6.16 Mẫu thiết kế MEMENTO

6.16.1 Mở đầu

Đôi khi cần thiết ghi lại trạng thái bên trong của đối tượng. Điều này được đòi hỏi khi cài đặt các điểm kiểm tra *checkpoints* và cơ chế *"undo"* mà cho phép người sử dụng quay lại các thao tác tạm thời hoặc khôi phục từ lỗi. Bạn cần lưu thông tin trạng thái ở đâu đó, sao cho bạn có thể khôi phục đối tượng về điều kiện trước đó của chúng. Nhưng các đối tượng thường đóng gói một số hoặc tất cả về trạng thái của chúng, làm cho nó không truy cập được đến các đối tượng này và không thể lưu từ bên ngoài. Mở trạng thái đó có thể vi phạm tính đóng gói, mà cần được thỏa hiệp tính tin cậy và tính mở rộng được của ứng dụng.

Mẫu *Memento* có thể được sử dụng để thực hiện điều đó mà không cần mở cấu trúc bên trong của đối tượng. Đối tượng mà trạng thái cần nắm bắt được tham chiếu như gốc originator.

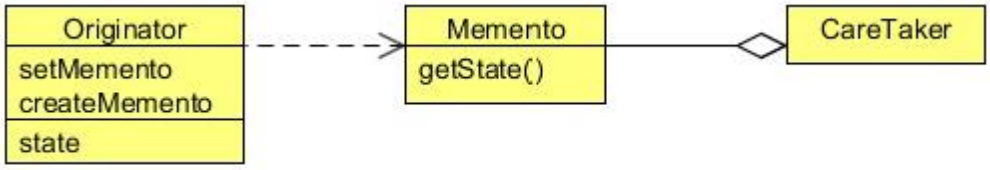
Để minh họa sử dụng mẫu *Memento*, chúng ta xét ví dụ sau. Chúng ta tạo lớp mà chứa hai trường kiểu số thực và chúng ta sẽ chạy một số phép toán trên nó. Chúng ta sẽ cung cấp cho người sử dụng thao tác *undo*. Nếu kết quả sau một số phép toán mà không thỏa mãn người sử dụng, thì người sử dụng có thể gọi thao tác *undo* mà sẽ khôi phục trạng thái của đối tượng về điểm lưu trước đó.

Ví dụ cũng sẽ bao gồm cả cơ chế điểm sao lưu mà được sử dụng bởi người sử dụng để lưu trạng thái của đối tượng. Chúng ta cũng sẽ cung cấp sự đa dạng của thao tác *undo. Undo* đơn giản có thể khôi phục trạng thái đối tượng về điểm lưu trước đó. *Undo* với điểm lưu được chỉ rõ có thể khôi phục một trạng thái cụ thể của đối tượng đó và *undo all* sẽ xóa mọi trạng thái lưu của đối tượng và khôi phục đối tượng trong trạng thái khởi tạo, khi đối tượng được tạo ra.

Trước khi cài đặt mẫu, chúng ta sẽ tìm hiểu mẫu thiết kế *Memento*.

6.16.2 Mẫu thiết kế Memento là gì

Mục đích của mẫu *Memento* là không vi phạm tính đóng gói, hãy nắm bắt và xuất ra trạng thái bên trong của đối tượng sao cho đối tượng đó có thể được khôi phục đến trạng thái này về sau.



Memento

- Lưu trạng thái bên trong của đối tượng gốc *Originator*. *Memento* có thể lưu giữ nhiều hoặc ít về trạng thái bên trong *originator* như cần thiết theo quyết định của *originator*.
- Bảo vệ chống truy cập bởi các đối tượng khác ngoài *originator*. *Memento* có hai giao diện hiệu quả. Người chăm lo thấy giao diện hẹp đến Memento nó có thể truyền memento cho các đối tượng khác. *Originator*, ngược lại, nhìn thấy giao diện rộng, người ta cho phép nó truy cập mọi dữ liệu cần thiết để khôi phục bản thân nó về trạng thái trước đó của nó. Ly tưởng là, chỉ originator mà tạo ra *memento* có thể được cho phép truy cập trạng thái bên trong của *memento*.

Originator

- Tạo *memento* chứa bản chụp trạng thái bên trong hiện thời của nó
- Sử dụng *memento* để khôi phục trạng thái bên trong của nó.

Người chăm lo

- Có trách nhiệm lưu giữ *memento*.
- Không bao giờ thao tác trên hoặc kiểm tra nội dung của *memento*.

