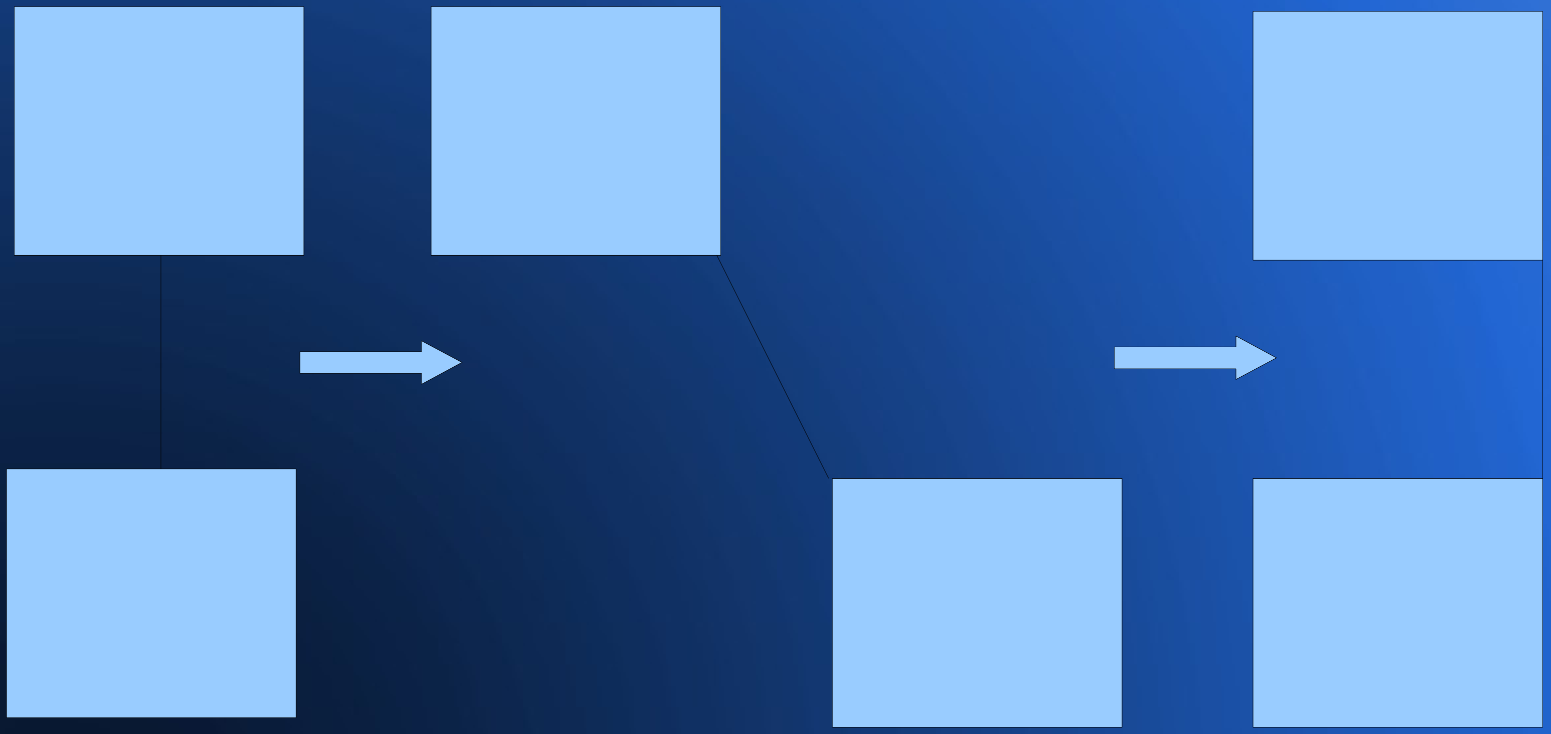
Memento Design Pattern

Tony Ankrapp

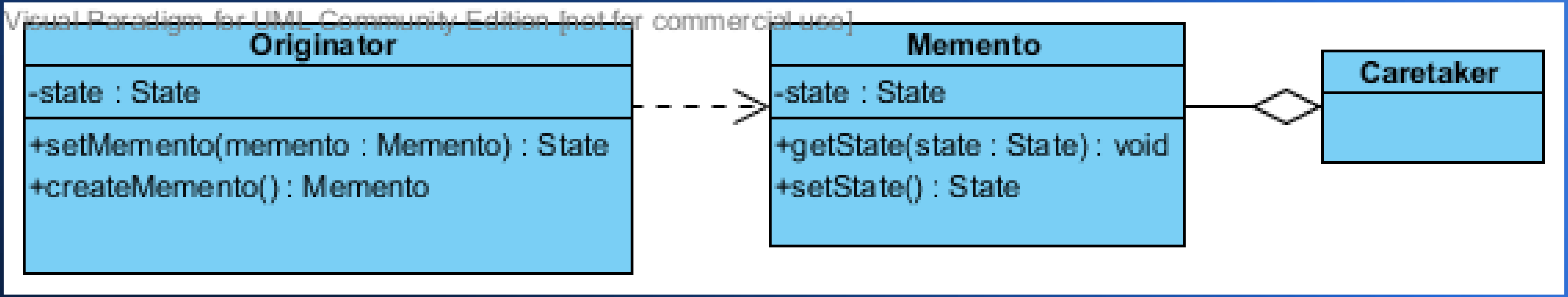
Memento Design Pattern

- Originator
  - object that needs storing (potentially large)
- Memento
  - object that stores (minimal required for undo)
- Caretaker
  - object that knows when to store

