Memento

Linda Marshall

Department of Computer Science
University of Pretoria

19 August 2021

Name and Classification:

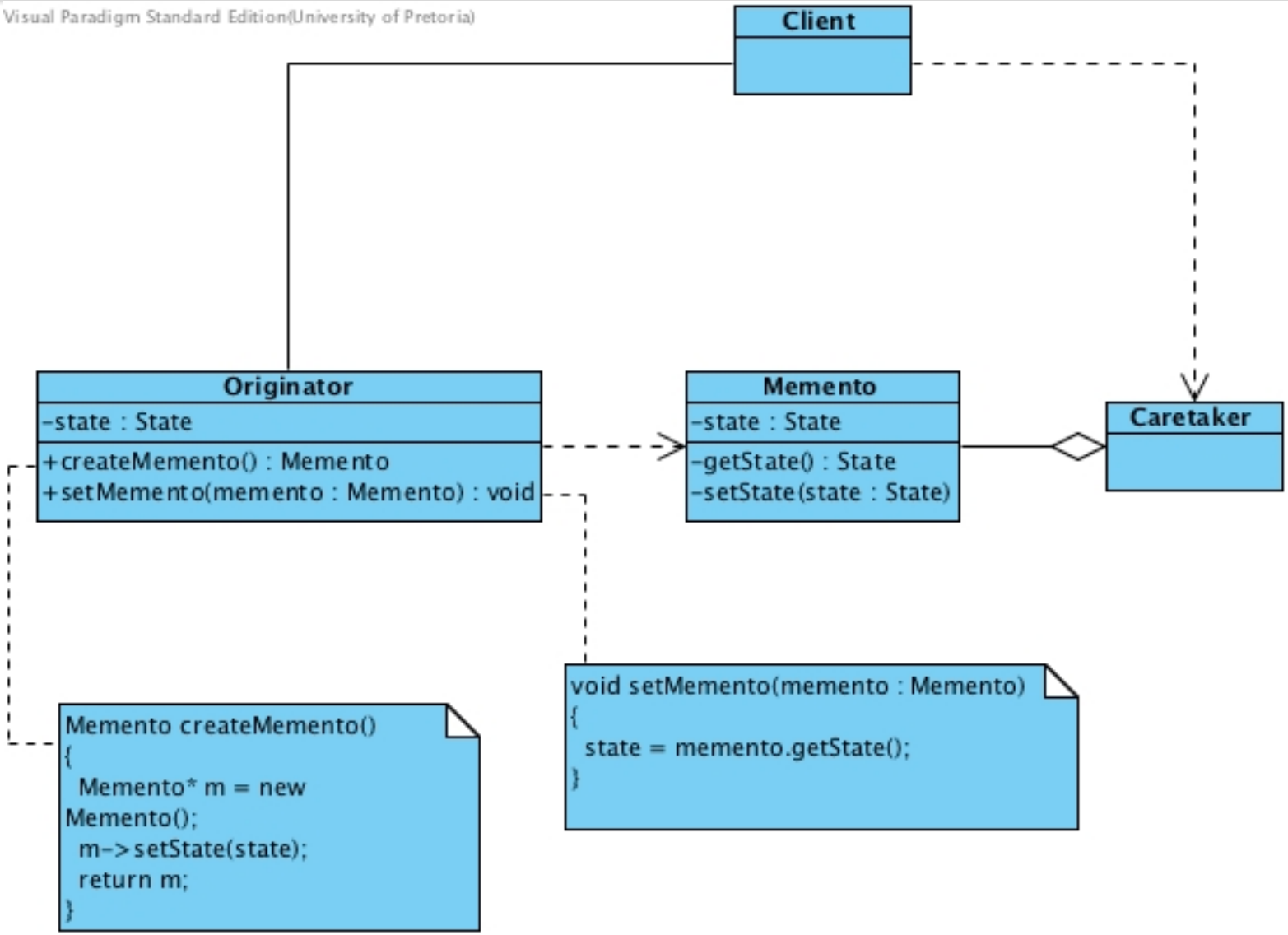
Memento (Behavioural)

Delegation (Object)

Intent:

"Without violating encapsulation, capture and externalise an object's internal state so that the object can be restored to this state later." GoF(283)

"Without violating encapsulation, capture and externalise an object's internal state so that the object can be restored to this state later." GoF(283)


Visual Paradigm Standard Edition(University of Pretoria)
Client
Originator
-state : State
+createMemento() : Memento
+setMemento(memento : Memento) : void
Memento
-state : State
-getState() : State
-setState(state : State)
Caretaker
Memento createMemento()
{
Memento* m = new
Memento();
m->setState(state);
return m;
}
void setMemento(memento : Memento)
{
state = memento.getState();
}

The **Memento**:

- Stores internal state of the Originator object.
  - The memento may store as much or as little of the originator's internal state as necessary, at its originator's discretion.

- Protects against access by objects other than the originator.
  - Caretaker sees a **narrow interface** to the Memento - it can only pass the memento to other objects.
  - Originator sees a **wide interface**, one that lets it access all the data necessary to restore itself to its previous state. Ideally, only the originator that produced the memento would be permitted to access the memento's internal state.

The **Originator**:

- Creates a memento containing a snapshot of its current internal state (`createMemento`).
- Uses the memento to restore its internal state (`setMemento`).

The **Caretaker**:

- is responsible for the safekeeping of the memento's state,
- never operates on or examines the contents of a memento.

Related Patterns

- **Command** (263): Commands can use mementos to maintain state for undoable operations.
- **Iterator** (289): Mementos can be used for iteration to maintain the state of the iterator.

What is a complex number?

