```java
import Data.PC;
import Data.RandomDataGenerator;
import Data.Student;
import Strategy.LinuxAllocation;
import Strategy.StrategyContext;
import Strategy.WindowsAllocation;

import java.util.Map;

public class Main {
    public static void main(String[] args) {
        Student[] students = RandomDataGenerator.generateStudents(10);
        Map<Integer, PC> pcs = RandomDataGenerator.generatePCs(10);
        StrategyContext context = new StrategyContext();
        context.setStrategy(new LinuxAllocation());
        context.executeStrategy(students, pcs);

        System.out.println("Linux Allocation:");
        for (String s : context.getAllocation()) {
            System.out.println(s);
        }


        context.setStrategy(new WindowsAllocation());
        context.executeStrategy(students, pcs);


        System.out.println("Windows Allocation:");
        for (String s : context.getAllocation()) {
            System.out.println(s);
        }
    }
}
```

```java
package Data;

public record PC(String os,String[] tools) {
}

```

```java
package Data;

public record Student(int regNo, String name, String discipline,
String semester) {

}
```

```java
package Data;

import java.util.HashMap;
import java.util.Map;
import java.util.Random;

public class RandomDataGenerator {
    Random random = new Random();

    public static Student[] generateStudents(int n) {
        Student[] students = new Student[n];
        for (int i = 0; i < n; i++) {
            students[i] = new Student(i, "Student" + i, "Discipline"
 + i, "Semester" + i);
        }
        return students;
    }

    public static Map<Integer, PC> generatePCs(int n) {
        Map<Integer, PC> pcs = new HashMap<>();
        for (int i = 0; i < n; i++) {
            pcs.put(i++, new PC("Windows", new String[]{"Paint", "VS
Code", "Chrome"}));
            pcs.put(i++, new PC("Linux", new String[]{"Gimp", "VS Code
", "Chrome"}));
        }
        return pcs;
    }
}

```

```java
package Iterator;


public interface Iterator {
    Object next();

    boolean hasNext();

    Object next(String key);

    boolean hasNext(String key);

}

```

```java
package Iterator;

import Data.Student;
import jdk.jfr.BooleanFlag;

public class StudentsIterator implements Iterator {

    private final Student[] array;
    private int index;

    public StudentsIterator(Student[] array) {
        this.array = array;
        this.index = 0;
    }

    @Override
    public Student next() {
        return array[index++];
    }

    @Override
    public boolean hasNext() {
        return index < array.length;
    }

    @Override
    public Student next(String key) {
        return next();
    }

    @Override
    public boolean hasNext(String key) {
        if (index >= array.length) return false;
        for (int i = index; i < array.length; i++) {
            if (array[i].regNo() % 2 == Integer.parseInt(key)) {
                index = i;
                return true;
            }
        }
        return false;
    }
}

```

```java
package Iterator;

import Data.PC;

import java.util.Map;

public class ComputersIterator implements Iterator {

    private final Map<Integer, PC> computers;
    private final PC[] computersArray;
    private int index;

    public ComputersIterator(Map<Integer, PC> computers) {
        this.computers = computers;
        this.computersArray = computers.values().toArray(new PC[0]);
        this.index = 0;
    }

    @Override
    public PC next() {
        return computersArray[index++];
    }

    @Override
    public boolean hasNext() {
        return index <= computersArray.length - 1;
    }

    @Override
    public PC next(String key) {
        for (int i = index; i < computersArray.length; i++) {
            if (computersArray[i].os().equals(key)) {
                index=i;
                return computersArray[index++];
            }
        }
        return null;
    }

    @Override
    public boolean hasNext(String key) {
        if(index>=computersArray.length) return false;
        for (int i = index; i < computersArray.length; i++) {
            if (computersArray[i].os().equals(key)) {
                return true;
            }
        }
        return false;
    }


}

```

```java
package Strategy;

import Data.PC;
import Data.Student;
import Iterator.ComputersIterator;
import Iterator.Iterator;
import Iterator.StudentsIterator;

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

public class LinuxAllocation implements AllocationStrategy {
    private final List<String> allocation = new LinkedList<>();

    @Override
    public void allocatePC(Student[] std, Map<Integer, PC> pc) {
        Iterator studentIterator = new StudentsIterator(std);
        Iterator computerIterator = new ComputersIterator(pc);
        while (studentIterator.hasNext("1") && computerIterator.
hasNext("Linux")) {
            Student tempStd = (Student) studentIterator.next("1");
            PC tempPC = (PC) computerIterator.next("Linux");
            allocation.add(tempStd.name() + " got " + tempPC.os() + "
PC");
        }
    }


    public List<String> getAllocations() {
        return allocation;
    }
}

```

```java
package Strategy;

import Data.PC;
import Data.Student;

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

public class StrategyContext {
    private AllocationStrategy strategy;

    public void setStrategy(AllocationStrategy strategy){
        this.strategy=strategy;
    }

    public void executeStrategy(Student[] std, Map<Integer, PC> pc){
        strategy.allocatePC(std,pc);
    }

    public List<String> getAllocation() {
        return strategy.getAllocations();
    }
}

```

```java
package Strategy;

import Data.PC;
import Data.Student;
import Iterator.*;

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

public class WindowsAllocation implements AllocationStrategy{
    private final List<String> allocation = new LinkedList<>();

    @Override
    public void allocatePC(Student[] std, Map<Integer, PC> pc) {
        Iterator studentIterator = new StudentsIterator(std);
        Iterator computerIterator = new ComputersIterator(pc);
        while (studentIterator.hasNext("0") && computerIterator.
hasNext("Windows")) {
            Student tempStd = (Student) studentIterator.next("0");
            PC tempPC = (PC) computerIterator.next("Windows");
            allocation.add(tempStd.name() + " got " + tempPC.os() + "
PC");
        }
    }

    public List<String> getAllocations() {
        return allocation;
    }
}

```

```java
package Strategy;

import Data.PC;
import Data.Student;

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

public interface AllocationStrategy {
    void allocatePC(Student[] std, Map<Integer,PC> pc);
    List<String> getAllocations();
}

```