Министерство образования Республики Беларусь
Учреждение Образования
«Брестский Государственный Технический Университет»
Кафедра ИИТ

Лабораторная работа №6
По дисциплине СПП за 5 семестр
Тема: «Java»

Выполнил:
Студент 3-го курса
Группы ПО-5
Крощук В.В.
Проверил:
Крощенко А.А.

Брест 2021

Лабораторная работа №6

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

Вариант: 3 Согласно зачетной книжке

Задания и выполненные решения:

- 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.

Варианты работ определяются по последней цифре в зачетной книжке (190363).

Первая группа заданий

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.

Для решения данного задания был использован **паттерн Builder (Строитель),** который позволяет создавать сложные объекты пошагово. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объектов.

Код:

Main.java

```
package com.company;

public class Main {

    public static void main(String[] args) {
        FastFoodOrder order = new FastFoodOrder.Builder("Крощук В.В.")
                .setBurger(FastFoodOrderBurgerType.CHEESEBURGER_WITH_BACON)
                .setDrink(FastFoodOrderDrinkType.SPRITE)
                .setSide(FastFoodOrderSideType.MOZZARELLA_STICKS)
                .setLocation(FastFoodLocationType.IN_RESTAURANT)
                .build();
        System.out.println(order.toString());

        System.out.println("-------------------------------------");
```

```
        FastFoodOrder order2 = new FastFoodOrder.Builder("Цуман В.А.")
                .setBurger(FastFoodOrderBurgerType.BEEF_BURGER)
                .setDrink(FastFoodOrderDrinkType.COCA_COLA)
                .setSide(FastFoodOrderSideType.FRENCH_FRIES)
                .setLocation(FastFoodLocationType.DELIVERY)
                .build();
        System.out.println(order2.toString());

        System.out.println("-----------------------------------");

        FastFoodOrder order3 = new FastFoodOrder.Builder("Ляшевич Т.А.")
                .setBurger(FastFoodOrderBurgerType.EGG_BURGER)
                .setDrink(FastFoodOrderDrinkType.FANTA)
                .setSide(FastFoodOrderSideType.POTATO_WEDGES)
                .setLocation(FastFoodLocationType.TAKEOUT)
                .build();
        System.out.println(order3.toString());
    }
}
```

FastFoodOrder.java

```
package com.company;
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 заказчик=\"%s\"\n бургер=\"%s\"\n
напиток=\"%s\"\n дополнение=\"%s\"\n тип_упаковки=\"%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;
        }
    }
}
```

Результат:

```
C:\Users\Viktor\.jdks\openjdk-17\bin\java.exe "-ja
<FastFoodOrder заказчик="Крощук В.В."
 бургер="CHEESEBURGER_WITH_BACON"
 напиток="SPRITE"
 дополнение="MOZZARELLA_STICKS"
 тип_упаковки="IN_RESTAURANT">
-----------------------------------
<FastFoodOrder заказчик="Цуман В.А."
 бургер="BEEF_BURGER"
 напиток="COCA_COLA"
 дополнение="FRENCH_FRIES"
 тип_упаковки="DELIVERY">
-----------------------------------
<FastFoodOrder заказчик="Ляшевич Т.А."
 бургер="EGG_BURGER"
 напиток="FANTA"
 дополнение="POTATO_WEDGES"
 тип_упаковки="TAKEOUT">

Process finished with exit code 0
```

Выполнение происходит согласно поставленной задачи. Работает корректно!

Вторая группа заданий

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.

Для решения данного задания был использован **паттерн Composite (Компоновщик),** который позволяет сгруппировать объекты в древовидную структуру, а затем работать с ними так, как будто это единичный объект.

Код:

Main.java

```
package com.company;

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);
        ITCompany company2 = new ITCompany("EPAM", manager);

        Employee worker = new Employee("Ilya Kulinkovich", 10,
WorkDepartment.PROJECTS, WorkField.DEVELOPMENT);

        Employee worker2 = new Employee("Yana Danilyuk", 8,
WorkDepartment.PROJECTS, WorkField.DESIGN);

        ceo.addSubordinate(manager);
        manager.addSubordinate(worker);
        worker.addSubordinate(worker2);

        System.out.println(ceo);
        System.out.println("Его подчиненные:");
        System.out.println(ceo.getSubordinates().get(0).getSubordinates());
        System.out.println(ceo.getSubordinates());
        System.out.println();

        System.out.println(manager);
        System.out.println("Его подчиненные:");
```

```
System.out.println(manager.getSubordinates().get(0).getSubordinates());
        System.out.println(manager.getSubordinates());
        System.out.println();

        System.out.println(worker);
        System.out.println("Его подчиненные:");

System.out.println(worker.getSubordinates().get(0).getSubordinates());
        System.out.println(worker.getSubordinates());
        System.out.println();

        System.out.println(worker2);
        System.out.println();
    }
}
```

