УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Программирование

Лабораторная работа №3

Вариант 15432

Преподаватель: Гаврилов Антон Валерьевич

Выполнил: Кульбако Артемий Юрьевич

P3112

Санкт-Петербург

2018

Задание:

Программа должна удовлетворять следующим требованиям:

1. Доработанная модель должна соответствовать принципам SOLID.
2. Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).
3. В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().
4. Программа должна содержать как минимум один перечисляемый тип (enum).

Порядок выполнения работы:

1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями.

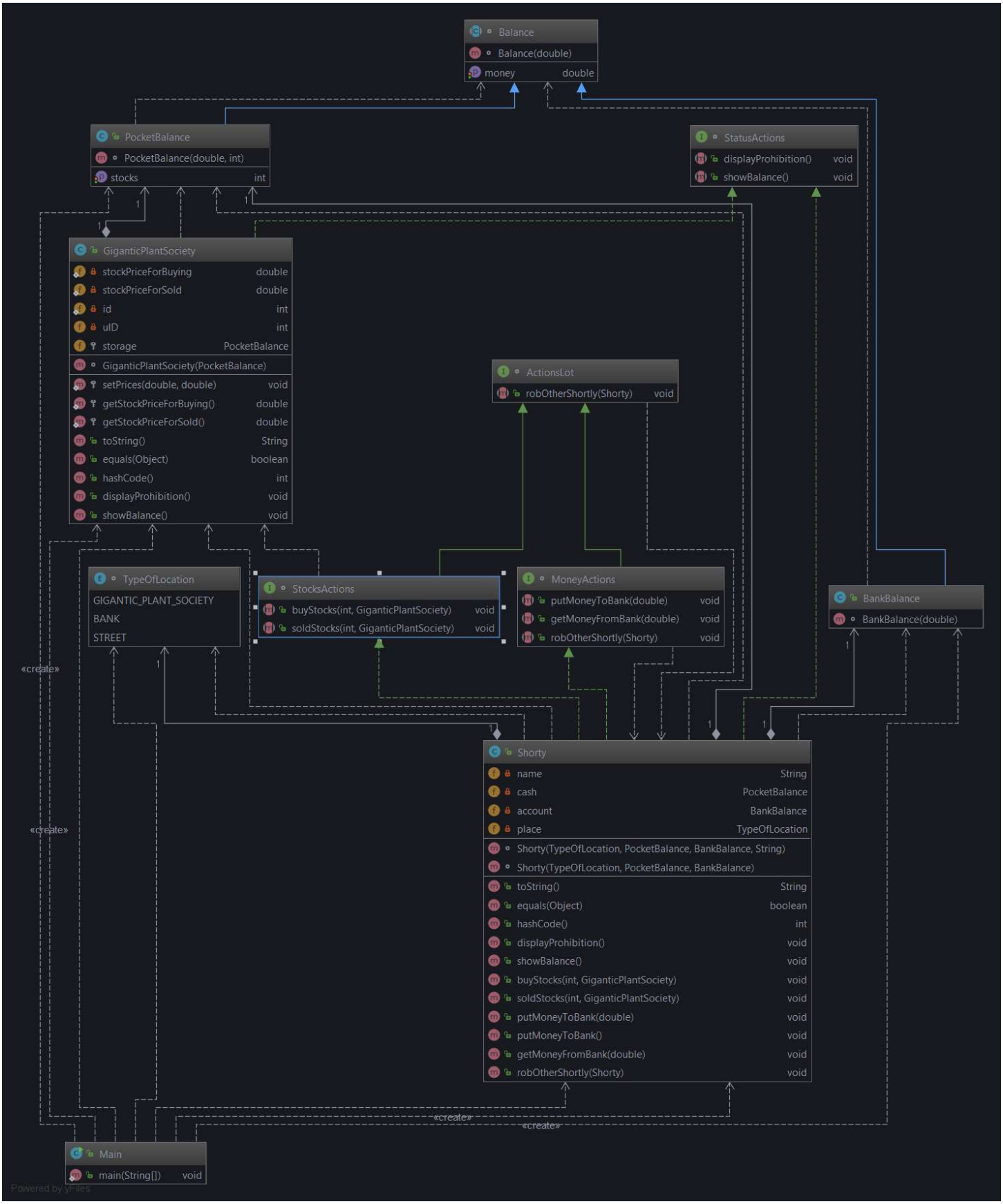
Отчёт по работе должен содержать:

1. Текст задания.
2. Диаграмма классов объектной модели.
3. Исходный код программы.
4. Результат работы программы.
5. Выводы по работе.

