

目 录

一、 实验内容	1
二、 类图及代码结构	2
三、 代码实现	7
四、 运行效果	35
五、 课设心得	41

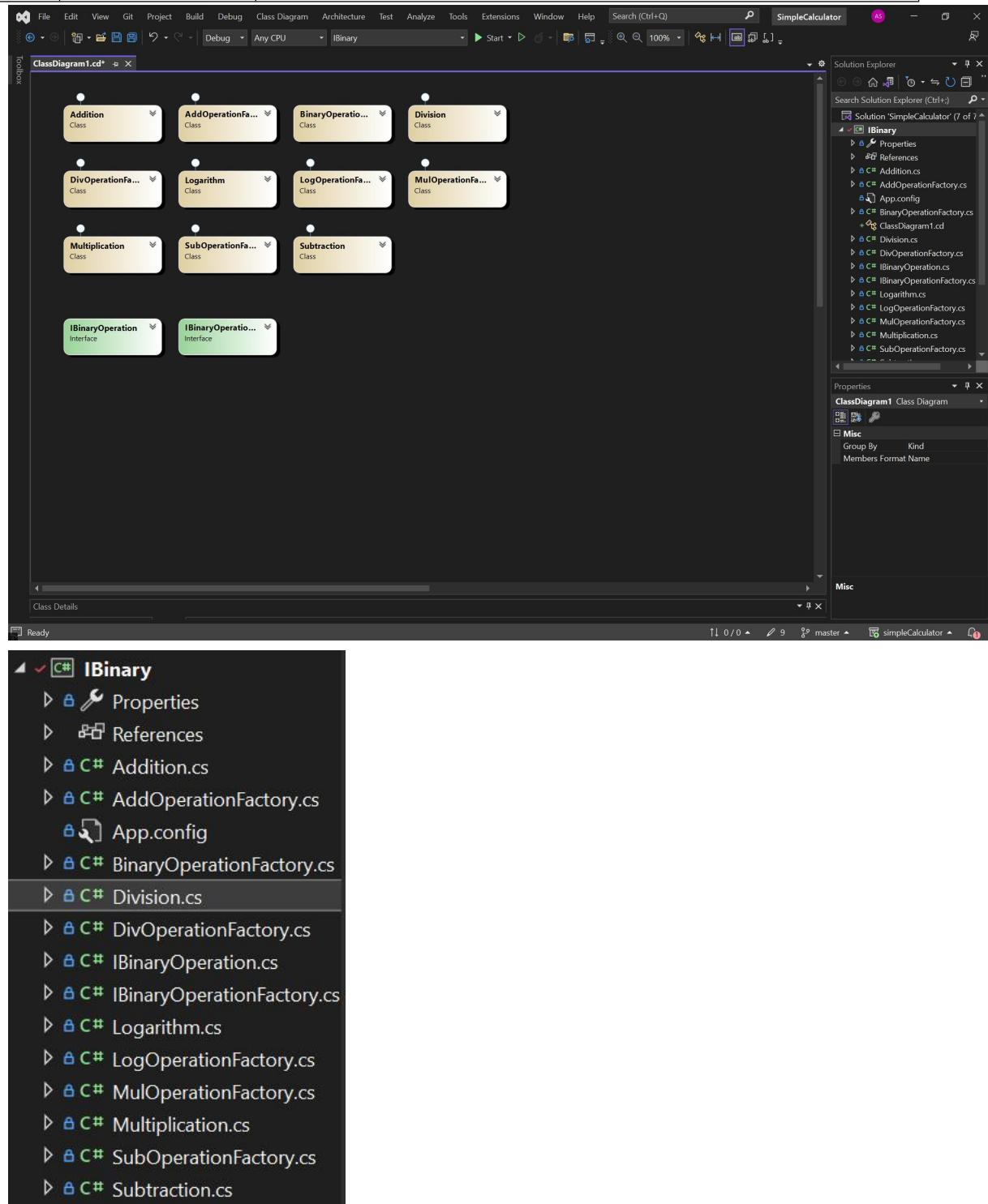
一、实验内容

基本环境包括 IDE 开发工具 Visual Studio、数据库 Sql Server Express LocalDb、单元测试框架 MsTest、文档 Word, Orm 数据访问框架 Simple.Data 和 EF 框架, IOC 框架 Autofac 等。

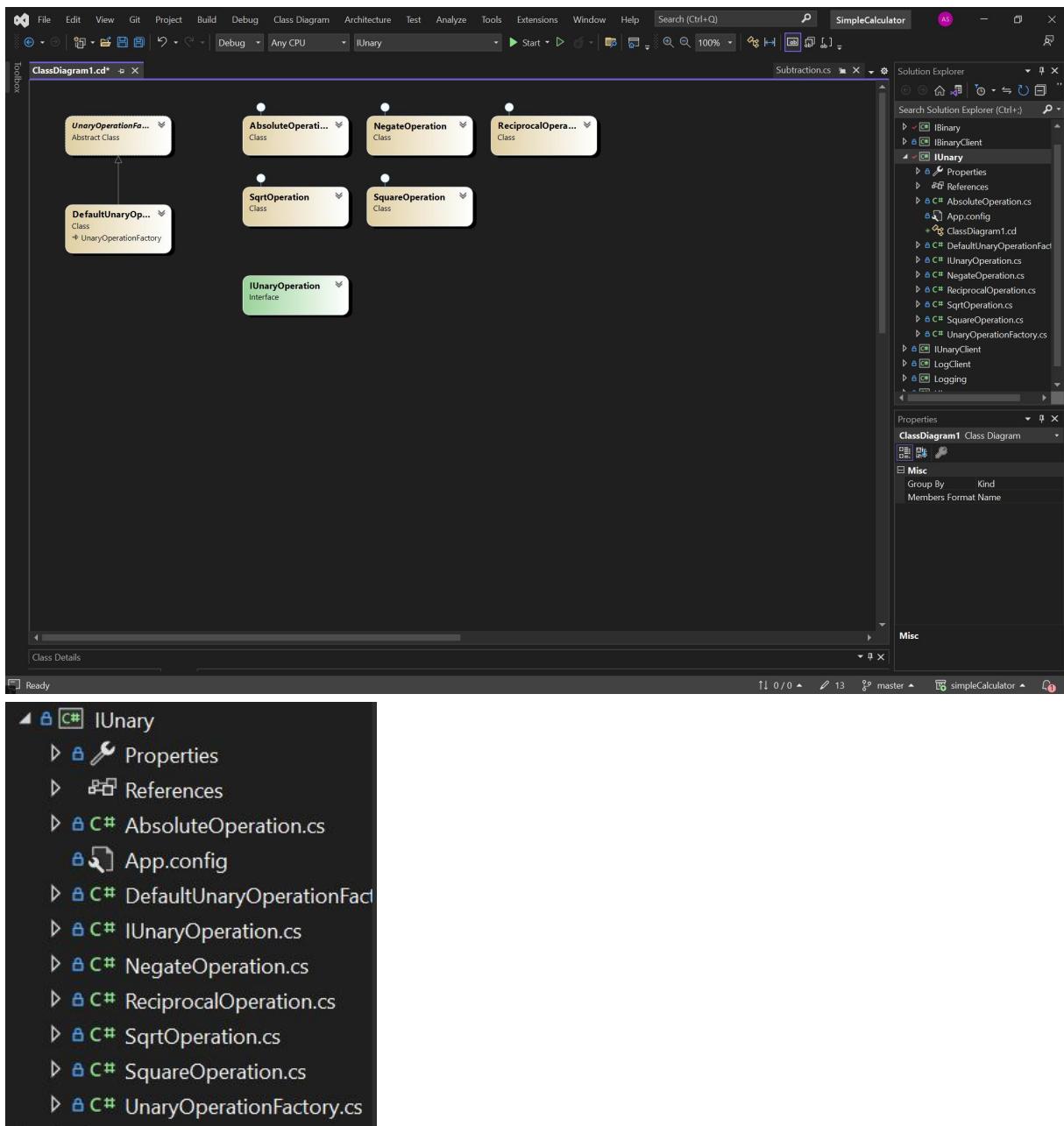
序号	内容	具体要求	考核比例
1	计算器的领域模型 1	采用 工厂方法模式 实现计算器 二元计算领域模型 , 实现加、减、乘、除、对数 $\log_a x$ 、指数 x^y 等二元运算	10
2	对于领域模型 1 的单元测试	对领域模型 1 进行单元测试	5
3	计算器的领域模式 2	采用工厂方法模式实现计算器 一元计算领域模型 , 实现取反、求 $1/x$ 、 $2\sqrt{x}$ 、 x^2 、 10^x 、 $ x $ 、 \log 、 \ln 、 \exp 等一元运算	10
4	对于领域模型 2 的单元测试	对领域模型 2 进行单元测试	5
5	计算器的 UI 界面	使用 配置文件与反射机制 实现二元计算工厂对象和一元计算工厂对象的实例化	10
6	实现日志读写模块	将操作日志写入文本文档、使用 ADO.NET 写入数据库、使用 Simple.Data 和 EF 框架将计算器操作日志写入数据库	10
7	数据库日志读写模块单元测试	对于日志读写的多种不同的实现, 对其进行单元测试	10
8	在 UI 项目中增加日志读写功能	在 UI 项目中使用日志读写模块实现日志读写功能, 使用 IOC (Autofac) 实现日志读写对象的实例化	10
9	MEF 实现日志读写功能	重构 UI 项目和日志读写模块项目, 使用 MEF 实现日志读写功能 (类似实验 2 的第 2 部分)	15
10	实现计算器主题设置和切换	采用 抽象工厂模式 设计计算器主题功能, 定义至少两种主题, 每种主题下, 计算器的特性是不同的, 比如背景图, 字体颜色, 字体等不相同。	15

二、类图及代码结构

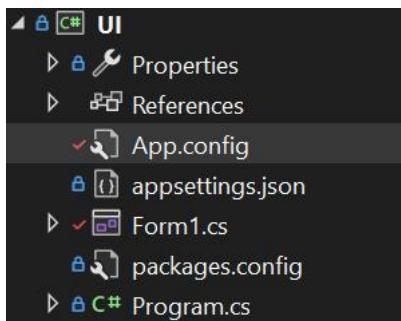
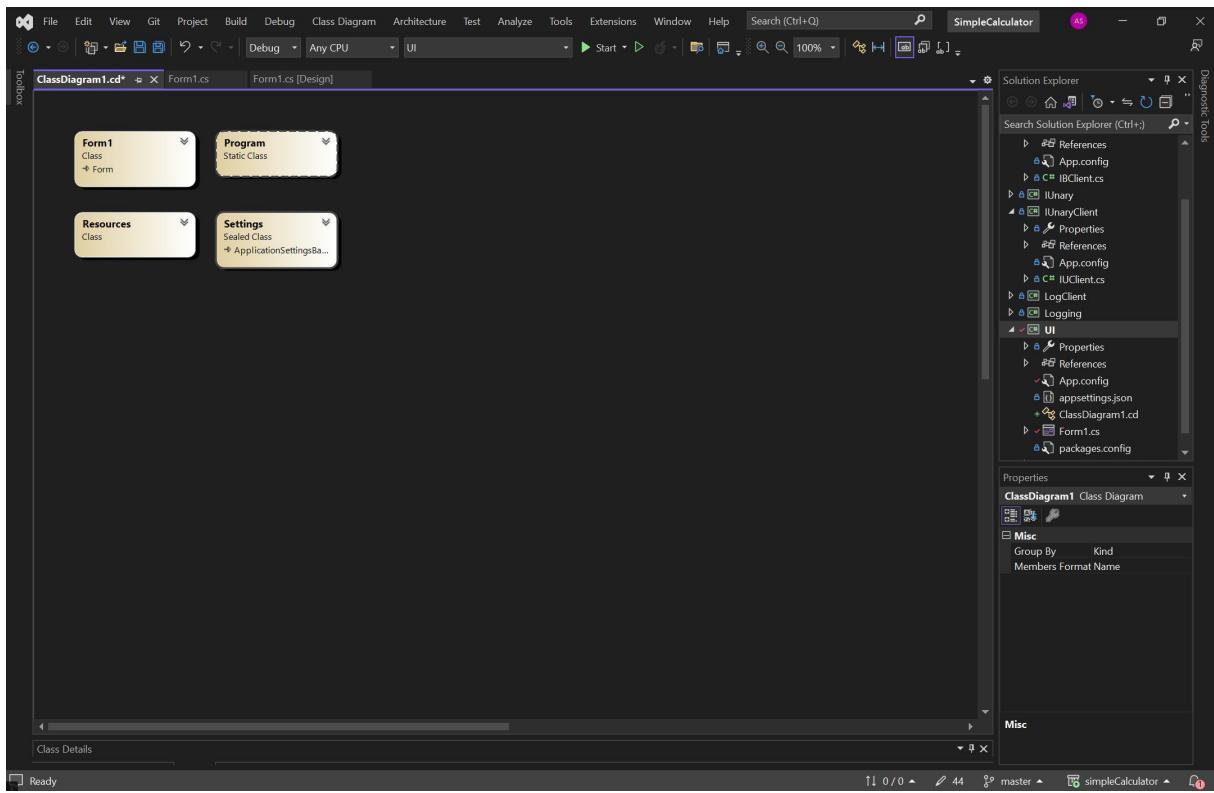
1	计算器的领域模型 1	采用工厂方法模式实现计算器二元计算领域模型，实现加、减、乘、除、对数 $\log_a x$ 、指数 x^y 等二元运算
---	------------	---



3	计算器的领域模式 2	采用工厂方法模式实现计算器 一元计算领域模型 , 实现取反、求 $1/x$ 、 $2\sqrt{x}$ 、 x^2 、 10^x 、 $ x $ 、 \log 、 \ln 、 \exp 等一元运算
---	------------	--



5	计算器的 UI 界面	使用配置文件与反射机制实现二元计算工厂对象和一元计算工厂对象的实例化
---	------------	------------------------------------



6	实现日志读写模块	将操作日志写入文本文档、使用 ADO.NET 写入数据库、使用 Simple.Data 和 EF 框架将计算器操作日志写入数据库
8	在 UI 项目中增加日志读写功能	在 UI 项目中使用日志读写模块实现日志读写功能，使用 IOC (Autofac) 实现日志读写对象的实例化

The screenshot shows the Microsoft Visual Studio interface. The main area displays a Class Diagram titled "ClassDiagram1.cd" with several classes and interfaces defined:

- Classes:** AdoNetLogger, ContainerConfig, EfLogger, FileLogger, Log, LogDbContext.
- Interface:** ILogger.

