

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 строитель

Код:

```
public interface Builder {
    void setSmartphoneType(SmartphoneType type);
    void setName(String name);
    void setColor(Color color);
    void setNFC(NFC nfc);
    void setFingerprintScanner(FingerprintScanner fingerprintScanner);
}
class Color {
    private String color;

    public Color(String color) {
        this.setColor(color);
    }

    public void setColor(String color) {
        this.color = color;
    }

    public String getColor() {
        return this.color;
    }
}
```

```

public class Director {

    public void constructSmarthoneDAY(Builder builder) {
        builder.setSmartphoneType(SmartphoneType.SMARTPHONE_DAY);
        builder.setColor(new Color("white"));
        builder.setName("DAY");
        builder.setNFC(new NFC("4.1"));
    }

    public void constructSmarthoneNIGHT(Builder builder) {
        builder.setSmartphoneType(SmartphoneType.SMARTPHONE_NIGHT);
        builder.setColor(new Color("black"));
        builder.setName("NIGHT");
        builder.setFingerprintScanner(new FingerprintScanner("8.9"));
    }

    public void constructSmarthoneSTAR(Builder builder) {
        builder.setSmartphoneType(SmartphoneType.SMARTPHONE_STAR);
        builder.setColor(new Color("yellow"));
        builder.setName("STAR");
        builder.setNFC(new NFC("3.2"));
        builder.setFingerprintScanner(new FingerprintScanner("10.0"));
    }
}

class FingerprintScanner {
    private String version;

    public FingerprintScanner(String version){
        this.setScannerVersion(version);
    }

    public void setScannerVersion(String version) {
        this.version = version;
    }

    public String getScannerVersion() {
        return this.version;
    }
}

```

```

public class Manual {
    private SmartphoneType type;
    private String name;
    private Color color;
    private NFC nfc;

    public Manual (SmartphoneType type, String name, Color color, NFC
nfc,
    FingerprintScanner FingerprintScanner) {
        this.setSmartphoneType(type);
        this.setName(name);
        this.setColor(color);
        this.setNFC(nfc);
    }

    public void setSmartphoneType(SmartphoneType type){
        this.type = type;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setColor(Color color) {
        this.color = color;
    }

    public void setNFC(NFC nfc) {
        this.nfc = nfc;
    }

    public SmartphoneType setSmartphoneType(){
        return this.type;
    }

    public String setName() {
        return this.name;
    }

    public Color setColor() {
        return this.color;
    }

    public NFC setNFC() {
        return this.nfc;
    }

    public String print() {
        String info = "";
        info += "Type of smartphone: " + this.type + "\n";
        info += "Name: " + this.name + "\n";
        info += "Color: " + this.color.getColor() + "\n";
        if (this.nfc != null) {
            info += "NFC: " + this.nfc.getNFCVersion() + "\n";
        } else {
            info += "NFC: -" + "\n";
        }
        return info;
    }
}

```

```

    }
}
class NFC {
    private String version;

    public NFC(String version){
        this.setNFCVersion(version);
    }

    public void setNFCVersion(String version) {
        this.version = version;
    }

    public String getNFCVersion() {
        return this.version;
    }
}

```

```

public class Smartphone {
    private SmartphoneType type;
    private String name;
    private Color color;
    private NFC nfc;
    private FingerprintScanner FingerprintScanner;

    public Smartphone (SmartphoneType type, String name, Color color,
NFC nfc,
    FingerprintScanner FingerprintScanner) {
        this.setSmartphoneType(type);
        this.setName(name);
        this.setColor(color);
        this.setNFC(nfc);
        this.setFingerprintScanner(FingerprintScanner);
    }

    public void setSmartphoneType(SmartphoneType type){
        this.type = type;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setColor(Color color) {
        this.color = color;
    }

    public void setNFC(NFC nfc) {
        this.nfc = nfc;
    }
}