Описание предметной области:

Коротышка выложил из кармана денежки и, получив акции, удалился. А желающих приобрести акции с каждым днем становилось все больше. Незнайка и Козлик с утра до вечера продавали акции, Мига же только и делал, что ездил в банк. Там он обменивал вырученные от продажи мелкие деньги на крупные и складывал их в несгораемый шкаф. Многие покупатели являлись в контору слишком рано. От нечего делать они толклись на улице, дожидаясь открытия конторы. Это привлекало внимание прохожих. Постепенно всем в городе стало известно, что акции Общества гигантских растений пользуются большим спросом.

Диаграмма классов:



Исходный код:

Main.java - класс отвечающий за работу программы.

```
//var15432
public class Main {
    public static void main(String[] args){
        GiganticPlantSociety.setPrices(100, 80);
        GiganticPlantSociety office1 = new GiganticPlantSociety(new PocketBalance(0, 500));
        Shorty unnamed1 = new Shorty(TypeOfLocation.GIGANTIC_PLANT_SOCIETY, new
PocketBalance(Math.random() * 7000, 0), new BankBalance(Math.random() * 300));
        unnamed1.showBalance();
        Shorty unnamed2 = new Shorty(TypeOfLocation.GIGANTIC_PLANT_SOCIETY, new
PocketBalance(Math.random() * 7000, 0), new BankBalance(0));
        unnamed2.showBalance();
        Shorty unnamed3 = new Shorty(TypeOfLocation.GIGANTIC_PLANT_SOCIETY, new
PocketBalance(Math.random() * 7000, 0), new BankBalance(Math.random() * 532));
        unnamed3.showBalance();
        unnamed1.buyStocks((int)(Math.random() * 20), office1);
        unnamed2.buyStocks(5, office1);
        unnamed3.buyStocks(43, office1);
        office1.showBalance();
        Shorty Miga = new Shorty(TypeOfLocation.BANK, new
PocketBalance(office1.storage.getMoney(), 0), new BankBalance(0), "Miga");
        office1.storage.setMoney(0);
        Miga.putMoneyToBank();
    }
}
```

