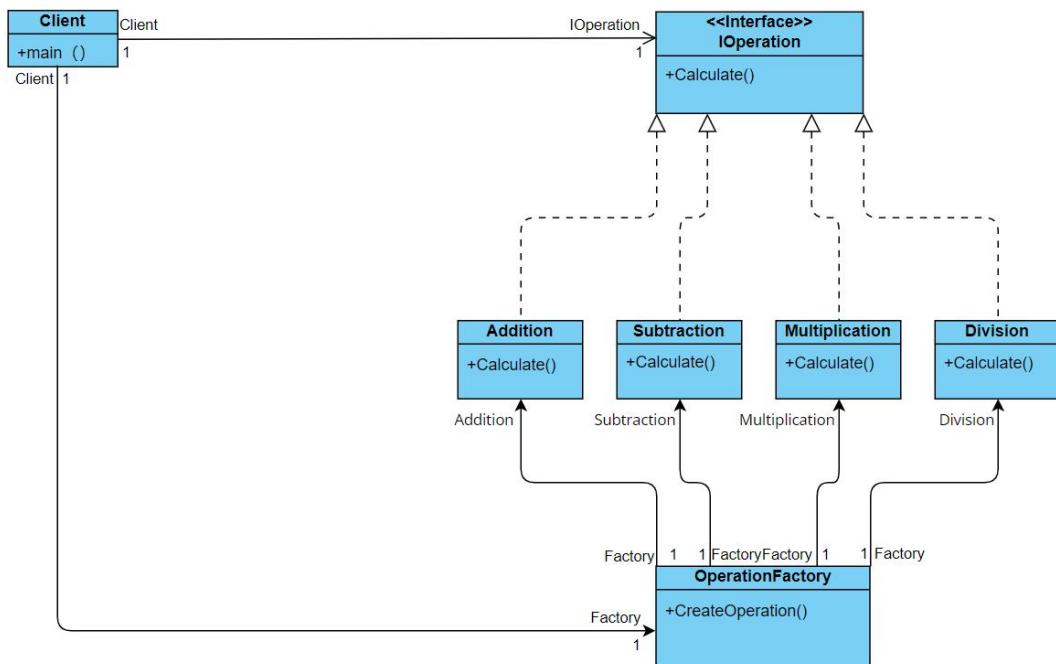


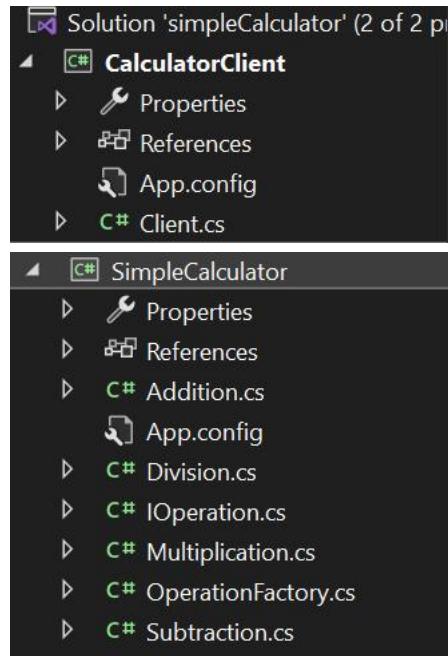
作业 1———计算机

(1) 采用静态工程模式设计实现简易计算器，实现加减乘除运算

①类图



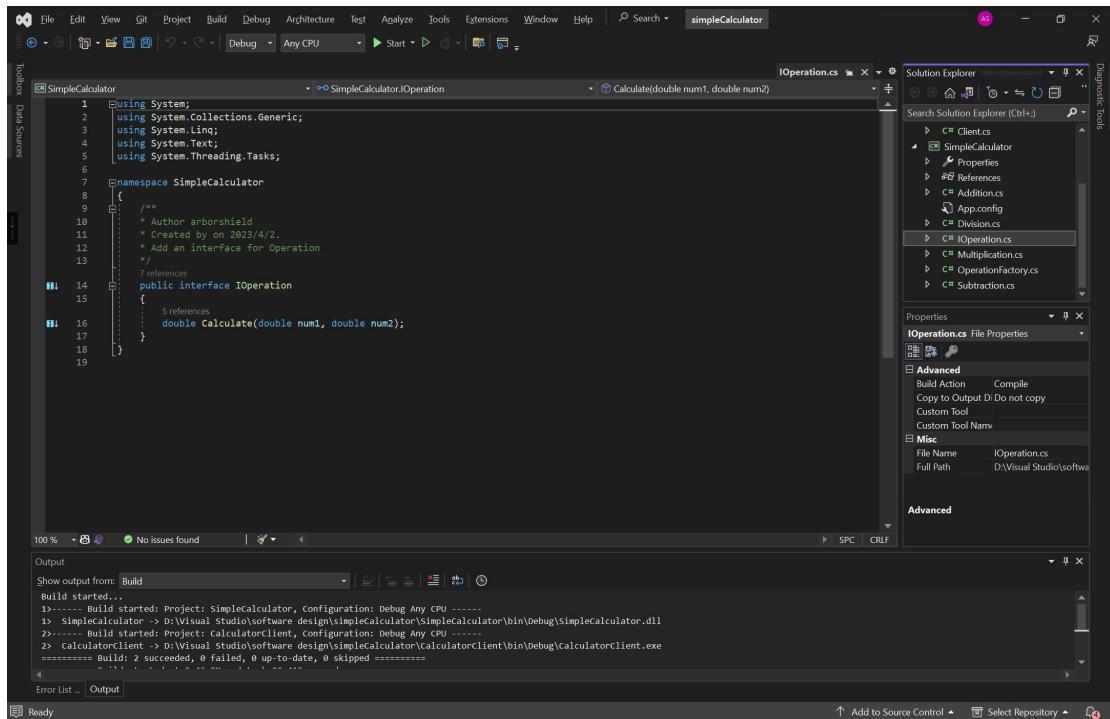
②代码结构



③核心类代码

Project SimpleCalculator

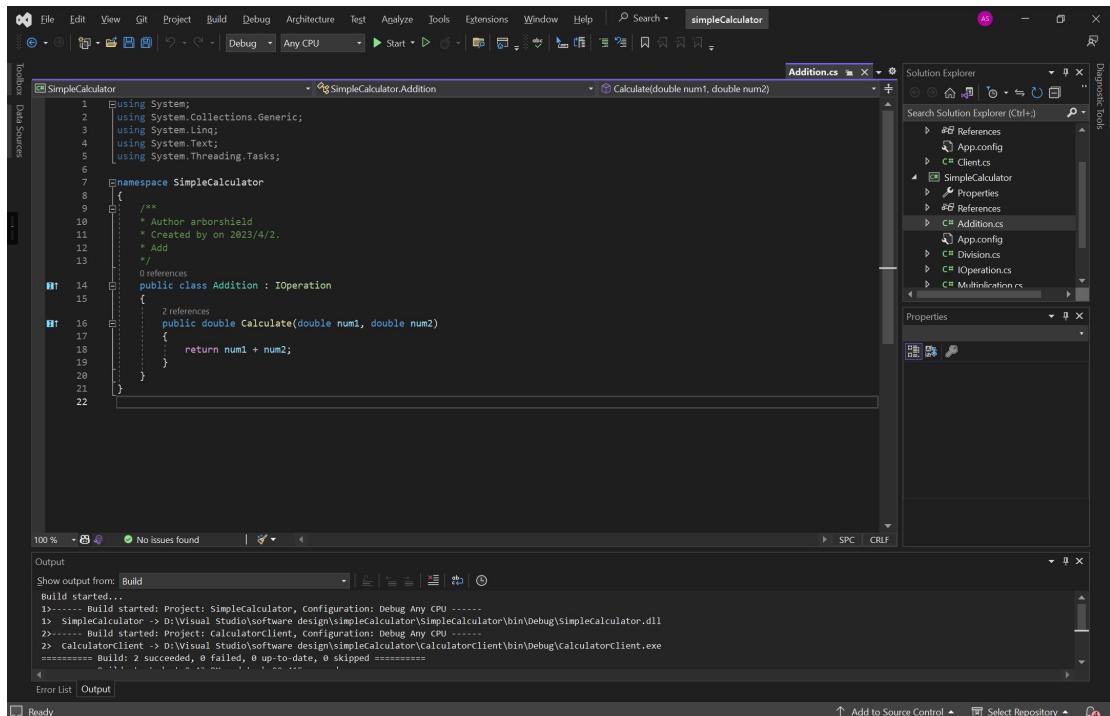
Interface IOperation



```
1 //using System;
2 //using System.Collections.Generic;
3 //using System.Linq;
4 //using System.Text;
5 //using System.Threading.Tasks;
6
7 namespace SimpleCalculator
8 {
9     /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      * Add an interface for Operation
13      */
14     public interface IOperation
15     {
16         double Calculate(double num1, double num2);
17     }
18 }
```

Output
Show output from: Build
Build started...
1>----- Build started: Project: simplecalculator, configuration: Debug Any CPU -----
1> simplecalculator > D:\Visual Studio\software design\simplecalculator\bin\debug\simplecalculator.dll
2>----- Build started: Project: calculatorclient, configuration: Debug Any CPU -----
2> calculatorclient > D:\Visual Studio\software design\simplecalculator\calculatorclient\bin\debug\calculatorclient.exe
===== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ======

Class Addition



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

Output
Show output from: Build
Build started...
1>----- Build started: Project: SimpleCalculator, Configuration: Debug Any CPU -----
1> SimpleCalculator > D:\Visual Studio\software design\simplecalculator\SimpleCalculator\bin\Debug\SimpleCalculator.dll
2>----- Build started: Project: calculatorclient, Configuration: Debug Any CPU -----
2> calculatorclient > D:\Visual Studio\software design\simplecalculator\calculatorclient\bin\debug\calculatorclient.exe
===== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ======

Class Subtraction

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      */
13
14      public class Subtraction : IOperation
15      {
16          public double Calculate(double num1, double num2)
17          {
18              return num1 - num2;
19          }
20      }
21  }

```

The Solution Explorer shows files: Addition.cs, Division.cs, IOperation.cs, Multiplication.cs, OperationFactory.cs, and Subtraction.cs. The Properties pane shows Subtraction.cs has a Build Action of 'Compile' and a Copy to Output of 'Do not copy'.

Class Multiplication

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      */
13
14      public class Multiplication : IOperation
15      {
16          public double Calculate(double num1, double num2)
17          {
18              return num1 * num2;
19          }
20      }
21  }

```

The Solution Explorer shows files: Addition.cs, Division.cs, IOperation.cs, Multiplication.cs, OperationFactory.cs, and Subtraction.cs. The Properties pane shows Multiplication.cs has a Build Action of 'Compile' and a Copy to Output of 'Do not copy'.

Class Division

The screenshot shows the Visual Studio IDE interface. The main area displays the code for `Division.cs` in the `SimpleCalculator` namespace. The code implements the `IOperation` interface with a `Calculate` method that performs division and handles division by zero. The Solution Explorer on the right shows the project structure with files like `Division.cs`, `IOperation.cs`, `Multiplication.cs`, `OperationFactory.cs`, and `Subtraction.cs`. The Properties window is open for `Division.cs`, showing build actions and other settings. The Output window at the bottom shows a successful build process.

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

Output

```
1>----- Build started: Project: simplecalculator, Configuration: Debug Any CPU -----
1> Simplecalculator > D:\Visual Studio\software design\simplecalculator\bin\debug\simplecalculator.dll
2>----- Build started: Project: calculatorclient, Configuration: Debug Any CPU -----
2> calculatorclient > D:\Visual Studio\software design\simplecalculator\calculatorclient\bin\debug\calculatorclient.exe
===== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

Class OperationFactory

```
1 using System;
2 using System.Configuration;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace SimpleCalculator
9 {
10     /**
11      * Author: arborschild
12      * Created by on 2023/4/2.
13      * Operation Factory
14      */
15     public static class OperationFactory
16     {
17         reference
18         public static IOperation CreateOperation(string operationName)
19         {
20             string typeName = ConfigurationManager.AppSettings[operationName];
21
22             if (string.IsNullOrEmpty(typeName))
23             {
24                 throw new ArgumentException($"Invalid operation name: {operationName}");
25             }
26
27             Type type = Type.GetType(typeName);
28
29             if (type == null)
30             {
31                 throw new ArgumentException($"Invalid operation type: {typeName}");
32             }
33
34             return (IOperation)Activator.CreateInstance(type);
35         }
36     }
37 }
```

No issues found

Output

```
Build started.
1>----- Build started: Project: simplecalculator, Configuration: Debug Any CPU -----
1> Simplecalculator > D:\Visual Studio\software design\simplecalculator\bin\debug\simplecalculator.dll
2>----- Build started: Project: calculatorclient, Configuration: Debug Any CPU -----
2> Calculatorclient > D:\Visual Studio\software design\simplecalculatorcalculatorclient\bin\debug\calculatorclient.exe
===== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ======
```

Error List | Output

```
1 using System;
2 using System.Configuration;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace SimpleCalculator
9 {
10     /**
11      * Author: arborschild
12      * Created by on 2023/4/2.
13      * Operation Factory
14      */
15     public static class OperationFactory
16     {
17         reference
18         public static IOperation CreateOperation(string operationName)
19         {
20             string typeName = ConfigurationManager.AppSettings[operationName];
21
22             if (string.IsNullOrEmpty(typeName))
23             {
24                 throw new ArgumentException($"Invalid operation name: {operationName}");
25             }
26
27             Type type = Type.GetType(typeName);
28
29             if (type == null)
30             {
31                 throw new ArgumentException($"Invalid operation type: {typeName}");
32             }
33
34             return (IOperation)Activator.CreateInstance(type);
35         }
36     }
37 }
```

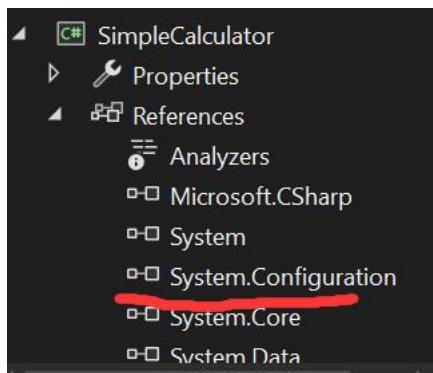
No issues found

Output

```
Build started.
Build started.
1>----- Build started: Project: simplecalculator, Configuration: Debug Any CPU -----
1> Simplecalculator > D:\Visual Studio\software design\simplecalculator\bin\debug\simplecalculator.dll
2>----- Build started: Project: calculatorclient, Configuration: Debug Any CPU -----
2> Calculatorclient > D:\Visual Studio\software design\simplecalculatorcalculatorclient\bin\debug\calculatorclient.exe
===== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ======
```

Error List | Output

④客户端代码



Project CalculatorClient

Class Client

The screenshot shows the Visual Studio IDE interface. The main window displays the `Client.cs` file content:

```
1  using SimpleCalculator;
2  using System;
3  using System.Configuration;
4
5  namespace CalculatorClient
6  {
7      /**
8       * Author: arborschild
9       * Created by: 2023/4/2.
10      * ClientTest
11
12      0 references
13
14      class Client
15      {
16          0 references
17          static void Main(string[] args)
18          {
19              try
20              {
21                  Console.WriteLine("Enter the first number: ");
22                  double num1 = Convert.ToDouble(Console.ReadLine());
23
24                  Console.WriteLine("Enter the second number: ");
25                  double num2 = Convert.ToDouble(Console.ReadLine());
26
27                  Console.WriteLine("Enter the operation (+, -, *, /): ");
28                  string operationName = Console.ReadLine();
29
30                  IOperation operation = OperationFactory.CreateOperation(operationName);
31
32                  double result = operation.Calculate(num1, num2);
33
34                  Console.WriteLine($"The result is: {result}");
35
36              }
37              catch (Exception ex)
38              {
39                  Console.WriteLine($"An error occurred: {ex.Message}");
40              }
41
42          }
43      }
44
45  }
```

The Solution Explorer pane on the right shows the project structure:

- Solution: simpleCalculator (2 of 2 projects)
- CalculatorClient:
 - Properties
 - References
 - App.config
 - C# Client.cs
- SimpleCalculator:
 - Properties
 - References
 - App.config
 - C# Additions.cs
 - Ann.conf

The Properties pane for `Client.cs` shows the following settings:

- Advanced:
 - Build Action: Compile
 - Copy to Output D: Do not copy
 - Custom Tool
 - Custom Tool Name
- Misc:
 - File Name: Client.cs
 - Full Path: D:\Visual Studio\software\simplecalculator\CalculatorClient\CalculatorClient.cs

The screenshot shows the Visual Studio IDE interface. The main window displays the `Client.cs` file content, identical to the one in the previous screenshot:

```
1  using SimpleCalculator;
2  using System;
3  using System.Configuration;
4
5  namespace CalculatorClient
6  {
7      /**
8       * Author: arborschild
9       * Created by: 2023/4/2.
10      * ClientTest
11
12      0 references
13
14      class Client
15      {
16          0 references
17          static void Main(string[] args)
18          {
19              try
20              {
21                  Console.WriteLine("Enter the first number: ");
22                  double num1 = Convert.ToDouble(Console.ReadLine());
23
24                  Console.WriteLine("Enter the second number: ");
25                  double num2 = Convert.ToDouble(Console.ReadLine());
26
27                  Console.WriteLine("Enter the operation (+, -, *, /): ");
28                  string operationName = Console.ReadLine();
29
30                  IOperation operation = OperationFactory.CreateOperation(operationName);
31
32                  double result = operation.Calculate(num1, num2);
33
34                  Console.WriteLine($"The result is: {result}");
35
36              }
37              catch (Exception ex)
38              {
39                  Console.WriteLine($"An error occurred: {ex.Message}");
40              }
41
42          }
43      }
44
45  }
```

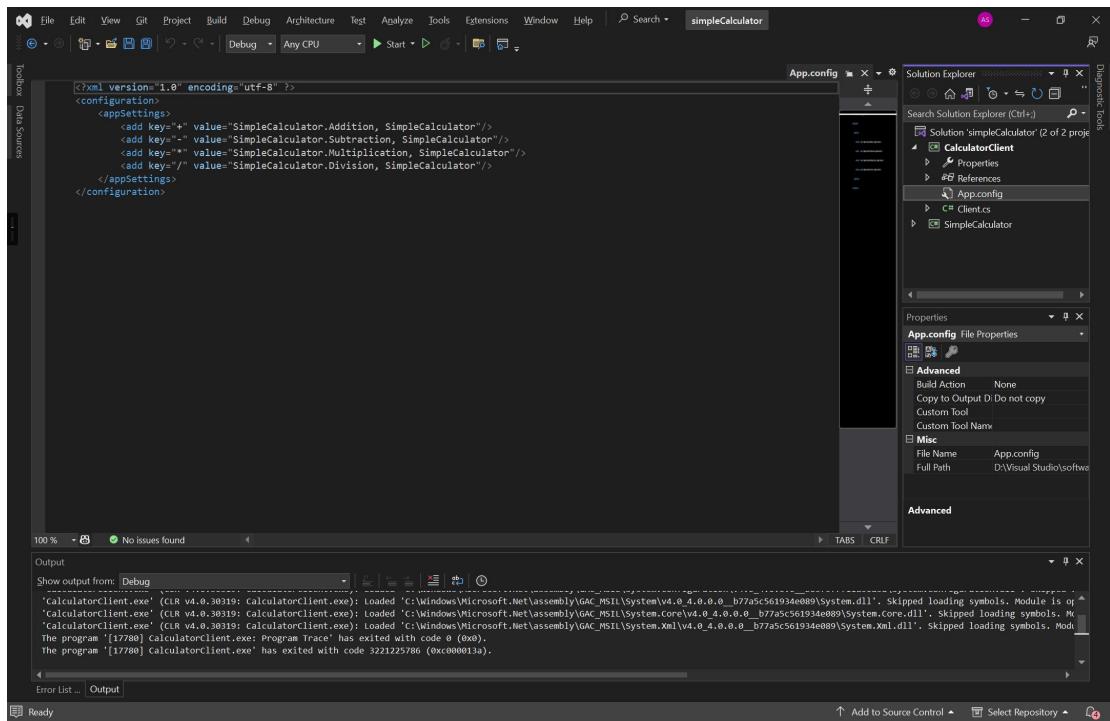
The Solution Explorer pane on the right shows the project structure:

- Solution: simpleCalculator (2 of 2 projects)
- CalculatorClient:
 - Properties
 - References
 - App.config
 - C# Client.cs
- SimpleCalculator:
 - Properties
 - References
 - App.config
 - C# Additions.cs
 - Ann.conf

The Properties pane for `Client.cs` shows the following settings:

- Advanced:
 - Build Action: Compile
 - Copy to Output D: Do not copy
 - Custom Tool
 - Custom Tool Name
- Misc:
 - File Name: Client.cs
 - Full Path: D:\Visual Studio\software\simplecalculator\CalculatorClient\CalculatorClient.cs

⑤配置文件



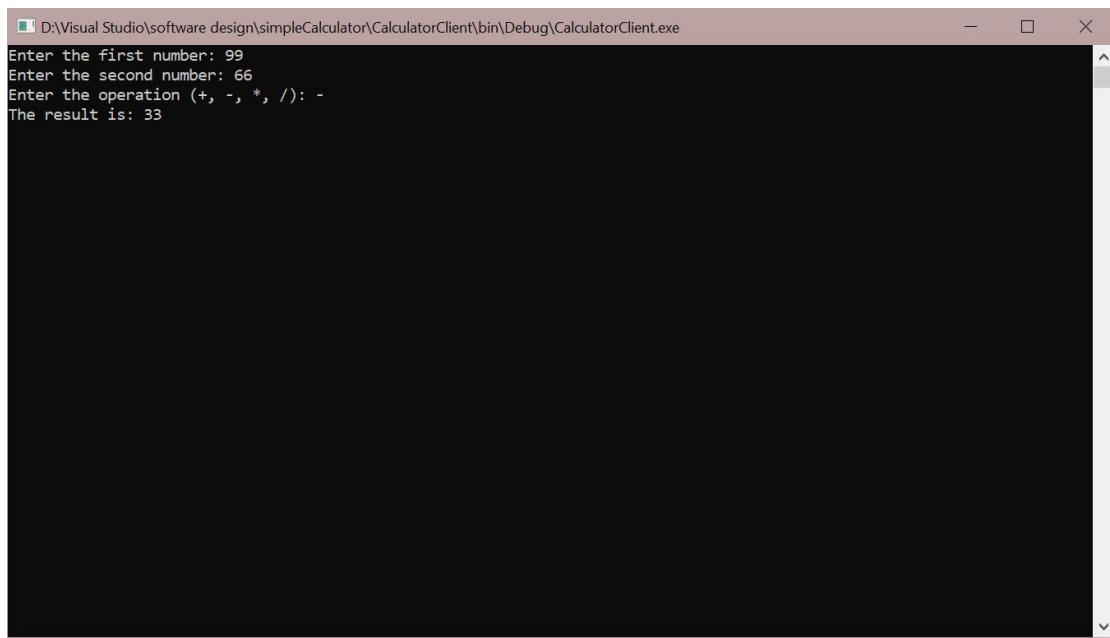
⑥客户端测试

Add

The screenshot shows a terminal window titled "D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe". The window displays the following interaction:

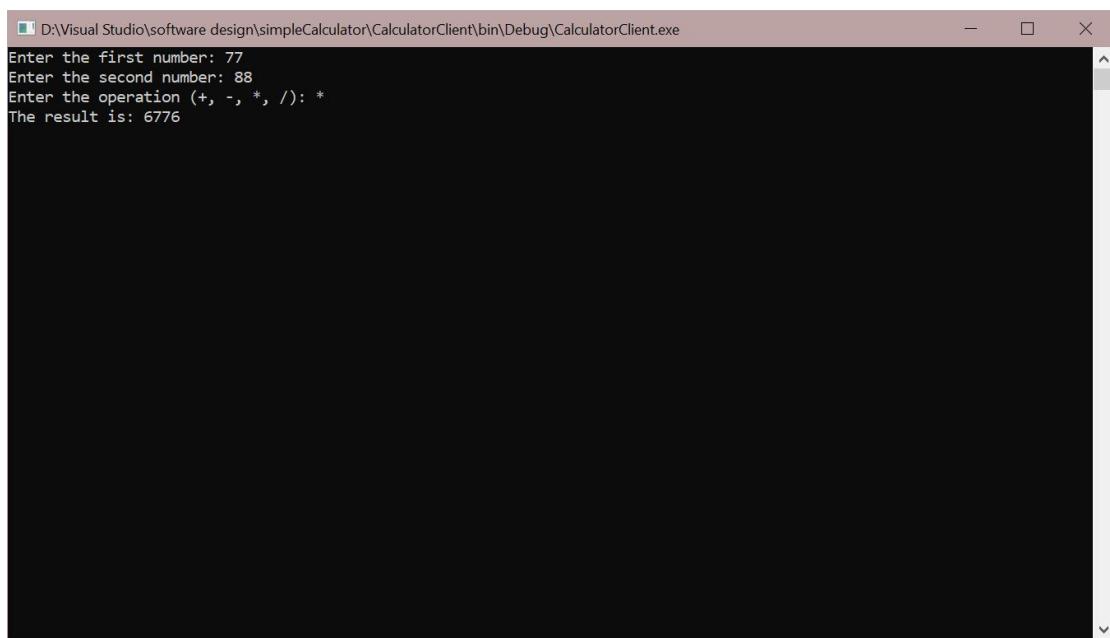
```
Enter the first number: 3
Enter the second number: 6
Enter the operation (+, -, *, /): +
The result is: 9
```

Sub



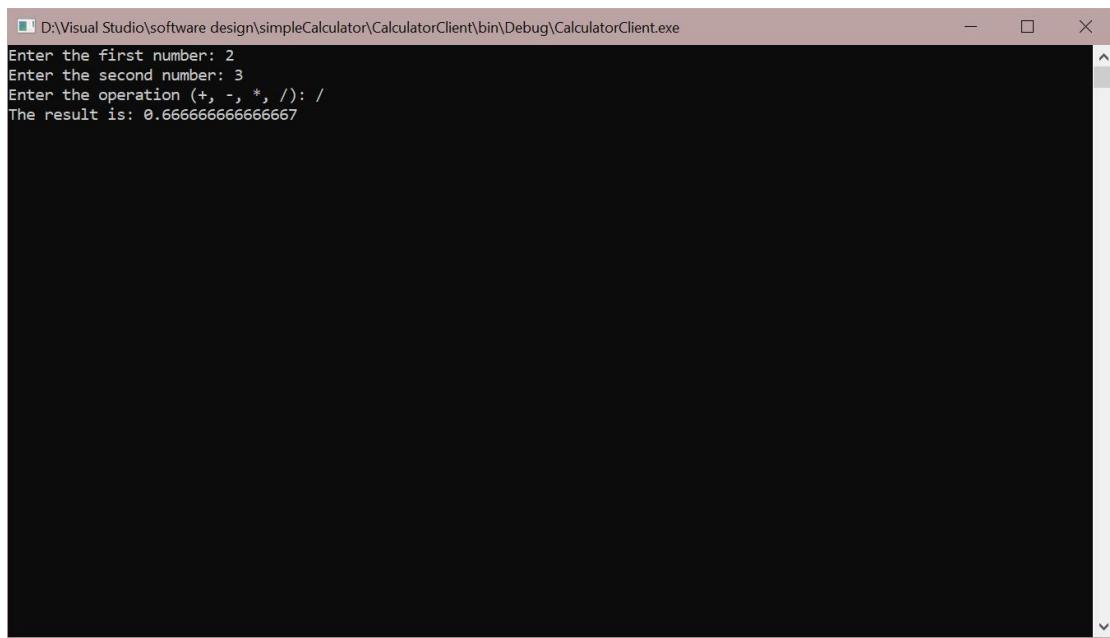
```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 99
Enter the second number: 66
Enter the operation (+, -, *, /): -
The result is: 33
```

Mul



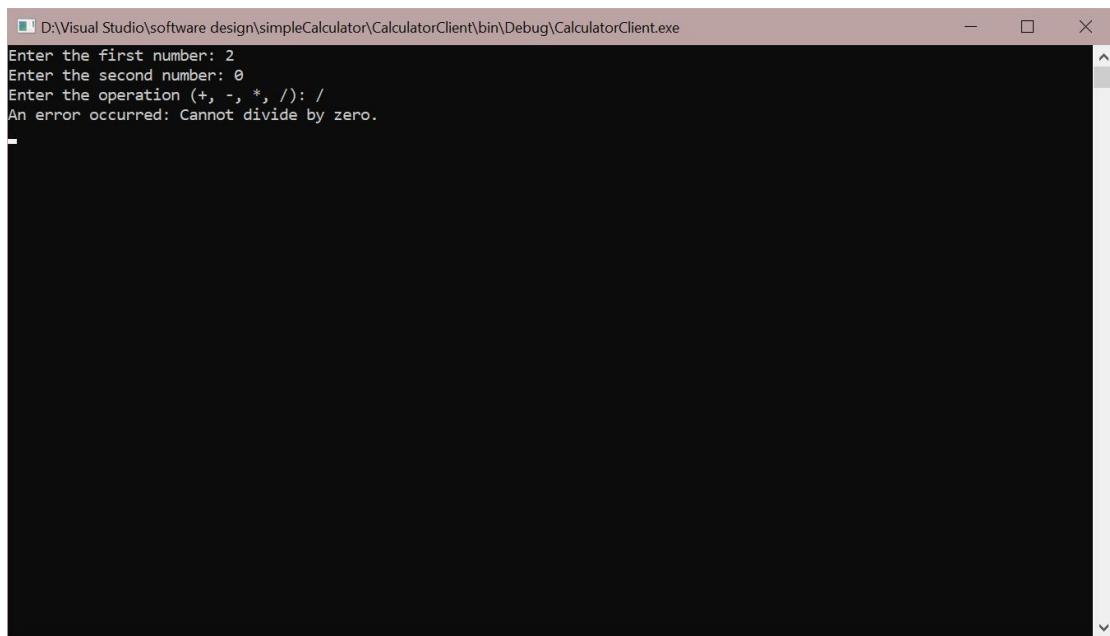
```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 77
Enter the second number: 88
Enter the operation (+, -, *, /): *
The result is: 6776
```

Div



```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 2
Enter the second number: 3
Enter the operation (+, -, *, /): /
The result is: 0.6666666666666667
```

除数为 0



```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 2
Enter the second number: 0
Enter the operation (+, -, *, /):
An error occurred: Cannot divide by zero.
```

(2) 为计算机添加对数运算，代码会发生什么变化？

其一：创建一个新的操作类“Logarithm”，实现 IOperation 接口，并且计算两个数的对数。

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      */
13
14      public class Logarithm : IOperation
15      {
16          public double Calculate(double num1, double num2)
17          {
18              if (num1 <= 0 || num2 <= 0)
19              {
20                  throw new ArgumentException("The arguments must be positive.");
21              }
22
23              return Math.Log(num1, num2);
24          }
25      }
26  }

```

其二：更新 App.config

```

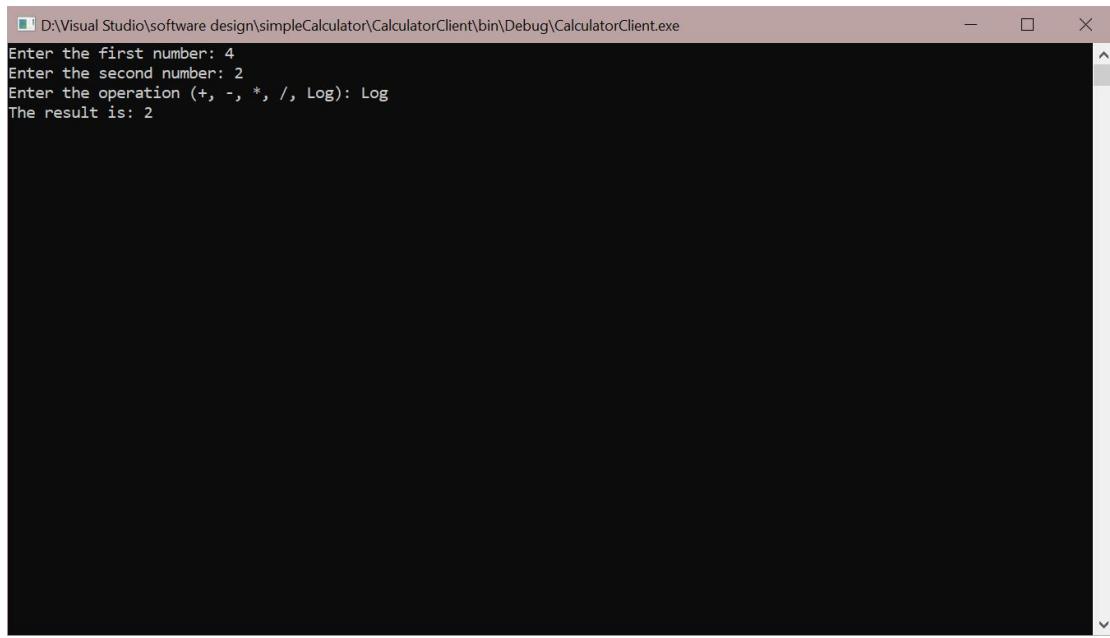
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
        <add key="+" value="Simplecalculator.Addition, Simplecalculator"/>
        <add key="-" value="Simplecalculator.Subtraction, Simplecalculator"/>
        <add key="/" value="Simplecalculator.Division, Simplecalculator"/>
        <add key="Log" value="Simplecalculator.Logarithm, Simplecalculator"/>
    </appSettings>
</configuration>