Employee.java

```
package com.company;
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() {
        System.out.printf("%s%s его зарплата: %f$\n", " ", name,
MONEY_PER_PROJECT * numProjects);
    }

    /* java.lang.Object */

    @Override
    public String toString() {
        return String.format(
            "<Employee имя=\"%s\" назв_проекта=\"%d\" отдел=\"%s\"
сфера=\"%s\" подчинение=<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);
    }
}
```

EmployeeIterator.java

```
package com.company;
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++);
    }
}
```

ITCompany.java

```
package com.company;
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(Employee employee) {
        Iterator<Employee> iterator = employee.iterator();
        while (iterator.hasNext()) {
            Employee next = iterator.next();
            next.logSalary();
            logSalaries(next);
        }
    }

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

}
```

Результат:

```
C:\Users\Viktor\.jdks\openjdk-17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.3\lib\idea_rt
<Employee имя="Raman Harhun" назв_проекта="5" отдел="LEAD" сфера="MANAGEMENT" подчинение=<arrayList of "1" elements>>
Его подчиненные:
[<Employee имя="Ilya Kulinkovich" назв_проекта="10" отдел="PROJECTS" сфера="DEVELOPMENT" подчинение=<arrayList of "1" elements>>]
[<Employee имя="Tsimafei Harhun" назв_проекта="4" отдел="LEAD" сфера="MANAGEMENT" подчинение=<arrayList of "1" elements>>]

<Employee имя="Tsimafei Harhun" назв_проекта="4" отдел="LEAD" сфера="MANAGEMENT" подчинение=<arrayList of "1" elements>>
Его подчиненные:
[<Employee имя="Yana Danilyuk" назв_проекта="8" отдел="PROJECTS" сфера="DESIGN" подчинение=<arrayList of "0" elements>>]
[<Employee имя="Ilya Kulinkovich" назв_проекта="10" отдел="PROJECTS" сфера="DEVELOPMENT" подчинение=<arrayList of "1" elements>>]

<Employee имя="Ilya Kulinkovich" назв_проекта="10" отдел="PROJECTS" сфера="DEVELOPMENT" подчинение=<arrayList of "1" elements>>
Его подчиненные:
[]
[<Employee имя="Yana Danilyuk" назв_проекта="8" отдел="PROJECTS" сфера="DESIGN" подчинение=<arrayList of "0" elements>>]

<Employee имя="Yana Danilyuk" назв_проекта="8" отдел="PROJECTS" сфера="DESIGN" подчинение=<arrayList of "0" elements>>


Process finished with exit code 0
```

Исходя из полученных результатов, все работает корректно, согласно поставленному заданию.

Третья группа заданий

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

Для решения данного задания был использован **паттерн Iterator (Итератор),** который даёт возможность последовательно обходить элементы составных объектов, не раскрывая их внутреннего представления.

Код:

Main.java

```
package main;

public class Main {
public static void main(String[] args) {

        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);
         ITCompany company2 = new ITCompany("EPAM", manager);

         Employee worker = new Employee("Ilya Kulinkovich", 10,
WorkDepartment.PROJECTS, WorkField.DEVELOPMENT);

         Employee worker2 = new Employee("Yana Danilyuk", 8,
WorkDepartment.PROJECTS, WorkField.DESIGN);

         ceo.addSubordinate(manager);
         manager.addSubordinate(worker);
         worker.addSubordinate(worker2);

         System.out.println(ceo);
         System.out.println("Его подчиненные:");
         System.out.println(ceo.getSubordinates().get(0).getSubordinates());
         System.out.println(ceo.getSubordinates());
         System.out.println();

         System.out.println(manager);
         System.out.println("Его подчиненные:");

System.out.println(manager.getSubordinates().get(0).getSubordinates());
         System.out.println(manager.getSubordinates());
         System.out.println();

         System.out.println(worker);
         System.out.println("Его подчиненные:");

System.out.println(worker.getSubordinates().get(0).getSubordinates());
         System.out.println(worker.getSubordinates());
         System.out.println();

         System.out.println(worker2);
         System.out.println();

         //сдесь вывод суммы
         company.logSalaries();
         System.out.println();
         company2.logSalaries();


    }
}
```

