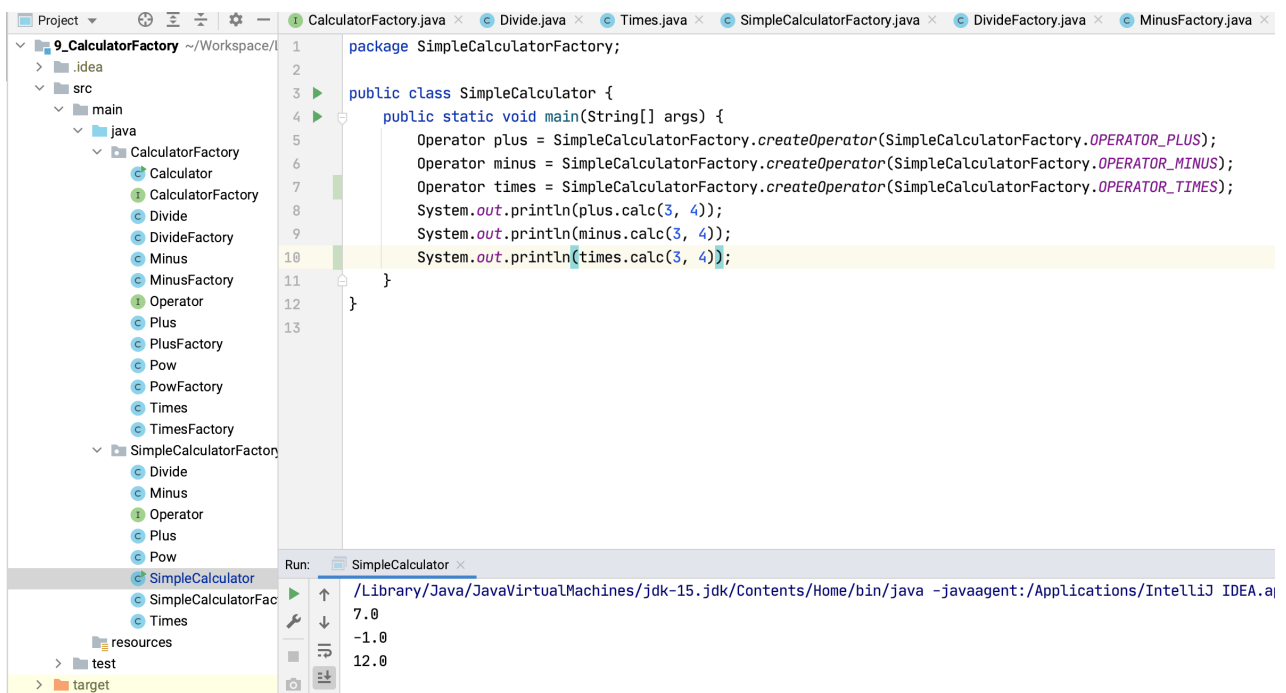


# Calculator Factory

10185101210 陈俊潼

## Simple Calculator Factory

运行截图：



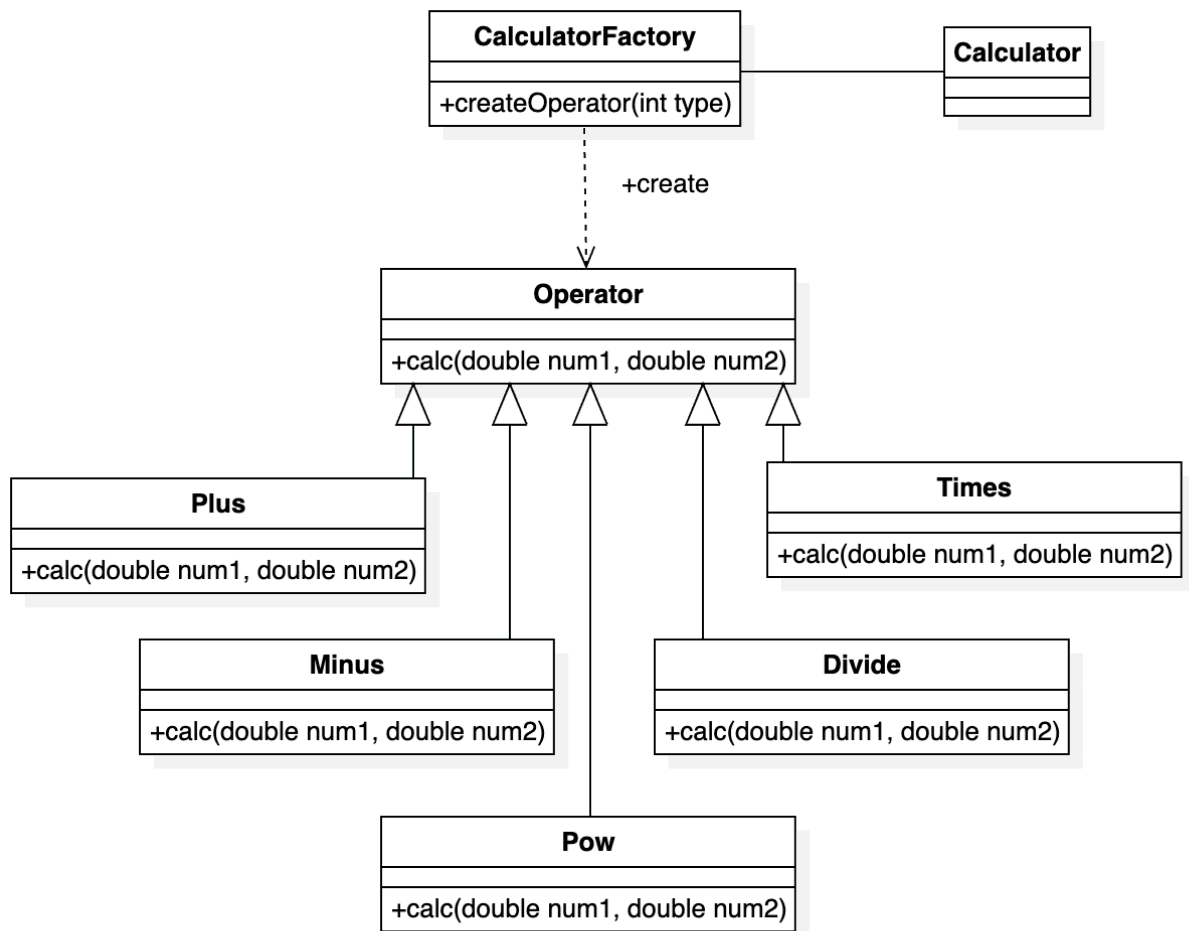
The screenshot displays an IDE window with the project '9\_CalculatorFactory' open. The file 'SimpleCalculatorFactory.java' is selected in the Project Explorer on the left. The main editor shows the following Java code:

```
1 package SimpleCalculatorFactory;
2
3 public class SimpleCalculator {
4     public static void main(String[] args) {
5         Operator plus = SimpleCalculatorFactory.createOperator(SimpleCalculatorFactory.OPERATOR_PLUS);
6         Operator minus = SimpleCalculatorFactory.createOperator(SimpleCalculatorFactory.OPERATOR_MINUS);
7         Operator times = SimpleCalculatorFactory.createOperator(SimpleCalculatorFactory.OPERATOR_TIMES);
8         System.out.println(plus.calc(3, 4));
9         System.out.println(minus.calc(3, 4));
10        System.out.println(times.calc(3, 4));
11    }
12 }
13
```

The Run tab at the bottom shows the execution output for 'SimpleCalculator':

```
Run: SimpleCalculator
/Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.a
7.0
-1.0
12.0
```

类图：



## Calculator Factory

运行截图：

9\_CalculatorFactory ~/Workspace/...

