УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Программирование

# Лабораторная работа №4

Вариант 15432.1669

Преподаватель: Гаврилов Антон Валерьевич

Выполнил: Кульбако Артемий Юрьевич

P3112

Санкт-Петербург

2018

# Задание:

**Программа должна удовлетворять следующим требованиям:**

1. В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.
2. В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).

**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.

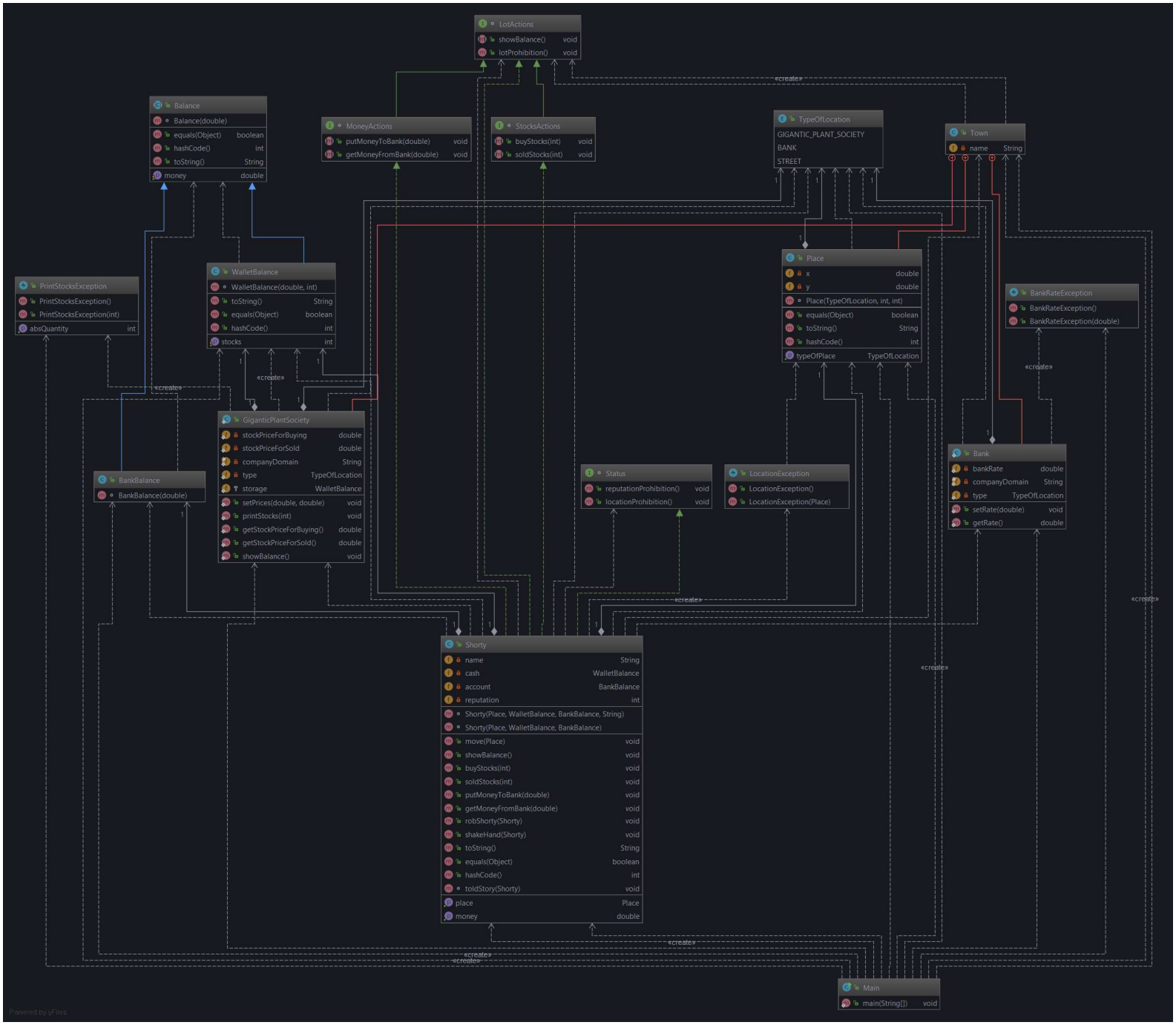
**Вопросы к защите лабораторной работы:**

1. Обработка исключительных ситуаций, три типа исключений.
2. Вложенные, локальные и анонимные классы.
3. Механизм рефлексии (reflection) в Java. Класс `Class`.

