Debug
IBinaryClient.exe' (CLR v4.0.30319: IBinaryClient.exe): Loaded 'D:\VS\softwareDesign\simpleCalculator\calculatorClient\bin\Debug\IBinary.dll'. Symbols loaded.
The program '[9104] IBinaryClient.exe: Program Trace' has exited with code 0 (0x0).
The program '[9104] IBinaryClient.exe' has exited with code 3221225786 (0xc000013a).

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace IUnary
8  {
9      /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Neg
13      */
14     public class NegateOperation : IUnaryOperation
15     {
16         public int CalculateInt(int num)
17         {
18             return -num;
19         }
20         public double CalculateDouble(double num)
21         {
22             return -num;
23         }
24     }
25 }
26

```

Solution Explorer shows files: IUnary, Properties, References, AbsoluteOperation.cs, App.config, DefaultUnaryOperationFactory.cs, IUnaryOperation.cs, NegateOperation.cs, ReciprocalOperation.cs, SqrtOperation.cs, SquareOperation.cs, UnaryOperationFactory.cs.

Properties for NegateOperation.cs:

- Advanced: Build Action: Compile, Copy to Output Direct: Do not copy, Custom Tool, Custom Tool Namespace:
- Misc: File Name: NegateOperation.cs, Full Path: D:\VS\softwareDesign\simpleCalculator\IUnary\NegateOperation.cs

Output window shows:

```

Show output from: Debug
IBinaryClient.exe' (CLR v4.0.30319: IBinaryClient.exe): Loaded 'D:\VS\softwareDesign\simpleCalculator\calculatorClient\bin\Debug\IBinary.dll'. Symbols loaded.
The program '[9104] IBinaryClient.exe: Program Trace' has exited with code 0 (0x0).
The program '[9104] IBinaryClient.exe' has exited with code 3221225786 (0xc000013a).

```

# 《软件设计与规范实验报告》

The screenshot displays two instances of the Microsoft Visual Studio IDE. Both instances have the title bar "SimpleCalculator" and are set to the "Debug" configuration and "Any CPU" platform.

**Left Editor (ReciprocalOperation.cs):**

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace IUnary
8 {
9     /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Rec
13      */
14
15     public class ReciprocalOperation : IUnaryOperation
16     {
17
18         public int CalculateInt(int num)
19         {
20             if (num == 0)
21             {
22                 throw new ArgumentException("Cannot divide by zero.");
23             }
24             return 1 / num;
25
26         public double CalculateDouble(double num)
27         {
28             if (num == 0)
29             {
30                 throw new ArgumentException("Cannot divide by zero.");
31             }
32             return 1 / num;
33         }
34     }
35 }
```

**Right Editor (SqrtOperation.cs):**

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace IUnary
8 {
9     /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Sqrt
13      */
14
15     public class SqrtOperation : IUnaryOperation
16     {
17
18         public int CalculateInt(int num)
19         {
20             if (num < 0)
21             {
22                 throw new InvalidOperationException("Cannot take square root of negative number.");
23             }
24             double result = Math.Sqrt(num);
25             if (double.IsNaN(result))
26             {
27                 throw new InvalidOperationException("Invalid argument for square root operation.");
28             }
29             return (int)result;
30         }
31
32         public double CalculateDouble(double num)
33         {
34             if (num < 0)
35                 throw new InvalidOperationException("Cannot take square root of negative number.");
36         }
37     }
38 }
```

In both editors, lyrics are overlaid on the code:

**Left Editor (ReciprocalOperation.cs):**

"I really think we're meant to me  
Every nigga I tried to date, ]

**Right Editor (SqrtOperation.cs):**

"But every time I fall in love  
I don't wanna sound emotional

The Solution Explorer, Properties, and Output windows are visible on the right side of both instances of Visual Studio.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace IUnary
8  {
9      /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Square
13      */
14      public class SquareOperation : IUnaryOperation
15      {
16          /**
17          * Author: arbor, 23 days ago | 1 author, 2 changes
18          * public int CalculateInt(int num)
19          {
20              /**
21              * Author: arbor, 23 days ago | 1 author, 1 change
22              *     return num * num;
23          }
24          /**
25          * Author: arbor, 23 days ago | 1 author, 1 change
26          *     public double CalculateDouble(double num)
27          {
28              /**
29              * Author: arbor, 23 days ago | 1 author, 1 change
30              *     return num * num;
31          }
32      }
33  }

```

二元：

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace IBinary
8  {
9      /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Create an interface for Binary Operation
13      */
14      public interface IBinaryOperation
15      {
16          /**
17          * Author: arbor, 23 days ago | 1 author, 1 change
18          *     int CalculateInt(int num1, int num2);
19          /**
20          * Author: arbor, 23 days ago | 1 author, 1 change
21          *     double CalculateDouble(double num1, double num2);
22      }
23  }

```

I gotta pull it together  
Don't wanna hurt anymore, I n-

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace IBinary
8  {
9      /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Add
13      */
14     public class Addition : IBinaryOperation
15     {
16         public int CalculateInt(int num1, int num2)
17         {
18             return num1 + num2;
19         }
20         public double CalculateDouble(double num1, double num2)
21         {
22             return num1 + num2;
23         }
24     }
25 }

```

Hold it in, cause I want a mai  
Some things I'm avoidin'

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace IBinary
8  {
9      /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Div
13      */
14     public class Division : IBinaryOperation
15     {
16         public int CalculateInt(int num1, int num2)
17         {
18             if (num2 == 0)
19             {
20                 throw new DivideByZeroException("Cannot divide by zero.");
21             }
22             return num1 / num2;
23         }
24         public double CalculateDouble(double num1, double num2)
25         {
26             if (num2 == 0)
27             {
28                 throw new DivideByZeroException("Cannot divide by zero.");
29             }
30             return num1 / num2;
31         }
32     }
33 }
```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace IBinary
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      * Log
13      */
14      public class Logarithm : IBinaryOperation
15      {
16          public int CalculateInt(int num1, int num2)
17          {
18              if (num1 <= 0 || num2 <= 0)
19              {
20                  throw new InvalidOperationException("The arguments must be positive.");
21              }
22
23              double result = Math.Log(num1, num2);
24              if (double.IsNaN(result))
25              {
26                  throw new InvalidOperationException("Invalid arguments for log operation.");
27              }
28
29              return (int)result;
30          }
31
32      }
33
34      public double CalculateDouble(double num1, double num2)
35      {
36          if (num1 <= 0 || num2 <= 0)
37          {
38              throw new ArgumentException("The arguments must be positive.");
39          }
40
41          return Math.Log(num1, num2);
42      }
43
44  }

```

Output window:

```

Show output from: Debug
'IBinaryClient.exe' (CLR v4.0.30319: IBinaryClient.exe): Loaded 'D:\VS\softwareDesign\simpleCalculator\calculatorClient\bin\Debug\IBinary.dll'. Symbols loaded.
The program '[9104] IBinaryClient.exe: Program Trace' has exited with code 0 (0x0).
The program '[9104] IBinaryClient.exe' has exited with code 3221225786 (0xc000013a).

```

```

1 //using System;
2 //using System.Collections.Generic;
3 //using System.Linq;
4 //using System.Text;
5 //using System.Threading.Tasks;
6
7 namespace IBinary
8 {
9     /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Mul
13      */
14     public class Multiplication : IBinaryOperation
15     {
16         /**
17          * reference | arbor, 23 days ago | 1 author, 2 changes
18          * Author: arbor, 23 days ago | 1 author, 1 change
19          */
20         public int CalculateInt(int num1, int num2)
21         {
22             return num1 * num2;
23         }
24         /**
25          * reference | arbor, 23 days ago | 1 author, 1 change
26          */
27         public double CalculateDouble(double num1, double num2)
28         {
29             return num1 * num2;
30         }
31     }
32 }

```

```

1 //using System;
2 //using System.Collections.Generic;
3 //using System.Linq;
4 //using System.Text;
5 //using System.Threading.Tasks;
6
7 namespace IBinary
8 {
9     /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Sub
13      */
14     public class Subtraction : IBinaryOperation
15     {
16         /**
17          * reference | arbor, 23 days ago | 1 author, 1 change
18          * Author: arbor, 23 days ago | 1 author, 1 change
19          */
20         public int CalculateInt(int num1, int num2)
21         {
22             return num1 - num2;
23         }
24         /**
25          * reference | arbor, 23 days ago | 1 author, 1 change
26          */
27         public double CalculateDouble(double num1, double num2)
28         {
29             return num1 - num2;
30         }
31     }
32 }

```

(2) 对于整数除法和浮点型数的除法，考虑分母为零的情况

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace IBinary
8  {
9      /**
10      * Author: arborshield
11      * Created by: on 2023/4/2.
12      * Div
13      */
14     public class Division : IBinaryOperation
15     {
16         public int CalculateInt(int num1, int num2)
17         {
18             if (num2 == 0)
19             {
20                 throw new DivideByZeroException("Cannot divide by zero.");
21             }
22
23             return num1 / num2;
24         }
25         public double CalculateDouble(double num1, double num2)
26         {
27             if (num2 == 0)
28             {
29                 throw new DivideByZeroException("Cannot divide by zero.");
30             }
31
32             return num1 / num2;
33         }
34     }
35 }