It is a number that can be expressed in the form `a + b`i.

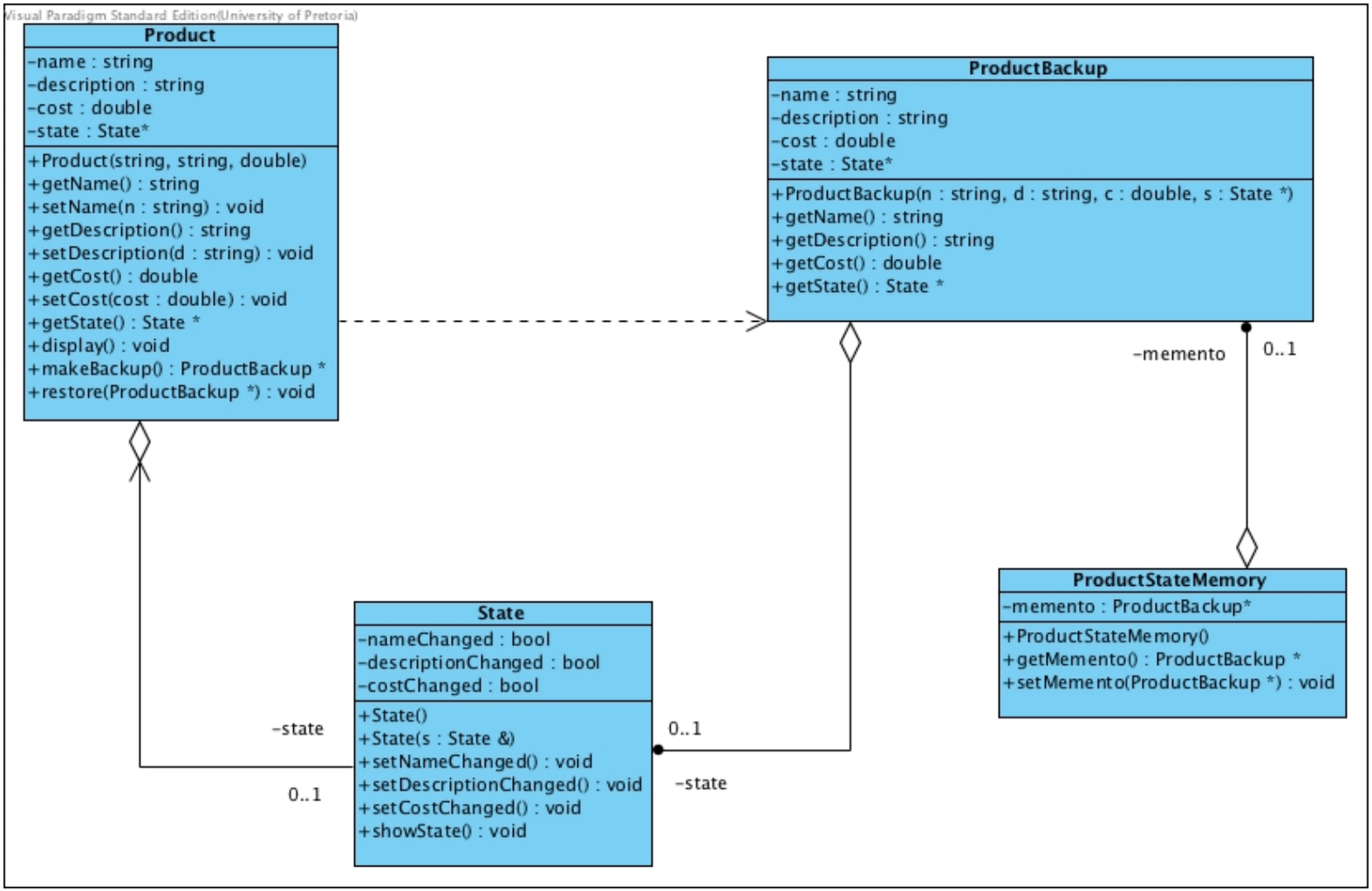
`a` and `b` are real numbers and i is the a solution to the equation $i^2 = -1$. `a` is called the *real-part* of the equation and `b` the *imaginary-part*. i is called the *imaginary number* – no real number satisfies the equation.

We want to "store" complex numbers, therefore we have the following classes:

```
class ComplexNumber;  // Originator
class Memento; // Memento
class Store; // Caretaker
```

What do you think the *state* of the complex number is?

```
class ComplexNumber;  // Originator
class Memento; // Memento
class Store; // Caretaker
class State; // The object to be stored by the Caretaker
```


Visual Paradigm Standard Edition(University of Pretoria)
Product
-name : string
-description : string
-cost : double
-state : State*
+Product(string, string, double)
+getName() : string
+setName(n : string) : void
+getDescription() : string
+setDescription(d : string) : void
+getCost() : double
+setCost(cost : double) : void
+getState() : State *
+display() : void
+makeBackup() : ProductBackup *
+restore(ProductBackup *) : void
ProductBackup
-name : string
-description : string
-cost : double
-state : State*
+ProductBackup(n : string, d : string, c : double, s : State *)
+getName() : string
+getDescription() : string
+getCost() : double
+getState() : State *
-memento
0..1
ProductStateMemory
-memento : ProductBackup*
+ProductStateMemory()
+getMemento() : ProductBackup *
+setMemento(ProductBackup *) : void
State
-nameChanged : bool
-descriptionChanged : bool
-costChanged : bool
+State()
+State(s : State &)
+setNameChanged() : void
+setDescriptionChanged() : void
+setCostChanged() : void
+showState() : void
-state
0..1
0..1
-state

Identify the Originator, Memento and Caretaker participants.