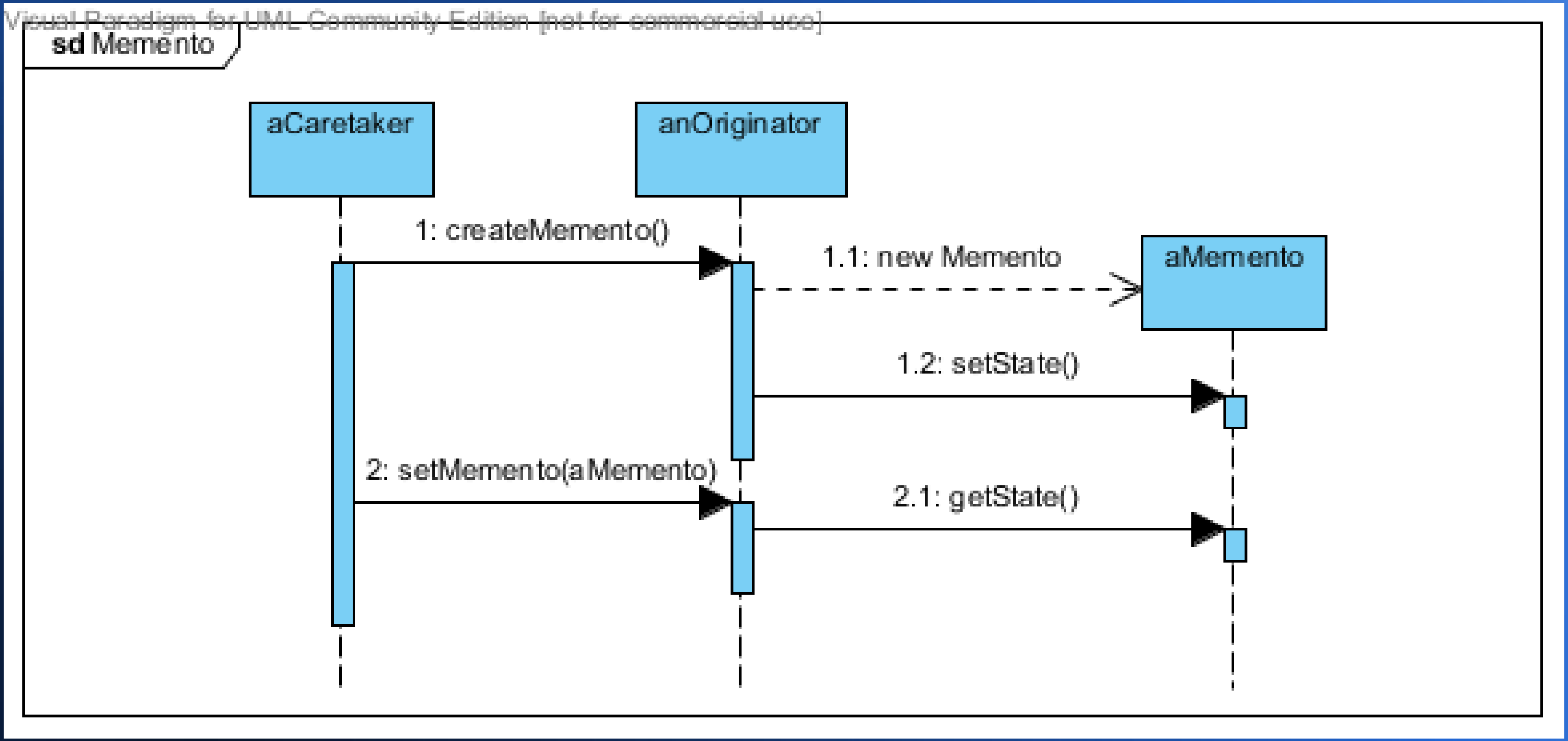
Graphical Example



UML Class Diagram


Visual Paradigm for UML Community Edition [not for commercial use]
Originator
-state : State
+setMemento(memento : Memento) : State
+createMemento() : Memento
Memento
-state : State
+getState(state : State) : void
+setState() : State
Caretaker

