Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Отчёт
По лабораторной работе №6
По дисциплине СПП

Выполнил
Студент группы ПО-3
3-го курса
Кулинкович И. Т.

Проверил
Крощенко А. А.

Брест 2020

Лабораторная работа №6

ВАРИАНТ 13

- Прочитать задания, взятые из каждой группы.
- Определить паттерн проектирования, который может использоваться при реализации задания.
  Пояснить свой выбор.
- Реализовать фрагмент программной системы, используя выбранный паттерн.
  Реализовать все необходимые дополнительные классы.

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.

Задание 2. 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.

Код программы

live.ilyusha.spp6.task1.FastFoodOrder

```
package live.ilyusha.spp6.task1;

enum FastFoodOrderBurgerType {
    BEEF_BURGER,
    CHICKEN_BURGER,
    EGG_BURGER,
    CHEESEBURGER_WITH_BACON,
    BURGER_WITH_SALAMI,
    SPICY_BURGER
}

enum FastFoodOrderDrinkType {
    /* cold drinks */
    COCA_COLA,
    FANTA,
    SPRITE,
    FUZE_TEA,
    BONAQUA,
    /* hot drinks */
    TEA,
    COFFEE
}

enum FastFoodOrderSideType {
    FRENCH_FRIES,
    POTATO_WEDGES,
    CHICKEN_NUGGETS,
    MOZZARELLA_STICKS
}

enum FastFoodLocationType {
    IN_RESTAURANT,
    TAKEOUT,
    DELIVERY
```

```
}

class FastFoodOrder {

    private String orderer;

    private FastFoodOrderBurgerType burger;
    private FastFoodOrderDrinkType drink;
    private FastFoodOrderSideType side;
    private FastFoodLocationType location;

    private FastFoodOrder(String orderer) {
        this.orderer = orderer;
    }

    /* java.lang.Object */

    @Override
    public String toString() {
        return String.format(
            "<FastFoodOrder orderer=\"%s\" burger=%s drink=%s side=%s
location=%s>",
            orderer, burger.name(), drink.name(), side.name(),
location.name()
        );
    }

    /* builder */

    public static class Builder {

        private final FastFoodOrder order;

        public Builder(String orderer) {
            order = new FastFoodOrder(orderer);
            order.burger = null;
            order.drink = null;
            order.side = null;
            order.location = null;
        }

        private Builder(
            String orderer,
            FastFoodOrderBurgerType burger,
            FastFoodOrderDrinkType drink,
            FastFoodOrderSideType side,
            FastFoodLocationType location
        ) {
            order = new FastFoodOrder(orderer);
            order.burger = burger;
            order.drink = drink;
            order.side = side;
            order.location = location;
        }

        public Builder setOrderer(String orderer) {
            return new Builder(orderer, order.burger, order.drink,
order.side, order.location);
        }
```

```
        public Builder setBurger(FastFoodOrderBurgerType burger) {
            return new Builder(order.orderer, burger, order.drink,
order.side, order.location);
        }

        public Builder setDrink(FastFoodOrderDrinkType drink) {
            return new Builder(order.orderer, order.burger, drink,
order.side, order.location);
        }

        public Builder setSide(FastFoodOrderSideType side) {
            return new Builder(order.orderer, order.burger, order.drink,
side, order.location);
        }

        public Builder setLocation(FastFoodLocationType location) {
            return new Builder(order.orderer, order.burger, order.drink,
order.side, location);
        }

        public FastFoodOrder build() {
            return order;
        }

    }

}
```

live.ilyusha.spp6.task1.Main

```
package live.ilyusha.spp6.task1;

public class Main {

    public static void main(String[] args) {
        FastFoodOrder order = new FastFoodOrder.Builder("Kulinkovich I.T.")
            .setBurger(FastFoodOrderBurgerType.CHEESEBURGER_WITH_BACON)
            .setDrink(FastFoodOrderDrinkType.SPRITE)
            .setSide(FastFoodOrderSideType.MOZZARELLA_STICKS)
            .setLocation(FastFoodLocationType.IN_RESTAURANT)
            .build();

        System.out.println(order.toString());
    }

}
```

