Университет ИТМО

Факультет ПИиКТ

Программирование

Лабораторная работа №4

Вариант №1920

Студент:

Курносова Ирина Викторовна
Группа P3231

Преподаватель:

Письмак Алексей Евгеньевич

г. Санкт-Петербург
2021 год

Задание

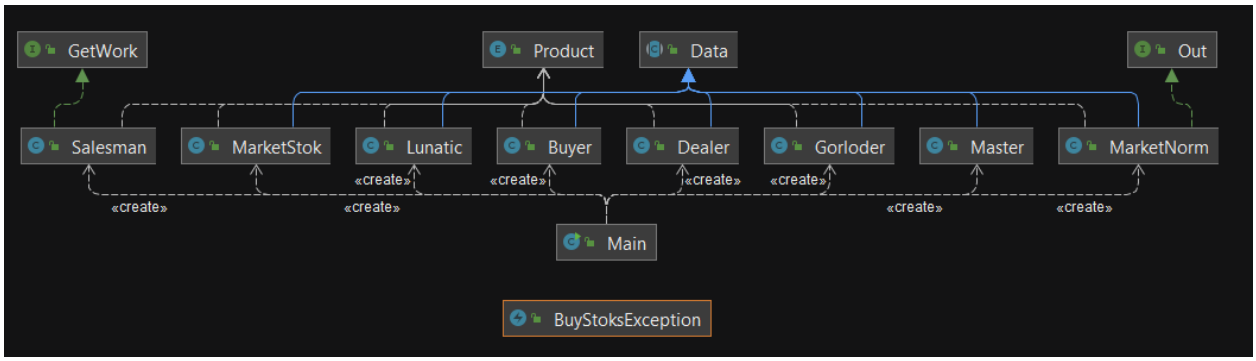
Описание предметной области, по которой должна быть построена объектная модель:

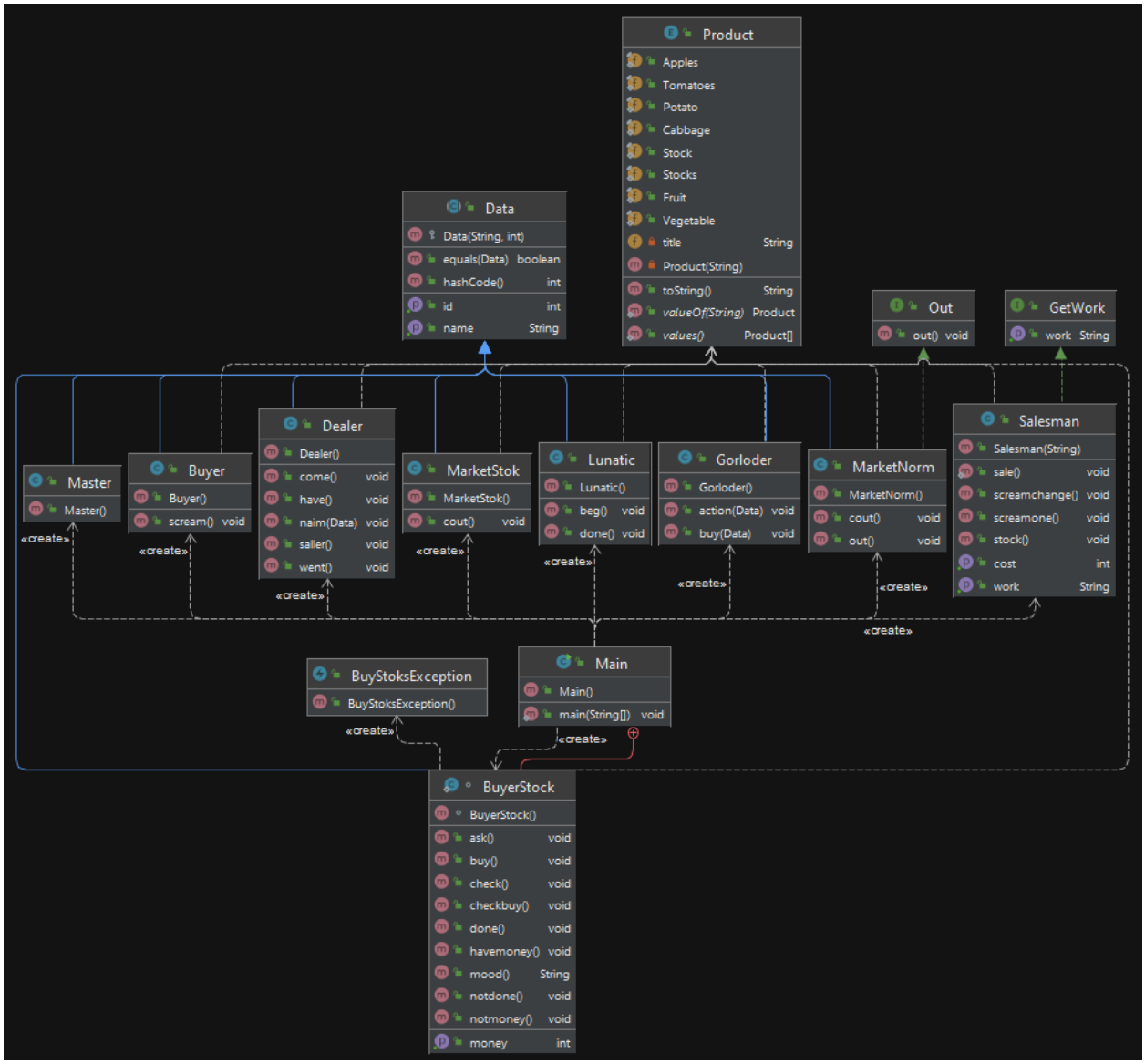
Нужно сказать, что рынок, на котором торгуют акциями, очень отличается от обычного рынка, где торгуют яблоками, помидорами, картофелем или капустой. Дело в том, что продавцу фруктов или овощей достаточно разложить свой товар на прилавке, чтобы все видели, чем он торгует. Продавец акций носит свой товар в кармане, и единственное, что может делать, это выкрикивать название своих акций и цену, по которой он желает их продавать. Покупателю тоже остается только выкрикивать название тех акций, которые он хочет купить. С тех пор как появились акционерские рынки, некоторые лунатики стали покупать акции не только для того, чтоб иметь долю в барышах какого-нибудь предприятия, но и для того, чтоб продавать их по более высокой цене. Появились торговцы, которые покупали и продавали акции в огромных количествах и получали на этом большие прибыли. Такие торговцы уже не ходили сами на рынок, а нанимали для этого специальных крикунов или так называемых горлодериков. Многие горлодерики работали не на одного, а сразу на нескольких хозяев. Для одного хозяина такой горлодерик покупал одни акции, для другого - другие, для третьего не покупал, а, наоборот, продавал.

