Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №6**

По дисциплине: «Современные платформы программирования»

Выполнила:

Студентка 3 курса

Группы ПО-6

Юсковец М.А.

Проверил:

Монтик Н.С.

Брест, 2022

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

**Ход работы:**

Вариант 25

**•** 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.

**Задание 1:**

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

**Текст программы:**

**Main.java**

package pack;  
  
public class Main {  
 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");  
 }  
}

**Builder.java**

package pack;  
  
public interface Builder {  
 void setSmartphoneType(SmartphoneType type);  
 void setName(String name);  
 void setColor(Color color);  
 void setNFC(NFC nfc);  
 void setFingerprintScanner(FingerprintScanner FingerprintScanner);  
}

**Director.java**

package pack;  
  
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"));  
 }  
}

**SmartphoneBuilder.java**

package pack;  
  
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);  
 }  
}

**SmartphoneManualBuilder.java**

package pack;  
  
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);  
 }  
}

**Manual.java**

package pack;  
  
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;  
 }  
}

**SmartphoneType.java**

package pack;  
  
public enum SmartphoneType {  
 *SMARTPHONE\_STAR*, *SMARTPHONE\_DAY*, *SMARTPHONE\_NIGHT*}

**FingerprintScanner.java**

package pack;  
  
public 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;  
 }  
}

**NFC.java**

package pack;  
  
public 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;  
 }  
}

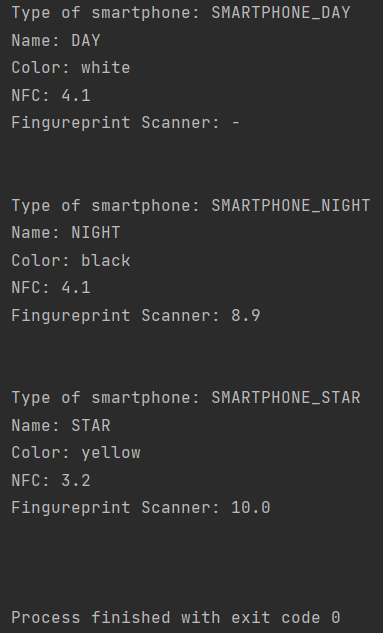
**Color.java**

package pack;  
  
public 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;  
 }  
}

**Smartphone.java**

package pack;  
  
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;  
 }  
}

**Результат 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).

**Текст программы:**

**Main.java**

package pack;  
  
public class Main {  
 public static void main(String[] args) {  
 Person person = new Person();  
 ElectronicThermometer electronic\_Thermometer = new ElectronicThermometer();  
 person.MeasureYourTemperature(electronic\_Thermometer);  
 MercuryThermometer mercuryThermometer = new MercuryThermometer();  
 Thermometer mercury\_Thermometer = new MercuryThermometerToElectronicThermometer(mercuryThermometer);  
 person.MeasureYourTemperature(mercury\_Thermometer);  
 }  
}

**AnalogThermometer.java**

package pack;  
  
public interface AnalogThermometer  
{  
 public void RoughlyMeasureTheTemperature();  
}

**Thermometer.java**

package pack;  
  
public interface Thermometer  
{  
 void MeasureTheTemperature();  
}

**ElectronicThermometer.java**

package pack;  
import java.util.Random;  
  
public class ElectronicThermometer implements Thermometer {  
 private int Temperature = 0;  
  
 public int GetTemperature() {  
 return Temperature;  
 }  
 public void SetTemperature(int temperature) {  
 Temperature = temperature;  
 }  
 public void MeasureTheTemperature() {  
 Random rand = new Random();  
 SetTemperature(rand.nextInt(35, 40));  
 System.*out*.println("Температура тела(эл. градусник): "+Temperature);  
 }  
}

**MercuryThermometer**

package pack;  
  
public class MercuryThermometer implements AnalogThermometer {  
 private int HeightOfTheMercuryColumn = 0;  
 private int UpperBound = 100, BottomLine = 0;  
  
 public int GetTemperature() {  
 return HeightOfTheMercuryColumn;  
 }  
 public void SetTemperature(int heightOfTheMercuryColumn) {  
 HeightOfTheMercuryColumn = heightOfTheMercuryColumn;  
 }  
  
 public int GetUpperBound() {  
 return UpperBound;  
 }  
 public void SetUpperBound(int upperBound) {  
 UpperBound = upperBound;  
 }  
  
 public int GetBottomLine() {  
 return HeightOfTheMercuryColumn;  
 }  
 public void SetBottomLine(int bottomLine) {  
 BottomLine = bottomLine;  
 }  
  
 public void RoughlyMeasureTheTemperature() {  
 System.*out*.println("Высота ртутного столба: "+HeightOfTheMercuryColumn  
 +". Нижняя граница градусника: "+BottomLine  
 +", верхняя граница градусника "+UpperBound);  
 }  
}

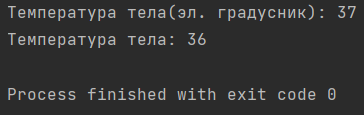
**MercuryThermometerToElectronicThermometer.java**

package pack;  
import java.util.Random;  
  
public class MercuryThermometerToElectronicThermometer implements Thermometer {  
 MercuryThermometer mercuryThermometer;  
  
 public MercuryThermometerToElectronicThermometer(MercuryThermometer thermometer) {  
 mercuryThermometer = thermometer;  
 }  
  
 public void MeasureTheTemperature() {  
 Random rand = new Random();  
 mercuryThermometer.SetTemperature(rand.nextInt(35, 40));  
 System.*out*.println("Температура тела: "+mercuryThermometer.GetTemperature());  
 }  
}

