Федеральное государственное автономное образовательное учреждение

высшего образования

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа «Киберфизические системы и управление»

**Отчет по итоговому проекту**

по дисциплине «Системный подход к разработке программного обеспечения»

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. 3530902/00201 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Позолотин О. В. |
|  | <*подпись*> |  |
|  |  |  |
|  |  |  |
| Руководитель:  Кандидат т.н. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Нестеров С. А. |
|  | <*подпись*> |  |

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2022 г.

Санкт-Петербург

2022

**Постановка задачи**

Тема проекта – продуктовый магазин.

Пользователь покупает товар у поставщиков, назначая им свою цену в своём магазине, товары и у пользователя и у поставщика хранятся на складе, за покупку и продажу отвечает терминал, также присутствуем окно для вывода сообщении об имени клиента и его уровне удовлетворённости.

Use-case диаграмма проекта

Diagram, schematic

Description automatically generated

Диаграмма классов

Graphical user interface

Description automatically generated

Код проекта

**Main**

package com.example.project;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class Main extends Application {  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("main\_page.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 stage.setTitle("Hello!");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

**MainPageController**

package com.example.project;  
  
import com.example.focus.\*;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextArea;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.\*;  
  
import java.util.\*;  
  
public class MainPageController {  
 public VBox deliversVBox;  
 public TextField moneyTextField;  
 public TextArea loggerTextArea;  
 public VBox vBoxBreadDelivery;  
 public VBox vBoxConfectioneryDelivery;  
 public VBox vBoxDairyDelivery;  
 public VBox vBoxMeatDelivery;  
 public VBox vBoxFishDelivery;  
 public VBox vBoxVegetableDelivery;  
 public VBox vBoxFruitDelivery;  
 public VBox vBoxBeverageDelivery;  
 public VBox vBoxBreadUser;  
 public VBox vBoxConfectioneryUser;  
 public VBox vBoxDairyUser;  
 public VBox vBoxMeatUser;  
 public VBox vBoxFishUser;  
 public VBox vBoxVegetableUser;  
 public VBox vBoxFruitUser;  
 public VBox vBoxBeverageUser;  
 public static ArrayList<VBox> *deliversGoodsVBox* = new ArrayList<>();  
 public static ArrayList<VBox> *usersGoodsVBox* = new ArrayList<>();  
 public TextField fieldDepartmentInfo;  
 public TextField fieldProductInfo;  
 public TextField fieldCostInfo;  
 public TextField fieldCountInput;  
 public TextField fieldDeliverInfo;  
 public Button buttonBuy;  
 public TextField fieldNewCostInput;  
 public TextField ratingTextField;  
 public AnchorPane back;  
 public Pane backUserStorage;  
 public Pane backDeliversStorage;  
  
 private void setUpVBox(ArrayList<VBox> listGoods, VBox vBoxBreadUser, VBox vBoxConfectioneryUser, VBox vBoxDairyUser, VBox vBoxMeatUser, VBox vBoxFishUser, VBox vBoxVegetableUser, VBox vBoxFruitUser, VBox vBoxBeverageUser) {  
 listGoods.add(vBoxBreadUser);  
 listGoods.add(vBoxConfectioneryUser);  
 listGoods.add(vBoxDairyUser);  
 listGoods.add(vBoxMeatUser);  
 listGoods.add(vBoxFishUser);  
 listGoods.add(vBoxVegetableUser);  
 listGoods.add(vBoxFruitUser);  
 listGoods.add(vBoxBeverageUser);  
 }  
 public void initialize() {  
  
 setUpVBox(*deliversGoodsVBox*, vBoxBreadDelivery, vBoxConfectioneryDelivery, vBoxDairyDelivery, vBoxMeatDelivery,  
 vBoxFishDelivery, vBoxVegetableDelivery, vBoxFruitDelivery, vBoxBeverageDelivery);  
  
 setUpVBox(*usersGoodsVBox*, vBoxBreadUser, vBoxConfectioneryUser, vBoxDairyUser, vBoxMeatUser, vBoxFishUser,  
 vBoxVegetableUser, vBoxFruitUser, vBoxBeverageUser);  
  