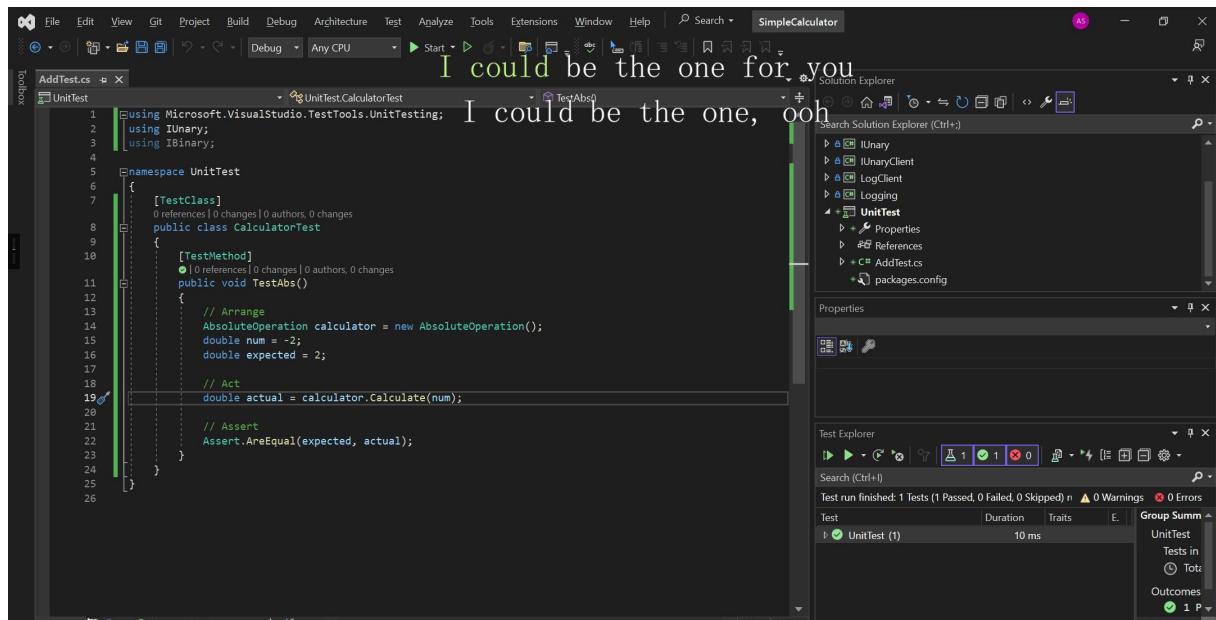
```

The screenshot shows the Visual Studio IDE with the following components:

- Solution Explorer:** Shows the project structure with files like Properties, References, Additions.cs, AddOperationFactory.cs, App.config, BinaryOperationFactory.cs, Divisions.cs, DivOperationFactory.cs, IBinaryOperation.cs, IBinaryOperationFactory.cs, Logarithm.cs, and Main.cs.
- Properties Window:** Shows file properties for Division.cs, including Build Action (Compile), Copy to Output Directory (Do not copy), and File Name (Division.cs).
- Output Window:** Displays build logs, including the loading of symbols for 'IBinaryClient.exe' and the exit of the program with code 0.
- Code Editor:** Shows the C# code for the Division class, which implements the IBinaryOperation interface. It includes two methods: CalculateInt and CalculateDouble, both of which include a check for division by zero.

(3) 基于 AAA 原则对于一元计算模块、二元计算模块和日志读写模块进行单元测试

一元：



I could be the one for you I could be the one, ooh

```

1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using IUnary;
3  using IBinary;
4
5  namespace UnitTest
6  {
7      [TestClass]
8      public class CalculatorTest
9      {
10         [TestMethod]
11         public void TestAbs()
12         {
13             // Arrange
14             AbsoluteOperation calculator = new AbsoluteOperation();
15             double num = -2;
16             double expected = 2;
17
18             // Act
19             double actual = calculator.Calculate(num);
20
21             // Assert
22             Assert.AreEqual(expected, actual);
23         }
24     }
25 }

```

Output

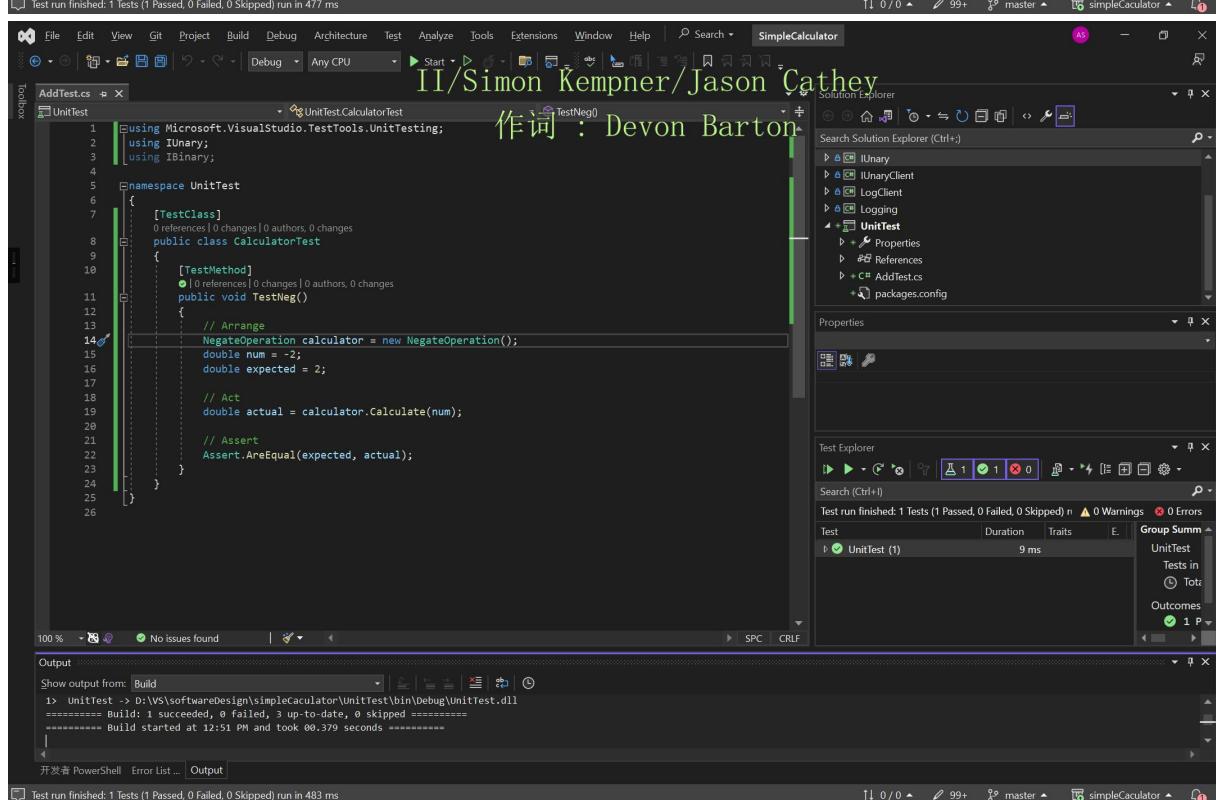
```

Show output from: Build
1> UnitTest -> D:\VS\softwareDesign\simpleCalculator\UnitTest\bin\Debug\UnitTest.dll
===== Build: 1 succeeded, 0 failed, 3 up-to-date, 0 skipped ======
===== Build started at 12:50 PM and took 00.086 seconds ======

```

开发者 PowerShell Error List Output

Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 477 ms



II/Simon Kempner/Jason Cathey 作词：Devon Barton

```

1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using IUnary;
3  using IBinary;
4
5  namespace UnitTest
6  {
7      [TestClass]
8      public class CalculatorTest
9      {
10         [TestMethod]
11         public void TestNeg()
12         {
13             // Arrange
14             NegateOperation calculator = new NegateOperation();
15             double num = -2;
16             double expected = 2;
17
18             // Act
19             double actual = calculator.Calculate(num);
20
21             // Assert
22             Assert.AreEqual(expected, actual);
23         }
24     }
25 }

```

Output

```

Show output from: Build
1> UnitTest -> D:\VS\softwareDesign\simpleCalculator\UnitTest\bin\Debug\UnitTest.dll
===== Build: 1 succeeded, 0 failed, 3 up-to-date, 0 skipped ======
===== Build started at 12:51 PM and took 00.379 seconds ======

```

开发者 PowerShell Error List ... Output

Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 483 ms

```

4
5     namespace UnitTest
6     {
7         [TestClass]
8         public class CalculatorTest
9         {
10             [TestMethod]
11             public void TestRec()
12             {
13                 // Arrange
14                 ReciprocalOperation calculator = new ReciprocalOperation();
15                 double num = -2;
16                 double expected = -0.5;
17
18                 // Act
19                 double actual = calculator.Calculate(num);
20
21                 // Assert
22                 Assert.AreEqual(expected, actual);
23             }
24
25             [TestMethod]
26             public void TestSqr()
27             {
28                 // Arrange
29                 SqrtOperation calculator = new SqrtOperation();
30                 double num = 4;
31                 double expected = 2;
32
33                 // Act
34                 double actual = calculator.Calculate(num);
35
36                 // Assert
37                 Assert.AreEqual(expected, actual);
38             }
39
40             [TestMethod]
41             public void TestSquare()
42             {
43                 // Arrange
44                 SquareOperation calculator = new SquareOperation();
45                 double num = -2;
46                 double expected = 4;
47
48                 // Act
49                 double actual = calculator.Calculate(num);
50
51                 // Assert
52                 Assert.AreEqual(expected, actual);
53             }
54         }
55     }

```

I'm always runnin' into you  
I hate the situation

Output

```

1>  UnitTest -> D:\VS\softwareDesign\simpleCalculator\UnitTest\bin\Debug\UnitTest.dll
===== Build: 1 succeeded, 0 failed, 6 up-to-date, 0 skipped ======
===== Build started at 12:55 PM and took 01.166 seconds ======

```

Test run finished: 3 Tests (3 Passed, 0 Failed, 0 Skipped) run in 499 ms

Don't wanna feel the way I do  
Just what I'm goin' through

Output

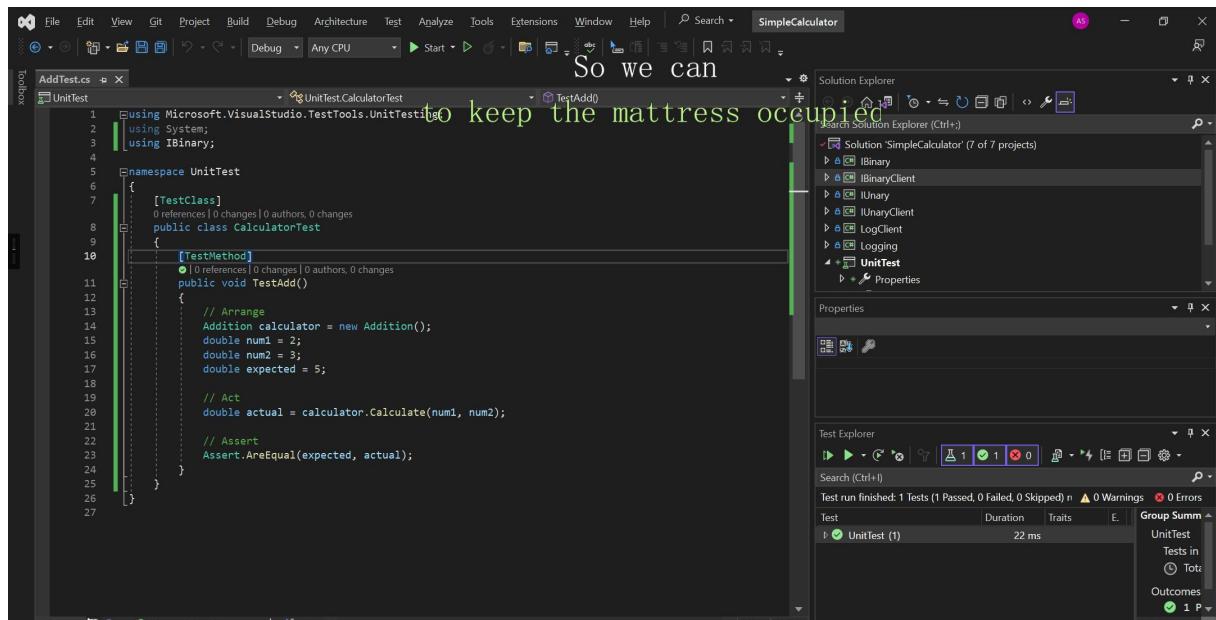
```

