

```
1 import Data.PC;
2 import Data.RandomDataGenerator;
3 import Data.Student;
4 import Strategy.LinuxAllocation;
5 import Strategy.StrategyContext;
6 import Strategy.WindowsAllocation;
7
8 import java.util.Map;
9
10 public class Main {
11     public static void main(String[] args) {
12         Student[] students = RandomDataGenerator.generateStudents(10);
13         Map<Integer, PC> pcs = RandomDataGenerator.generatePCs(10);
14         StrategyContext context = new StrategyContext();
15         context.setStrategy(new LinuxAllocation());
16         context.executeStrategy(students, pcs);
17
18         System.out.println("Linux Allocation:");
19         for (String s : context.getAllocation()) {
20             System.out.println(s);
21         }
22
23
24         context.setStrategy(new WindowsAllocation());
25         context.executeStrategy(students, pcs);
26
27
28         System.out.println("Windows Allocation:");
29         for (String s : context.getAllocation()) {
30             System.out.println(s);
31         }
32     }
33 }
```

```
1 package Data;  
2  
3 public record PC(String os,String[] tools) {  
4 }  
5
```

```
1 package Data;
2
3 public record Student(int regNo, String name, String discipline,
4     String semester) {
5 }
```

```
1 package Data;
2
3 import java.util.HashMap;
4 import java.util.Map;
5 import java.util.Random;
6
7 public class RandomDataGenerator {
8     Random random = new Random();
9
10    public static Student[] generateStudents(int n) {
11        Student[] students = new Student[n];
12        for (int i = 0; i < n; i++) {
13            students[i] = new Student(i, "Student" + i, "Discipline"
14 + i, "Semester" + i);
15        }
16        return students;
17    }
18
19    public static Map<Integer, PC> generatePCs(int n) {
20        Map<Integer, PC> pcs = new HashMap<>();
21        for (int i = 0; i < n; i++) {
22            pcs.put(i++, new PC("Windows", new String[]{"Paint", "VS
23 Code", "Chrome"}));
24            pcs.put(i++, new PC("Linux", new String[]{"Gimp", "VS Code
25 ", "Chrome"}));
26        }
27        return pcs;
28    }
29 }
```

```
1 package Iterator;
2
3
4 public interface Iterator {
5     Object next();
6
7     boolean hasNext();
8
9     Object next(String key);
10
11     boolean hasNext(String key);
12
13 }
14
```

```
1 package Iterator;
2
3 import Data.Student;
4 import jdk.jfr.BooleanField;
5
6 public class StudentsIterator implements Iterator {
7
8     private final Student[] array;
9     private int index;
10
11     public StudentsIterator(Student[] array) {
12         this.array = array;
13         this.index = 0;
14     }
15
16     @Override
17     public Student next() {
18         return array[index++];
19     }
20
21     @Override
22     public boolean hasNext() {
23         return index < array.length;
24     }
25
26     @Override
27     public Student next(String key) {
28         return next();
29     }
30
31     @Override
32     public boolean hasNext(String key) {
33         if (index >= array.length) return false;
34         for (int i = index; i < array.length; i++) {
35             if (array[i].regNo() % 2 == Integer.parseInt(key)) {
36                 index = i;
37                 return true;
38             }
39         }
40         return false;
41     }
42 }
43
```

```
1 package Iterator;
2
3 import Data.PC;
4
5 import java.util.Map;
6
7 public class ComputersIterator implements Iterator {
8
9     private final Map<Integer, PC> computers;
10    private final PC[] computersArray;
11    private int index;
12
13    public ComputersIterator(Map<Integer, PC> computers) {
14        this.computers = computers;
15        this.computersArray = computers.values().toArray(new PC[0]);
16        this.index = 0;
17    }
18
19    @Override
20    public PC next() {
21        return computersArray[index++];
22    }
23
24    @Override
25    public boolean hasNext() {
26        return index <= computersArray.length - 1;
27    }
28
29    @Override
30    public PC next(String key) {
31        for (int i = index; i < computersArray.length; i++) {
32            if (computersArray[i].os().equals(key)) {
33                index=i;
34                return computersArray[index++];
35            }
36        }
37        return null;
38    }
39
40    @Override
41    public boolean hasNext(String key) {
42        if(index>=computersArray.length) return false;
43        for (int i = index; i < computersArray.length; i++) {
44            if (computersArray[i].os().equals(key)) {
45                return true;
46            }
47        }
48        return false;
49    }
50
51
52 }
53
```

```
1 package Strategy;
2
3 import Data.PC;
4 import Data.Student;
5 import Iterator.ComputersIterator;
6 import Iterator.Iterator;
7 import Iterator.StudentsIterator;
8
9 import java.util.LinkedList;
10 import java.util.List;
11 import java.util.Map;
12
13 public class LinuxAllocation implements AllocationStrategy {
14     private final List<String> allocation = new LinkedList<>();
15
16     @Override
17     public void allocatePC(Student[] std, Map<Integer, PC> pc) {
18         Iterator studentIterator = new StudentsIterator(std);
19         Iterator computerIterator = new ComputersIterator(pc);
20         while (studentIterator.hasNext("1") && computerIterator.
hasNext("Linux")) {
21             Student tempStd = (Student) studentIterator.next("1");
22             PC tempPC = (PC) computerIterator.next("Linux");
23             allocation.add(tempStd.name() + " got " + tempPC.os() + "
PC");
24         }
25     }
26
27
28     public List<String> getAllocations() {
29         return allocation;
30     }
31 }
32
```



```
1 package Strategy;
2
3 import Data.PC;
4 import Data.Student;
5
6 import java.util.List;
7 import java.util.Map;
8
9 public class StrategyContext {
10     private AllocationStrategy strategy;
11
12     public void setStrategy(AllocationStrategy strategy){
13         this.strategy=strategy;
14     }
15
16     public void executeStrategy(Student[] std, Map<Integer, PC> pc){
17         strategy.allocatePC(std,pc);
18     }
19
20     public List<String> getAllocation() {
21         return strategy.getAllocations();
22     }
23 }
24
```

```
1 package Strategy;
2
3 import Data.PC;
4 import Data.Student;
5 import Iterator.*;
6
7 import java.util.LinkedList;
8 import java.util.List;
9 import java.util.Map;
10
11 public class WindowsAllocation implements AllocationStrategy{
12     private final List<String> allocation = new LinkedList<>();
13
14     @Override
15     public void allocatePC(Student[] std, Map<Integer, PC> pc) {
16         Iterator studentIterator = new StudentsIterator(std);
17         Iterator computerIterator = new ComputersIterator(pc);
18         while (studentIterator.hasNext("0") && computerIterator.
hasNext("Windows")) {
19             Student tempStd = (Student) studentIterator.next("0");
20             PC tempPC = (PC) computerIterator.next("Windows");
21             allocation.add(tempStd.name() + " got " + tempPC.os() + "
PC");
22         }
23     }
24
25     public List<String> getAllocations() {
26         return allocation;
27     }
28 }
29
```

```
1 package Strategy;
2
3 import Data.PC;
4 import Data.Student;
5
6 import java.util.List;
7 import java.util.Map;
8
9 public interface AllocationStrategy {
10     void allocatePC(Student[] std, Map<Integer,PC> pc);
11     List<String> getAllocations();
12 }
13
```