پروژه آنلاین شاپ

در این گزارش ابتدا در مورد هرفایل در پروژه توضیحاتی ارائه شده است
همچنین داخل هر فایل پروژه توضیحات کاملی در مورد کد ها نوشته شده است

سپس در مورد چالش های پروژه و پیشنهاد هایی برای بهبود آن مطالبی نوشته شده است.

Account

```
import java.util.List;
```

این خط کد کلاس List رو از پکیج java.util وارد می‌کنه. این کلاس به درد زمانی می‌خوره که بخوایم با مجموعه‌هایی از اشیاء (مثلاً لیست سفارشات یا محصولات) کار کنیم.

2. تعریف کلاس:

```
public abstract class Account {
    // ... (بقیه کد کلاس)
}
```

اینجا کلاس Account رو تعریف کردم. کلمه کلیدی abstract یعنی این یه کلاس انتزاعیه و نمی‌تونم مستقیماً ازش یه شی بسازم. در عوض، این کلاس مثل یه الگوی پایه برای انواع حساب‌های کاربری تو سیستم فروشگاهی‌ام عمل می‌کنه (مثلاً حساب مشتری، حساب مدیر و غیره).

3. متغیرهای عضو:

```
protected String username;
protected String password;
protected String email;
```

این سه تا خط، متغیرهایی هستن که اطلاعات حساب کاربری رو نگه می‌دارن:

- username: اسم کاربری حساب.
- password: رمز عبور حساب (البته تو یه برنامه واقعی، رمز عبور رو به صورت هش‌شده ذخیره می‌کنم).
- email: آدرس ایمیل حساب.

کلمه کلیدی protected یعنی این متغیرها هم تو کلاس Account و هم تو همه زیر کلاس‌هاش قابل دسترسی هستن.

4. سازنده (Constructor):

```
public Account(String username, String password, String email) {
    this.username = username;
    this.password = password;
    this.email = email;
}
```

این یه سازنده‌ست، یه متد خاص که هر وقت یه شی از کلاس Account یا یکی از زیر کلاس‌هاش ساخته می‌شه، صدا زده می‌شه. این سازنده سه تا پارامتر می‌گیره (اسم کاربری، رمز عبور و ایمیل) و ازشون برای مقداردهی اولیه متغیرهای عضو استفاده می‌کنه.

5. متدهای Getter و Setter:

```
public String getUsername() { ... }
public String getPassword() { ... }
public String getEmail() { ... }
public void setPassword(String password) { ... }
public void setEmail(String email) { ... }
public void setUsername(String username) { ... }
```

این متدها برای دسترسی و تغییر متغیرهای عضو استفاده می‌شن. متدهای getter (مثل getUsername) مقادیر متغیرها رو برمی‌گردونن، در حالی که متدهای setter (مثل setPassword) مقادیر جدیدی رو برای متغیرها تنظیم می‌کنن.

6. متدهای info():

```
public String info() { ... }
public String info(List<Order> orders, int id) { ... }
public String info(List<Product> products) { ... }
```

این سه تا متد برای نمایش اطلاعات حساب به صورت رشته طراحی شدن. نسخه اول (info()) فقط اطلاعات اولیه حساب رو نشون می‌ده. دو تا نسخه دیگه برای نمایش اطلاعات مربوط به سفارش‌ها و محصولات در نظر گرفته شدن، اما فعلاً کامل پیاده‌سازی نشدن.

Admin

هدف کلاس:

کلاس Admin یه جورایی "مدیر" سیستم فروشگاهی ماست. این کلاس از کلاس Account ارث‌بری می‌کنه، یعنی همه ویژگی‌های یه حساب کاربری (مثل نام کاربری، رمز عبور و ایمیل) رو داره، به علاوه قابلیت‌های مدیریتی خاص خودش.

متغیرهای عضو:

- id: این یه شناسه منحصر به فرده که به هر مدیر اختصاص داده می‌شه.