1>  UnitTest -> D:\VS\softwareDesign\simpleCalculator\UnitTest\bin\Debug\UnitTest.dll
===== Build: 1 succeeded, 0 failed, 6 up-to-date, 0 skipped ======
===== Build started at 12:55 PM and took 01.166 seconds ======

```

Test run finished: 3 Tests (3 Passed, 0 Failed, 0 Skipped) run in 499 ms

二元：



So we can  
to keep the mattress occupied

```

1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using System;
3  using IBinary;
4
5  namespace UnitTest
6  {
7      [TestClass]
8          0 references | 0 authors, 0 changes
9      public class CalculatorTest
10     {
11         [TestMethod]
12             0|0 references | 0 authors, 0 changes
13         public void TestAdd()
14         {
15             // Arrange
16             Addition calculator = new Addition();
17             double num1 = 2;
18             double num2 = 3;
19             double expected = 5;
20
21             // Act
22             double actual = calculator.Calculate(num1, num2);
23
24             // Assert
25             Assert.AreEqual(expected, actual);
26         }
27     }

```

Output

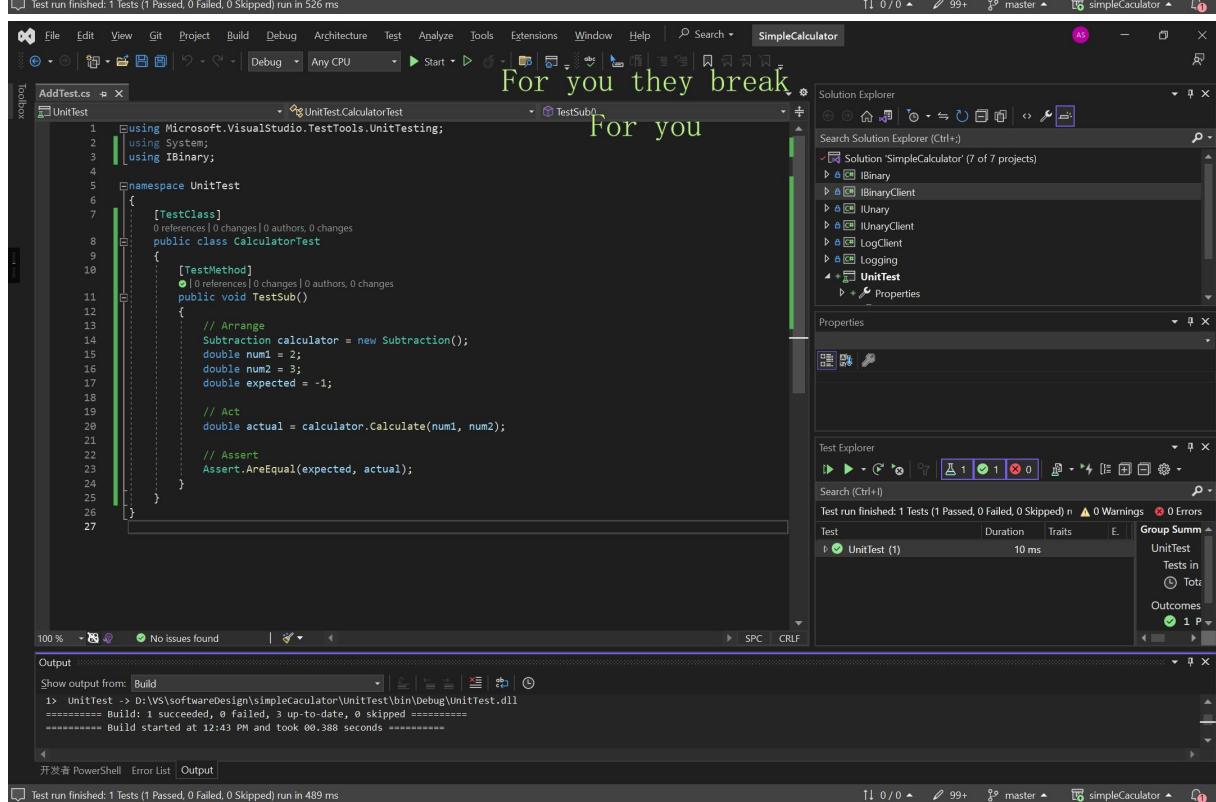
```

Show output from: Build
1> UnitTest -> D:\VS\softwareDesign\simpleCalculator\UnitTest\bin\Debug\UnitTest.dll
===== Build: 1 succeeded, 0 failed, 3 up-to-date, 0 skipped ======
===== Build started at 12:40 PM and took 02.485 seconds ======

```

开发者 PowerShell Error List Output

Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 526 ms



For you they break  
For you

```

1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using System;
3  using IBinary;
4
5  namespace UnitTest
6  {
7      [TestClass]
8          0 references | 0 authors, 0 changes
9      public class CalculatorTest
10     {
11         [TestMethod]
12             0|0 references | 0 authors, 0 changes
13         public void TestSub()
14         {
15             // Arrange
16             Subtraction calculator = new Subtraction();
17             double num1 = 2;
18             double num2 = 3;
19             double expected = -1;
20
21             // Act
22             double actual = calculator.Calculate(num1, num2);
23
24             // Assert
25             Assert.AreEqual(expected, actual);
26         }
27     }

```

Output

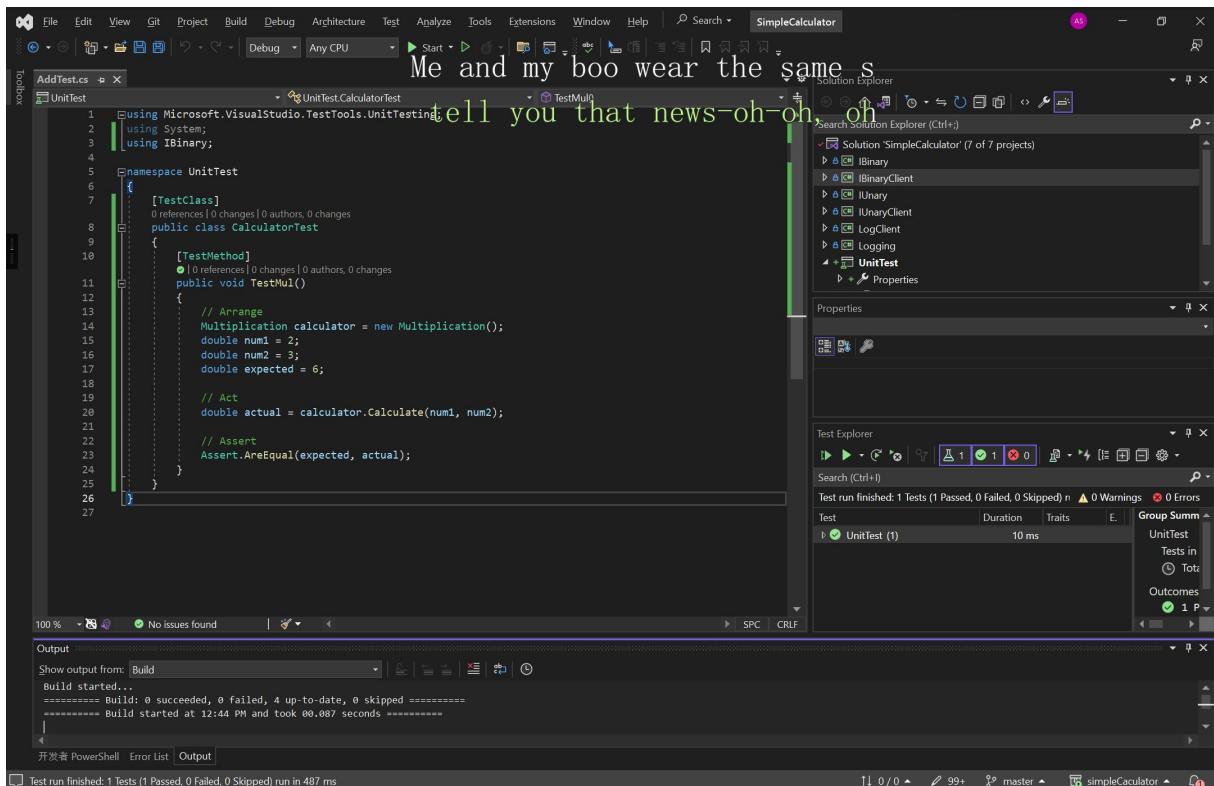
```

Show output from: Build
1> UnitTest -> D:\VS\softwareDesign\simpleCalculator\UnitTest\bin\Debug\UnitTest.dll
===== Build: 1 succeeded, 0 failed, 3 up-to-date, 0 skipped ======
===== Build started at 12:43 PM and took 00.388 seconds ======

```

开发者 PowerShell Error List Output

Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 489 ms



The screenshot shows two instances of Visual Studio running side-by-side. Both instances have the same solution structure:

- Solution Explorer:** Shows 7 projects: IBinary, IBinaryClient, IUnary, IUnaryClient, Logging, UnitTest, and Properties.
- Test Explorer:** Shows 1 test passed in the UnitTest project.
- Output Window:** Displays build logs indicating successful builds.
- Code Editor:** Displays C# unit test code for division and logarithm calculations.

**Top Instance (Division Test):**

```

using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using IBinary;

namespace UnitTest
{
    [TestClass]
    public class CalculatorTest
    {
        [TestMethod]
        public void TestDiv()
        {
            // Arrange
            Division calculator = new Division();
            double num1 = 3;
            double num2 = 3;
            double expected = 1;

            // Act
            double actual = calculator.Calculate(num1, num2);

            // Assert
            Assert.AreEqual(expected, actual);
        }
    }
}

```

**Bottom Instance (Logarithm Test):**

```

using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using IBinary;

namespace UnitTest
{
    [TestClass]
    public class CalculatorTest
    {
        [TestMethod]
        public void TestLog()
        {
            // Arrange
            Logarithm calculator = new Logarithm();
            double num1 = 2;
            double num2 = 4;
            double expected = 0.5;

            // Act
            double actual = calculator.Calculate(num1, num2);

