PANDIT DEENDAYAL ENERGY UNIVERSITY A RESERVOIR OF KNOWLEDGE	**Course Name: Design Patterns/Thinking LAB**	**EXPERIMENT NO. 13**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
Submitted by: Rhythm Shah **Roll no:** 22BCP071			

Objective: To familiarize students with standard Behavioral design patterns.
Experiment: Explain the Memento design pattern and write a program using any object-oriented programming language to demonstrate the working of Memento design pattern.

Theory: -

Memento is a behavioural design pattern that lets you save and restore the previous state of an object without revealing the details of its implementation.

Components of Memento Design Pattern:

1. Originator
This component is responsible for creating and managing the state of an object. It has methods to set and get the object's state, and it can create Memento objects to store its state. The Originator communicates directly with the Memento to create snapshots of its state and to restore its state from a snapshot.

2. Memento
The Memento is an object that stores the state of the Originator at a particular point in time. It only provides a way to retrieve the state, without allowing direct modification. This ensures that the state remains

3. Caretaker
The Caretaker is responsible for keeping track of Memento objects. It doesn’t know the details of the state stored in the Memento but can request Mementos from the Originator to save or restore the object’s state.

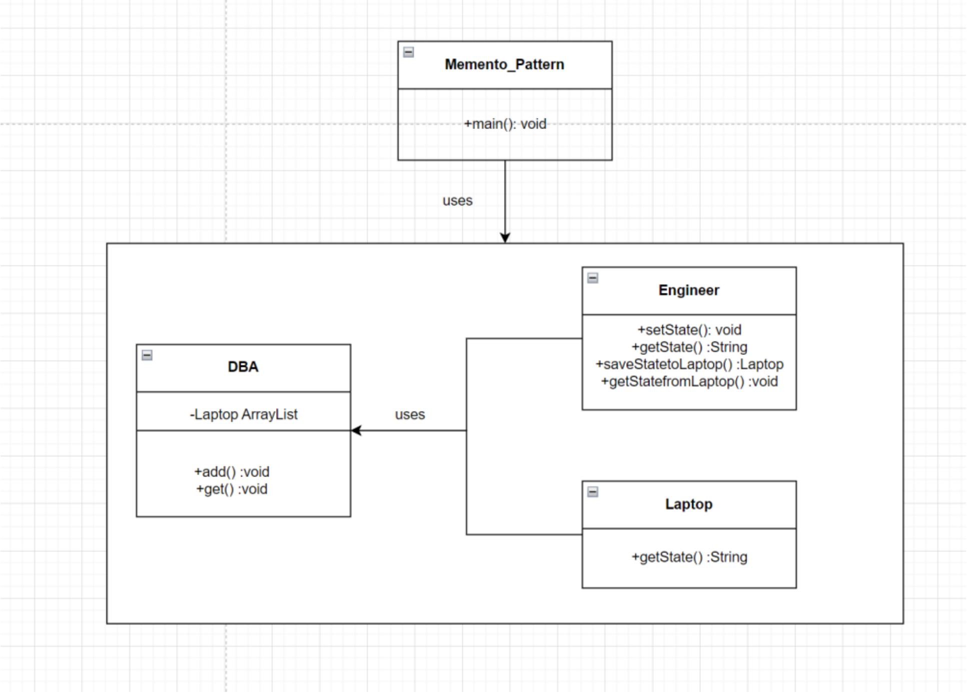
4. Client
Typically represented as the part of the application or system that interacts with the Originator and Caretaker to achieve specific functionality. The client initiates requests to save or restore the state of the Originator through the Caretaker.

