

Pr. 6 modularization

2017.5

回忆函数封装

```
C:\Users\hzs\Desktop\new.cpp - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T) 宏(M) 运行(R) 插件(P) 窗口(W) ?
宏(M) 运行(R) 插件(P) 窗口(W) ?

1 void SmartCalculator::Calculate()
2 {
3     double res = 0.0f;
4     double f = this->getFirst();
5     double s = this->getSecond();
6
7     if (ADD == m_eType)
8         res = f + s;
9     else if (MINUS == m_eType)
10        res = f - s;
11
12    out(res);
13 }
```

```
C:\Users\hzs\Desktop\new.cpp - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T) 宏(M) 运行(R) 插件(P) 窗口(W) ?
窗口(W) ?

15 double SmartCalculator::getFirst()
16 {
17     CString first = "";
18     GetDlgItem(IDC_EDIT_FIRST)->GetWindowText(first);
19     if (first.IsEmpty())
20         Alarm("没有输入第 1 个操作数");
21     return atof(first.GetBuffer());
22 }
```

```
C:\Users\hzs\Desktop\new.cpp - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T) 宏(M) 运行(R) 插件(P) 窗口(W) ?
窗口(W) ?

24 double SmartCalculator::getSecond()
25 {
26     CString second = "";
27     GetDlgItem(IDC_EDIT_SECOND)->GetWindowText(second);
28     if (second.IsEmpty())
29         Alarm("没有输入第 2 个操作数");
30     return atof(second.GetBuffer());
31 }
```

```
C:\Users\hzs\Desktop\new.cpp - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T) 宏(M) 运行(R) 插件(P) 窗口(W) ?
窗口(W) ?

32 void out(double res)
33 {
34     CString strRes;
35     strRes.Format("%lf", res);
36     GetDlgItem(IDC_STATIC_RESULT)->SetWindowText(strRes);
37 }
```


回忆多态设计

```
Calculator.java  Operation.java  Add.java  Minus.java
1
2
3 public class Calculator
4 {
5     static Operation op = null;
6
7     public static void main(String[] args)
8     {
9         double f = 100.0f;
10        double s = 20.0f;
11
12        double value = op.calculate(f, s);
13
14        System.out.println(value);
15    }
16
17    static void ButtonDownAdd()
18    {
19        op = new Add();
20    }
21
22    static void ButtonDownMinus()
23    {
24        op = new Minus();
25    }
26 }
```

```
Calculator.java  Operation.java  Add.java  Minus.java
1 public interface Operation
2 {
3     public double calculate(double f, double s);
4 }
```

```
Calculator.java  Operation.java  Add.java  Minus.java
1 public class Add implements Operation
2 {
3     public Add(){}
4     public double calculate(double f, double s)
5     {
6         return f + s;
7     }
8 }
```

```
Calculator.java  Operation.java  Add.java  Minus.java
1 public class Minus implements Operation
2 {
3     public Minus(){}
4     public double calculate(double f, double s)
5     {
6         return f - s;
7     }
8 }
```

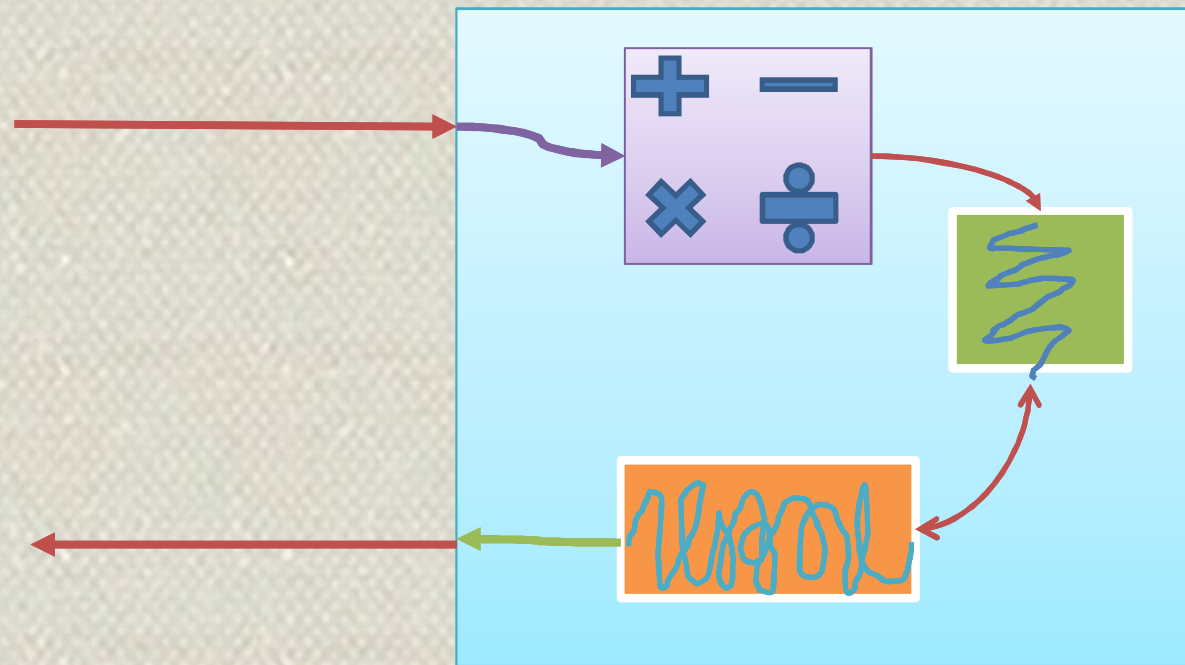
表达式求值模块化

- 要什么样的调用接口比较好？



好的模块

- 低耦合
- 高内聚



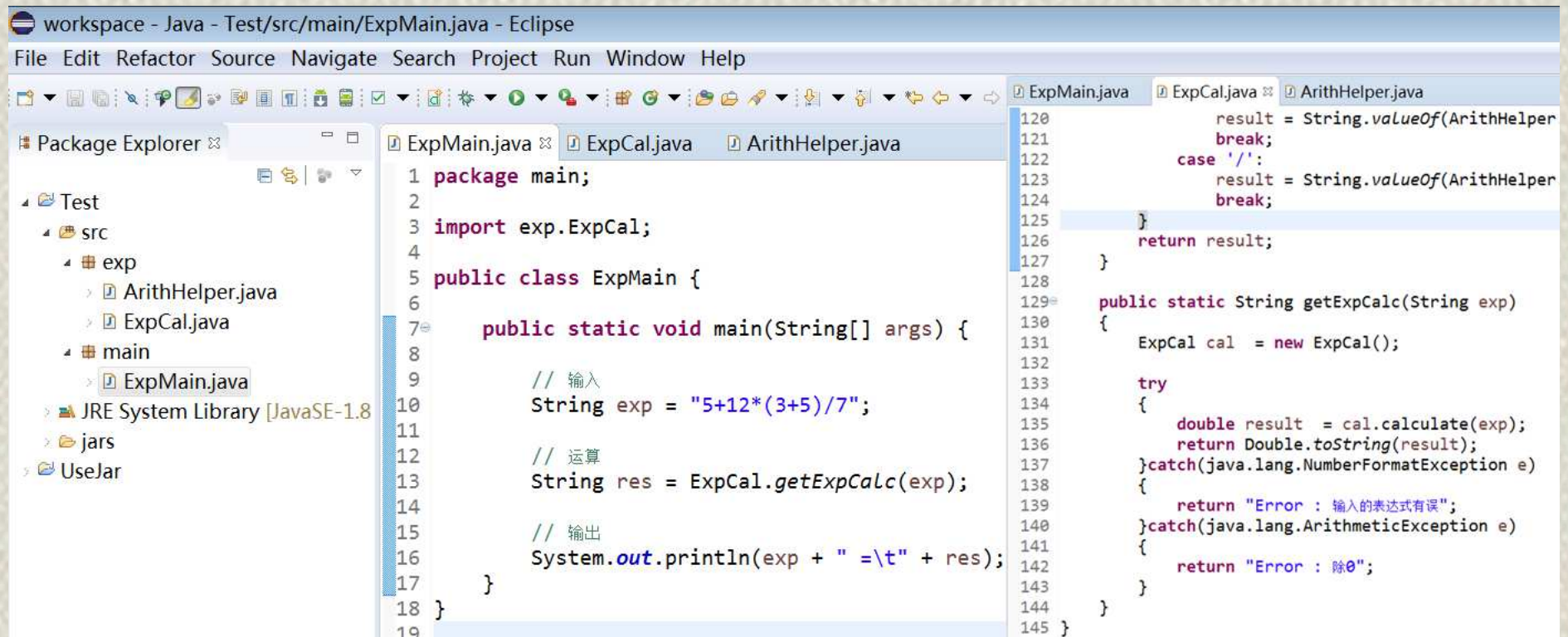
模块化的一个方案——库

- Jar
- Lib
- Dll

为什么要使用库

- 会使项目看起来更简捷
- 库无处不在
- 你要自己写完所有代码，包括一个 `<iostream>` ？
- 别人的源码被不小心改掉了
- 更有利于分工协作

Java方案



workspace - Java - Test/src/main/ExpMain.java - Eclipse

File Edit Refactor Source Navigate Search Project Run Window Help

Package Explorer

- Test
 - src
 - exp
 - ArithHelper.java
 - ExpCal.java
 - main
 - ExpMain.java
 - JRE System Library [JavaSE-1.8]
 - jars
 - UseJar