The Solution Explorer on the right shows the project structure for "SimpleCalculator" (4 of 4 projects):

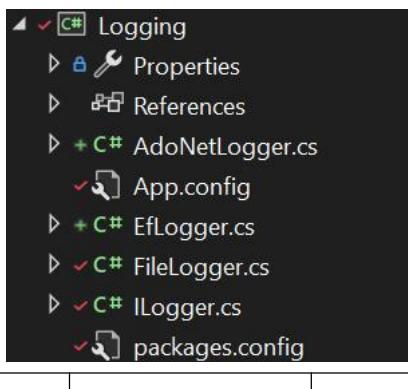
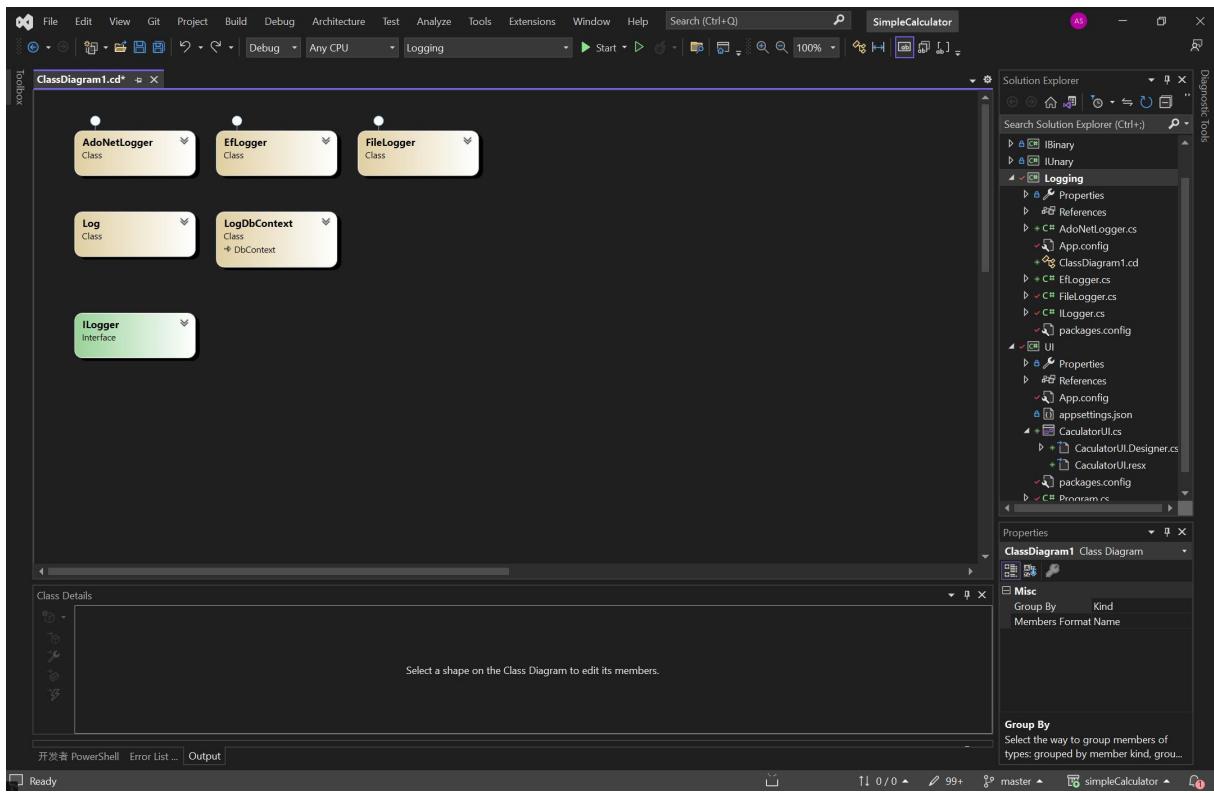
- Logging
 - Properties
 - References
 - AdoNetLogger.cs
 - App.config
 - ContainerConfig.cs
 - EfLogger.cs
 - FileLogger.cs
 - ILogger.cs
 - LoggerFactory.cs
 - packages.config
- Binary
- Unary

The Properties window shows "ClassDiagram1 Class Diagram".

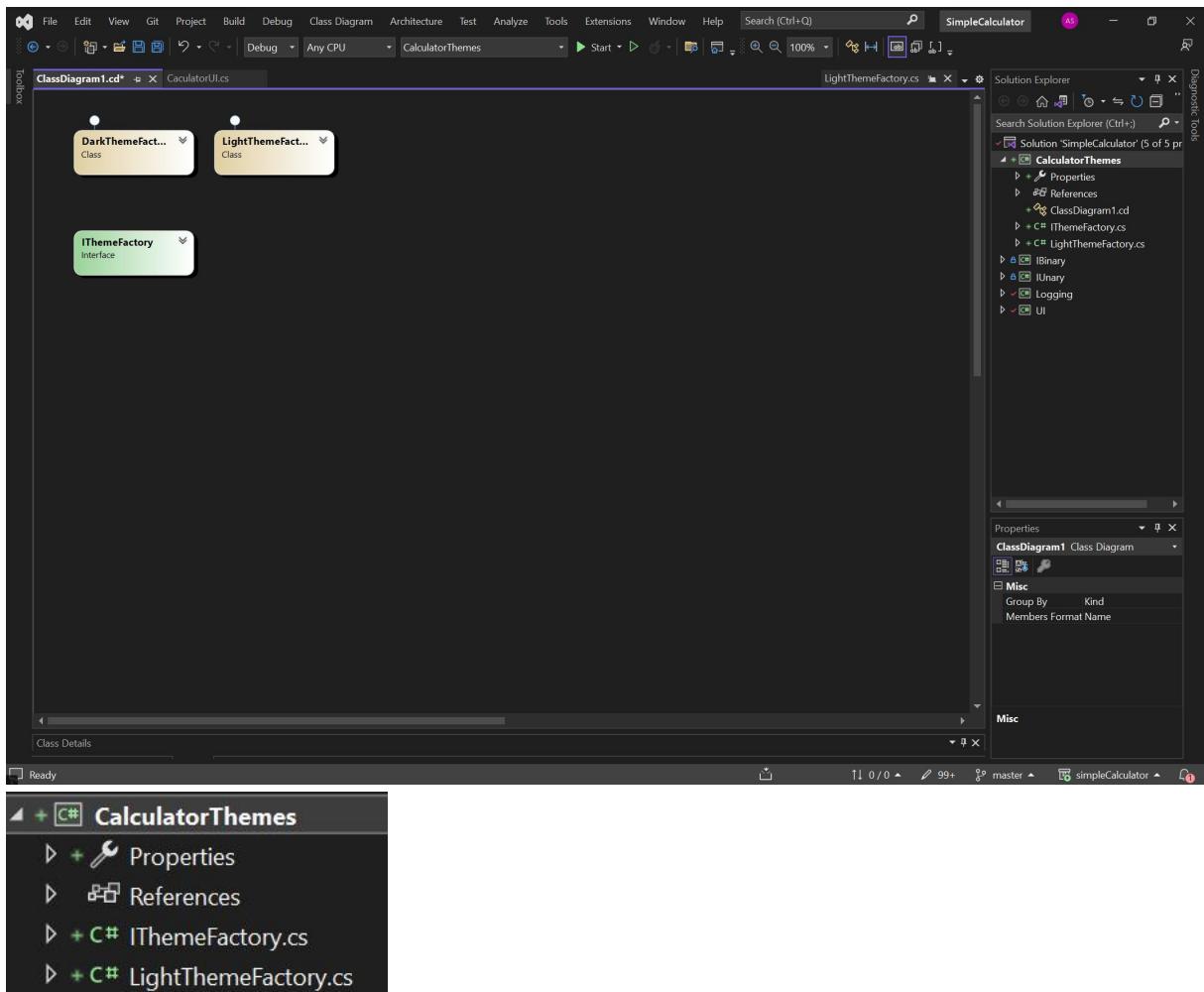
A detailed sidebar on the left lists the contents of the "Logging" folder:

- Properties
- References
- AdoNetLogger.cs
- App.config
- ContainerConfig.cs
- EfLogger.cs
- FileLogger.cs
- ILogger.cs
- LoggerFactory.cs
- packages.config

9	MEF 实现日志读写功能	重构 UI 项目和日志读写模块项目，使用 MEF 实现日志读写功能（类似于实验 2 的第 2 部分）
---	--------------	--



10	<p>实现计算器主题 设置和切换</p>	<p>采用抽象工厂模式设计计算器主题功能，定义至少两种主题，每种主题下，计算器的特性是不同的，比如背景图，字体颜色，字体等不相同。</p>
----	--------------------------	--



三、代码实现

1	计算器的领域模型 1	采用 工厂方法模式 实现计算器 二元计算领域模型 ，实现加、减、乘、除、对数 $\log_a x$ 、指数 x^y 等二元运算
---	------------	---

```

namespace IBinary
{
    public class Addition : IBinaryOperation
    {
        public double Calculate(double num1, double num2)
        {
            return num1 + num2;
        }
    }
}

namespace IBinary
{
    public class AddOperationFactory : IBinaryOperationFactory
    {
        public IBinaryOperation CreateOperation()
        {
            return new Addition();
        }
    }
}

```

The screenshot displays two instances of Microsoft Visual Studio. Both instances have the same solution structure:

- Solution Explorer:** Shows the solution 'SimpleCalculator' with seven projects:
 - IBinary
 - Addition.cs
 - BinaryOperationFactory.cs
 - Division.cs
 - DivOperationFactory.cs
 - IbinaryOperation.cs
 - IbinaryOperationFactory.cs
 - Logarithm.cs
 - LogOperationFactory.cs
 - MulOperationFactory.cs
 - Multiplication.cs
 - SubOperationFactory.cs
 - Subtraction.cs
- Properties:** Shows file properties for 'BinaryOperationFactory.cs' and 'Division.cs'.
- Advanced:** Shows build action, copy to output, and other advanced settings.
- Toolbox:** Standard .NET development toolbox.
- Status Bar:** Shows 'Ready' and various system status indicators.

Left Editor (BinaryOperationFactory.cs):

```

1  using System;
2  using System.Configuration;
3
4  namespace IBinary
5  {
6      public class BinaryOperationFactory
7      {
8          public static IBinaryOperation CreateOperation(string operatorName)
9          {
10             // Get the fully-qualified name of the operation class from the config file
11             string operationClassName = ConfigurationManager.AppSettings[operatorName];
12
13             // Use reflection to create an instance of the operation class
14             Type operationClassType = Type.GetType(operationClassName);
15             IBinaryOperation operation = (IBinaryOperation)Activator.CreateInstance(operationClassType);
16
17             return operation;
18         }
19     }
20 }
21
22

```

Right Editor (Division.cs):

```

1  using System;
2
3  namespace IBinary
4  {
5      public class Division : IBinaryOperation
6      {
7          public double Calculate(double num1, double num2)
8          {
9              if (num2 == 0)
10              {
11                  throw new DivideByZeroException("Cannot divide by zero.");
12              }
13
14              return num1 / num2;
15          }
16      }
17 }
18
19

```

```

namespace IBinary
{
    public class DivOperationFactory : IBinaryOperationFactory
    {
        public IBinaryOperation CreateOperation()
        {
            return new Division();
        }
    }
}

namespace IBinary
{
    public interface IBinaryOperation
    {
        double Calculate(double num1, double num2);
    }
}

```

```

using System;
namespace IBinary
{
    public interface IBinaryOperationFactory
    {
        IBinaryOperation CreateOperation();
    }
}

class Logarithm : IBinaryOperation
{
    public double Calculate(double num1, double num2)
    {
        if (num1 <= 0 || num2 <= 0)
        {
            throw new ArgumentException("The arguments must be positive.");
        }
        return Math.Log(num1, num2);
    }
}

```

```

namespace IBinary
{
    public class LogOperationFactory : IBinaryOperationFactory
    {
        public IBinaryOperation CreateOperation()
        {
            return new Logarithm();
        }
    }
}

namespace IBinary
{
    public class MulOperationFactory : IBinaryOperationFactory
    {
        public IBinaryOperation CreateOperation()
        {
            return new Multiplication();
        }
    }
}

```

```

namespace IBinary
{
    public class Multiplication : IBinaryOperation
    {
        public double Calculate(double num1, double num2)
        {
            return num1 * num2;
        }
    }
}

namespace IBinary
{
    public class SubOperationFactory : IBinaryOperationFactory
    {
        public IBinaryOperation CreateOperation()
        {
            return new Subtraction();
        }
    }
}

```

```

using System;
namespace IBinary
{
    public class Subtraction : IbinaryOperation
    {
        public double Calculate(double num1, double num2)
        {
            return num1 - num2;
        }
    }
}

```

3	计算器的领域模式 2	采用工厂方法模式实现计算器 一元计算领域模型 , 实现取反、求 $1/x$ 、 $2\sqrt{x}$ 、 x^2 、 10^x 、 $ x $ 、 \log 、 \ln 、 \exp 等一元运算
---	------------	--

```

using System;
namespace IUnary
{
    public class AbsoluteOperation : IUnaryOperation
    {
        public double Calculate(double num)
        {
            return Math.Abs(num);
        }
    }
}

```

The screenshot displays two instances of Microsoft Visual Studio. Both instances have the same interface and project structure.

