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:

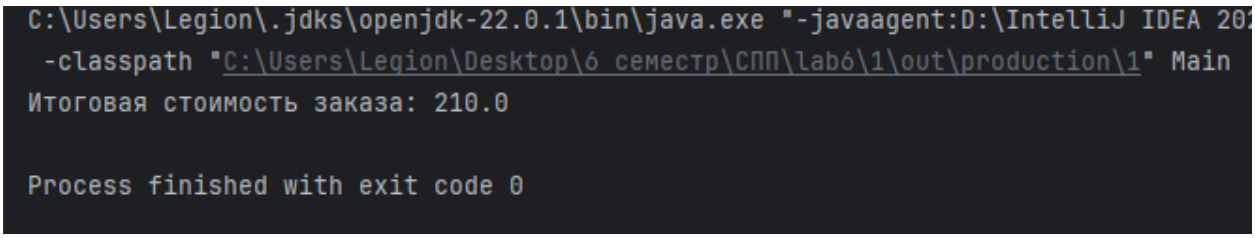
```
class Burger {
    private String type;
    private double price;

    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:

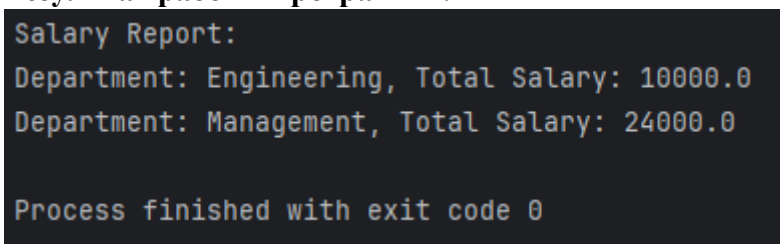
```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

interface Visitor {
    void visit(Employee employee);
}

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 accept(Visitor visitor) {
        visitor.visit(this);
        for (Employee subordinate : subordinates) {
            subordinate.accept(visitor);
        }
    }

    public String getName() {
        return name;
    }

    public String getDepartment() {
        return department;
    }

    public String getPosition() {
        return position;
    }

    public double getSalary() {
        return salary;
    }
}

class SalaryCalculator implements Visitor {
    private Map<String, Double> departmentSalaries;

    public SalaryCalculator() {
        this.departmentSalaries = new HashMap<>();
    }
```

```
    @Override
    public void visit(Employee employee) {
        String department = employee.getDepartment();
        double salary = employee.getSalary();
        departmentSalaries.put(department, departmentSalaries.getOrDefault(department, 0.0) +
salary);
    }

    public void printReport() {
        System.out.println("Salary Report:");
        for (Map.Entry<String, Double> entry : departmentSalaries.entrySet()) {
            System.out.println("Department: " + entry.getKey() + ", Total Salary: " +
entry.getValue());
        }
    }
}

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);

        SalaryCalculator calculator = new SalaryCalculator();

        ceo.accept(calculator);

        calculator.printReport();
    }
}
```

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:



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