```

其三：更新客户端代码

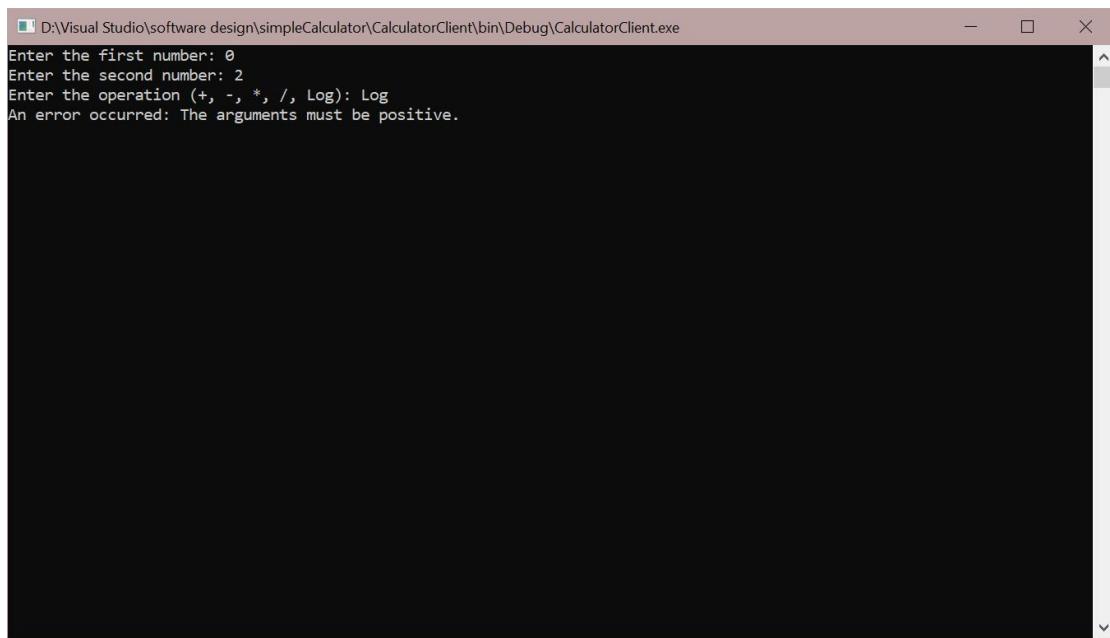
```
23 |           Console.WriteLine("Enter the operation (+, -, *, /, Log): ");
```

客户端测试：



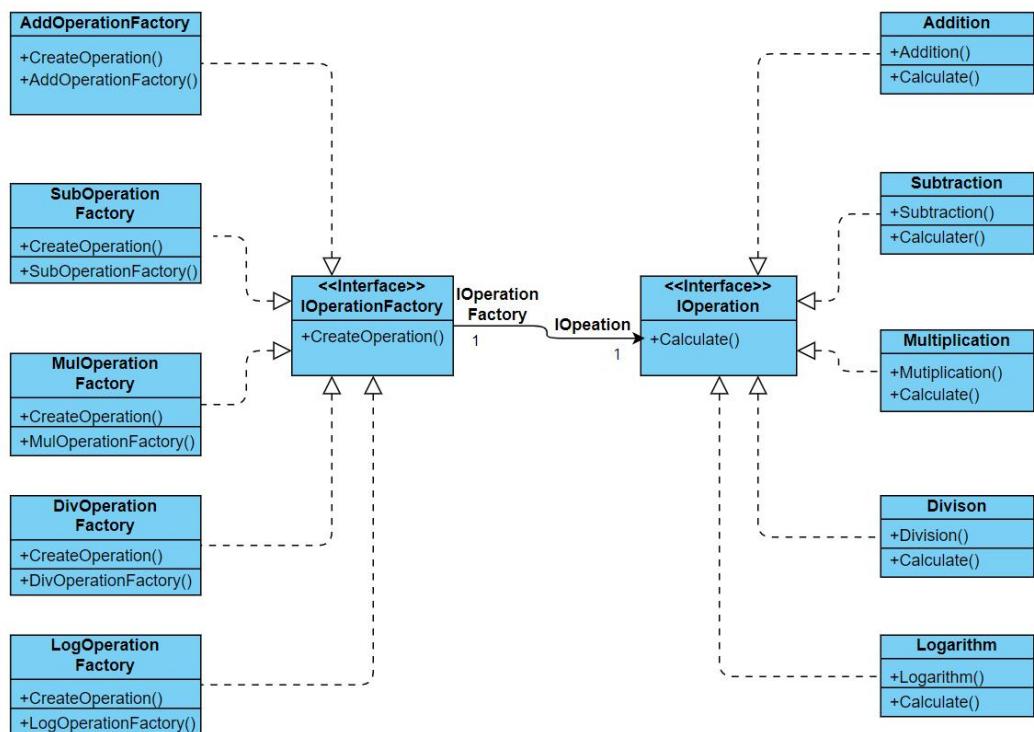
```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 4
Enter the second number: 2
Enter the operation (+, -, *, /, Log): Log
The result is: 2
```

参数非正



```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 0
Enter the second number: 2
Enter the operation (+, -, *, /, Log): Log
An error occurred: The arguments must be positive.
```

(3) 采用工厂方法模式重构计算器的二元计算模块，该计算模块是方便重用扩展的
①类图



②代码结构

Solution 'simpleCalculator' (2 of 2 projects)

- CalculatorClient**
 - Properties
 - References
 - App.config
 - Client.cs
- SimpleCalculator**
 - Properties
 - References
 - Addition.cs
 - AddOperationFactory.cs
 - App.config
 - Division.cs
 - DivOperationFactory.cs
 - IOperation.cs
 - IOperationFactory.cs

```

    ▷ C# DivOperationFactory.cs
    ▷ C# IOperation.cs
    ▷ C# IOperationFactory.cs
    ▷ C# Logarithm.cs
    ▷ C# LogOperationFactory.cs
    ▷ C# MulOperationFactory.cs
    ▷ C# Multiplication.cs
    ▷ C# OperationFactory.cs
    ▷ C# SubOperationFactory.cs
    ▷ C# Subtraction.cs

```

③核心类代码

Project SimpleCalculator

Interface IOperation

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

- Solution Explorer:** Shows the project structure with files like SimpleCalculator.csproj, Properties, References, Additions.cs, AddOperationFactory.cs, App.config, Division.cs, DivOperationFactory.cs, IOperation.cs, IOperationFactory.cs, Logarithm.cs, and LogOperationFactory.cs.
- Properties Window:** The properties for IOperation.cs are displayed, showing Build Action as 'Compile' and Copy to Output as 'Do not copy'.
- Code Editor:** The IOperation.cs file is open, containing the following code:

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10     * Author: arborshield
11     * Created by on 2023/4/2.
12     * Create an interface for Operation
13     */
14     public interface IOperation
15     {
16         double Calculate(double num1, double num2);
17     }
18 }

```

Class Addition

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

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Architecture, Test, Analyze, Tools, Extensions, Window, Help.
- Toolbox:** Toolbox, Data Sources.
- Solution Explorer:** Shows the project structure for "SimpleCalculator" with files like "Addition.cs", "AddOperationFactory.cs", "Division.cs", etc.
- Properties Window:** Shows file properties for "Addition.cs" including build action (Compile), copy to output (Do not copy), and file name (Addition.cs).
- Output Window:** Shows the output from the "Debug" build configuration, indicating the program has exited with code 0.
- Status Bar:** Ready.

```

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

```

Class Subtraction

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

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Architecture, Test, Analyze, Tools, Extensions, Window, Help.
- Toolbox:** Toolbox, Data Sources.
- Solution Explorer:** Shows the project structure for "SimpleCalculator" with files like "Subtraction.cs", "DivOperationFactory.cs", etc.
- Properties Window:** Shows file properties for "Subtraction.cs" including build action (Compile), copy to output (Do not copy), and file name (Subtraction.cs).
- Output Window:** Shows the output from the "Debug" build configuration, indicating the program has exited with code 0.
- Status Bar:** Ready.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      */
13
14      public class Subtraction : IOperation
15      {
16          public double Calculate(double num1, double num2)
17          {
18              return num1 - num2;
19          }
20      }
21  }
22

```

Class Multiplication

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

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Architecture, Test, Analyze, Tools, Extensions, Window, Help.
- Toolbox:** Toolbox, Data Sources.
- Solution Explorer:** Shows files: Division.cs, IOperation.cs, Logarithm.cs, MultiOperationFactory.cs, Multiplication.cs, OperationFactory.cs, SubOperationFactory.cs, Subtraction.cs.
- Properties Window:** Advanced, Misc (File Name: Multiplication.cs, Full Path: D:\Visual Studio\software).
- Code Editor:** Displays the `Multiplication.cs` file content:

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborsshield
11      * Created by: on 2023/4/2.
12      * Mul
13      */
14
15      public class Multiplication : IOperation
16      {
17
18          public double Calculate(double num1, double num2)
19          {
20              return num1 * num2;
21          }
22      }

```
- Output Window:** Shows build output for 'calculatorclient.exe'.
- Status Bar:** Ready.

Class Division

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

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Architecture, Test, Analyze, Tools, Extensions, Window, Help.
- Toolbox:** Toolbox, Data Sources.
- Solution Explorer:** Shows files: Division.cs, IOperation.cs, Logarithm.cs, MultiOperationFactory.cs, Multiplication.cs, OperationFactory.cs, SubOperationFactory.cs, Subtraction.cs.
- Properties Window:** Advanced, Misc (File Name: Division.cs, Full Path: D:\Visual Studio\software).
- Code Editor:** Displays the `Division.cs` file content:

```

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

```
- Output Window:** Shows build output for 'calculatorclient.exe'.
- Status Bar:** Ready.

Class Logarithm

The screenshot shows the Visual Studio IDE interface. The main code editor displays the `Logarithm.cs` file, which contains the implementation of the `IOperation` interface. The `Solution Explorer` pane on the right lists several files including `Division.cs`, `DivOperationFactory.cs`, `IOperation.cs`, `IOperationFactory.cs`, `Logarithm.cs`, `LogOperationFactory.cs`, `MulOperationFactory.cs`, `Multiplication.cs`, `OperationFactory.cs`, `SubOperationFactory.cs`, and `Subtraction.cs`. The properties pane shows the file `Logarithm.cs` with details like build action (Compile), copy to output (Do not copy), and file name (Logarithm.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 SimpleCalculator
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      */
13      public class Logarithm : IOperation
14  {
15      public double Calculate(double num1, double num2)
16      {
17          if (num1 <= 0 || num2 <= 0)
18          {
19              throw new ArgumentException("The arguments must be positive.");
20          }
21
22          return Math.Log(num1, num2);
23      }
24  }
25
26 }
27

```

Interface IOperationFactory

The screenshot shows the Visual Studio IDE interface. The main code editor displays the `IOperationFactory.cs` file, which defines an interface `IOperationFactory` with a single method `CreateOperation()`. The `Solution Explorer` pane on the right lists the same set of files as the previous screenshot. The properties pane shows the file `IOperationFactory.cs` with details like build action (Compile), copy to output (Do not copy), and file name (IOperationFactory.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 SimpleCalculator
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      */
13      public interface IOperationFactory
14  {
15      IOperation CreateOperation();
16  }
17
18 }
19

```

Class AddOperationFactory

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborsshield
11      * Created by on 2023/4/2.
12      */
13
14      public class AddOperationFactory : IOperationFactory
15      {
16          public IOperation CreateOperation()
17          {
18              return new Addition();
19          }
20      }
21 }

```

The screenshot shows the Visual Studio IDE interface. The main window displays the code for `AddOperationFactory.cs`. The code defines a class `AddOperationFactory` that implements the `IOperationFactory` interface. It contains a single method `CreateOperation` which returns a new instance of `Addition`. The code is well-formatted with proper indentation and comments. The Solution Explorer on the right shows the project structure with files like `Division.cs`, `DivOperationFactory.cs`, `IOperation.cs`, `IOperationFactory.cs`, `Logarithm.cs`, and `MulOperationFactory.cs`. The Properties window on the right shows file properties for `AddOperationFactory.cs`.

Class SubOperationFactory

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborsshield
11      * Created by on 2023/4/2.
12      */
13
14      public class SubOperationFactory : IOperationFactory
15      {
16          public IOperation CreateOperation()
17          {
18              return new Subtraction();
19          }
20      }
21 }

```

The screenshot shows the Visual Studio IDE interface. The main window displays the code for `SubOperationFactory.cs`. The code defines a class `SubOperationFactory` that implements the `IOperationFactory` interface. It contains a single method `CreateOperation` which returns a new instance of `Subtraction`. The code is well-formatted with proper indentation and comments. The Solution Explorer on the right shows the project structure with files like `Division.cs`, `DivOperationFactory.cs`, `IOperation.cs`, `IOperationFactory.cs`, `Logarithm.cs`, and `MulOperationFactory.cs`. The Properties window on the right shows file properties for `SubOperationFactory.cs`.

Class MulOperationFactory

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      */
13
14     public class MulOperationFactory : IOperationFactory
15     {
16         /**
17         * Author: arborshield
18         * Created by on 2023/4/2.
19         */
20         public IOperation CreateOperation()
21         {
22             return new Multiplication();
23         }
24     }
25 }

```

Class DivOperationFactory

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      */
13
14     public class DivOperationFactory : IOperationFactory
15     {
16         /**
17         * Author: arborshield
18         * Created by on 2023/4/2.
19         */
20         public IOperation CreateOperation()
21         {
22             return new Division();
23         }
24     }
25 }

```

Class LogOperationFactory

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleCalculator
8  {
9      /**
10      * Author: arborsshield
11      * Created by on 2023/4/2.
12      * Log Factory
13      */
14      public class LogOperationFactory : IOperationFactory
15      {
16          public IOperation CreateOperation()
17          {
18              return new Logarithm();
19          }
20      }
21  }
```

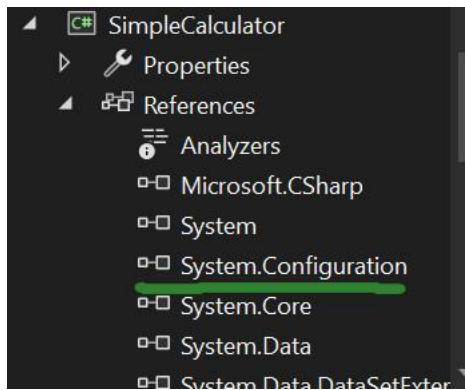
The screenshot shows the Visual Studio IDE interface with the LogOperationFactory.cs file open in the code editor. The code implements a factory pattern for creating logarithmic operations. The Solution Explorer on the right lists other files like Division.cs, MulOperationFactory.cs, etc. The Properties window shows the file is named LogOperationFactory.cs and has a Full Path of D:\Visual Studio\software.

Class Operation Factory

```
1  using System;
2  using System.Configuration;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace SimpleCalculator
9  {
10     /**
11      * Author: arborsshield
12      * Created by on 2023/4/2.
13      * Operation Factory
14      */
15     public class OperationFactory
16     {
17         public static IOperation CreateOperation(string operatorName)
18         {
19             // Get the fully-qualified name of the operation class from the config file
20             string operationClassName = ConfigurationManager.AppSettings[operatorName];
21
22             // Use reflection to create an instance of the operation class
23             Type operationClassType = Type.GetType(operationClassName);
24             IOperation operation = (IOperation)Activator.CreateInstance(operationClassType);
25
26             return operation;
27         }
28     }
29  }
```

The screenshot shows the Visual Studio IDE interface with the OperationFactory.cs file open in the code editor. This version of the factory uses reflection to dynamically create instances of operation classes based on their names. The Solution Explorer and Properties windows are visible on the right.

④客户端代码



Project CalculatorClient

Class Client

The screenshot shows the Visual Studio IDE interface. The main window displays the `Client.cs` file content:

```
1  using SimpleCalculator;
2  using System;
3
4  namespace CalculatorClient
5  {
6      /**
7       * Author: arborshield
8       * Created by: 2023/4/2.
9       * ClientTest
10      */
11     class Client
12     {
13         static void Main(string[] args)
14         {
15             try
16             {
17                 Console.WriteLine("Enter the first number: ");
18                 double num1 = Convert.ToDouble(Console.ReadLine());
19
20                 Console.WriteLine("Enter the second number: ");
21                 double num2 = Convert.ToDouble(Console.ReadLine());
22
23                 Console.WriteLine("Enter the operation (+, -, *, /, Log): ");
24                 string operationName = Console.ReadLine();
25
26                 IOperation operation = OperationFactory.CreateOperation(operationName);
27
28                 double result = operation.Calculate(num1, num2);
29
30                 Console.WriteLine($"The result is: {result}");
31             }
32         }
33     }
34 }
```

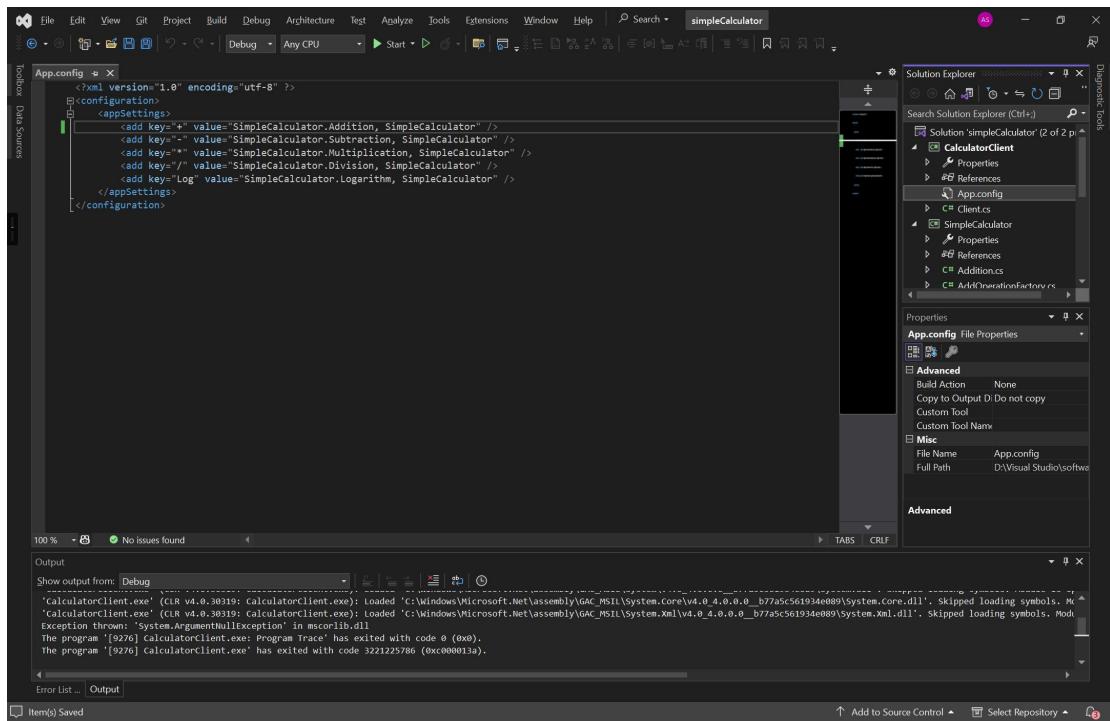
The Solution Explorer on the right shows two projects: `CalculatorClient` and `SimpleCalculator`. The `Client.cs` file is selected in the `CalculatorClient` project. The Properties panel shows the file name is `Client.cs` and the full path is `D:\Visual Studio\software\CalculatorClient\CalculatorClient\Client.cs`.