Project Structure:

- Solution Explorer:** Shows the solution 'SimpleCalculator' containing projects 'IBinary', 'IBinaryClient', and 'IUnary'. 'IUnary' contains files: DefaultUnaryOperationFactory.cs, IUnaryOperation.cs, AbsoluteOperation.cs, NegateOperation.cs, ReciprocalOperation.cs, SqrtOperation.cs, SquareOperation.cs, and UnaryOperationFactory.cs.
- Properties Window:** Shows file properties for 'DefaultUnaryOperationFactory.cs' and 'IUnaryOperation.cs'.
- Toolbox:** Standard .NET development tools.
- Status Bar:** Shows 'Ready' and other standard status information.

Code Editors (Left): DefaultUnaryOperationFactory.cs

```

1  using System;
2
3  namespace IUnary
4  {
5      public class DefaultUnaryOperationFactory : UnaryOperationFactory
6      {
7          public override IUnaryOperation CreateOperation(string operationName)
8          {
9              switch (operationName)
10             {
11                 case "abs":
12                     return new AbsoluteOperation();
13                 case "neg":
14                     return new NegateOperation();
15                 case "rec":
16                     return new ReciprocalOperation();
17                 case "sqrt":
18                     return new SqrtOperation();
19                 case "square":
20                     return new SquareOperation();
21                 default:
22                     throw new ArgumentException($"Unsupported operation: {operationName}");
23             }
24         }
25     }
26 }
27

```

Code Editors (Right): IUnaryOperation.cs

```

1  namespace IUnary
2  {
3      public interface IUnaryOperation
4      {
5          double Calculate(double num);
6      }
7 }
8

```

```

1  // IUnary
2  {
3      public class NegateOperation : IUnaryOperation
4      {
5          public double Calculate(double num)
6          {
7              return -num;
8          }
9      }
10 }
11

```

```

1  using System;
2
3  namespace IUnary
4  {
5      public class ReciprocalOperation : IUnaryOperation
6      {
7          public double Calculate(double num)
8          {
9              if (num == 0)
10                  throw new ArgumentException("Cannot divide by zero.");
11              return 1 / num;
12          }
13      }
14 }
15
16
17

```

```

1  using System;
2
3  namespace IUnary
4  {
5      public class SqrtOperation : IUnaryOperation
6      {
7          public double Calculate(double num)
8          {
9              if (num < 0)
10             {
11                 throw new ArgumentException("Cannot take square root of negative number.");
12             }
13             return Math.Sqrt(num);
14         }
15     }
16 }
17

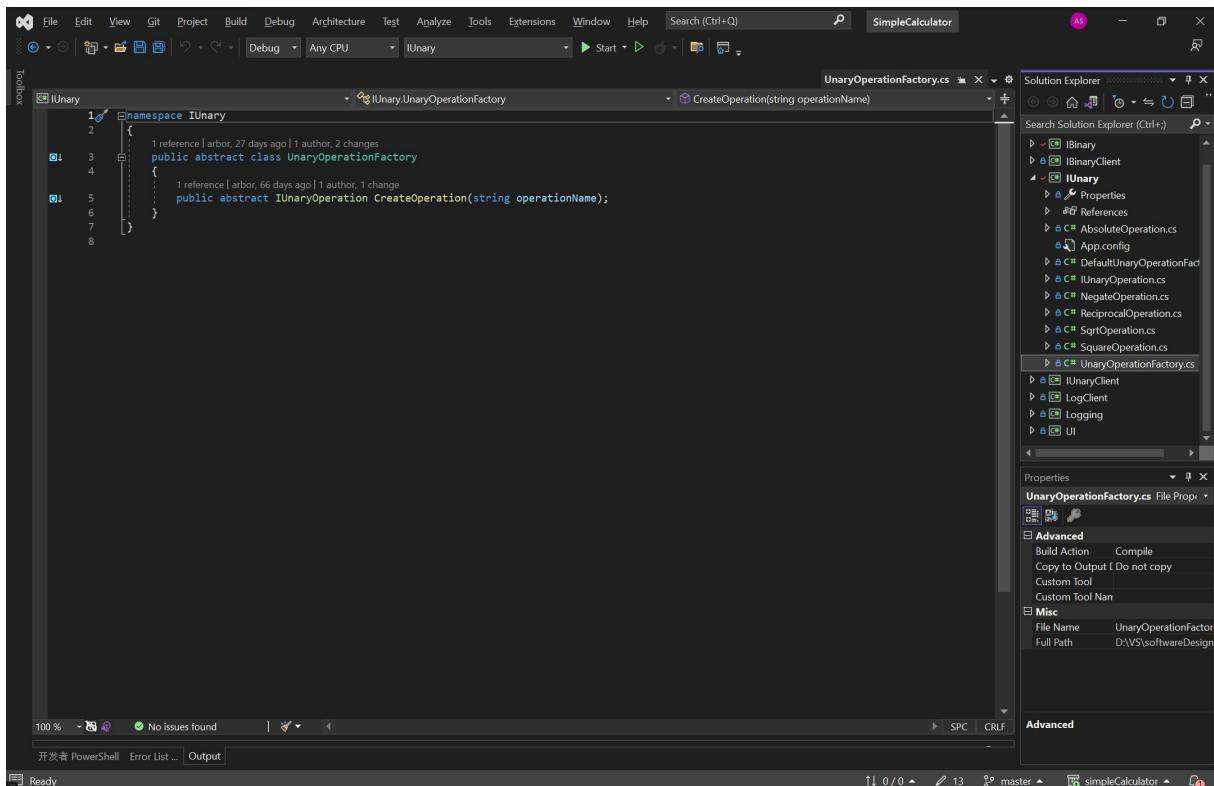
```



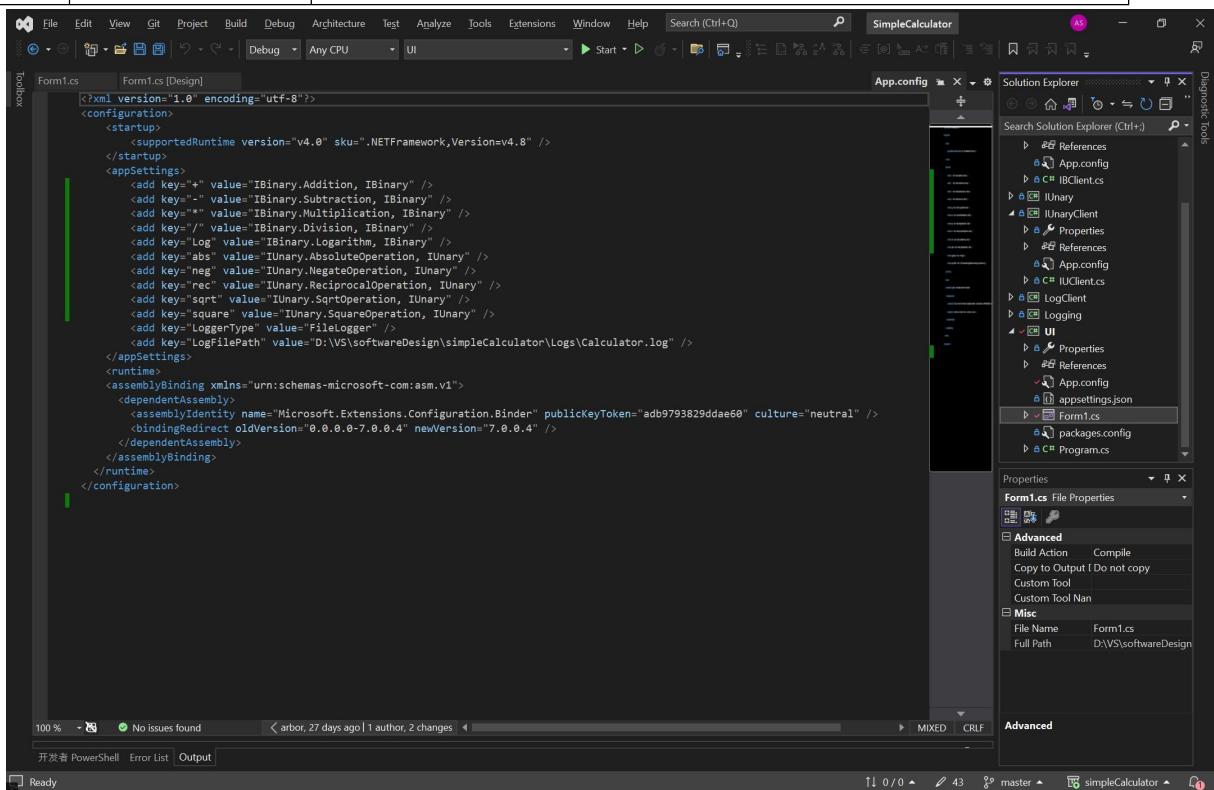
```

1  using System;
2
3  namespace IUnary
4  {
5      public class SquareOperation : IUnaryOperation
6      {
7          public double Calculate(double num)
8          {
9              return num * num;
10         }
11     }
12 }
13

```



5	计算器的 UI 界面
使用配置文件与反射机制实现二元计算工厂对象和一元计算工厂对象的实例化	



```

1  //using System;
2  //using System.IO;
3  //using System.Text.RegularExpressions;
4  //using System.Windows.Forms;
5  //using System.Configuration;
6  //using Ibinary;
7  //using Ibinary;
8  //using Logging;
9
10 //namespace UI
11 {
12     // 3 references | arbor, 27 days ago | 1 author, 4 changes
13     public partial class Form1 : Form
14     {
15         private IBinaryOperation currentOperation;
16
17         public Form1()
18         {
19             InitializeComponent();
20
21             // 1 reference | arbor, 42 days ago | 1 author, 1 change
22             private void zero_Click(object sender, EventArgs e)
23             {
24                 Button button = sender as Button;
25                 if (button != null)
26                 {
27                     formation.Text += button.Text;
28                 }
29
30             // 1 reference | arbor, 42 days ago | 1 author, 1 change
31             private void one_Click(object sender, EventArgs e)
32             {
33                 Button button = sender as Button;
34                 if (button != null)
35                 {
36                     formation.Text += button.Text;
37                 }
38
39             // 1 reference | arbor, 42 days ago | 1 author, 1 change
40             private void two_Click(object sender, EventArgs e)
41             {
42                 Button button = sender as Button;
43                 if (button != null)
44                 {
45                     formation.Text += button.Text;
46                 }
47
48             // 1 reference | arbor, 42 days ago | 1 author, 1 change
49             private void three_Click(object sender, EventArgs e)
50             {
51                 Button button = sender as Button;
52                 if (button != null)
53                 {
54                     formation.Text += button.Text;
55                 }
56
57             // 1 reference | arbor, 42 days ago | 1 author, 1 change
58             private void four_Click(object sender, EventArgs e)
59             {
60                 Button button = sender as Button;
61                 if (button != null)
62                 {
63                     formation.Text += button.Text;
64                 }
65
66             // 1 reference | arbor, 42 days ago | 1 author, 1 change
67             private void five_Click(object sender, EventArgs e)
68             {
69                 Button button = sender as Button;
70                 if (button != null)
71                 {
72                     formation.Text += button.Text;
73                 }
74
75             // 1 reference | arbor, 42 days ago | 1 author, 1 change
76             private void six_Click(object sender, EventArgs e)
77             {
78                 Button button = sender as Button;
79
80             }
81
82             // 1 reference | arbor, 42 days ago | 1 author, 1 change
83             private void seven_Click(object sender, EventArgs e)
84             {
85                 Button button = sender as Button;
86                 if (button != null)
87                 {
88                     formation.Text += button.Text;
89                 }
90
91             // 1 reference | arbor, 42 days ago | 1 author, 1 change
92             private void eight_Click(object sender, EventArgs e)
93             {
94                 Button button = sender as Button;
95                 if (button != null)
96                 {
97                     formation.Text += button.Text;
98                 }
99
100            // 1 reference | arbor, 42 days ago | 1 author, 1 change
101            private void nine_Click(object sender, EventArgs e)
102            {
103                Button button = sender as Button;
104                if (button != null)
105                {
106                    formation.Text += button.Text;
107                }
108
109            }
110
111        }
112    }
113}

