## 备忘录（Memento）

### Intent

在不违反封装的情况下获得对象的内部状态，从而在需要时可以将对象恢复到最初状态。

### Class Diagram

* Originator：原始对象
* Caretaker：负责保存好备忘录
* Memento：备忘录，存储原始对象的状态。备忘录实际上有两个接口，一个是提供给 Caretaker 的窄接口：它只能将备忘录传递给其它对象；一个是提供给 Originator 的宽接口，允许它访问到先前状态所需的所有数据。理想情况是只允许 Originator 访问本备忘录的内部状态。

### Implementation

以下实现了一个简单计算器程序，可以输入两个值，然后计算这两个值的和。备忘录模式允许将这两个值存储起来，然后在某个时刻用存储的状态进行恢复。

实现参考：[Memento Pattern - Calculator Example - Java Sourcecode](https://www.oodesign.com/memento-pattern-calculator-example-java-sourcecode.html)

/\*\*  
 \* Originator Interface  
 \*/  
public interface Calculator {  
  
 // Create Memento  
 PreviousCalculationToCareTaker backupLastCalculation();  
  
 // setMemento  
 void restorePreviousCalculation(PreviousCalculationToCareTaker memento);  
  
 int getCalculationResult();  
  
 void setFirstNumber(int firstNumber);  
  
 void setSecondNumber(int secondNumber);  
}

/\*\*  
 \* Originator Implementation  
 \*/  
public class CalculatorImp implements Calculator {  
  
 private int firstNumber;  
 private int secondNumber;  
  
 @Override  
 public PreviousCalculationToCareTaker backupLastCalculation() {  
 // create a memento object used for restoring two numbers  
 return new PreviousCalculationImp(firstNumber, secondNumber);  
 }  
  
 @Override  
 public void restorePreviousCalculation(PreviousCalculationToCareTaker memento) {  
 this.firstNumber = ((PreviousCalculationToOriginator) memento).getFirstNumber();  
 this.secondNumber = ((PreviousCalculationToOriginator) memento).getSecondNumber();  
 }  
  
 @Override  
 public int getCalculationResult() {  
 // result is adding two numbers  
 return firstNumber + secondNumber;  
 }  
  
 @Override  
 public void setFirstNumber(int firstNumber) {  
 this.firstNumber = firstNumber;  
 }  
  
 @Override  
 public void setSecondNumber(int secondNumber) {  
 this.secondNumber = secondNumber;  
 }  
}

/\*\*  
 \* Memento Interface to Originator  
 \*  
 \* This interface allows the originator to restore its state  
 \*/  
public interface PreviousCalculationToOriginator {  
 int getFirstNumber();  
 int getSecondNumber();  
}

/\*\*  
 \* Memento interface to CalculatorOperator (Caretaker)  
 \*/  
public interface PreviousCalculationToCareTaker {  
 // no operations permitted for the caretaker  
}

/\*\*  
 \* Memento Object Implementation  
 \* <p>  
 \* Note that this object implements both interfaces to Originator and CareTaker  
 \*/  
public class PreviousCalculationImp implements PreviousCalculationToCareTaker,  
 PreviousCalculationToOriginator {  
  
 private int firstNumber;  
 private int secondNumber;  
  
 public PreviousCalculationImp(int firstNumber, int secondNumber) {  
 this.firstNumber = firstNumber;  
 this.secondNumber = secondNumber;  
 }  
  
 @Override  
 public int getFirstNumber() {  
 return firstNumber;  
 }  
  
 @Override  
 public int getSecondNumber() {  
 return secondNumber;  
 }  
}

/\*\*  
 \* CareTaker object  
 \*/  
public class Client {  
  
 public static void main(String[] args) {  
 // program starts  
 Calculator calculator = new CalculatorImp();  
  
 // assume user enters two numbers  
 calculator.setFirstNumber(10);  
 calculator.setSecondNumber(100);  
  
 // find result  
 System.out.println(calculator.getCalculationResult());  
  
 // Store result of this calculation in case of error  
 PreviousCalculationToCareTaker memento = calculator.backupLastCalculation();  
  
 // user enters a number  
 calculator.setFirstNumber(17);  
  
 // user enters a wrong second number and calculates result  
 calculator.setSecondNumber(-290);  
  
 // calculate result  
 System.out.println(calculator.getCalculationResult());  
  
 // user hits CTRL + Z to undo last operation and see last result  
 calculator.restorePreviousCalculation(memento);  
  
 // result restored  
 System.out.println(calculator.getCalculationResult());  
 }  
}

110  
-273  
110

### JDK

* java.io.Serializable