Программа должна удовлетворять следующим требованиям:

1. Объектная модель должна соответствовать принципам SOLID.
2. Модель должна содержать как минимум два интерфейса или абстрактных класса
3. Модель должна содержать не менее четырех конкретных классов, связанных отношениями наследования и композиции.
4. Сценарий должен предусматривать как минимум три варианта выполнения программы в зависимости от начальных условий, в сценарии должно быть предусмотрено повторяющееся действие, реализуемое с помощью цикла.
5. В разработанных классах при необходимости должны быть переопределены методы equals(), toString() и hashCode().
6. Один из объектов необходимо реализовать с помощью анонимного класса.
7. Программа должна содержать как минимум один перечисляемый тип (enum).
8. В программе необходимо реализовать как минимум один класс собственного исключения.
9. Исключения, возникающие во время работы программы должны быть обработаны.

Диаграмма классов:





Код программы:

Buyer.java

```
public class Buyer extends Data{
    public Buyer() {
        super("Покупатель", 48);
    }
    public void scream() {
        System.out.println(getName() + " также кричит название " +
Product.Stock + ", которую хочет купить");
    }
    public String mood() {
        return("Нам не особо важно его настроение");
    }
}
```

Dealer.java

```
public class Dealer extends Data {
    public Dealer() {
        super("Торговцы", 50);
    }
    public void come() {
        System.out.println("Появились " + getName());
    }
    public  void saller() {
        System.out.println(getName() + " покупали и продавали " +
Product.Stock + " в больших количествах");
    }
    public void have() {
        System.out.println(getName() + " получали на этом большие деньги");
    }
    public void went() {
        System.out.println(getName() + " уже не ходили на рынок");
    }
    public void naim(Data gorl) {
        System.out.println(getName() + " нанимали специальных " +
gorl.getName() + "ов");
    }

    public String mood() {
        return("Нам не особо важно его настроение");
    }
}
```