Implementation: -

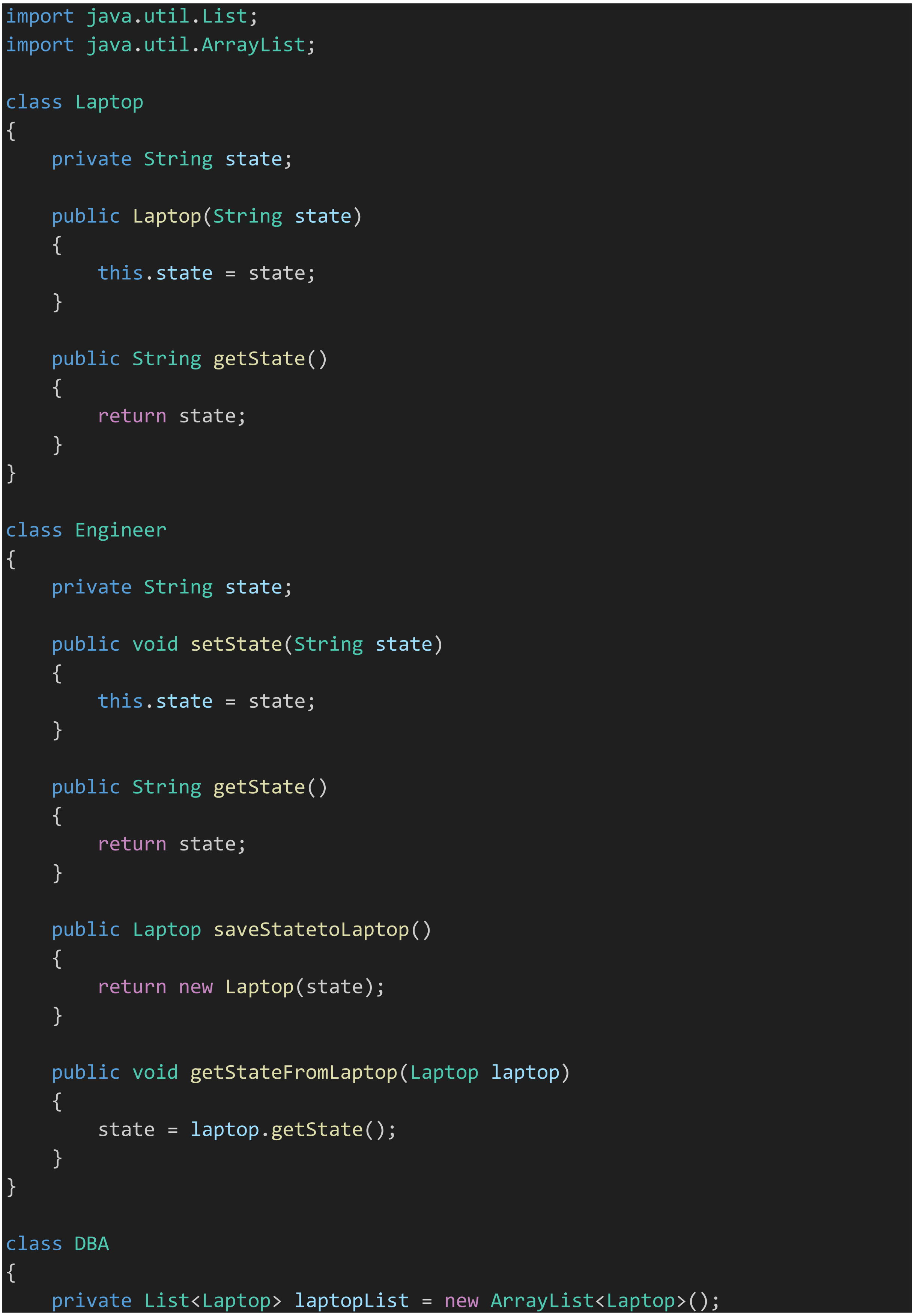
Here I have implemented memento by taking example of Laptop. The classes made are of Laptop, Engineer, DBA and one main class which is creating objects. Laptop is the originator; Engineer is the caretaker and memento is the DBA which is database stored in Laptop.

Getter Setters methods are applied as parameters are defined private and an array is created of Laptops through Array List.