 ObjectSwamp.*loadClass*();  
 Timer myTimer;  
 myTimer = new Timer();  
 myTimer.schedule(new TimerTask() {  
 public void run() {  
 timerTick();  
 }  
 }, 0, 1000); // каждые 5 секунд  
 for(String deliver : ObjectSwamp.*delivers*.keySet()) {  
 Button buttonDeliverToAdd = new Button(ObjectSwamp.*delivers*.get(deliver).getName());  
 buttonDeliverToAdd.setPrefWidth(deliversVBox.getPrefWidth());  
 buttonDeliverToAdd.setOnAction(actionEvent -> {  
 ObjectSwamp.*currentDeliver* = ObjectSwamp.*delivers*.get(deliver);  
 for (VBox vBox : *deliversGoodsVBox*) {  
 vBox.getChildren().clear();  
 }  
 for (VBox vBox : *usersGoodsVBox*) {  
 vBox.getChildren().clear();  
 }  
 for (int i = 0; i < *deliversGoodsVBox*.size(); i++){  
 Storage currentDeliverStorage = ObjectSwamp.*delivers*.get(deliver).storage;  
 Storage currentUserStorage = ObjectSwamp.*myStorage*;  
 setUpVBox(i, *deliversGoodsVBox*);  
 setUpVBox(i, *usersGoodsVBox*);  
 for(int j = 0; j < 3; j++){  
 String currentDivision = ObjectSwamp.*divisionsName*.get(i);  
 setUpDeliverGoods(i, j, deliver, currentDeliverStorage, currentDivision);  
 setUpUserGoods(i, j, currentUserStorage, currentDivision);  
 }  
 }  
 });  
 deliversVBox.getChildren().add(buttonDeliverToAdd);  
 }  
 moneyTextField.setText(Integer.*toString*(ObjectSwamp.*myTerminal*.getMoney()));  
 }  
 private void setUpDeliverGoods(int i, int j, String deliver, Storage currentDeliverStorage, String currentDivision) {  
 HBox hBoxDeliver = new HBox();  
 Button buttonGoodDeliver = new Button(currentDeliverStorage.goodsByDivision.get(currentDivision).get(j).getName()  
 + currentDeliverStorage.goodsByDivision.get(currentDivision).get(j).getExtraInfo());  
 Label labelCostDeliver = new Label(Integer.*toString*(currentDeliverStorage.goodsByDivision.get(currentDivision).get(j).getOldCost()));  
 Label labelCountDeliver = new Label(Integer.*toString*(currentDeliverStorage.goodsByDivision.get(currentDivision).get(j).getCount()));  
 buttonGoodDeliver.setOnAction(actionEvent1 -> {  
 ObjectSwamp.*currentGood* = currentDeliverStorage.goodsByDivision.get(currentDivision).get(j);  
 ObjectSwamp.*currentDepartment* = currentDivision;  
 fieldDeliverInfo.setText(ObjectSwamp.*delivers*.get(deliver).getName());  
 fieldDepartmentInfo.setText(currentDivision);  
 fieldProductInfo.setText(buttonGoodDeliver.getText());  
 fieldCostInfo.setText(labelCostDeliver.getText());  
 });  
 buttonGoodDeliver.setPrefWidth(110);  
 labelCostDeliver.setPrefWidth(40);  
 hBoxDeliver.getChildren().add(buttonGoodDeliver);  
 hBoxDeliver.getChildren().add(labelCostDeliver);  
 hBoxDeliver.getChildren().add(labelCountDeliver);  
 *deliversGoodsVBox*.get(i).getChildren().add(hBoxDeliver);  
 }  
 private void setUpUserGoods(int i, int j, Storage currentUserStorage, String currentDivision){  
 HBox hBoxUser = new HBox();  
 Button buttonGoodUser = new Button(currentUserStorage.goodsByDivision.get(currentDivision).get(j).getName()  
 + currentUserStorage.goodsByDivision.get(currentDivision).get(j).getExtraInfo());  
 Label labelCostUser = new Label(Integer.*toString*(currentUserStorage.goodsByDivision.get(currentDivision).get(j).getNewCost()));  
 Label labelCountUser = new Label(Integer.*toString*(currentUserStorage.goodsByDivision.get(currentDivision).get(j).getCount()));  
 buttonGoodUser.setOnAction(actionEvent1 -> {  
  