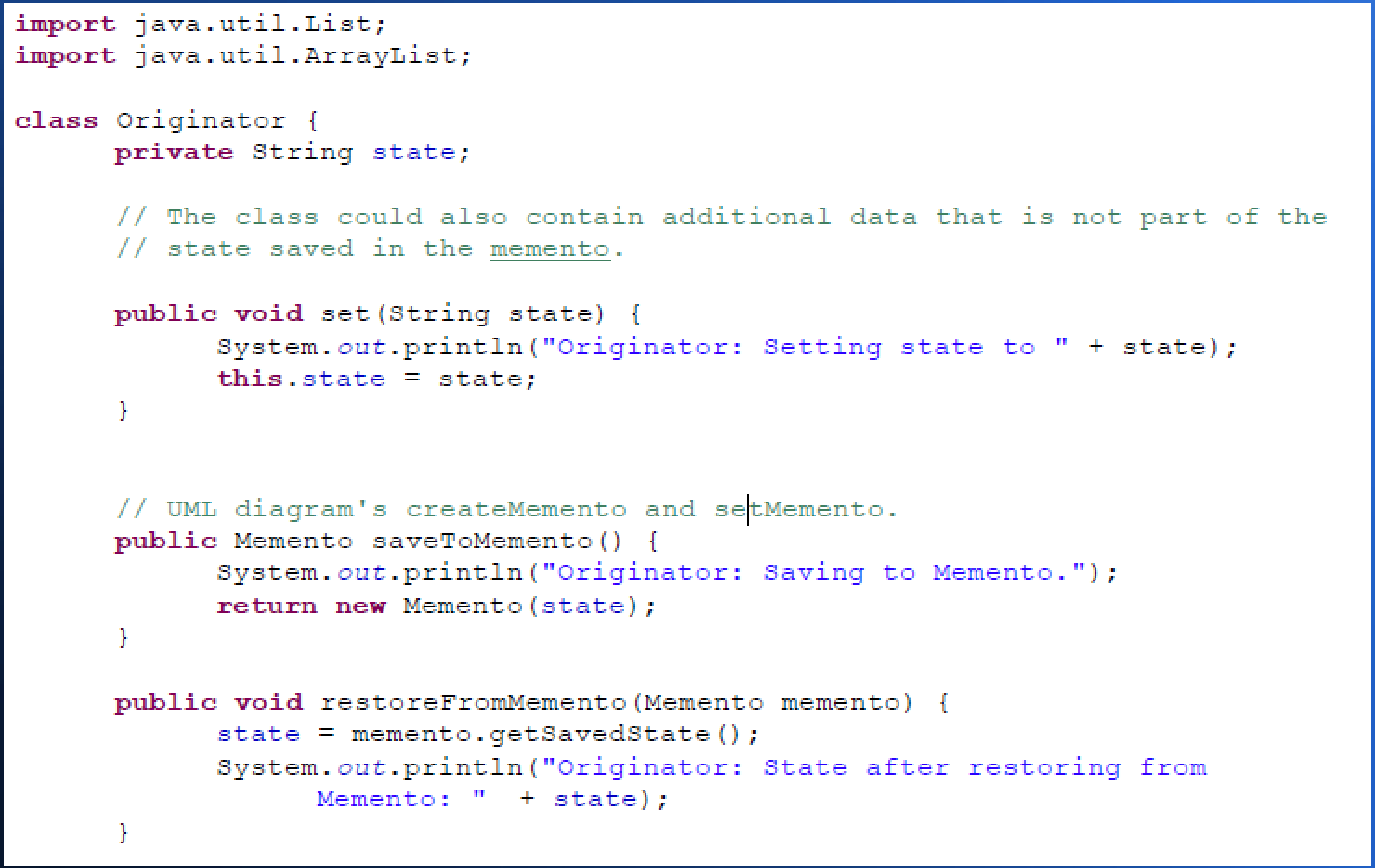
UML Sequence Diagram



Consequences

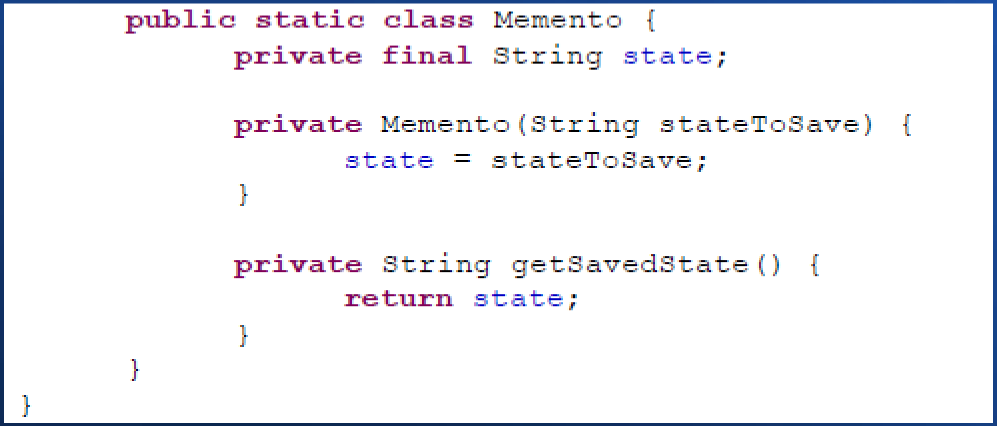
- + Preserves encapsulation boundaries
- + Simplifies Originator
- - Expensive
- - Interfaces
- - Caretaker cost