ExpMain.java

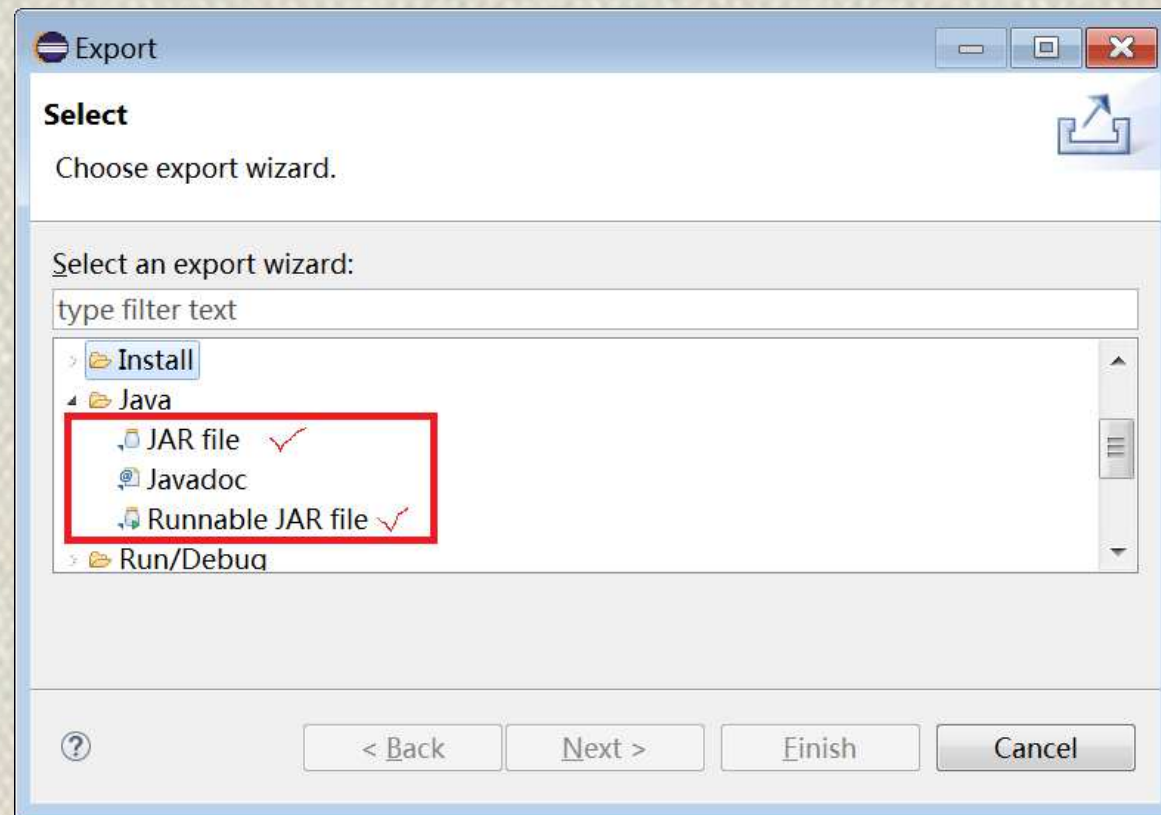
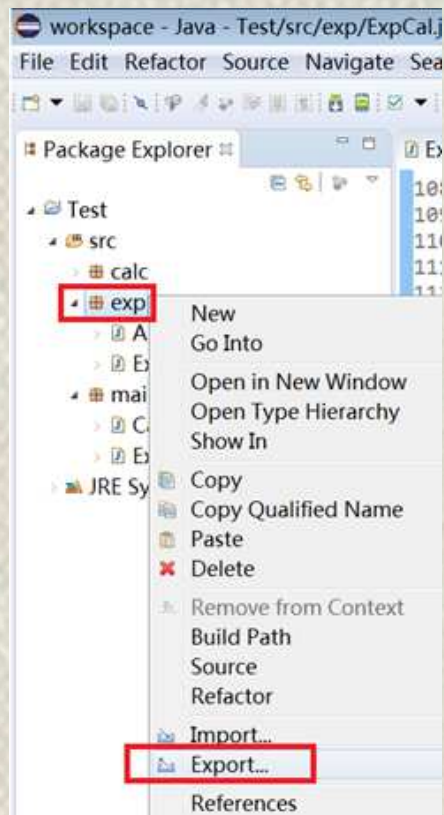
```
1 package main;
2
3 import exp.ExpCal;
4
5 public class ExpMain {
6
7     public static void main(String[] args) {
8
9         // 输入
10        String exp = "5+12*(3+5)/7";
11
12        // 运算
13        String res = ExpCal.getExpCalc(exp);
14
15        // 输出
16        System.out.println(exp + " =\t" + res);
17    }
18 }
19
```