# Описание предметной области:

Пожав каждому из покупателей руку, Мига выпроводил их всех из конторы и бросился обнимать Незнайку и Козлика. Дело действительно быстро пошло на лад. Правда, в этот день покупатели больше не появлялись, зато когда Мига и Жулио пришли в контору на следующий день, они обнаружили, что торговля акциями идет довольно бойко. Перед Незнайкой и Козликом то и дело появлялись разные коротышки и выкладывали на стол свои денежки. Здесь были уже не только деревенские жители, но даже и городские. Один из них рассказал нашим друзьям, что когда-то давно он ушел из деревни, где у него остался небольшой клочок земли. Он мечтал поступить куда-нибудь на завод или на фабрику и подзаработать денег, чтоб прикупить земли, так как его клочок давал очень небольшой урожай. В конце концов ему удалось устроиться рабочим на фабрику, однако за долгие годы работы он так и не смог скопить сумму, которой хватило бы на покупку земли. 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. Городские жители сообразили, что с течением времени цена на акции может повыситься. Все вспоминали об удивительном случае, когда акции одного нефтяного общества, купленные по одному фертингу штука, впоследствии продавались сначала по два, потом по три, потом по пять фертингов, а в тот день, когда стало известно, что из-под земли, где велись изыскательные работы, забил наконец нефтяной фонтан, цена на акции подскочила до десяти фертингов штука. Каждый, кто продал свои акции в этот день, получил в десять раз больше денег, чем истратил вначале. Наслушавшись подобных рассказов, каждый, кому удалось сберечь на черный день сотню-другую фертингов, спешил накупить гигантских акций, с тем чтоб продать их, как только они повысятся в цене. В результате два миллиона акций, хранившиеся в двух несгораемых сундуках, были быстро распроданы.

# Диаграмма классов:



# Исходный код:

## Main.java – класс, в котором создаётся и заполняется мир.

```
//var15432.1669

public class Main {
    public static void main(String[] args){
        Town.GiganticPlantSociety.setPrices(3, 2);
        try {
            Town.GiganticPlantSociety.printStocks(430);
        }
        catch (PrintStocksException ex){
            ex.printStackTrace();
            Town.GiganticPlantSociety.printStocks(ex.getAbsQuantity());
        }
        Town planet = new Town();
        Town.Place office = planet.new Place(TypeOfLocation.GIGANTIC_PLANT_SOCIETY, 1, 0);
        Town.Place cityBank = planet.new Place(TypeOfLocation.BANK, 5, 9);
        Town.Place darkStreet = planet.new Place(TypeOfLocation.STREET, 1, 1);
        Shorty Miga = new Shorty(office, new WalletBalance(500, 0), new BankBalance(0), "Miga");
        Shorty Lupa = new Shorty(office, new WalletBalance(300, 0), new BankBalance(0), "Pupa");
        Shorty Pupa = new Shorty(office, new WalletBalance(250, 0), new BankBalance(0), "Lupa");
        Shorty Kojima = new Shorty(office, new WalletBalance(12000, 0), new BankBalance(0), "Хидео");
        Shorty Kulbako = new Shorty(darkStreet, new WalletBalance(23, 0), new BankBalance(0), "Пугалол");
        try {
            Town.Bank.setRate(8);
        }
        catch (BankRateException ex1){
            ex1.printStackTrace();
        }
        Lupa.buyStocks(5);
        Pupa.buyStocks(2);
        Kojima.buyStocks(270);
        Miga.shakeHand(Kojima);
        Pupa.toldStory(Miga);
        Kojima.move(darkStreet);
        Kulbako.robShorty(Kojima);
        Miga.showBalance();
        Miga.move(cityBank);
        Miga.putMoneyToBank(Miga.getMoney());
        Miga.showBalance();
        }
    }
```

## Actions.java – класс, содержащий интерфейсы для действий.

```
interface LotActions {

    void showBalance();

    default void lotProhibition() {System.out.println("> не имеет столь фертингов или акций");}

}


interface StocksActions extends LotActions {

    void buyStocks(int n) throws LocationException;

    void soldStocks(int n) throws LocationException;

}
```

```
 interface MoneyActions extends LotActions {
    void putMoneyToBank(double money) throws LocationException;
    void getMoneyFromBank(double money) throws LocationException;
}
```