The screenshot shows the Visual Studio IDE interface. The main window displays the `Client.cs` file content with exception handling added:

```
10     /*
11      * References:
12      * Class Client
13      */
14     class Client
15     {
16         static void Main(string[] args)
17         {
18             try
19             {
20                 Console.WriteLine("Enter the first number: ");
21                 double num1 = Convert.ToDouble(Console.ReadLine());
22
23                 Console.WriteLine("Enter the second number: ");
24                 double num2 = Convert.ToDouble(Console.ReadLine());
25
26                 Console.WriteLine("Enter the operation (+, -, *, /, Log): ");
27                 string operationName = Console.ReadLine();
28
29                 IOperation operation = OperationFactory.CreateOperation(operationName);
30
31                 double result = operation.Calculate(num1, num2);
32
33                 Console.WriteLine($"The result is: {result}");
34             }
35             catch (Exception ex)
36             {
37                 Console.WriteLine($"An error occurred: {ex.Message}");
38             }
39         }
40     }
41 }
```

The Solution Explorer on the right shows the same two projects: `CalculatorClient` and `SimpleCalculator`. The `Client.cs` file is selected in the `CalculatorClient` project. The Properties panel shows the file name is `Client.cs` and the full path is `D:\Visual Studio\software\CalculatorClient\CalculatorClient\Client.cs`.

⑤配置文件

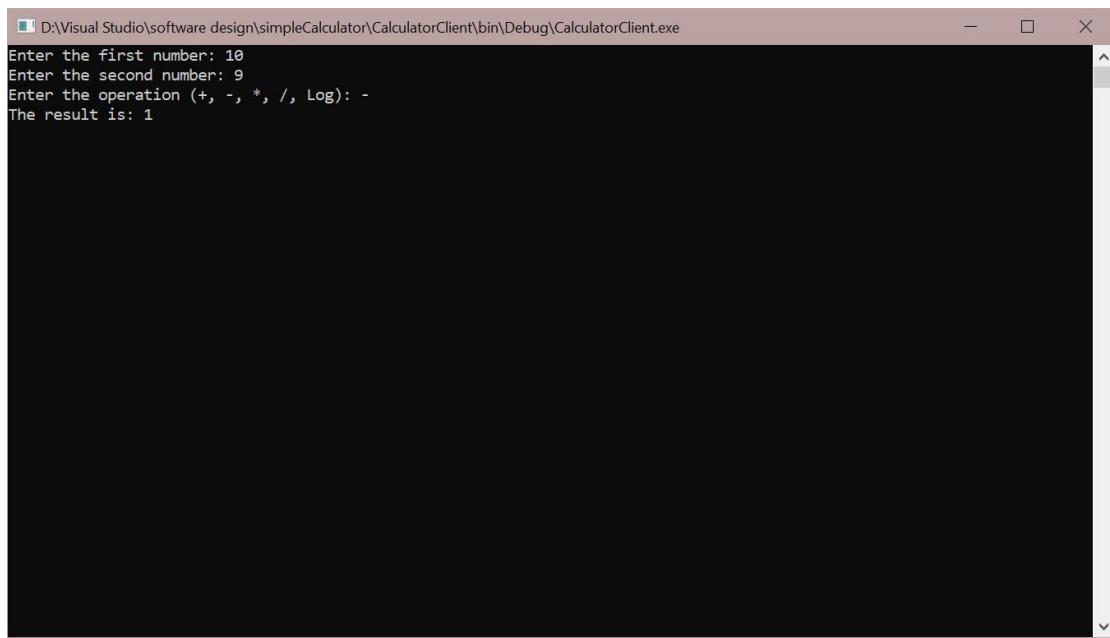


⑥客户端测试

Add

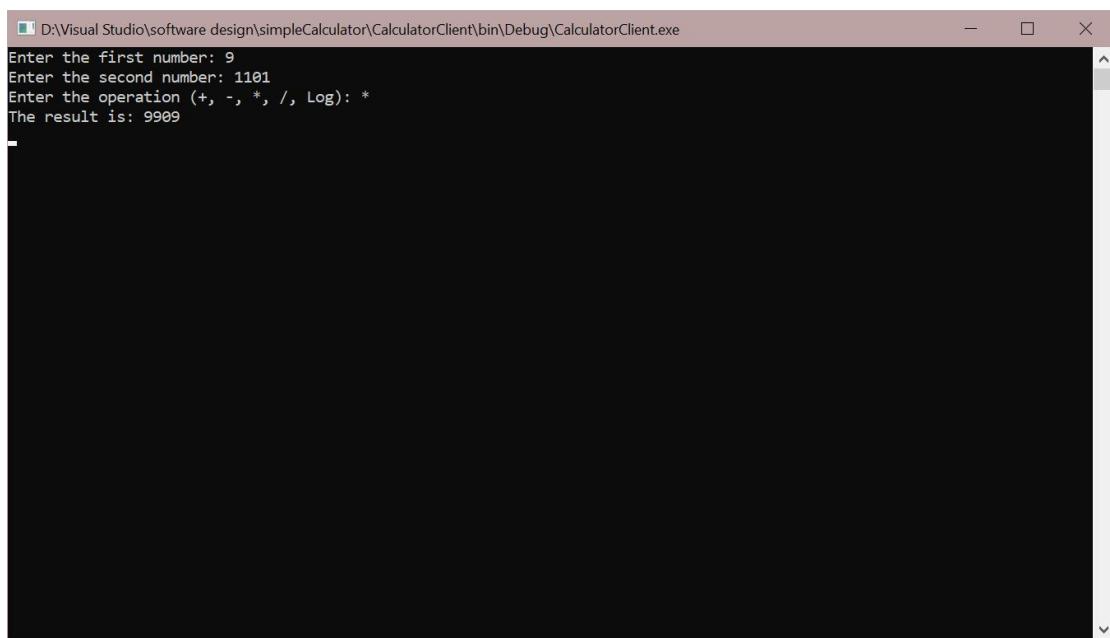
```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 10
Enter the second number: 9
Enter the operation (+, -, *, /, Log): +
The result is: 19
```

Sub



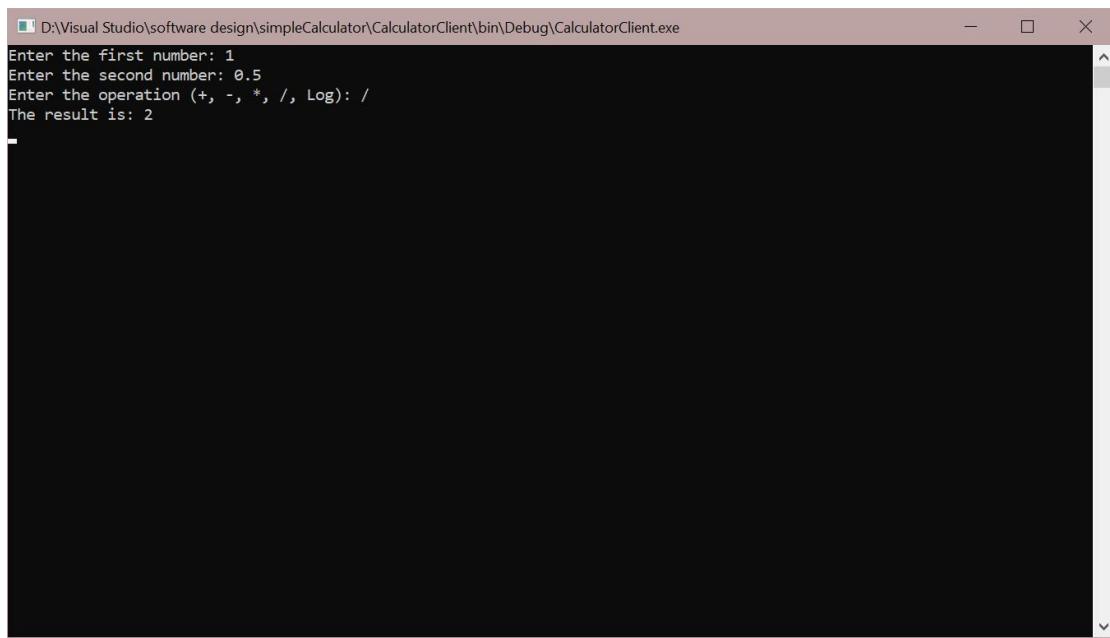
```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 10
Enter the second number: 9
Enter the operation (+, -, *, /, Log): -
The result is: 1
```

Mul



```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 9
Enter the second number: 1101
Enter the operation (+, -, *, /, Log): *
The result is: 9909
```

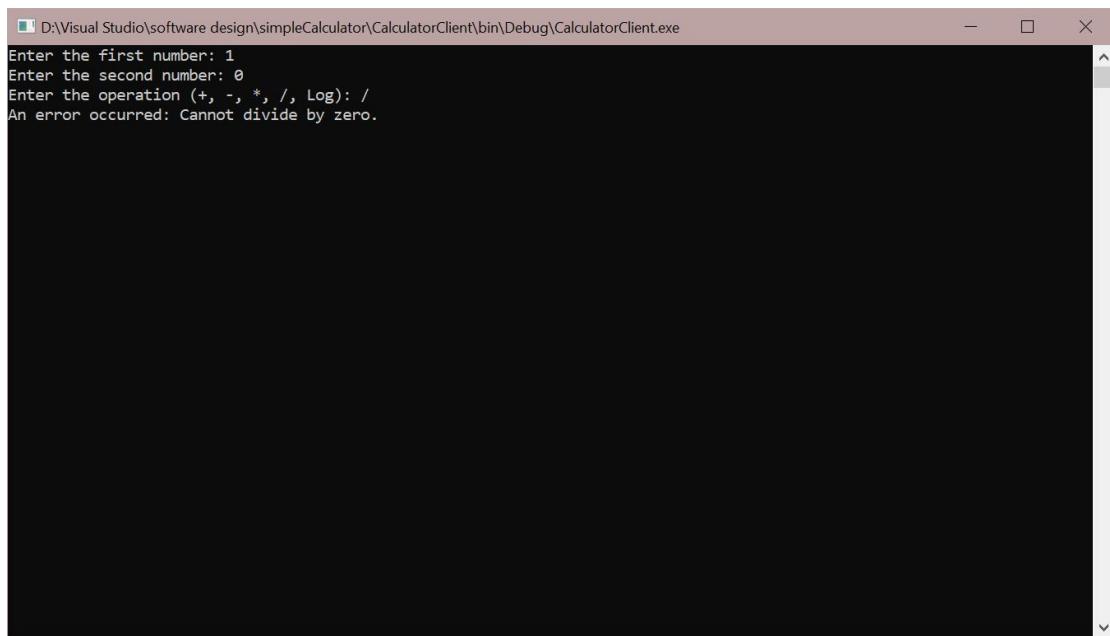
Div



D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe

```
Enter the first number: 1
Enter the second number: 0.5
Enter the operation (+, -, *, /, Log): /
The result is: 2
```

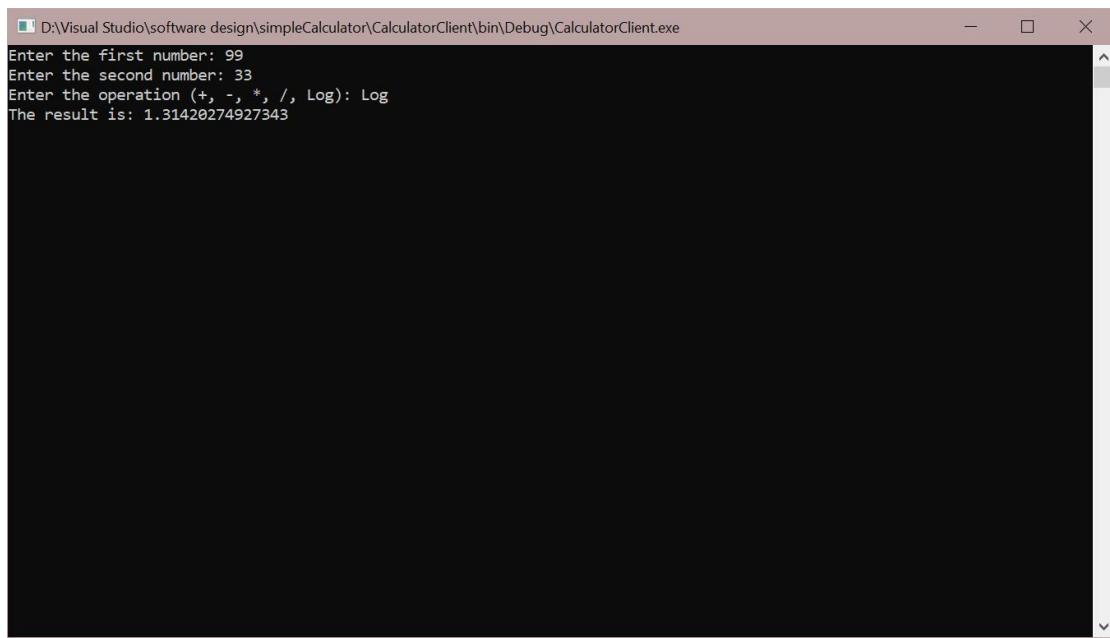
除数为 0



D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe

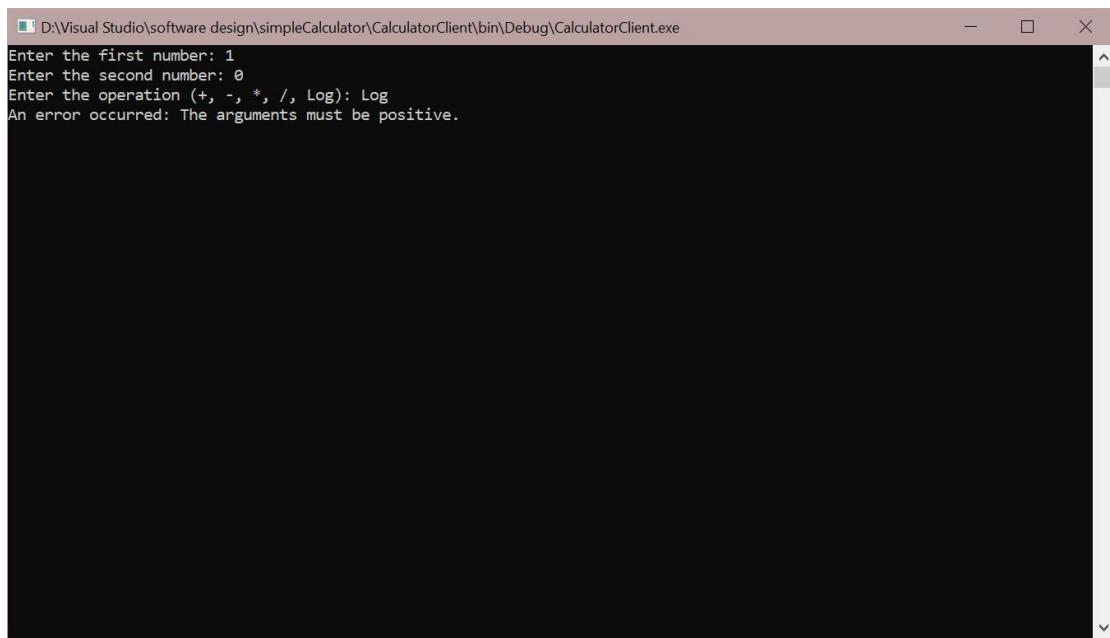
```
Enter the first number: 1
Enter the second number: 0
Enter the operation (+, -, *, /, Log): /
An error occurred: Cannot divide by zero.
```

Log



```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 99
Enter the second number: 33
Enter the operation (+, -, *, /, Log): Log
The result is: 1.31420274927343
```