```

```

1 reference | arbor, 42 days ago | 1 author, 1 change
private void seven_Click(object sender, EventArgs e)
{
    Button button = sender as Button;
    if (button != null)
    {
        formation.Text += button.Text;
    }
}

1 reference | arbor, 42 days ago | 1 author, 1 change
private void eight_Click(object sender, EventArgs e)
{
    Button button = sender as Button;
    if (button != null)
    {
        formation.Text += button.Text;
    }
}

1 reference | arbor, 42 days ago | 1 author, 1 change
private void nine_Click(object sender, EventArgs e)
{
    Button button = sender as Button;
    if (button != null)
    {
        formation.Text += button.Text;
    }
}

1 reference | arbor, 42 days ago | 1 author, 1 change
private void point_Click(object sender, EventArgs e)
{
    Button button = sender as Button;
    if (button != null)
    {
        formation.Text += button.Text;
    }
}

```



```

1 reference | arbor, 42 days ago | 1 author, 1 change
private void add_Click(object sender, EventArgs e)
{
    Button button = sender as Button;
    if (button != null)
    {
        formation.Text += button.Text;
    }
}

1 reference | arbor, 42 days ago | 1 author, 1 change
private void sub_Click(object sender, EventArgs e)
{
    Button button = sender as Button;
    if (button != null)
    {
        formation.Text += button.Text;
    }
}

1 reference | arbor, 42 days ago | 1 author, 1 change
private void mul_Click(object sender, EventArgs e)
{
    Button button = sender as Button;
    if (button != null)
    {
        formation.Text += button.Text;
    }
}

1 reference | arbor, 42 days ago | 1 author, 1 change
private void div_Click(object sender, EventArgs e)
{
    Button button = sender as Button;
    if (button != null)
    {
        formation.Text += button.Text;
    }
}

```

```

1 reference | arbor, 42 days ago | 1 author, 1 change
154 } }
155 }
156 }
157 private void log_Click(object sender, EventArgs e)
158 {
159     Button button = sender as Button;
160     if (button != null)
161     {
162         formation.Text += button.Text;
163     }
164 }
165 }

1 reference | arbor, 27 days ago | 1 author, 3 changes
166 private void equal_Click(object sender, EventArgs e)
167 {
168     // Get the expression in the text box
169     string expression = formation.Text;
170
171     // Define regular expressions to extract numbers and operators
172     string pattern = @"(\d+\.\?\d*)|([+\-\*/]|Log)|(d+\.\?\d*)";
173
174     // Use regular expressions to match expressions and extract numbers and operators
175     Match match = Regex.Match(expression, pattern);
176
177     if (match.Success)
178     {
179         // Extract the first digit, the second digit, and the operator
180         double num1 = double.Parse(match.Groups[1].Value);
181         double num2 = double.Parse(match.Groups[3].Value);
182         string op = match.Groups[2].Value;
183
184         // Create the corresponding IBinaryOperation object from the operator
185         IBinaryOperation binaryOperation = null;
186         string operationClassName = ConfigurationManager.AppSettings[op];
187         if (IsString.IsNotNullOrEmpty(operationClassName))
188         {
189             binaryOperation = (IBinaryOperation)Activator.CreateInstance(Type.GetType(operationClassName));
190         }
191
192         // If a match is found, calculate the result
193     }
194 }

195 // If a match is found, calculate the result
196 if (binaryOperation != null)
197 {
198     double result = binaryOperation.Calculate(num1, num2);
199
200     ILogger logger = LoggerFactory.CreateLogger();
201     logger.Log($"Operation {expression} = {result}");
202
203     // Write the result back into the text box
204     formation.Text = result.ToString();
205 }
206 }

207

208 1 reference | arbor, 27 days ago | 1 author, 2 changes
209 private void rec_Click(object sender, EventArgs e)
210 {
211     PerformUnaryOperation("rec");
212 }

213 1 reference | arbor, 27 days ago | 1 author, 2 changes
214 private void square_Click(object sender, EventArgs e)
215 {
216     PerformUnaryOperation("square");
217 }

218 1 reference | arbor, 27 days ago | 1 author, 2 changes
219 private void sqrt_Click(object sender, EventArgs e)
220 {
221     PerformUnaryOperation("sqrt");
222 }

223 1 reference | arbor, 27 days ago | 1 author, 2 changes
224 private void abs_Click(object sender, EventArgs e)
225 {
226     PerformUnaryOperation("abs");
227 }

228 1 reference | arbor, 27 days ago | 1 author, 2 changes
229 private void neg_Click(object sender, EventArgs e)
230 {
231     PerformUnaryOperation("neg");
232 }

```

```

229     {
230         PerformUnaryOperation("neg");
231     }
232 
233     private void PerformUnaryOperation(string operationKey)
234     {
235         double inputNum;
236         if (double.TryParse(formation.Text, out inputNum))
237         {
238             string operationClassName = ConfigurationManager.AppSettings[operationKey];
239             IUnaryOperation operation = null;
240             if (!string.IsNullOrEmpty(operationClassName))
241             {
242                 operation = (IUnaryOperation)Activator.CreateInstance(Type.GetType(operationClassName));
243             }
244             if (operation != null)
245             {
246                 double result = operation.Calculate(inputNum);
247 
248                 ILogger logger = LoggerFactory.CreateLogger();
249                 logger.LogInformation($"Operation {operationKey} {inputNum} = {result}");
250 
251                 formation.Text = result.ToString();
252             }
253             else
254             {
255                 MessageBox.Show("The operation is not defined in the configuration file.");
256             }
257         }
258     }
259 
260     1 reference | arbor, 42 days ago | 1 author, 1 change
261     private void clear_Click(object sender, EventArgs e)
262     {
263         formation.Text = "";
264     }
265 
266     1 reference | arbor, 42 days ago | 1 author, 1 change
267     private void del_Click(object sender, EventArgs e)
268     {
269         if (formation.Text.Length > 0)
270     }

```



```

266     1 reference | arbor, 42 days ago | 1 author, 1 change
267     private void del_Click(object sender, EventArgs e)
268     {
269         if (formation.Text.Length > 0)
270         {
271             formation.Text = formation.Text.Substring(0, formation.Text.Length - 1);
272         }
273     }
274 
275     1 reference | arbor, 42 days ago | 1 author, 1 change
276     private void per_Click(object sender, EventArgs e)
277     {
278         double num;
279         if (double.TryParse(formation.Text, out num))
280         {
281             num /= 100;
282             formation.Text = num.ToString();
283         }
284     }
285 
286     1 reference | arbor, 27 days ago | 1 author, 2 changes
287     private void logToolStripMenuItem_Click(object sender, EventArgs e)
288     {
289         ILogger logger = LoggerFactory.CreateLogger();
290         string logContent = GetLogContentFromLogger(logger); // Get log content from logger object
291 
292         // Display log information
293         MessageBox.Show(logContent, "Log Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
294     }
295 
296     1 reference | arbor, 27 days ago | 1 author, 2 changes
297     private string GetLogContentFromLogger	ILogger logger)
298     {
299         // Read the log file path from the configuration file
300         string filePath = ConfigurationManager.AppSettings["LogFilePath"];
301 
302         // Check if the log file exists
303         if (!File.Exists(filePath))
304         {
305             return "Log file does not exist.";
306         }
307 
308         try
309         {
310             using (var reader = new StreamReader(filePath))
311             {
312                 string content = reader.ReadToEnd();
313                 return content;
314             }
315         }
316         catch (Exception ex)
317         {
318             return ex.Message;
319         }
320     }

```

```

282     }
283     }
284     }
285     }
286     }
287     }
288     }
289     }
290     }
291     }
292     }
293     }
294     }
295     }
296     }
297     }
298     }
299     }
300     }
301     }
302     }
303     }
304     }
305     }
306     }
307     }
308     }
309     }
310     }
311     }
312     }
313     }
314     }
315     }
316     }
317     }

```

The code in Form1.cs handles log operations. It uses a logger to get log content and displays it in a message box. It also reads the log file path from the configuration file and checks if the file exists. If the file does not exist, it returns a specific message. If an exception occurs while reading the file, it returns a failure message.

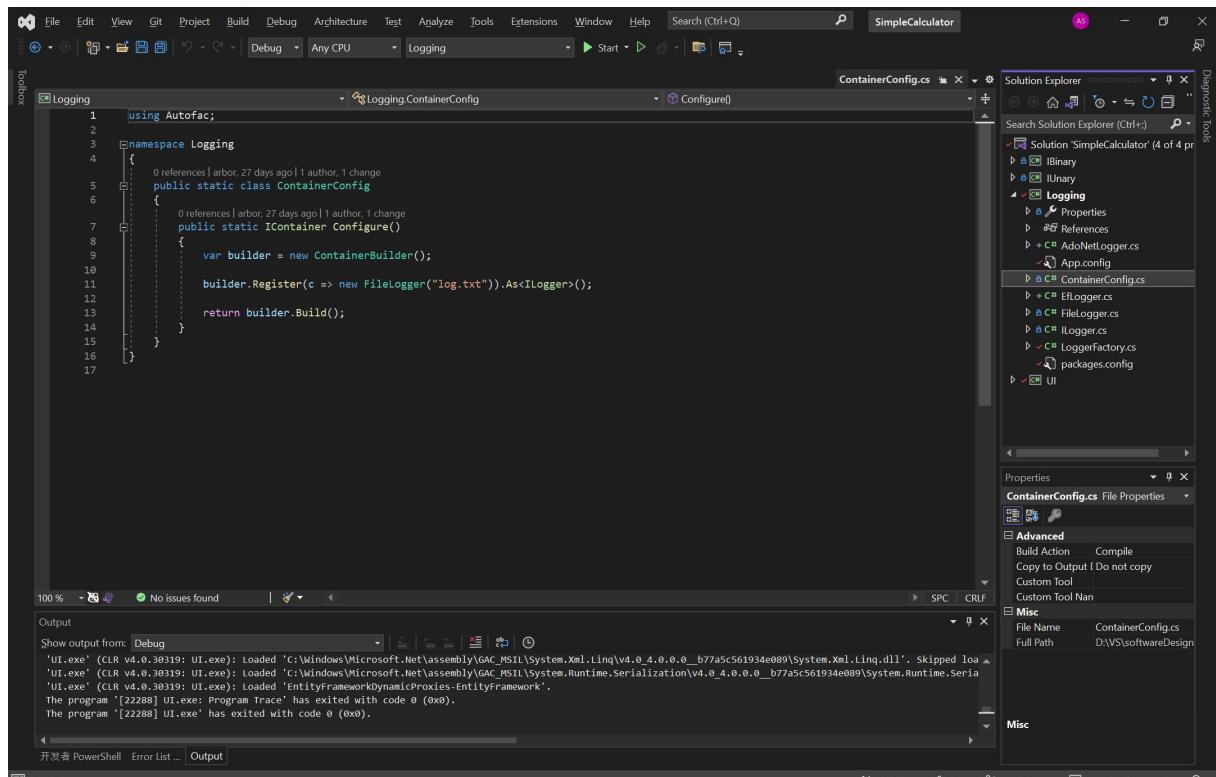
6	实现日志读写模块	将操作日志写入文本文档、使用 ADO.NET 写入数据库、使用 Simple.Data 和 EF 框架将计算器操作日志写入数据库
8	在 UI 项目中增加日志读写功能	在 UI 项目中使用日志读写模块实现日志读写功能，使用 IOC (Autofac) 实现日志读写对象的实例化

```

1  using System;
2  using System.Data;
3  using System.Data.SqlClient;
4
5  namespace Logging
6  {
7      public class AdoNetLogger : ILogger
8      {
9          private string connectionString;
10
11         public AdoNetLogger(string connectionString)
12         {
13             this.connectionString = connectionString;
14         }
15
16         public void Log(string message)
17         {
18             string query = "INSERT INTO Logs (Message, Date) VALUES (@Message, @Date)";
19
20             using (SqlConnection connection = new SqlConnection(connectionString))
21             {
22                 using (SqlCommand command = new SqlCommand(query, connection))
23                 {
24                     command.Parameters.Add("@Message", SqlDbType.NVarChar).Value = message;
25                     command.Parameters.Add("@Date", SqlDbType.DateTime).Value = DateTime.Now;
26
27                     connection.Open();
28                     command.ExecuteNonQuery();
29                 }
30             }
31         }
32     }

