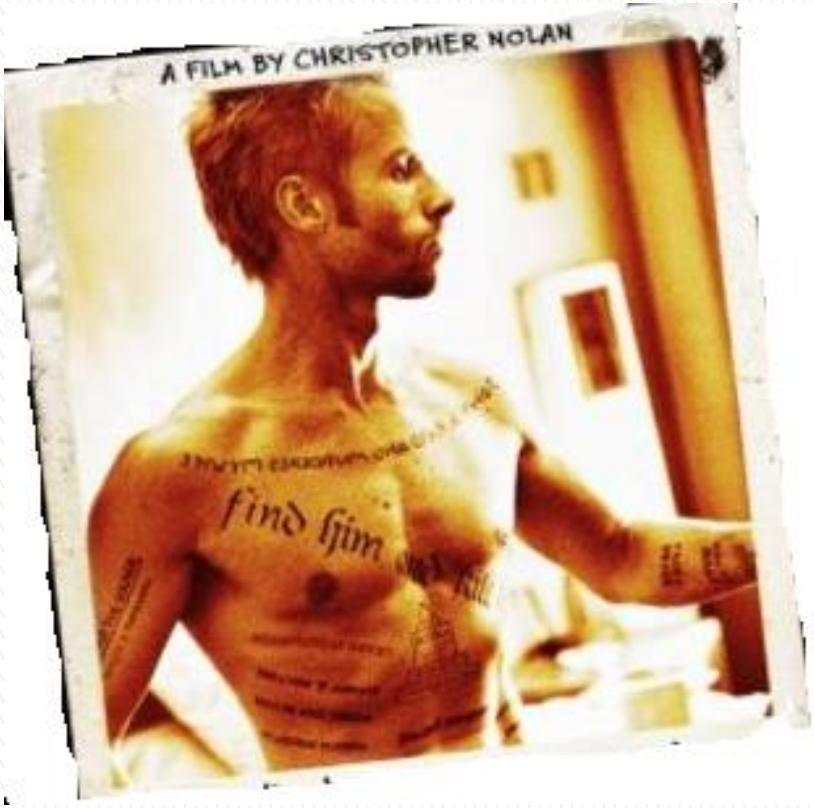
Memento Pattern

By Ravichandra Malipati, Gajendranath Puli,
and Jim Fawcett

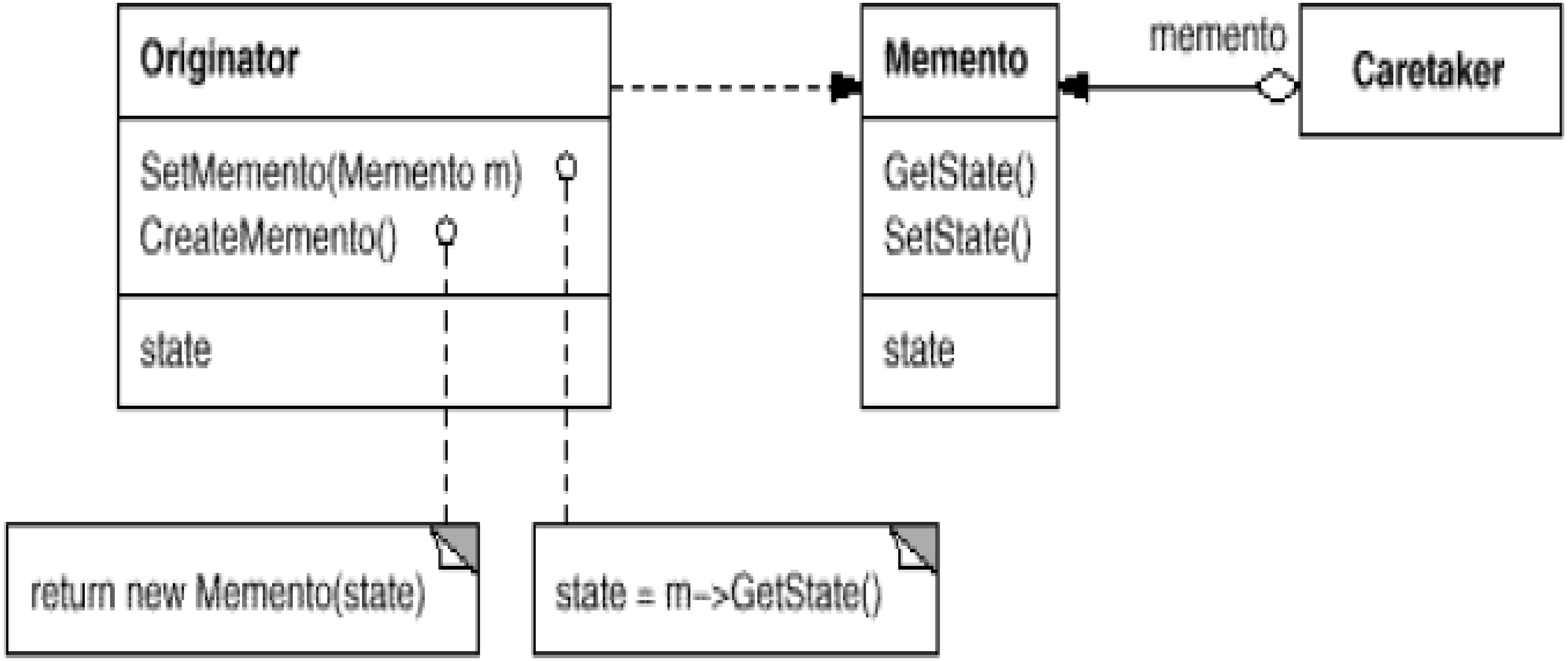
What does "Memento" mean?