- .idea
- src
  - main
    - java
      - CalculatorFactory
        - Calculator
        - CalculatorFactory
        - Divide
        - DivideFactory
        - Minus
        - MinusFactory
        - Operator
        - Plus
        - PlusFactory
        - Pow
        - PowFactory
        - Times
        - TimesFactory
      - SimpleCalculatorFactory
        - Divide
        - Minus
        - Operator
        - Plus
        - Pow
        - SimpleCalculator
        - SimpleCalculatorFac
        - Times
  - resources
  - test
  - target

```

1 package CalculatorFactory;
2
3 public class Calculator{
4     public static void main(String[] args) {
5         CalculatorFactory plusFactory = new PlusFactory();
6         Operator plus = plusFactory.createOperator();
7         CalculatorFactory minusFactory = new MinusFactory();
8         Operator minus = minusFactory.createOperator();
9         CalculatorFactory timesFactory = new TimesFactory();
10        Operator times = timesFactory.createOperator();
11
12        System.out.println(plus.calc(3, 4));
13        System.out.println(minus.calc(3, 4));
14        System.out.println(times.calc(3, 4));
15    }
16 }
17

```

Run: Calculator

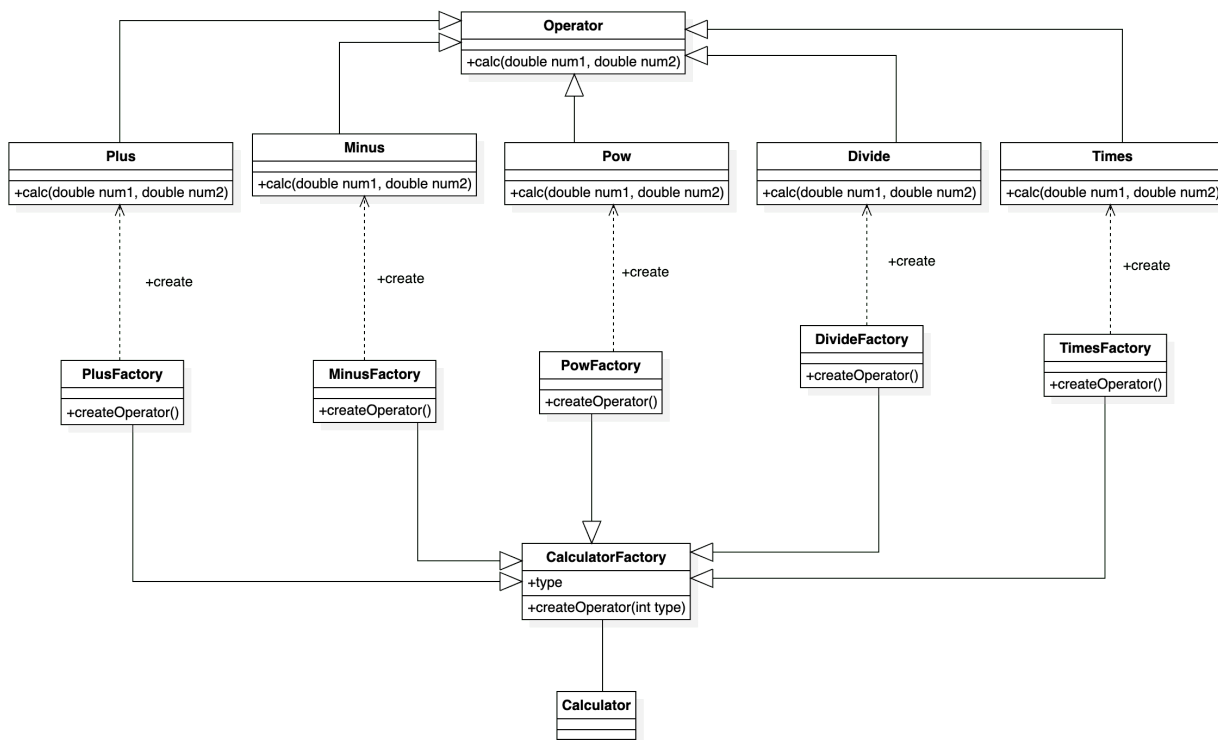
```

/Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java -j
7.0
-1.0
12.0

```

Process finished with exit code 0

类图



## 对比

工厂模式相对简单和工厂模式而言，遵循了开闭原则，又保持了封装对象创建过程的优点。因为在简单工厂模式中，如果增加了新的运算符号，需要“打开”工厂类来进行修改，改变 `switch` 语句的行为，所以对修改就不封闭了。而在工厂模式中，可以通过新增一个新的工厂类来继承基工厂，从而避免了修改之前已有的代码，对修改封闭。而此时又能够在原有的系统上进行拓展，所以遵循了对拓展的开放性。

程序代码见附件 2。