سازنده (Constructor):

- Admin(String username, String password, String email, int id): این سازنده، اطلاعات اولیه مدیر (شامل نام کاربری، رمز عبور، ایمیل و شناسه) رو می‌گیره و متغیرهای عضو رو مقداردهی می‌کنه. از super(username, password, email) هم برای صدا زدن سازنده کلاس پدر (Account) استفاده می‌کنه تا اون متغیرها هم مقداردهی بشن.

متدها:

- getId(): این متد شناسه مدیر رو برمی‌گردونه.
- addUser(User user, Shop shop): این متد یه کاربر جدید به فروشگاه اضافه می‌کنه.
- removeUser(User user, Shop shop): این متد یه کاربر رو از فروشگاه حذف می‌کنه.
- approveSeller(Seller seller): این متد یه فروشنده رو تأیید می‌کنه.
- addProduct(Product product, Shop shop): این متد یه محصول جدید به فروشگاه اضافه می‌کنه.
- removeProduct(Product product, Shop shop): این متد یه محصول رو از فروشگاه حذف می‌کنه.
- viewUsers(Shop shop): این متد لیست کاربران فروشگاه رو نشون می‌ده. این کار رو با پیمایش لیست حساب‌های فروشگاه (shop.getAccounts()) و بررسی اینکه هر حساب از نوع User هست یا نه، انجام می‌ده.

User

هدف کلاس:

کلاس User اطلاعات و رفتارهای مربوط به یه کاربر عادی تو سیستم فروشگاه رو تعریف می‌کنه. این کلاس هم از کلاس Account ارث‌بری می‌کنه، یعنی همه ویژگی‌های یه حساب کاربری (مثل نام کاربری، رمز عبور و ایمیل) رو داره، به علاوه اطلاعات و قابلیت‌های خاص یه کاربر عادی.

متغیرهای عضو:

- phoneNumber: شماره تلفن کاربر.
- address: آدرس کاربر.
- cart: سبد خرید کاربر (یه لیست از اشیاء OrderItem).
- purchasedProducts: لیست محصولاتی که کاربر خریده (یه لیست از اشیاء OrderItem).
- wallet: موجودی کیف پول کاربر.

سازنده (Constructor):

- User(String username, String password, String email, String phoneNumber, String address) :

این سازنده اطلاعات اولیه کاربر (مثل نام کاربری، رمز عبور، ایمیل، شماره تلفن و آدرس) رو می‌گیره و متغیرهای عضو رو مقداردهی می‌کنه. از super(username, password, email) هم برای صدا زدن سازنده کلاس پدر (Account) استفاده می‌کنه تا اون متغیرها هم مقداردهی بشن. لیست‌های cart و purchasedProducts هم به صورت لیست‌های خالی مقداردهی میشن و موجودی کیف پول (wallet) هم صفر می‌شه.

متدهای Getter و Setter:

- getPhoneNumber(), getAddress(), getCart(), getPurchasedProducts(), getWallet(): این متدها مقادیر متغیرهای عضو مربوطه رو برمی‌گردونن.
- setPhoneNumber(String phoneNumber), setAddress(String address): این متدها مقادیر جدیدی رو برای شماره تلفن و آدرس کاربر تنظیم می‌کنن.

متدهای مربوط به سبد خرید:

- addToCart(OrderItem product): این متد یه محصول رو به سبد خرید اضافه می‌کنه.

- addToPurchasedProducts(OrderItem product): این متد یه محصول رو به لیست محصولات خریداری شده اضافه می‌کنه.
- removeFromCart(OrderItem product): این متد یه محصول رو از سبد خرید حذف می‌کنه.
- clearCart(): این متد سبد خرید رو خالی می‌کنه.

متد مربوط به کیف پول:

- chargeWallet(double amount): این متد به موجودی کیف پول کاربر اضافه می‌کنه.

