# DESIGN PATTERNS LAB MID

PREPARED FOR,

SIR MUKHTIAR ZAMIN

PREPARED BY,

NABIL, FA20-BSE-009

```
package LabMid;

public class Computer {
    private String computerId;
    private String OS;
    private String tools;

    public Computer(String computerId, String os, String tools) {
        this.computerId = computerId;
        OS = os;
        this.tools = tools;
    }

    public String getOS() {
        return OS;
    }

    public void setOS(String OS) {
        this.OS = OS;
    }
}
```

```
package LabMid;

public interface Iterator {
    Boolean hasNext();
    Student current();
    void next();
```

```
}

package LabMid;

import java.util.ArrayList;

public class LabAllocator {
    private OSStrategy osStrategy;



    public LabAllocator(OSStrategy osStrategy) {
        this.osStrategy = osStrategy;
    }

    public ArrayList allocate() {
        return osStrategy.allocate();
    }
}


package LabMid;

import java.util.ArrayList;

public class LinuxList {
   private ArrayList allocatedStudents;



    public LinuxList(ArrayList allocatedStudents) {
        this.allocatedStudents = allocatedStudents;
    }

    public ArrayList getAllocatedStudents() {
        return allocatedStudents;
    }
}



package LabMid;

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

public class LinuxStrategy implements OSStrategy {
    private StudentArrayList students;
    private ArrayList linuxComputers;

    private ArrayList<Student> linuxList = new ArrayList<>();

    LinuxStrategy(StudentArrayList students, Map<String, ArrayList<Computer>> computersDictionary) {
        this.students = students;
        this.linuxComputers = computersDictionary.get("linux");
    }

    @Override
    public ArrayList allocate() {
        int i = 0;
        Iterator iterator = students.createIterator();
        while(iterator.hasNext()) {
            if (i == linuxComputers.size())
                break;
            var student = iterator.current();
            if (student.getReg() % 2 != 0) {
                student.setAllocatedComputer((Computer) linuxComputers.get(i));
                linuxList.add(student);
                i++;
            }

            iterator.next();
        }
        return linuxList;
    }

}



package LabMid;

import java.util.ArrayList;

public interface OSStrategy {
    public ArrayList allocate();
}



package LabMid;
```

```
public class Student {

   private String name;

   private int reg;

   private String description;

   private int semester;

   private Computer allocatedComputer;


   public Student(String name, int reg, String description, int semester) {

      this.name = name;

      this.reg = reg;

      this.description = description;

      this.semester = semester;

   }


   public int getSemester() {

      return semester;

   }


   public void setSemester(int semester) {

      this.semester = semester;

   }


   public String getName() {

      return name;

   }


   public int getReg() {

      return reg;

   }
```

```
    public void setAllocatedComputer(Computer allocatedComputer) {
        this.allocatedComputer = allocatedComputer;
    }

    public Computer getAllocatedComputer() {
        return allocatedComputer;
    }
}


package LabMid;

public class StudentArrayList {
    private Student [] students = new Student[10];
    private int index;

    public void push(Student student) {
        students[index] = student;
        index++;
    }

    public String pop() {
        index -= 1;
        var lastStudent = students[index];
        return lastStudent.getName();
    }

    public ArrayIterator createIterator() {
        return new ArrayIterator(this);
    }

    private class ArrayIterator implements Iterator {
        private StudentArrayList students;
        private int index = 0;

        private ArrayIterator(StudentArrayList students) {
            this.students = students;
        }

        @Override
        public Boolean hasNext() {
            return (index < students.index);
        }

        @Override
        public Student current() {
            return students.students[index];
        }

        @Override
        public void next() {
            index++;
        }
    }
}
```

```
package LabMid;


import java.util.ArrayList;

import java.util.Map;


public class WindowsStrategy implements OSStrategy {

  private StudentArrayList students;

  private ArrayList linuxComputers;


  private ArrayList<Student> linuxList = new ArrayList<>();


  WindowsStrategy(StudentArrayList students, Map<String, ArrayList<Computer>>
computersDictionary) {

    this.students = students;

    this.linuxComputers = computersDictionary.get("windows");

  }


  @Override

  public ArrayList allocate() {

    int i = 0;

    Iterator iterator = students.createIterator();

    while(iterator.hasNext()) {

      if (i == linuxComputers.size())

        break;

      var student = iterator.current();

      if (student.getReg() % 2 == 0) {

        student.setAllocatedComputer((Computer) linuxComputers.get(i));

        linuxList.add(student);

        i++;
```

```
            }
            iterator.next();
        }
        return linuxList;
    }

}
```