Министерство образования Республики Беларусь

Учреждение образования «Брестский государственный технический университет»

Кафедра ИИТ

Отчёт по лабораторной работе № 6

По дисциплине «Современные платформы программирования»

Выполнил:
студент 3-го курса
группы ПО-9(2)
Николайчик Н.С.
Проверил:
Крощенко А. А.

Брест, 2024

Вариант 5

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.

Варианты работ определяются по последней цифре в зачетной книжке

Задание 1:

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

Паттерном будет выбран Абстрактная фабрика

Код:

```
interface PhoneFactory {
    SmartPhone createSmartPhone();
}

interface SmartPhone {
    void displayInfo();
}

class AppleFactory implements PhoneFactory {
    public SmartPhone createSmartPhone() {
        return new iPhone();
    }
}

class SamsungFactory implements PhoneFactory {
    public SmartPhone createSmartPhone() {
        return new Galaxy();
    }
}

class iPhone implements SmartPhone {
    public void displayInfo() {
        System.out.println("This is an iPhone.");
    }
}

class Galaxy implements SmartPhone {
    public void displayInfo() {
        System.out.println("This is a Samsung Galaxy.");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        PhoneFactory appleFactory = new AppleFactory();
        SmartPhone iphone = appleFactory.createSmartPhone();
        iphone.displayInfo();
        PhoneFactory samsungFactory = new SamsungFactory();
        SmartPhone galaxy = samsungFactory.createSmartPhone();
        galaxy.displayInfo();
    }
}
```

Работа:

```
This is an iPhone.
This is a Samsung Galaxy.
```

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).

Использован будет паттерн адаптер.

Код:

```
public class AnalogThermometer {
    protected double top;
    protected double bottom;
    protected double height;
    protected double length;
    double indications(){
        return height;
    }
    AnalogThermometer(){
        this.top=42;
        this.bottom=35;
        this.length=10;
        this.height=5;
    }
    AnalogThermometer(double top,double bottom,double length,double
height){
        this.top=top;
        this.bottom=bottom;
        this.length=length;
        this.height=height;
    }

    public double getTop() {
        return top;
    }

    public double getBottom() {
```

```
        return bottom;
    }

    public double getHeight() {
        return height;
    }

    public double getLength() {
        return length;
    }
}
```

```
public class DigitalThermometer{
    AnalogThermometer analogThermometer;

    DigitalThermometer(){
        this.analogThermometer=new AnalogThermometer();
    }
    DigitalThermometer(AnalogThermometer analogThermometer){
        this.analogThermometer=analogThermometer;
    }
    DigitalThermometer(double top,double bottom,double height,double
length){
        this.analogThermometer=new AnalogThermometer( top, bottom,
height, length);
    }

    public void setAnalogThermometer(AnalogThermometer
analogThermometer) {
        this.analogThermometer = analogThermometer;
    }
    public double indication(){
        return
analogThermometer.indications()/analogThermometer.getLength()*
                (analogThermometer.getTop()-
analogThermometer.getBottom())+
                analogThermometer.getBottom();
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        AnalogThermometer analogThermometer=new AnalogThermometer();
        DigitalThermometer digitalThermometer=new
DigitalThermometer(analogThermometer);
        System.out.println(digitalThermometer.indication());
    }
```

```
}
```

Работа:

Градусник цифровой выдает показания на основе аналогового.

```
38.5
```

При границах от 35 до 42, длине 10 и высоте столбца в 5.

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.

Буду использовать паттерн состояние.

Код

```
public class ATM {
    private ATMState state;
    private int totalCash;
    public ATM(int totalCash) {
        this.totalCash = totalCash;
        this.state = new WaitingState(this);
    }
    public void setState(ATMState state) {
        this.state = state;
    }
    public void insertCard() {
        state.insertCard();
    }
    public void enterPin(int pin) {
        state.enterPin(pin);
    }
    public void withdrawCash(int amount) {
        state.withdrawCash(amount);
    }
    public void ejectCard() {
        state.ejectCard();
    }
    public int getTotalCash() {
        return totalCash;
    }
    public void setTotalCash(int totalCash) {
        this.totalCash = totalCash;
    }
    public boolean getPin() {
        return true;
    }
}
```

```
interface ATMState {
    void insertCard();
    void enterPin(int pin);
    void withdrawCash(int amount);
    void ejectCard();
}

class WaitingState implements ATMState {
    private ATM atm = null;
    public WaitingState(ATM atm){
        this.atm = atm;
    }
    public void insertCard() {
        System.out.println("Card inserted.");
        atm.setState(new AuthenticatedState(atm));
    }

    public void enterPin(int pin) {
        System.out.println("Please insert card first.");
    }

    public void withdrawCash(int amount) {
        System.out.println("Please insert card and enter pin first.");
    }

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



class AuthenticatedState implements ATMState {
    private ATM atm;

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

    public void insertCard() {
        System.out.println("Card already inserted.");
    }

    public void enterPin(int pin) {
        if (atm.getPin()) {
            System.out.println("PIN correct.");
            atm.setState(new TransactionState(atm));
        } else {
            System.out.println("Invalid PIN.");
            atm.setState(new BlockedState());
        }
    }

    public void withdrawCash(int amount) {
        System.out.println("Please enter PIN first.");
    }

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

```
class TransactionState implements ATMState {
    private ATM atm;

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

    public void insertCard() {
        System.out.println("Card already inserted.");
    }

    public void enterPin(int pin) {
        System.out.println("PIN already entered.");
    }

    public void withdrawCash(int amount) {
        if (amount <= atm.getTotalCash()) {
            System.out.println("Cash withdrawn: " + amount);
            atm.setTotalCash(atm.getTotalCash() - amount);
        } else {
            System.out.println("Not enough cash in ATM.");
        }
    }

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


class BlockedState implements ATMState {
    public void insertCard() {
        System.out.println("Card cannot be inserted. ATM blocked.");
    }

    public void enterPin(int pin) {
        System.out.println("PIN cannot be entered. ATM blocked.");
    }

    public void withdrawCash(int amount) {
        System.out.println("Cash cannot be withdrawn. ATM blocked.");
    }

    public void ejectCard() {
        System.out.println("Card cannot be ejected. ATM blocked.");
    }
}

public class Main {
    public static void main(String[] args) {
        ATM atm = new ATM(1000);
        atm.insertCard();
        atm.enterPin(1234);
        atm.withdrawCash(500);
        atm.withdrawCash(500000);
        atm.ejectCard();
    }
}
```

Работа:

```
Card inserted.
PIN correct.
Cash withdrawn: 500
Not enough cash in ATM.
Card ejected.
```

Вывод: паттерны позволяют использовать шаблоны для решения многих задач. Я попрактиковался в этом.