Gorloder.java

```
public class Gorloder extends Data {
    public Gorloder() {
        super("Горлодёрик", 10);
    }
    public void action(Data mas) {
        System.out.println(getName() + "и работали на нескольких " +
mas.getName());
    }
    public void buy(Data mas) {
        System.out.println("Для одного " + mas.getName() + "а они покупали
одни " + Product.Stock);
        System.out.println("Для другого - другие");
```

```
        System.out.println("А для третьего " + mas.getName() + "а " +
getName() + "и вообще не покупали " + Product.Stock + ", а наоборот -
продавали");
    }
    public String mood() {
        return("Нам не особо важно его настроение");
    }
}
```

Lunatic.java

```
public class Lunatic extends Data{
    public Lunatic() {
        super("Лунатики", 20);
    }
    public void beg()
    {
        System.out.println("Появились " + getName());
    }
    public void done() {
        System.out.println(getName() + " покупают " + Product.Stock + " не
только для себя");
        System.out.println(" но и для перепродажи по более высокой цене");
    }
    public String mood() {
        return("Нам не особо важно его настроение");
    }
}
```

MarketNorm.java

```
public class MarketNorm extends Data implements Out{
    public MarketNorm() {
        super("Обычный рынок", 21);
    }
    public void cout() {
        System.out.println(getName()+", там продают: " + Product.Apples + ",
" + Product.Tomatoes + ", " + Product.Potato + ", " + Product.Cabbage);
    }

    @Override
    public void out() {
        Out.super.out();
    }
    public String mood() {
        return("Нам не особо важно его настроение");
    }
}
```

MarketStok.java

```
public class MarketStok extends Data {
    public MarketStok() {
        super("Акционный рынок", 22);
    }
    public void cout() {System.out.println(getName() + ", там продают: " +
Product.Stock);}
    public String mood() {
        return("Нам не особо важно его настроение");
```

```
    }
}
```

Master.java

```
public class Master extends Data {
    public Master() {
        super("Хозяин", 18);
    }
    public String mood() {
        return("Нам не особо важно его настроение");
    }
}
```

Salesman.java

```
public class Salesman implements GetWork{
    private String work = "Продавец";
    public Salesman(String specific) {
        if (specific == "F") {
            this.work = work + ' ' + Product.Fruit;
        } else if (specific == "V") {
            this.work = work + ' ' + Product.Vegetable;
        } else if (specific == "S") {
            this.work = work + ' ' + Product.Stocks;
        }
    }
    @Override
    public String getWork() {
        return work;
    }
    public static void sale() {
        System.out.println(" раскладывают свой товар на прилавке");
    }
    public void stock() {
        System.out.println(getWork() + " носит свои акции в кармане");
        System.out.println(getWork() + " может лишь выкрикивать их название и
цену");
    }
    private int cost = (int)Math.round(Math.random()*1000);
    public int getCost() {return cost;}
    public void screamone() {
        System.out.println("Один " + getWork() + " кричит стоимость: " +
getCost() + " рублей");
    }
    public void screamchange() {
        cost = (int)Math.round(Math.random()*1000);
        System.out.println("Следующий " + getWork() + " предлагает цену: " +
getCost() + " рублей");
    }
}
```

Data.java

```
abstract public class Data {
    private final String Name;
    private final int Id;
    protected Data(String name, int id) {
        this.Name = name;
```

```
        this.Id = id;
    }
    public String getName() {return Name;}
    public int getId() {return Id;}
    public abstract String mood();
    public boolean equals(Data num){
        if (this.Id == num.getId()) {
            return(true);
        }
        else
            return(false);
    }
    public int hashCode() {
        int result = Name.hashCode();
        result = 31 * result + Id;
        return result; }
}
```

BuyStoksException.java

```
public class BuyStoksException extends Exception{
}
```

GetWork.java

```
public interface GetWork {
    String getWork();
}
```

Out.java

```
public interface Out {
    default void out() {
        System.out.println("Существуют рынки двух типов:");
    }
}
```

Product.java

```
public enum Product {
    Apples("яблоки"),
    Tomatoes("помидоры"),
    Potato("картофель"),
    Cabbage("капуста"),
    Stock("акции"),
    Stocks("акций"),
    Fruit("фруктов"),
    Vegetable("овощей");
    private String title;
    Product(String title) {
        this.title=title;
    }
    @Override
    public String toString() {
        return this.title;
    }
}
```

