МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №6

Выполнил:

Студент 3 курса
группы ПО-9
Кучко Ярослав Валерьевич

Проверил:

Крощенко А. А.

Брест 2024

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

Вариант 2 (210659)

Задание 1. Заводы по производству автомобилей. Реализовать возможность создавать автомобили различных типов на различных заводах.
Паттерн – абстрактная фабрика. Абстрактная фабрика объявляет методы создания различных абстрактных продуктов. Конкретные фабрики относятся каждая к своей вариации продуктов и реализуют методы абстрактной фабрики, позволяя создавать все продукты определённой вариации.

Код программы:
Класс автомобиля.
```
public abstract class Car {
    protected String type;

    @Override
    public String toString() {
        return "Car{" +
                "type='" + type + '\'' +
                '}';
    }
}
```

Его наследники – седан и хетчбэк.
```
public class SedanCar extends Car {
    public SedanCar() {
        this.type = "Sedan";
    }
}

public class HatchbackCar extends Car {
    public HatchbackCar() {
        this.type = "Hatchback";
    }
}
```

Абстрактная фабрика автомобилей.
```
public interface CarAbstractFactory {
    Car createCar();
}
```

Её реализации – фабрика седанов и фабрика хетчбэков.
```
public class SedanFactory implements CarAbstractFactory {
    @Override
    public Car createCar() {
        return new SedanCar();
    }
}

public class HatchbackFactory implements CarAbstractFactory{
    @Override
    public Car createCar() {
        return new HatchbackCar();
    }
}
```

Пример использования.

```
public static void main(String[] args) {
    CarAbstractFactory factory = new SedanFactory();
    Car car = factory.createCar();
    System.out.println(car);

    factory = new HatchbackFactory();
    car = factory.createCar();
    System.out.println(car);
}
```

Результат работы:

```
Car{type='Sedan'}
Car{type='Hatchback'}
```

Задание 2. Проект «Универсальная электронная карта». В проекте должна быть реализована универсальная электронная карта, в которой есть функции паспорта, страхового полиса, банковской карты и т. д.

Паттерн – фасад. Фасад предоставляет быстрый доступ к определённой функциональности подсистемы. Он «знает», каким классам нужно переадресовать запрос, и какие данные для этого нужны.

Код программы:

Класс универсальной карты.

```
public class DigitalCard {
    protected String personalId;
    protected String bankCardNumber;
    protected String insuranceNumber;

    public DigitalCard(String personalId, String bankCardNumber, String
insuranceNumber) {
        this.personalId = personalId;
        this.bankCardNumber = bankCardNumber;
        this.insuranceNumber = insuranceNumber;
    }

    public String getPassportInfo() {
        return PassportSubsystem.getInfo(personalId);
    }

    public String getBankCardInfo() {
        return BankSubsystem.getInfo(bankCardNumber);
    }

    public String getInsuranceInfo() {
        return InsuranceSubsystem.getInfo(insuranceNumber);
    }

}
```

Классы подсистем.

```
public class BankSubsystem {

    public static String getInfo(String cardNumber) {
        double balance = 100;
        // ... (поиск информации)
        return "На балансе карты №" + cardNumber + " " + balance;

    }
}
```

```
public class InsuranceSubsystem {

    public static String getInfo(String insuranceNumber) {
        // ... (поиск информации)
        return "Данные о страховании №" + insuranceNumber;
    }

}
public class PassportSubsystem {

    public static String getInfo(String personalId) {
        // ... (поиск информации)
        return "Паспортные данные по ИН №" + personalId;
    }

}
```

Пример использования.

```
public static void main(String[] args) {
    DigitalCard card = new DigitalCard("12344321", "4567890711114332",
"4567898764537212 34");

    System.out.println(card.getPassportInfo());
    System.out.println(card.getBankCardInfo());
    System.out.println(card.getInsuranceInfo());
}
```

Результат работы:

```
Паспортные данные по ИН №12344321
На балансе карты №4567890711114332 100.0
Данные о страховании №456789876453721234
```

Задание 3. Проект «Принтеры». В проекте должны быть реализованы разные модели принтеров, которые выполняют разные виды печати.

Паттерн – стратегия.

Код программы:

Класс контекста.

```
public class PrinterContext {
    private PrinterStrategy strategy;

    public void setStrategy(PrinterStrategy strategy) {
        this.strategy = strategy;
    }

    public PrintResult executePrint(String message) {
        return strategy.print(message);
    }
}
```

Интерфейс стратегии печати и её реализации.

```
public interface PrinterStrategy {
    PrintResult print(String message);
}

public class PhotoPrintStrategy implements PrinterStrategy {
    @Override
    public PrintResult print(String message) {
        System.out.println("Printing photo with URI " + message);
        PhotoPrintResult photo = new PhotoPrintResult(message);
        return photo;
    }
}
```

```
public class TextPrintStrategy implements PrinterStrategy {
    @Override
    public PrintResult print(String message) {
        System.out.println("Printing text: " + message);
        TextPrintResult text = new TextPrintResult(message);
        return text;
    }
}
```

Пример использования.

```
public class Task3 {
    public static void main(String[] args) {
        PrinterContext context = new PrinterContext();

        context.setStrategy(new TextPrintStrategy());
        System.out.println(context.executePrint("Text text text text"));

        context.setStrategy(new PhotoPrintStrategy());
        System.out.println(context.executePrint("./photo.png"));
    }
}
```

Результат работы:

```
Printing text: Text text text text
TextPrintResult{text='Text text text text', type='null'}
Printing photo with URI ./photo.png
PhotoPrintResult{URI='./photo.png'}
```