- An Object or item that serves to remind you of a person or past event



Memento In terms of Software

Structure


Originator
SetMemento(Memento m)
CreateMemento()
state
Memento
GetState()
SetState()
state
memento
Caretaker
return new Memento(state)
state = m->GetState()

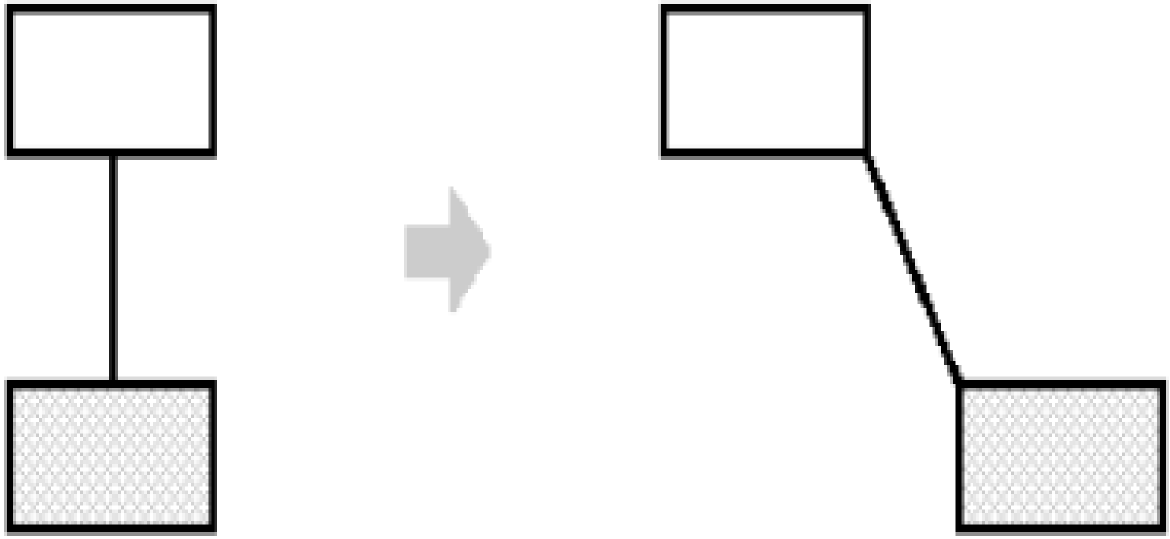
Intent

- Without violating encapsulation, capture and externalize an object’s internal state so that object can be restored to this state later.