```

```

        public void setFingerprintScanner(FingerprintScanner
FingerprintScanner) {
            this.FingerprintScanner = FingerprintScanner;
        }

        public SmartphoneType setSmartphoneType(){
            return this.type;
        }

        public String setName() {
            return this.name;
        }

        public Color setColor() {
            return this.color;
        }

        public NFC setNFC() {
            return this.nfc;
        }

        public FingerprintScanner setFingerprintScanner() {
            return this.FingerprintScanner;
        }

        public String print() {
            String info = "";
            info += "Type of smartphone: " + this.type + "\n";
            info += "Name: " + this.name + "\n";
            info += "Color: " + this.color.getColor() + "\n";
            if (this.nfc != null) {
                info += "NFC: " + this.nfc.getNFCVersion() + "\n";
            } else {
                info += "NFC: -" + "\n";
            }
            if (this.FingerprintScanner != null) {
                info += "Fingureprint Scanner: " +
this.FingerprintScanner.getScannerVersion() + "\n";
            } else {
                info += "Fingureprint Scanner: -" + "\n";
            }
            return info;
        }
    }
}

```

```

public class SmartphoneBuilder implements Builder {
    private SmartphoneType type;
    private String name;
    private Color color;
    private NFC nfc;
    private FingerprintScanner FingerprintScanner;

    @Override
    public void setSmartphoneType(SmartphoneType type){
        this.type = type;
    }

    @Override
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public void setColor(Color color) {
        this.color = color;
    }

    @Override
    public void setNFC(NFC nfc) {
        this.nfc = nfc;
    }

    @Override
    public void setFingerprintScanner(FingerprintScanner
FingerprintScanner) {
        this.FingerprintScanner = FingerprintScanner;
    }

    public Smartphone getResult() {
        return new Smartphone(this.type, this.name, this.color,
this.nfc, this.FingerprintScanner);
    }
}

```

```

public class SmartphoneManualBuilder implements Builder {
    private SmartphoneType type;
    private String name;
    private Color color;
    private NFC nfc;
    private FingerprintScanner FingerprintScanner;

    @Override
    public void setSmartphoneType(SmartphoneType type){
        this.type = type;
    }

    @Override
    public void setName(String name) {

```

```

        this.name = name;
    }

    @Override
    public void setColor(Color color) {
        this.color = color;
    }

    @Override
    public void setNFC(NFC nfc) {
        this.nfc = nfc;
    }

    @Override
    public void setFingerprintScanner(FingerprintScanner
FingerprintScanner) {
        this.FingerprintScanner = FingerprintScanner;
    }

    public Manual getResult() {
        return new Manual(this.type, this.name, this.color, this.nfc,
this.FingerprintScanner);
    }
}
public enum SmartphoneType {
    SMARTPHONE_STAR, SMARTPHONE_DAY, SMARTPHONE_NIGHT
}

```



```

class Task_1{
    public static void main(String[] args) {
        Director director = new Director();

        SmartphoneBuilder builder = new SmartphoneBuilder();

        director.constructSmarthoneDAY(builder);

        Smartphone sm = builder.getResult();
        System.out.println(sm.print() + "\n");

        director.constructSmarthoneNIGHT(builder);

        sm = builder.getResult();
        System.out.println(sm.print() + "\n");

        director.constructSmarthoneSTAR(builder);

        sm = builder.getResult();
        System.out.println(sm.print() + "\n");
    }
}

```

Работа:

```

C:\Users\strau\.jdk\openjdk-21.0.2\bin\java.exe
Type of smartphone: SMARTPHONE_DAY
Name: DAY
Color: white
NFC: 4.1
Fingerprint Scanner: -

Type of smartphone: SMARTPHONE_NIGHT
Name: NIGHT
Color: black
NFC: 4.1
Fingerprint Scanner: 8.9

Type of smartphone: SMARTPHONE_STAR
Name: STAR
Color: yellow
NFC: 3.2
Fingerprint Scanner: 10.0

```

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 команда.