`Balance.java` – абстрактный класс, содержащий фертинги коротышки.

```
import java.util.Objects;


abstract public class Balance {
    private double sum;
    Balance(double sum){
        this.sum = sum;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Balance)) return false;
        Balance balance = (Balance) o;
        return Double.compare(balance.sum, sum) == 0;
    }

    @Override
    public int hashCode() {
        return Objects.hash(sum);
    }

    @Override
    public String toString() {
        return "Balance{" +
                "sum=" + sum +
                '}';
    }

    public void setMoney(double sum) {
        this.sum = sum;
    }
```

```
    public double getMoney() {
        return sum;
    }
}
```

### BankBalance.java – класс, содержащий деньги коротышки, которые лежат в банке.

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

### LocationException.java – ошибки, связанные с местом коротышки.

```
public class LocationException extends RuntimeException{

    public LocationException(){
    }

    public LocationException(Town.Place space){
        super("Недопустимая локация, " + space);
    }
}
```

### PrintStockException.java – ошибки, связанные с местом коротышки.

```
import java.lang.Math.*;

public class PrintStocksException extends RuntimeException{
    private int quantity;

    public int getAbsQuantity() {
        return quantity;
    }

    public PrintStocksException(){
    }

    public PrintStocksException(int quantity){
        super("Нельзя напечатать отрицательное число акций, " + quantity);
        this.quantity = Math.abs(quantity);
    }
}
```

### BankRateException.java – ошибки, связанные с установленной процентной ставкой.

```
public class BankRateException extends Exception{

    public BankRateException(){
    }

    public BankRateException(double n){
        super("Процентная ставка не может быть отрицательной или нулевой, " + n);
    }
}
```

`Shorty.java` – класс, описывающий объект коротышка, который может совершать действия.

```
import java.util.Objects;

public class Shorty implements StocksActions, MoneyActions, Status, LotActions {
    private String name;
    private WalletBalance cash;
    private BankBalance account;
    private Town.Place space;
    private int reputation = 0;


    Shorty(Town.Place space, WalletBalance cash, BankBalance account, String name) {
        this.name = name;
        this.cash = cash;
        this.account = account;
        this.space = space;
    }

    Shorty(Town.Place space, WalletBalance cash, BankBalance account) {
        this.name = "неизвестный коротышка";
        this.cash = cash;
        this.account = account;
        this.space = space;
    }

    public Town.Place getPlace() {
        return space;
    }

    public double getMoney(){
        return cash.getMoney();
    }

    public void move(Town.Place space) {
        this.space = space;
        System.out.println("> " + name + " перешёл в локацию " + space.toString());
    }

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

    @Override
    public void buyStocks(int n) {
        if (reputation >= 0) {
            if (!space.getTypeOfPlace().equals(TypeOfLocation.GIGANTIC_PLANT_SOCIETY)) throw new
LocationException(space);
                int x = (int) (cash.getMoney() / Town.GiganticPlantSociety.getStockPriceForBuying());
                if ((n <= x) && (n <= Town.GiganticPlantSociety.storage.getStocks())) {
                    cash.setStocks(n);
                    Town.GiganticPlantSociety.storage.setMoney(Town.GiganticPlantSociety.storage.getMoney() + n *
Town.GiganticPlantSociety.getStockPriceForBuying());
                    Town.GiganticPlantSociety.storage.setStocks(Town.GiganticPlantSociety.storage.getStocks() - n);
                    System.out.println("> " + name + " приобрёл " + n + " акций");
                    cash.setMoney(cash.getMoney() - (n * Town.GiganticPlantSociety.getStockPriceForBuying()));
                } else lotProhibition();
        } else reputationProhibition();
    }
```

```
    @Override
    public void soldStocks(int n) {
        if (reputation >= 0) {
            if (!space.getTypeOfPlace().equals(TypeOfLocation.GIGANTIC_PLANT_SOCIETY)) throw new
LocationException(space);
                if ((n <= cash.getStocks()) & (n * Town.GiganticPlantSociety.getStockPriceForSold() <=
Town.GiganticPlantSociety.storage.getMoney())) {
                    int x = 0 - n;
                    cash.setMoney(x * Town.GiganticPlantSociety.getStockPriceForSold());
                    Town.GiganticPlantSociety.storage.setStocks(Town.GiganticPlantSociety.storage.getStocks() + n);
                    Town.GiganticPlantSociety.storage.setMoney(Town.GiganticPlantSociety.storage.getMoney() - x *
Town.GiganticPlantSociety.getStockPriceForSold());
                    System.out.println("> " + name + " продал " + n + " акций");
                    cash.setStocks(cash.getStocks() - x);
                } else lotProhibition();
        } else reputationProhibition();
    }

    @Override
    public void putMoneyToBank(double money) {
        if (reputation >= 0) {
            if (!space.getTypeOfPlace().equals(TypeOfLocation.BANK)) throw new LocationException(space);
                if (money <= cash.getMoney()) {
                    account.setMoney(money * Town.Bank.getRate());
                    cash.setMoney(cash.getMoney() - money);
                    System.out.println("> " + name + " положил " + money + " в банк");
                } else lotProhibition();
        } else reputationProhibition();
    }

    @Override
    public void getMoneyFromBank(double money) {
        if (reputation >= 0) {
            if (!space.getTypeOfPlace().equals(TypeOfLocation.BANK)) throw new LocationException(space);
                if (money <= account.getMoney()) {
                    cash.setMoney(cash.getMoney() + money);
                    account.setMoney(account.getMoney() - money);
                } else lotProhibition();
        } else reputationProhibition();
    }

    public void robShorty(Shorty man) {
        int chance = (int) (1 + Math.random() * 10);
        if (!(space.equals(man.space))) throw new LocationException(space);
        if (chance > 3) {
            cash.setMoney(cash.getMoney() + man.cash.getMoney());
            man.cash.setMoney(0);
            System.out.println("> " + name + " ограбил коротышку - " + man.name);
            reputation -= 50;
        } else System.out.println("> Ограбление не удалось");
    }

    public void shakeHand(Shorty man) throws LocationException{
        if (!space.equals(man.space)) throw new LocationException(space);
            reputation++;
            man.reputation++;
            System.out.println("> " + name + " пожал руку " + man.name + ".");
            System.out.println("    репутация коротышки " + name + " = " + reputation);
            System.out.println("    репутация коротышки " + man.name + " = " + man.reputation);
    }
```

