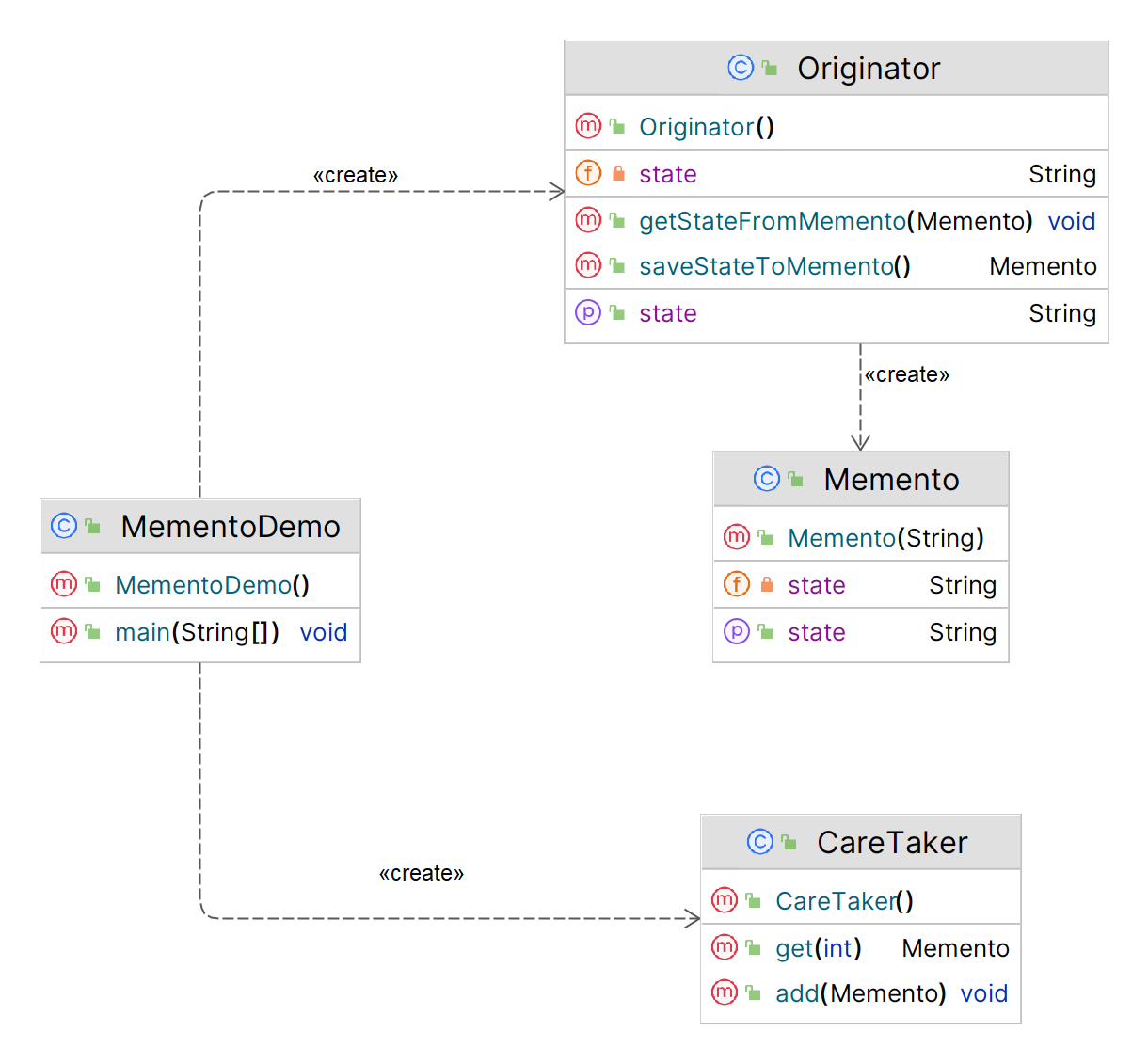
**Assignment 14: Memento Design Pattern**

**What is Memento Design Pattern?**

The **Memento** design pattern is a **behavioural** pattern that allows you to ***capture*** and ***save*** the state of an object **without violating encapsulation**.

**Structure (Class Diagram)**



**Implementation (Code)**

|  |
| --- |
| import java.util.\*;  ***// Memento class***  public class Memento {  private String state;  public Memento(String state) {  this.state = state;  }  public String getState() {  return state;  } }  ***// Originator Class***  public class Originator {  private String state;   public String getState() {  return state;  }  public void setState(String state) {  this.state = state;  }  public Memento saveStateToMemento() {  return new Memento(state);  }  public void getStateFromMemento(Memento memento) {  state = memento.getState();  } }  ***// Caretaker class*** public class CareTaker {  private List<Memento> mementoList = new ArrayList<Memento>();  public void add(Memento state) {  mementoList.add(state);  }  public Memento get (int index) {  return mementoList.get(index);  } }  **// *Main - Demo***  public class MementoDemo {  public static void main(String[] args) {  Originator originator = new Originator();  CareTaker careTaker = new CareTaker();   originator.setState("State 1");  careTaker.add(originator.saveStateToMemento()); *// Saved state 1 at index 0* originator.setState("State 2");  careTaker.add(originator.saveStateToMemento()); *// Saved state 2 at index 1* originator.setState("State 3");  careTaker.add(originator.saveStateToMemento()); *// Saved state 3 at index 2* originator.setState("State 4");  System.*out*.println("Current State: " + originator.getState());   originator.getStateFromMemento(careTaker.get(0));  System.*out*.println("First saved state: " + originator.getState());   originator.getStateFromMemento(careTaker.get(1));  System.*out*.println("Second saved state: " + originator.getState());   originator.getStateFromMemento(careTaker.get(2));  System.*out*.println("Second saved state: " + originator.getState());  } } |

**Applicability**

1. Use the **Memento** pattern when you want to produce **snapshots** of the object’s state to be able to restore a previous state of the object.
2. Use the pattern when direct access to the object’s fields/getters/setters violates its **encapsulation**.