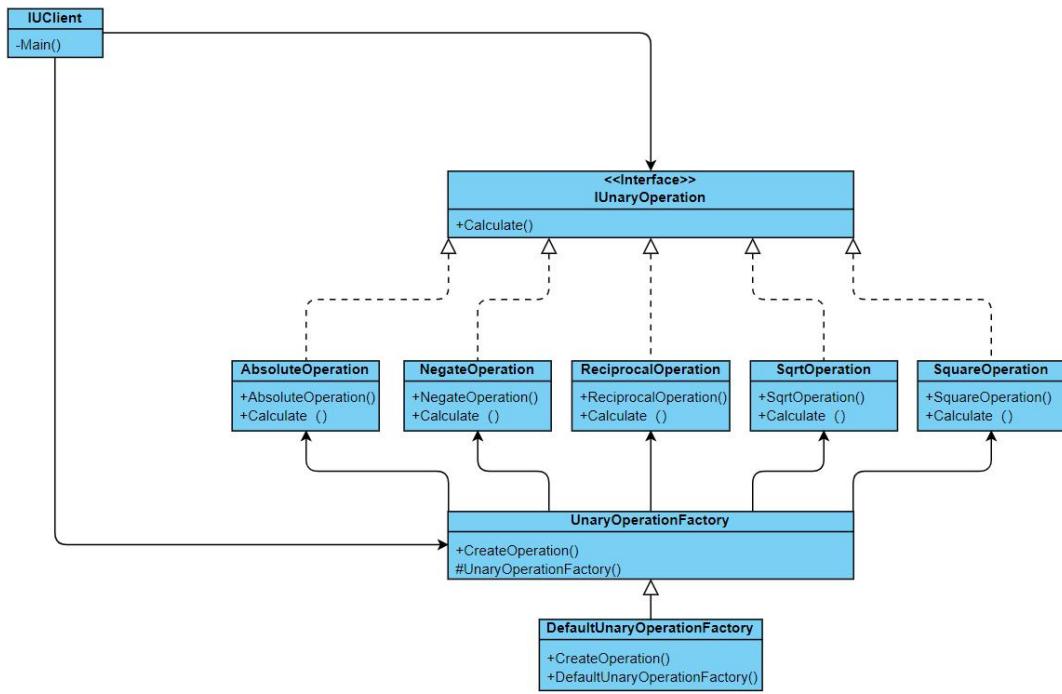
参数非正



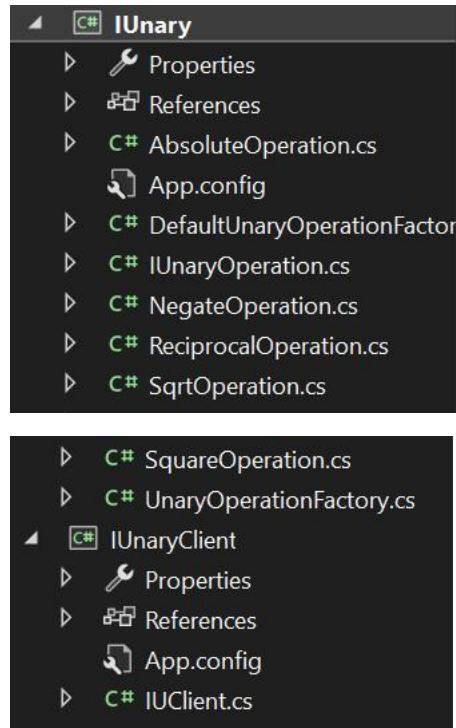
```
D:\Visual Studio\software design\simpleCalculator\CalculatorClient\bin\Debug\CalculatorClient.exe
Enter the first number: 1
Enter the second number: 0
Enter the operation (+, -, *, /, Log): Log
An error occurred: The arguments must be positive.
```

(4) 采用工厂方法模式设计实现计算器的一元计算模块，实现求绝对值、相反数、倒数、平方根、平方等操作、该模块是方便重用扩展的

①类图



②代码结构



③核心类代码

Project IUnary

Interface `IUnaryOperation`

The screenshot shows the Visual Studio IDE interface with the code editor displaying the `IUnary` class. The code defines an interface with a single method `Calculate(double num)`. The Solution Explorer on the right shows the project structure with files like `IUnaryOperation.cs`, `AbsoluteOperation.cs`, `NegateOperation.cs`, `ReciprocalOperation.cs`, `SqrtOperation.cs`, and `SquareOperation.cs`. The Properties window is open for `IUnaryOperation.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 ILUnary
8  {
9      /**
10      * Author: arborshield
11      * Created on: 2023/4/2.
12      * Create an interface for Unary Operation
13      */
14     public interface IUnaryOperation
15     {
16         double Calculate(double num);
17     }
18 }

```

Class Abs

The screenshot shows the Visual Studio IDE interface with the code editor displaying the `AbsoluteOperation` class. It implements the `IUnaryOperation` interface and contains a single method `Calculate(double num)` which returns the absolute value of the input. The Solution Explorer and Properties windows are visible on the right.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ILUnary
8  {
9      /**
10      * Author: arborshield
11      * Created on: 2023/4/2.
12      * Abs
13      */
14     public class AbsoluteOperation : IUnaryOperation
15     {
16         public double Calculate(double num)
17         {
18             return Math.Abs(num);
19         }
20     }
21 }

```

Class Neg

```

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     public class NegateOperation : IUnaryOperation
15     {
16         /**
17         * Author: arborshield
18         * Created by on 2023/4/2.
19         */
20         public double Calculate(double num)
21         {
22             return -num;
23         }
24     }
25 }

```

The Solution Explorer shows files for IUnary, AbsoluteOperation.cs, DefaultUnaryOperationFactor.cs, IUnaryOperation.cs, NegateOperation.cs, ReciprocalOperation.cs, SqrtOperation.cs, and SquareOperation.cs. The Properties window shows settings for NegateOperation.cs.

Class Rec

```

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     public class ReciprocalOperation : IUnaryOperation
15     {
16         /**
17         * Author: arborshield
18         * Created by on 2023/4/2.
19         */
20         public double Calculate(double num)
21         {
22             if (num == 0)
23             {
24                 throw new ArgumentException("Cannot divide by zero.");
25             }
26             return 1 / num;
27         }
28     }
29 }

```

The Solution Explorer shows files for IUnary, AbsoluteOperation.cs, DefaultUnaryOperationFactor.cs, IUnaryOperation.cs, NegateOperation.cs, ReciprocalOperation.cs, SqrtOperation.cs, and SquareOperation.cs. The Properties window shows settings for ReciprocalOperation.cs.

Class Sqrt

```

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      */
13
14     public class SqrtOperation : IUnaryOperation
15     {
16         /**
17         * Author: arborshield
18         * Created by on 2023/4/2.
19         */
20
21         public double Calculate(double num)
22         {
23             if (num < 0)
24             {
25                 throw new ArgumentException("Cannot take square root of negative number.");
26             }
27             return Math.Sqrt(num);
28         }
29     }
30 }

```

The screenshot shows the Visual Studio IDE with the code editor open to SqrtOperation.cs. The code defines a class SqrtOperation that implements the IUnary interface. It includes a check for negative numbers and returns the square root of the number. The Solution Explorer and Properties windows are also visible.

Class Square

```

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      */
13
14     public class SquareOperation : IUnaryOperation
15     {
16         /**
17         * Author: arborshield
18         * Created by on 2023/4/2.
19         */
20
21         public double Calculate(double num)
22         {
23             return num * num;
24         }
25     }
26 }

```

The screenshot shows the Visual Studio IDE with the code editor open to SquareOperation.cs. The code defines a class SquareOperation that implements the IUnary interface. It calculates the square of the input number. The Solution Explorer and Properties windows are also visible.

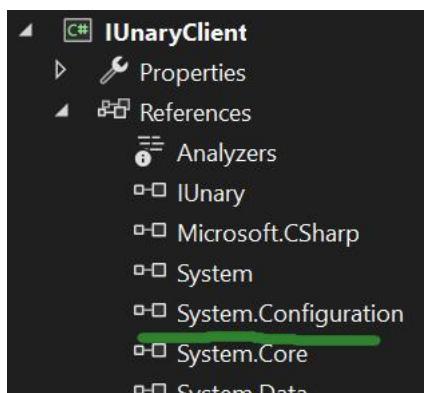
Abstract Class Unary Operation Factory

```

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      * IUnary Operation Factory
13      */
14     public abstract class UnaryOperationFactory
15     {
16         /**
17          * Create operation
18         */
19     }

```

④ 客户端代码



Project IUnaryClient

Class IUnaryClient

When the silence comes and yo
Look at me and stay for a whi

```

1  using System;
2  using IUnary;
3  using System.Collections.Generic;
4  using System.Configuration;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace IUnaryClient
10 {
11     /*
12     * Author: arborshield
13     * Created by on 2023/4/2.
14     * ClientTest
15     */
16     class IUClient
17     {
18         static void Main(string[] args)
19         {
20             try
21             {
22                 Console.Write("Enter the number: ");
23                 double num = Convert.ToDouble(Console.ReadLine());
24
25                 Console.Write("Enter the operation (abs, neg, rec, sqrt, square): ");
26                 string operationName = Console.ReadLine();
27                 string operationTypeName = ConfigurationManager.AppSettings[operationName];
28
29                 IUnaryOperation operation = (IUnaryOperation)Activator.CreateInstance(Type.GetType(operationTypeName));
30
31                 double result = operation.Calculate(num);
32
33                 Console.WriteLine($"The result is: {result}");
34             }
35             catch (Exception ex)
36             {
37                 Console.WriteLine($"An error occurred: {ex.Message}");
38             }
39
40             Console.ReadKey();
41         }
42     }
43 }
```

No issues found

Show output from: Debug

'IBinaryClient.exe' [CLR v4.0.30319; DefaultDomain]: Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_32\mscorlib\v4.0.0.0__b77a5c561934e089\mscorlib.dll'. Skipped loading symbols. Module is optimized.
'IBinaryClient.exe' [CLR v4.0.30319; DefaultDomain]: Loaded 'D:\Visual Studio\software design\simplecalculator\CalculatorClient\bin\Debug\IBinaryClient.exe'. Symbols loaded.
'IBinaryClient.exe' [CLR v4.0.30319; IBinaryClient.exe]: Loaded 'D:\Visual Studio\software design\simplecalculator\CalculatorClient\bin\Debug\IBinary.dll'. Symbols loaded.
The program '[24788] IBinaryClient.exe: Program Trace' has exited with code 0 (0x0).
The program '[24788] IBinaryClient.exe' has exited with code 3221225786 (0xc000013a).

Error List | Output

And I'm reaching for your ha
And I'm asking for your love

```

13     /*
14     * Created by on 2023/4/2.
15     */
16     class IUClient
17     {
18         static void Main(string[] args)
19         {
20             try
21             {
22                 Console.Write("Enter the number: ");
23                 double num = Convert.ToDouble(Console.ReadLine());
24
25                 Console.Write("Enter the operation (abs, neg, rec, sqrt, square): ");
26                 string operationName = Console.ReadLine();
27                 string operationTypeName = ConfigurationManager.AppSettings[operationName];
28
29                 IUnaryOperation operation = (IUnaryOperation)Activator.CreateInstance(Type.GetType(operationTypeName));
30
31                 double result = operation.Calculate(num);
32
33                 Console.WriteLine($"The result is: {result}");
34             }
35             catch (Exception ex)
36             {
37                 Console.WriteLine($"An error occurred: {ex.Message}");
38             }
39
40             Console.ReadKey();
41         }
42     }
43 }
```

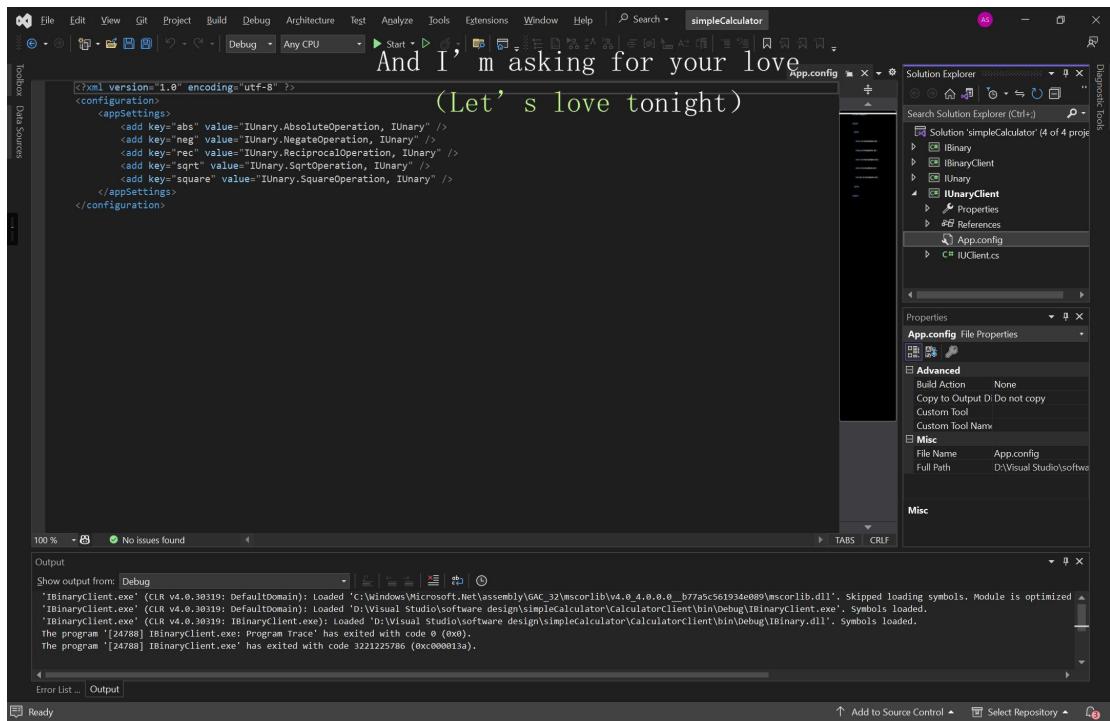
No issues found

Show output from: Debug

'IBinaryClient.exe' [CLR v4.0.30319; DefaultDomain]: Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_32\mscorlib\v4.0.0.0__b77a5c561934e089\mscorlib.dll'. Skipped loading symbols. Module is optimized.
'IBinaryClient.exe' [CLR v4.0.30319; DefaultDomain]: Loaded 'D:\Visual Studio\software design\simplecalculator\CalculatorClient\bin\Debug\IBinaryClient.exe'. Symbols loaded.
'IBinaryClient.exe' [CLR v4.0.30319; IBinaryClient.exe]: Loaded 'D:\Visual Studio\software design\simplecalculator\CalculatorClient\bin\Debug\IBinary.dll'. Symbols loaded.
The program '[24788] IBinaryClient.exe: Program Trace' has exited with code 0 (0x0).
The program '[24788] IBinaryClient.exe' has exited with code 3221225786 (0xc000013a).

Error List | Output

⑤配置文件

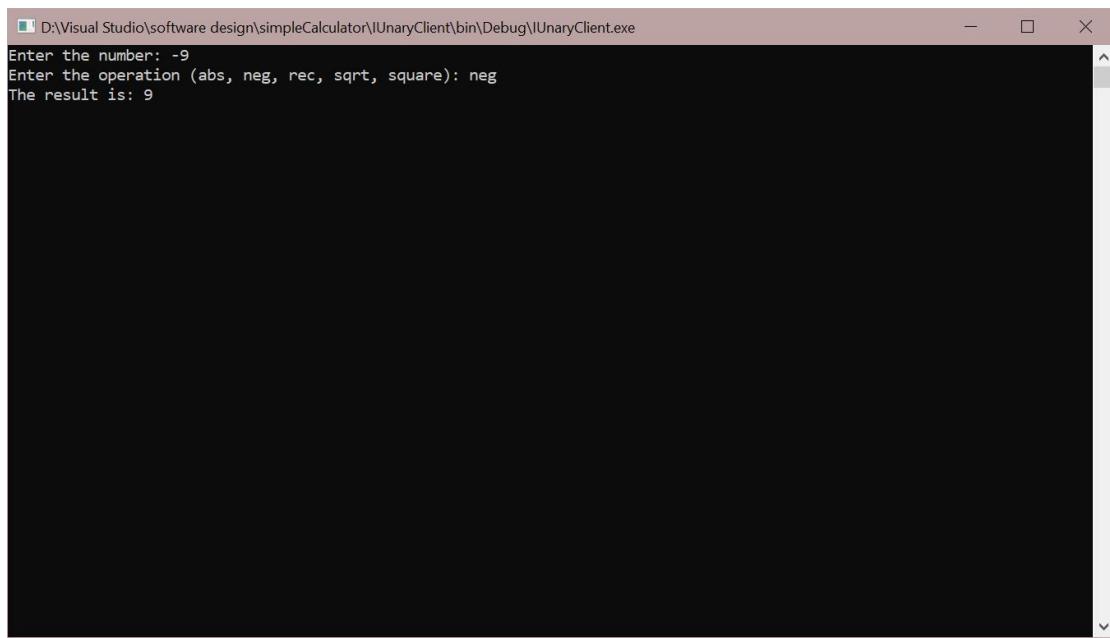


⑥客户端测试

Abs

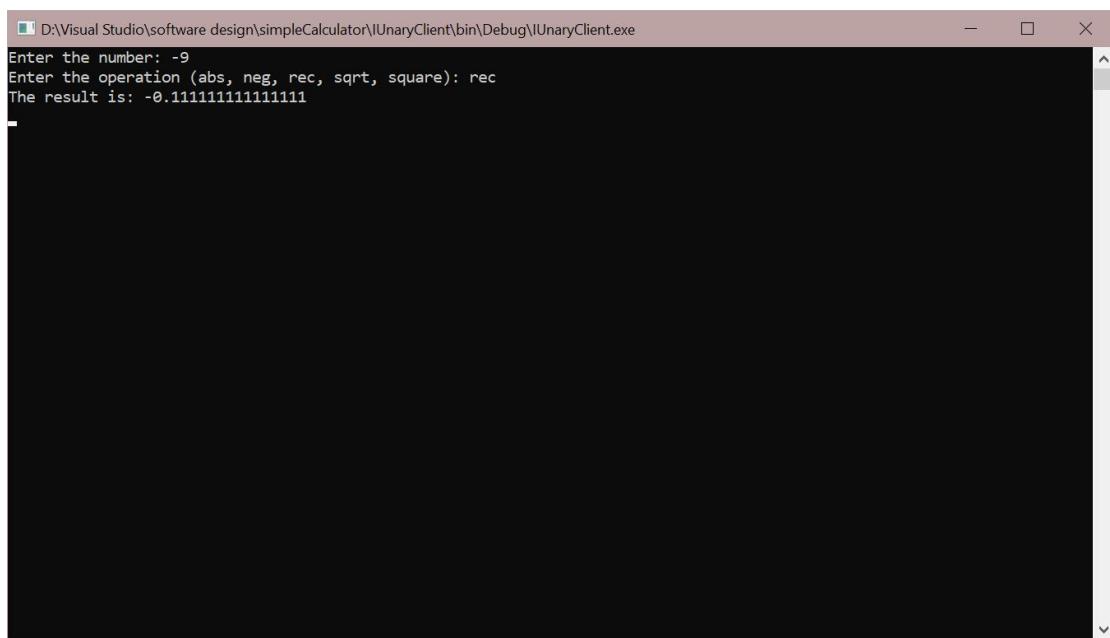
```
D:\Visual Studio\software design\simpleCalculator\IUnaryClient\bin\Debug\IUnaryClient.exe
Enter the number: -9
Enter the operation (abs, neg, rec, sqrt, square): abs
The result is: 9
```

