Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №6
По дисциплине: «СПП»
Вариант 11

Выполнил:
Студент 3 курса
Группы ПО-9
Лебедович В.А.
Проверил:
Крощенко А.А

Брест 2024

Лабораторная работа №6

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

Задание 1: Завод по производству смартфонов. Обеспечить создание нескольких различных моделей мобильных телефонов с заранее выбранными характеристиками.

Для данного задания воспользуемся порождающим паттерном Фабричный метод для создания объектов без указания их конкретных классов. Фабричный метод определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать. В данном задании паттерн Фабричный метод подходит для создания различных моделей мобильных телефонов с заранее выбранными характеристиками.

Код программы:

Smartphone.java

```
public abstract class Smartphone {
    public String modelName;
    public String CPU;
    public int storageRAM;
    public int storageROM;

    public Smartphone(String modelName, String CPU, int storageRAM, int storageROM) {
        this.modelName = modelName;
        this.CPU = CPU;
        this.storageRAM = storageRAM;
        this.storageROM = storageROM;
    }

    public abstract void printInfo();
}
```

SmartphoneCreator.java

```
public abstract class SmartphoneCreator {
    public abstract Smartphone createSmartphone();
}
```

LaserPrinter.java

```
public class LaserPrinter extends Printer{
    @Override
    public void printData() {
        System.out.println("Лазерный принтер печатает текст: \n" + data);
    }
}
```

SamsungCreator.java

```
public class SamsungCreator extends SmartphoneCreator{
    @Override
    public Smartphone createSmartphone() {
        return new Samsung("Galaxy S24 Ultra", "Qualcomm Snapdragon 8 Gen 3", 12, 256);
    }
}
```

XiaomiCreator.java

```
public class XiaomiCreator extends SmartphoneCreator{
    @Override
    public Smartphone createSmartphone() {
        return new Xiaomi("POCO X4 GT", "MediaTek Dimensity 8100", 8, 256);
    }
}
```

Samsung.java

```
public class Samsung extends Smartphone{
    public Samsung(String modelName, String CPU, int storageRAM, int storageROM) {
        super(modelName, CPU, storageRAM, storageROM);
    }

    @Override
    public void printInfo() {
        System.out.println("Информация о смартфоне Samsung:\nМодель: " + modelName
+"\nПроцессор: " +
                CPU + "\nОперативная память: " + storageRAM + "Гб\nПамять: " +
storageROM + "Гб");
    }
}
```

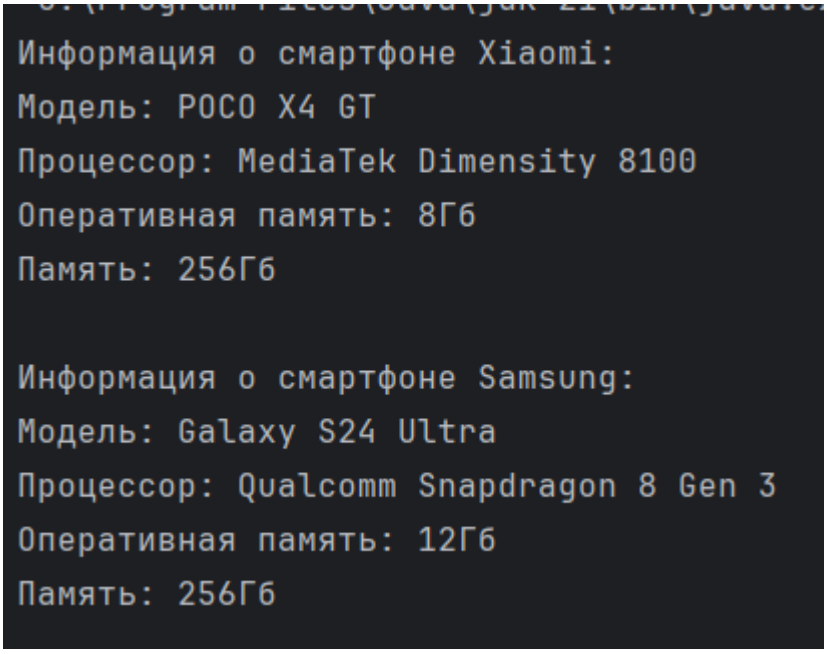
Xiaomi.java

```
public class Xiaomi extends Smartphone{
    public Xiaomi(String modelName, String CPU, int storageRAM, int storageROM) {
        super(modelName, CPU, storageRAM, storageROM);
    }

    @Override
    public void printInfo() {
        System.out.println("Информация о смартфоне Xiaomi:\nМодель: " + modelName
+"\nПроцессор: " +
                CPU + "\nОперативная память: " + storageRAM + "Гб\nПамять: " +
storageROM + "Гб");
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        XiaomiCreator xiaomiCreator = new XiaomiCreator();
        Smartphone xiaomi = xiaomiCreator.createSmartphone();
        xiaomi.printInfo();

        System.out.println("");

        SamsungCreator samsungCreator = new SamsungCreator();
        Smartphone samsung = samsungCreator.createSmartphone();
        samsung.printInfo();
    }
}
```

