



Design Patterns Tutorial

- ▣ [Design Patterns - Home](#)
- ▣ [Design Patterns - Overview](#)
- ▣ [Design Patterns - Factory Pattern](#)
- ▣ [Abstract Factory Pattern](#)
- ▣ [Design Patterns - Singleton Pattern](#)
- ▣ [Design Patterns - Builder Pattern](#)
- ▣ [Design Patterns - Prototype Pattern](#)
- ▣ [Design Patterns - Adapter Pattern](#)
- ▣ [Design Patterns - Bridge Pattern](#)
- ▣ [Design Patterns - Filter Pattern](#)
- ▣ [Design Patterns - Composite Pattern](#)
- ▣ [Design Patterns - Decorator Pattern](#)
- ▣ [Design Patterns - Facade Pattern](#)
- ▣ [Design Patterns - Flyweight Pattern](#)
- ▣ [Design Patterns - Proxy Pattern](#)
- ▣ [Chain of Responsibility Pattern](#)
- ▣ [Design Patterns - Command Pattern](#)
- ▣ [Design Patterns - Interpreter Pattern](#)
- ▣ [Design Patterns - Iterator Pattern](#)



▣ Design Patterns - Observer Pattern

▣ Design Patterns - State Pattern

▣ Design Patterns - Null Object Pattern

▣ Design Patterns - Strategy Pattern

▣ Design Patterns - Template Pattern

▣ Design Patterns - Visitor Pattern

▣ Design Patterns - MVC Pattern

▣ Business Delegate Pattern

▣ Composite Entity Pattern

▣ Data Access Object Pattern

▣ Front Controller Pattern

▣ Intercepting Filter Pattern

▣ Service Locator Pattern

▣ Transfer Object Pattern

Design Patterns Resources

▣ Design Patterns - Questions/Answers

▣ Design Patterns - Quick Guide

▣ Design Patterns - Useful Resources

▣ Design Patterns - Discussion

Design Patterns - Memento Pattern

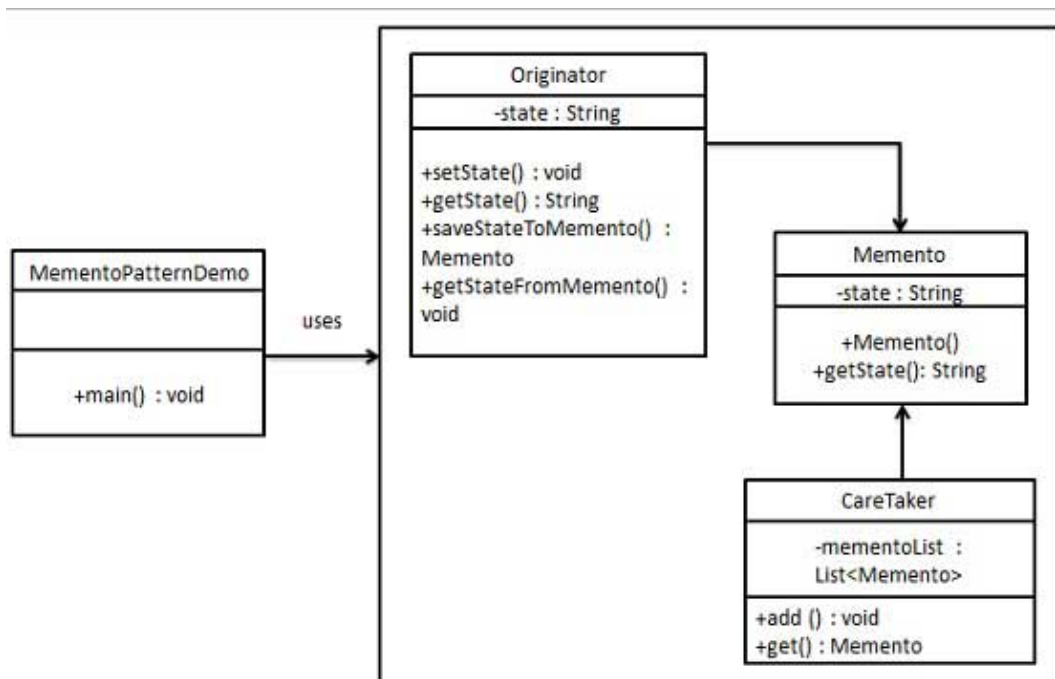

[⬅ Previous Page](#)
[Next Page ➡](#)

Memento pattern is used to restore state of an object to a previous state. Memento pattern falls under behavioral pattern category.

Implementation

Memento pattern uses three actor classes. Memento contains state of an object to be restored. Originator creates and stores states in Memento objects and Caretaker object is responsible to restore object state from Memento. We have created classes *Memento*, *Originator* and *Caretaker*.

MementoPatternDemo, our demo class, will use *Caretaker* and *Originator* objects to show restoration of object states.



Step 1

Create Memento class.

Memento.java

```

public class Memento {
    private String state;

    public Memento(String state){

```



```
}  
}
```

Step 2

Create Originator class

Originator.java

```
public class Originator {  
    private String state;  
  
    public void setState(String state){  
        this.state = state;  
    }  
  
    public String getState(){  
        return state;  
    }  
  
    public Memento saveStateToMemento(){  
        return new Memento(state);  
    }  
  
    public void getStateFromMemento(Memento Memento){  
        state = Memento.getState();  
    }  
}
```

Step 3

Create CareTaker class

CareTaker.java

```
import java.util.ArrayList;  
import java.util.List;  
  
public class CareTaker {  
    private List<Memento> mementoList = new ArrayList<Memento>();  
  
    public void add(Memento state){  
        mementoList.add(state);  
    }  
  
    public Memento get(int index){  
        return mementoList.get(index);  
    }  
}
```

Step 4



```
public class MementoPatternDemo {  
    public static void main(String[] args) {  
  
        Originator originator = new Originator();  
        CareTaker careTaker = new CareTaker();  
  
        originator.setState("State #1");  
        originator.setState("State #2");  
        careTaker.add(originator.saveStateToMemento());  
  
        originator.setState("State #3");  
        careTaker.add(originator.saveStateToMemento());  
  
        originator.setState("State #4");  
        System.out.println("Current State: " + originator.getState());  
  
        originator.getStateFromMemento(careTaker.get(0));  
        System.out.println("First saved State: " + originator.getState());  
        originator.getStateFromMemento(careTaker.get(1));  
        System.out.println("Second saved State: " + originator.getState());  
    }  
}
```

Step 5

Verify the output.

```
Current State: State #4  
First saved State: State #2  
Second saved State: State #3
```

[⬅ Previous Page](#)[Next Page ➡](#)

Advertisements



marketplace.openshift.com

Add Databases, Monitoring, Search, Messaging, Scheduling, and More.





© Copyright 2015. All Rights Reserved.

Enter email for newsletter

go