Neg



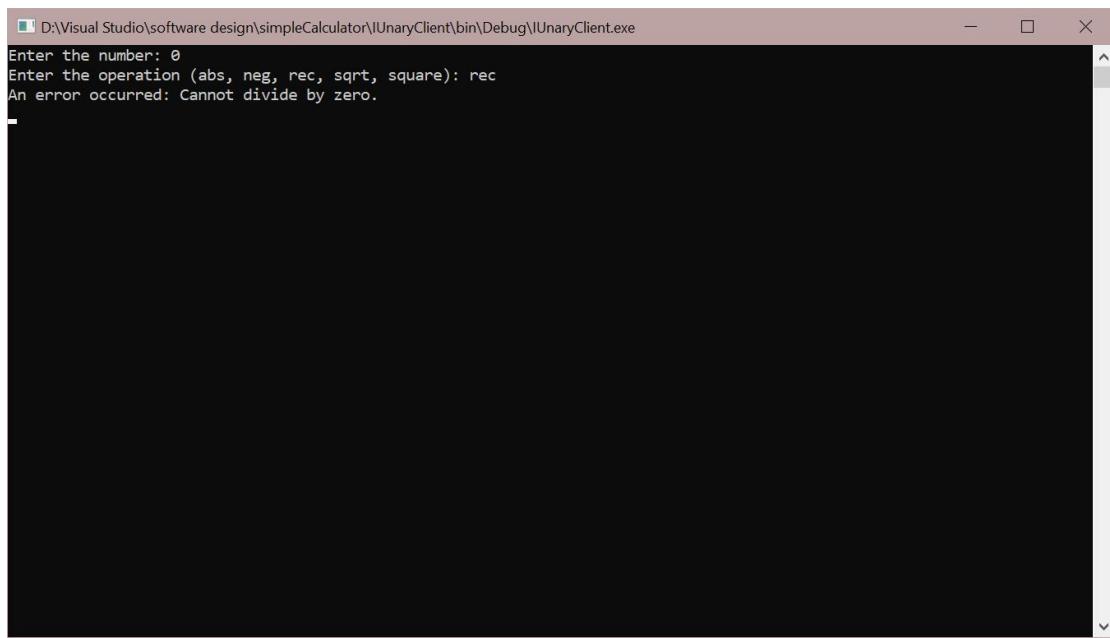
```
D:\Visual Studio\software design\simpleCalculator\UnaryClient\bin\Debug\UnaryClient.exe
Enter the number: -9
Enter the operation (abs, neg, rec, sqrt, square): neg
The result is: 9
```

Rec



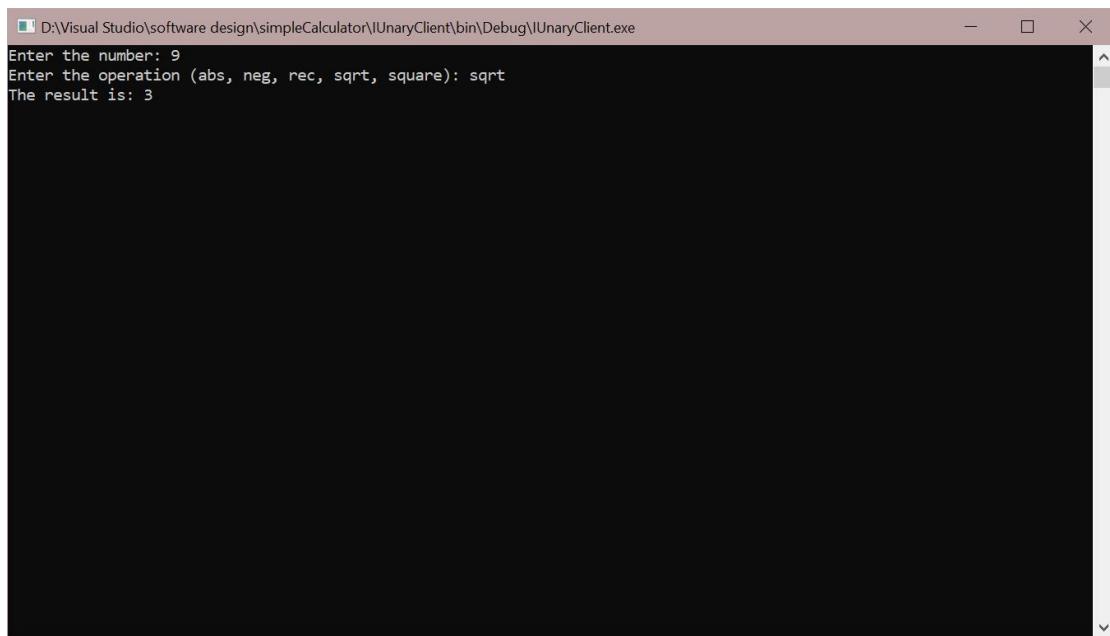
```
D:\Visual Studio\software design\simpleCalculator\UnaryClient\bin\Debug\UnaryClient.exe
Enter the number: -9
Enter the operation (abs, neg, rec, sqrt, square): rec
The result is: -0.1111111111111111
```

参数为 0



```
D:\Visual Studio\software design\simpleCalculator\IUnaryClient\bin\Debug\IUnaryClient.exe
Enter the number: 0
Enter the operation (abs, neg, rec, sqrt, square): rec
An error occurred: Cannot divide by zero.
```

Sqrt



```
D:\Visual Studio\software design\simpleCalculator\IUnaryClient\bin\Debug\IUnaryClient.exe
Enter the number: 9
Enter the operation (abs, neg, rec, sqrt, square): sqrt
The result is: 3
```

参数为负

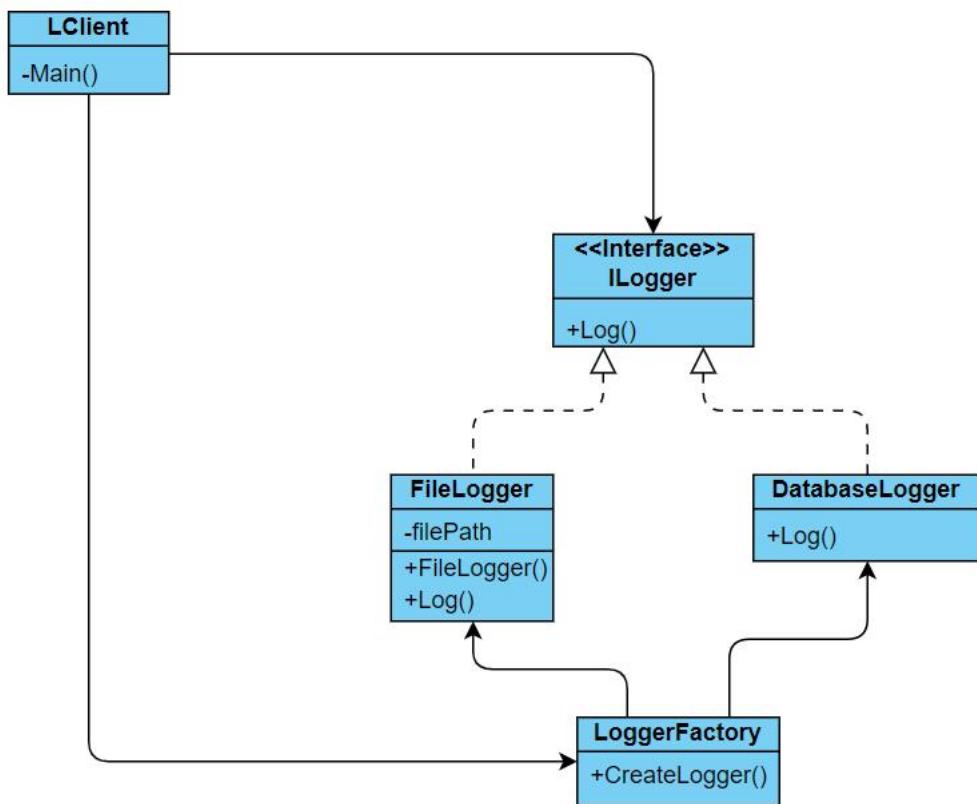
```
D:\Visual Studio\software design\simpleCalculator\IUnaryClient\bin\Debug\IUnaryClient.exe
Enter the number: -9
Enter the operation (abs, neg, rec, sqrt, square): sqrt
An error occurred: Cannot take square root of negative number.
```

Square

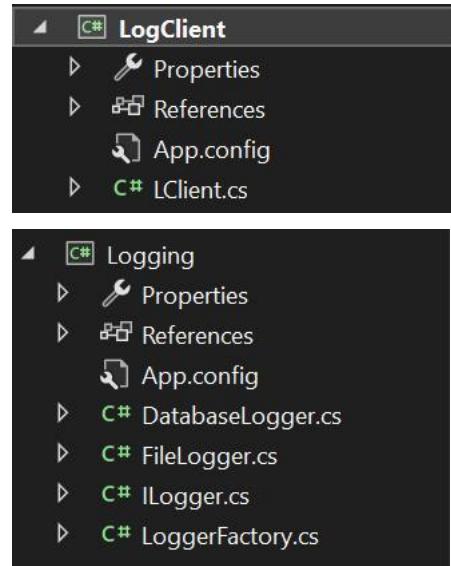
```
D:\Visual Studio\software design\simpleCalculator\IUnaryClient\bin\Debug\IUnaryClient.exe
Enter the number: -9
Enter the operation (abs, neg, rec, sqrt, square): square
The result is: 81
```

(5) 采用工厂方法模式设计实现计算器的写日志功能，将日志写入到本地文档，要考虑写日志功能的扩展性，即后期可能需要将日志写入数据库。

①类图



②代码结构



③核心类代码

Project Logging

Interface ILoger

The screenshot shows a Microsoft Visual Studio interface with the following components:

- Code Editor:** Displays the `ILogger.cs` file from the `Logging` namespace. The code defines a public interface `ILogger` with a single method `Log(string message)`.
- Solution Explorer:** Shows the project structure with files like `App.config`, `DatabaseLogger.cs`, `FileLogger.cs`, and `LoggerFactory.cs`.
- Properties Window:** Shows the properties for `ILogger.cs`, including build action (Compile), copy to output (Do not copy), and advanced settings.
- Output Window:** Displays the command-line output of the build process, showing the compilation of `LogClient.exe` and its dependencies.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Logging
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      * Create an interface for Log
13      */
14     public interface ILogger
15     {
16         void Log(string message);
17     }
18 }
```

```
Show output from: Debug
LogClient.exe [CLB v4.0.30319; LogClient.exe]: Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Core\v4.0_4.0.0.0_b77a5c561934e009\System.Core.dll'. Skipped loading symbols. Module is optimized.
LogClient.exe [CLB v4.0.30319; LogClient.exe]: Loaded 'D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\library.dll'. Symbols loaded.
LogClient.exe [CLB v4.0.30319; LogClient.exe]: Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Xml\v4.0_4.0.0.0_b77a5c561934e009\System.Xml.dll'. Skipped loading symbols. Module is optimized.
The program '[29808] LogClient.exe: Program Trace' has exited with code 0 (0x0).
The program '[29808] LogClient.exe' has exited with code 0 (0x0).
```

Class FileLogger

```

1  using System;
2  using System.IO;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  using System.Configuration;
8
9  namespace Logging
10 {
11     /**
12      * Author: arborshield
13      * Created by: on 2023/4/2.
14      * FileLogger
15     */
16     public class FileLogger : ILoggger
17     {
18         private string filePath;
19
20         public FileLogger(string filePath)
21         {
22             this.filePath = filePath;
23             if (!Directory.Exists(Path.GetDirectoryName(filePath)))
24             {
25                 Directory.CreateDirectory(Path.GetDirectoryName(filePath));
26             }
27         }
28     }

```

No issues found

Show output from: Debug

```

'LogClient.exe' (CLR v4.0.30319; LogClient.exe): Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Core\v4.0_4.0.0.0_b7a5c561934e089\System.Core.dll'. Skipped loading symbols. Module is optimized.
'LogClient.exe' (CLR v4.0.30319; LogClient.exe): Loaded 'D:\Visual Studio\software design\simpleCalculator\LogClient\bin\debug\lUnary.dll'. Symbols loaded.
'LogClient.exe' (CLR v4.0.30319; LogClient.exe): Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Xml\v4.0_4.0.0.0_b77a5c561934e089\System.Xml.dll'. Skipped loading symbols. Module is optimized.
The program '[29868] LogClient.exe: Program Trace' has exited with code 0 (0x0).
The program '[29868] LogClient.exe' has exited with code 0 (0x0).

```

```

13     * Created by: on 2023/4/2.
14     * FileLogger
15    */
16    public class FileLogger : ILoggger
17    {
18        private string filePath;
19
20        public FileLogger(string filePath)
21        {
22            this.filePath = filePath;
23            if (!Directory.Exists(Path.GetDirectoryName(filePath)))
24            {
25                Directory.CreateDirectory(Path.GetDirectoryName(filePath));
26            }
27        }
28
29        public void Log(string message)
30        {
31            using (StreamWriter writer = File.AppendText(filePath))
32            {
33                writer.WriteLine($"{DateTime.Now}: {message}");
34            }
35        }
36    }

```

No issues found

Show output from: Debug

```

'LogClient.exe' (CLR v4.0.30319; LogClient.exe): Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Core\v4.0_4.0.0.0_b7a5c561934e089\System.Core.dll'. Skipped loading symbols. Module is optimized.
'LogClient.exe' (CLR v4.0.30319; LogClient.exe): Loaded 'D:\Visual Studio\software design\simpleCalculator\LogClient\bin\debug\lUnary.dll'. Symbols loaded.
'LogClient.exe' (CLR v4.0.30319; LogClient.exe): Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Xml\v4.0_4.0.0.0_b77a5c561934e089\System.Xml.dll'. Skipped loading symbols. Module is optimized.
The program '[29868] LogClient.exe: Program Trace' has exited with code 0 (0x0).
The program '[29868] LogClient.exe' has exited with code 0 (0x0).

```

Class DatabaseLogger

The screenshot shows the Microsoft Visual Studio IDE interface. The code editor displays the `DatabaseLogger.cs` file, which contains the following C# code:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Logging
8  {
9      /**
10      * Author: arborshield
11      * Created by on 2023/4/2.
12      * Database Logger
13      */
14     public class DatabaseLogger : ILogger
15     {
16         /**
17         * Author: arborshield
18         * Created by on 2023/4/2.
19         * Database Logger
20         */
21         public void Log(string message)
22         {
23             // 将日志写入数据库的代码
24         }
25     }
26 }
```

The Solution Explorer window shows the project structure with files like `App.config`, `FileLogger.cs`, `ILogger.cs`, and `LoggerFactory.cs`. The Properties window shows the file properties for `DatabaseLogger.cs`. The Output window shows the build logs.

Class Logger Factory

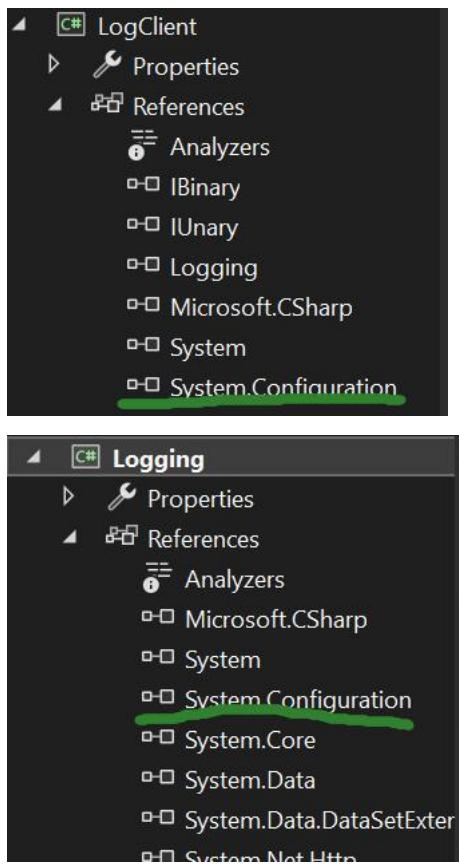
The screenshot shows the Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Architecture, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The toolbar below has icons for file operations like Open, Save, and Print. The main window displays two code files: IClient.cs and LoggerFactory.cs. IClient.cs contains a class definition for Logging. LoggerFactory.cs contains a static CreateLogger method that returns a FileLogger based on configuration settings. The Solution Explorer on the right shows the project structure with files like App.config, DatabaseLogger.cs, FileLogger.cs, ILogger.cs, and LoggerFactory.cs. The Properties window shows the file properties for LoggerFactory.cs. The Output window at the bottom shows the build log.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Configuration;
7  using System.Reflection;
8
9  namespace Logging
10 {
11     /**
12      * Author: arborshield
13      * Created by on 2023/4/2.
14      * Logger Factory
15     */
16     public class LoggerFactory
17     {
18         2 references
19         public static ILogger CreateLogger()
20         {
21             string loggerType = ConfigurationManager.AppSettings["LoggerType"];
22
23             if (loggerType == "FileLogger")
24             {
25                 string filePath = ConfigurationManager.AppSettings["LogFilePath"];
26                 return new FileLogger(filePath);
27             }
28             else
29             {
30                 throw new NotSupportedException($"Logger type '{loggerType}' is not supported.");
31             }
32         }
33     }
34 }
```

This screenshot shows the same Visual Studio environment with a modified LoggerFactory.cs file. The CreateLogger method now throws a NotImplementedException instead of a NotSupportedException. The rest of the code remains the same. The Solution Explorer, Properties window, and Output window are also visible.