            // Assert
            Assert.AreEqual(expected, actual);
        }
    }
}

```

日志：

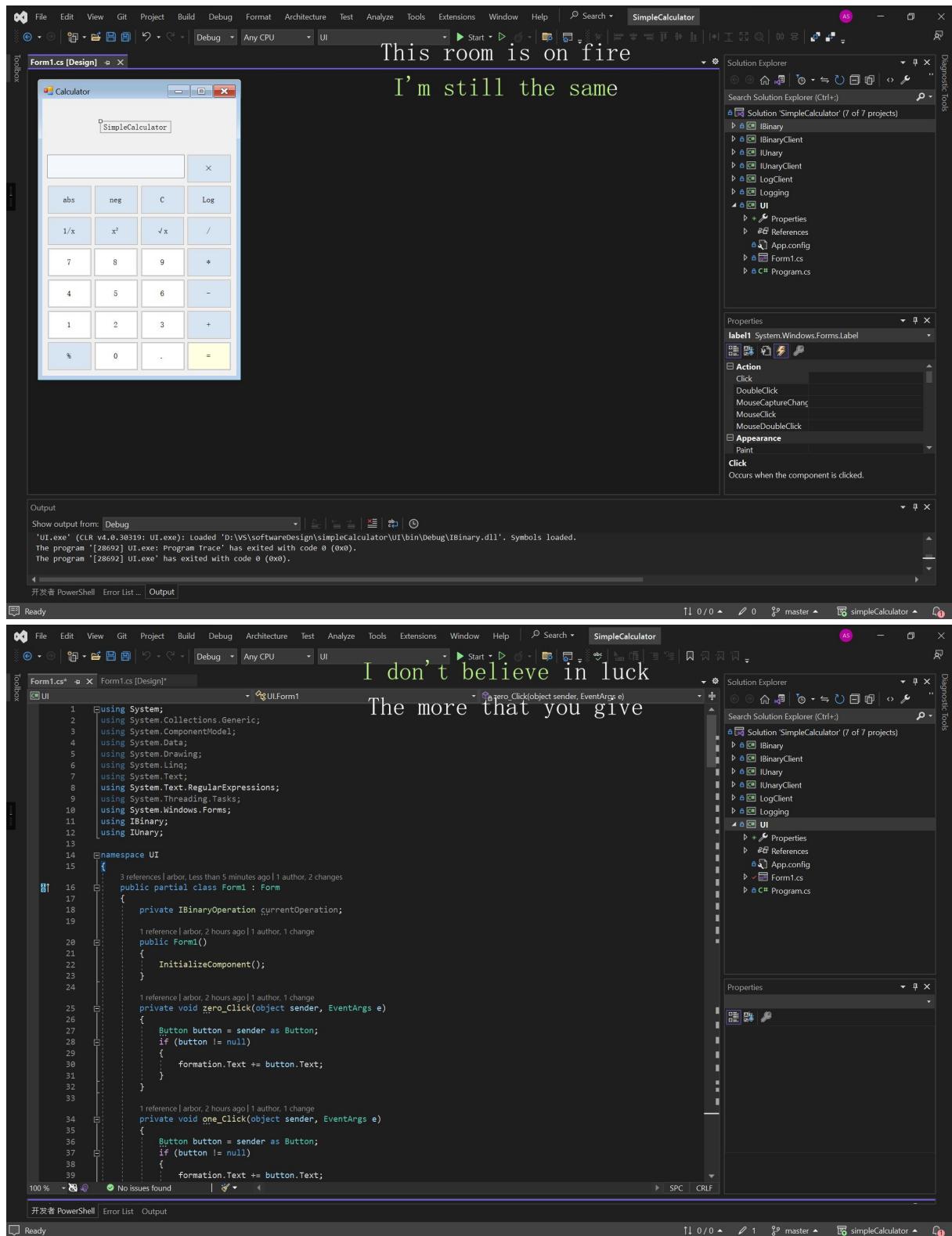
The screenshot shows the Visual Studio IDE interface with the following details:

- Code Editor:** Displays the `LogTest.cs` file under the `UnitTest` project. The code implements a unit test for a `FileLogger` class, reading logs from a file and asserting the message content.
- Solution Explorer:** Shows the solution structure with files like `Properties`, `References`, `App.config`, `FileLogger.cs`, `ILogger.cs`, `LoggerFactory.cs`, and the `UnitTest` project itself.
- Properties Window:** Shows standard properties for the selected file.
- Test Explorer:** Shows the test results: 1 Test Passed, 0 Failed, 0 Skipped, 2 Warnings, 0 Errors.
- Output Window:** Shows the build log output.

Annotations in the code editor area:

- "Keeping myself busy, tryna fix" is written above the code editor.
- "but I can no longer pretend" is written below the code editor.
- "So what do I do?" is written above the code editor in the second screenshot.
- "But everytime I try, somehow" is written below the code editor in the second screenshot.

(4) 参考 windows 自带计算器实现计算器的 UI 界面，在 UI 界面中使用之前实现的一元计算机和二元计算库



```

    // Top Window (Screenshot 1)
    So go head and say it
    Yes I've been jaded

    // Bottom Window (Screenshot 2)
    I got my ways
    But what's how we made it
  
```

Both screenshots show the following C# code for a Windows Form application:

```

using System;
using System.Windows.Forms;

public partial class Form1 : Form
{
    private void zero_Click(object sender, EventArgs e)
    {
        formation.Text += "0";
    }

    private void one_Click(object sender, EventArgs e)
    {
        formation.Text += "1";
    }

    private void two_Click(object sender, EventArgs e)
    {
        formation.Text += "2";
    }

    private void three_Click(object sender, EventArgs e)
    {
        formation.Text += "3";
    }

    private void four_Click(object sender, EventArgs e)
    {
        formation.Text += "4";
    }

    private void five_Click(object sender, EventArgs e)
    {
        formation.Text += "5";
    }

    private void six_Click(object sender, EventArgs e)
    {
        formation.Text += "6";
    }

    private void seven_Click(object sender, EventArgs e)
    {
        formation.Text += "7";
    }

    private void eight_Click(object sender, EventArgs e)
    {
        formation.Text += "8";
    }

    private void nine_Click(object sender, EventArgs e)
    {
        formation.Text += "9";
    }
}
  
```

The code is identical in both windows, with the exception of the text displayed above the code area.

Had nothing in between  
Before we had these issues

```

116     Button button = sender as Button;
117     if (button != null)
118     {
119         formation.Text += button.Text;
120     }
121
122 }
123
124
125     private void add_Click(object sender, EventArgs e)
126     {
127         Button button = sender as Button;
128         if (button != null)
129         {
130             formation.Text += button.Text;
131         }
132     }
133
134     private void sub_Click(object sender, EventArgs e)
135     {
136         Button button = sender as Button;
137         if (button != null)
138         {
139             formation.Text += button.Text;
140         }
141     }
142
143     private void mul_Click(object sender, EventArgs e)
144     {
145         Button button = sender as Button;
146         if (button != null)
147         {
148             formation.Text += button.Text;
149         }
150     }
151
152     private void div_Click(object sender, EventArgs e)
153     {
154         Button button = sender as Button;
155         if (button != null)
156         {
157             formation.Text += button.Text;
158         }
159     }
160
161     private void zero_Click(object sender, EventArgs e)
162     {
163         Button button = sender as Button;
164         if (button != null)
165         {
166             formation.Text += button.Text;
167         }
168     }
169
170     private void equal_Click(object sender, EventArgs e)
171     {
172         // Get the expression in the text box
173         string expression = formation.Text;
174
175         // Define regular expressions to extract numbers and operators
176         string pattern = @"(\d+\.?\d*)([+\-*/\^]\d*)";
177
178         // Use regular expressions to match expressions and extract numbers and operators
179         Match match = Regex.Match(expression, pattern);
180
181         if (match.Success)
182         {
183             // Extract the first digit, the second digit, and the operator
184             double num1 = double.Parse(match.Groups[1].Value);
185             double num2 = double.Parse(match.Groups[3].Value);
186             string op = match.Groups[2].Value;
187
188             // Create the corresponding IBinaryOperation object from the operator
189             IBinaryOperation binaryOperation = null;
190             switch (op)
191             {
192                 case "+":
193                     binaryOperation = new Addition();
194             }
195         }
196     }
197
198 }
199
200 }
```

Properties

开发者 PowerShell Error List Output

Ready 0/0 1 master simpleCalculator

When love was so easy  
They way it used to be

```

154     Button button = sender as Button;
155     if (button != null)
156     {
157         formation.Text += button.Text;
158     }
159
160
161     private void log_Click(object sender, EventArgs e)
162     {
163         Button button = sender as Button;
164         if (button != null)
165         {
166             formation.Text += button.Text;
167         }
168     }
169
170     private void equal_Click(object sender, EventArgs e)
171     {
172         // Get the expression in the text box
173         string expression = formation.Text;
174
175         // Define regular expressions to extract numbers and operators
176         string pattern = @"(\d+\.?\d*)([+\-*/\^]\d*)";
177
178         // Use regular expressions to match expressions and extract numbers and operators
179         Match match = Regex.Match(expression, pattern);
180
181         if (match.Success)
182         {
183             // Extract the first digit, the second digit, and the operator
184             double num1 = double.Parse(match.Groups[1].Value);
185             double num2 = double.Parse(match.Groups[3].Value);
186             string op = match.Groups[2].Value;
187
188             // Create the corresponding IBinaryOperation object from the operator
189             IBinaryOperation binaryOperation = null;
190             switch (op)
191             {
192                 case "+":
193                     binaryOperation = new Addition();
194                 | |
195             }
196         }
197     }
198
199 }
200
201 }
```

Properties

开发者 PowerShell Error List Output

Ready 0/0 1 master simpleCalculator

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Form1.cs` in the `UI` project. The code implements a calculator application with various operations like addition, subtraction, multiplication, division, and logarithms. It also handles reciprocal and square calculations. The `Solution Explorer` pane on the right shows the solution structure with multiple projects: `IBinary`, `iBinaryClient`, `IUnary`, `UnaryClient`, `LogClient`, `Logging`, and the `UI` project which contains `App.config`, `Form1.cs`, and `Program.cs`. The `Properties` pane is also visible.