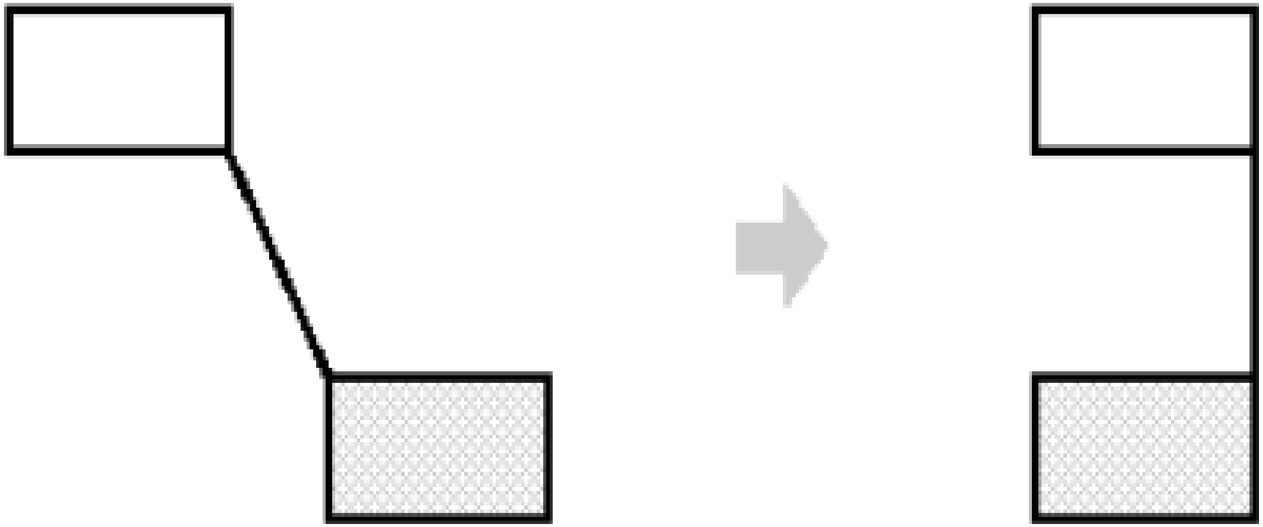
Motivation

- Save internal state of an object
- Implement checkpoints and undo mechanisms
- State of objects are partially or completely encapsulated
- To save state externally, violates encapsulation

Motivating Example



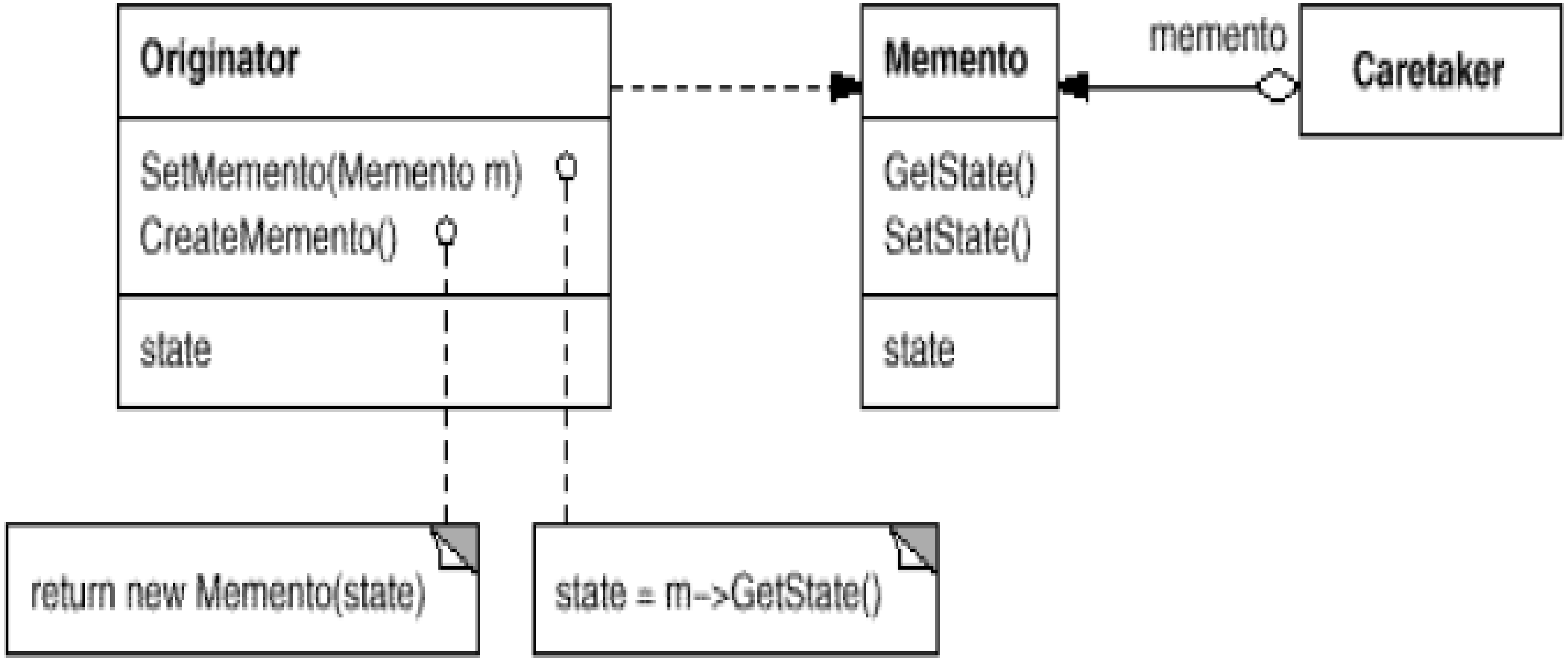
Motivating Example



Applicability

- When snapshot of (some portion of) an object’s state must be saved
- When you do not want to compromise encapsulation of the object
- You have an Application in which you want to go back to any one of a sequence of past states.