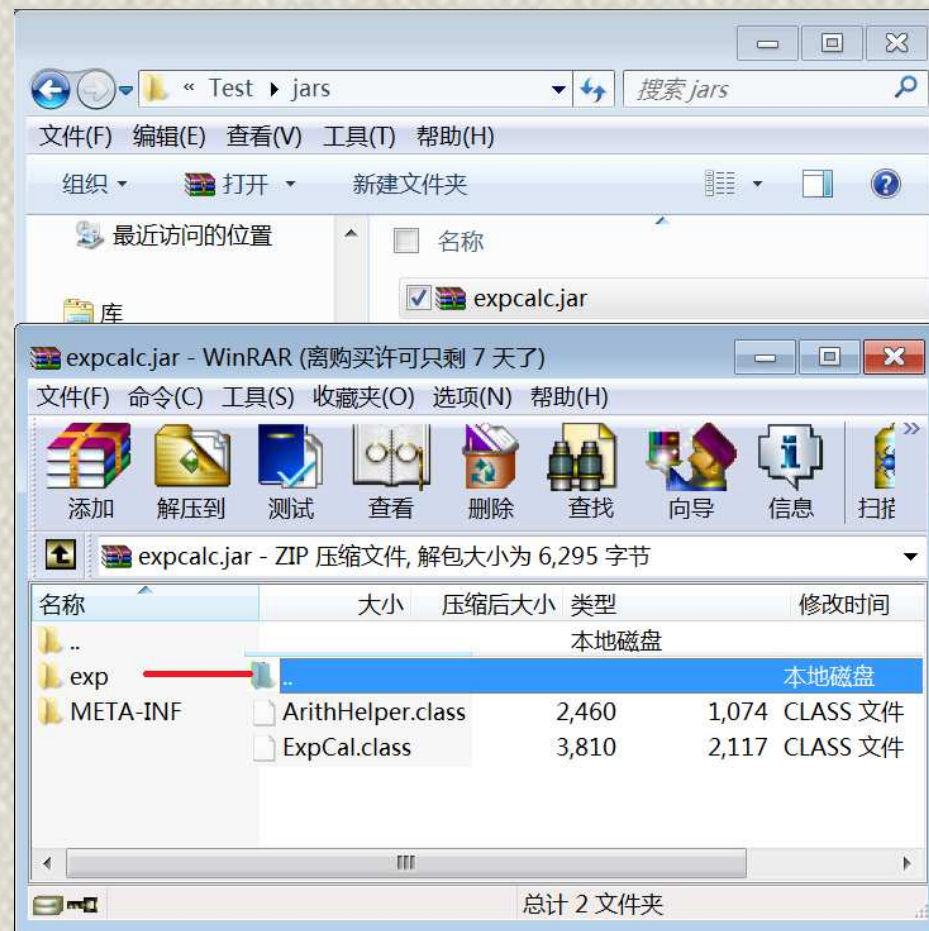
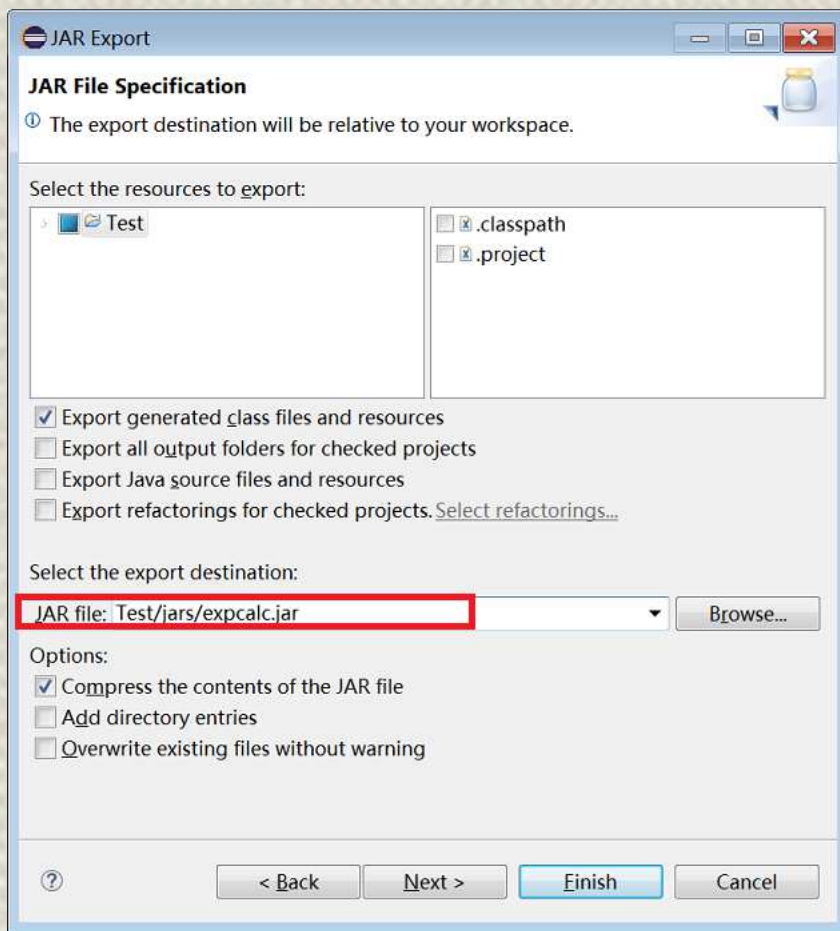
ExpCal.java

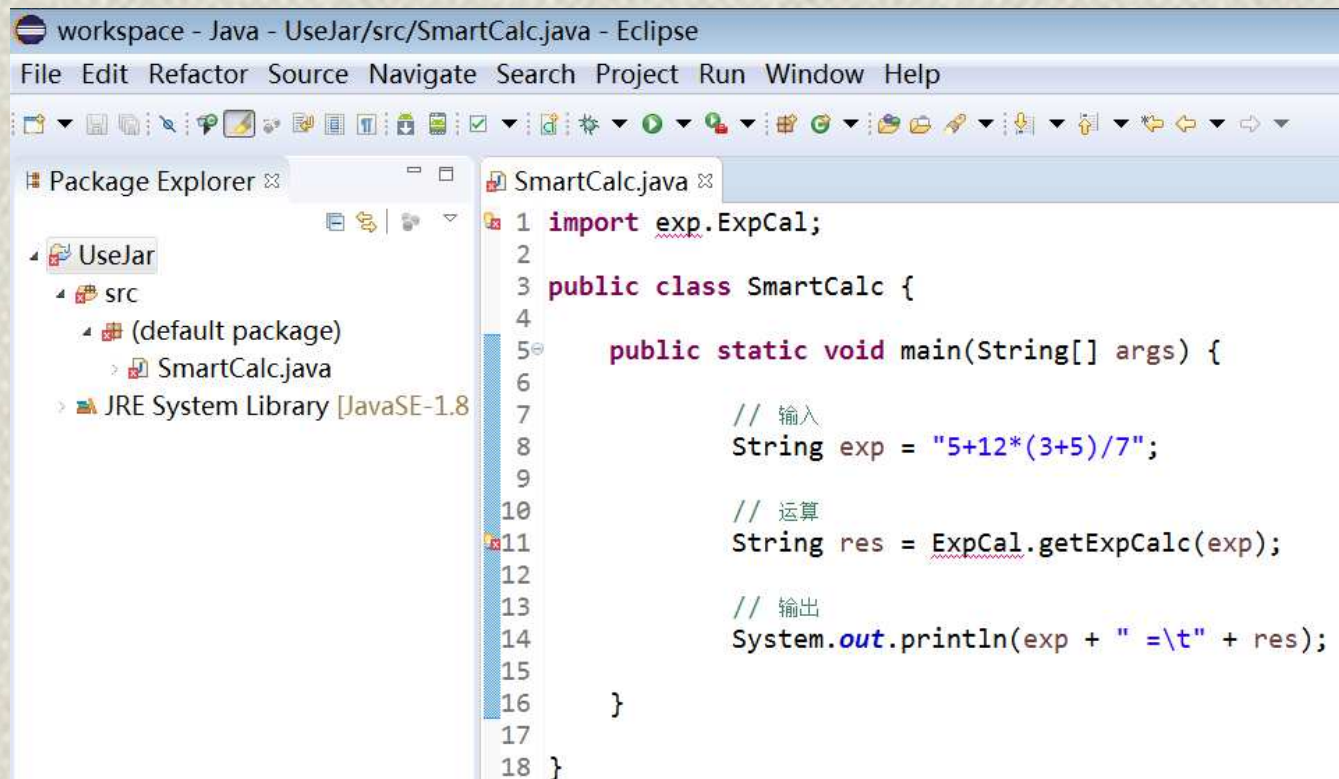
```
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
```