متد info():

- info(List<Order >orders, int user_id) :

این متد اطلاعات کاربر رو به همراه سفارش‌هاش به صورت یه رشته برمی‌گردونه. اول از متد info() کلاس پدر (Account) برای گرفتن اطلاعات پایه استفاده می‌کنه، بعد اطلاعات مربوط به شماره تلفن، آدرس، کیف پول، سبد خرید و محصولات خریداری شده رو بهش اضافه می‌کنه. در نهایت، سفارش‌های مربوط به کاربر رو از لیست orders پیدا می‌کنه و اطلاعاتشون رو هم به رشته اضافه می‌کنه.

Seller

هدف کلاس:

کلاس Seller اطلاعات و رفتارهای مربوط به یه فروشنده تو سیستم فروشگاه رو تعریف می‌کنه. این کلاس هم از کلاس Account ارث‌بری می‌کنه، یعنی همه ویژگی‌های یه حساب کاربری (مثل نام کاربری، رمز عبور و ایمیل) رو داره، به علاوه اطلاعات و قابلیت‌های خاص یه فروشنده.

متغیرهای عضو:

- companyName: اسم شرکت فروشنده.
- wallet: موجودی کیف پول فروشنده.
- isApproved: نشون می‌ده که فروشنده تأیید شده یا نه (در ابتدا تأیید نشده).
- id: شناسه منحصر به فرد فروشنده.

سازنده (Constructor):

- Seller(String username, String password, String email, int id, String companyName) :

این سازنده اطلاعات اولیه فروشنده (مثل نام کاربری، رمز عبور، ایمیل، شناسه و اسم شرکت) رو می‌گیره و متغیرهای عضو رو مقداردهی می‌کنه. از super(username, password, email) هم برای صدا زدن سازنده کلاس پدر (Account) استفاده می‌کنه تا اون متغیرها هم مقداردهی بشن. موجودی کیف پول (wallet) هم صفر می‌شه و وضعیت تأیید (isApproved) هم در ابتدا false (تأیید نشده) قرار می‌گیره.

متدهای Getter و Setter:

- getCompanyName ,()getId ,()getWallet ,()isApproved :()

این متدها مقادیر متغیرهای عضو مربوطه رو برمی‌گردونن.

- setCompanyName(String companyName) ,setApproved(boolean approved) :

این متدها مقادیر جدیدی رو برای اسم شرکت و وضعیت تأیید فروشنده تنظیم می‌کنن.

متد مربوط به پرداخت:

- receivePayment(double amount): این متد یه مبلغ رو به عنوان پرداخت دریافت می‌کنه و به موجودی کیف پول فروشنده اضافه می‌کنه. البته قبلش چک می‌کنه که مبلغ مثبت باشه.

متد info():

- info(List<Product >products)

این متد اطلاعات فروشنده رو به همراه محصولاتی که می‌فروشه به صورت یه رشته برمی‌گردونه. اول از متد info() کلاس پدر (Account) برای گرفتن اطلاعات پایه استفاده می‌کنه، بعد اطلاعات مربوط به اسم شرکت، کیف پول و وضعیت تأیید رو بهش اضافه می‌کنه. در نهایت، محصولات مربوط به فروشنده رو از لیست products پیدا می‌کنه و اطلاعاتشون رو هم به رشته اضافه می‌کنه.

Bank

هدف کلاس:

کلاس Bank یه جورایی "حساب بانکی" داخل سیستم فروشگاه ماست. این کلاس اطلاعات مربوط به موجودی حساب و اینکه این حساب به کدوم کاربر تعلق داره رو نگه می‌داره.

متغیرهای عضو:

- money: این متغیر نشون می‌ده چقدر پول تو حساب بانکی وجود داره.
- user_id: این متغیر مشخص می‌کنه که این حساب بانکی به کدوم کاربر تعلق داره.

سازنده (Constructor):

- Bank(double money, int user_id): این سازنده اطلاعات اولیه حساب بانکی (شامل موجودی اولیه و شناسه کاربر) رو می‌گیره و متغیرهای عضو رو مقداردهی می‌کنه.

متدها:

- getMoney(): این متد موجودی حساب بانکی رو برمی‌گردونه.
- getUserID(): این متد شناسه کاربری که حساب بهش تعلق داره رو برمی‌گردونه.