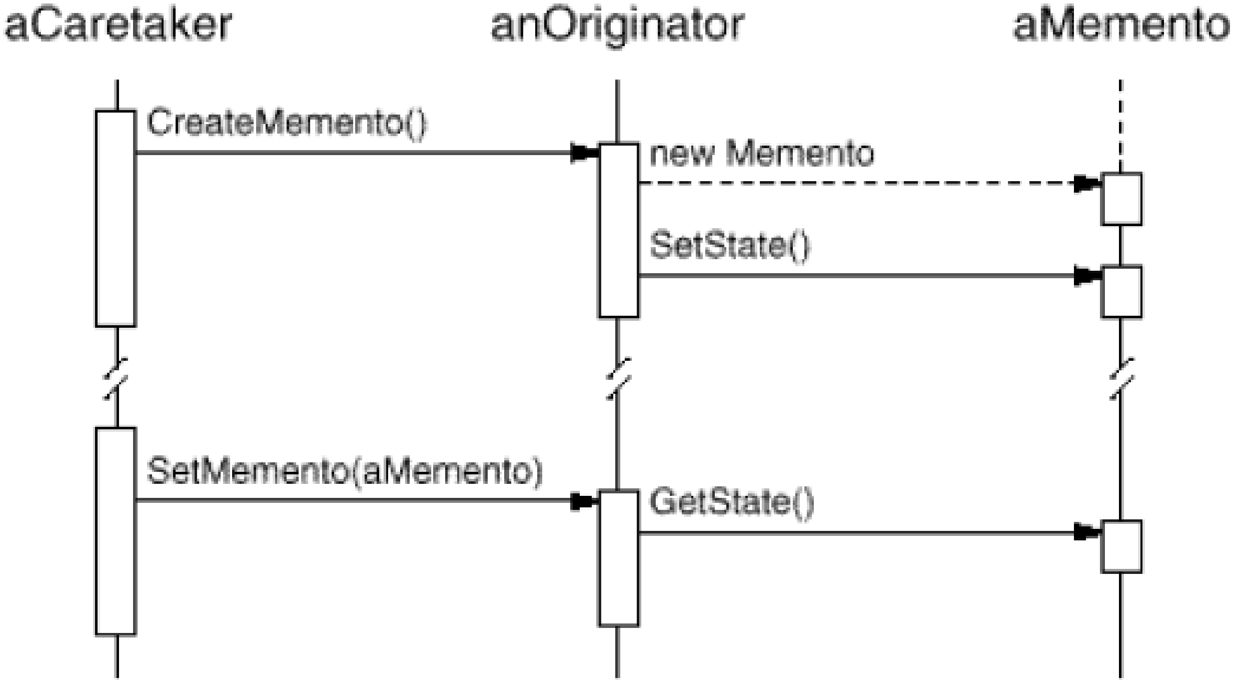
Structure


Originator
SetMemento(Memento m)
CreateMemento()
state
Memento
GetState()
SetState()
state
memento
Caretaker
return new Memento(state)
state = m->GetState()

Participants

- Memento– Object that holds one snapshot of originator state.
- Originator– the source of memento's state, usually the originator decides when to take and retrieve a memento.
- Caretaker– when asked, simply adds the state (Memento) to its collection of Mementos.

Collaborations



Consequences

- Preventing Violation of Encapsulation boundaries
- Simplifies Originator
- Using mementoes could be expensive
- Defining narrow and wide interfaces
  - Wide: Originator to and from Memento
  - Narrow: Caretaker to Memento
- Hidden costs in caring for memento

Implementation

- Probably need sharing relationship between Originator and Memento
  - friend in C++
  - internal declaration in C#
- Caretaker Data Structure
  - often holds stack of mementos
  - However, application may need something more complex then simple linear history.

Implementation Issues

- Language support for wide and narrow interface
- Storing Incremental changes

Code

- Skeleton
- Graph walk

Known Uses

- Word
- Git
- Memento-based iteration Interface
  - More than one state can work on same collection
  - Doesn’t break collection’s encapsulation

Related Patterns

- Command
- Iterator

References

- Design Patterns, Elements of Reusable Object-Oriented Software, Erich Gamma, et. al.

QUESTIONS?

THANK YOU