 });  
 buttonGoodUser.setPrefWidth(110);  
 labelCostUser.setPrefWidth(40);  
 hBoxUser.getChildren().add(buttonGoodUser);  
 hBoxUser.getChildren().add(labelCostUser);  
 hBoxUser.getChildren().add(labelCountUser);  
 *usersGoodsVBox*.get(i).getChildren().add(hBoxUser);  
 }  
 private void setUpVBox(int i, ArrayList<VBox> goodsVBox) {  
 HBox hBoxTitleUser = new HBox();  
 Button buttonGoodTitleUser = new Button("Product");  
 Label labelCostTitleUser = new Label("Cost");  
 Label labelCountTitleUser = new Label("Count");  
 buttonGoodTitleUser.setDisable(true);  
 buttonGoodTitleUser.setPrefWidth(110);  
 labelCostTitleUser.setPrefWidth(40);  
 hBoxTitleUser.getChildren().add(buttonGoodTitleUser);  
 hBoxTitleUser.getChildren().add(labelCostTitleUser);  
 hBoxTitleUser.getChildren().add(labelCountTitleUser);  
 goodsVBox.get(i).getChildren().add(hBoxTitleUser);  
 }  
  
 public void buttonBuyClick(){  
 int count = 0;  
 int newCost = 0;  
 try{  
 count = Integer.*parseInt*(fieldCountInput.getText());  
 newCost = Integer.*parseInt*(fieldNewCostInput.getText());  
 ArrayList<Good> myGoods = ObjectSwamp.*myStorage*.goodsByDivision.get(ObjectSwamp.*currentDepartment*);  
 for (Good myGood : myGoods) {  
 if (myGood.getName().equals(ObjectSwamp.*currentGood*.getName())) {  
 if (ObjectSwamp.*currentGood*.getCount() >= count){  
 if (ObjectSwamp.*myTerminal*.getMoney() >= ObjectSwamp.*currentGood*.getCount()  
 \* ObjectSwamp.*currentGood*.getOldCost()){  
  
 ObjectSwamp.*myTerminal*.decreaseMoney(ObjectSwamp.*currentGood*.getCount()  
 \* ObjectSwamp.*currentGood*.getOldCost());  
 ObjectSwamp.*currentGood*.setCount(ObjectSwamp.*currentGood*.getCount() - count);  
 myGood.setNewCost(newCost);  
 myGood.setCount(myGood.getCount() + count);  
 for (VBox vBox : *deliversGoodsVBox*) {  
 vBox.getChildren().clear();  
 }  
 for (VBox vBox : *usersGoodsVBox*) {  
 vBox.getChildren().clear();  
 }  
 for (int i = 0; i < *deliversGoodsVBox*.size(); i++){  
 setUpVBox(i, *deliversGoodsVBox*);  
 setUpVBox(i, *usersGoodsVBox*);  
 for(int j = 0; j < 3; j++){  
 String currentDivision = ObjectSwamp.*divisionsName*.get(i);  
 setUpDeliverGoods(i, j, ObjectSwamp.*currentDeliver*.getName(), ObjectSwamp.*currentDeliver*.storage, currentDivision);  
 setUpUserGoods(i, j, ObjectSwamp.*myStorage*, currentDivision);  
 }  
 }  
 moneyTextField.setText(Integer.*toString*(ObjectSwamp.*myTerminal*.getMoney()));  
 } else {  
 loggerTextArea.setText("\nNot enough money!!!");  
 }  
 } else {  
 loggerTextArea.setText("\nToo much!!!");  
 }  
 if (ObjectSwamp.*currentGood*.getCount() == 0){  
 ObjectSwamp.*currentGood*.setCount(100);  
 }  
 }  
 }  
 } catch (Exception exception){  
 loggerTextArea.setText("\nInvalid input!");  
 } finally {  
 fieldDeliverInfo.setText("");  
 fieldDepartmentInfo.setText("");  
 fieldProductInfo.setText("");  
 fieldCostInfo.setText("");  
 fieldCountInput.setText("");  
 fieldNewCostInput.setText("");  
 }  
 }  
 public int timerTickIndex = 0;  
 public void timerTick() {  
 timerTickIndex++;  
 if(timerTickIndex % 10 == 0) {  
 loggerTextArea.setText("");  
 Random randClientAmount = new Random();  
 int randAmount = randClientAmount.nextInt(6) + 2;  
 for(int i = 0; i < randAmount; i++){  
 Client client = new Client();  
 loggerTextArea.setText(loggerTextArea.getText() + client.getName() + " " + client.getRating() + "\n");  
 }  
 ratingTextField.setText(String.*format*("%.3f", ObjectSwamp.*rating*));  
 }  
 }  
}