Category

هدف کلاس:

کلاس Category اطلاعات مربوط به دسته‌بندی محصولات تو فروشگاه رو نگه می‌داره. این کلاس به ما کمک می‌کنه تا محصولات رو به صورت سلسله مراتبی (دسته‌بندی اصلی، زیر دسته‌ها و ...) سازماندهی کنیم.

متغیرهای عضو:

- name: اسم دسته‌بندی.
- subcategories: لیستی از زیر دسته‌های این دسته‌بندی (اگه داشته باشه).
- parentCategory: دسته‌بندی والد این دسته‌بندی (اگه داشته باشه).

سازنده (Constructor):

- Category(String name): این سازنده اسم دسته‌بندی رو می‌گیره و متغیر name رو مقداردهی می‌کنه. لیست subcategories هم به صورت یه لیست خالی ساخته می‌شه و parentCategory در ابتدا null قرار می‌گیره (چون در ابتدا دسته‌بندی والد نداره).

متدهای Getter و Setter:

- getName(), getSubcategories(), getParentCategory(): این متدها مقادیر متغیرهای عضو مربوطه رو برمی‌گردونن.
- setName(String name): این متد اسم دسته‌بندی رو تغییر می‌ده.
- setParentCategory(Category parentCategory) :

این متد دسته‌بندی والد رو تنظیم می‌کنه. یه نکته جالب اینجاست که وقتی دسته‌بندی والد تنظیم می‌شه، اسم دسته‌بندی هم به‌روزرسانی می‌شه تا اسم دسته‌بندی والد رو هم شامل بشه (مثلاً اگه دسته‌بندی والد "پوشاک" باشه و اسم دسته‌بندی "مردانه" باشه، اسم دسته‌بندی به "پوشاک - مردانه" تغییر می‌کنه).

متدهای دیگه:

- addSubcategory(Category subcategory): این متد یه زیر دسته رو به لیست subcategories اضافه می‌کنه و دسته‌بندی والد اون زیر دسته رو هم به this (یعنی دسته‌بندی فعلی) تنظیم می‌کنه.
- toString(): این متد یه نمایش متنی از دسته‌بندی (که در واقع اسم دسته‌بندی هست) برمی‌گردونه. این متد برای وقتی که بخوایم اطلاعات دسته‌بندی رو چاپ کنیم یا تو جاهای دیگه ازش استفاده کنیم، مفیده.

Category

هدف کلاس:

کلاس Category اطلاعات مربوط به دسته‌بندی محصولات تو فروشگاه رو نگه می‌داره. این کلاس به ما کمک می‌کنه تا محصولات رو به صورت سلسله مراتبی (دسته‌بندی اصلی، زیر دسته‌ها و ...) سازماندهی کنیم.

متغیرهای عضو:

- name: اسم دسته‌بندی.
- subcategories: لیستی از زیر دسته‌های این دسته‌بندی (اگه داشته باشه).
- parentCategory: دسته‌بندی والد این دسته‌بندی (اگه داشته باشه).

سازنده (Constructor):

- Category(String name): این سازنده اسم دسته‌بندی رو می‌گیره و متغیر name رو مقداردهی می‌کنه. لیست subcategories هم به صورت یه لیست خالی ساخته می‌شه و parentCategory در ابتدا null قرار می‌گیره (چون در ابتدا دسته‌بندی والد نداره).

متدهای Getter و Setter:

- getName(), getSubcategories(), getParentCategory(): این متدها مقادیر متغیرهای عضو مربوطه رو برمی‌گردونن.
- setName(String name): این متد اسم دسته‌بندی رو تغییر می‌ده.
- setParentCategory(Category parentCategory): این متد دسته‌بندی والد رو تنظیم می‌کنه. یه نکته جالب اینجاست که وقتی دسته‌بندی والد تنظیم می‌شه، اسم دسته‌بندی هم به‌روزرسانی می‌شه تا اسم دسته‌بندی والد رو هم شامل بشه (مثلاً اگه دسته‌بندی والد "پوشاک" باشه و اسم دسته‌بندی "مردانه" باشه، اسم دسته‌بندی به "پوشاک - مردانه" تغییر می‌کنه).