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

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

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

    void toldStory(Shorty man) {
            if (!space.equals(man.space)) throw new LocationException(space);
            int a = (int) (1 + Math.random() * 6);
            switch (a) {
                case 1:
                case 2:
                    System.out.println("> " + name + " делится своей биографией с " + man.name);
                    System.out.println("    \"Я мечтал поступить куда-нибудь на завод или на фабрику и подзаработать денег, чтоб прикупить земли, так как мой клочок давал очень небольшой урожай. В конце концов мне удалось устроиться рабочим на фабрику, однако за долгие годы работы я так и не смог скопить сумму, которой хватило бы на покупку земли.\"");
                    break;
                case 3:
                case 4:
                    System.out.println("> " + name + " делится своей биографией с " + man.name);
                    System.out.println("    \"Друг рассказал мне о вашем обществе. Решил прикупил немного акций, вдруг не прогарю.\"");
                    break;
                case 5:
                case 6:
                    System.out.println("> " + name + " делится своей биографией с " + man.name);
                    System.out.println("    \"Я помню как акции одного нефтяного общества, вот название уже позабыл, выросли в десять раз. Тогда я акции не покупал, а сейчас куплю. Контора у вас от народа.\"");
                    break;
            }
            man.reputation += 10;
    }
    }
```

`Status.java` – интерфейс, реализующий репутацию коротышки.

```
interface Status {
    default void reputationProhibition() {System.out.println("> Репутация коротышки отрицательна"); }
}
```

`TypeOfLocation.java` – перечислимый тип, содержащий виды локаций.

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

`WalletBalance.java` – класс, содержащий фертинги и акции коротышки.

