Memento Overview

1. Keep a memento of an object’s state to return to that state
2. An object or system goes through changes
   1. Eg: bank account gets deposits and withdrawals
3. There are different ways of navigating those changes
4. One way is to record every change (Command) and teach a command to undo itself (Command pattern)
5. Another is to simply save snapshots of the system.
6. Memento: A token/handle representing the system state. Lets us roll back to the state when the token was generated. May or may not directly expose state information.

Memento

1. We don't want memento object to be changeable so they are typically immutable object (read only)
2. We can store the state of something in the memento object and write a code to load that state again.
3. If we want to keep a lot of stores in a lot of states, this is an issue for the memory so in this case you can use the command design pattern. Because the commander lists all the state changes, not the whole state itself.
4. Or you can compress the data(state) somehow in the memento for the memory issue.

Memento for Interop

1. Memento can be used to communicate between two languages.
2. Example: we have a C++ code and want to use that code in Java. In a sense C++ will return an int that may be the location in the memory of the function to java and Java can go to that location to trigger that function.

Summary

1. Mementos are used to roll back states arbitrarily.
2. A memento is simply a token/handle class with (typically) no functions of its own
3. A memento is not required to expose directly the state(s) to which it reverts the system
4. Can be used to implement undo/redo