```

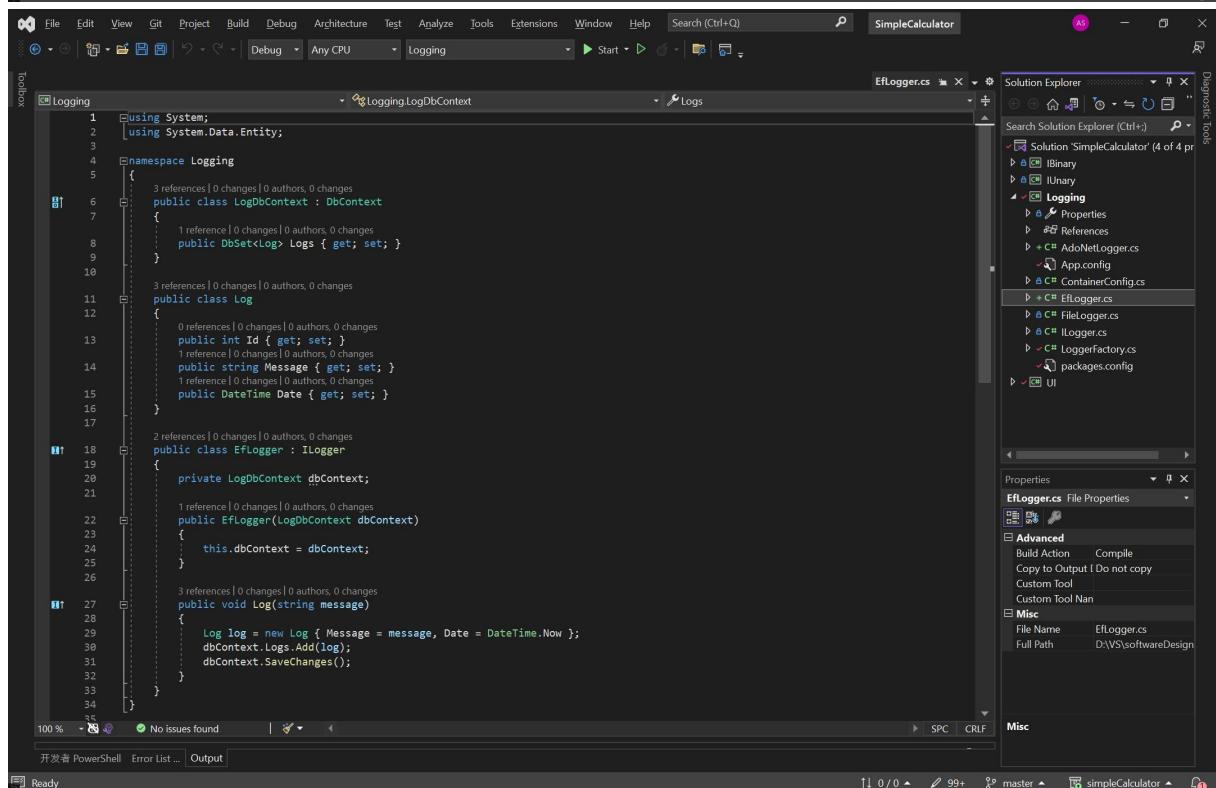
The code in AdoNetLogger.cs defines a logger that uses Entity Framework to log messages to a SQL database. It creates a connection string, opens a connection, and executes a non-query command to insert the log entry into the Logs table.



```

1  using Autofac;
2
3  namespace Logging
4  {
5      public static class ContainerConfig
6      {
7          public static.IContainer Configure()
8          {
9              var builder = new ContainerBuilder();
10
11             builder.Register(c => new FileLogger("log.txt")).As<ILogger>();
12
13             return builder.Build();
14         }
15     }
16 }
17

```

```

1  using System;
2  using System.Data.Entity;
3
4  namespace Logging
5  {
6      public class LogDbContext : DbContext
7      {
8          public DbSet<Log> Logs { get; set; }
9      }
10
11     public class Log
12     {
13         public int Id { get; set; }
14         public string Message { get; set; }
15         public DateTime Date { get; set; }
16     }
17
18     public class EflLogger : ILogger
19     {
20         private LogDbContext dbContext;
21
22         public EflLogger(LogDbContext dbContext)
23         {
24             this.dbContext = dbContext;
25         }
26
27         public void Log(string message)
28         {
29             Log log = new Log { Message = message, Date = DateTime.Now };
30             dbContext.Logs.Add(log);
31             dbContext.SaveChanges();
32         }
33     }
34 }

```

The screenshot shows two instances of the Microsoft Visual Studio IDE. Both instances have the same project structure and file content, demonstrating a comparison or a step-by-step process.

Solution Explorer:

- FileLogger.cs** (selected)
- ILogger.cs**
- AdoNetLogger.cs**
- ContainerConfig.cs**
- EfLogger.cs**
- Loggerfactory.cs**
- packages.config**
- UI**

Properties Window (FileLogger.cs):

- File Name: FileLogger.cs
- Full Path: D:\VS\SoftwareDesign
- Advanced:
 - Build Action: Compile
 - Copy to Output: Do not copy
 - Custom Tool
 - Custom Tool Nm
- Misc:
 - File Name: FileLogger.cs
 - Full Path: D:\VS\SoftwareDesign

Task List:

- No issues found

Code Editor (FileLogger.cs):

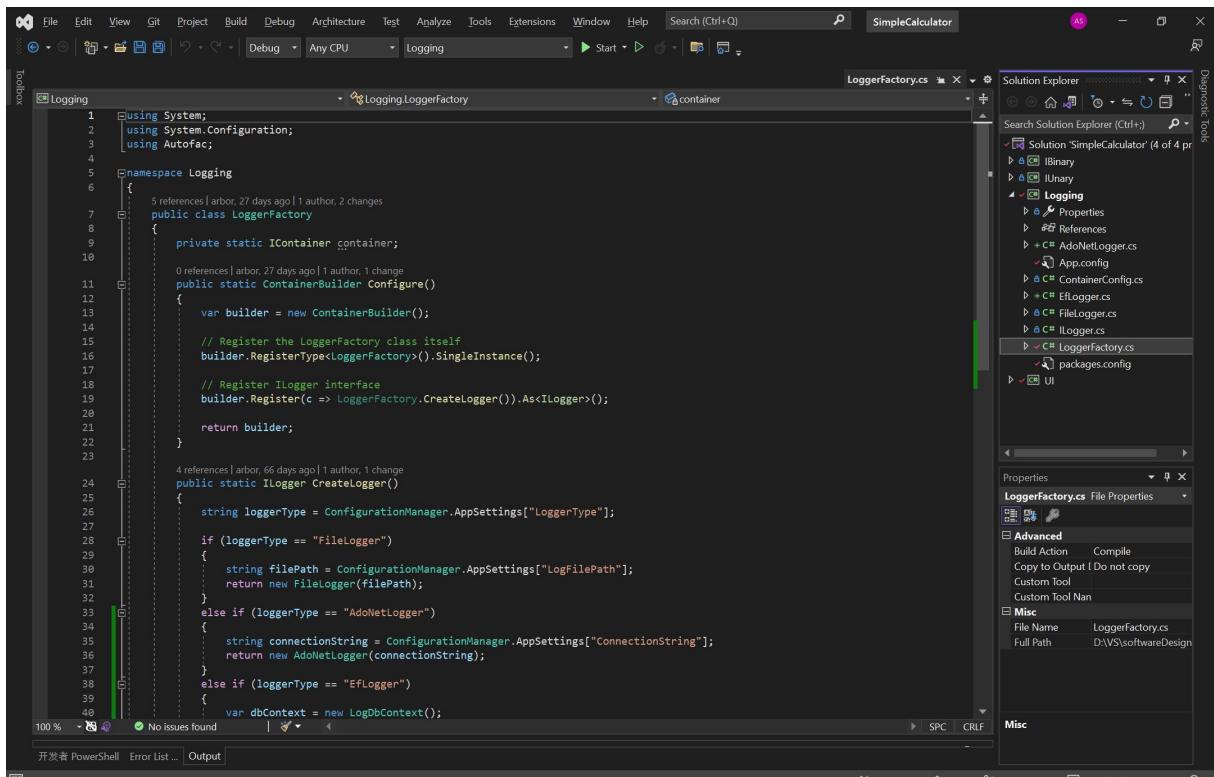
```

1  using System;
2  using System.IO;
3
4  namespace Logging
5  {
6      // 4 references | arbor, 27 days ago | 1 author, 2 changes
7      public class Filelogger : ILogger
8      {
9          private string filePath;
10         // 3 references | arbor, 66 days ago | 1 author, 1 change
11         public Filelogger(string filePath)
12         {
13             this.filePath = filePath;
14             if (!Directory.Exists(Path.GetDirectoryName(filePath)))
15             {
16                 Directory.CreateDirectory(Path.GetDirectoryName(filePath));
17             }
18         }
19         // 3 references | arbor, 66 days ago | 1 author, 1 change
20         public void Log(string message)
21         {
22             using (StreamWriter writer = File.AppendText(filePath))
23             {
24                 writer.WriteLine($"{DateTime.Now}: {message}");
25             }
26         }
27     }
28 
```

Code Editor (ILogger.cs):

```

1  namespace Logging
2  {
3      // 11 references | arbor, 27 days ago | 1 author, 2 changes
4      public interface ILogger
5      {
6          // 5 references | arbor, 66 days ago | 1 author, 1 change
7          void Log(string message);
8      }
9 
```

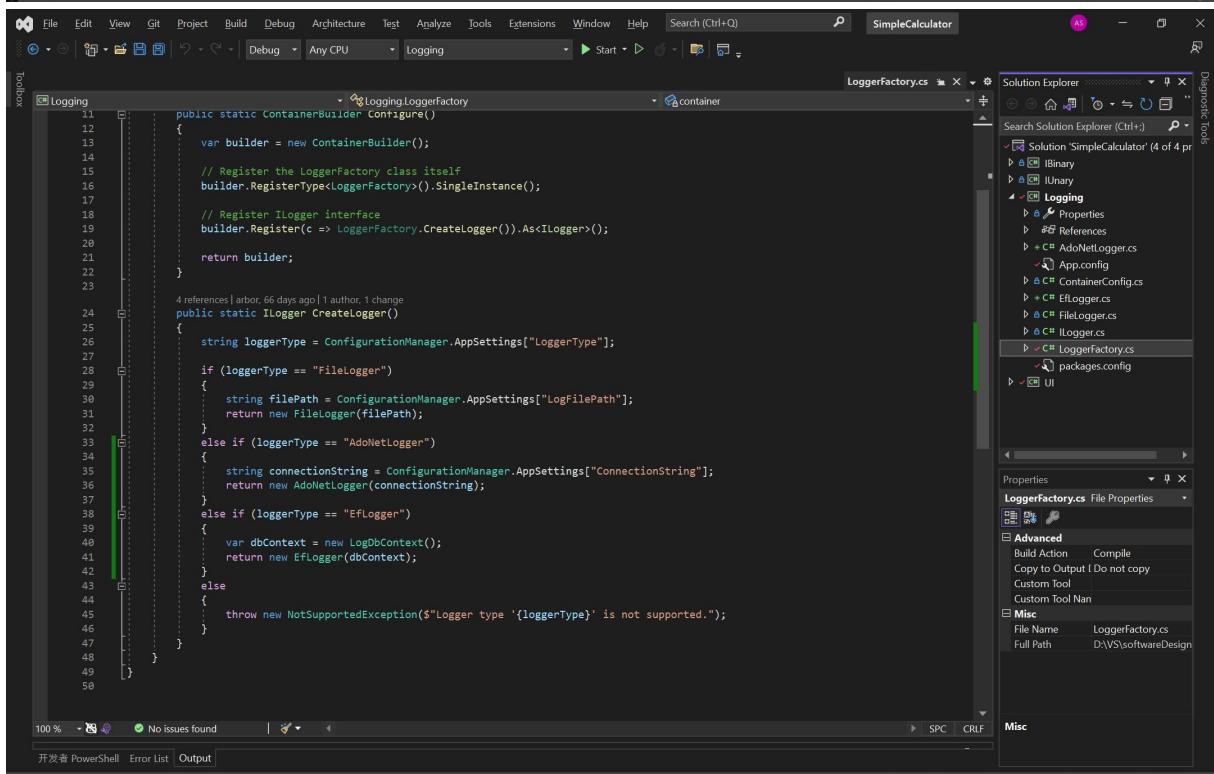


```

1  using System;
2  using System.Configuration;
3  using Autofac;
4
5  namespace Logging
6  {
7      public class LoggerFactory
8      {
9          private static IContainer container;
10
11         public static ContainerBuilder Configure()
12         {
13             var builder = new ContainerBuilder();
14
15             // Register the LoggerFactory class itself
16             builder.RegisterType<LoggerFactory>().SingleInstance();
17
18             // Register ILogger interface
19             builder.Register(c => LoggerFactory.CreateLogger()).As<ILogger>();
20
21             return builder;
22         }
23
24         public static ILogger CreateLogger()
25         {
26             string loggerType = ConfigurationManager.AppSettings["LoggerType"];
27
28             if (loggerType == "FileLogger")
29             {
30                 string filePath = ConfigurationManager.AppSettings["LogFilePath"];
31                 return new FileLogger(filePath);
32             }
33             else if (loggerType == "AdoNetLogger")
34             {
35                 string connectionString = ConfigurationManager.AppSettings["ConnectionString"];
36                 return new AdoNetLogger(connectionString);
37             }
38             else if (loggerType == "EfLogger")
39             {
40                 var dbContext = new LogDbContext();
41                 return new EfLogger(dbContext);
42             }
43             else
44             {
45                 throw new NotSupportedException($"Logger type '{loggerType}' is not supported.");
46             }
47         }
48     }
49 }