```
1  using System.Reflection;
2
3  namespace Logging
4  {
4  /**
5      * Author: arborshield
6      * Created by on 2023/4/2.
7      * Logger Factory
8  */
9  public class LoggerFactory
10 {
11     2 references
12     public static ILogger CreateLogger()
13     {
14         string loggerType = ConfigurationManager.AppSettings["LoggerType"];
15
16         if (loggerType == "FileLogger")
17         {
18             string filePath = ConfigurationManager.AppSettings["LogFilePath"];
19             return new FileLogger(filePath);
20         }
21         else
22         {
23             throw new NotImplementedException();
24         }
25     }
26 }
```

④客户端代码



Project LogClient

Class LClient

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

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Architecture, Test, Analyze, Tools, Extensions, Window, Help.
- Toolbar:** Standard toolbar with icons for New, Open, Save, Print, etc.
- Solution Explorer:** Shows the solution structure: simpleCalculator (6 of 6 p), Ibinary, IbinaryClient, Ibinary, IbinaryClient, LogClient (Properties, References, App.config, LClient.cs, Logging).
- Properties Window:** Advanced tab selected, showing Build Action: Compile, Copy to Output: Do not copy, Custom Tool, Custom Tool N.
- LClient.cs Code Editor:** Contains the following C# code:

```
1  using System;
2  using Ibinary;
3  using IBinary;
4  using Logging;
5  using System.Configuration;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 
11 namespace LogClient
12 {
13     /**
14      * Author: arborshield
15      * Created by: on 2023/4/2.
16      * ClientTest
17      */
18     class LClient
19     {
20         static void Main(string[] args)
21         {
22             try
23             {
24                 Console.WriteLine("Unary or Binary?");
25                 string op = Convert.ToString(Console.ReadLine());
26 
27                 if (op == "Unary")
28                 {
29                     if (op == "Unary")
30                     {
31                         Console.WriteLine("Enter the number: ");
32                         double num = Convert.ToDouble(Console.ReadLine());
33 
34                         Console.WriteLine("Enter the operation (abs, neg, rec, sqrt, square): ");
35                         string operationName = Console.ReadLine();
36                         string operationTypeName = ConfigurationManager.AppSettings[operationName];
37 
38                         IUnaryOperation operation = (IUnaryOperation)Activator.CreateInstance(Type.GetType(operationTypeName));
39 
40                         double result = operation.Calculate(num);
41 
42                         Console.WriteLine($"The result is: {result}");
43                         ILogger logger = LoggerFactory.CreateLogger();
44                         logger.LogInformation($"Operation {num} {operationTypeName} = {result}");
45 
46                     if (op == "Binary")
47                     {
48                         Console.WriteLine("Enter the first number: ");
49                         double num1 = Convert.ToDouble(Console.ReadLine());
50 
51                         Console.WriteLine("Enter the second number: ");
52                         double num2 = Convert.ToDouble(Console.ReadLine());
53 
54                         Console.WriteLine("Enter the operation (+, -, *, /, Log): ");
55                         string operationName = Console.ReadLine();
56 
57                         IBinaryOperation operation = (IBinaryOperation)Activator.CreateInstance(Type.GetType(operationName));
58 
59                         double result = operation.Calculate(num1, num2);
60 
61                         Console.WriteLine($"The result is: {result}");
62                         ILogger logger = LoggerFactory.CreateLogger();
63                         logger.LogInformation($"Operation {num1} {operationName} {num2} = {result}");
64 
65                     }
66                 }
67             }
68         }
69     }
70 }
```

Output Window: Shows the build log output for LogClient.exe.

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

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Architecture, Test, Analyze, Tools, Extensions, Window, Help.
- Toolbar:** Standard toolbar with icons for New, Open, Save, Print, etc.
- Solution Explorer:** Shows the solution structure: simpleCalculator (6 of 6 p), Ibinary, IbinaryClient, Ibinary, IbinaryClient, LogClient (Properties, References, App.config, LClient.cs, Logging).
- Properties Window:** Advanced tab selected, showing Build Action: Compile, Copy to Output: Do not copy, Custom Tool, Custom Tool N.
- LClient.cs Code Editor:** Contains the following C# code, which is identical to the one in the first screenshot but includes additional logic for binary operations:

```
1  using System;
2  using Ibinary;
3  using IBinary;
4  using Logging;
5  using System.Configuration;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 
11 namespace LogClient
12 {
13     /**
14      * Author: arborshield
15      * Created by: on 2023/4/2.
16      * ClientTest
17      */
18     class LClient
19     {
20         static void Main(string[] args)
21         {
22             try
23             {
24                 Console.WriteLine("Unary or Binary?");
25                 string op = Convert.ToString(Console.ReadLine());
26 
27                 if (op == "Unary")
28                 {
29                     if (op == "Unary")
30                     {
31                         Console.WriteLine("Enter the number: ");
32                         double num = Convert.ToDouble(Console.ReadLine());
33 
34                         Console.WriteLine("Enter the operation (abs, neg, rec, sqrt, square): ");
35                         string operationName = Console.ReadLine();
36                         string operationTypeName = ConfigurationManager.AppSettings[operationName];
37 
38                         IUnaryOperation operation = (IUnaryOperation)Activator.CreateInstance(Type.GetType(operationTypeName));
39 
40                         double result = operation.Calculate(num);
41 
42                         Console.WriteLine($"The result is: {result}");
43                         ILogger logger = LoggerFactory.CreateLogger();
44                         logger.LogInformation($"Operation {num} {operationTypeName} = {result}");
45 
46                     if (op == "Binary")
47                     {
48                         Console.WriteLine("Enter the first number: ");
49                         double num1 = Convert.ToDouble(Console.ReadLine());
50 
51                         Console.WriteLine("Enter the second number: ");
52                         double num2 = Convert.ToDouble(Console.ReadLine());
53 
54                         Console.WriteLine("Enter the operation (+, -, *, /, Log): ");
55                         string operationName = Console.ReadLine();
56 
57                         IBinaryOperation operation = (IBinaryOperation)Activator.CreateInstance(Type.GetType(operationName));
58 
59                         double result = operation.Calculate(num1, num2);
60 
61                         Console.WriteLine($"The result is: {result}");
62                         ILogger logger = LoggerFactory.CreateLogger();
63                         logger.LogInformation($"Operation {num1} {operationName} {num2} = {result}");
64 
65                     }
66                 }
67             }
68         }
69     }
70 }
```

Output Window: Shows the build log output for LogClient.exe.

```

    47     {
    48         Console.WriteLine("Enter the first number: ");
    49         double num1 = Convert.ToDouble(Console.ReadLine());
    50
    51         Console.WriteLine("Enter the second number: ");
    52         double num2 = Convert.ToDouble(Console.ReadLine());
    53
    54         Console.WriteLine("Enter the operation (+, -, *, /, Log): ");
    55         string operationName = Console.ReadLine();
    56
    57         IBinaryOperation operation = BinaryOperationFactory.CreateOperation(operationName);
    58
    59         double result = operation.Calculate(num1, num2);
    60
    61         Console.WriteLine($"The result is: {result}");
    62         ILogger logger = LoggerFactory.CreateLogger();
    63         logger.LogInformation($"Operation {num1} {operationName} {num2} = {result}");
    64
    65     }
    66     catch (Exception ex)
    67     {
    68         Console.WriteLine($"An error occurred: {ex.Message}");
    69     }
    70
    71     Console.ReadKey();
    72 }
    73 }
    74

```

Output window shows the application running and exiting with code 0.

⑤配置文件

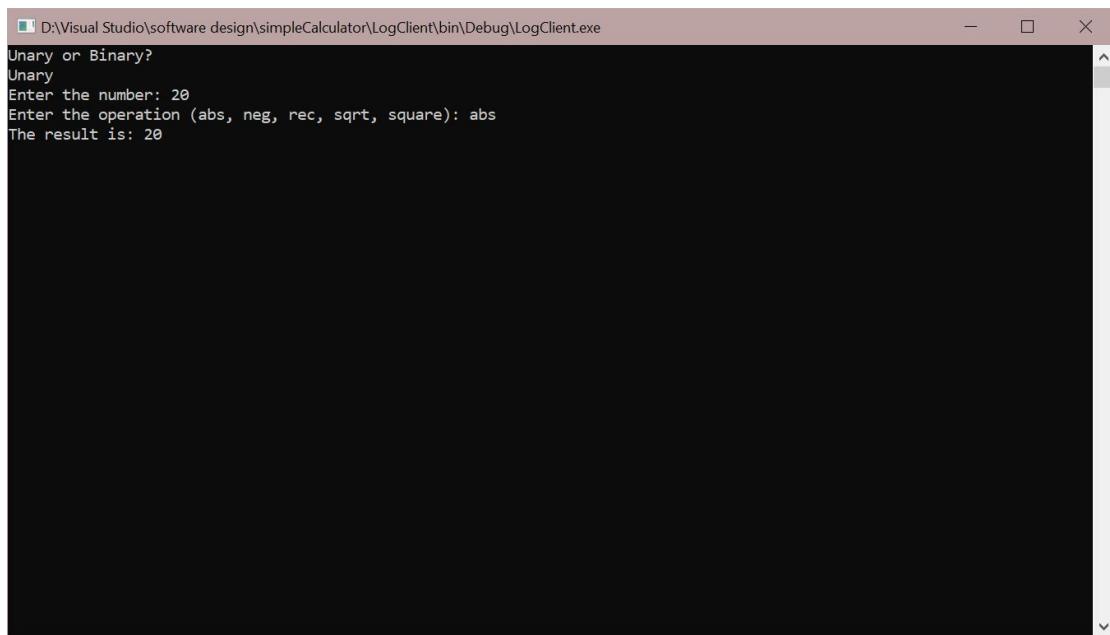
```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <appSettings>
        <add key="+" value="IBinary.Addition, IBinary" />
        <add key="-" value="IBinary.Subtraction, IBinary" />
        <add key="*" value="IBinary.Multiplication, IBinary" />
        <add key="/" value="IBinary.Division, IBinary" />
        <add key="Log" value="IBinary.Logarithm, IBinary" />
        <add key="abs" value="IUnary.AbsoluteOperation, IUnary" />
        <add key="neg" value="IUnary.NegatedOperation, IUnary" />
        <add key="rec" value="IUnary.ReciprocalOperation, IUnary" />
        <add key="sqrt" value="IUnary.SqrtOperation, IUnary" />
        <add key="square" value="IUnary.SquareOperation, IUnary" />
        <add key="loggerType" value="FileLogger" />
        <add key="logFilePath" value="D:\Logs\Calculator.log" />
    </appSettings>
</configuration>

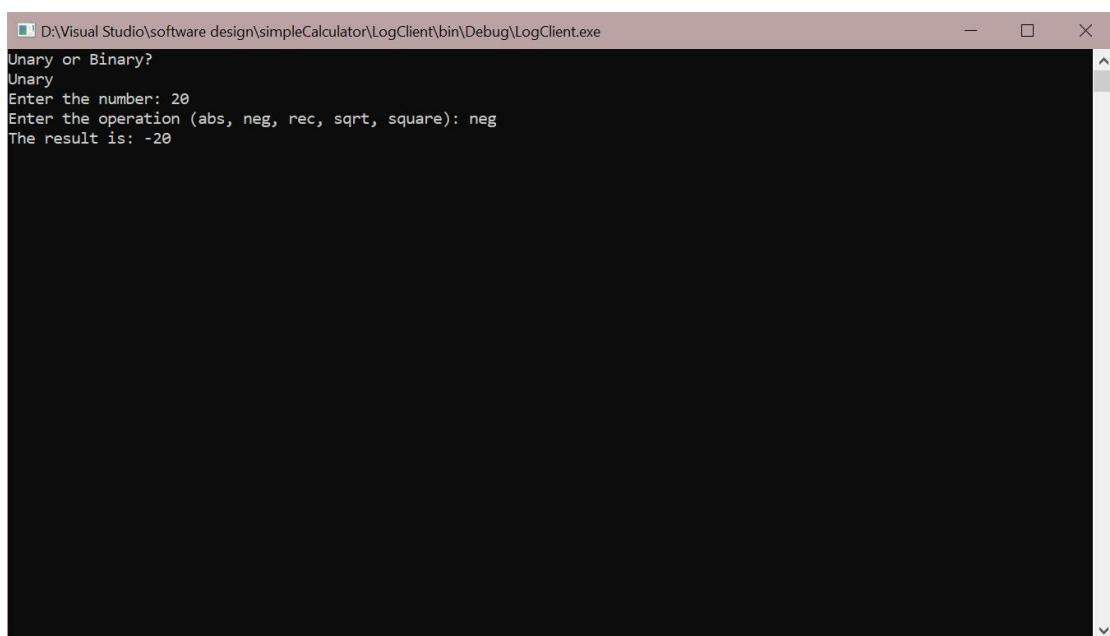
```

Output window shows the application running and exiting with code 0.

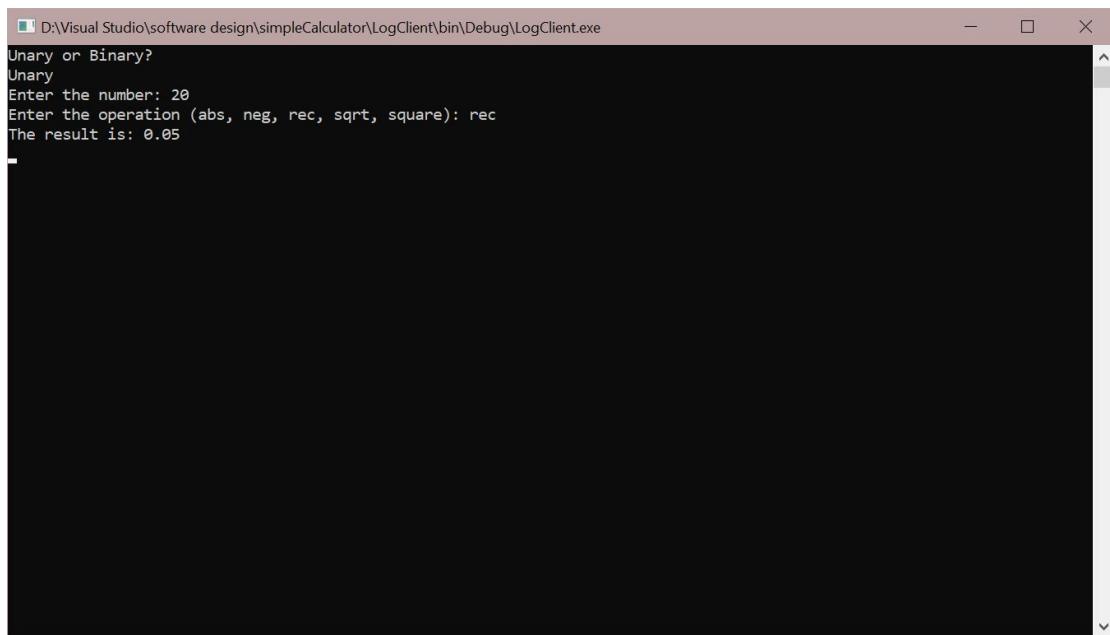
⑥客户端测试



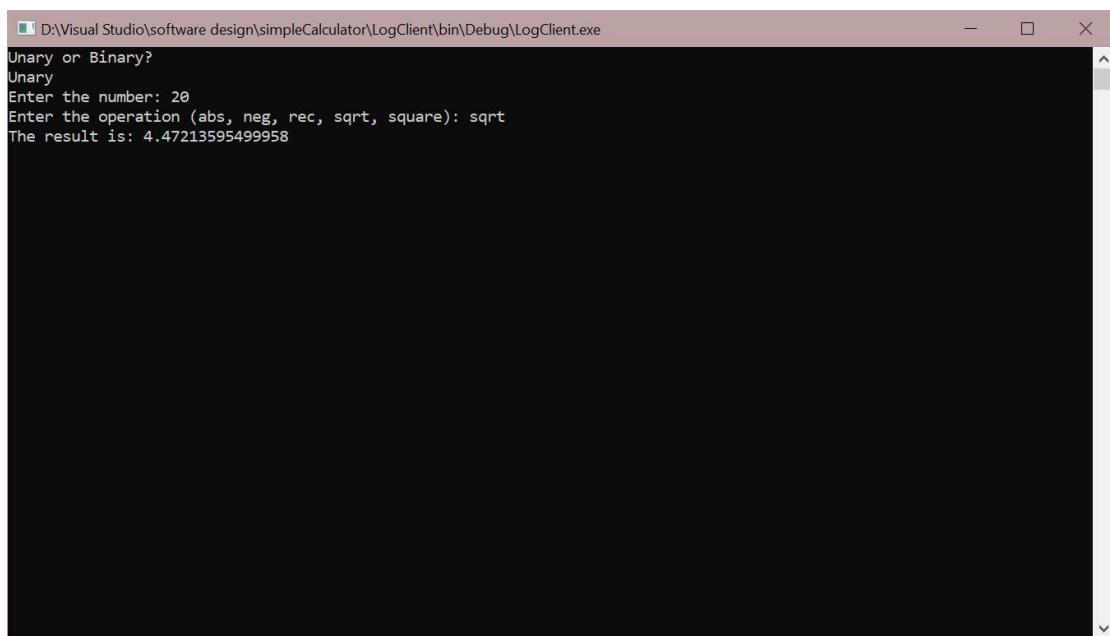
```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Unary
Enter the number: 20
Enter the operation (abs, neg, rec, sqrt, square): abs
The result is: 20
```



