МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ

по лабораторной работе №6

Выполнил:
студент группы ПО-9
Дарашкевич Д.И.

Проверил:
Крощенко А.А.

Брест 2024

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

Общее задание:

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.

Вариант 3

Задание 1:

3) 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.

Код программы:

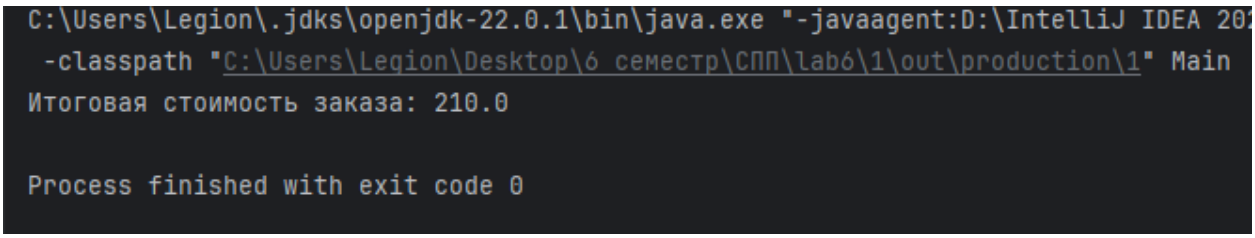
```
class Burger {
    private String type;
    private double price;

    public Burger(String type, double price) {
        this.type = type;
        this.price = price;
    }

    public double getPrice() {
        return price;
    }
}

class Beverage {
    private String type;
    private double price;

    public Beverage(String type, double price) {
        this.type = type;
        this.price = price;
    }

    public double getPrice() {
        return price;
    }
}

class Packaging {
    private String type;
    private double price;

    public Packaging(String type, double price) {
        this.type = type;
        this.price = price;
    }

    public double getPrice() {
        return price;
    }
}
```

```
class Order {
    private Burger burger;
    private Beverage beverage;
    private Packaging packaging;

    public Order(Burger burger, Beverage beverage, Packaging packaging) {
        this.burger = burger;
        this.beverage = beverage;
        this.packaging = packaging;
    }

    public double calculateTotalCost() {
        return burger.getPrice() + beverage.getPrice() + packaging.getPrice();
    }
}

class OrderBuilder {
    private Burger burger;
    private Beverage beverage;
    private Packaging packaging;

    public OrderBuilder addBurger(Burger burger) {
        this.burger = burger;
        return this;
    }

    public OrderBuilder addBeverage(Beverage beverage) {
        this.beverage = beverage;
        return this;
    }

    public OrderBuilder addPackaging(Packaging packaging) {
        this.packaging = packaging;
        return this;
    }

    public Order build() {
        return new Order(burger, beverage, packaging);
    }
}

public class Main {
    public static void main(String[] args) {
        Burger burger = new Burger("Веганский", 150.0);
        Beverage beverage = new Beverage("Пепси", 50.0);
        Packaging packaging = new Packaging("С собой", 10.0);

        Order order = new OrderBuilder()
                .addBurger(burger)
                .addBeverage(beverage)
                .addPackaging(packaging)
                .build();

        double totalCost = order.calculateTotalCost();

        System.out.println("Итоговая стоимость заказа: " + totalCost);
    }
}
```

Входные данные:

```
Burger burger = new Burger("Веганский", 150.0);
Beverage beverage = new Beverage("Пепси", 50.0);
Packaging packaging = new Packaging("С собой", 10.0);
```

Результат работы программы:



Задание 2:

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.

Код программы:

```
import java.util.ArrayList;
import java.util.List;


class Employee {
    private String name;
    private String department;
    private String position;
    private double salary;
    private List<Employee> subordinates;

    public Employee(String name, String department, String position, double salary) {
        this.name = name;
        this.department = department;
        this.position = position;
        this.salary = salary;
        this.subordinates = new ArrayList<>();
    }

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

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

    public void printEmployee() {
        System.out.println("Name: " + name);
        System.out.println("Department: " + department);
        System.out.println("Position: " + position);
        System.out.println("Salary: " + salary);
        System.out.println("Subordinates:");
        for (Employee subordinate : subordinates) {
            subordinate.printEmployee();
        }
    }
}

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

        Employee ceo = new Employee("John Doe", "Management", "CEO", 10000);
        Employee manager1 = new Employee("Alice Smith", "Management", "Manager", 7000);
        Employee manager2 = new Employee("Bob Johnson", "Management", "Manager", 7000);
        Employee developer1 = new Employee("Charlie Brown", "Engineering", "Developer", 5000);
        Employee developer2 = new Employee("David Miller", "Engineering", "Developer", 5000);
```

```
        ceo.addSubordinate(manager1);
        ceo.addSubordinate(manager2);
        manager1.addSubordinate(developer1);
        manager2.addSubordinate(developer2);


        System.out.println("Company Structure:");
        ceo.printEmployee();
    }
}
```