```

No issues found



```

11    public static ContainerBuilder Configure()
12    {
13        var builder = new ContainerBuilder();
14
15        // Register the LoggerFactory class itself
16        builder.RegisterType<LoggerFactory>().SingleInstance();
17
18        // Register ILogger interface
19        builder.Register(c => LoggerFactory.CreateLogger()).As<ILogger>();
20
21        return builder;
22    }
23
24    public static ILogger CreateLogger()
25    {
26        string loggerType = ConfigurationManager.AppSettings["LoggerType"];
27
28        if (loggerType == "FileLogger")
29        {
30            string filePath = ConfigurationManager.AppSettings["LogFilePath"];
31            return new FileLogger(filePath);
32        }
33        else if (loggerType == "AdoNetLogger")
34        {
35            string connectionString = ConfigurationManager.AppSettings["ConnectionString"];
36            return new AdoNetLogger(connectionString);
37        }
38        else if (loggerType == "EfLogger")
39        {
40            var dbContext = new LogDbContext();
41            return new EfLogger(dbContext);
42        }
43        else
44        {
45            throw new NotSupportedException($"Logger type '{loggerType}' is not supported.");
46        }
47    }
48 }

```

No issues found

8	在 UI 项目中增加日志读写功能	在 UI 项目中使用日志读写模块实现日志读写功能，使用 IOC (Autofac) 实现日志读写对象的实例化
---	------------------	--

```

1 //using System;
2 //using System.IO;
3 //using System.Text.Json;
4 //using System.Windows.Forms;
5 //using Autofac;
6 //using Logging;
7
8 namespace UI
9 {
10    static class Program
11    {
12        private static.IContainer container;
13
14        [STAThread]
15        static void Main()
16        {
17            // Read settings from JSON file
18            var jsonString = File.ReadAllText("appsettings.json");
19            var config = JsonSerializer.Deserialize<Config>(jsonString);
20
21            var builder = new ContainerBuilder();
22
23            if (config.Logger.Type == "FileLogger")
24            {
25                builder.Register(c => new FileLogger(config.Logger.LogFilePath)).As<ILogger>();
26            }
27            else
28            {
29                throw new NotSupportedException($"Logger type '{config.Logger.Type}' is not supported.");
30            }
31
32            container = builder.Build();
33
34            Application.EnableVisualStyles();
35            Application.SetCompatibleTextRenderingDefault(false);
36            Application.Run(new Form1());
37        }
38
39        public class Config
40        {
41            public LoggerConfig Logger { get; set; }
42
43            public class LoggerConfig
44            {
45                public string Type { get; set; }
46                public string LogFilePath { get; set; }
47            }
48        }
49    }
50

```

100 % No issues found | SPC CRLF

开发者 PowerShell Error List Output

Ready

9	MEF 实现日志读写功能	重构 UI 项目和日志读写模块项目，使用 MEF 实现日志读写功能（类似于实验 2 的第 2 部分）
---	--------------	--

The image shows two side-by-side screenshots of the Microsoft Visual Studio IDE interface. Both screenshots display the same solution structure and code editor, illustrating the implementation of two different logging mechanisms.

Solution Explorer:

- Logging** folder contains:
 - AdoNetLogger.cs** (selected)
 - EfLogger.cs**
 - FileLogger.cs**
 - ILogger.cs**
 - packages.config**
- UI** folder contains:
 - App.config**
 - CalculatorUI.cs**
 - CalculatorUI.Designer.cs**
 - CalculatorUI.resx**
 - packages.config**
 - Program.cs**
- Properties** folder contains:
 - AdoNetLogger.cs** File Properties
 - Advanced** section (Build Action: Compile, Copy to Output: Do not copy, etc.)
 - Misc** section (File Name: AdoNetLogger.cs, Full Path: D:\VS\softwareDesign)

Properties Bar:

- File Name: AdoNetLogger.cs
- Full Path: D:\VS\softwareDesign
- Advanced

Code Editor (Top Screenshot): AdoNetLogger.cs

```

1  using System;
2  using System.ComponentModel.Composition;
3  using System.Data;
4  using System.Data.SqlClient;
5
6  namespace Logging
7  {
8      [Export(typeof(ILogger))]
9      public class AdoNetLogger : ILogger
10     {
11         private string connectionString;
12
13         public AdoNetLogger(string connectionString)
14         {
15             this.connectionString = connectionString;
16         }
17
18         public void Log(string message)
19         {
20             string query = "INSERT INTO Logs (Message, Date) VALUES (@Message, @Date)";
21
22             using (SqlConnection connection = new SqlConnection(connectionString))
23                 using (SqlCommand command = new SqlCommand(query, connection))
24                 {
25                     command.Parameters.Add("@Message", SqlDbType.NVarChar).Value = message;
26                     command.Parameters.Add("@Date", SqlDbType.DateTime).Value = DateTime.Now;
27
28                     connection.Open();
29                     command.ExecuteNonQuery();
30                 }
31         }
32     }
33 }
34

```

Code Editor (Bottom Screenshot): EfLogger.cs

```

1  using System;
2  using System.ComponentModel.Composition;
3  using System.Data.Entity;
4
5  namespace Logging
6  {
7      [Export(typeof(ILogger))]
8      public class LogDbContext : DbContext
9      {
10         public DbSet<Log> Logs { get; set; }
11     }
12
13     public class Log
14     {
15         public int Id { get; set; }
16         public string Message { get; set; }
17         public DateTime Date { get; set; }
18     }
19
20     public class EfLogger : ILogger
21     {
22         private LogDbContext dbContext;
23
24         public EfLogger(LogDbContext dbContext)
25         {
26             this.dbContext = dbContext;
27         }
28
29         public void Log(string message)
30         {
31             Log log = new Log { Message = message, Date = DateTime.Now };
32             dbContext.Logs.Add(log);
33             dbContext.SaveChanges();
34         }
35     }
36 }
37

```

The screenshot shows two instances of the Visual Studio IDE side-by-side, both displaying the same software design project.

Left Window (FileLogger.cs):

- Code Editor:** Shows the implementation of the `FileLogger` class. It includes imports for `System`, `System.IO`, and `System.ComponentModel.Composition`. The class is annotated with `[Export(typeof ILogger))]`. It has a private field `filePath` and a constructor that initializes it. A checkmark indicates that the file path exists or is created if it doesn't. The `Log` method uses a `StreamWriter` to append the current date and time followed by the message to the file.
- Solution Explorer:** Shows the project structure under the `Logging` folder. It includes `FileLogger.cs`, `ILogger.cs`, and other files like `AdoNetLogger.cs`, `EfLogger.cs`, and `CalculatorUI.cs`.
- Properties:** Shows the properties for `FileLogger.cs`, including build action as `Compile` and file name as `FileLogger.cs`.
- Task List:** Shows a single entry: "No issues found".

Right Window (ILogger.cs):

- Code Editor:** Shows the interface definition for `ILogger`. It includes imports for `System.ComponentModel.Composition`. The interface has a single method `Log(string message)`.
- Solution Explorer:** Shows the project structure under the `Logging` folder. It includes `ILogger.cs`, `FileLogger.cs`, `AdoNetLogger.cs`, `EfLogger.cs`, and `CalculatorUI.cs`.
- Properties:** Shows the properties for `ILogger.cs`, including build action as `Compile` and file name as `ILogger.cs`.
- Task List:** Shows a single entry: "No issues found".

```

using System;
using System.Windows.Forms;

namespace UI
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new CalulatorUI());
        }
    }
}

using System;
using System.IO;
using System.Text.RegularExpressions;
using System.Windows.Forms;
using System.Configuration;
using Binary;
using Ibinary;
using Logging;
using CalculatorThemes;
using System.Collections.Generic;
using System.ComponentModel.Composition.Hosting;
using System.ComponentModel.Composition;
using System.ComponentModel;

namespace UI
{
    public partial class CalulatorUI : Form
    {
        [ImportMany]
        public Ienumerable<ILogger> Loggers { get; set; }

        public CalulatorUI()
        {
            InitializeComponent();
            _themeFactory = new LightThemeFactory();

            // Configure the MEF container
            try
            {
                var catalog = new DirectoryCatalog(AppDomain.CurrentDomain.BaseDirectory);
                var container = new CompositionContainer(catalog);
                container.ComposeParts(this);
            }
            catch (CompositionException ex)
            {
                Console.WriteLine("CompositionException: " + ex.ToString());
            }
        }

        private ILogger GetLogger()
        {
        }
    }
}

```

The screenshot shows two instances of the Microsoft Visual Studio IDE. Both windows have the title bar "SimpleCalculator" and the menu bar "File", "Edit", "View", "Git", "Project", "Build", "Debug", "Architecture", "Test", "Analyze", "Tools", "Extensions", "Window", "Help".

Top Window:

```

    39
    40     private ILogger GetLogger()
    41     {
    42         string loggerType = ConfigurationManager.AppSettings["LoggerType"];
    43
    44         if (loggerType == "FileLogger")
    45         {
    46             string filePath = ConfigurationManager.AppSettings["LogFilePath"];
    47             return new FileLogger(filePath);
    48         }
    49         else if (loggerType == "AdoNetLogger")
    50         {
    51             string connectionString = ConfigurationManager.AppSettings["ConnectionString"];
    52             return new AdoNetLogger(connectionString);
    53         }
    54         else if (loggerType == "EfLogger")
    55         {
    56             var dbContext = new LogDbContext();
    57             return new EfLogger(dbContext);
    58         }
    59         else
    60         {
    61             throw new Exception("Invalid logger type specified in configuration.");
    62         }
    63     }
    64
    65     private void zero_Click(object sender, EventArgs e)
    66     {
    67         Button button = sender as Button;
    68         if (button != null)
    69         {
    70             formation.Text += button.Text;
    71         }
    72     }
    73
    74     private void one_Click(object sender, EventArgs e)
    75     {
    76         Button button = sender as Button;
    77         if (button != null)
    78         {
    79             formation.Text += button.Text;
    80         }
    81     }
  
```

Bottom Window:

```

    213
    214     // Get the expression in the text box
    215     string expression = formation.Text;
    216
    217     // Define regular expressions to extract numbers and operators
    218     string pattern = @"(\d+\.?\d*)[+\-*/\|\Log](\d+\.?\d*)";
    219
    220     // Use regular expressions to match expressions and extract numbers and operators
    221     Match match = Regex.Match(expression, pattern);
    222
    223     if (match.Success)
    224     {
    225         // Extract the first digit, the second digit, and the operator
    226         double num1 = double.Parse(match.Groups[1].Value);
    227         double num2 = double.Parse(match.Groups[3].Value);
    228         string op = match.Groups[2].Value;
    229
    230         // Create the corresponding IBinaryOperation object from the operator
    231         IBinaryOperation binaryOperation = null;
    232         string operationClassName = ConfigurationManager.AppSettings[op];
    233         if (!string.IsNullOrEmpty(operationClassName))
    234         {
    235             binaryOperation = (IBinaryOperation)Activator.CreateInstance(Type.GetType(operationClassName));
    236         }
    237
    238         // If a match is found, calculate the result
    239         if (binaryOperation != null)
    240         {
    241             double result = binaryOperation.Calculate(num1, num2);
    242             ILogger logger = GetLogger();
    243             logger.Log($"Operation {expression} = {result}");
    244
    245             // Write the result back into the text box
    246             formation.Text = result.ToString();
    247         }
    248
    249     }
    250
    251
    252     private void rec_Click(object sender, EventArgs e)
    253     {
    254         PerformUnaryOperation("rec");
  
```

A red arrow points from the line `logger.Log(\$"Operation {expression} = {result}");` in the bottom window to the line `logger.Log(Log) in the top window.

```

100 % No issues found | SPC CRLF
    
```

```

private void PerformUnaryOperation(string operationKey)
{
    double inputNum;
    if (double.TryParse(formation.Text, out inputNum))
    {
        string operationClassName = ConfigurationManager.AppSettings[operationKey];
        IUnaryOperation operation = null;
        if (!string.IsNullOrEmpty(operationClassName))
        {
            operation = (IUnaryOperation)Activator.CreateInstance(Type.GetType(operationClassName));
        }
        if (operation != null)
        {
            double result = operation.Calculate(inputNum);

            ILogger logger = GetLogger();
            logger.Log($"Operation {operationKey} {inputNum} = {result}");

            formation.Text = result.ToString();
        }
        else
        {
            MessageBox.Show("The operation is not defined in the configuration file.");
        }
    }
}

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

private void del_Click(object sender, EventArgs e)
{
    if (formation.Text.Length > 0)
    {
        formation.Text = formation.Text.Substring(0, formation.Text.Length - 1);
    }
}
    
```



```

private void logToolStripMenuItem_Click(object sender, EventArgs e)
{
    ILogger logger = GetLogger();
    string logContent = GetLogContentFromLogger(logger); // Get log content from logger object

    // Display log information
    MessageBox.Show(logContent, "Log Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

private string GetLogContentFromLogger(ILogger logger)
{
    // Read the log file path from the configuration file
    string logFilePath = ConfigurationManager.AppSettings["LogFilepath"];

    // Check if the log file exists
    if (!File.Exists(logFilePath))
    {
        return "Log file does not exist.";
    }

    try
    {
        // Read the contents of the log file
        string logContent = File.ReadAllText(logFilePath);

        return logContent;
    }
    catch (IOException ex)
    {
        // Handling read file exceptions
        return $"Failed to read log file: {ex.Message}";
    }
}

private IThemeFactory _themeFactory;
private void themeToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (_themeFactory is LightThemeFactory)
    {
        _themeFactory = new DarkThemeFactory();
    }
}
    
```

