**Memento Design Pattern**

Memento [design pattern](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) is one of the behavioral design pattern. Memento design pattern is used when we want to save the state of an object so that we can restore later on. Memento pattern is used to implement this in such a way that the saved state data of the object is not accessible outside of the object, this protects the integrity of saved state data.

Memento design pattern is implemented with two objects – Originator and Caretaker.

Originator is the object whose state needs to be saved and restored and it uses an [inner class](https://www.journaldev.com/996/java-inner-class) to save the state of Object. The inner class is called **Memento** and it’s private, so that it can’t be accessed from other objects.

Caretaker is the helper class that is responsible for storing and restoring the Originator’s state through Memento object. Since Memento is private to Originator, Caretaker can’t access it and it’s stored as an Object within the caretaker.

### Memento Design Pattern Java

One of the best real life example is the text editors where we can save it’s data anytime and use undo to restore it to previous saved state.

We will implement the same feature and provide a utility where we can write and save contents to a File anytime and we can restore it to last saved state. For simplicity, I will not use any IO operations to write data into file.

Memento pattern is simple and easy to implement, one of the thing needs to take care is that Memento class should be accessible only to the Originator object. Also in client application, we should use caretaker object for saving and restoring the originator state.

Also if Originator object has properties that are not [immutable](https://www.journaldev.com/129/how-to-create-immutable-class-in-java), we should use deep copy or cloning to avoid data integrity issue like I have used in above example. We can use [Serialization](https://www.journaldev.com/2452/serialization-in-java) to achieve memento pattern implementation that is more generic rather than Memento pattern where every object needs to have it’s own Memento class implementation.

One of the drawback is that if Originator object is very huge then Memento object size will also be huge and use a lot of memory.