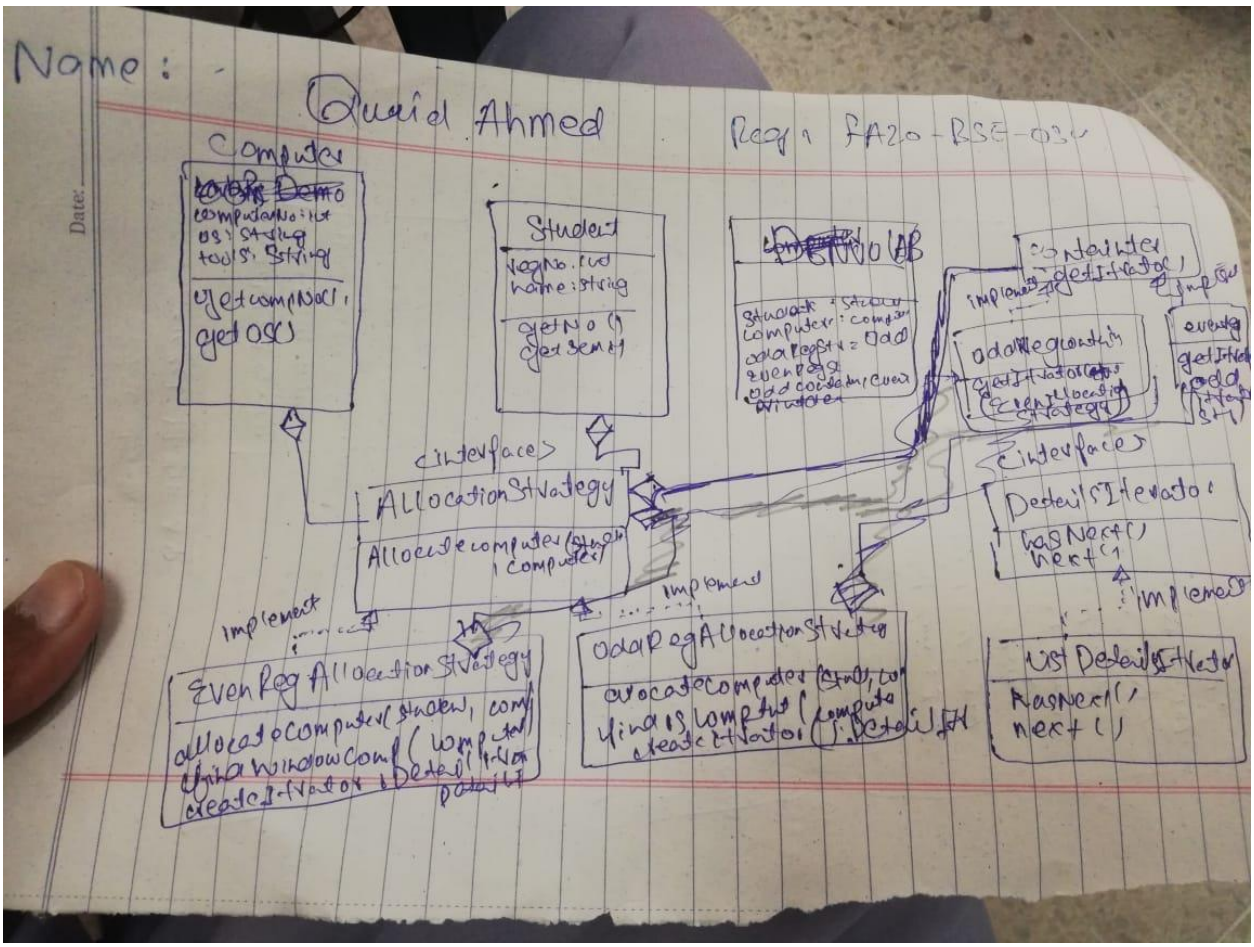
COMSATS University Islamabad,
Abbottabad Campus
Department of Computer Science