10	实现计算器主题设置和切换	<p>采用抽象工厂模式设计计算器主题功能，定义至少两种主题，每种主题下，计算器的特性是不同的，比如背景图，字体颜色，字体等不相同。</p>
----	---------------------	--

```

1  using System.Drawing;
2
3  namespace CalculatorThemes
4  {
5      public class LightThemeFactory : IThemeFactory
6      {
7          public Color GetBackgroundColor()
8          {
9              return Color.White;
10         }
11
12         public Font GetFont()
13         {
14             return new Font("SimSun", 9);
15         }
16
17         public Color GetFontColor()
18         {
19             return Color.Black;
20         }
21
22
23         public class DarkThemeFactory : IThemeFactory
24         {
25             public Color GetBackgroundColor()
26             {
27                 return Color.Black;
28             }
29
30             public Font GetFont()
31             {
32                 return new Font("Verdana", 9);
33             }
34
35             public Color GetFontColor()
36         }
37
38
39
40
41
42

```

CalculatorThemes.cs

```

1  using System.Drawing;
2
3  namespace CalculatorThemes
4  {
5      public interface IThemeFactory
6      {
7          Color GetBackgroundColor();
8          Font GetFont();
9          Color GetFontColor();
10     }
11 }
12

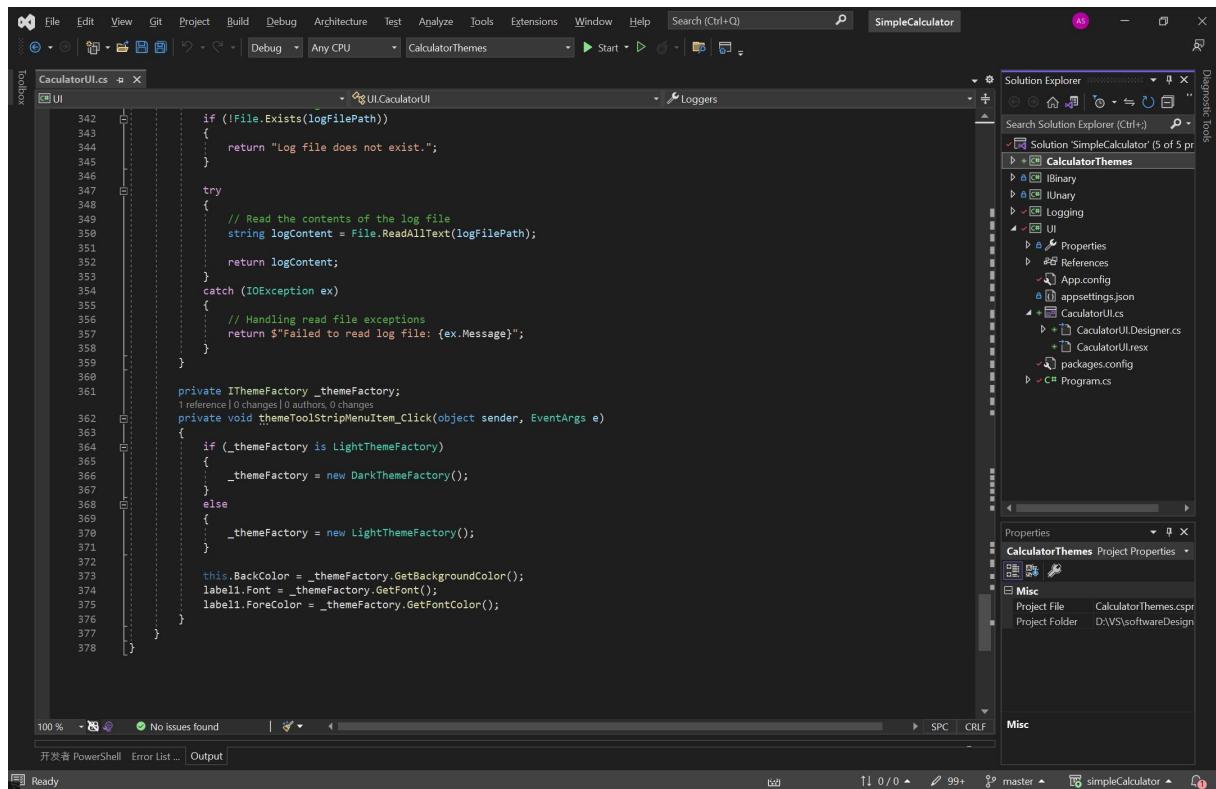
```

CalculatorUI.cs

```

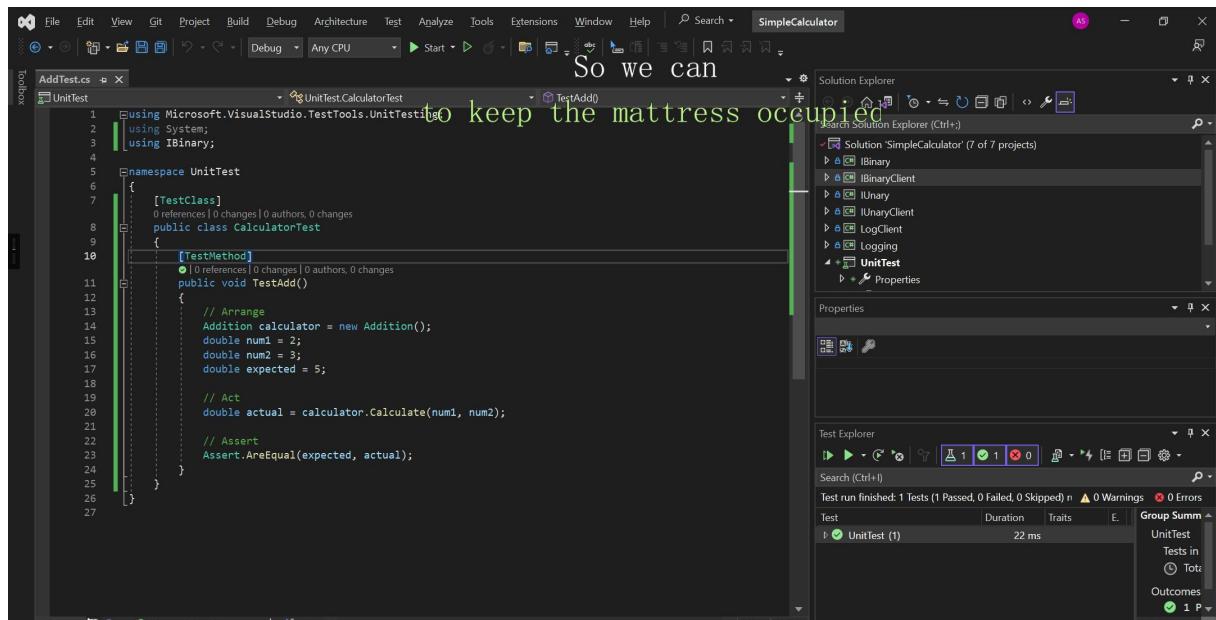
1  using System;
2  using System.IO;
3  using System.Text.RegularExpressions;
4  using System.Windows.Forms;
5  using System.Configuration;
6  using Binary;
7  using Unary;
8  using Logging;
9  using CalculatorThemes;
10 using System.Collections.Generic;
11 using System.ComponentModel.Composition.Hosting;
12 using System.ComponentModel.Composition;
13
14 namespace UI
15 {
16     public partial class CalculatorUI : Form
17     {
18         [ImportMany]
19         public IEnumerable<ILogger> Loggers { get; set; }
20
21         public CalculatorUI()
22         {
23             InitializeComponent();
24             _themeFactory = new LightThemeFactory();
25
26             // Configure the MEF container
27             try
28             {
29                 var catalog = new DirectoryCatalog(AppDomain.CurrentDomain.BaseDirectory);
30                 var container = new CompositionContainer(catalog);
31                 container.ComposeParts(this);
32             }
33             catch (CompositionException ex)
34             {
35                 Console.WriteLine("CompositionException: " + ex.ToString());
36             }
37         }
38
39         private ILogger GetLogger()
40     }
41

```



四、运行效果

2	对于领域模型 1 的单元测试	对领域模型 1 进行单元测试
---	----------------	----------------



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

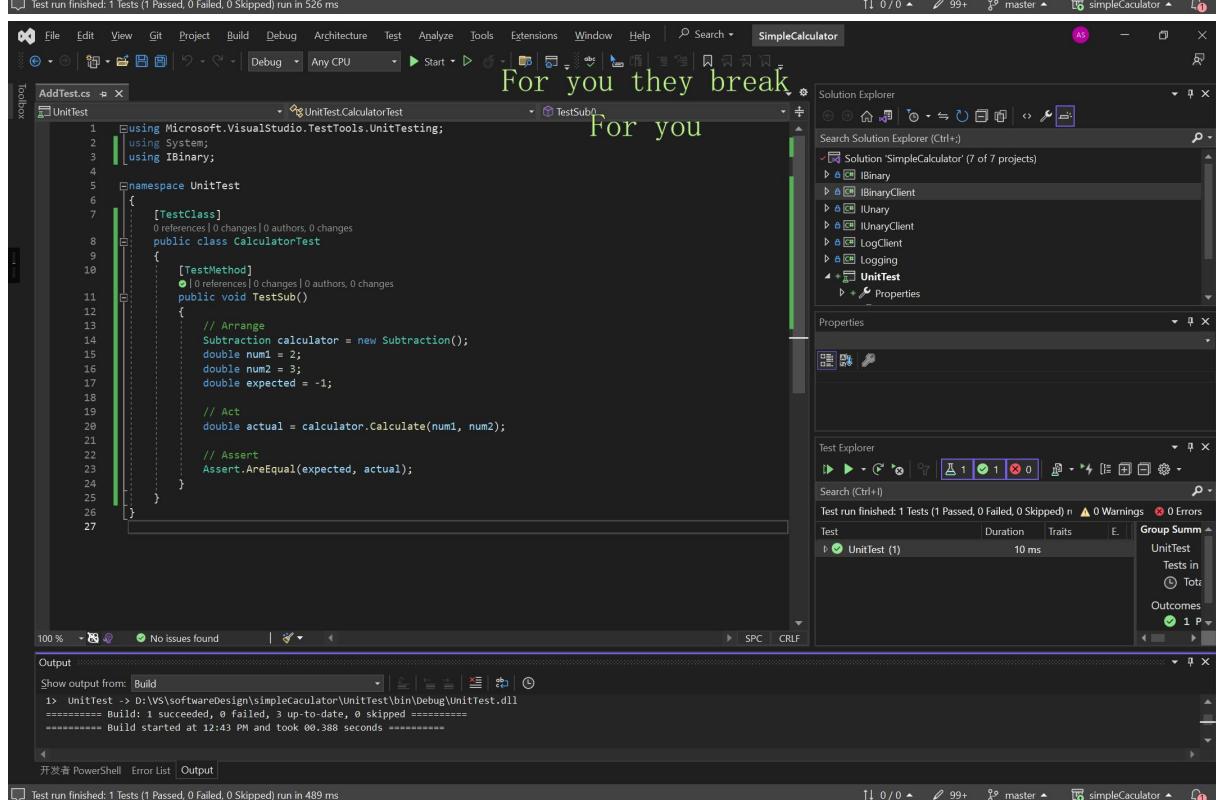
```

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

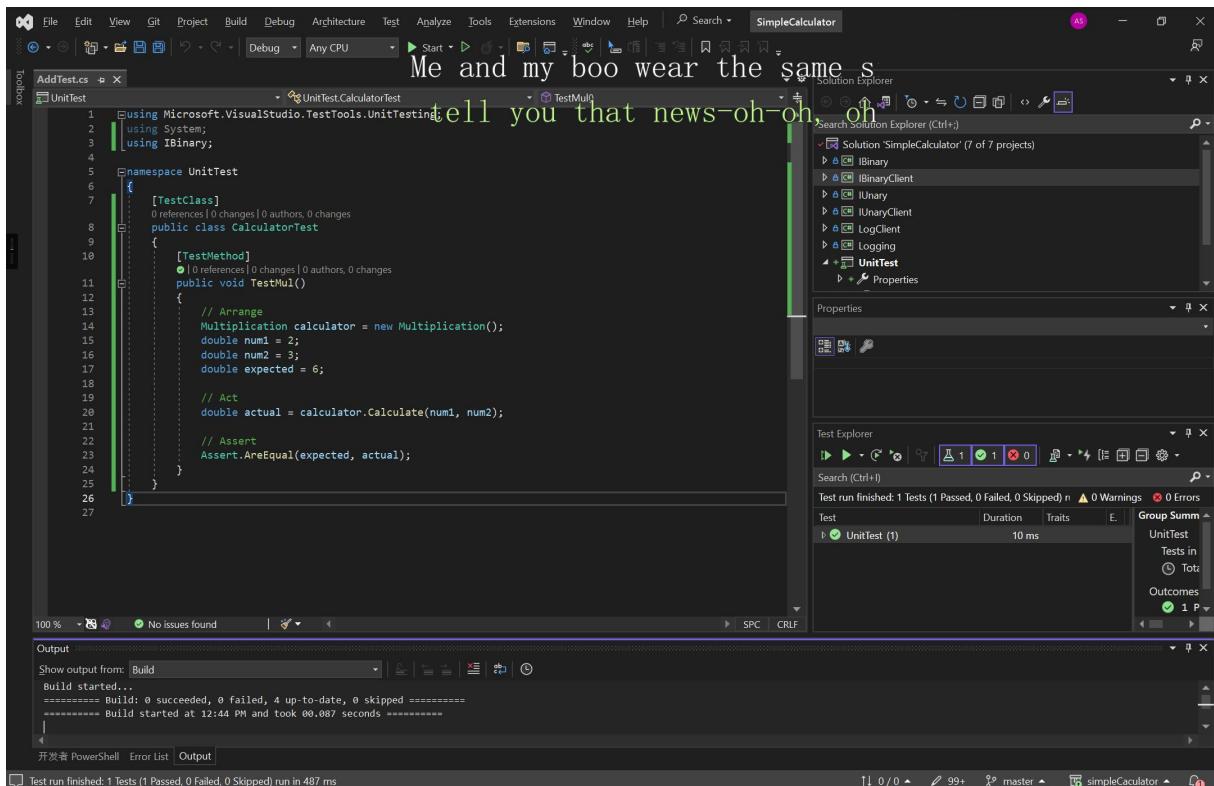
```