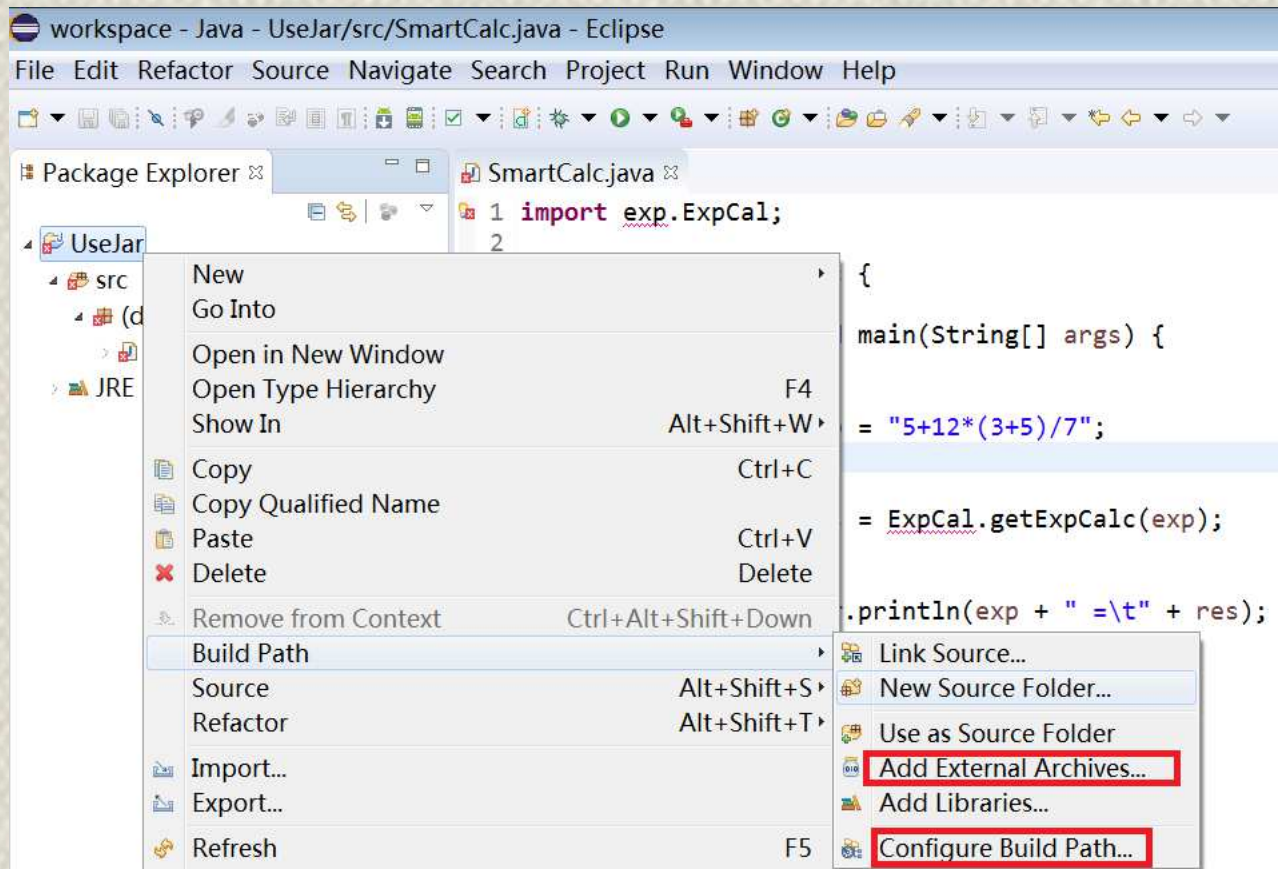
ArithHelper.java

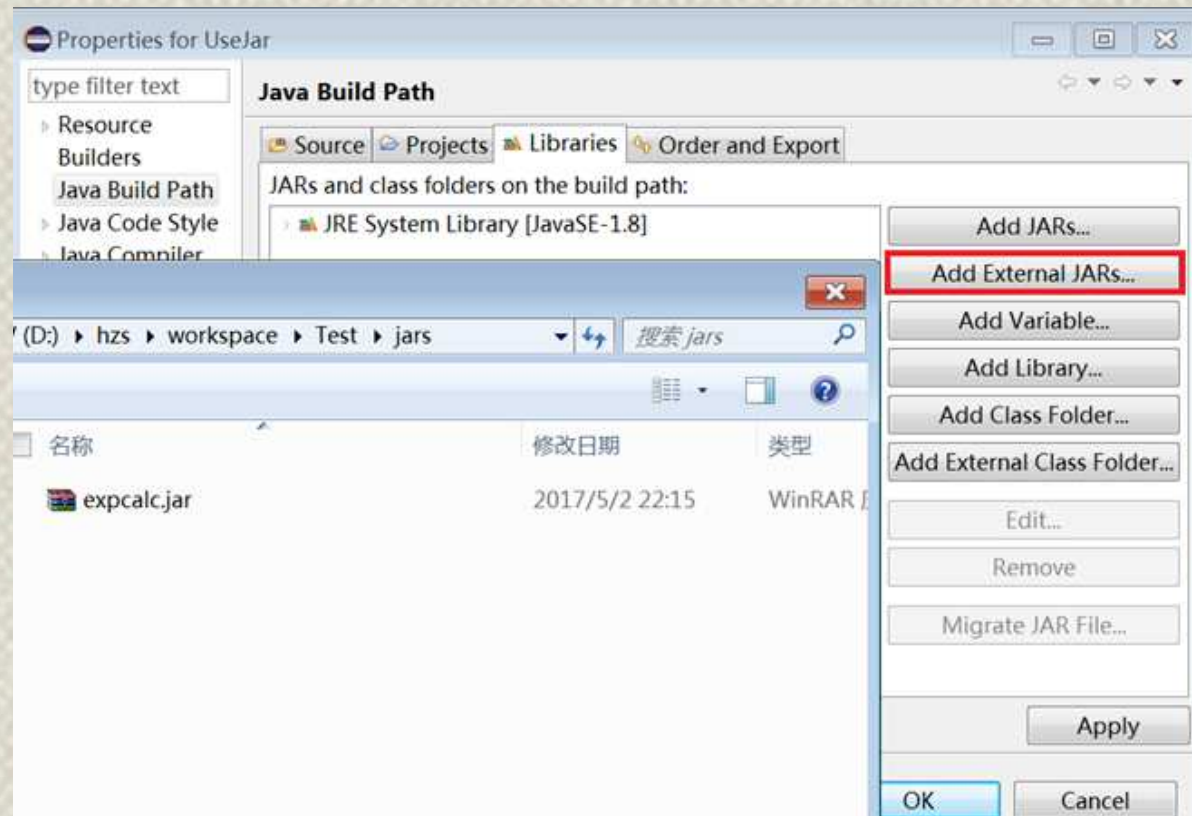
```
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
```