Mid-Term Lab – FALL 2023

Class: **BSE 7A** Date: 25 Oct 2023

Subject:Design pattern Instructor: Mukhtiar Zamin

Name:**Quaid Ahmed** Registration #**FA20-BSE-034**




```java
package Midterm_Exam.source_code;

import java.util.List;
import java.util.Map;

interface AllocationStrategy {
    void allocateComputers(List<Student> students, Map<Integer, Computer>
computers);
}
```

```java
package Midterm_Exam.source_code;

// Computer class representing individual computers
class Computer {
    private int computerNo;
    private String os;
    private String tools;

    public Computer(int computerNo, String os, String tools) {
        this.computerNo = computerNo;
        this.os = os;
        this.tools = tools;
    }

    public int getComputerNo() {
        return computerNo;
    }

    public String getOs() {
        return os;
    }

    public String getTools() {
        return tools;
    }

    public void setTools(String tools) {
        this.tools = tools;
    }

}
package Midterm_Exam.source_code;

interface Container {
    DetailsIterator getIterator();
}
package Midterm_Exam.source_code;

interface DetailsIterator {
    boolean hasNext();
    String next();
}
package Midterm_Exam.source_code;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

class EvenRegAllocationStrategy implements AllocationStrategy {
    private List<String> allocatedDetails = new ArrayList<>();

    @Override
    public void allocateComputers(List<Student> students, Map<Integer,
Computer> computers) {
        for (Student student : students) {
            if (student.getRegNo() % 2 == 0) {
                Computer windowsComputer = findWindowsComputer(computers);
```

```
                allocatedDetails.add(student.getName() + " allocated Windows
computer " + windowsComputer.getComputerNo());
            }
        }
    }

    private Computer findWindowsComputer(Map<Integer, Computer> computers) {
        for (Computer computer : computers.values()) {
            if (computer.getOs().equalsIgnoreCase("Windows")) {
                return computer;
            }
        }
        throw new IllegalStateException("No Windows computers available");
    }

    public DetailsIterator createIterator() {
        return new ListDetailsIterator(allocatedDetails);
    }
}
package Midterm_Exam.source_code;

class EvenRegContainer implements Container {
    private EvenRegAllocationStrategy strategy;

    public EvenRegContainer(EvenRegAllocationStrategy strategy) {
        this.strategy = strategy;
    }

    @Override
    public DetailsIterator getIterator() {
        return strategy.createIterator();
    }
}
package Midterm_Exam.source_code;

import java.util.List;
import java.util.NoSuchElementException;

class ListDetailsIterator implements DetailsIterator {
    private List<String> details;
    private int position = 0;

    public ListDetailsIterator(List<String> details) {
        this.details = details;
    }

    @Override
    public boolean hasNext() {
        return position < details.size();
    }

    @Override
    public String next() {
        if (!hasNext()) {
            throw new NoSuchElementException();
        }
        return details.get(position++);
```

```
    }
}
package Midterm_Exam.source_code;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

class OddRegAllocationStrategy implements AllocationStrategy {
    private final List<String> allocatedDetails = new ArrayList<>();

    @Override
    public void allocateComputers(List<Student> students, Map<Integer,
Computer> computers) {
        for (Student student : students) {
            if (student.getRegNo() % 2 != 0) {
                Computer linuxComputer = findLinuxComputer(computers);
                allocatedDetails.add(student.getName() + " allocated Linux
computer " + linuxComputer.getComputerNo());
            }
        }
    }

    private Computer findLinuxComputer(Map<Integer, Computer> computers) {
        for (Computer computer : computers.values()) {
            if (computer.getOs().equalsIgnoreCase("Linux")) {
                return computer;
            }
        }
        throw new IllegalStateException("No Linux computers available");
    }

    public DetailsIterator createIterator() {
        return new ListDetailsIterator(allocatedDetails);
    }
}
package Midterm_Exam.source_code;

class OddRegContainer implements Container {
    private OddRegAllocationStrategy strategy;

    public OddRegContainer(OddRegAllocationStrategy strategy) {
        this.strategy = strategy;
    }

    @Override
    public DetailsIterator getIterator() {
        return strategy.createIterator();
    }
}
package Midterm_Exam.source_code;

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

```
public class Main {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student(1, "Alice", "Computer Science", 2));
        students.add(new Student(2, "Bob", "Electrical Engineering", 3));
        students.add(new Student(3, "Charlie", "Mechanical Engineering", 1));
        students.add(new Student(4, "Diana", "Physics", 2));

        Map<Integer, Computer> computers = new HashMap<>();
        computers.put(1, new Computer(1, "Linux", "Programming Tools"));
        computers.put(2, new Computer(2, "Windows", "Office Suite"));
        computers.put(3, new Computer(3, "Linux", "Engineering Software"));
        computers.put(4, new Computer(4, "Windows", "Data Analysis Tools"));


        OddRegAllocationStrategy oddRegStrategy = new
OddRegAllocationStrategy();
        EvenRegAllocationStrategy evenRegStrategy = new
EvenRegAllocationStrategy();


        oddRegStrategy.allocateComputers(students, computers);
        evenRegStrategy.allocateComputers(students, computers);


        Container oddContainer = new OddRegContainer(oddRegStrategy);
        Container evenContainer = new EvenRegContainer(evenRegStrategy);

        System.out.println("Details for Odd Registration:");
        printDetails(oddContainer.getIterator());

        System.out.println("\nDetails for Even Registration:");
        printDetails(evenContainer.getIterator());
    }

    private static void printDetails(DetailsIterator iterator) {
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```