**Client**

package com.example.focus;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Random;  
  
public class Client {  
 private final String name;  
 private double rating;  
 public Client(){  
 Random rand = new Random();  
 ArrayList<String> names = new ArrayList<>(Arrays.*asList*("Mark", "Joe", "Jeffry", "Ten", "Sixty", "Robbie",  
 "Bobbie", "Barrie", "Bilbo", "Roy", "Jack"));  
 int index = rand.nextInt(names.size());  
 this.name = names.get(index);  
 Random choiceDepartmentRand = new Random();  
 int rateWant = 0;  
 int rateGain = 0;  
 for(int i = 0; i < ObjectSwamp.*divisionsName*.size(); i++){  
 int choiceDepartment = choiceDepartmentRand.nextInt(2);  
 if (choiceDepartment == 1) {  
 String departmentName = ObjectSwamp.*divisionsName*.get(i);  
 Random choiceGoodRand = new Random();  
 for (int j = 0; j < 3; j++){  
 int choiceGood = choiceGoodRand.nextInt(2);  
 if (choiceGood == 1) {  
 rateWant++;  
 Random amountRand = new Random();  
 int amount = amountRand.nextInt(6) + 1;  
 int amountOnStorage = ObjectSwamp.*myStorage*.goodsByDivision.get(departmentName).get(j).getCount();  
 int costOnStorage = ObjectSwamp.*myStorage*.goodsByDivision.get(departmentName).get(j).getNewCost();  
 if (amountOnStorage >= amount){  
 rateGain++;  
 ObjectSwamp.*myTerminal*.decreaseMoney(false, amount \* costOnStorage);  
 ObjectSwamp.*myStorage*.goodsByDivision.get(departmentName).get(j).setCount(amountOnStorage - amount);  
 }  
 }  
 }  
 }  
 if (rateWant != 0) {  
 this.rating = (double) rateGain / rateWant;  
 ObjectSwamp.*refreshRating*(this.rating);  
 }  
 }  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public double getRating() {  
 return rating;  
 }  
}

**Deliver**

package com.example.focus;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Random;  
  
public class Deliver {  
 private final String name;  
 public final Storage storage;  
 public Deliver(String name){  
 this.name = name;  
 this.storage = new Storage(true);  
 System.*out*.println("");  
 }  
 public String getName() {  
 return name;  
 }  
  
 @Override  
 public String toString(){  
 return this.name;  
 }  
}

**Good**

package com.example.focus;  
  
public class Good {  
 private final String division;  
 private final String name;  
 private final String extraInfo;  
 private int count;  
 private final int oldCost;  
 private int newCost;  
 public Good(String division, String name, String extraInfo, int cost, int count) {  
 this.division = division;  
 this.name = name;  
 this.extraInfo = extraInfo;  
 this.oldCost = cost;  
 this.count = count;  
 }  
  
 public String getName(){  
 return this.name;  
 }  
 public int getNewCost(){  
 return this.newCost;  
 }  
 public int getOldCost(){  
 return this.oldCost;  
 }  
  
 public int getCount() {  
 return count;  
 }  
  
 public String getExtraInfo() {  
 return extraInfo;  
 }  
  
 public void setNewCost(int newCost) {  
 this.newCost = newCost;  
 }  
 public void setCount(int count) {  
 this.count = count;  
 }  
  