Example Memento

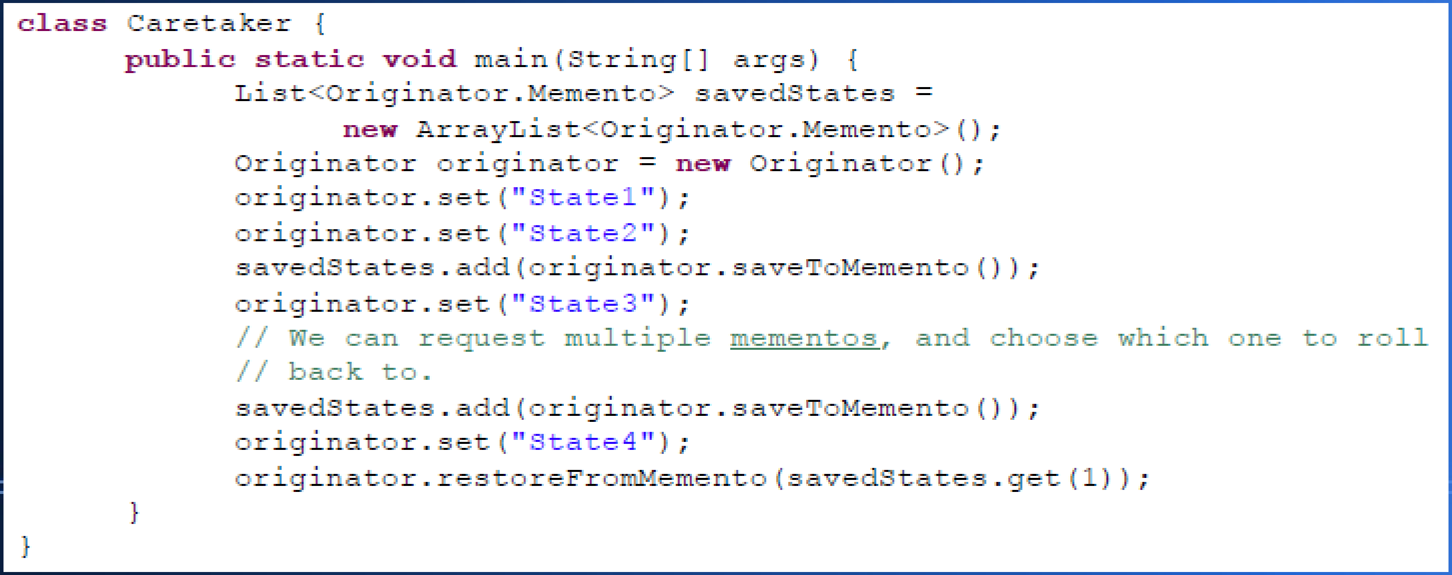


```java
import java.util.List;
import java.util.ArrayList;

class Originator {
        private String state;

        // The class could also contain additional data that is not part of the
        // state saved in the memento.

        public void set(String state) {
                System.out.println("Originator: Setting state to " + state);
                this.state = state;
        }


        // UML diagram's createMemento and setMemento.
        public Memento saveToMemento() {
                System.out.println("Originator: Saving to Memento.");
                return new Memento(state);
        }

        public void restoreFromMemento(Memento memento) {
                state = memento.getSavedState();
                System.out.println("Originator: State after restoring from
                        Memento: "  + state);
        }
```

Example Memento cont



```
    public static class Memento {
        private final String state;

        private Memento(String stateToSave) {
            state = stateToSave;
        }

        private String getSavedState() {
            return state;
        }
    }
}
```



```
class Caretaker {
    public static void main(String[] args) {
        List<Originator.Memento> savedStates =
            new ArrayList<Originator.Memento>();
        Originator originator = new Originator();
        originator.set("State1");
        originator.set("State2");
        savedStates.add(originator.saveToMemento());
        originator.set("State3");
        // We can request multiple mementos, and choose which one to roll
        // back to.
        savedStates.add(originator.saveToMemento());
        originator.set("State4");
        originator.restoreFromMemento(savedStates.get(1));
    }
}
```