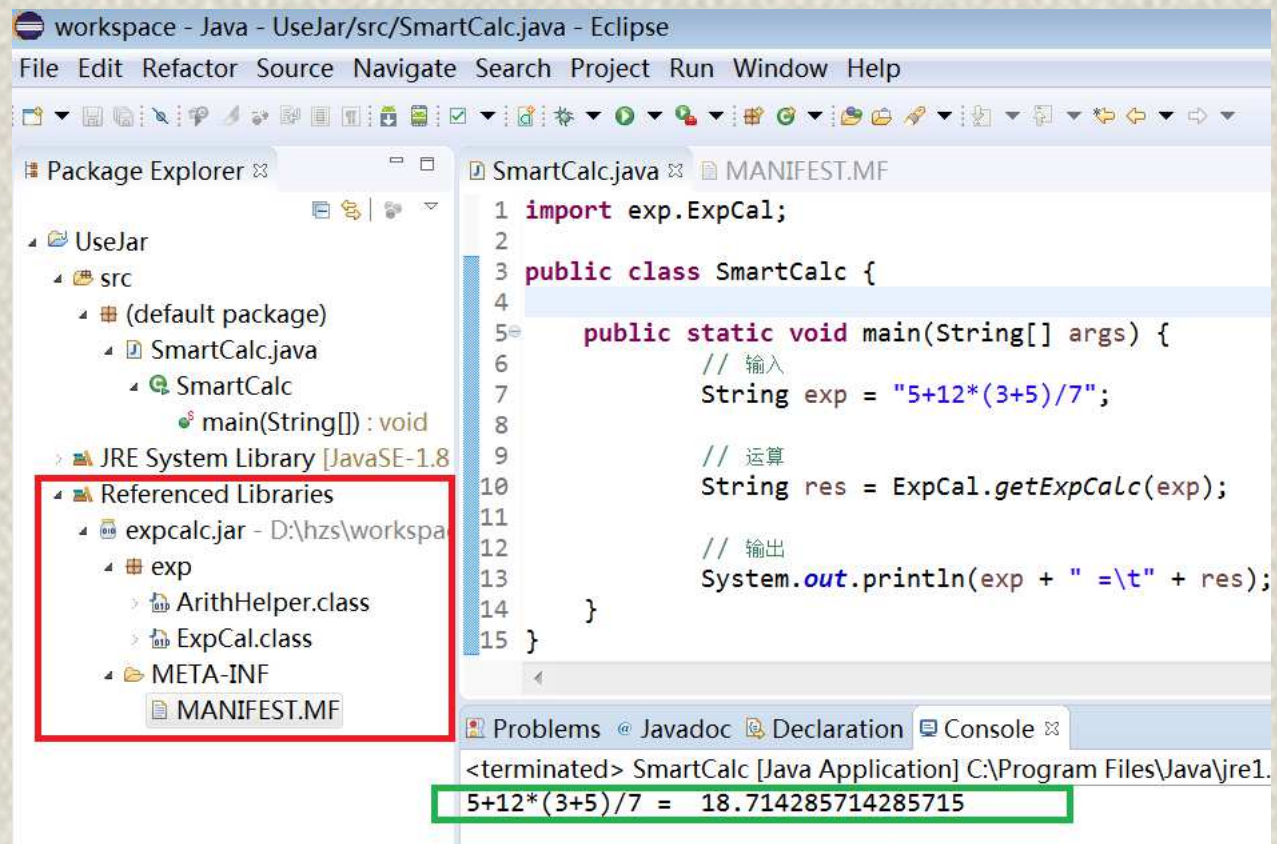











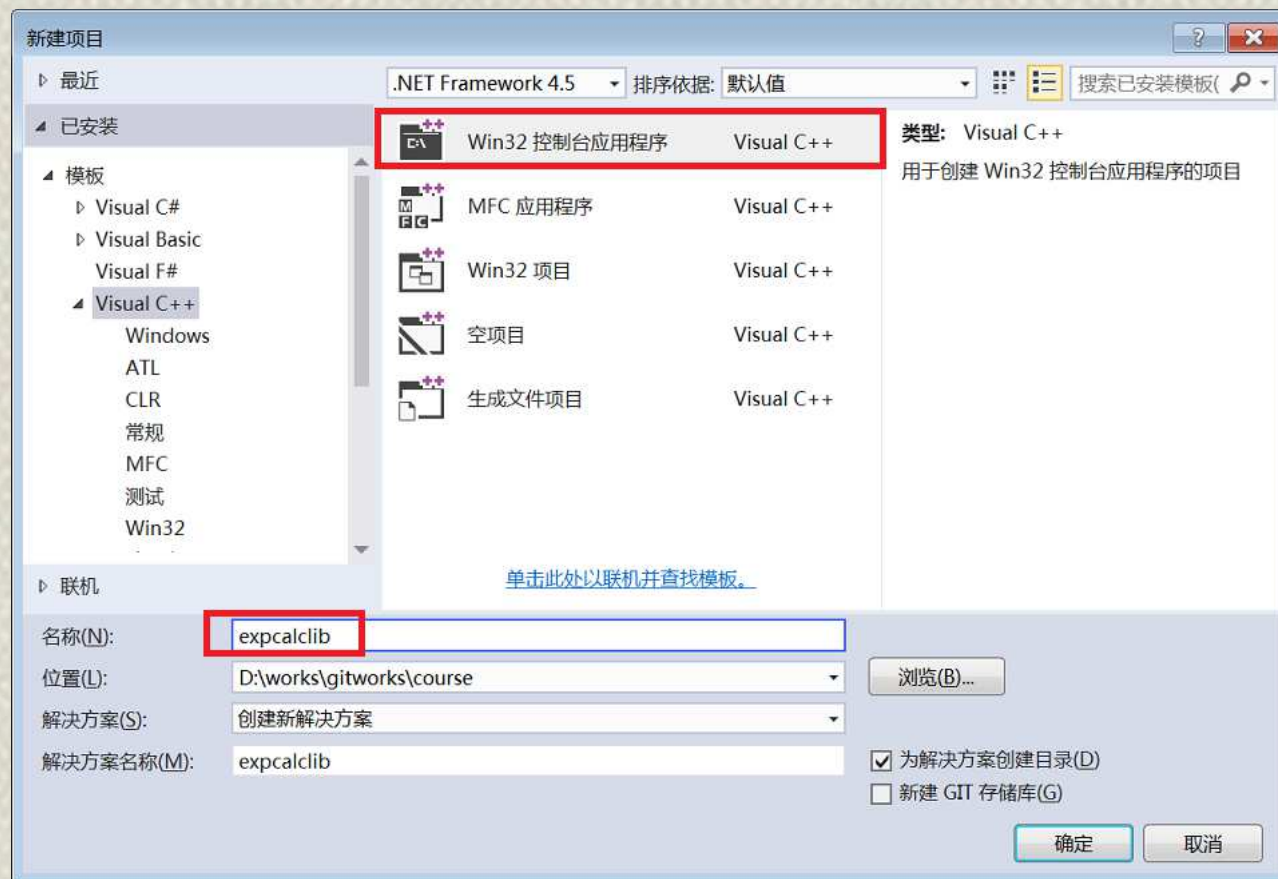


使用命令

- `Javac ExpCal.java`
- `jar cvf expcalc.jar exp/ExpCal.class
exp/ArithHelper.class`
- `jar cvf expcalc.jar exp/*.class`

Lib

- 一个静态库的制作方法



Win32 应用程序向导 - expcalclib_d

应用程序设置

概述
应用程序设置

应用程序类型:

☐ Windows 应用程序 (W)
☐ 控制台应用程序 (C)
☐ DLL (D)
☒ 静态库 (S)

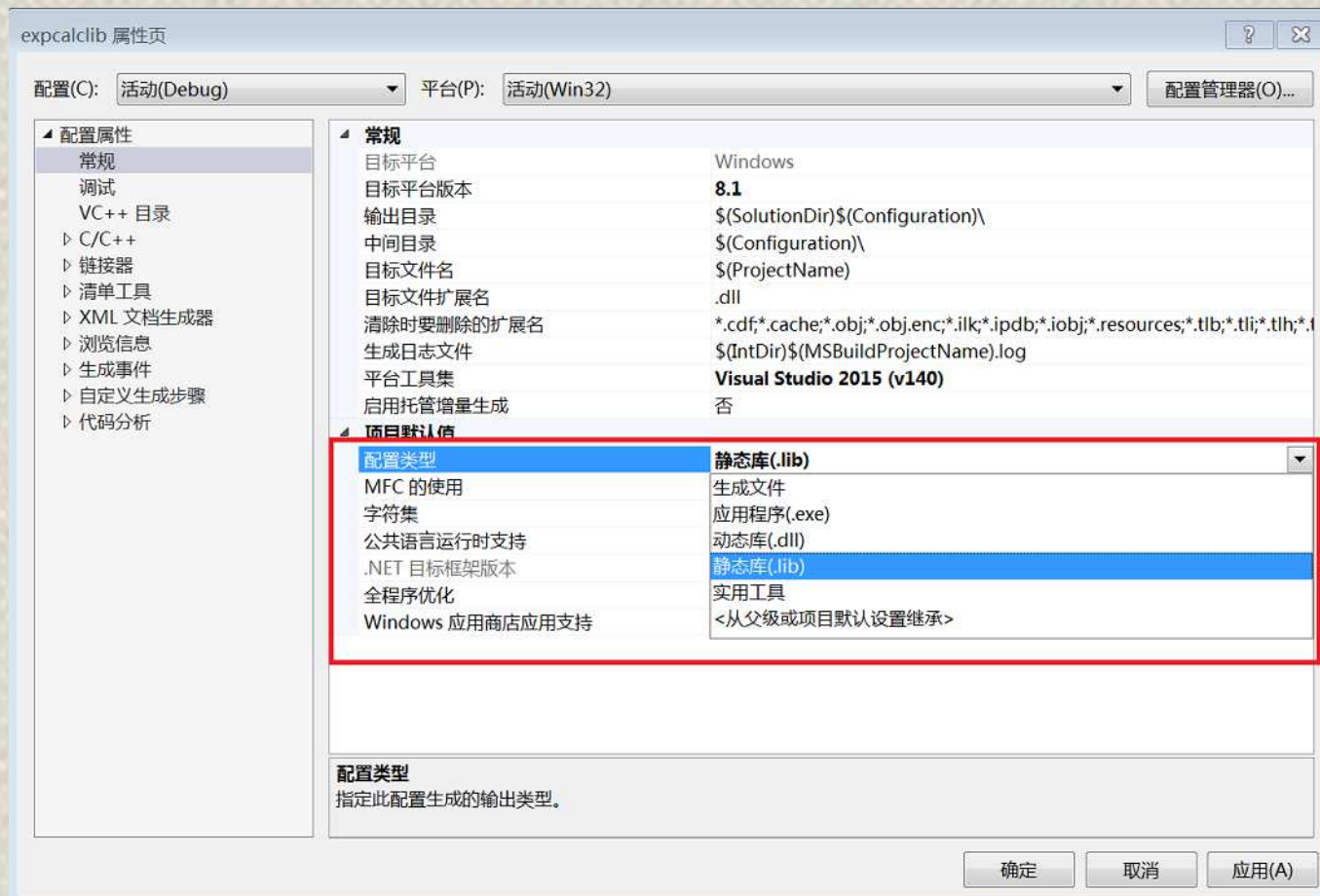
附加选项:

☐ 空项目 (E)
☐ 导出符号 (X)
☒ 预编译头 (P)
☒ 安全开发生命周期 (SDL) 检查 (C)