```
194     binaryOperation = new Addition();
195     break;
196     case "-":
197     binaryOperation = new Subtraction();
198     break;
199     case "*":
200     binaryOperation = new Multiplication();
201     break;
202     case "/":
203     binaryOperation = new Division();
204     break;
205     case "Log":
206     binaryOperation = new Logarithm();
207     break;
208   }
209
210   // If a match is found, calculate the result
211   if (binaryOperation != null)
212   {
213     double result = binaryOperation.Calculate(num1, num2);
214
215     // Write the result back into the text box
216     formation.Text = result.ToString();
217   }
218 }
219
220
221 1 reference | arbor, 2 hours ago | 1 author, 1 change
222 private void rec_Click(object sender, EventArgs e)
223 {
224   double inputNum;
225   if (double.TryParse(formation.Text, out inputNum))
226   {
227     IUnaryOperation operation = new ReciprocalOperation();
228     double result = operation.Calculate(inputNum);
229     formation.Text = result.ToString();
230   }
231 }
232
233
234 1 reference | arbor, 2 hours ago | 1 author, 1 change
235 private void square_Click(object sender, EventArgs e)
236 {
237   double inputNum;
```

**Top Screenshot (Dark Theme):**

```

1 reference | arbor, 2 hours ago | 1 author, 1 change
private void zero_Click(object sender, EventArgs e)
{
    double inputNum;
    if (double.TryParse(formation.Text, out inputNum))
    {
        IUnaryOperation operation = new SquareOperation();
        double result = operation.Calculate(inputNum);
        formation.Text = result.ToString();
    }
}

1 reference | arbor, 2 hours ago | 1 author, 1 change
private void sqrt_Click(object sender, EventArgs e)
{
    double inputNum;
    if (double.TryParse(formation.Text, out inputNum))
    {
        IUnaryOperation operation = new SquareRootOperation();
        double result = operation.Calculate(inputNum);
        formation.Text = result.ToString();
    }
}

1 reference | arbor, 2 hours ago | 1 author, 1 change
private void abs_Click(object sender, EventArgs e)
{
    double inputNum;
    if (double.TryParse(formation.Text, out inputNum))
    {
        IUnaryOperation operation = new AbsoluteOperation();
        double result = operation.Calculate(inputNum);
        formation.Text = result.ToString();
    }
}

1 reference | arbor, 2 hours ago | 1 author, 1 change
private void neg_Click(object sender, EventArgs e)
{
    double inputNum;
    if (double.TryParse(formation.Text, out inputNum))
    {
        IUnaryOperation operation = new NegateOperation();
        double result = operation.Calculate(inputNum);
        formation.Text = result.ToString();
    }
}

```

**Bottom Screenshot (Light Theme):**

```

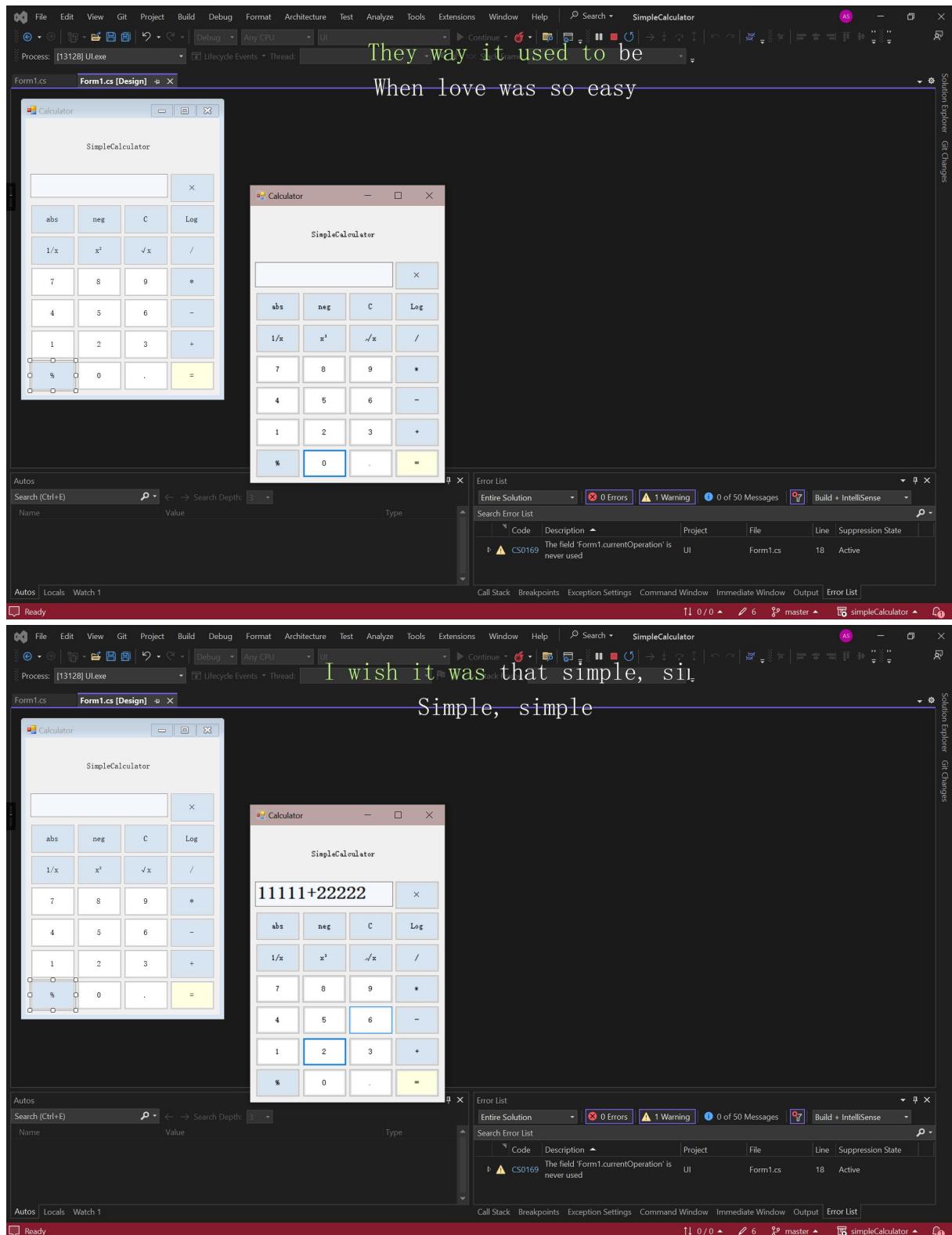
1 reference | arbor, 2 hours ago | 1 author, 1 change
private void clear_Click(object sender, EventArgs e)
{
    formation.Text = "";
}

1 reference | arbor, 2 hours ago | 1 author, 1 change
private void del_Click(object sender, EventArgs e)
{
    if (formation.Text.Length > 0)
    {
        formation.Text = formation.Text.Substring(0, formation.Text.Length - 1);
    }
}

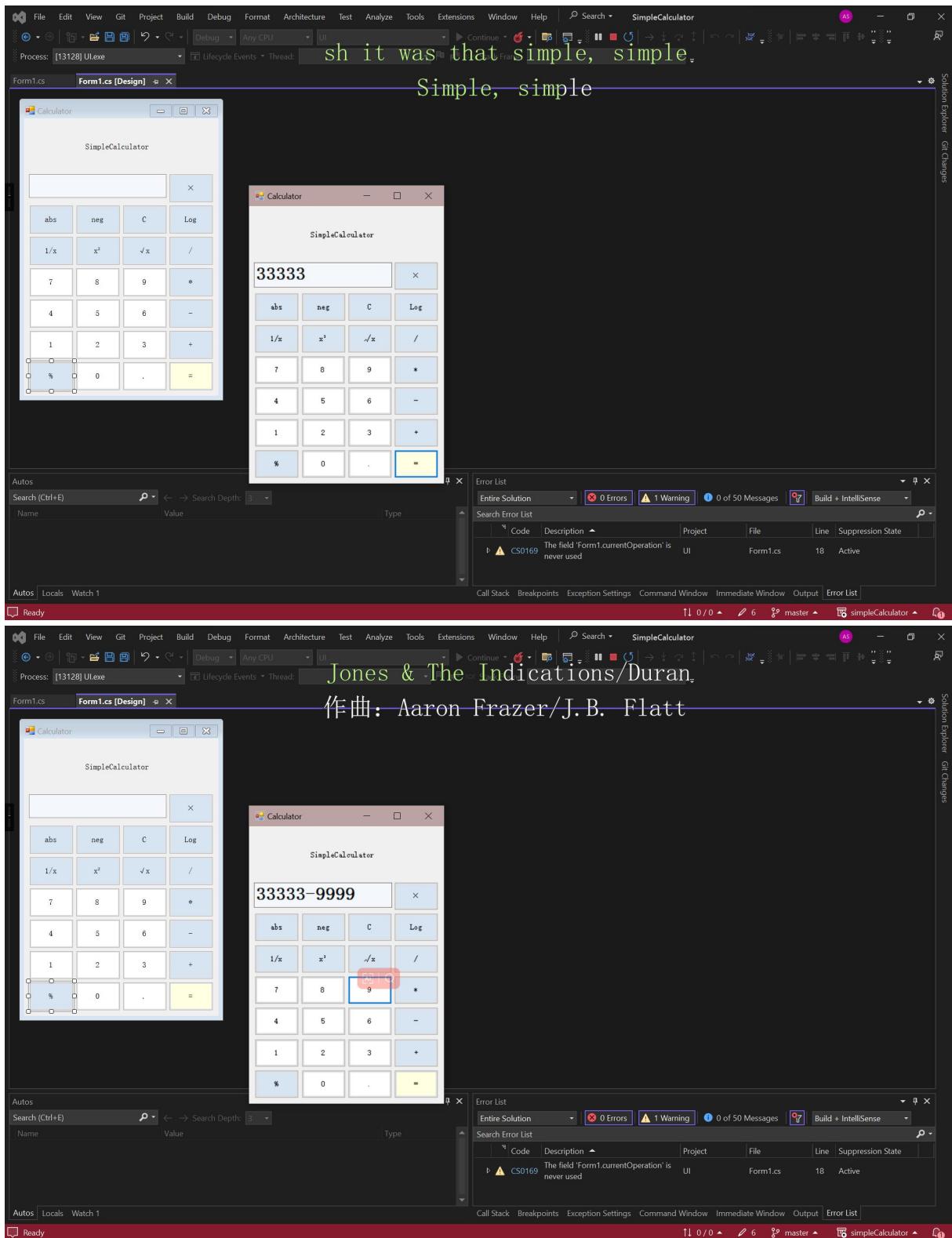
1 reference | arbor, 2 hours ago | 1 author, 1 change
private void pgr_Click(object sender, EventArgs e)
{
    double num;
    if (double.TryParse(formation.Text, out num))
    {
        num /= 100;
        formation.Text = num.ToString();
    }
}