live.ilyusha.spp6.task2_3

```
package live.ilyusha.spp6.task2_3;

import java.util.ArrayList;
import java.util.Iterator;

enum WorkDepartment {
    LEAD,
    RESEARCH,
    PROJECTS,
    MARKETING
}
```

```
enum WorkField {
    DESIGN,
    DEVELOPMENT,
    MANAGEMENT
}

class Employee implements Iterable<Employee> {

    public static double MONEY_PER_PROJECT = 200;

    private String name;
    private int numProjects;
    private WorkDepartment department;
    private WorkField field;

    private ArrayList<Employee> subordinates = new ArrayList<>();

    public Employee(String name, int numProjects, WorkDepartment department,
WorkField field) {
        this.name = name;
        this.numProjects = numProjects;
        this.department = department;
        this.field = field;
    }

    /* helper methods */

    public void addSubordinate(Employee employee) {
        subordinates.add(employee);
    }

    public void removeSubordinate(Employee employee) {
        subordinates.remove(employee);
        employee.removeAllSubordinates();
    }

    public void removeAllSubordinates() {
        for (Employee e: subordinates) {
            e.removeAllSubordinates();
            e.subordinates.clear();
        }
        subordinates.clear();
    }

    public void logSalary(int padding) {
        System.out.printf(
            "%s%s has salary: %f\n",
            " ".repeat(padding), name,
            MONEY_PER_PROJECT * numProjects
        );
    }

    /* java.lang.Object */

    @Override
    public String toString() {
        return String.format(
            "<Employee name=\"%s\" numProjects=%d department=%s field=%s
subordinates=<arrayList of %d elements>>",
```

```
            name, numProjects, department.name(), field.name(),
subordinates.size()
        );
    }

    /* codegen */

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getNumProjects() {
        return numProjects;
    }

    public void setNumProjects(int numProjects) {
        this.numProjects = numProjects;
    }

    public WorkDepartment getDepartment() {
        return department;
    }

    public void setDepartment(WorkDepartment department) {
        this.department = department;
    }

    public WorkField getField() {
        return field;
    }

    public void setField(WorkField field) {
        this.field = field;
    }

    public ArrayList<Employee> getSubordinates() {
        return subordinates;
    }

    /* Iterable */

    @Override
    public Iterator<Employee> iterator() {
        return new EmployeeIterator(subordinates);
    }
}
```

live.ilyusha.spp6.task2_3.EmployeeIterator

```
package live.ilyusha.spp6.task2_3;

import java.util.Iterator;
import java.util.List;

public class EmployeeIterator implements Iterator<Employee> {
    private List<Employee> files;
```

```
    private int position;

    public EmployeeIterator(List<Employee> files) {
        this.files = files;
        position = 0;
    }

    @Override
    public boolean hasNext() {
        return position < files.size();
    }

    @Override
    public Employee next() {
        return files.get(position++);
    }

}
```

live.ilyusha.spp6.task2_3.ITCompany

```
package live.ilyusha.spp6.task2_3;

import java.util.Iterator;

class ITCompany {

    private String name;
    private Employee ceo;

    public ITCompany(String name, Employee ceo) {
        this.name = name;
        this.ceo = ceo;
    }

    /* helper methods */

    private void logSalaries(int padding, Employee employee) {
        Iterator<Employee> iterator = employee.iterator();
        while (iterator.hasNext()) {
            Employee next = iterator.next();
            next.logSalary(padding + 1);
            logSalaries(padding + 1, next);
        }
    }

    public void logSalaries() {
        System.out.println("====== SALARY LOG BEGIN ========================
");
        ceo.logSalary(1);
        logSalaries(1, ceo);
        System.out.println("====== SALARY LOG END ==========================
");
    }

    /* codegen */

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public Employee getCeo() {
        return ceo;
    }

    public void setCeo(Employee ceo) {
        this.ceo = ceo;
    }

}
```

live.ilyusha.spp6.task2_3.Main

```
package live.ilyusha.spp6.task2_3;

public class Main {

    public static void main(String[] args) {
        // task 2
        Employee ceo = new Employee("Raman Harhun", 5, WorkDepartment.LEAD,
WorkField.MANAGEMENT);
        ITCompany company = new ITCompany("Harbros Solutions", ceo);
        Employee manager = new Employee("Tsimafei Harhun", 4,
WorkDepartment.LEAD, WorkField.MANAGEMENT);
        ceo.addSubordinate(manager);
        Employee worker = new Employee("Ilya Kulinkovich", 10,
WorkDepartment.PROJECTS, WorkField.DEVELOPMENT);
        manager.addSubordinate(worker);
        System.out.println(ceo.getSubordinates().get(0).getSubordinates());
        manager.removeAllSubordinates();
        System.out.println(ceo.getSubordinates());
        System.out.println(ceo);
        // task 3
        manager.addSubordinate(worker);
        ceo.addSubordinate(new Employee("Yana Danilyuk", 8,
WorkDepartment.PROJECTS, WorkField.DESIGN));
        company.logSalaries();
    }

}
```