Employee.java

```
package main;

import java.util.ArrayList;

enum WorkDepartment {
    LEAD,
    RESEARCH,
    PROJECTS,
    MARKETING
}

enum WorkField {
    DESIGN,
    DEVELOPMENT,
    MANAGEMENT
}

class Employee implements EmployeeIterator {

    public static double MONEY_PER_PROJECT = 200;

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


    int position = 0;

    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() {
        System.out.printf("%s%s его зарплата: %f$\n", " ", name,
MONEY_PER_PROJECT * numProjects);
    }

    /* java.lang.Object */

    @Override
    public String toString() {
        return String.format(
            "<Employee имя=\"%s\" назв_проекта=\"%d\" отдел=\"%s\"
сфера=\"%s\" подчинение=<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;
    }

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

    @Override
    public Employee next() {
        if (!hasNext()) {
            return null;
        }
        Employee employee = subordinates.get(position);
        position++;
        return employee;
    }

    @Override
    public void reset() {
        position = 0;
    }

    /* Iterable */

}
```

EmployeeIterator.java

```
package main;

public interface EmployeeIterator {
    boolean hasNext();

    Employee next();

    void reset();
}
```

ITCompany.java

```
package main;

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(Employee employee) {
        while (employee.hasNext()) {
            Employee next = employee.next();
            next.logSalary();
            logSalaries(next);
        }
        employee.reset();
    }

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

}
```

Результат:

```
C:\Users\Viktor\.jdks\openjdk-17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.3\lib\id
<Employee имя="Raman Harhun" назв_проекта="5" отдел="LEAD" сфера="MANAGEMENT" подчинение=<arrayList of "1" elements>>
Его подчиненные:
[<Employee имя="Ilya Kulinkovich" назв_проекта="10" отдел="PROJECTS" сфера="DEVELOPMENT" подчинение=<arrayList of "1" elements>>]
[<Employee имя="Tsimafei Harhun" назв_проекта="4" отдел="LEAD" сфера="MANAGEMENT" подчинение=<arrayList of "1" elements>>]

<Employee имя="Tsimafei Harhun" назв_проекта="4" отдел="LEAD" сфера="MANAGEMENT" подчинение=<arrayList of "1" elements>>
Его подчиненные:
[<Employee имя="Yana Danilyuk" назв_проекта="8" отдел="PROJECTS" сфера="DESIGN" подчинение=<arrayList of "0" elements>>]
[<Employee имя="Ilya Kulinkovich" назв_проекта="10" отдел="PROJECTS" сфера="DEVELOPMENT" подчинение=<arrayList of "1" elements>>]

<Employee имя="Ilya Kulinkovich" назв_проекта="10" отдел="PROJECTS" сфера="DEVELOPMENT" подчинение=<arrayList of "1" elements>>
Его подчиненные:
[]
[<Employee имя="Yana Danilyuk" назв_проекта="8" отдел="PROJECTS" сфера="DESIGN" подчинение=<arrayList of "0" elements>>]

<Employee имя="Yana Danilyuk" назв_проекта="8" отдел="PROJECTS" сфера="DESIGN" подчинение=<arrayList of "0" elements>>
```

```
=============== SALARY LOG BEGIN ===============
 Raman Harhun его зарплата: 1000,000000$
 Tsimafei Harhun его зарплата: 800,000000$
 Ilya Kulinkovich его зарплата: 2000,000000$
 Yana Danilyuk его зарплата: 1600,000000$
=============== SALARY LOG END ===============

=============== SALARY LOG BEGIN ===============
 Tsimafei Harhun его зарплата: 800,000000$
 Ilya Kulinkovich его зарплата: 2000,000000$
 Yana Danilyuk его зарплата: 1600,000000$
=============== SALARY LOG END ===============

Process finished with exit code 0
```

Исходя из полученных результатов, все работает корректно, согласно поставленному заданию.

Вывод: приобрел навыки применения паттернов проектирования при решении практических задач с использованием языка Java. В процессе были использованы следующие паттерны: Builder (Строитель), Composite (Компоновщик), Iterator (Итератор).