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. The top instance displays a unit test for division, and the bottom instance displays a unit test for logarithms. Both tests pass successfully.

```

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

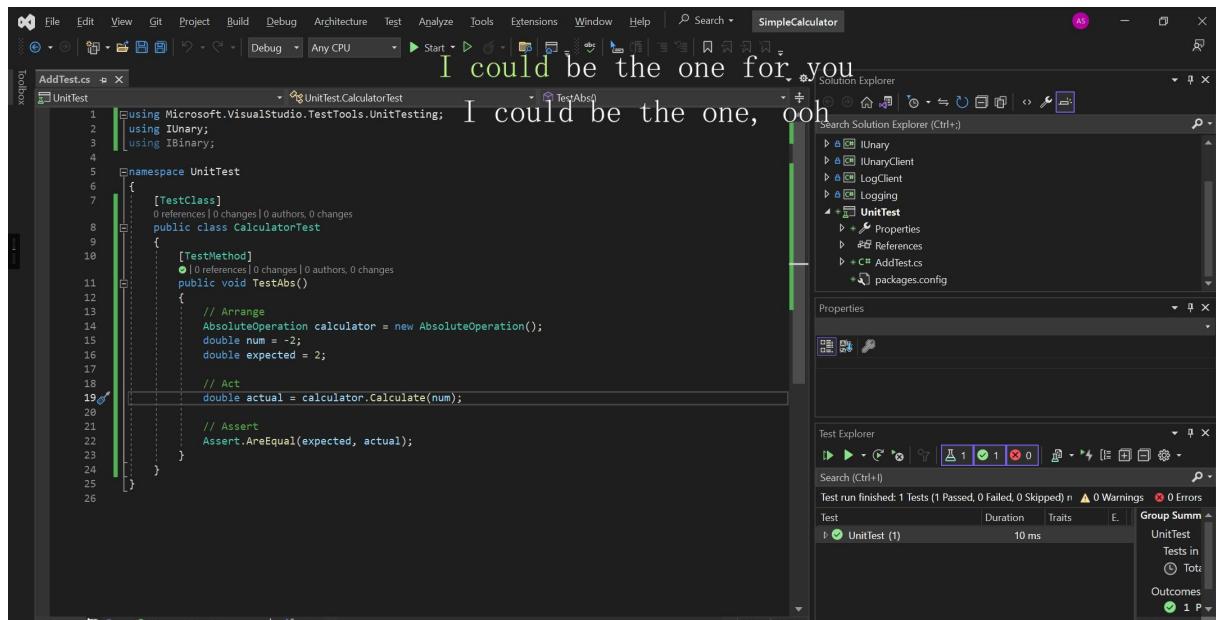
```

```

Bottom Instance (Logarithm Test):
AddTest.cs
1 using Microsoft.VisualStudio.TestTools.UnitTesting;
2 using System;
3 using IBinary;
4
5 namespace UnitTest
6 {
7     [TestClass]
8     public class CalculatorTest
9     {
10         [TestMethod]
11         public void TestLog()
12         {
13             // Arrange
14             Logarithm calculator = new Logarithm();
15             double num1 = 2;
16             double num2 = 4;
17             double expected = 0.5;
18
19             // Act
20             double actual = calculator.Calculate(num1, num2);
21
22             // Assert
23             Assert.AreEqual(expected, actual);
24         }
25     }
26 }

```

4	对于领域模型 2 的 单元测试	对领域模型 2 进行单元测试
---	--------------------	----------------



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

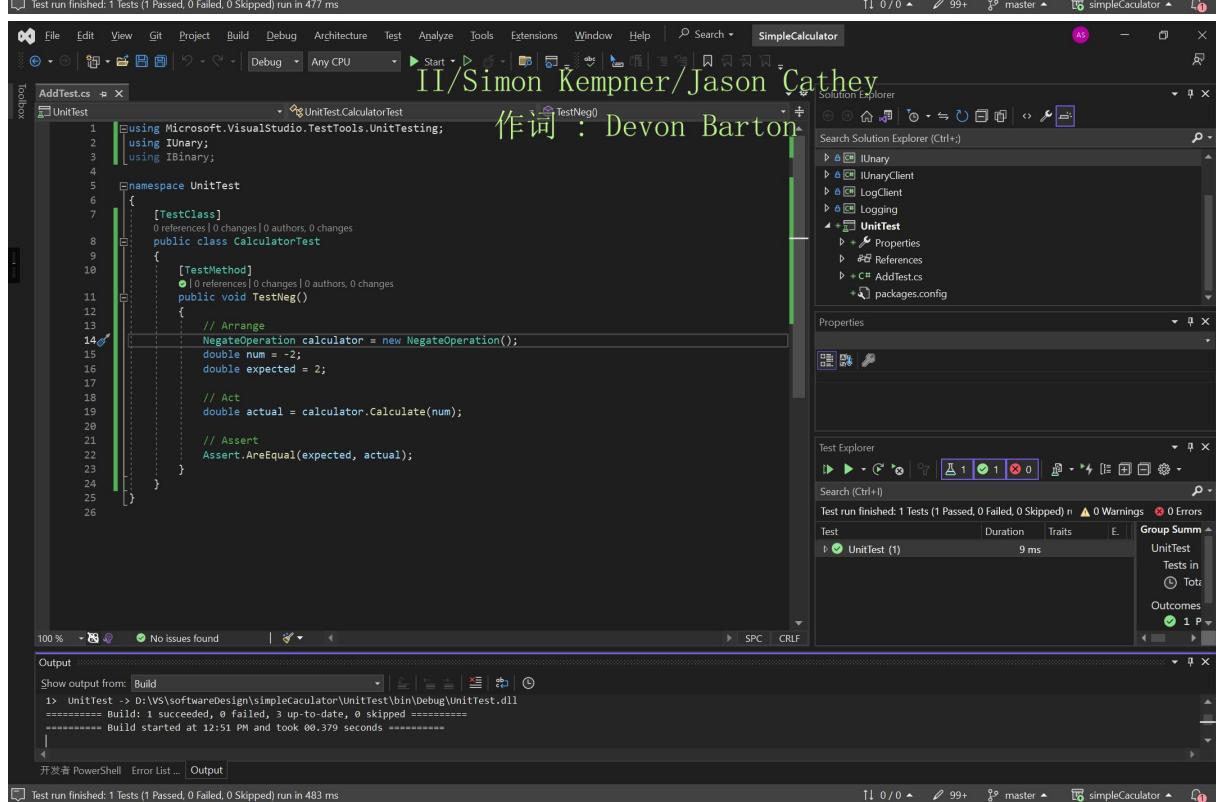
```

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
26                     [TestMethod]
27                         public void TestSqr()
28                         {
29                             // Arrange
30                             SqrtOperation calculator = new SqrtOperation();
31                             double num = 4;
32                             double expected = 2;
33
34                             // Act
35                             double actual = calculator.Calculate(num);
36
37                             // Assert
38                             Assert.AreEqual(expected, actual);
39
40                     }
41
42                     [TestMethod]
43                         public void TestSquare()
44                         {
45                             // Arrange
46                             SquareOperation calculator = new SquareOperation();
47                             double num = -2;
48                             double expected = 4;
49
50                             // Act
51                             double actual = calculator.Calculate(num);
52
53                         }
    
```

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

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

Output

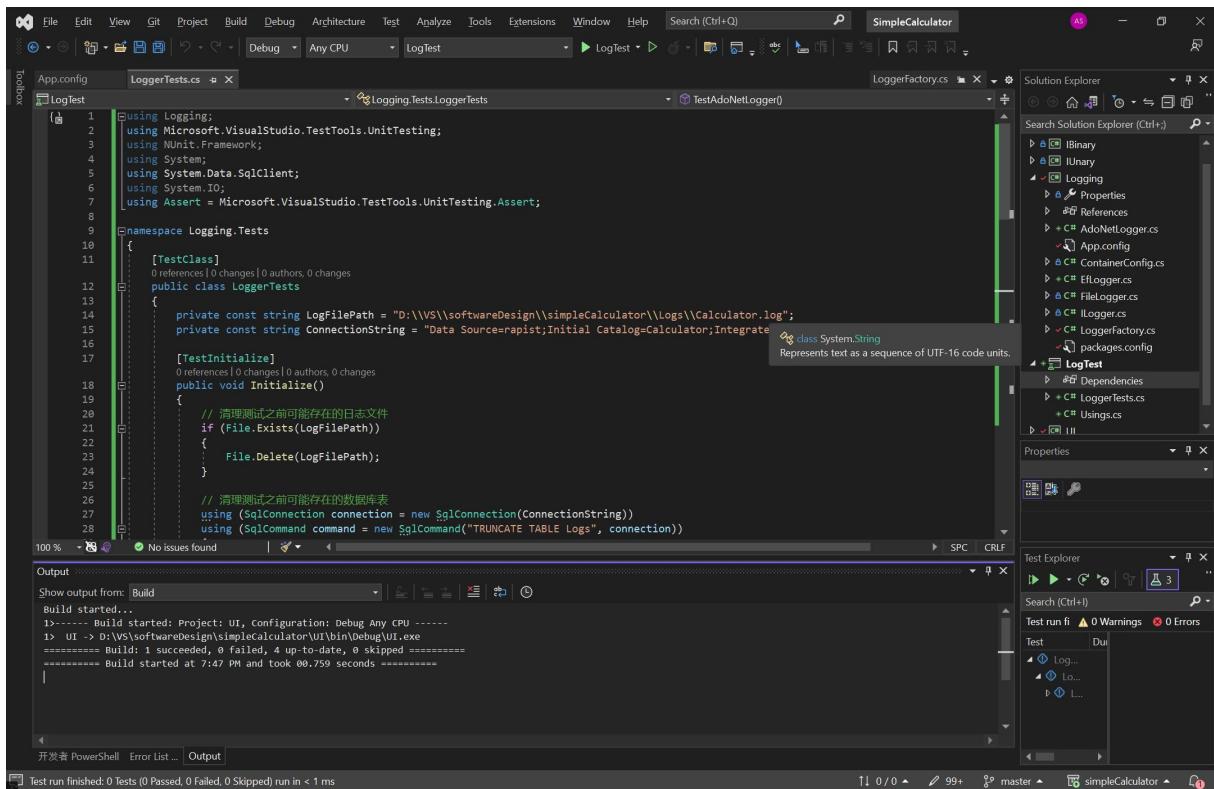
```

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

7	数据库日志读写 模块单元测试	对于日志读写的多种不同的实现，对其进行单元测试
---	-------------------	-------------------------



五、课设心得

在完成这个课设的过程中，我获得了许多宝贵的经验和心得。这个课设要求我设计并实现了一个计算器应用程序，其中包括基本的四则运算以及日志记录和主题设置功能。以下是我对这次课设的总结心得：

首先，通过这个课设，我深入理解了面向对象设计的重要性和实践。通过将功能模块划分为不同的类和接口，我能够更好地组织和管理代码。我使用了抽象工厂模式和依赖注入来实现灵活的日志记录和主题设置功能。这种面向对象的设计方法使得代码更加模块化、可扩展和可维护。

其次，我学会了使用常见的设计模式来解决实际问题。在这个课设中，我使用了抽象工厂模式来实现主题设置功能，这样可以轻松地切换和扩展不同的主题。我还使用了依赖注入来解耦具体的日志记录器实现和使用它的类。这些设计模式帮助我更好地组织代码，并提供了灵活性和可扩展性。

第三，我体验到了测试驱动开发（TDD）的价值。在实现计算器的各个功能模块时，我首先编写了单元测试，然后逐步实现功能以使测试通过。这种迭代的开发方式使得代码更加可靠和健壮，并减少了出错的可能性。通过编写单元测试，我能够更好地理解和验证代码的行为，确保每个模块都按预期工作。

最后，我学会了合理规划项目的结构和组织代码。通过将不同的功能模块分成不同的命名空间和文件夹，我可以更好地管理和维护代码。同时，使用适当的命名和注释可以提高代码的可读性和可理解性。这种良好的项目结构和代码组织可以减少开发过程中的混乱，并帮助其他开发人员更轻松地理解和参与项目。

总的来说，这个课设是一个很好的实践机会，让我将所学的知识应用到一个具体的项目中。通过设计和实现一个完整的计算器应用程序，我不仅加深了对面向对象设计原则和设计模式的理解，还提高了自己的编码能力和解决问题的能力。我相信这些经验和技能将对我未来的软件开发工作产生积极的影响。