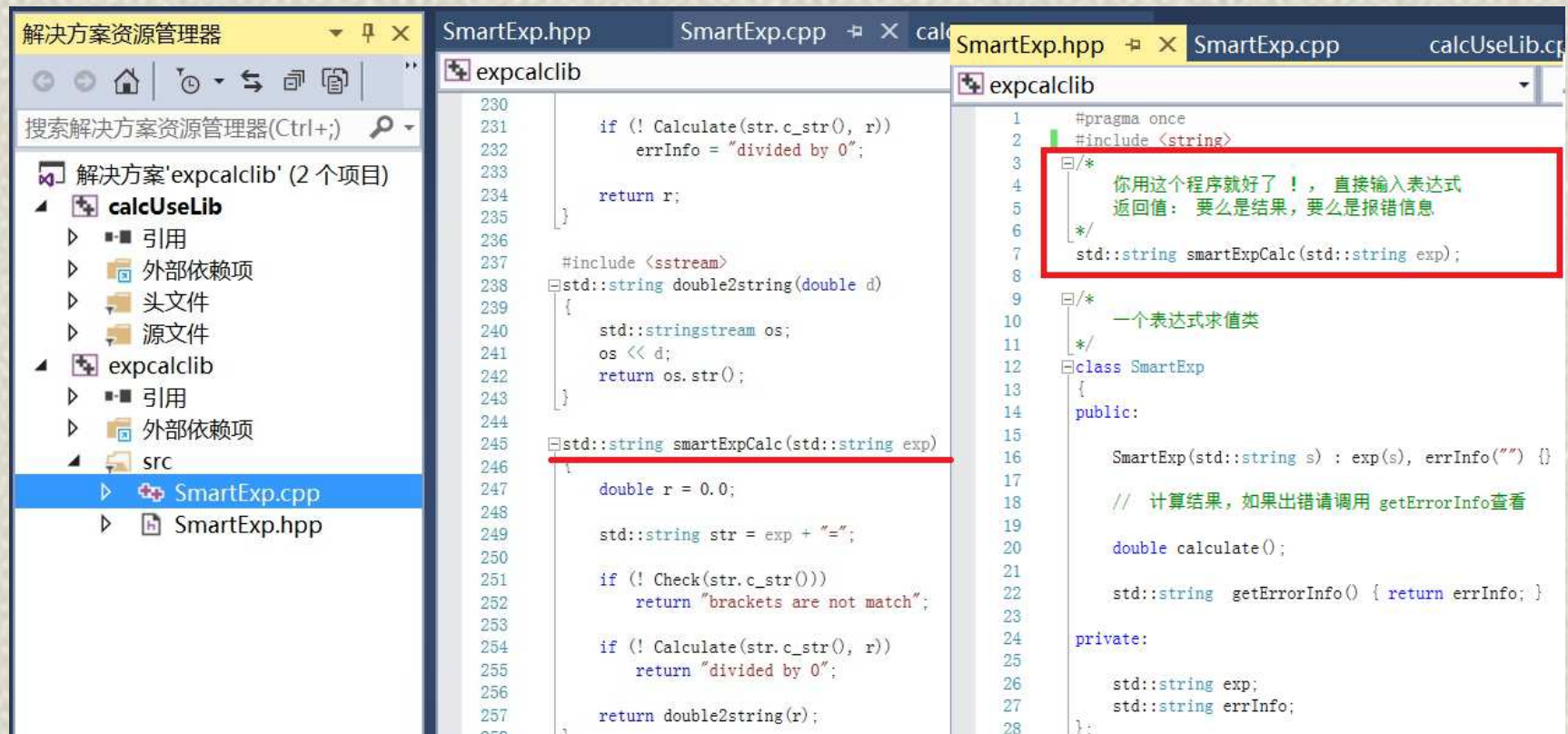
添加公共头文件以用于:

☐ ATL (A)
☐ MFC (M)

