درس برنامه نویسی شی گرا

استاد: بیژن وثوقی وحدت

دانشگاه صنعتی شریف

دانشکده مهندسی برق

محمد حسین شفیعی زادگان

99104781

mohammad hossein shafizadegan

گزارش سوال شریف مارکت تمرین شماره 3

معرفی کلاس های برنامه :

1- کلاس Store :

این کلاس در واقع ارتباط دهنده بخش های مربوط به فروشنده و مشتری است به صورتی که فایل های موردنیاز برای ذخیره اطلاعات در این کلاس قرار دارد و همچنین توابعی که برای هر دو سمت فروشنده و نشتری یکسان است در این کلاس پیاده سازی می شوند.

متغییرهای این کلاس به صورت زیر می باشند:

```
public class Store {
    File products = new File( pathname: "products.txt");
    File orders = new File( pathname: "orders.txt");
    File checkedOutOrders = new File( pathname: "checkedOutOrders.txt");
    File costumers = new File( pathname: "costumers.txt");
```

اطلاعات مربوط به کالاها در فایل products.txt ذخیره می شوند و فرمت ذخیره سازی آنها به این صورت است:

<Good Name>,<Good ID>,<buying price>,<selling price>,<count/weight/item>

اطلاعات مربوط به سفارش های مشتری در فایل orders.txt ذخیره می شوند و فرمت ذخیره سازی آنها به این صورت است:

<Customer ID>,<Good ID>,<count/weight/item>,<Order ID>

اطلاعات مربوط به سفارش های بررسی شده در فایلcheckedOutOrders.txt ذخیره می شوند و فرمت ذخیره سازی آنها مانند سفارش های مشتری است.

اطلاعات مربوط به مشتری ها در فایل costumers.txt ذخیره می شوند و فرمت ذخیره سازی آنها به این صورت است: <Costumer ID>,<password>

متدهای این کلاس به صورت زیر می باشد:

```
public void allProducts(){...}
public void availableProducts(){...}
public void notAvailableProducts(){...}
public void allOrders(){...}
public void checkedOutOrder(){...}
```

allProducts() : نمایش تمام کالاها

availableProducts() : نمایش کالاهای موجود

allOrders() : نمایش سفارش های مشتری

checkedOutOrder() : نمایش سفارش های بررسی شده

2- کلاس Product :

این کلاس دارای متغییرهای زیر است:

```
public class Product {
    String name;
    int code;
    long buyingPrice;
    long sellingPrice;
    int inventory;
```

Constructor این کلاس به صورت زیر است:

```
public Product(String name, long buyingPrice, long sellingPrice, int inventory) {
    this.name = name;
    this.buyingPrice = buyingPrice;
    this.sellingPrice = sellingPrice;
    this.inventory = inventory;
    Random random = new Random();
    this.code = random.nextInt( bound: 9000)+1000;
}
```

در اینجا یک عدد 4رقمی به صورت تصادفی به عنوان کد کالا تعیین می شود.

3- کلاس Order :

این کلاس دارای متغییرهای زیر است:

```
public class Order {
    String costumerID;
    int productCode;
    int productInventory;
    String orderID;
```

Constructor این کلاس به صورت زیر است:

```
public Order(String costumerID, int productCode, int productInventory,int orderNum) {
    this.costumerID = costumerID;
    this.productCode = productCode;
    this.productInventory = productInventory;
    this.orderID = costumerID;
    for (int i = 0; i < 4-String.valueOf(orderNum).length(); i++) {
        orderID += "0";
    }
    this.orderID += String.valueOf(orderNum);
```

شناسه سفارش از شناسه مشتری و شماره سفارش تشکیل شده است. شماره سفارش با توجه به کل سفارش ها به دست می آید و برابر است با حاصل جمع تعداد سفارش های مشتری ها و تعداد سفارش های بررسی شده.

4- کلاس FileManager :

این کلاس به منظور سهولت در استفاده از فایل و ذخیره سازی و خواندن اطلاعات از فایل ایجاد شده است.

متد های آن به صورت زیر است:

```
public class FileManager {
    public static void addToFile(File file, String input) {...}

    public static void remove(File file, String input) {...}

    public static boolean isFind(File file, String input) {...}

    public static String getUserPass (File file, String userID) {...}

    public static int getProductNum (File file, int productID) {...}

    public static int getFileItems (File file) {...}
}
```

- در متد addToFile با گرفتن یک فایل و یک رشته به عنوان ورودی، آن را رشته را به آخر آن فایل اضافه می کند.
- در متد remove با گرفتن یک فایل و یک رشته به عنوان ورودی، آن را رشته را از فایل حذف می کند به این صورت که ابتدا خطوطی از فایل را که برابر آن رشته نیستند را درون یک لیست کمکی ذخیره می کند و سپس تمام رشته های موجود در لیست کمکی را از اول درون فایل می نویسد.
- در متد isFind با گرفتن یک فایل و یک رشته به عنوان ورودی، آن را رشته را در فایل جست و جو می کند و در صورت یافتن آن true و در غیراین صورت false برمی گرداند.
- در متد getUserPass با گرفتن یک فایل و یک رشته (شناسه مشتری) به عنوان ورودی، با پیمایش خط به خط فایل costumers.txt در صورت یافتن شناسه مشتری، رمز آن را به عنوان یک String برمی گرداند.
- در متد getProductNum با گرفتن شناسه کالا، تعداد آن را در فایل products.txt جست و جو می کند و آن را برمی گرداند.
- در متد getFileItems تعداد خطوط فایل که در واقع برابر است با تعداد موارد ذخیره شده در فایل را برمی گرداند.