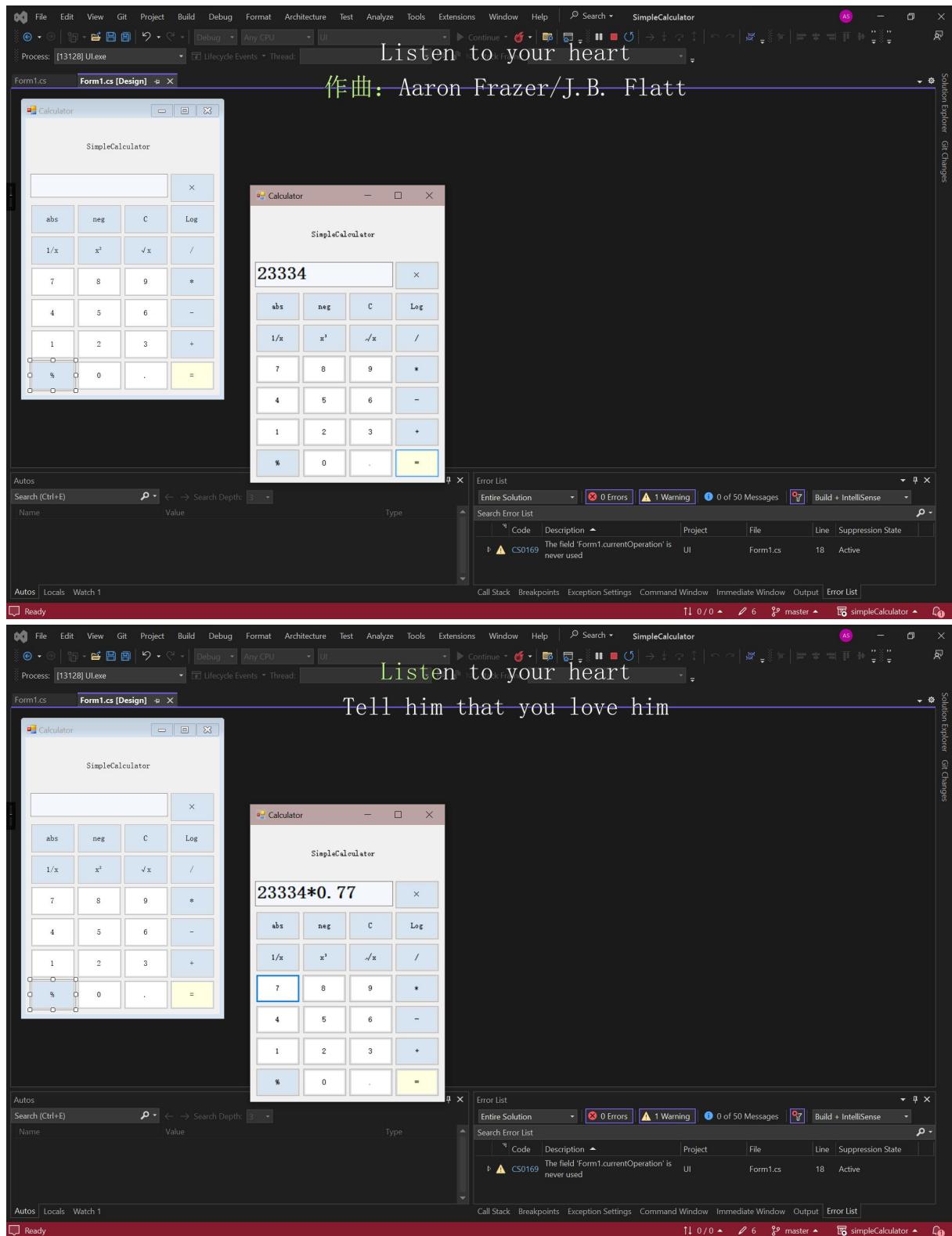
```

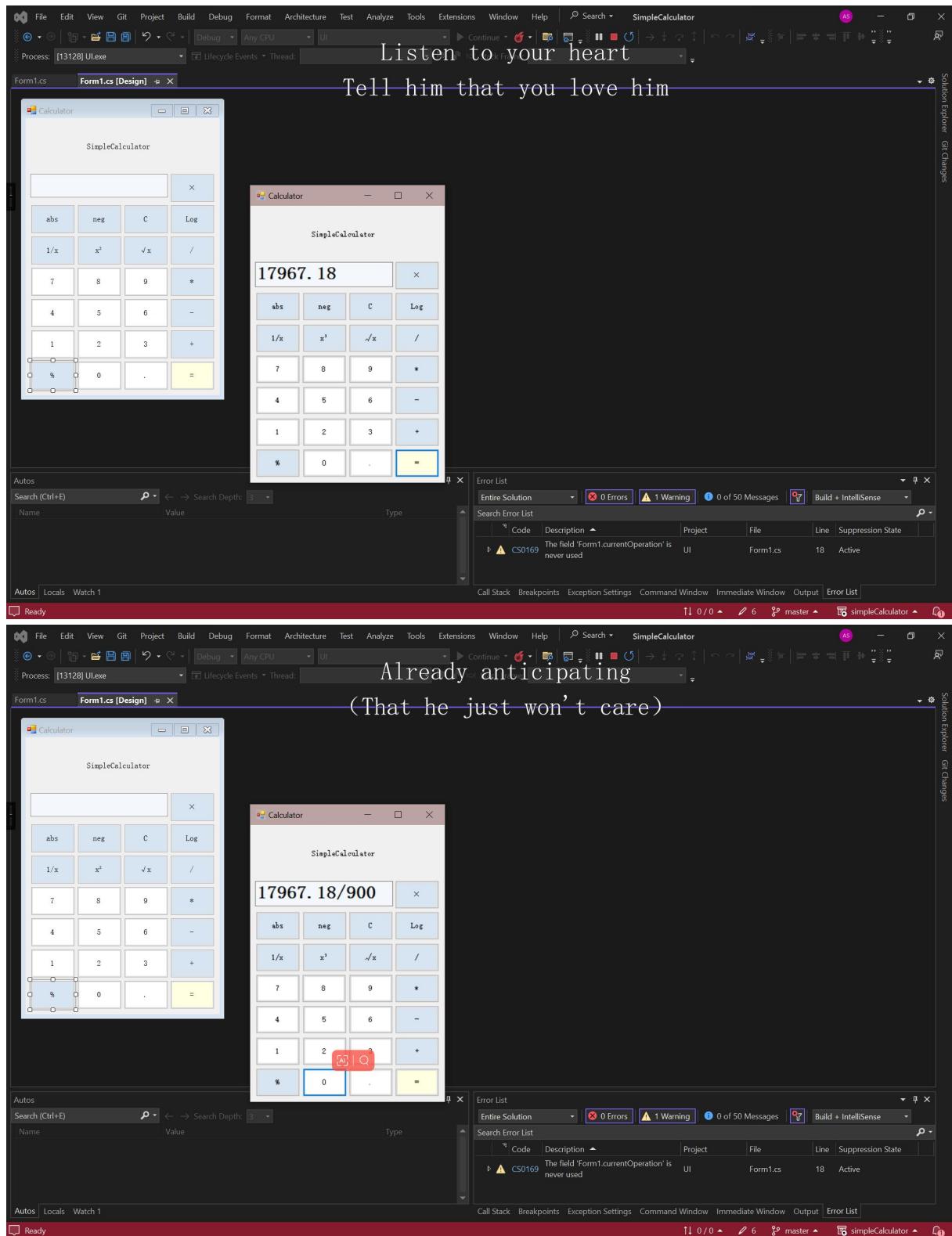
## 四、运行效果

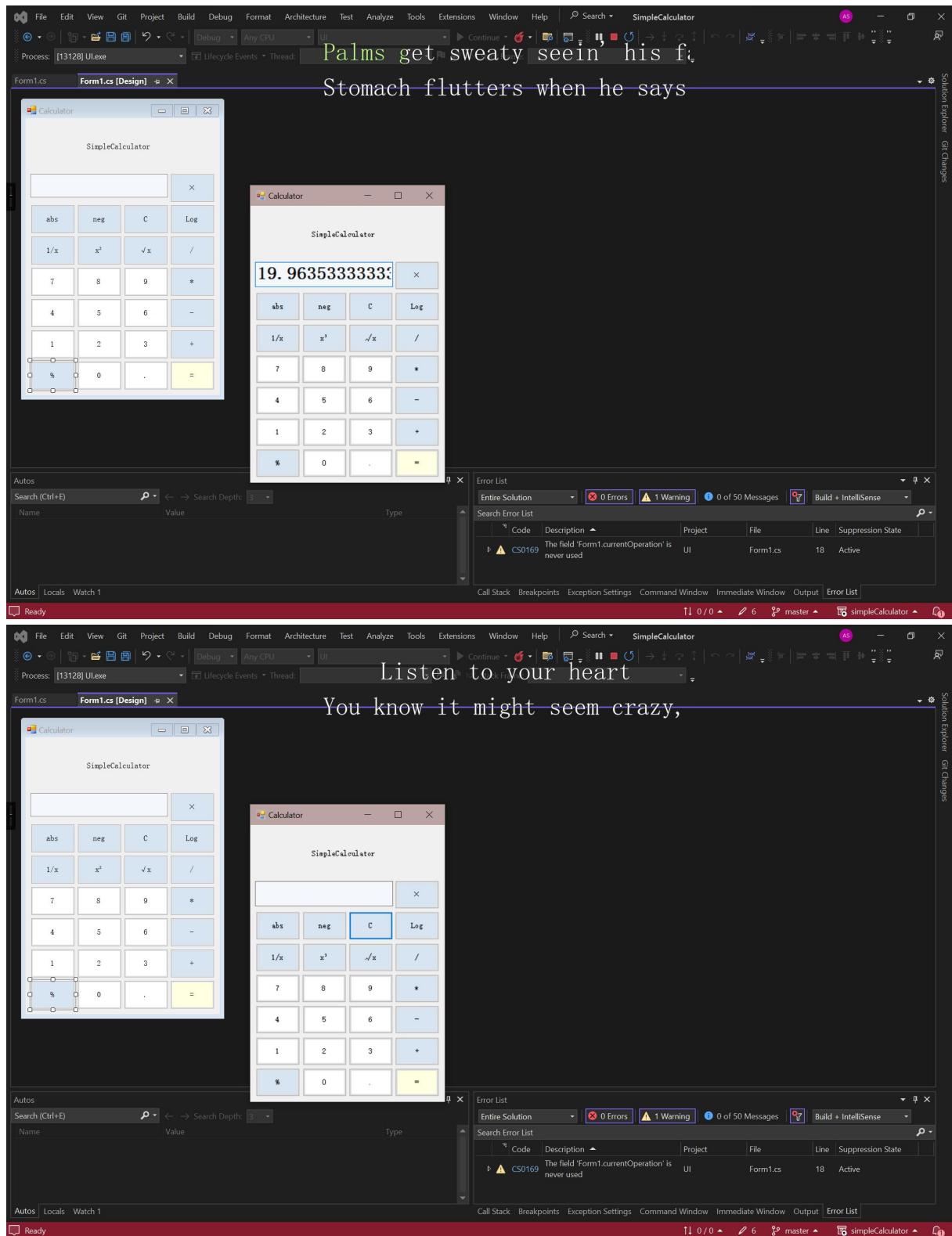


# 《软件设计与规范实验报告》

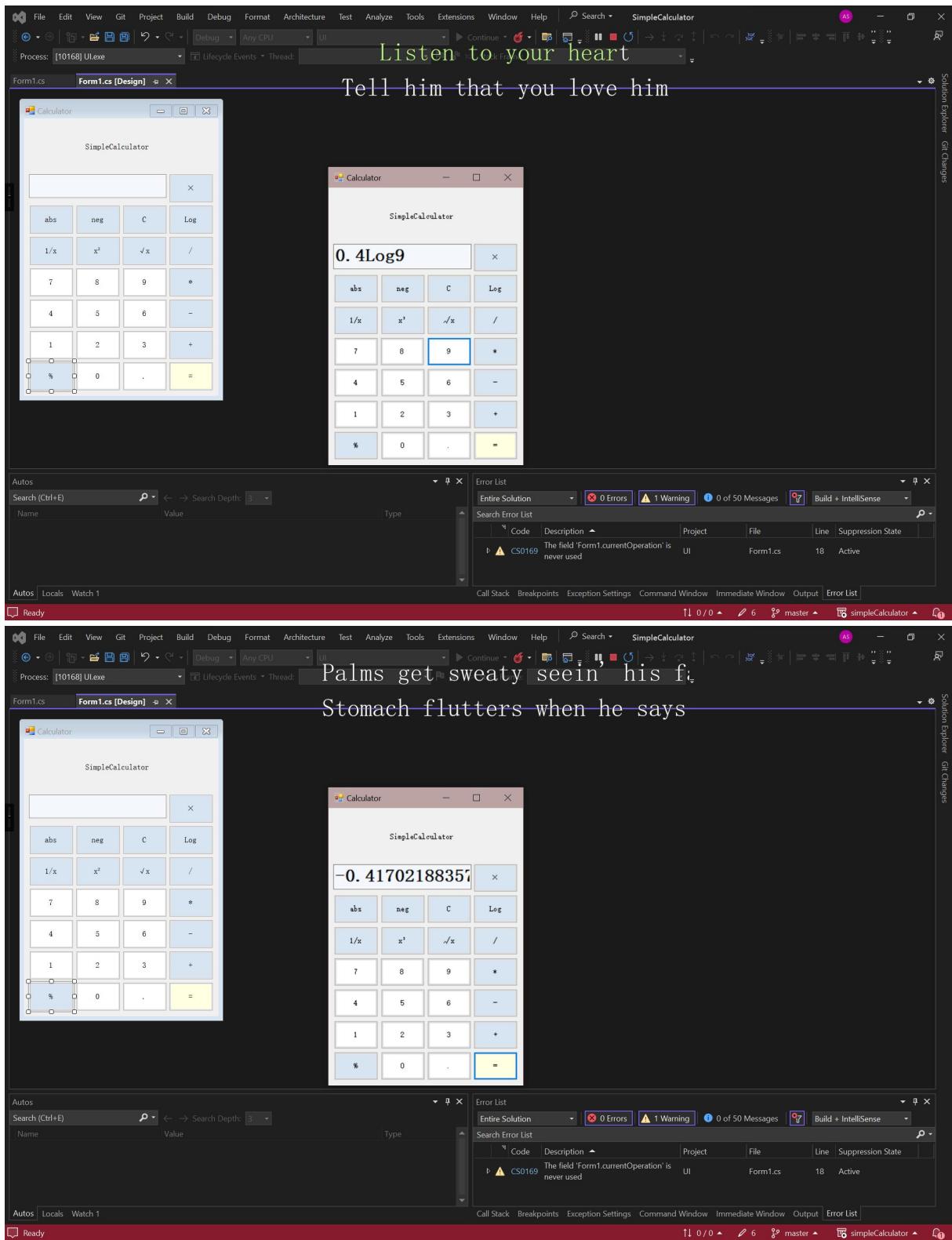




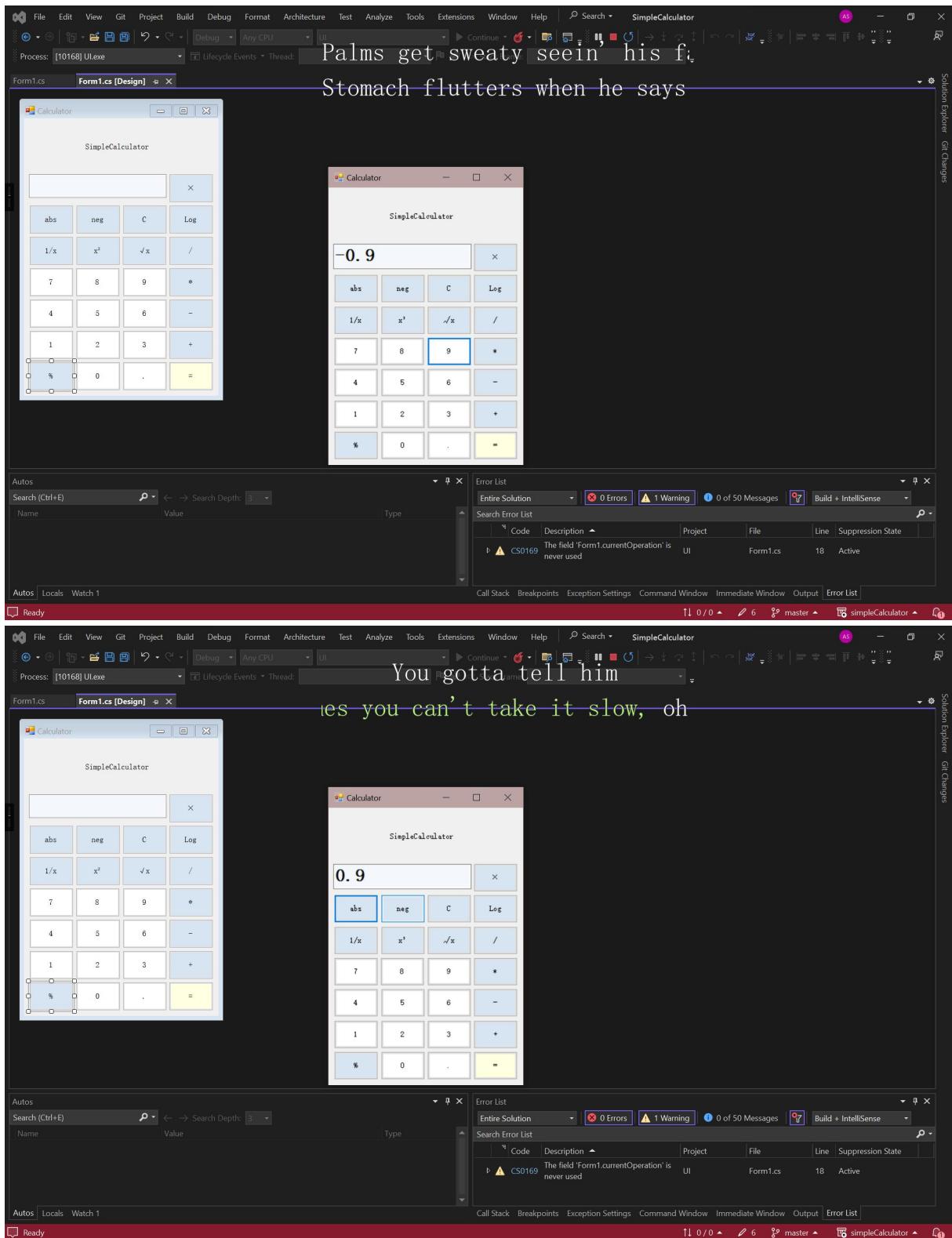