Результат работы программы:



Задание 2: Проект «Электронный градусник». В проекте должен быть реализован класс, который дает возможность пользоваться аналоговым градусником так же, как и электронным. В классе «Аналоговый градусник» хранится высота ртутного столба и границы измерений (верхняя и нижняя).

Для данного задания воспользуемся структурным паттерном Адаптер. Паттерн Адаптер преобразует интерфейс класса к другому интерфейсу, на который рассчитан клиент. В данном случае мы можем создать адаптер, который преобразует интерфейс аналогового градусника в интерфейс электронного градусника.

Код программы:

ElectronicThermometer.java

```
public interface ElectronicThermometer {
    double getTemperature();
}
```

AnalogAdapter.java

```
public class AnalogAdapter implements ElectronicThermometer{
    public AnalogThermometer analogThermometer;

    public AnalogAdapter(AnalogThermometer analogThermometer) {
        this.analogThermometer = analogThermometer;
    }

    @Override
    public double getTemperature() {
        return analogThermometer.getLowerLimit() + analogThermometer.getMercuryHeight()
/ analogThermometer.getGradeScale();
    }
}
```

AnalogThermometer.java

```java
public class AnalogThermometer {
    private double mercuryHeight;
    private final double lowerLimit;
    private final double upperLimit;
    private final double gradeScale;

    public AnalogThermometer(double lowerLimit, double upperLimit) {
        this.mercuryHeight = 0;
        this.lowerLimit = lowerLimit;
        this.upperLimit = upperLimit;
        this.gradeScale = 10;
    }

    public double getMercuryHeight() {
        return mercuryHeight;
    }

    public void setMercuryHeight(double mercuryHeight) {
        try {
            if (mercuryHeight < 0){
                throw new Exception("Вы замёрзли!");
            }
            else if (mercuryHeight > ((upperLimit - lowerLimit)*gradeScale)){
                this.mercuryHeight = (upperLimit - lowerLimit)*gradeScale;
                throw new Exception("Тише-тише, остынь, парень!");
            }
            else {
                this.mercuryHeight = mercuryHeight;
            }
        }
        catch (Exception e){
            System.out.println(e.getMessage());
        }
    }

    public double getLowerLimit() {
        return lowerLimit;
    }

    public double getUpperLimit() {
        return upperLimit;
    }

    public double getGradeScale() {
        return gradeScale;
    }
}
```

Main.java

```java
public class Main {
    public static void main(String[] args) {
        AnalogThermometer analogThermometer = new AnalogThermometer(35,42);
        AnalogAdapter analogAdapter = new AnalogAdapter(analogThermometer);
        analogThermometer.setMercuryHeight(16);
        System.out.println("Температура: " + analogAdapter.getTemperature() + "C°");
    }
}
```

Результаты работы программы:

```
"C:\Program Files\Java
Температура: 36.6C°
```

Задание 3: Проект «Банкомат». Предусмотреть выполнение основных операций (ввод пин-кода, снятие суммы, завершение работы) и наличие различных режимов работы (ожидание, аутентификация, выполнение операции, блокировка – если нет денег). Атрибуты: общая сумма денег в банкомате, ID.

Для данного задания воспользуемся поведенческим паттерном Состояние. Этот паттерн позволяет объекту изменять свое поведение в зависимости от внутреннего состояния. В данном случае, банкомат может находиться в различных состояниях: ожидание, аутентификация, выполнение операции, блокировка.

Код программы:

ATM.java

```
public class ATM {
    public double amountOfMoney;
    public String PIN_code;
    public StateOfATM state;

    public ATM(double amountOfMoney, String PIN_code) {
        this.amountOfMoney = amountOfMoney;
        this.PIN_code = PIN_code;
        this.state = new WaitingState(this);
    }

    public void setState(StateOfATM state) {
        this.state = state;
    }

    public double getAmountOfMoney(){
        return amountOfMoney;
    }

    public void setAmountOfMoney(double amountOfMoney){
        this.amountOfMoney = amountOfMoney;
    }

    public void insertCard(){
        state.insertCard();
    }

    public void enterPIN_code(String PIN_code){
        state.enterPIN_code(PIN_code);
    }

    public void withdrawMoney(int amountOfMoney){
        state.withdrawMoney(amountOfMoney);
    }

    public void getCard(){
        state.getCard();
    }
}
```