 @Override  
 public String toString(){  
 return this.name + " " + this.count;  
 }  
}

**Card**

package com.example.classes;  
  
// класс банковской карты  
public class Card {  
 // Данные карты  
 private final int password;  
 private final int cardNumber;  
 private int money;  
 private final String date;  
  
 // конструктор класса  
 public Card(int password,int money, int cardNumber, String date){  
 this.password = password;  
 this.cardNumber = cardNumber;  
 this.money = money;  
 this.date = date;  
 }  
 // сеттеры для суммы на карте  
 public void setMoney(int money) {  
 this.money = money;  
 }  
  
 // геттеры для всех полей класса  
 public int getMoney() {  
 return money;  
 }  
 public int getPassword() {  
 return password;  
 }  
 public int getCardNumber() {  
 return cardNumber;  
 }  
 public String getDate() {  
 return date;  
 }  
}

**ObjectSwamp**

package com.example.focus;  
  
import java.util.\*;  
  
public class ObjectSwamp {  
 public static final ArrayList<String> *divisionsName* = new ArrayList<> (Arrays.*asList*("Bread department",  
 "Confectionery department", "Dairy department", "Meat section", "Fish department", "Vegetable department",  
 "Fruit department", "Beverage department"));  
 public static final ArrayList<String> deliverName = new ArrayList<> (Arrays.asList("Jemmie Lannister",  
 "Christofer Nolan", "Mark Twen", "Chris Evans", "Bobby Dylan", "John Snow", "Ilya Muromets",  
 "Vasya Perevalov", "Shrek Swampov", "Tony Start", "Gandalf White", "Bilbo Begins"));  
 public static HashMap<String, Deliver> delivers = new HashMap<>();  
 public static Deliver currentDeliver;  
 public static String currentDepartment;  
 public static Good currentGood;  
  
 public static Terminal myTerminal;  
 public static Storage myStorage;  
 public static double rating = 0;  
 public static int amountRatings = 1;  
 public static double sumRatings = 0.5;  
 static {  
 Random rand\_deliver = new Random();  
 for (int i = 0; i < 9; i++) {  
 int rand = rand\_deliver.nextInt(deliverName.size());  
 delivers.put(deliverName.get(rand), new Deliver(deliverName.get(rand)));  
 }  
 myTerminal = new Terminal(1000000);  
 myStorage = new Storage(false);  
 }  
 public static void refreshRating(double result){  
 amountRatings++;  
 sumRatings+=result;  
 rating = sumRatings/amountRatings;  
 }  
 public static void loadClass(){}  
}

**Storage**

package com.example.focus;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Random;  
  