Main.java

```
public class Main{

    static class BuyerStock extends Data{

        BuyerStock() {
            super("Покупатель акций", (int)Math.round(Math.random()));
        }


        public void buy() {
            System.out.println("Один " + getName() + " собрался купить " +
Product.Stock);
            System.out.println("Он посчитал, сколько у него денег");
        }

        public String mood() {
            if (Math.round(Math.random())==0) {
                return("грустный");
            }
            else {
                return("весёлый");
            }
        }

        public int money = 0;
        public void notmoney() {
            money = 0;
            System.out.println("Но оказалось, что у него нет денег");
            System.out.println("Поэтому он " + mood() + " вернулся домой");
        }
        public void havemoney() {
            money = (int)Math.round(Math.random()*1000);
            System.out.println("У него оказалось " + money + " рублей");
        }
        public int getMoney() {return money;}

        public void checkbuy() throws BuyStoksException{
            if (getId()==0) {
                throw new BuyStoksException();
            }
        }
        public void check() {
            try {
                checkbuy();
                havemoney();
            } catch (Exception e) {
                notmoney();
            }
        }

        public void ask() {
            System.out.println(getName() +  " пошёл узнавать цены на " +
Product.Stock);
        }
        public void done() {
            System.out.println("Отлично! " + getName() + " купил нужные акции
и " + mood() + " пошёл домой.");
        }
        public void notdone() {
            System.out.println(getName() + " подумал, что это слишком дорого
и пошёл искать другие предложения");
        }
```

```
    }

    public static void main(String[] args) {
        MarketNorm marketn = new MarketNorm();
        Out markets = new Out() {
            public void out() {System.out.println("Акционный рынок, там
продают: " + Product.Stock);}
        };
        //MarketStok markets = new MarketStok();
        Buyer buyer = new Buyer();
        Salesman salesmanF = new Salesman("F");
        Salesman salesmanV = new Salesman("V");
        Salesman salesmanS = new Salesman("S");
        Lunatic lunatic = new Lunatic();
        Dealer dealer = new Dealer();
        Gorloder gorloder = new Gorloder();
        Master master = new Master();
        BuyerStock buyerStock = new BuyerStock();

        marketn.out();
        marketn.cout();
        markets.out();
        System.out.println(salesmanF.getWork() + " и " +
salesmanV.getWork());
        Salesman.sale();
        salesmanS.stock();
        buyer.scream();
        System.out.println(buyer.mood());
        lunatic.beg();
        lunatic.done();
        dealer.come();
        dealer.saller();
        dealer.have();
        dealer.went();
        dealer.naim(gorloder);
        gorloder.action(master);
        gorloder.buy(master);
        buyerStock.buy();
        buyerStock.check();

        if (buyerStock.getMoney()!=0) {
            buyerStock.ask();
            salesmanS.screamone();
            while (buyerStock.getMoney() < salesmanS.getCost()) {
                buyerStock.notdone();
                salesmanS.screamchange();
            }
            buyerStock.done();
        }
    }
}
```

Результат работы программы:

Существуют рынки двух типов:
Обычный рынок, там продают: яблоки, помидоры, картофель, капуста
Акционный рынок, там продают: акции
Продавец фруктов и Продавец овощей
раскладывают свой товар на прилавке

Продавец акций носит свои акции в кармане
Продавец акций может лишь выкрикивать их название и цену
Покупатель также кричит название акции, которую хочет купить
Нам не особо важно его настроение
Появились Лунатики
Лунатики покупают акции не только для себя
но и для перепродажи по более высокой цене
Появились Торговцы
Торговцы покупали и продавали акции в больших количествах
Торговцы получали на этом большие деньги
Торговцы уже не ходили на рынок
Торговцы нанимали специальных Горлодёриков
Горлодёрики работали на нескольких Хозяин
Для одного Хозяина они покупали одни акции
Для другого - другие
А для третьего Хозяина Горлодёрики вообще не покупали акции, а наоборот - продавали
Один Покупатель акций собрался купить акции
Он посчитал, сколько у него денег
У него оказалось 817 рублей
Покупатель акций пошёл узнавать цены на акции
Один Продавец акций кричит стоимость: 445 рублей
Отлично! Покупатель акций купил нужные акции и грустный пошёл домой.

Вывод:
При выполнении данной работы мною были освоены навыки работы с обработкой исключительных ситуаций; локальными и абстрактными классами и интерфейсами в Java. Помимо этого, были изучены методы equals(), toString(), hashCode(), а также принципы SOLID.