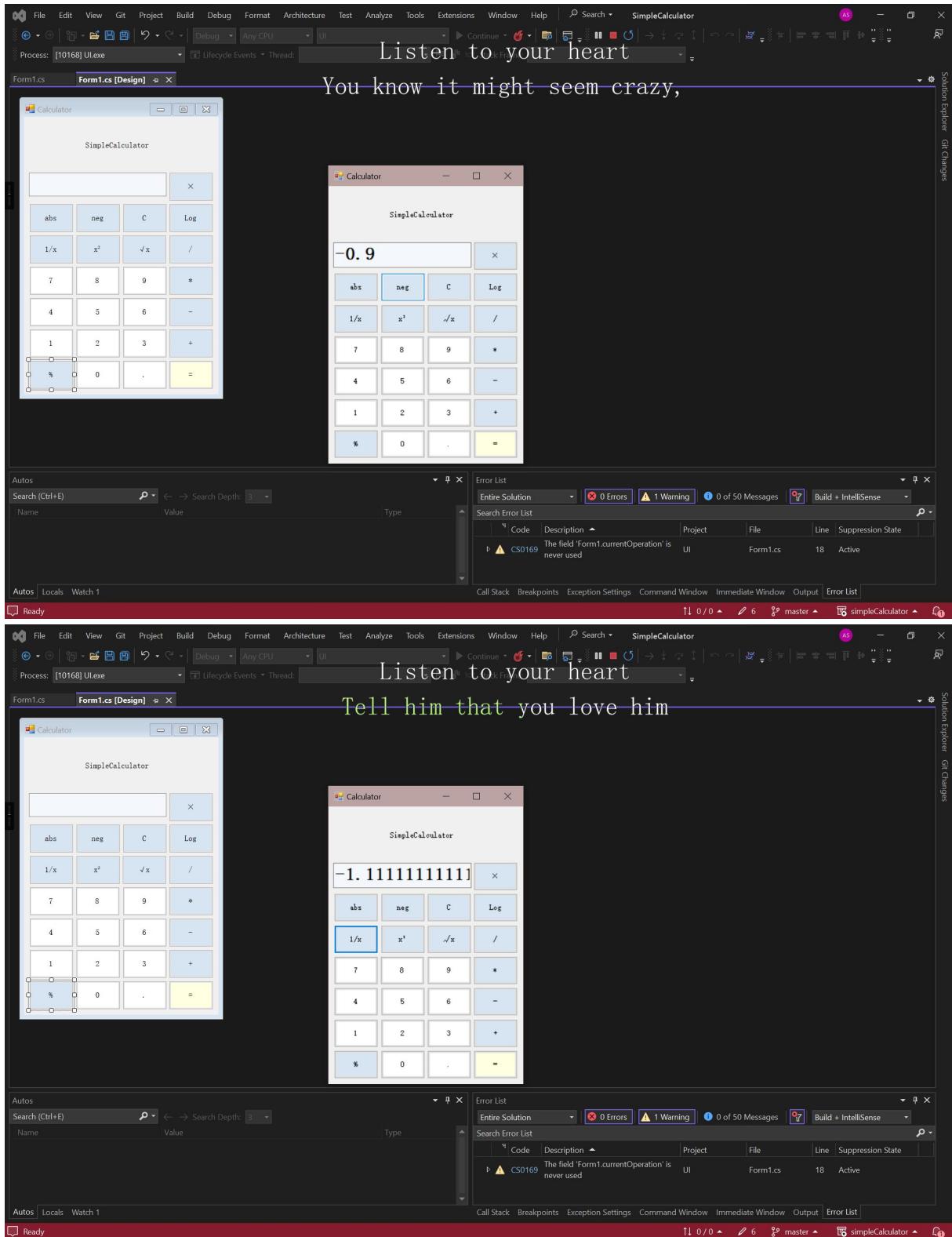


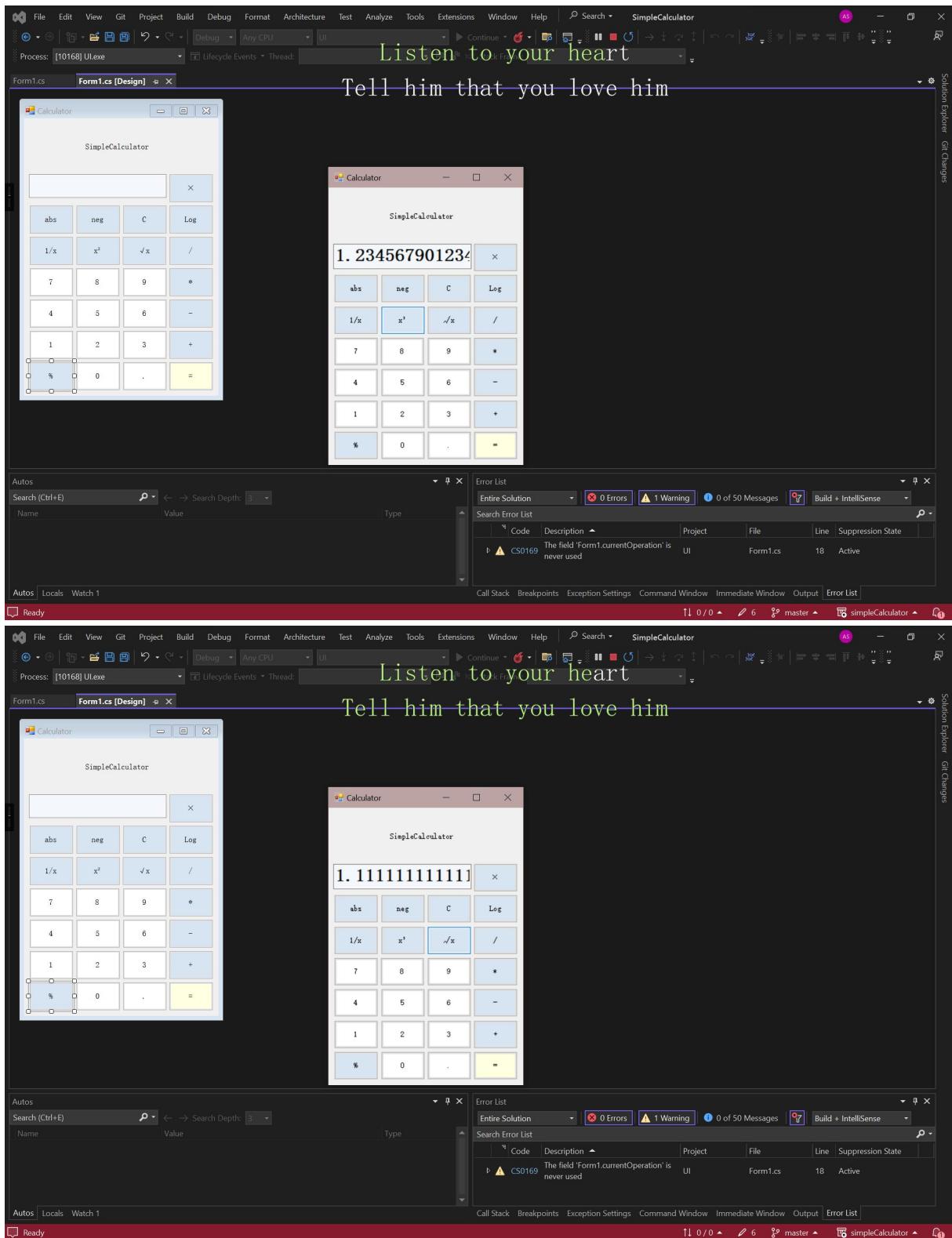
## 《软件设计与规范实验报告》

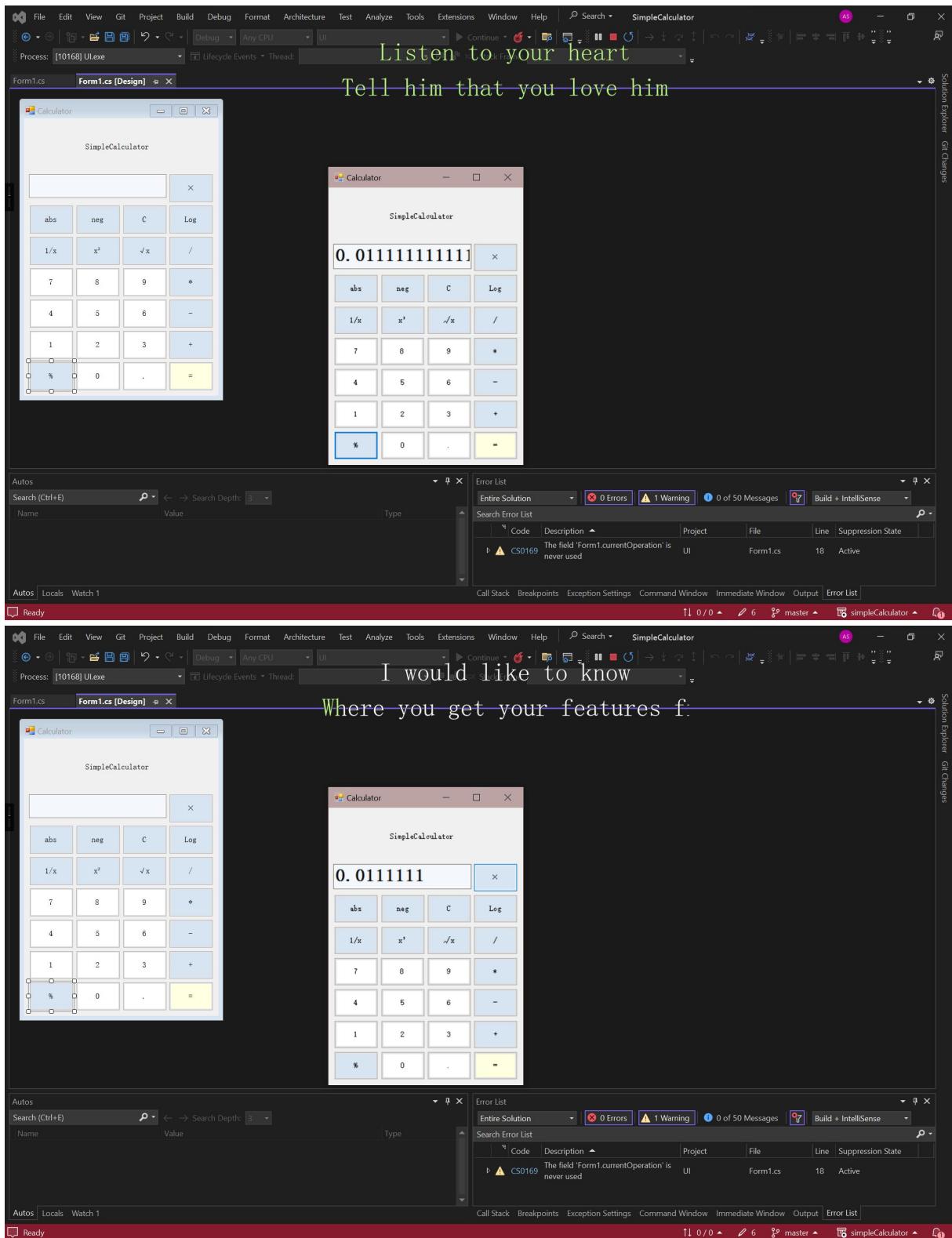


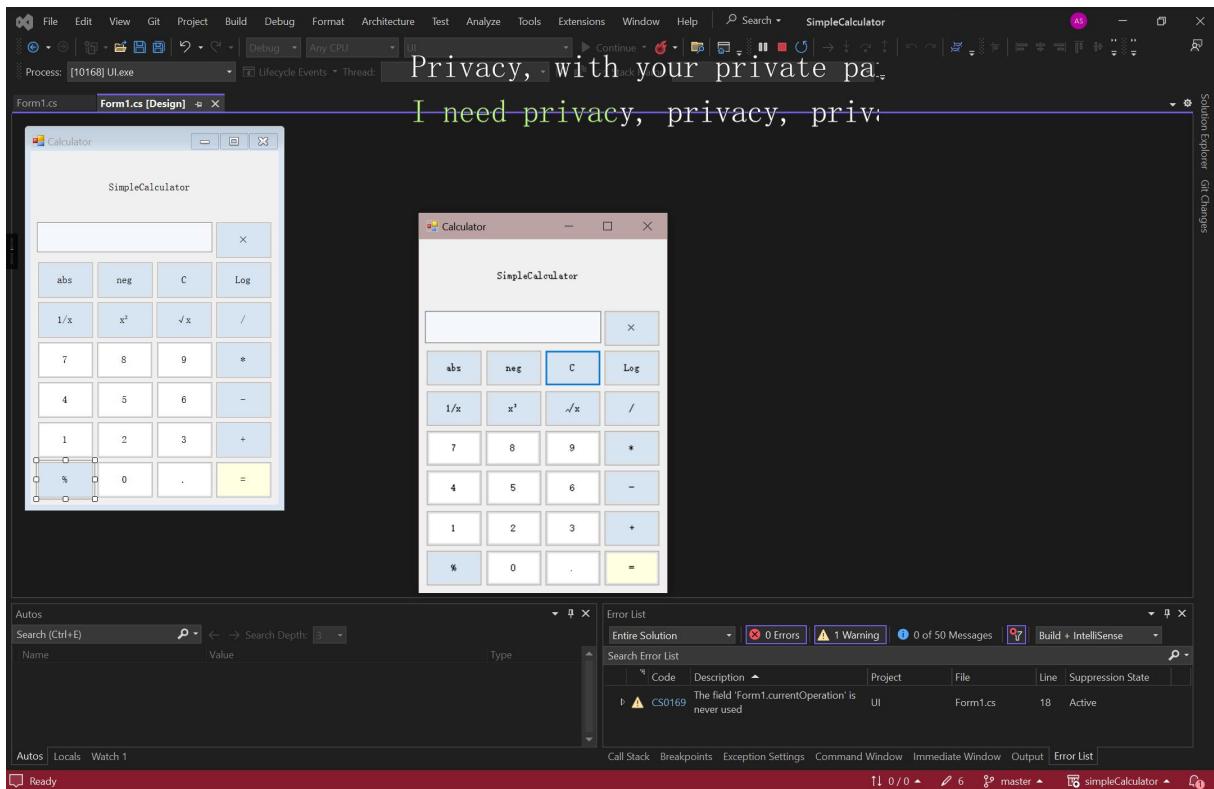
# 《软件设计与规范实验报告》











## 五、实验心得

在这次实验中，我通过编写一个简单的计算器程序来学习了 C# 编程语言和 Windows 窗体应用程序开发。我学习了如何使用接口和面向对象编程概念来设计程序，并使用正则表达式和异常处理来实现程序的一些核心功能。此外，我还学习了一些基本的 Windows 窗体控件，并使用它们来构建您的计算器用户界面。

在实验的过程中，我遇到了一些问题，例如如何使用正则表达式匹配字符串中的操作符、如何在用户输入不合法时进行异常处理等等。通过解决这些问题，我对 C# 编程语言和 Windows 窗体应用程序开发的理解和掌握程度得到了提高。

总的来说，这次实验为我提供了一个良好的学习机会，让我更深入地了解了 C# 编程语言和 Windows 窗体应用程序开发。