**Person.java**

package pack;  
  
public class Person  
{  
 public void MeasureYourTemperature(Thermometer thermometer)  
 {  
 thermometer.MeasureTheTemperature();  
 }  
}

**Результат программы:**



**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.

**Текст программы:**

**Main.java**

package pack;  
  
public class Main {  
 public static void main(String[] args) {  
 Person Bob = new Person("Bob", 1000, 1234);  
 ATM atm = new ATM(new Authentication(Bob));  
 atm.Expectation();  
 atm.Authentication();  
 atm.PerformingOperation();  
  
 }  
}

**ATM.java**

package pack;  
  
public class ATM  
{  
 private ATMState State;  
  
 public ATMState getState() {  
 return State;  
 }  
  
 public void setState(ATMState state) {  
 State = state;  
 }  
  
 public ATM(ATMState state) {  
 State = state;  
 }  
  
 public void Expectation() {  
 State.Expectation\_(this);  
 }  
  
 public void Authentication() {  
 State.Authentication\_(this);  
 }  
  
 public void PerformingOperation() {  
 State.PerformingOperation\_(this);  
 }  
  
 public void Blocking() {  
 State.Blocking\_(this);  
 }  
}

**ATMState.java**

package pack;  
  
public interface ATMState  
{  
 void Expectation\_(ATM atm);  
 void Authentication\_(ATM atm);  
 void PerformingOperation\_(ATM atm);  
 void Blocking\_(ATM atm);  
}

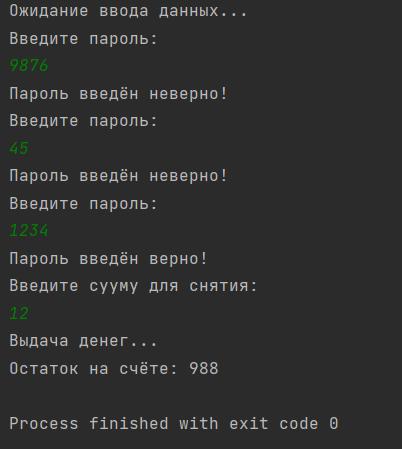
**Authentication.java**

package pack;  
import java.util.Scanner;  
  
public class Authentication implements ATMState {  
 Person person;  
  
 public Authentication(Person per) {  
 person = per;  
 }  
 public void Expectation\_(ATM atm) {  
 if (person.GetIsBlocked() == true) {  
 Blocking\_(atm);  
 return;  
 }  
 System.*out*.println("Ожидание ввода данных...");  
 }  
 public void Authentication\_(ATM atm) {  
 Scanner scanner = new Scanner(System.*in*);  
 if (person.GetIsBlocked() == true) {  
 Blocking\_(atm);  
 return;  
 }  
 int pin, numTry = 0;  
  
 do {  
 System.*out*.println("Введите пароль:");  
 pin = scanner.nextInt();  
 if (person.GetPassword() == pin) {  
 System.*out*.println("Пароль введён верно!");  
 return;  
 } else {  
 System.*out*.println("Пароль введён неверно!");  
 numTry++;  
 if (numTry == 3) {  
 Blocking\_(atm);  
 return;  
 }  
 }  
 }  
 while (person.GetPassword() != pin);  
 }  
 public void PerformingOperation\_(ATM atm) {  
 if (person.GetIsBlocked() == true) {  
 Blocking\_(atm);  
 return;  
 }  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введите сууму для снятия: ");  
 int sum = scanner.nextInt();  
  
 if (sum > person.GetBill()) {  
 System.*out*.println("Недостаточно денег на счёте!");  
 }  
 if (sum <= person.GetBill()) {  
 person.SetBill(person.GetBill() - sum);  
 System.*out*.println("Выдача денег...");  
 System.*out*.println("Остаток на счёте: " + person.GetBill());  
 }  
 }  
 public void Blocking\_(ATM atm) {  
 person.SetIsBlocked(true);  
 System.*out*.println("Ваша карта заблокирована!");  
 }  
 }

**Person.java**

package pack;  
  
public class Person  
{  
 String Name;  
 int Bill, Password;  
 Boolean isBlocked = false;  
  
 public Person(String name, int bill, int password) {  
 Name = name;  
 Bill = bill;  
 Password = password;  
 }  
 public void SetName(String name) {  
 Name = name;  
 }  
  
 public String GetName() {  
 return Name;  
 }  
 public void SetBill(int bill) {  
 Bill = bill;  
 }  
 public int GetBill() {  
 return Bill;  
 }  
  
 public void SetPassword(int password) {  
 Password = password;  
 }  
 public int GetPassword() {  
 return Password;  
 }  
 public void SetIsBlocked(Boolean is\_Blocked) {  
 isBlocked = is\_Blocked;  
 }  
 public Boolean GetIsBlocked() {  
 return isBlocked;  
 }  
}

**Результат программы:**



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