public class Storage {  
 public final HashMap<String, ArrayList<Good>> goodsByDivision = new HashMap<>();  
 private Good fillGoods(int min, int max, String goodName, String divisionName, String extraInfo, boolean loadFactor){  
 int costByDivision = 0;  
 int countByDivision = 0;  
 if(loadFactor) {  
 Random randCost = new Random();  
 costByDivision = randCost.nextInt(max - min + 1) + min;  
 Random randCount = new Random();  
 countByDivision = randCount.nextInt(201) + 200;  
 }  
 return new Good(divisionName, goodName, extraInfo, costByDivision, countByDivision);  
 }  
 public Storage(boolean loadFactor){  
 for (String division : ObjectSwamp.divisionsName){  
 if (division.equals("Bread department")) {  
 ArrayList<Good> goodsBread = new ArrayList<>();  
 goodsBread.add(fillGoods(30, 45,"Bread", division, "(1 loaf)", loadFactor));  
 goodsBread.add(fillGoods(50,60, "Bun", division, "(1 loaf)", loadFactor));  
 goodsBread.add(fillGoods(15, 30,"Patty", division,"(1 thing)", loadFactor));  
 goodsByDivision.put(division, goodsBread);  
 }  
 if (division.equals("Confectionery department")) {  
 ArrayList<Good> goodsConfectionery = new ArrayList<>();  
 goodsConfectionery.add(fillGoods(200, 235, "Jam", division, "(1 kg)",loadFactor));  
 goodsConfectionery.add(fillGoods(210, 260, "Cookies", division, "(1 kg)",loadFactor));  
 goodsConfectionery.add(fillGoods(400, 490, "Pie", division, "(1 kg)",loadFactor));  
 goodsByDivision.put(division, goodsConfectionery);  
 }  
 if (division.equals("Dairy department")) {  
 ArrayList<Good> goodsDairy = new ArrayList<>();  
 goodsDairy.add(fillGoods(60, 80, "Milk", division, "(1 l)",loadFactor));  
 goodsDairy.add(fillGoods(500, 600, "Cheese", division, "(1 kg)",loadFactor));  
 goodsDairy.add(fillGoods(350, 400, "Curd", division, "(1 kg)",loadFactor));  
 goodsByDivision.put(division, goodsDairy);  
 }  
 if (division.equals("Meat section")) {  
 ArrayList<Good> goodsMeat = new ArrayList<>();  
 goodsMeat.add(fillGoods(380, 460, "Stuffing", division, "(1 kg)",loadFactor));  
 goodsMeat.add(fillGoods(330, 380, "Сhicken", division, "(1 kg)",loadFactor));  
 goodsMeat.add(fillGoods(320, 360, "Sausage", division, "(1 kg)",loadFactor));  
 goodsByDivision.put(division, goodsMeat);  
 }  
 if (division.equals("Fish department")) {  
 ArrayList<Good> goodsFish = new ArrayList<>();  
 goodsFish.add(fillGoods(290, 330, "Fish", division, "(1 kg)",loadFactor));  
 goodsFish.add(fillGoods(280, 330, "Sea kale", division, "(1 kg)",loadFactor));  
 goodsFish.add(fillGoods(460, 530, "Shrimps", division, "(1 kg)",loadFactor));  
 goodsByDivision.put(division, goodsFish);  
 }  
 if (division.equals("Vegetable department")) {  
 ArrayList<Good> goodsVegetable = new ArrayList<>();  
 goodsVegetable.add(fillGoods(180, 210, "Cucumbers", division, "(1 kg)",loadFactor));  
 goodsVegetable.add(fillGoods(180, 220, "Tomatoes", division, "(1 kg)",loadFactor));  
 goodsVegetable.add(fillGoods(15, 25, "Onions", division, "(1 kg)",loadFactor));  
 goodsByDivision.put(division, goodsVegetable);  
 }  
 if (division.equals("Fruit department")) {  
 ArrayList<Good> goodsFruit = new ArrayList<>();  
 goodsFruit.add(fillGoods(80, 100, "Apples", division, "(1 kg)",loadFactor));  
 goodsFruit.add(fillGoods(100, 120, "Bananas", division, "(1 kg)",loadFactor));  
 goodsFruit.add(fillGoods(90, 110, "Peaches", division, "(1 kg)",loadFactor));  
 goodsByDivision.put(division, goodsFruit);  
 }  
 if (division.equals("Beverage department")) {  
 ArrayList<Good> goodsBeverage = new ArrayList<>();  
 goodsBeverage.add(fillGoods(100, 130, "Soda", division, "(1 l)",loadFactor));  
 goodsBeverage.add(fillGoods(55, 70, "Water", division, "(1 l)",loadFactor));  
 goodsBeverage.add(fillGoods(100, 130, "Juice", division, "(1 l)",loadFactor));  
 goodsByDivision.put(division, goodsBeverage);  
 }  
 }  
 }  
}

**Terminal**

package com.example.focus;  
  
public class Terminal {  
 private int money;  
 public final Storage storage;  
 public Terminal(int money){  
 this.money = money;  
 this.storage = new Storage(false);  
 }  
 public int getMoney(){  
 return this.money;  
 }  
 public void decreaseMoney(int money){  
 this.money -= money;  
 }  
 public void decreaseMoney(boolean bool, int money){  
 this.money += money;  
 }  
}

**Работа программы**

Общий вид приложения:

Graphical user interface, table

Description automatically generated

Оценки нулевые так как, магазин пуст

**Graphical user interface, application, table

Description automatically generated**

Некоторым людям удалось что-то купить, что и повысило их оценку