Shortly.java – класс «Корышка» - объект, совершающий действия в программе.

```
import java.util.Objects;

public class Shorty implements StocksActions, MoneyActions, StatusActions {
    private String name;
    private PocketBalance cash;
    private BankBalance account;
    private TypeOfLocation place;


    @Override
    public String toString() {
        return "Shorty{" +
                "name='" + name + '\'' +
                ", cash=" + cash +
                ", account=" + account +
                ", place=" + place +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Shorty)) return false;
        Shorty shorty = (Shorty) o;
        return Objects.equals(name, shorty.name) &&
                Objects.equals(cash, shorty.cash) &&
                Objects.equals(account, shorty.account) &&
                place == shorty.place;
    }
```

```
    @Override
    public int hashCode() {
        return Objects.hash(name, cash, account, place);
    }

    Shorty(TypeOfLocation place, PocketBalance cash, BankBalance account, String name) {
        this.name = name;
        this.cash = cash;
        this.account = account;
        this.place = place;
    }

    Shorty(TypeOfLocation place, PocketBalance cash, BankBalance account) {
        this.name = "неизвестный коротышка";
        this.cash = cash;
        this.account = account;
        this.place = place;
    }

/*
        protected typeOfLocation getLocation(){
            return place;
        }
*/

@Override
    public void displayProhibition(){
        System.out.println("> нельзя совершить это действие в данной локации");
    }

    @Override
    public void showBalance() {
        System.out.println("> Баланс коротышки - " + name);
        if (cash != null) System.out.println("    Корманный баланс: Деньги = " +
cash.getMoney() + " | Акции = " + cash.getStocks());
        if (account != null) System.out.println("    Банковский баланс: Деньги = " +
account.getMoney());
    }

    @Override
    public void buyStocks(int n, GiganticPlantSociety officeID) {
        if (place.equals(TypeOfLocation.GIGANTIC_PLANT_SOCIETY)) {
            int x = (int) (cash.getMoney() / GiganticPlantSociety.getStockPriceForBuying());
            if ((n <= x) && (n <= officeID.storage.getStocks())) {
                cash.setStocks(n);
                officeID.storage.setMoney(officeID.storage.getMoney() + n *
GiganticPlantSociety.getStockPriceForBuying());
                officeID.storage.setStocks(officeID.storage.getStocks() - n);
                System.out.println("> " + name + " приобрёл " + n + " акций");
                cash.setMoney(cash.getMoney() - (n *
GiganticPlantSociety.getStockPriceForBuying()));
            } else System.out.println("> " + name + " не имеет достаточно денег для
приобретения такого количества акций, или акции кончились");
        } else displayProhibition();
    }

    @Override
    public void soldStocks(int n, GiganticPlantSociety officeID) {
        if (place.equals(TypeOfLocation.GIGANTIC_PLANT_SOCIETY)) {
            if ((n <= cash.getStocks()) & (n * GiganticPlantSociety.getStockPriceForSold() <=
officeID.storage.getMoney())) {
                cash.setMoney(n * GiganticPlantSociety.getStockPriceForSold());
                officeID.storage.setStocks(officeID.storage.getStocks() + n);
                officeID.storage.setMoney(officeID.storage.getMoney() - n *
```

```
GiganticPlantSociety.getStockPriceForSold());
                System.out.println("> " + name + " продал " + n + " акций");
                cash.setStocks(cash.getStocks() - n);
            }
            else System.out.println("> " + name + " не имеет столько акций, или отделение ОГР
не имеет денег");
        } else displayProhibition();
    }

    @Override
    public void putMoneyToBank(double money){
            if (place.equals(TypeOfLocation.BANK)){
                if (money <= cash.getMoney()){
                    account.setMoney(account.getMoney() + money);
                    cash.setMoney(cash.getMoney() - money);
                    System.out.println("> " + name + " положил " + money + " в банк");
                }
                else System.out.println("> " + name + " не имеет столько наличных или Общество
не может купить его акции");
            }
            else displayProhibition();
    }

    public void putMoneyToBank(){
        if (place.equals(TypeOfLocation.BANK)){
                account.setMoney(account.getMoney() + cash.getMoney());
                System.out.println("> " + name + " положил " + cash.getMoney() + " в банк");
                cash.setMoney(0);
            }
            else System.out.println("> " + name + " не имеет столько наличных или Общество не
может купить его акции");
    }

    @Override
    public void getMoneyFromBank(double money){
            if (place.equals(TypeOfLocation.BANK)){
                if (money <= account.getMoney()){
                    cash.setMoney(cash.getMoney() + money);
                    account.setMoney(account.getMoney() - money);
                }
                else System.out.println("> " + name + " не имеет столько денег на банковском
счету");
            }
            else displayProhibition();
    }
    @Override
    public void robOtherShortly(Shorty man) {
        if (place.equals(TypeOfLocation.STREET)) {
            cash.setMoney(cash.getMoney() + man.cash.getMoney());
            man.cash.setMoney(0);
            System.out.println(name + " ограбил " + name);
        }
        else displayProhibition();
    }
}
```

Actions.java – действия, которые может совершать коротышка.

```
interface ActionsLot {
    void robOtherShortly(Shorty man);
}
```

```java
interface StocksActions extends ActionsLot {
    void buyStocks(int n, GiganticPlantSociety officeID);
    void soldStocks(int n, GiganticPlantSociety officeID);
}

 interface MoneyActions extends ActionsLot {
    void putMoneyToBank(double money);
    void getMoneyFromBank(double money);
    void robOtherShortly(Shorty man);
}

interface StatusActions {
    void displayProhibition();
    void showBalance();
}
```

