Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный техническийуниверситет»

Кафедра ИИТ

Лабораторнаяработа№6

По дисциплине

“Современные платформы программирования”

Выполнил:

Студент3курса

Группы ПО-8

Бубен С.О.

Проверил:

Крощенко А.А.

Брест2024

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

Ход работы

Вариант 2

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

Для данного задания воспользуемся порождающим паттерном Фабричный методдлясозданияобъектовбезуказанияихконкретныхклассов.Фабричныйметод 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:

Класс Смартфон:

```
publicabstractclassSmartphone{
   protected String model;
   protected int productionYear;
   protected int batteryCapacity;
   protected double screenSize;

   publicSmartphone(Stringmodel,intproductionYear,intbatteryCapacity,doublescreenSize){
      this.model = model;
      this.productionYear = productionYear;
      this.batteryCapacity=batteryCapacity;
      this.screenSize=screenSize;
   }

   publicabstractvoidprintInfo();
}
```

Класс смартфонов Bluemi:

```
publicclassBluemiextendsSmartphone{
   publicBluemi(Stringmodel,intproductionYear,intbatteryCapacity,doublescreenSize){ super(model,
      productionYear, batteryCapacity, screenSize);
   }

   @Override
   publicvoidprintInfo(){
      System.out.println("СмартфонBluemiмодели"+super.model
            +"\nГодвыпуска:"+super.productionYear
            +"\nЁмкостьбатерии:"+super.batteryCapacity+"мА·ч"
```

```
            +"\nРазмерэкрана:"+super.screenSize+"\"");
    }
}
```

Класс смартфонов ISmartphone:

```
publicclassISmartphoneextendsSmartphone{
    publicISmartphone(Stringmodel,intproductionYear,intbatteryCapacity,doublescreenSize){
        super(model, productionYear, batteryCapacity, screenSize);
    }

    @Override
    publicvoidprintInfo(){
        System.out.println("СмартфонISmartphoneмодели"+super.model
                +"\nГодвыпуска:"+super.productionYear
                +"\nЁмкостьбатерии:"+super.batteryCapacity+"мА·ч"
                +"\nРазмерэкрана:"+super.screenSize+"\"");
    }
}
```

Класс Фабрики создания смартфонов:

```
publicabstractclassSmartphoneCreator{
    publicabstractSmartphonecreateSmartphone();
}
```

Класс Фабрики создания смартфонов Bluemi:

```
publicclassBluemiCreatorextendsSmartphoneCreator{ @Override
    publicSmartphonecreateSmartphone(){returnnewBluemi("12Pro",2022,4500,6.5);}
}
```

Класс Фабрики создания смартфонов ISmartphone:

```
publicclassISmartphoneCreatorextendsSmartphoneCreator{
    @Override
    publicSmartphonecreateSmartphone(){returnnewISmartphone("2",2009,3200,4.5);}
}
```

Класс с методом main:

```
publicclassMain{
    publicstaticvoidmain(String[]args){
        BluemiCreator bluemiCreator = new BluemiCreator();
        Smartphonebluemi=bluemiCreator.createSmartphone();
        bluemi.printInfo();

        System.out.println("");
        ISmartphoneCreatoriSmartphoneCreator=newISmartphoneCreator();
        Smartphone iSmartphone = iSmartphoneCreator.createSmartphone();
        iSmartphone.printInfo();
    }
}
```

Результаты работы программы:

```
Смартфон Bluemi модели 12 Pro
Год выпуска: 2022
Ёмкость батерии: 4500 мА·ч
Размер экрана: 6.5"

Смартфон ISmartphone модели 2
Год выпуска: 2009
Ёмкость батерии: 3200 мА·ч
Размер экрана: 4.5"
```

Задание 2. Проект «Электронный градусник». В проекте должен быть реализованкласс,которыйдаетвозможностьпользоватьсяаналоговымградусником 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 рассчитанклиент.Вданномслучаемыможемсоздатьадаптер,которыйпреобразует интерфейс аналогового градусника в интерфейс электронного градусника.

Код программы:

Интерфейс Электронный градусник:

```
publicinterfaceElectronicThermometer{
   double getTemperature();
}
```

Класс Аналоговый градусник:

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

   publicAnalogThermometer(doublelowerLimit,doubleupperLimit,doublemercuryHeight,double
graduationScale) {
      this.lowerLimit = lowerLimit;
      this.upperLimit = upperLimit;
      this.mercuryHeight = mercuryHeight;
      this.graduationScale=graduationScale;
   }

   publicdoublegetLowerLimit(){ returnlowerLimit;}

   publicdoublegetUpperLimit(){returnupperLimit;}
```

```
    publicdoublegetMercuryHeight(){
        return mercuryHeight;
    }

    publicdoublegetGraduationScale(){ return
        graduationScale;
    }

    publicvoidsetMercuryHeight(doublemercuryHeight){ if
        (mercuryHeight <0) {
            mercuryHeight=lowerLimit;
            return;
        }

        if(lowerLimit+mercuryHeight/graduationScale>upperLimit){
            mercuryHeight = lowerLimit;
            return;
        }

        this.mercuryHeight=mercuryHeight;
    }
}
```

Класс Адаптер:

```
publicclassThermometerAdapterimplementsElectronicThermometer{ private
    AnalogThermometer analogThermometer;
    publicThermometerAdapter(AnalogThermometeranalogThermometer){
        this.analogThermometer = analogThermometer;
    }

    @java.lang.Override
    publicdoublegetTemperature(){
        returnanalogThermometer.getLowerLimit()+analogThermometer.getMercuryHeight()/
analogThermometer.getGraduationScale();
    }
}
```

Класс с методом main:

```
publicclassMain{
    publicstaticvoidmain(String[]args){
        ThermometerAdapterthermometerAdapter=newThermometerAdapter( new
                AnalogThermometer(35.0, 42.0, 10.0, 2.0));
        System.out.println("Температуранаградуснике:"+thermometerAdapter.getTemperature()+"Cº");
    }
}
```

Результаты работыпрограммы:

```
Температура на градуснике: 40.0 C°
```

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:

Интерфейс Состояние банкомата:

```
interfaceATMState{
   void insertCard();
   void ejectCard();
   void enterPinCode(int pinCode);
   voidrequestCash(intcashAmount);
}
```

Класс Состояние ожидания:

```
publicclassATMWaitingStateimplementsATMState{ private
   final ATM atm;

   publicATMWaitingState(ATMatm){
      this.atm = atm;
   }

   @java.lang.Override
   publicvoidinsertCard(){
      System.out.println("Карта вставлена!");
      atm.setCurrentState(newATMAuthenticationState(atm));
   }

   @java.lang.Override
   publicvoidejectCard(){
      System.out.println("Невозможноизвлечькарту-картаневставлена!");
   }

   @java.lang.Override
   public void enterPinCode(int pinCode) {
      System.out.println("Картаневставлена!");
   }

   @java.lang.Override
   publicvoidrequestCash(intcashAmount){
      System.out.println("Вставьтесначалакартуивведитепин-код!");
   }
}
```

Класс Состояние аутентификации:

```
publicclassATMAuthenticationStateimplementsATMState{ private
   final ATM atm;

   publicATMAuthenticationState(ATMatm){ this.atm
      = atm;
   }

   @java.lang.Override
   publicvoidinsertCard(){
      System.out.println("Картаужевставлена!");
   }

   @java.lang.Override
   publicvoidejectCard(){
      System.out.println("Карта извлечена!");
      atm.setCurrentState(newATMWaitingState(atm));
   }

   @java.lang.Override
   public void enterPinCode(int pinCode) {
      System.out.println("Пин-кодвведен!");
      atm.setCurrentState(newATMOperationPerformingState(atm));
   }

   @java.lang.Override
   public void requestCash(int cashAmount) {
      System.out.println("Введитесначалапин-код!");
   }
}
```

Класс Состояние выполнения операции:

```
publicclassATMOperationPerformingStateimplementsATMState{ private
   final ATM atm;

   publicATMOperationPerformingState(ATMatm){
      this.atm = atm;
   }

   @java.lang.Override
   publicvoidinsertCard(){
      System.out.println("Картаужевставлена!");
   }

   @java.lang.Override
   publicvoidejectCard(){
      System.out.println("Карта извлечена!");
      atm.setCurrentState(newATMWaitingState(atm));
   }

   @java.lang.Override
   publicvoidenterPinCode(intpinCode){
```

```
        System.out.println("Пин-кодужевведен!");
    }

    @java.lang.Override
    publicvoidrequestCash(intcashAmount){ if
        (cashAmount >atm.getTotalCash()) {
            System.out.println("Недостаточносредствнакарте!");
            atm.setCurrentState(new ATMBlockingState(atm));
            return;
        }
        intremainingCash=atm.getTotalCash()-cashAmount;
        atm.setTotalCash(remainingCash);
        System.out.println("Операциявыполнена.Снято"+remainingCash);
    }
}
```

Класс Состояние блокировки:

```
publicclassATMBlockingStateimplementsATMState{ private
    final ATM atm;

    publicATMBlockingState(ATMatm){
        this.atm = atm;
    }

    @java.lang.Override
    publicvoidinsertCard(){
        System.out.println("Банкоматзаблокирован!");
    }

    @java.lang.Override
    publicvoidejectCard(){
        System.out.println("Банкоматразблокирован,картаизвлечена!");
        atm.setCurrentState(new ATMWaitingState(atm));
    }

    @java.lang.Override
    public void enterPinCode(int pinCode) {
        System.out.println("Банкоматзаблокирован!");
    }

    @java.lang.Override
    public void requestCash(int cashAmount) {
        System.out.println("Банкоматзаблокирован!");
    }
}
```

Класс Банкомат:

```
public class ATM {
    privateinttotalCash;
    private int pinCode;
    privateATMStatecurrentState;
```

```
    public ATM(int totalCash) {
        this.totalCash=totalCash;
        currentState=newATMWaitingState(this);
    }

    publicvoidsetCurrentState(ATMStatecurrentState){this.currentState=currentState;} public

    void insertCard() {
        currentState.insertCard();
    }

    public void ejectCard() {
        currentState.ejectCard();
    }

    publicvoidenterPinCode(intpinCode){
        this.pinCode = pinCode;
        currentState.enterPinCode(pinCode);
    }

    publicvoidrequestCash(intcashAmount){
        currentState.requestCash(cashAmount);
    }

    publicvoidsetTotalCash(inttotalCash){this.totalCash=totalCash;}

    publicintgetTotalCash(){returntotalCash;}
}
```

Класс с методом main:

```
publicclassMain{
    publicstaticvoidmain(String[]args){
        ATM atm = new ATM(100);
        atm.insertCard();
        atm.enterPinCode(1234);
        atm.requestCash(50);
        atm.requestCash(60);
        atm.ejectCard();
    }
}
```

Результаты работы программы:

```
Карта вставлена!
Пин-код введен!
Операция выполнена. Снято 50
Недостаточно средств на карте!
Банкомат разблокирован, карта извлечена!
```

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