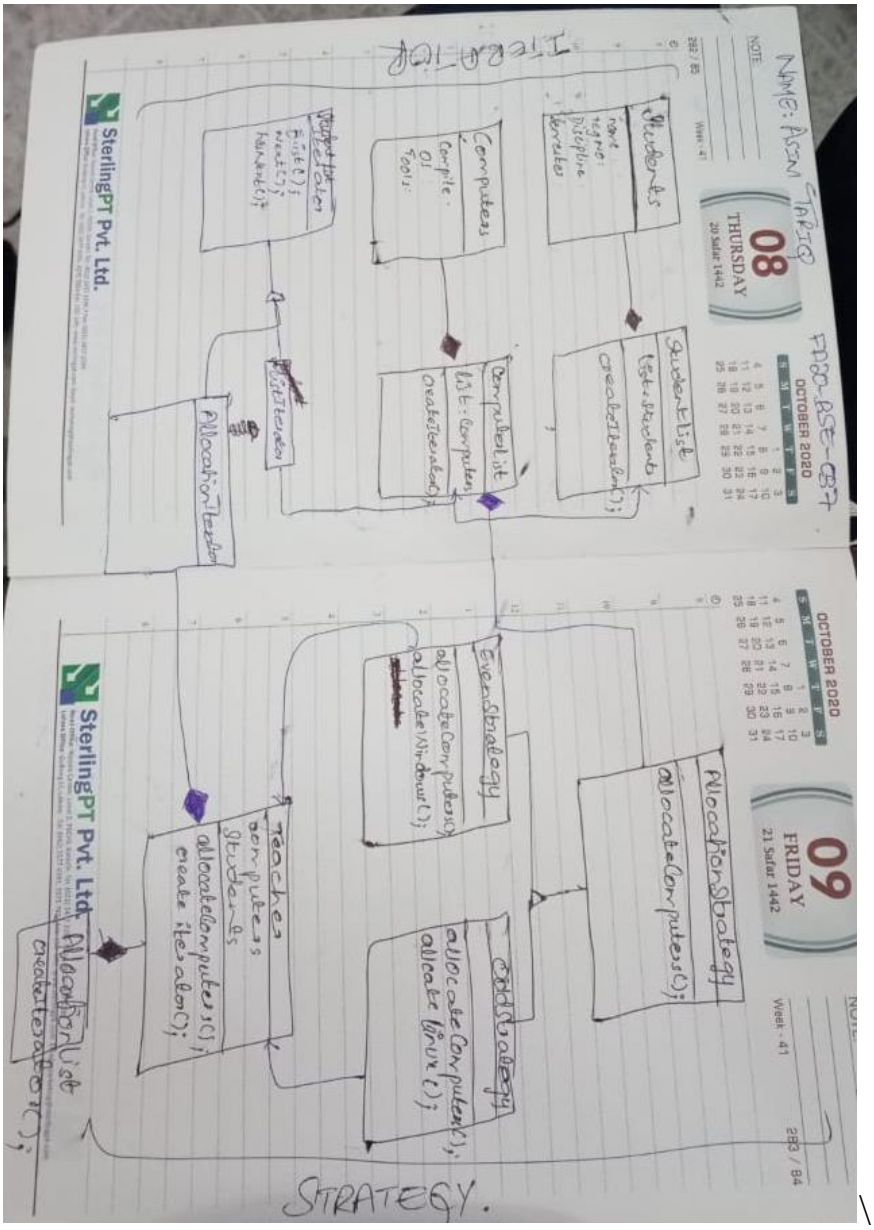
LAB MID

DESIGN PATTERNS

NAME: ASIM TARIQ

REG.NO: FA20-BSE-037

CLASS DIAGRAM:



CODE:

```
public class Main {
    public static void main(String[] args) {
        // Instantiate a Classroom
        Classroom classroom = new Classroom();

        // Add computers to the classroom
        classroom.addComputer("C001", "Linux", "Eclipse");
        classroom.addComputer("C002", "Windows", "Visual Studio");
        classroom.addComputer("C003", "Linux", "NetBeans");

        // Add students to the classroom
        classroom.addStudent("John Doe", "S001", 3);
        classroom.addStudent("Jane Smith", "S002", 4);
        classroom.addStudent("Bob Johnson", "S003", 2);

        // Allocate computers using a specific strategy (Linux in this case)
        classroom.allocateComputers(new
LinuxStudentAllocationStrategy());

        // Create an iterator and iterate over the allocation list
        AllocationIterator iterator = classroom.createIterator();
        while (iterator.hasNext()) {
```

```
            Object currentItem = iterator.next();
            if (currentItem instanceof Student) {
                Student student = (Student) currentItem;
                System.out.println("Student: " + student.getName() + " -
Computer: "
                        + student.getAllocatedComputer().getCNumber());
            } else if (currentItem instanceof Computer) {
                Computer computer = (Computer) currentItem;
                System.out.println("Computer: " + computer.getCNumber() +
" - Allocated to: "
                        + (computer.getAllocatedStudent() != null ?
computer.getAllocatedStudent().getName() : "None"));
            }
        }
    }
}
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package midterm;


/**
```

```
 *
 * @author LAPTOP HOUSE
 */
public class MIDTerm {
    public static void main(String[] args) {
        // Instantiate a Classroom
        Teacher classroom = new Classroom();

        // Add computers to the classroom
        classroom.addComputer("C001", "Linux", "Eclipse");
        classroom.addComputer("C002", "Windows", "Visual Studio");
        classroom.addComputer("C003", "Linux", "NetBeans");

        // Add students to the classroom
        classroom.addStudent("John Doe", "S001", 3);
        classroom.addStudent("Jane Smith", "S002", 4);
        classroom.addStudent("Bob Johnson", "S003", 2);

        // Allocate computers using a specific strategy (Linux in this case)
        classroom.allocateComputers(new OddStrategy());

        // Create an iterator and iterate over the allocation list
        AllocationIterator iterator = classroom.createIterator();
```

```
        while (iterator.hasNext()) {
            Object currentItem = iterator.next();
            if (currentItem instanceof Student) {
                Student student = (Student) currentItem;
                System.out.println("Student: " + student.getName() + " -
Computer: "
                        + student.getAllocatedComputer().getCNumber());
            } else if (currentItem instanceof Computers) {
                computer = (Computers) currentItem;
                System.out.println("Computer: " + computer.getCNumber() +
" - Allocated to: "
                        + (computer.getAllocatedStudent() != null ?
computer.getAllocatedStudent().getName() : "None"));
            }
        }
    }
}
```