Balance.java – деньги коротышки или ОГР. Могут быть физическими или на банковском аккаунте.

```java
abstract class Balance {
    private double sum;
    Balance(double sum){
        this.sum = sum;
    }
    public void setMoney(double sum){
        this.sum = sum;
    }
    public double getMoney() {
        return sum;
    }
}
```

PocketBalance.java – деньги и акции находящиеся у коротышки физически.

```java
public class PocketBalance extends Balance {
    private int amount;
    PocketBalance(double sum, int amount){
        super(sum);
        this.amount = amount;
    }
    public void setStocks(int amount){
        this.amount = amount;
    }
    public int getStocks() {
        return amount;
    }
}
```

BankBalance.java – банковский счёт коротышки.

```java
public class BankBalance extends Balance {
    BankBalance(double sum){
        super(sum);
    }
}
```

Locations.java – типы локаций, на которых могут находиться коротышки. Влиет на доступность действий.

```
enum TypeOfLocation {
    GIGANTIC_PLANT_SOCIETY,
    BANK,
    STREET
}
```

GiganticPlantSociety.java – Общество Гигантских Растений. Устанавливает цену на акции.

```
import java.util.Objects;

public class GiganticPlantSociety implements StatusActions {
    private static double stockPriceForBuying;
    private static double stockPriceForSold;
    private static int id = 0;
    private int uID;
    protected PocketBalance storage;

    /*    GiganticPlantSociety(){
            System.out.println("> Общество Гигантский Растений сформировано");
        }
    */

    protected static void setPrices(double _stockPriceForBying, double _stockPriceForSold){
        stockPriceForBuying = _stockPriceForBying;
        stockPriceForSold = _stockPriceForSold;
        System.out.println("> Общество Гигантский Растений установило цены на акции:");
        System.out.println("    Цена для покупки = " + stockPriceForBuying + " | Цена для
продажи = " + stockPriceForSold);
    }

    protected static double getStockPriceForBuying(){
        return stockPriceForBuying;
    }

    protected static double getStockPriceForSold(){
        return stockPriceForSold;
    }

    GiganticPlantSociety(PocketBalance storage){
        id++;
        uID = id;
        this.storage = storage;
        System.out.println("> Офис ОГР №" + uID + " открыт");
    }

    @Override
    public String toString() {
        return "GiganticPlantSociety{" +
                "uID=" + uID +
                ", storage=" + storage +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof GiganticPlantSociety)) return false;
        GiganticPlantSociety that = (GiganticPlantSociety) o;
        return uID == that.uID &&
                Objects.equals(storage, that.storage);
    }
```

```java
    @Override
    public int hashCode() {
        return Objects.hash(uID, storage);
    }

    @Override
    public void displayProhibition(){
        System.out.println("> В офисе ОГР можно приобретать и продавать акции");
    }

    @Override
    public void showBalance(){
        System.out.println("> Баланс отделения №" + uID + " = " + storage.getMoney());
    }
}
```

Результат работы:

```
> Общество Гигантский Растений установило цены на акции:

    Цена для покупки = 100.0 | Цена для продажи = 80.0

> Офис ОГР №1 открыт

> Баланс коротышки - неизвестный коротышка

    Корманный баланс: Деньги = 845.5002454872886 | Акции = 0

    Банковский баланс: Деньги = 3.2840562476289525

> Баланс коротышки - неизвестный коротышка

    Корманный баланс: Деньги = 3776.231286074658 | Акции = 0

    Банковский баланс: Деньги = 0.0

> Баланс коротышки - неизвестный коротышка

    Корманный баланс: Деньги = 501.73164789488254 | Акции = 0

    Банковский баланс: Деньги = 97.15380019393429

> неизвестный коротышка не имеет достаточно денег для приобретения такого количества акций, или акции кончились

> неизвестный коротышка приобрёл 5 акций

> неизвестный коротышка не имеет достаточно денег для приобретения такого количества акций, или акции кончились

> Баланс отделения №1 = 500.0

> Miga положил 500.0 в банк
```

Вывод:

В процессе выполнения лабораторной работы были получены навыки использования объектно-ориентированного подхода программирования при использовании языка Java.