Bookmark this on Delicious or ShareThis search. Main Menu Memento Pattern - Calculator Example - Java Sourcecode Home Design Principles Java Source Code Example for the Memento Pattern - Calculator Open Close Principle Dependency Inversion Principle This simple example is a calculator that finds the result of addition of two numbers, with the additional option to undo last operation and Interface Segregation Principle restore previous result. Single Responsibility Principle Liskov's Substitution Principle <<interface>> Creational Patterns <<interface>> Singleton PreviousCalculationToOriginator Factory backupLastCalculation(): PreviousCalculationToCareTaker PreviousCalculationToCareTaker Factory Method restorePreviousCalculation(memento : PreviousCalculationToCareTaker) : void getFirstNumber(): int Abstract Factory getCalculationResult(): int getSecondNumber() : inf Builder setFirstNumber(firstNumber : int) : void <u>Prototype</u> setSecondNumber(secondNumber : int) : void PreviousCalculationImp -> PreviousCalculationToCareTaker Object Pool <<realize>> PreviousCalculationImp -> PreviousCalculationToOriginator Behavioral Patterns CalculatorImp -> Calculator – – ≺≺reatizę>> Chain of Responsibility <<realize>> Command Interpreter CalculatorImp Iterator PreviousCalculationImp Mediator firstNumber : int Memento secondNumber : int secondNumber : int Observer backupLastCalculation(): PreviousCalculationToCareTaker <<create>> PreviousCalculationImp(firstNumber : int,secondNumber : int) getCalculationResult(): int Template Method getFirstNumber(): int restorePreviousCalculation(memento : PreviousCalculationToCareTaker) : void Visitor getSecondNumber(): int setFirstNumber(firstNumber : int) : void Null Object setSecondNumber(secondNumber : int) : void Structural Patterns Adapter The code below shows the memento object interface to caretaker. Note that this interface is a placeholder only and has no methods to honor Bridge encapsulation in that the memento is opaque to the caretaker. Composite Decorator Flyweight package memento; Proxy Design Pattern Books Memento interface to CalculatorOperator (Caretaker) Forum What Design Pattern To Choose? public interface PreviousCalculationToCareTaker { Design Principles and Patterns // no operations permitted for the caretaker **Enterprise Patterns** } **Books** The code below shows the Memento to Originator interface; note that this interface provides the necessary methods for the originator to restore its original state. package memento; * Memento Interface to Originator * This interface allows the originator to restore its state public interface PreviousCalculationToOriginator { public int getFirstNumber(); public int getSecondNumber(); } The code below shows the memento implementation, note that the memento must implement two interfaces, the one to the caretaker as well as the one to the originator. package memento;

* Note that this object implements both interfaces to Originator and CareTaker */
public class PreviousCalculationImp implements PreviousCalculationToCareTaker,

PreviousCalculationToOriginator {

* Memento Object Implementation

private int firstNumber; private int secondNumber;

The code below shows the calculator interface which is the originator interface

The code below shows the Calculator implementation which is the originator implementation. Note that the backupLastCalculation method corresponds to createMemento() method discussed previously, in this method the memento object is created and all originator state is saved to the memento. Also note that the method restorePreviousCalculation() method corresponds to setMemento() method . Inside this method the logic to restore the previous state is executed.

```
package memento;
* 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 int getCalculationResult() {
                // result is adding two numbers
                return firstNumber + secondNumber;
       }
        @Override
       public void restorePreviousCalculation(PreviousCalculationToCareTaker memento) {
                this.firstNumber = ((PreviousCalculationToOriginator)memento).getFirstNumber();
                this.secondNumber = ((PreviousCalculationToOriginator)memento).getSecondNumber();
       }
        @Override
        public void setFirstNumber(int firstNumber) {
                this firstNumber = firstNumber:
        }
        @Override
        public void setSecondNumber(int secondNumber) {
```

```
this.secondNumber = secondNumber;
         }
}
The code below shows the calculator driver which simulates a user using the calculator to add numbers, the user calculates a result, then enters
wrong numbers, he is not satisfied with the result and he hits Ctrl + Z to undo last operation and restore previous result.
package memento;
 * CareTaker object
 public class CalculatorDriver {
         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());
         }
 }
                                                 < Prev
                                                               Next >
[Back]
```

Traffic Rank oodesign.com 137,892 Powered by @Alexa