متدهای دیگه:

- addSubcategory(Category subcategory): این متد یه زیر دسته رو به لیست subcategories اضافه می‌کنه و دسته‌بندی والد اون زیر دسته رو هم به this (یعنی دسته‌بندی فعلی) تنظیم می‌کنه.
- toString(): این متد یه نمایش متنی از دسته‌بندی (که در واقع اسم دسته‌بندی هست) برمی‌گردونه. این متد برای وقتی که بخوایم اطلاعات دسته‌بندی رو چاپ کنیم یا تو جاهای دیگه ازش استفاده کنیم، مفیده.

Order

هدف کلاس:

کلاس Order اطلاعات مربوط به یه سفارش رو تو سیستم فروشگاه نگه می‌داره. این اطلاعات شامل لیست محصولات سفارش، هزینه کل، وضعیت سفارش (تحویل شده یا در انتظار) و شناسه کاربری که سفارش رو ثبت کرده، می‌شه.

متغیرهای عضو:

- products: لیستی از محصولات سفارش (از نوع OrderItem).
- totalCost: هزینه کل سفارش.
- status: وضعیت سفارش (true یعنی تحویل شده، false یعنی در انتظار).
- user_id: شناسه کاربری که سفارش رو ثبت کرده.

سازنده (Constructor):

- Order(List<OrderItem >products, int user_id)

این سازنده لیست محصولات و شناسه کاربر رو می‌گیره و متغیرهای عضو رو مقداردهی می‌کنه. لیست محصولات با استفاده از new ArrayList ساخته می‌شه و بعد با لیست ورودی مقداردهی می‌شه. وضعیت سفارش هم در ابتدا به false (در انتظار) تنظیم می‌شه. در نهایت، متد calculateTotalCost() برای محاسبه هزینه کل سفارش صدا زده می‌شه.

متدهای Getter:

- getProducts(), getTotalCost(), getStatus(), getUserID(): این متدها مقادیر متغیرهای عضو مربوطه رو برمی‌گردونن.

متد Setter:

- setStatus(boolean status): این متد وضعیت سفارش رو تغییر می‌ده.

متدهای دیگه:

- calculateTotalCost(): این یه متد خصوصیه (private) که هزینه کل سفارش رو با جمع کردن قیمت هر محصول (ضرب در تعدادش) محاسبه می‌کنه. این متد فقط از داخل کلاس Order قابل صدا زدنه.

- toString(): این متد یه نمایش متنی از سفارش (شامل شناسه کاربر، وضعیت، هزینه کل و لیست محصولات) برمی‌گردونه. این متد برای وقتی که بخوایم اطلاعات سفارش رو چاپ کنیم یا تو جاهای دیگه ازش استفاده کنیم، مفیده.

نکته:

- تو این کد، فرض بر اینه که کلاس OrderItem هم وجود داره و متدها و متغیرهای لازم (مثل getProduct_name(), getQuantity() و getPrice()) رو داره.

OrderItem

هدف کلاس:

کلاس OrderItem اطلاعات مربوط به یه آیتم (یا محصول) توی یه سفارش رو نگه می‌داره. این اطلاعات شامل شناسه محصول، تعداد، نام و قیمت می‌شه.

متغیرهای عضو:

- product_id: شناسه منحصر به فرد محصول.
- quantity: تعداد این محصول که تو سفارش هست.
- product_name: اسم محصول.
- price: قیمت واحد محصول.

سازنده (Constructor):

- OrderItem(int product_id, String product_name, int quantity, double price): این سازنده اطلاعات اولیه آیتم سفارش (شامل شناسه، نام، تعداد و قیمت) رو می‌گیره و متغیرهای عضو رو مقداردهی می‌کنه.