Спецификация ввода

```
>java Main
```

Пример

```
>java Main
```

Спецификация вывода

Для задачи 1:

```
<данные о заказе>
```

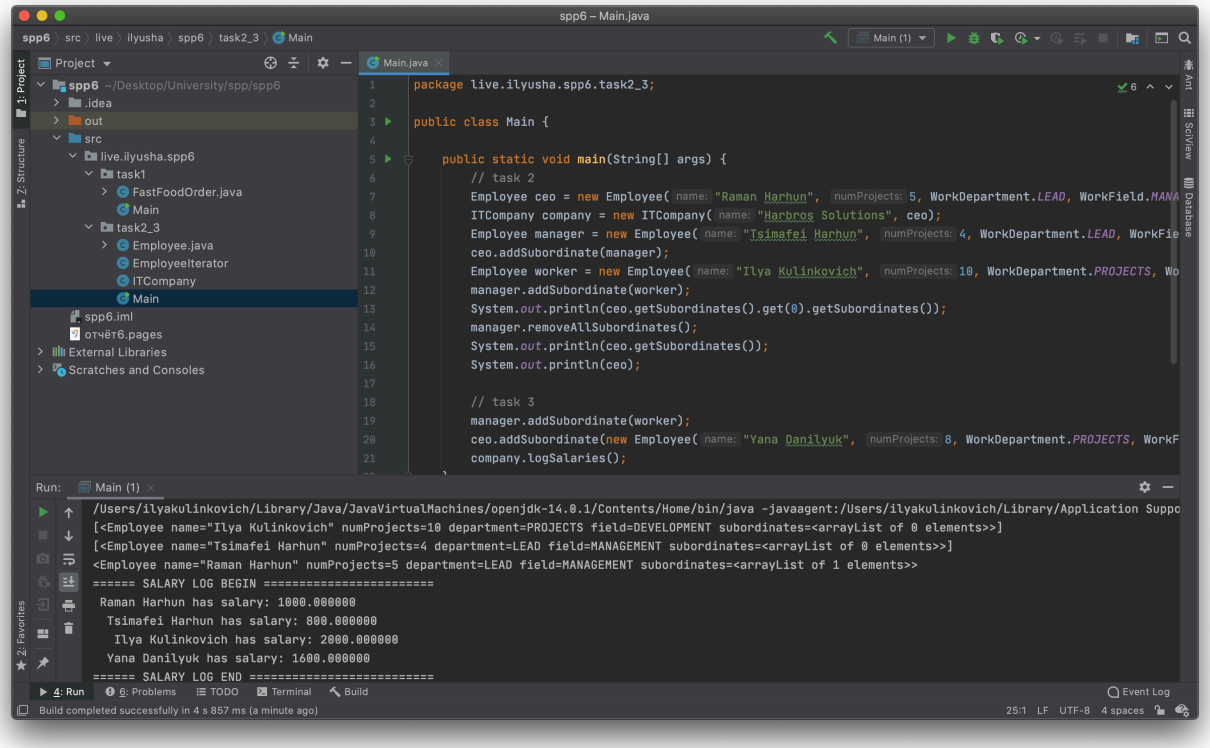
Для задачи 2:

```
<данные о работниках>
<история зарплат работников>
```

Пример

```
[<Employee name="Ilya Kulinkovich" numProjects=10 department=PROJECTS
field=DEVELOPMENT subordinates=<arrayList of 0 elements>>]
[<Employee name="Tsimafei Harhun" numProjects=4 department=LEAD
field=MANAGEMENT subordinates=<arrayList of 0 elements>>]
<Employee name="Raman Harhun" numProjects=5 department=LEAD field=MANAGEMENT
subordinates=<arrayList of 1 elements>>
====== SALARY LOG BEGIN ========================
 Raman Harhun has salary: 1000.000000
  Tsimafei Harhun has salary: 800.000000
   Ilya Kulinkovich has salary: 2000.000000
  Yana Danilyuk has salary: 1600.000000
====== SALARY LOG END ==========================
```

Рисунки с результатами работы программы



Вывод

В данной лабораторной работе я приобрел навыки применения паттернов проектирования при решении практических задач с использованием языка Java.