```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Unary
Enter the number: 20
Enter the operation (abs, neg, rec, sqrt, square): neg
The result is: -20
```



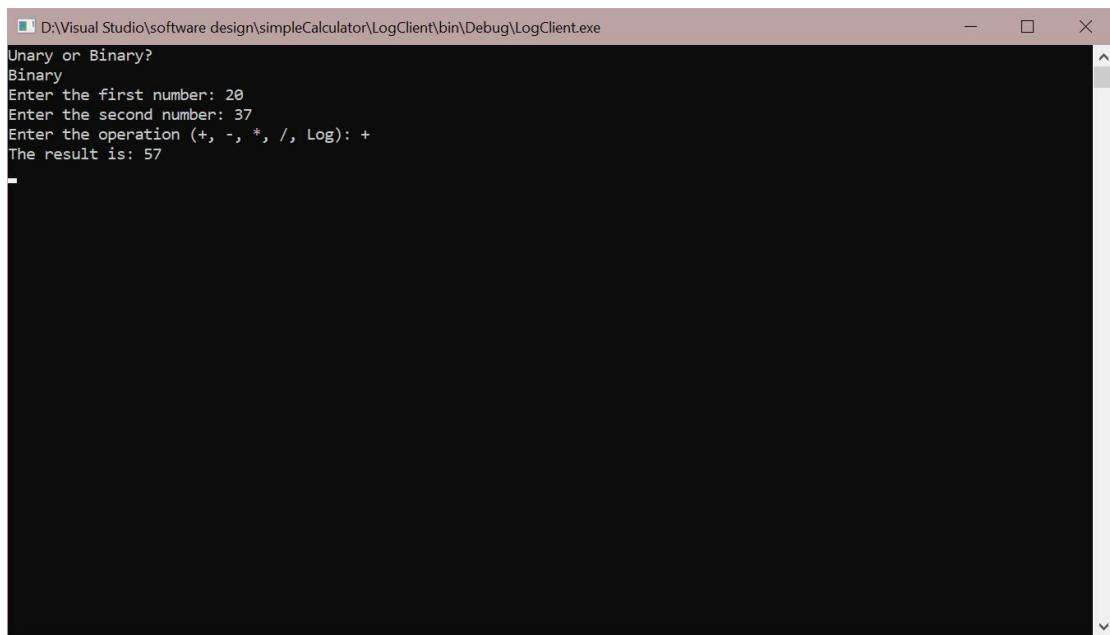
```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Unary
Enter the number: 20
Enter the operation (abs, neg, rec, sqrt, square): rec
The result is: 0.05
```



```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Unary
Enter the number: 20
Enter the operation (abs, neg, rec, sqrt, square): sqrt
The result is: 4.47213595499958
```

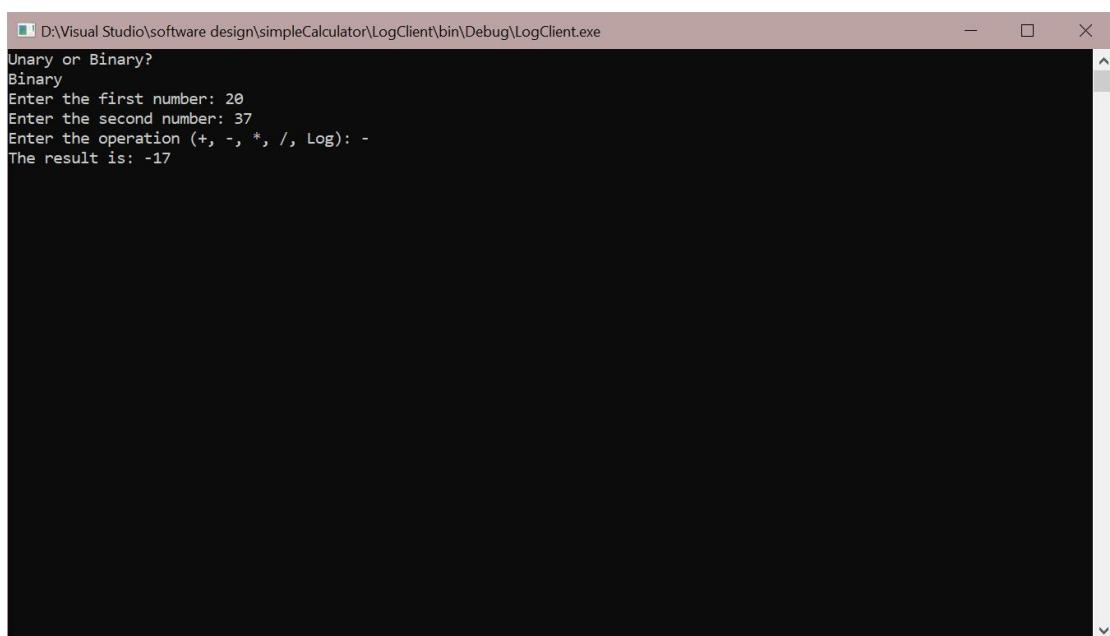
```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Unary
Enter the number: -37
Enter the operation (abs, neg, rec, sqrt, square): sqrt
An error occurred: Cannot take square root of negative number.
```

```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Unary
Enter the number: 37
Enter the operation (abs, neg, rec, sqrt, square): square
The result is: 1369
```



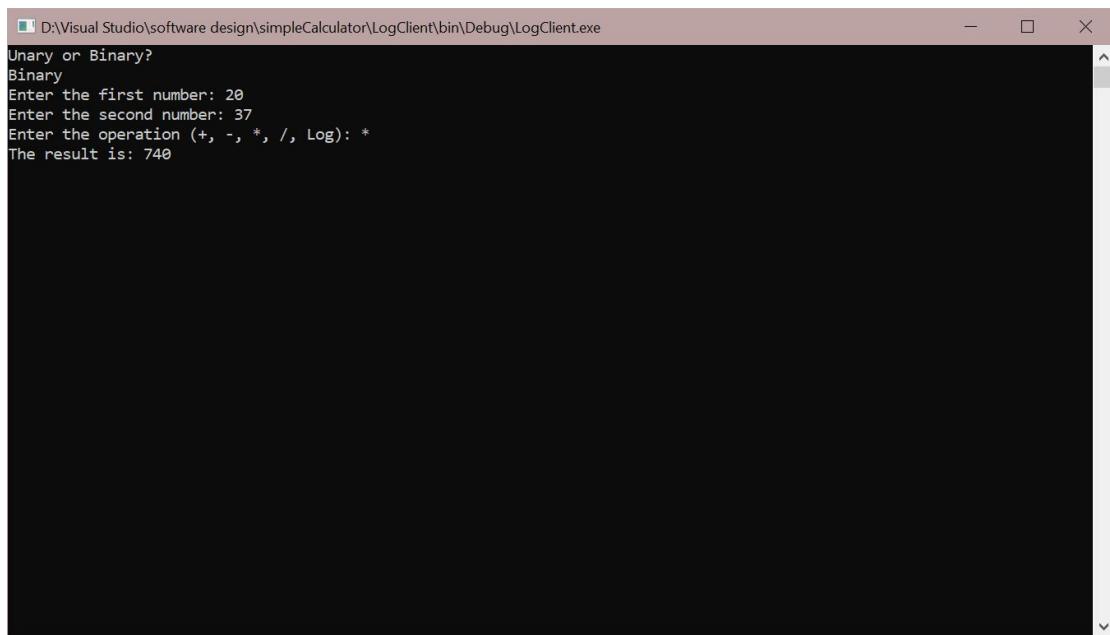
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe

```
Unary or Binary?
Binary
Enter the first number: 20
Enter the second number: 37
Enter the operation (+, -, *, /, Log): +
The result is: 57
```

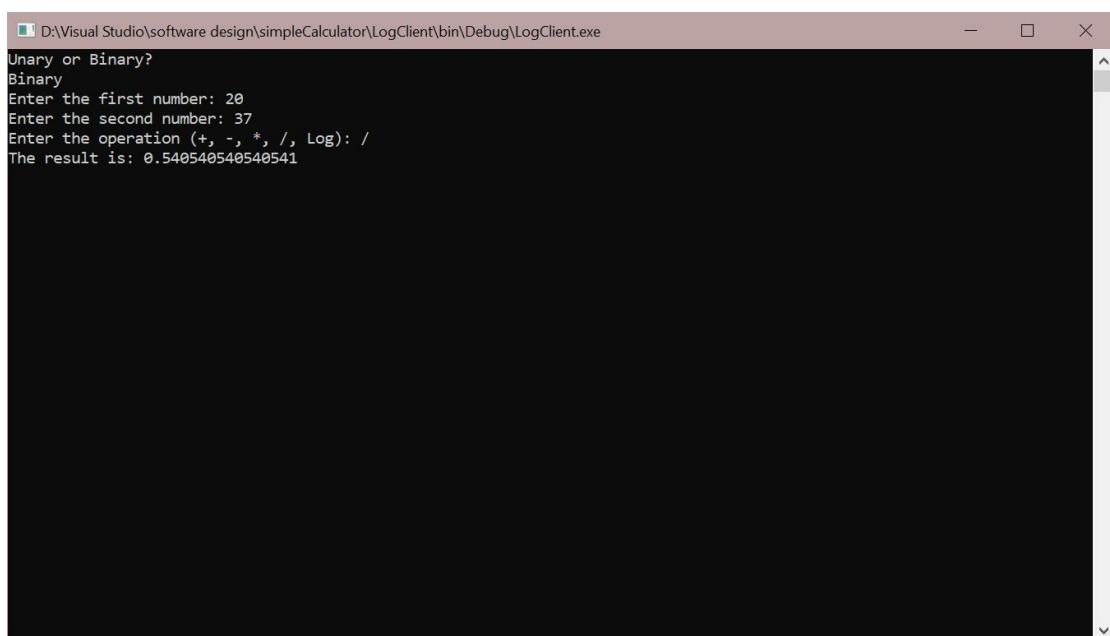


D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe

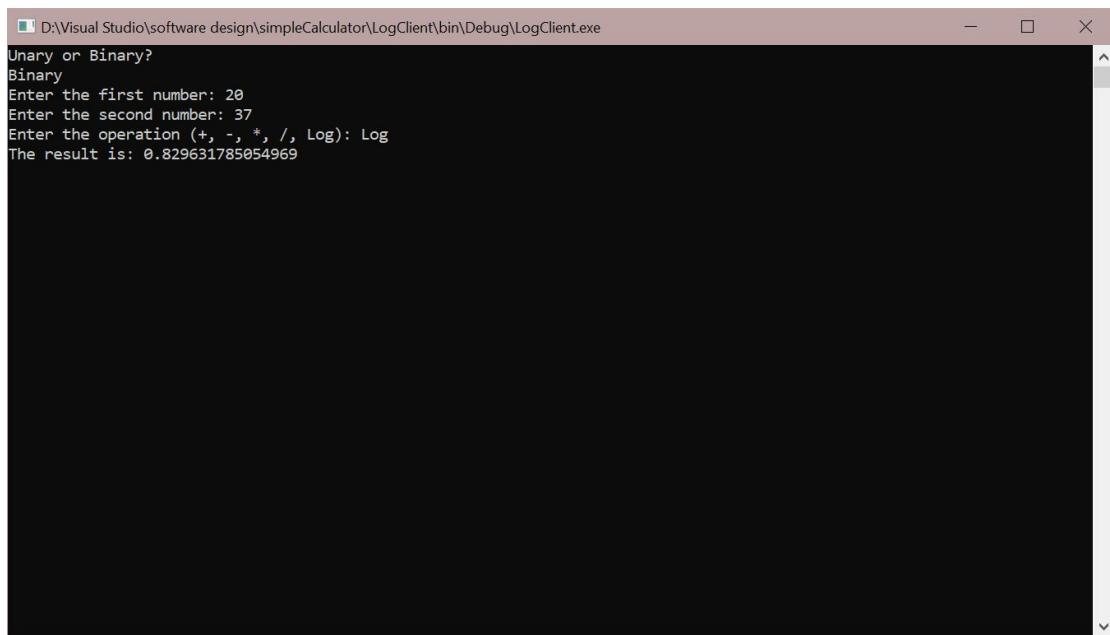
```
Unary or Binary?
Binary
Enter the first number: 20
Enter the second number: 37
Enter the operation (+, -, *, /, Log): -
The result is: -17
```



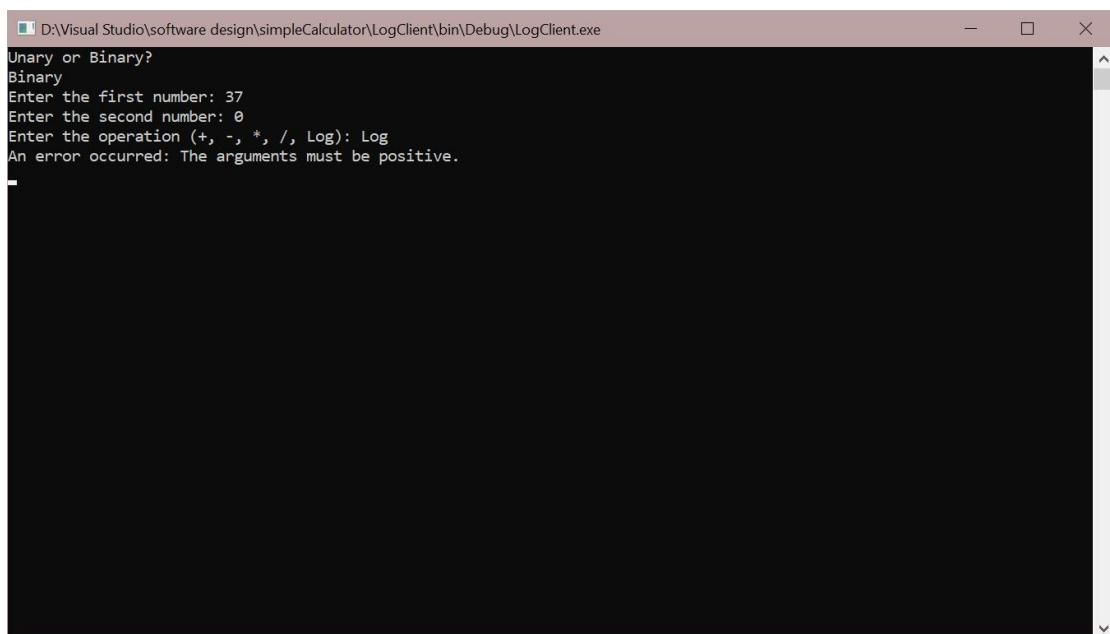
```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Binary
Enter the first number: 20
Enter the second number: 37
Enter the operation (+, -, *, /, Log): *
The result is: 740
```



```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Binary
Enter the first number: 20
Enter the second number: 37
Enter the operation (+, -, *, /, Log): /
The result is: 0.540540540541
```



```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Binary
Enter the first number: 20
Enter the second number: 37
Enter the operation (+, -, *, /, Log): Log
The result is: 0.829631785054969
```



```
D:\Visual Studio\software design\simpleCalculator\LogClient\bin\Debug\LogClient.exe
Unary or Binary?
Binary
Enter the first number: 37
Enter the second number: 0
Enter the operation (+, -, *, /, Log): Log
An error occurred: The arguments must be positive.
```

日志如下：



Calculator.log - Notepad

File Edit Format View Help

```
4/2/2023 5:33:55 PM: Operation 20 IUnary.AbsoluteOperation, IUnary = 20
4/2/2023 5:34:12 PM: Operation 20 IUnary.AbsoluteOperation, IUnary = 20
4/2/2023 5:34:35 PM: Operation 20 IUnary.NegateOperation, IUnary = -20
4/2/2023 5:34:59 PM: Operation 20 IUnary.ReciprocalOperation, IUnary = 0.05
4/2/2023 5:35:21 PM: Operation 20 IUnary.SqrtOperation, IUnary = 4.47213595499958
4/2/2023 5:36:53 PM: Operation 37 IUnary.SquareOperation, IUnary = 1369
4/2/2023 5:37:44 PM: Operation 20 + 37 = 57
4/2/2023 5:38:22 PM: Operation 20 - 37 = -17
4/2/2023 5:38:43 PM: Operation 20 * 37 = 740
4/2/2023 5:39:00 PM: Operation 20 / 37 = 0.540540540540541
4/2/2023 5:39:22 PM: Operation 20 Log 37 = 0.829631785054969
```

< >

Ln 12, Col 1 100% Windows (CRLF) UTF-8