متدهای Getter و Setter:

- getPrice(), getProduct_id(), getQuantity(), getProduct_name(): این متدها مقادیر متغیرهای عضو مربوطه رو برمی‌گردونن.
- setProduct_id(int product_id), setQuantity(int quantity): این متدها مقادیر جدیدی رو برای شناسه و تعداد محصول تنظیم می‌کنن.

نکته:

- تو این کد، فرض بر اینه که یه کلاس Product وجود داره که اطلاعات محصولات رو نگه می‌داره و یه شناسه (id) و اسم (name) برای هر محصول داره.

Product

هدف کلاس:

کلاس Product اطلاعات مربوط به یه محصول تو فروشگاه رو نگه می‌داره. این اطلاعات شامل شناسه فروشنده، نام محصول، قیمت، موجودی انبار، نظرات کاربران، دسته‌بندی و اطلاعات اضافی می‌شه.

متغیرهای عضو:

- seller_id: شناسه فروشنده‌ای که این محصول رو گذاشته برای فروش.
- name: اسم محصول.
- price: قیمت محصول.
- stockQuantity: تعداد موجودی محصول تو انبار.
- comments: لیستی از نظرات کاربران درباره محصول.
- category: دسته‌بندی محصول (مثلاً "کتاب"، "لپ‌تاپ" و ...).
- additionalData: یه دیکشنری (از نوع Map) که اطلاعات اضافی درباره محصول رو نگه می‌داره (مثلاً رنگ، سایز، جنس و ...).

سازنده (Constructor):

- Product(int seller_id, String name, double price, int stockQuantity, String category):

این سازنده اطلاعات اولیه محصول (مثل شناسه فروشنده، اسم، قیمت، موجودی و دسته‌بندی) رو می‌گیره و متغیرهای عضو رو مقداردهی می‌کنه. لیست نظرات (comments) و دیکشنری اطلاعات اضافی (additionalData) هم به صورت خالی ساخته می‌شن.

متدهای Getter:

- getSeller_id(), getName(), getPrice(), getStockQuantity(), getComments(), getCategory(), getAdditionalData(): این متدها مقادیر متغیرهای عضو مربوطه رو برمی‌گردونن.

متدهای Setter:

- setPrice(double price) ,setStockQuantity(int stockQuantity) , setCategory(String category) :

  این متدها مقادیر جدیدی رو برای قیمت، موجودی انبار و دسته‌بندی محصول تنظیم می‌کنن.

متدهای دیگه:

- addComment(String comment): این متد یه نظر جدید رو به لیست نظرات اضافه می‌کنه.
- addAdditionalData(String key, String value): این متد یه جفت کلید-مقدار جدید رو به دیکشنری اطلاعات اضافی اضافه می‌کنه.
- decreaseStock(int quantity): این متد موجودی انبار رو به اندازه تعداد مشخص شده کم می‌کنه. البته قبلش چک می‌کنه که موجودی کافی باشه. اگه موجودی کافی نباشه، یه پیغام خطا چاپ می‌کنه.
- toString(): این متد یه نمایش متنی از محصول (شامل همه اطلاعاتش) برمی‌گردونه. این متد برای وقتی که بخوایم اطلاعات محصول رو چاپ کنیم یا تو جاهای دیگه ازش استفاده کنیم، مفیده.

Shop

هدف کلاس:

کلاس Shop هسته اصلی سیستم فروشگاه ما رو تشکیل می‌ده. این کلاس اطلاعات کلی فروشگاه (مثل اسم، آدرس وب، شماره پشتیبانی)، لیست حساب‌های کاربری (مشتری‌ها، فروشنده‌ها و مدیرها)، لیست محصولات، لیست سفارش‌ها، دسته‌بندی‌های محصولات و سود کل فروشگاه رو نگه می‌داره. به علاوه، متدهایی برای انجام عملیات‌های مختلف فروشگاه مثل ثبت‌نام، ورود، اضافه کردن محصول، تأیید فروشنده و ... داره.