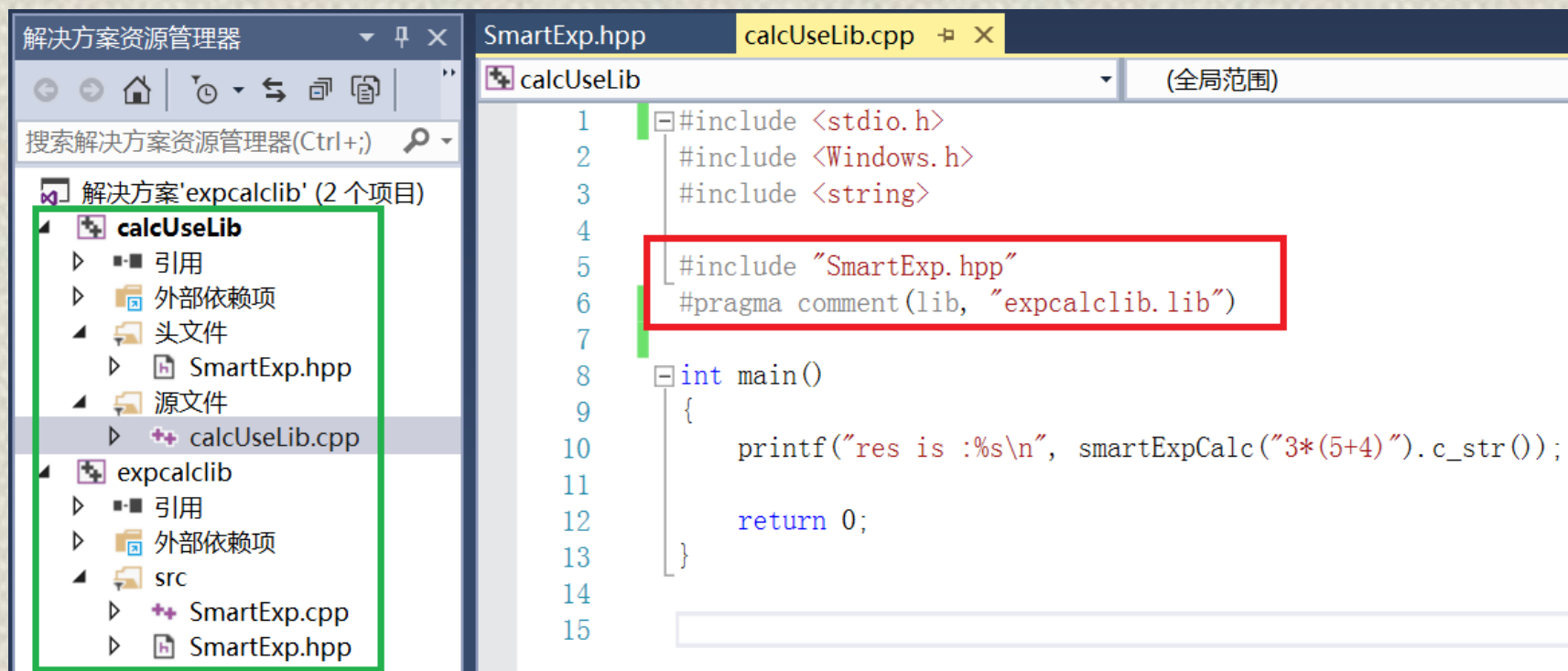
< 上一步 下一步 > 完成 取消

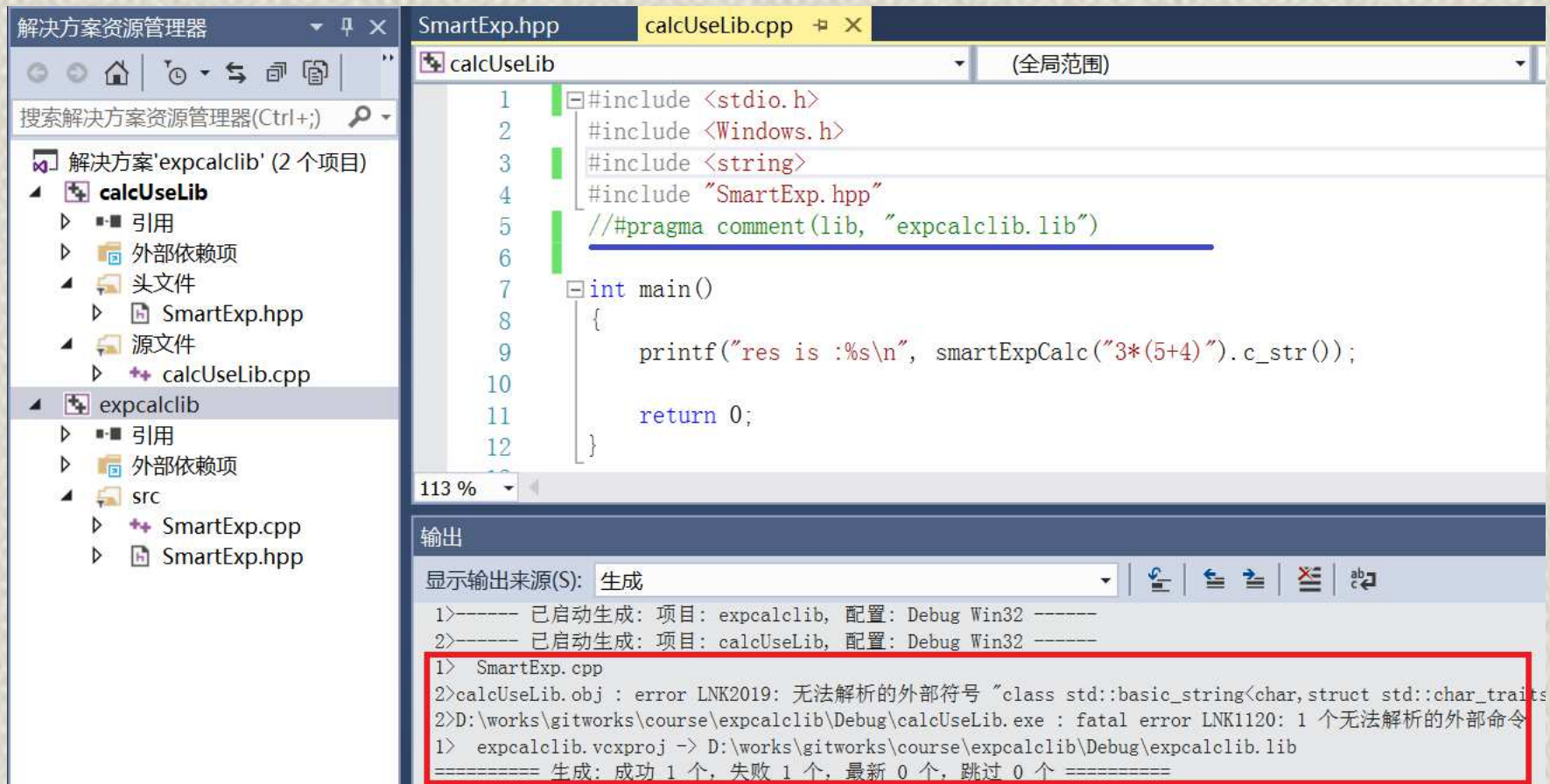


库源文件

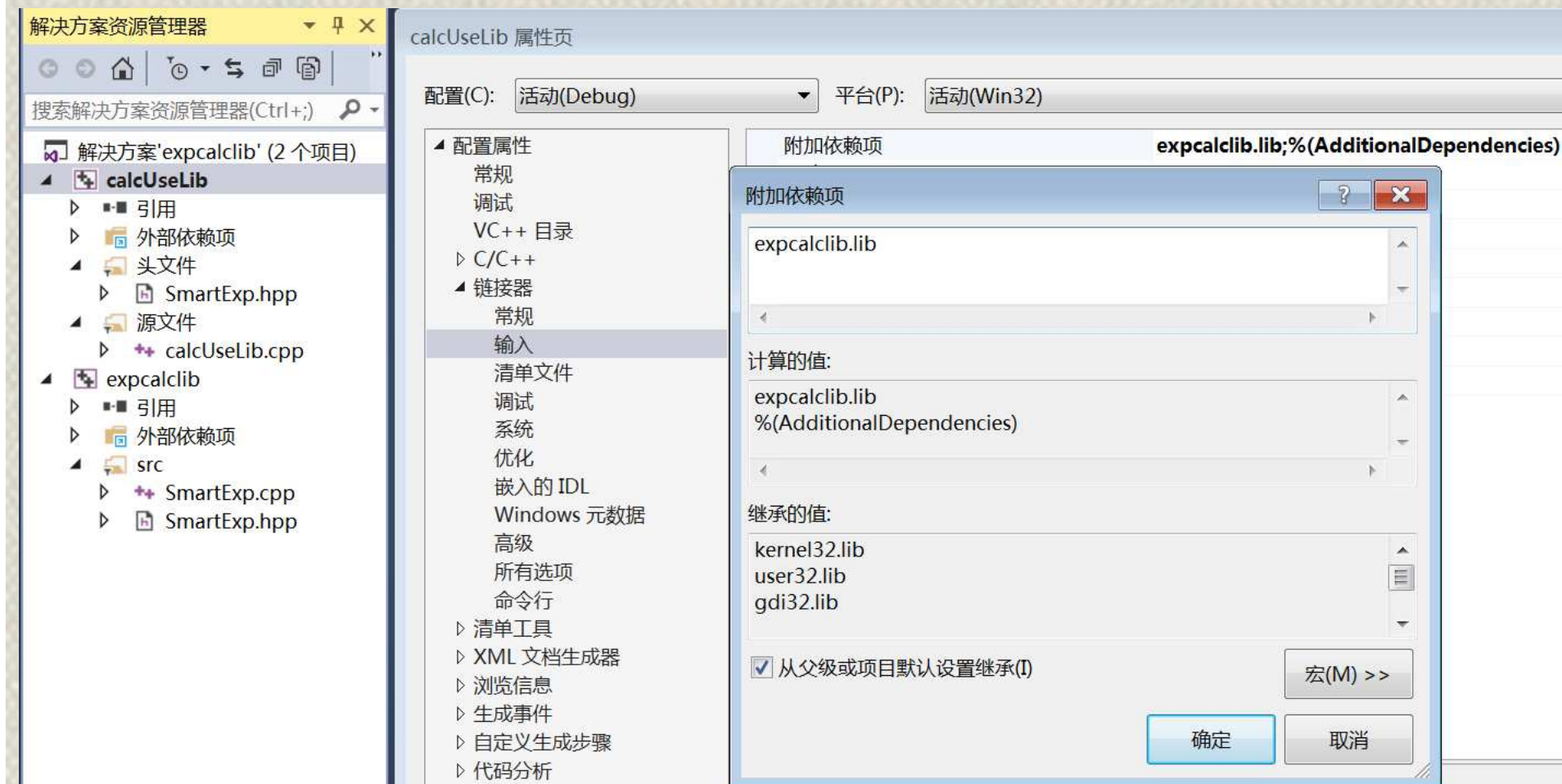


库的使用

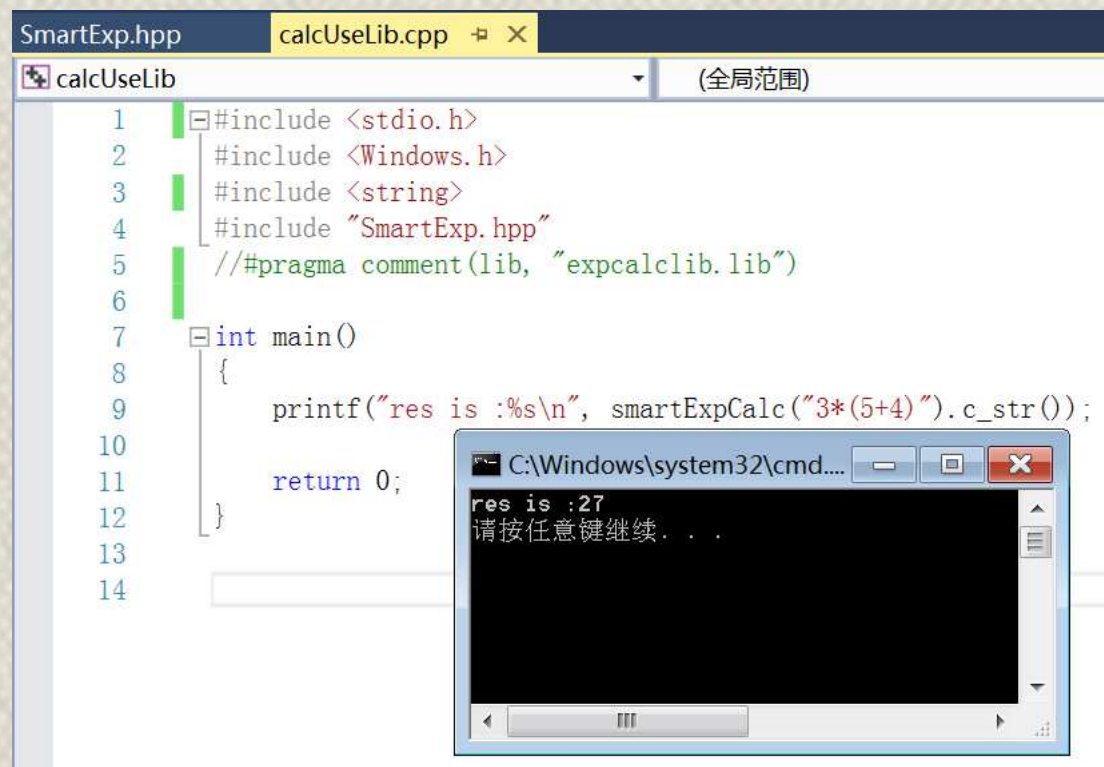




设置



结果是这样



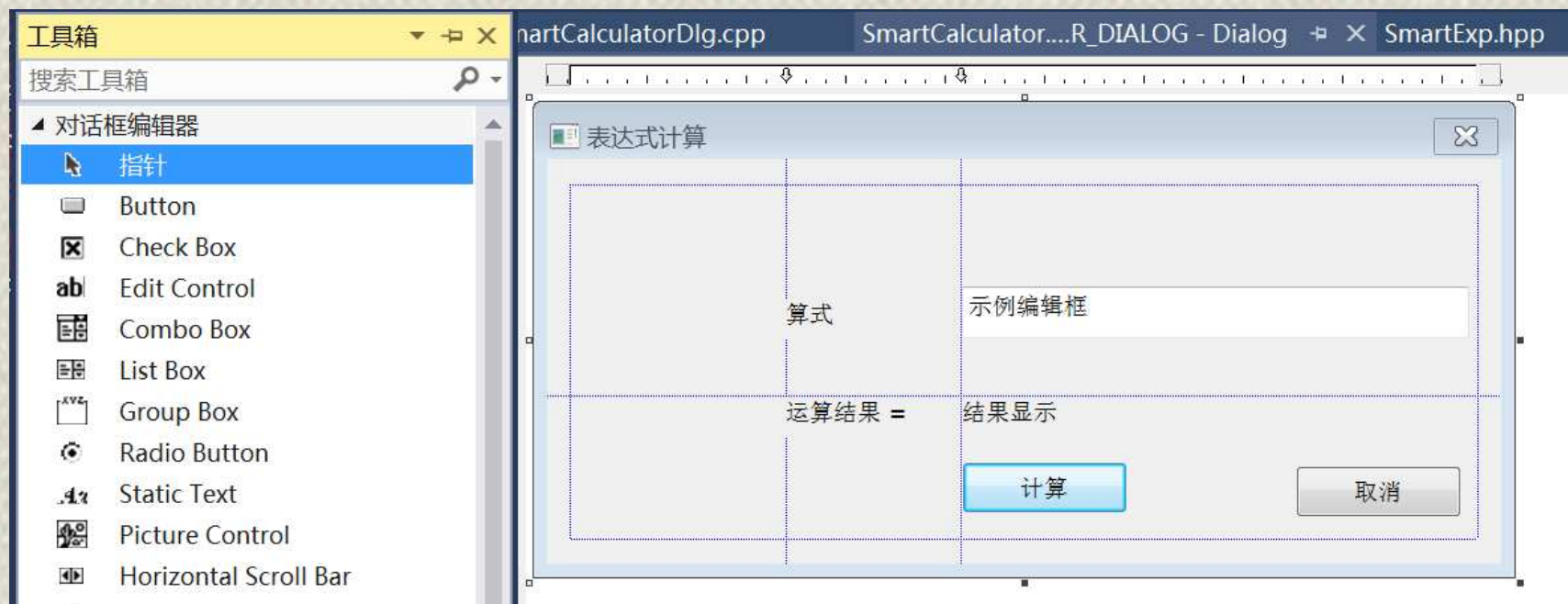
The image shows a C++ IDE with two tabs: SmartExp.hpp and calcUseLib.cpp. The calcUseLib.cpp tab is active, showing the following code:

```
1  #include <stdio.h>
2  #include <Windows.h>
3  #include <string>
4  #include "SmartExp.hpp"
5  // #pragma comment(lib, "expcalclib.lib")
6
7  int main()
8  {
9      printf("res is :%s\n", smartExpCalc("3*(5+4)").c_str());
10
11      return 0;
12  }
```

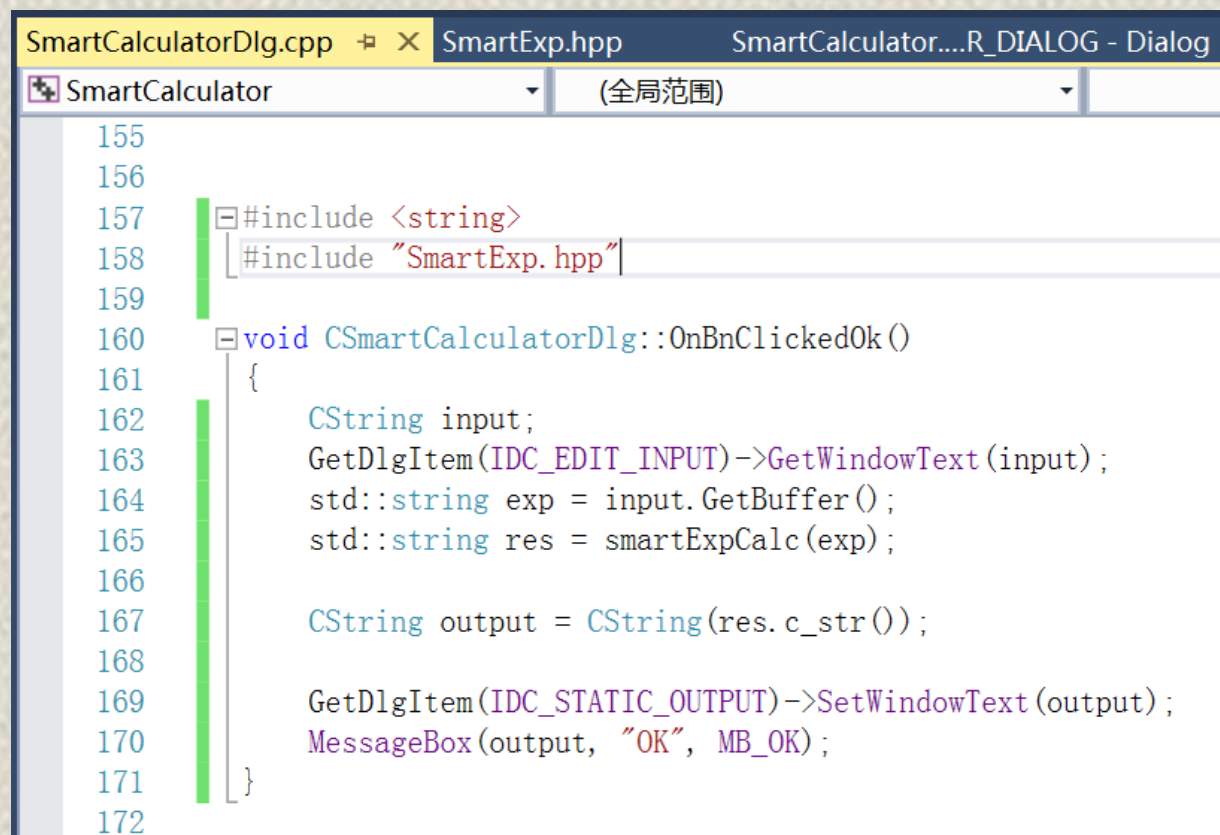
Below the code editor, a command prompt window is open, displaying the output of the program:

```
C:\Windows\system32\cmd...
res is :27
请按任意键继续...
```


改头换面



改头换面



```
SmartCalculatorDlg.cpp  SmartExp.hpp  SmartCalculator....R_DIALOG - Dialog
SmartCalculator  (全局范围)
155
156
157 #include <string>
158 #include "SmartExp.hpp"
159
160 void CSmartCalculatorDlg::OnBnClickedOk()
161 {
162     CString input;
163     GetDlgItem(IDC_EDIT_INPUT)->GetWindowText(input);
164     std::string exp = input.GetBuffer();
165     std::string res = smartExpCalc(exp);
166
167     CString output = CString(res.c_str());
168
169     GetDlgItem(IDC_STATIC_OUTPUT)->SetWindowText(output);
170     MessageBox(output, "OK", MB_OK);
171 }
172
```

输出结果

表达式计算

算式

运算结果 = 48