StateOfATM.java

```
interface StateOfATM {
    void insertCard();
    void enterPIN_code(String pinCode);
    void withdrawMoney(int cashAmount);
    void getCard();
}
```

WaitingState.java

```
public class WaitingState implements StateOfATM {
    private final ATM atm;

    public WaitingState(ATM atm) {
        this.atm = atm;
    }

    public void insertCard() {
        System.out.println("Card inserted");
        atm.setState(new AuthState(atm));
    }

    public void enterPIN_code(String pinCode) {
        System.out.println("Please insert your card first!");
    }

    public void withdrawMoney(int cashAmount) {
        System.out.println("Please insert your card first!");
    }

    public void getCard() {
        System.out.println("No card to eject!");
    }
}
```

AuthState.java

```
public class AuthState implements StateOfATM{
    private final ATM atm;

    public AuthState(ATM atm) {
        this.atm = atm;
    }

    public void insertCard() {
        System.out.println("Card already inserted!");
        atm.setState(new AuthState(atm));
    }

    public void enterPIN_code(String pinCode) {
        if (pinCode.equals(atm.PIN_code)) {
            System.out.println("PIN correct. You can now withdraw cash.");
            atm.setState(new TransactionState(atm));
        } else {
            System.out.println("Incorrect PIN.");
        }
    }

    public void withdrawMoney(int cashAmount) {
        System.out.println("Please enter your PIN first!");
    }

    public void getCard() {
        System.out.println("Card ejected!");
    }
}
```

TransactionState.java

```
public class TransactionState implements StateOfATM{
    private final ATM atm;

    public TransactionState(ATM atm) {
```

```
        this.atm = atm;
    }

    public void insertCard() {
        System.out.println("Card already inserted!");
        atm.setState(new AuthState(atm));
    }

    public void enterPIN_code(String pinCode) {
        System.out.println("PIN already entered");
    }

    public void withdrawMoney(int cashAmount) {
        if (cashAmount > atm.getAmountOfMoney()) {
            System.out.println("Insufficient funds!");
            atm.setState(new BlockState(atm));
            return;
        }
        atm.setAmountOfMoney(atm.getAmountOfMoney() - cashAmount);
        System.out.println("Withdrew " + cashAmount + " from your account.");
    }

    public void getCard() {
        System.out.println("Card ejected!");
        atm.setState(new WaitingState(atm));
    }
}
```

BlockState.java

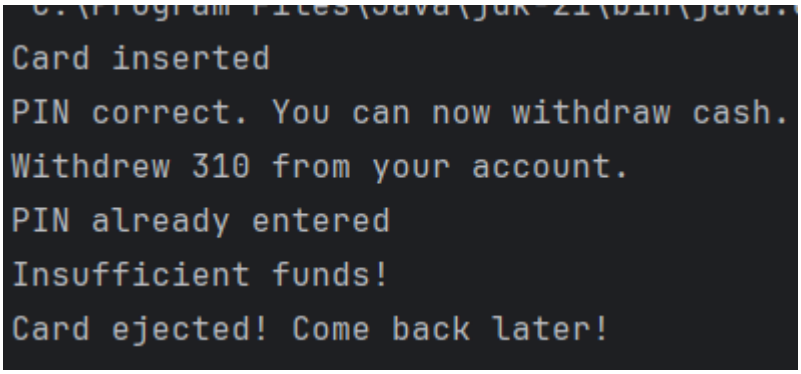
```
public class BlockState implements StateOfATM{
    private final ATM atm;

    public BlockState(ATM atm) {
        this.atm = atm;
    }

    public void insertCard() {
        System.out.println("ATM is blocked!");
        atm.setState(new AuthState(atm));
    }

    public void enterPIN_code(String pinCode) {
        System.out.println("ATM is blocked!");
    }

    public void withdrawMoney(int cashAmount) {
        System.out.println("ATM is blocked!");
    }

    public void getCard() {
        System.out.println("Card ejected! Come back later!");
        atm.setState(new WaitingState(atm));
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        ATM atm = new ATM(365, "1111");
        atm.insertCard();
        atm.enterPIN_code("1111");
        atm.withdrawMoney(310);
        atm.enterPIN_code("1111");
        atm.withdrawMoney(100);
```

```
        atm.getCard();
    }
}
```

Результаты работы программы:



Вывод: приобрел навыки применения паттернов проектирования при решении практических задач с использованием языка Java.