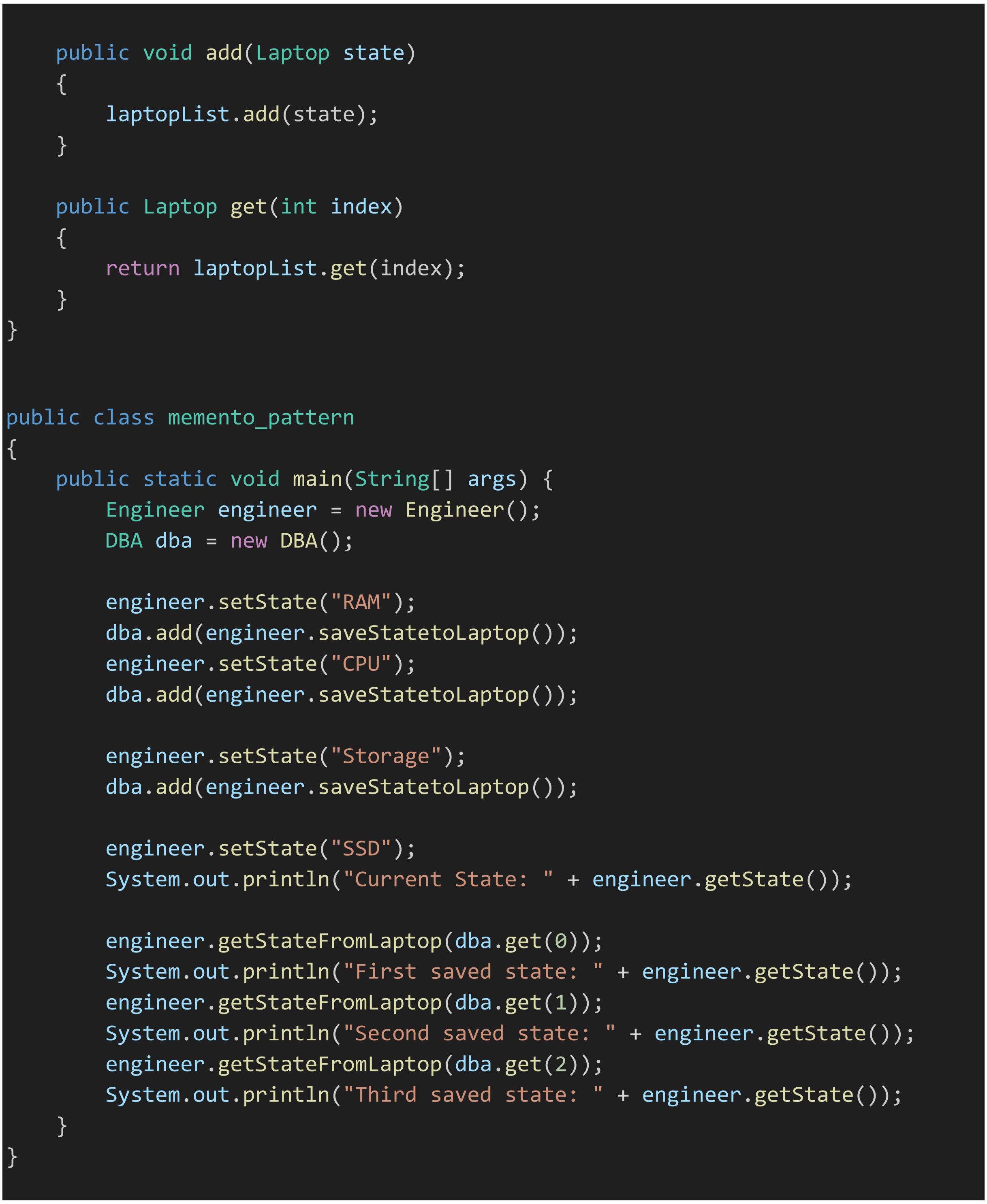
UML Diagram: -



Code: -

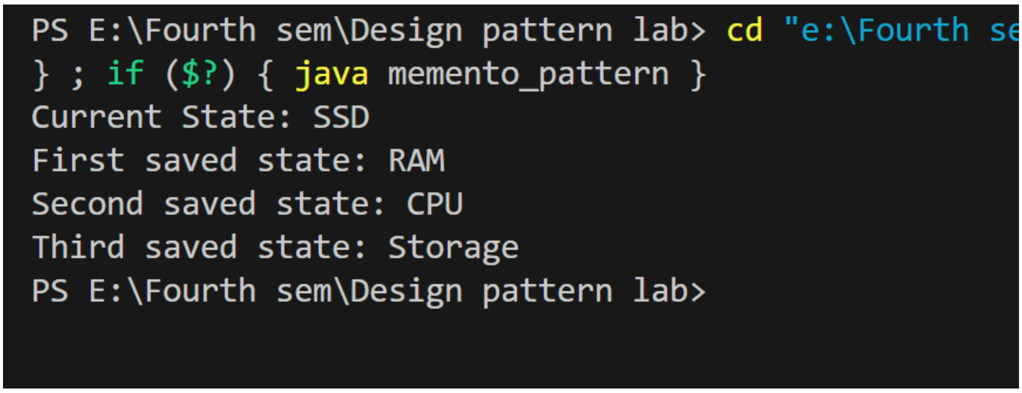


```
import java.util.List;
import java.util.ArrayList;

class Laptop
{
    private String state;

    public Laptop(String state)
    {
        this.state = state;
    }

    public String getState()
    {
        return state;
    }
}

class Engineer
{
    private String state;

    public void setState(String state)
    {
        this.state = state;
    }

    public String getState()
    {
        return state;
    }

    public Laptop saveStatetoLaptop()
    {
        return new Laptop(state);
    }

    public void getStateFromLaptop(Laptop laptop)
    {
        state = laptop.getState();
    }
}

class DBA
{
    private List<Laptop> laptopList = new ArrayList<Laptop>();
```



```java
    public void add(Laptop state)
    {
        laptopList.add(state);
    }

    public Laptop get(int index)
    {
        return laptopList.get(index);
    }
}


public class memento_pattern
{
    public static void main(String[] args) {
        Engineer engineer = new Engineer();
        DBA dba = new DBA();

        engineer.setState("RAM");
        dba.add(engineer.saveStatetoLaptop());
        engineer.setState("CPU");
        dba.add(engineer.saveStatetoLaptop());

        engineer.setState("Storage");
        dba.add(engineer.saveStatetoLaptop());

        engineer.setState("SSD");
        System.out.println("Current State: " + engineer.getState());

        engineer.getStateFromLaptop(dba.get(0));
        System.out.println("First saved state: " + engineer.getState());
        engineer.getStateFromLaptop(dba.get(1));
        System.out.println("Second saved state: " + engineer.getState());
        engineer.getStateFromLaptop(dba.get(2));
        System.out.println("Third saved state: " + engineer.getState());
    }
}
```

Output:-



```
PS E:\Fourth sem\Design pattern lab> cd "e:\Fourth se
} ; if ($?) { java memento_pattern }
Current State: SSD
First saved state: RAM
Second saved state: CPU
Third saved state: Storage
PS E:\Fourth sem\Design pattern lab>
```