Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №6
по дисциплине: **СПП**
Тема: Паттерны проектирования

Выполнил
студент 3 курса
Корнасевич И. Д.

Проверил
Крощенко А. А.

Брест 2021

Цель работы приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Java

Задание 1 Проект «Бургер-закусочная». Реализовать возможность формирования заказа из определенных позиций (тип бургера (веганский, куриный и т.д.)), напиток (холодный – пепси, кока-кола и т.д.; горячий – кофе, чай и т.д.), тип упаковки – с собой, на месте. Должна формироваться итоговая стоимость заказа.

Main.java

```java
package main6.task1;

import main6.task1.burger.ClassicBurger;
import main6.task1.burger.VeganBurger;
import main6.task1.drink.Cola;
import main6.task1.drink.Tea;
import main6.task1.pack.OnSide;
import main6.task1.pack.OutSide;
import java.util.List;


public final class Main {

    public static void main(String[] args) {
        var rest = new Restaurant();
        var prices = List.of(
                rest.makeAnOrder(new VeganBurger(), new Cola(), new OnSide()),
                rest.makeAnOrder(new ClassicBurger(), new Tea(), new OutSide())
        );
        System.out.println(prices);
    }
}
```

Position.java

```java
package main6.task1;

public interface Position {

    Long price();
}
```

Restaurant.java

```java
package main6.task1;

import main6.task1.burger.Burger;
import main6.task1.drink.Drink;
import main6.task1.pack.Package;
import java.util.stream.Stream;


public final class Restaurant {

    public Long makeAnOrder(Burger burger, Drink drink, Package pack) {
        return Stream.of(burger, drink, pack)
                .map(Position::price)
                .mapToLong(Long::valueOf)
                .sum();
    }
}
```

Burger.java

```java
package main6.task1.burger;

import main6.task1.Position;


public interface Burger extends Position {

}
```

Drink.java

```java
package main6.task1.drink;

import main6.task1.Position;


public interface Drink extends Position {

}
```

Package.java

```java
package main6.task1.pack;

import main6.task1.Position;


public interface Package extends Position {

}
```

Задание 2 и 3 Проект «IT-компания». В проекте должен быть реализован класс «Сотрудник» с субординацией (т.е. должна быть возможность определения кому подчиняется сотрудник и кто находится в его подчинении). Для каждого сотрудника помимо сведений о субординации хранятся другие данные (ФИО, отдел, должность, зарплата). Предусмотреть возможность удаления и добавления сотрудника.

Проект «Расчет зарплаты». Для задания, указанного во втором пункте («IT-компания») реализовать расчет зарплаты с выводом полного отчета. Порядок вывода сотрудников в отчете — по старшинству для каждого отдела.

Main.java

```java
package main6.task2;

import java.util.Arrays;


public final class Main {

    public static void main(String[] args) {
        var ivan = Employee.of("Ivan", Department.IT, 1000L);
        var liza = Employee.of("Liza", Department.IT, 1200L);
        var tuan = Employee.of("Tuan", Department.IT, 900L);
        var gray = Employee.of("Gray", Department.SALES, 800L);
        var soap = Employee.of("Soap", Department.SALES, 700L);
        var le = Employee.of("Le", Department.SALES, 900L);
        liza.assignManager(ivan);
        tuan.assignManager(liza);
        gray.assignManager(ivan);
        soap.assignManager(gray);
        le.assignManager(gray);
        var company = new Company(Arrays.asList(ivan, liza, tuan, gray, soap, le));
        company.pringSalaryReport();
    }
}
```

Company.java

```java
package main6.task2;

import java.util.*;
import java.util.stream.Collectors;


public final class Company {

    private final List<Employee> employees;

    public Company(List<Employee> employees) {
```

```
12          this.employees = employees;
13      }
14
15      public Company() {
16          this(new ArrayList<>());
17      }
18
19      public void hire(Employee employee) {
20          employees.add(employee);
21      }
22
23      public void fire(Employee employee) {
24          employees.stream()
25                  .filter(Employee::hasManager)
26                  .filter(e -> e.getManager().orElseThrow().equals(employee))
27                  .forEach(e -> e.assignManager(employee.getManager().orElse(null)));
28      }
29
30      public void pringSalaryReport() {
31          var group = employees.stream().collect(Collectors.groupingBy(Employee::getDepartment));
32          for (var list : group.entrySet()) {
33              printSalaryReport(list.getKey(), list.getValue());
34          }
35      }
36
37      public void printSalaryReport(Department department, List<Employee> employees) {
38          var leaders =
39                  employees.stream()
40                          .filter(x -> !x.hasManager() || x.getManager().isPresent() && !x.getManager
    ().get().getDepartment().equals(department))
41                          .collect(Collectors.toList());
42          var rest = new HashSet<>(employees);
43          while (!rest.isEmpty()) {
44              leaders.forEach(rest::remove);
45              leaders.forEach(System.out::println);
46              leaders = leaders.stream()
47                      .flatMap(x -> x.getSubordinates().stream())
48                      .filter(x -> x.getDepartment().equals(department))
49                      .collect(Collectors.toList());
50          }
51      }
52  }
```

Employee.java

```
1   package main6.task2;
2
3   import java.util.*;
4
5
6   public final class Employee {
7
8       private final String name;
9
10      private final Department department;
11
12      private final Long salary;
13
14      private final List<Employee> subordinates;
15
16      private Employee manager;
17
18      public Employee(String name, Department department, Long salary, Employee manager, List<Employee
    > subordinates) {
19          this.name = name;
20          this.department = department;
21          this.salary = salary;
22          this.manager = manager;
23          this.subordinates = subordinates;
24      }
25
26      public static Employee of(String name, Department department, Long salary) {
27          return new Employee(name, department, salary, null, new ArrayList<>());
28      }
29
30      @Override
31      public int hashCode() {
32          return Objects.hash(name, department, salary);
33      }
34
35      @Override
```

```
36      public boolean equals(Object o) {
37          if (this == o) return true;
38          if (o == null || getClass() != o.getClass()) return false;
39          Employee employee = (Employee) o;
40          return Objects.equals(name, employee.name) && department == employee.department && Objects.
        equals(salary, employee.salary);
41      }
42
43      @Override
44      public String toString() {
45          return "Employee{" +
46                  "name='" + name + '\'' +
47                  ", department=" + department +
48                  ", salary=" + salary +
49                  '}';
50      }
51
52      public void assignManager(Employee employee) {
53          if (manager != null) {
54              manager.subordinates.remove(this);
55          }
56          manager = employee;
57          manager.subordinates.add(this);
58      }
59
60      public Boolean hasManager() {
61          return manager != null;
62      }
63
64      public Optional<Employee> getManager() {
65          return Optional.ofNullable(manager);
66      }
67
68      public Department getDepartment() {
69          return department;
70      }
71
72      public List<Employee> getSubordinates() {
73          return subordinates;
74      }
75  }
```

Пример вывода:

```
1  Employee{name='Gray', department=SALES, salary=800}
2  Employee{name='Soap', department=SALES, salary=700}
3  Employee{name='Le', department=SALES, salary=900}
4  Employee{name='Ivan', department=IT, salary=1000}
5  Employee{name='Liza', department=IT, salary=1200}
6  Employee{name='Tuan', department=IT, salary=900}
```

Вывод Я смоделировал системы реального мира используя паттерны проектирования.