Khi *client* muốn lưu trạng thái của *originator*, nó yêu cầu trạng thái hiện tại từ *originator*. *Originator* lưu giữ mọi thuộc tính mà được yêu cầu để khôi phục trạng thái của nó trong đối tượng riêng biệt được tham chiếu như *Memento* và trả về nó cho *client*. Như vậy *Memento* có thể được xem như một đối tượng mà chứa trạng thái bên trong của một đối tượng khác tại một thời điểm nhất định. Đối tượng *Memento* cần che giấu giá trị biến đổi của *originator* khỏi mọi đối tượng trừ *originator*. Nói cách khác, nó cần phải bảo vệ trạng thái bên trong của nó chống việc truy cập bởi các đối tượng khác ngoài *originator*. Từ nay về sau, *Memento* cần được thiết kế để cung cấp truy cập hạn chế cho các đối tượng khác, trong khi *originator* được cho phép truy cập đến trạng thái bên trong của nó.

Khi *client* muốn khôi phục *originator* trở lại trạng thái trước đó, nó đơn giản truyền *memento* trở lại cho *originator*. *Originator* sử dụng thông tin trạng thái này trong *memento* và đẩy bản thân nó trở về trạng thái được lưu trong đối tượng *memento*.

6.16.3 Cài đặt mẫu Memento

```
package com.javacodegeeks.patterns.mementopattern;

public class Originator {

        private double x;
        private double y;

        private String lastUndoSavepoint;
        CareTaker careTaker;

        public Originator(double x, double y,CareTaker careTaker){
                this.x = x;
                this.y = y;

                this.careTaker = careTaker;

                createSavepoint("INITIAL");
        }

        public double getX(){
                return x;
        }

        public double getY(){
                return y;
        }
```

```
        public void setX(double x) {
                this.x = x;
        }

        public void setY(double y) {
                this.y = y;
        }

        public void createSavepoint(String savepointName){
                careTaker.saveMemento(new Memento(this.x, this.y), savepointName);
                lastUndoSavepoint = savepointName;
        }

        public void undo(){
                setOriginatorState(lastUndoSavepoint);
        }

        public void undo(String savepointName){
                setOriginatorState(savepointName);
        }

        public void undoAll(){
                setOriginatorState("INITIAL");
                careTaker.clearSavepoints();
        }

        private void setOriginatorState(String savepointName){
                Memento mem = careTaker.getMemento(savepointName);
                this.x = mem.getX();
                this.y = mem.getY();
        }

        @Override
        public String toString(){
                return "X: "+x+", Y: "+y;
        }

}
```

Trên đây là lớp *originator*, trạng thái đối tượng của nó cần được lưu trong *memento*. Lớp này chứa hai trường số thực x và y, và cũng co một tham chiếu của người chăm lo *CareTaker*. *CareTaker* được sử dụng để lưu và truy vấn đối tượng *memento* mà biểu diễn trạng thái của đối tượng *originator*.

Trong hàm tạo, chúng ta có lưu trạng thái ban đầu của đối tượng sử dụng phương thức *createSavepoint*. Phương thức này tạo đối tượng *memento* và yêu cầu *CareTaker* quan tâm đến đối tượng. Chúng ta đã sử dụng biến *LastUndoSavepoint* mà được sử dụng để lưu tên chính của *memento* được lưu lần cuối để cài đặt thao tác *undo*.

Lớp này cung cấp các kiểu của thao tác undo. Phương thức *undo* không có tham số khôi phục trạng thái lưu cuối cùng, *undo* với tên *savepoint* như tham số khôi phục trạng thái được lưu với tên *savepoint* cụ thể đó. Phương thức *undoAll* yêu cầu CareTaker xóa mọi điểm *savepoints* và đặt nó vào trạng thái ban đầu (trạng thái tại thời điểm khởi tạo đối tượng đó).

```
package com.javacodegeeks.patterns.mementopattern;

public class Memento {

        private double x;
        private double y;

        public Memento(double x, double y){
                this.x = x;
                this.y = y;
        }

        public double getX(){
                return x;
        }

        public double getY(){
                return y;
        }
}
```

Lớp *Memento* được sử dụng để lưu trạng thái *originator* và lưu giữ bởi *CareTaker*. Lớp này không có phương set nào cả, nó chỉ được sử dụng để get trạng thái của đối tượng.

```
package com.javacodegeeks.patterns.mementopattern;

import java.util.HashMap;
import java.util.Map;

public class CareTaker {

        private final Map<String, Memento>savepointStorage = new HashMap<String, Memento>() ↩
            ;

        public void saveMemento(Memento memento,String savepointName){
                System.out.println("Saving state..."+savepointName);
                savepointStorage.put(savepointName, memento);
        }

        public Memento getMemento(String savepointName){
                System.out.println("Undo at ..."+savepointName);
                return savepointStorage.get(savepointName);
        }

        public void clearSavepoints(){
                System.out.println("Clearing all save points...");
                savepointStorage.clear();
        }

}
```