متغیرهای عضو:

- name: اسم فروشگاه.
- webAddress: آدرس وب‌سایت فروشگاه.
- supportNumber: شماره تلفن پشتیبانی فروشگاه.
- accounts: لیستی از همه حساب‌های کاربری (از نوع Account).
- products: لیستی از همه محصولات فروشگاه (از نوع Product).
- orders: لیستی از همه سفارش‌های ثبت شده تو فروشگاه (از نوع Order).
- categories: لیستی از دسته‌بندی‌های محصولات (از نوع Category).
- categories_name: لیستی از اسم دسته‌بندی‌های محصولات (از نوع String).
- banks: لیستی از تراکنش‌های بانکی (از نوع Bank).
- totalProfit: سود کل فروشگاه.

سازنده (Constructor):

- Shop(String name, String webAddress, String supportNumber): این سازنده اطلاعات اولیه فروشگاه (مثل اسم، آدرس وب و شماره پشتیبانی) رو می‌گیره و متغیرهای عضو رو مقداردهی می‌کنه. لیست‌های accounts، products، orders و banks به صورت لیست‌های خالی ساخته می‌شن و سود کل (totalProfit) هم صفر می‌شه.

متدها:

- collectCategoryNames(List<Category> categories, List<String> categoryNamesList)

این متد به صورت بازگشتی اسم همه دسته‌بندی‌ها و زیر دسته‌ها رو از لیست categories استخراج می‌کنه و به لیست categoryNamesList اضافه می‌کنه.

Getters:

- متدهای getName(), getWebAddress(), getSupportNumber(), getAccounts(), getProducts(), getOrders(), getTotalProfit() برای دسترسی به متغیرهای عضو استفاده می‌شن.
- registerUser(Scanner sc): این متد اطلاعات یه کاربر جدید رو از ورودی می‌گیره و یه حساب کاربری از نوع User براش می‌سازه و به لیست accounts اضافه می‌کنه.
- regiserSeller(Scanner sc): این متد اطلاعات یه فروشنده جدید رو از ورودی می‌گیره و یه حساب کاربری از نوع Seller براش می‌سازه و به لیست accounts اضافه می‌کنه.
- regiserAdmin(Scanner sc): این متد اطلاعات یه مدیر جدید رو از ورودی می‌گیره و یه حساب کاربری از نوع Admin براش می‌سازه و به لیست accounts اضافه می‌کنه.
- login(String username, String password): این متد نام کاربری و رمز عبور رو می‌گیره و چک می‌کنه که آیا همچین کاربری وجود داره یا نه. اگه وجود داشت، شناسه کاربری رو برمی‌گردونه، وگرنه 1- رو برمی‌گردونه.
- geAccount(int id): این متد حساب کاربری با شناسه مشخص شده رو برمی‌گردونه.
- addProduct(Scanner sc, int seller_id): این متد به یه فروشنده اجازه می‌ده که یه محصول جدید به فروشگاه اضافه کنه.
- approveSeller(Scanner sc): این متد به مدیر اجازه می‌ده که یه فروشنده رو تأیید کنه.
- editUserInfo(int user_id, Scanner sc): این متد به یه کاربر یا مدیر اجازه می‌ده اطلاعات حساب کاربری خودش رو ویرایش کنه.
- munu(Scanner sc) ,SellerMenu(Scanner sc) ,adminMenu(Scanner sc), userMenu(Scanner sc): این متدها منوهای مختلف رو برای کاربر، فروشنده و مدیر نشون می‌دن.
- increaceWallet(int user_id, Scanner sc): این متد به یه کاربر اجازه می‌ده موجودی کیف پولش رو افزایش بده.
- buy(int user_id, Scanner sc): این متد به یه کاربر اجازه می‌ده محصول بخره.
- accept_buy(Scanner sc): این متد به مدیر اجازه می‌ده خریدها رو تأیید کنه.