Входные данные:

```
Employee ceo = new Employee("John Doe", "Management", "CEO", 10000);
        Employee manager1 = new Employee("Alice Smith", "Management",
"Manager", 7000);
        Employee manager2 = new Employee("Bob Johnson", "Management",
"Manager", 7000);
        Employee developer1 = new Employee("Charlie Brown", "Engineering",
"Developer", 5000);
        Employee developer2 = new Employee("David Miller", "Engineering",
"Developer", 5000);
```

Результат работы программы:

```
Company Structure:
Name: John Doe
Department: Management
Position: CEO
Salary: 10000.0
Subordinates:
Name: Alice Smith
Department: Management
Position: Manager
Salary: 7000.0
Subordinates:
Name: Charlie Brown
Department: Engineering
Position: Developer
Salary: 5000.0
Subordinates:
Name: Bob Johnson
Department: Management
Position: Manager
Salary: 7000.0
Subordinates:
Name: David Miller
Department: Engineering
Position: Developer
Salary: 5000.0
```

Задание 3:

3) 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.

Код программы:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.List;

class Employee {
    private String name;
    private String department;
    private double salary;
    private int yearsOfExperience;

    public Employee(String name, String department, double salary, int yearsOfExperience) {
        this.name = name;
        this.department = department;
        this.salary = salary;
        this.yearsOfExperience = yearsOfExperience;
    }

    public String getName() {
        return name;
    }

    public String getDepartment() {
        return department;
    }

    public double getSalary() {
        return salary;
    }

    public int getYearsOfExperience() {
        return yearsOfExperience;
    }

    @Override
    public String toString() {
        return "Employee{" +
                "name='" + name + '\'' +
                ", department='" + department + '\'' +
                ", salary=" + salary +
                ", yearsOfExperience=" + yearsOfExperience +
                '}';
    }
}

class Department {
    private String name;
    private List<Employee> employees;

    public Department(String name) {
        this.name = name;
        this.employees = new ArrayList<>();
    }

    public void addEmployee(Employee employee) {
        employees.add(employee);
    }

    public List<Employee> getEmployees() {
        return employees;
    }
```

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

    public void sortEmployeesByExperience() {
        Collections.sort(employees,
Comparator.comparingInt(Employee::getYearsOfExperience).reversed());
    }
}

class EmployeeIterator implements Iterator<Employee> {
    private List<Employee> employees;
    private int position = 0;

    public EmployeeIterator(List<Employee> employees) {
        this.employees = employees;
    }

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

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

class SalaryReport {
    private List<Department> departments;

    public SalaryReport() {
        this.departments = new ArrayList<>();
    }

    public void addDepartment(Department department) {
        departments.add(department);
    }

    public void generateReport() {
        for (Department department : departments) {
            System.out.println("Department: " + department.getName());
            department.sortEmployeesByExperience();
            EmployeeIterator iterator = new EmployeeIterator(department.getEmployees());

            while (iterator.hasNext()) {
                Employee employee = iterator.next();
                System.out.println(employee);
            }
            System.out.println();
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Department devDepartment = new Department("Development");
        Department hrDepartment = new Department("Human Resources");

        devDepartment.addEmployee(new Employee("Alice", "Development", 70000, 5));
        devDepartment.addEmployee(new Employee("Bob", "Development", 60000, 3));
        devDepartment.addEmployee(new Employee("Charlie", "Development", 80000, 7));
```

```
        hrDepartment.addEmployee(new Employee("David", "Human Resources", 50000, 4));
        hrDepartment.addEmployee(new Employee("Eve", "Human Resources", 45000, 2));

        SalaryReport report = new SalaryReport();
        report.addDepartment(devDepartment);
        report.addDepartment(hrDepartment);

        report.generateReport();
    }
}
```

Входные данные:

```
        Department devDepartment = new Department("Development");
        Department hrDepartment = new Department("Human Resources");

        devDepartment.addEmployee(new Employee("Alice", "Development", 70000, 5));
        devDepartment.addEmployee(new Employee("Bob", "Development", 60000, 3));
        devDepartment.addEmployee(new Employee("Charlie", "Development", 80000, 7));

        hrDepartment.addEmployee(new Employee("David", "Human Resources", 50000, 4));
        hrDepartment.addEmployee(new Employee("Eve", "Human Resources", 45000, 2));
```

Результат работы программы:

```
Department: Development
Employee{name='Charlie', department='Development', salary=80000.0, yearsOfExperience=7}
Employee{name='Alice', department='Development', salary=70000.0, yearsOfExperience=5}
Employee{name='Bob', department='Development', salary=60000.0, yearsOfExperience=3}

Department: Human Resources
Employee{name='David', department='Human Resources', salary=50000.0, yearsOfExperience=4}
Employee{name='Eve', department='Human Resources', salary=45000.0, yearsOfExperience=2}


Process finished with exit code 0
```

Вывод: в ходе выполнения данной лабораторной работы я приобрел навыки применения паттернов проектирования при решении практических задач с использованием языка Java.