Lớp trên là người chăm lo *CareTaker* được sử dụng để lưu và cung cấp đối tượng *memento* được yêu cầu. Lớp này chứa phương thức *saveMemento* được sử dụng để lưu đối tượng *Memento*, phương thức *getMemento* được sử dụng để trả về đối tượng *memento* yêu cầu và phương thức *clearSavepoints* mà được sử dụng để xóa mọi điêm *savepoints* và nó xóa các đối tượng *memento* đã lưu.

Bây giờ chúng ta kiểm tra ví dụ trên.

```
package com.javacodegeeks.patterns.mementopattern;

public class TestMementoPattern {

        public static void main(String[] args) {

                CareTaker careTaker = new CareTaker();
                Originator originator = new Originator(5, 10, careTaker);

                System.out.println("Default State: "+originator);

                originator.setX(originator.getY()*51);

                System.out.println("State: "+originator);
                originator.createSavepoint("SAVE1");
```

```
                originator.setY(originator.getX()/22);
                System.out.println("State: "+originator);

                originator.undo();
                System.out.println("State after undo: "+originator);

                originator.setX(Math.pow(originator.getX(),3));
                originator.createSavepoint("SAVE2");
                System.out.println("State: "+originator);
                originator.setY(originator.getX()-30);
                originator.createSavepoint("SAVE3");
                System.out.println("State: "+originator);
                originator.setY(originator.getX()/22);
                originator.createSavepoint("SAVE4");
                System.out.println("State: "+originator);

                originator.undo("SAVE2");
                System.out.println("Retrieving at: "+originator);

                originator.undoAll();
                System.out.println("State after undo all: "+originator);
        }

}
```

Nó cho kết quả đầu ra như sau:

```
Saving state...INITIAL
Default State: X: 5.0, Y: 10.0
State: X: 510.0, Y: 10.0
Saving state...SAVE1
State: X: 510.0, Y: 23.181818181818183
Undo at ...SAVE1
State after undo: X: 510.0, Y: 10.0
Saving state...SAVE2
State: X: 1.32651E8, Y: 10.0
Saving state...SAVE3
State: X: 1.32651E8, Y: 1.3265097E8
Saving state...SAVE4
State: X: 1.32651E8, Y: 6029590.909090909
Undo at ...SAVE2
Retrieving at: X: 1.32651E8, Y: 10.0
Undo at ...INITIAL
Clearing all save points...
State after undo all: X: 5.0, Y: 10.0
```

Trong lớp trên, chúng ta đã tạo đối tượng *CareTaker* và sau đó gán nó cho đối tượng *Originator*. Và chúng ta đặt giá trị của x và y tương ứng là 5 và 10. Sau đó, chúng ta áp dụng một thao tác nào đó trên x và lưu trạng thái của đối tượng là "SAVE1".

Sau một số thao tác, chúng ta gọi phương thức undo và khôi phục trạng thái cuois của đối tượng mà rõ ràng được chỉ ra ở đầu ra. Sau đó ta áp dụng một vài thao tác và lại lưu các trạng thái của đối tượng như "SAVE2", "SAVE3", và "SAVE4".

Sau đó chúng ta yêu cầu *originato*r khôi phục trạng thái 'SAVE2" và gọi phương thức *undoAll* mà đặt lại trạng thái ban đầu của đối tượng và xóa mọi *savepoints*.

Lưu ý rằng, trong ví dụ trên, *Originator* là có trách nhiệm cung cấp *memento* của nó cho *CareTaker*. Lý do là như vậy chúng ta không muốn trao trách nhiệm này cho người sử dụng. Trong ví dụ của chúng ta, người sử dụng chỉ có thể đòi hỏi yêu cầu cho *savepoint* và khôi phục trạng thái của đối tượng. Trong nhiều trường hợp, *CareTaker* thao tác bên ngoài *originator* bởi một lớp khác (như được chỉ ra trong sơ đồ lớp bên trên).

6.16.4 Khi nào dùng mẫu Memento

Sử dụng mẫu *Memento* trong các trường hợp sau:

- Bản chụp nhanh trạng thái (hoặc một phần) của đối tượng cần được lưu sao cho nó có thể được khôi phục về trạng thái này về sau và
- Một giao diện trực tiếp để nhận trạng thái có thể mở ra chi tiết cài đặt và phá vỡ tính đóng gói của đối tượng.