Код

```
public class ATM {
    public int ID;
    static public int ID_COUNTER=0;
    private double allMoney;
    Mode mode=Mode.WAIT;

    ATM(double allMoney) {
        this.allMoney = allMoney;
    }

    public double getAllMoney() {
        return allMoney;
    }

    public void setAllMoney(double allMoney) {
        this.allMoney = allMoney;
    }

    public void setMode(Mode mode){
        this.mode=mode;
    }

    public Mode getMode() {
        return mode;
    }

    static public boolean Verivication_prototype(int PIN){
        return true;
    }
}
```

```

public abstract class Command {
    public ATM atm;
    private double backup;

    Command(ATM atm) {
        this.atm = atm;
    }

    void backup() {
        backup = atm.getAllMoney();
    }

    public void undo() {
        atm.setAllMoney(backup);
    }

    public abstract boolean execute();
}
import java.util.Scanner;

public class CommandEnd extends Command{
    CommandEnd(ATM atm){
        super(atm);
    }
    @Override
    public boolean execute(){
        if(atm.getMode() != Mode.IN_PROCESS){
            return false;
        }

        System.out.println("Работа завершена");
        atm.setMode(Mode.WAIT);
        if(atm.getAllMoney() < 1){
            atm.setMode(Mode.BLOCKED);
            System.out.println("Денег нет, больше не выдаем");
        }
        return true;
    }
}

```

```

public class CommandEnterPIN extends Command{
    CommandEnterPIN(ATM atm){
        super(atm);
    }
    @Override
    public boolean execute(){
        if(atm.getMode()==Mode.BLOCKED){
            return false;
        }
        atm.setMode(Mode.AUTH);
        System.out.println("Введите пин");
        Scanner scanner=new Scanner(System.in);
        if(ATM.Verivication_prototype(scanner.nextInt())) {
            atm.setMode(Mode.IN_PROCESS);
            return true;
        }
        else {
            atm.setMode(Mode.WAIT);
            return false;
        }
    }
}

```

```

public class CommandGetMoney extends Command{
    CommandGetMoney(ATM atm){
        super(atm);
    }
    @Override
    public boolean execute(){
        if(atm.getMode()!=Mode.IN_PROCESS){
            System.out.println("Не можем дать денег");
            return false;
        }
        System.out.println("денег есть " + atm.getAllMoney());
        System.out.println("сколько хочешь снять?");
        Scanner scanner=new Scanner(System.in);
        double money = scanner.nextDouble();
        if(money> atm.getAllMoney()){
            atm.setMode(Mode.IN_PROCESS);
            System.out.println("Нет столько денег");
            if(atm.getAllMoney()<1){
                atm.setMode(Mode.BLOCKED);
                System.out.println("Денег нет, больше не выдаем");
            }
            return false;
        }
    }
}

```

```

    }
    else {
        atm.setAllMoney(atm.getAllMoney()-money);
        System.out.println("Вот ваши деньги");
        if(atm.getAllMoney()<1){
            atm.setMode(Mode.BLOCKED);
            System.out.println("Денег нет, больше не выдаем");
        }
        return true;
    }
}
}

```

```

public class CommandList {
    private Stack<Command> history = new Stack<>();

    public void push(Command c) {
        history.add(c);
    }

    Command remove() {
        return history.remove(0);
    }

    public boolean exec(){
        boolean temp = history.firstElement().execute();
        System.out.println(temp);
        this.remove();
        return temp;
    }

    public boolean isEmpty() { return history.isEmpty(); }
}

```

```

public enum Mode {

    WAIT ("Ожидание"),
    IN_PROCESS ("выполнение операции"),
    AUTH ("аутентификация"),
    BLOCKED ("блокировка");

    private String title;

    Mode(String title) {
        this.title = title;
    }

    public String getTitle() {
        return title;
    }

    @Override
    public String toString() {
        return title;
    }
}

public class Task_3 {
    public static void main(String[] args) {
        CommandList commandList = new CommandList();
        ATM atm = new ATM(150);
        commandList.push(new CommandGetMoney(atm));
        commandList.push(new CommandEnterPIN(atm));
        commandList.push(new CommandGetMoney(atm));
        commandList.push(new CommandGetMoney(atm));
        commandList.push(new CommandEnd(atm));
        commandList.push(new CommandGetMoney(atm));
        while (!commandList.isEmpty())
            commandList.exec();
    }
}

```


Работа:

```
Не можем дать денег
false
Введите пин
911
true
денег есть 150.0
сколько хочешь снять?
13
Вот ваши деньги
true
денег есть 137.0
сколько хочешь снять?
137
Вот ваши деньги
Денег нет, больше не выдаем
true
false
Не можем дать денег
false
```

Здесь сперва попытка снять без верификации

После снятие, завершение, закончились деньги и еще попытка снять.