# Overview:

* The Memento design will capture and externalize an object’s internal state so that the object can be restored to this state later, without violating encapsulation.
* The goal is to save the state of an object, o that in the future, we can go back to the specified state.
* Implemented in such a way that the saved state data of the object is not accessible outside of the object:
  + Protects the integrity of saved state data
  + Does not break encapsulation.
* You can use this pattern when you want to be able to return an object to one of its previous states:
  + If your user requests an “undo”

# Examples:

* In notepad, we use undo frequently by pressing ctrl+z.
* A classic example includes the state in a finite state machine.
* In real-world database programming, we often need to roll back a transaction operation.
* In a video game, a “save progress” option:
  + Players can store their game progress and at least recover most of their efforts when their character dies.
  + “Save progress” function would return a player to the last level he/she completed successfully.
* A graphical editor that supports connectivity between objects:
  + A user can connect two rectangles with a line.
  + The rectangles stay connected when the user moves either of them.
  + The editor ensures that the line stretches to maintain the connection.

# Why the memento?

* Sometimes it is necessary to record the internal state of an object
  + Required when implementing checkpoints and undo mechanisms.
    - Let users back out of tentative operations or recover from errors.
* You must save state information somewhere so that you can restore objects to their previous states.
* Objects normally encapsulate some or all of their state
  + Making it inaccessible to other objects and impossible to save externally.
  + Exposing this state would violate encapsulation
    - Can compromise the application’s reliability and extensibility.
* The memento addresses the above issues without violating encapsulation.

# Advantages and drawbacks:

* Simple and easy to implement.
* Provides easy-to-implement recovery capability.
* A drawback to using Memento is that saving and restoring state can be time consuming
  + Consider using Serialization to save a system’s state.

# Implementation

# Participants:

* **Memento:**
  + Stores the internal state of the Originator object.
  + May store as much or as little of the originator’s internal state as necessary
  + Protects against access by objects other than the originator
  + Has effectively two interfaces
    - Caretaker sees a narrow interface to the Memento.
      * Can only pass the memento to other objects
    - Originator sees a wide interface
      * Let us it access all the data necessary to restore itself to its previous state.
    - Only the originator that produced the memento would be permitted to access the memento’s internal state.
* **Originator:**
  + Creates a memento containing a snapshot of its current internal state.
  + Uses the memento to restore its internal state.
* **Caretaker:**
  + Responsible for the memento’s safekeeping.
  + Never operates on or examines the contents for a memento.

# Originator and Caretaker details:

* Originator is the object whose state needs to be saved and restored.
* Caretaker is the helper class that is responsible for storing and restoring the Originator’s state through the Memento object:
  + Keeps track of multiple mementos (maintaining save points)
  + Memento is stored as an Object within the caretaker.
* A caretaker requests a memento from an originator, holds it for a time, and passes it back to the originator.
* Sometimes the caretaker will not pass the memento back to the originator
  + The originator may never need to revert to an earlier state.
* Mementos are passive:
  + Only the originator that created a memento will assign or retrieve its state.

# Implementation consequences:

* The memento avoids exposing information that only an originator should manage but that must be stored nevertheless outside the originator
  + Shields other objects from potentially complex Originator internals, thereby preserving encapsulation boundaries.
* Having clients manage the state they ask for simplifies the Originator and keeps clients from having to notify originators when they are done.
* Mementos might incur considerable overhead if Originator must copy large amounts of information to store in the memento.
  + Unless encapsulating and restoring Originator state is cheap, the pattern might not be appropriate.

# Summary:

* The memento will store the internal states of the originator.
* The originator can access the internal states of the memento and it has the ability to restore into its earlier state.
  + Can also retrieve information from the memento.
* The caretaker takes care of the memento’s safekeeping or protection and it should not examine the contents of the memento.