```
import java.util.Objects;

public class WalletBalance extends Balance{
    private int amount;
    WalletBalance(double sum, int amount){
        super(sum);
        this.amount = amount;
    }

    @Override
    public String toString() {
        return "WalletBalance{" +
                "amount=" + amount +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof WalletBalance)) return false;
        if (!super.equals(o)) return false;
        WalletBalance that = (WalletBalance) o;
        return amount == that.amount;
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), amount);
    }

    public void setStocks(int amount){
        this.amount = amount;
    }
    public int getStocks() {
        return amount;
    }
}
```

## Town.java – класс, содержащий описание локаций и их объекты.

```
import java.util.Objects;

public class Town {

    private String name = "Цветочный город";

    {
        LotActions townStat = new LotActions() {
            @Override
            public void showBalance() {

                System.out.println("> У города - " + name + " нет казны. Он живёт на энтузиазме жителей.");
            }
        };
        townStat.showBalance();
    }

    public static class GiganticPlantSociety {
        private static double stockPriceForBuying;
        private static double stockPriceForSold;
        private static final String companyDomain = " ОАО \"ОГР\"";
        private static TypeOfLocation type = TypeOfLocation.GIGANTIC_PLANT_SOCIETY;
        protected static WalletBalance storage;

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

        public static void printStocks(int amount) {
            if (amount < 0) throw new PrintStocksException(amount);
            storage = new WalletBalance(0, amount);
            System.out.println("> " + companyDomain + " напечатало " + amount + " акций");
        }

        public static double getStockPriceForBuying() {
            return stockPriceForBuying;
        }

        public static double getStockPriceForSold() {
            return stockPriceForSold;
        }

        public static void showBalance() {
            System.out.println("> Для продажи доступно " + storage.getStocks() + " акций");
        }
    }

    public static class Bank {
```

```
    private static double bankRate;
    private static final String companyDomain = "Городской банк";
    private static TypeOfLocation type = TypeOfLocation.GIGANTIC_PLANT_SOCIETY;

    public static void setRate(double rate) throws BankRateException{
        if (rate <= 0) throw new BankRateException(rate);
        bankRate = 1 + 0.01 * rate;
        System.out.println(" > " + companyDomain + " установил процентную ставку по вкладам = " + bankRate);
    }

    public static double getRate() {
        return bankRate;
    }
}

public class Place {

    private TypeOfLocation type;
    private double x;
    private double y;

    Place(TypeOfLocation type, int x, int y){
        this.type = type;
        this.x = x;
        this.y = y;
    }

    public TypeOfLocation getTypeOfPlace() {
        return type;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Place)) return false;
        Place place = (Place) o;
        return Double.compare(place.x, x) == 0 &&
                Double.compare(place.y, y) == 0;
    }

    @Override
    public String toString() {
        return "Place{" +
                "type=" + type +
                ", x=" + x +
                ", y=" + y +
                '}';
    }

    @Override
    public int hashCode() {
        return Objects.hash(x, y);
    }
```

```
  }
}
```

## Результат работы:

```
>  ОАО "ОГР" установило цены на акции:

    Цена для покупки = 3.0 | Цена для продажи = 2.0

>  ОАО "ОГР" напечатало 430 акций

> У города - Цветочный город нет казны. Он живёт на энтузиазме жителей.

 > Городской банк установил процентную ставку по вкладам = 1.08

> Pupa приобрёл 5 акций

> Lupa приобрёл 2 акций

> Хидео приобрёл 270 акций

> Miga пожал руку Хидео.

    репутация коротышки Miga = 1

    репутация коротышки Хидео = 1

> Lupa делится своей биографией с Miga

    "Я мечтал поступить куда-нибудь на завод или на фабрику и подзаработать денег, чтоб
прикупить земли, так как мой клочок давал очень небольшой урожай. В конце концов мне удалось
устроиться рабочим на фабрику, однако за долгие годы работы я так и не смог скопить сумму,
которой хватило бы на покупку земли."

> Хидео перешёл в локацию Place{type=STREET, x=1.0, y=1.0}

> Пугалол ограбил коротышку - Хидео

> Баланс коротышки - Miga

    Корманный баланс: Фертинги = 500.0 | Акции = 0

    Банковский баланс: Фертинги = 0.0

> Miga перешёл в локацию Place{type=BANK, x=5.0, y=9.0}

> Miga положил 500.0 в банк

> Баланс коротышки - Miga

    Корманный баланс: Фертинги = 0.0 | Акции = 0

    Банковский баланс: Фертинги = 540.0


Process finished with exit code 0
```

## Вывод:

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