5- کلاس CustomerManager :

این کلاس برای دسترسی و هندل کردن توابع موردنیاز یک مشتری ایجاد شده است.

```
public class CustomerManager {
    Store store = new Store();

    public void showAllProducts() { store.allProducts(); }

    public void showAvailableProducts() { store.availableProducts(); }

    public void showNAProducts() { store.notAvailableProducts(); }

    public void registerOrder (String costumerID, int productID, int num){...}

    public void cancelOrder(String orderID){...}

    public void update(int productID, int count){...}
}
```

6- کلاس CustomerMain :

این کلاس نقطه شروع برنامه برای مشتری است و تابع main در آن قرار دارد.

در این کلاس یک متغییر customerManager از نوع CustomerManager ساخته می شود.

همچنین این کلاس یک متد detectCommand(String command) برای تشخیص و فهمیدن (parse) دستور ورودی کاربر دارد و بعد از تشخیص دستور، تابع های موردنیاز را با استفاده از costumerManager صدا می زند. همچنین کاربر برای ورود به برنامه احراز هویت می شود.

7- کلاس AdminManager :

این کلاس برای دسترسی و هندل کردن توابع موردنیاز فروشنده ایجاد شده است.

```
public class AdminManager {
    Store store = new Store();

    public void showAllProducts() { store.allProducts(); }

    public void showAvailableProducts() { store.availableProducts(); }

    public void showNAProducts() { store.notAvailableProducts(); }

    public void showAllOrders() { store.allOrders(); }

    public void checkOutOrder(String orderID){...}

    public void showCheckedOutOrders() { store.checkedOutOrder(); }

    public void addProduct(String productName, int count, long buyPrice, long sellPrice){...}

    public void removeProduct(int productID){...}

    public void editProduct(int productID, String newName){...}

    public void editProduct(int productID, String newName, int newCount){...}

    public void editProduct(int productID, int newCount){...}

    public void editProduct(int productID, long newSellPrice, long newBuyPrice, int newCount){...}

    public void calculateBenefit(){...}

    public void calculateSPB(int productID){...}

    public void calculateTotalSales(){...}

    public void calculateSPS(int productID){...}
```

- در متد checkOutOrder با پیدا کردن سفارش موردنظر از فایل orders.txt آن خط مربوط به آن سفارش را در یک متغییر کمکی ذخیره می کنیم و سپس آن خط را از فایل orders.txt حذف می کنیم و در آخر آن متغییر کمکی را به فایل checkedOutOrder.txt اضافه می کنیم.
- در متدهای editProduct بعد از پیدا کردن کالا موردنظر از فایل products.txt بخش های آن را در آرایه از String ذخیره می کنیم و بعد از اعمال تغییرات موردنظر روی ویژگی های کالا، ابتدا آن خط قبلی را از فایل خذف کرده و سپس خط جدیدی را که با توجه به تغییرات درست کرده ایم را به فایل اضافه می کنیم.
- در متد calculateBenefit سود کلی حاصل از فروش را محاسبه می کنیم. برای این کار باید از بین سفارش های بررسی شده در فایل checkedOutOrders.txt برای هر سفارش تعداد کالا خریده شده را و کد کالا را به دست آوریم و بعد در فایل products.txt قیمت خرید و فروش کالا را به دست آوریم سپس با ضرب تعداد کالا در اختلاف قیمت فروش و خرید، سود حاصل از هر سفارش به دست می آید. با انجام این کار برای هر سفارش و جمع کردن مقادیر به دست آمده، سود کل به دست می آید.
- در متد calculateSPB سود حاصل از فروش یک کالا به دست می آید. روش محاسبه مانند بالا است با این تفاوت که فقط سفارش هایی با کد کالا موردنظر بررسی می شود.
- در متد calculateTotalSales فروش کلی را محاسبه می کنیم. برای این کار باید از بین سفارش های بررسی شده در فایل checkedOutOrders.txt برای هر سفارش تعداد کالا خریده شده را و کد کالا را به دست آوریم و بعد در فایل products.txt قیمت فروش کالا را به دست آوریم سپس با ضرب تعداد کالا در قیمت فروش، فروش حاصل از هر سفارش به دست می آید. با انجام این کار برای هر سفارش و جمع کردن مقادیر به دست آمده، فروش کل به دست می آید.
- در متد calculateSPS فروش یک کالا به دست می آید. روش محاسبه مانند بالا است با این تفاوت که فقط سفارش هایی با کد کالا موردنظر بررسی می شود.

8- کلاس AdminMain :

این کلاس نقطه شروع برنامه برای فروشنده است و تابع main در آن قرار دارد.

در این کلاس یک متغییر adminManager از نوع AdminManager ساخته می شود.

همچنین این کلاس یک متد detectCommand(String command) برای تشخیص و فهمیدن (parse) دستور ورودی کاربر دارد و بعد از تشخیص دستور، تابع های موردنیاز را با استفاده از adminManager صدا می زند.