Memento Design Pattern

Barabás Hunor

30432

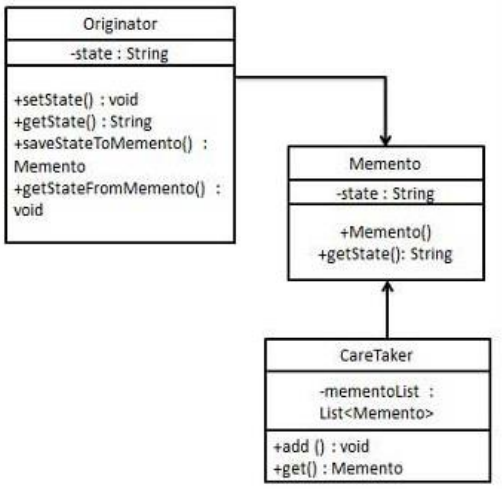
Intent

The Memento design pattern provides a way to save and restore the state of an object while respecting encapsulation, as opposed to the naive way of accessing an object's state directly, which would break encapsulation by exposing its inner structure.

Explanation

There are 3 main components:

- Memento - contains state of an object to be restored
- Originator - saves its state in and restores it from Memento objects
- Caretaker - keeps a list of all saved Mementos, but these are all opaque to it. When a previous state is to be restored, it passes the Memento back to the Originator.



In this case, the state is just a String, but it is usually more complex.

SOLID principle supported

The Memento DP respects and helps the Single Responsibility Principle. This is due to the fact that the responsibility of handling the state is separated from the responsibility of storing and restoring it.

Useful sources:

- https://www.javabrahman.com/design-patterns/memento-design-pattern-in-java/
- https://www.tutorialspoint.com/design_pattern/memento_pattern.htm
- http://w3sdesign.com/?gr=b06&ugr=proble

Related Patterns

While there is nothing quite what Memento does, it works well together with other design patterns. For example, the Observer Design Pattern can notify observers when a previous state was restored using Memento.

Common Situations of Use

Memento can be found in any application that has an undo/redo functionality. Text editors, paint-like programs and games with checkpoints can all utilize this design pattern.

Can be mistaken with…

It is easy to mistake Memento with State, due to the ambiguity of State's naming. While Memento deals with saving and restoring states, State is used to alter behavior when state changes.