

Supplementary Exercise on Memento Pattern

(a)	<pre> classDiagram class Caretaker { <<Caretaker>> } class Memento { -state -Originator orig +restore() +Memento(Originator o) } class Originator { ~state } Caretaker "0..*" --> Memento Memento "1" --> Originator </pre> <p>The diagram shows the Memento pattern structure. It consists of three classes: Caretaker, Memento, and Originator. The Caretaker class has a relationship with the Memento class marked with a multiplicity of 0..*. The Memento class contains attributes -state and -Originator orig, and methods +restore() and +Memento(Originator o). The Originator class contains an attribute ~state. There is a relationship between Memento and Originator marked with a multiplicity of 1.</p>
(b)	<p>Originator</p> <ul style="list-style-type: none"> - Object whose state we want to save <p>Memento</p> <ul style="list-style-type: none"> - Object that saves the state of originator <p>Caretaker</p> <ul style="list-style-type: none"> - Object that manages the timing of saving states, saves the memento, and uses memento to restore state of originator
(c)	<pre> public class WatchMemento { private String modelNo; private String name; private int price; private Watch w; public WatchMemento(Watch w) { this.w = w; modelNo = w.modelNo; name = w.name; price = w.price; } public void restore() { w.setModelNo(modelNo); w.setName(name); w.setPrice(price); } } </pre>

(d)	<pre>public class CareTaker { private Stack undoList; public CareTaker0 { undoList = new Stack0; } public void saveWatch(Watch w) { WatchMemento wm = new WatchMemento(w); undoList.push(wm); } public void undo0 { WatchMemento wm = (WatchMemento)undoList.pop0; wm.restore(); } }</pre>
(e)	<pre>public class TestMemento { public static void main(String[] args) { CareTaker ct = new CareTaker0; Watch w = new Watch("A123", "Solar Watch", 3000); ct.saveWatch(w); w.setName("Next Solar Watch"); ct.saveWatch(w); w.setPrice(3500); ct.undo0; System.out.println(w); } }</pre>