- main(String[] args): این متد اصلی برنامه‌ست که یه فروشگاه می‌سازه، یه حلقه بی‌نهایت اجرا می‌کنه تا کاربر از برنامه خارج بشه، و تو هر بار اجرای حلقه، منو رو نشون می‌ده و بسته به انتخاب کاربر، متدهای مربوطه رو صدا می‌زنه.

چالش ها :

نمایش اطلاعات برای هر اکانت :

به علت داشتن اکانت های مختلف و متغییر ها مختلف از چندریختی برای نمایش اطلاعات مربوط به نمایه هر اکانت استفاده کردیم
به این صورت که تابع info را به شکل های مختلفی نوشتیم برای هر اکانت

دریافت ورودی :

برای جلوگیری از ساخت اسکنر های مختلف یک اسکنر ساختیم و به توابعی که در آنها نیاز به گرفتن اطلاعات از کاربر بود آن اسکنر را ارسال کرده و از آن استفاده کردیم.

افزایش موجودی به اکانت ها:

برای این کار یک کلاس به اسم Bank ایجاد کردیم که شناسه کاربر و مبلغ درخواستی را در آن ذخیره کنیم.

تا همه تراکنش ها را داشته باشیم و هر کدام از ادمین ها تایید کرد موجودی حساب افزایش یابد.

---

cart.clear(); // خالی کردن سبد خرید

از این کد برای خالی کردن سبد خرید استفاده شد
اما به علت این که تمام اشیا را پاک میکرد در دیگر ماتغیر ها هم در دسترس نبودنت
برای همین از کد زیر برای خالی کردن سبد خرید اضافه شد.

cart = new ArrayList<>(); // خالی کردن سبد خرید با ایجاد یک لیست جدید

برای یکپارچه تر کردن برنامه

تمامی سفارشات تایید شده و تایید نشده همه در یک لیست داخل فروشگاه ذخیره میشوند و در کلاس کاربر ها ذخیره نمیشوند

هر کاربر با توجه به شماسه خود میتواند به سفارشات خود دسترسی داشته باشد.

ایجاد شناسه :

برای همه اکانت ها و محصولات و پرداخت ها و هر چیزی که در یک لیست وجود داشت ، یک شناسه وجود دارد و شناسه همان شماره اندیس ان متغیر یا شی در لیست هست

این کار به دسترسی سریع تر و ساده تر برنامه کمک میکند.

کلاس کتگوری:

از آنجایی که میتوان تعداد بسیار زیادی کتگوری ساخت و هر کتگوری میتواند زیر مجموعه های زیادی داشته باشد و نیاز به ساخت کلاس ها مختلف هست از ایده دیگری استفاده میکنیم:

استفاده از یک کلاس با ساختار درختی

این کار کمک میکند تا بتوان هر تعداد که میخواهیم کتگوری و زیر دسته ایجاد کنیم و زیر دسته ها را تا تعداد بینهایت ادامه بدهیم.

```
this.categories = new ArrayList<>(); // مقداردهی اولیه به لیست دسته‌بندی‌ها
this.categories.add(new Category(name:"Electronics")); // اضافه کردن دسته‌بندی "Electronics" به لیست دسته‌بندی‌ها
this.categories.add(new Category(name:"Clothing")); // اضافه کردن دسته‌بندی "Clothing" به لیست دسته‌بندی‌ها
this.categories.add(new Category(name:"Books")); // اضافه کردن دسته‌بندی "Books" به لیست دسته‌بندی‌ها

// اضافه کردن دسته‌بندی‌های فرعی به دسته‌بندی‌ها
this.categories.get(index:0).addSubcategory(new Category(name:"Laptops")); // اضافه کردن دسته‌بندی "Laptops" به دسته‌بندی
                                                                           // "Electronics"
this.categories.get(index:0).addSubcategory(new Category(name:"Phones")); // اضافه کردن دسته‌بندی "Phones" به دسته‌بندی
                                                                          // "Electronics"
this.categories.get(index:1).addSubcategory(new Category(name:"Shirts")); // اضافه کردن دسته‌بندی "Shirts" به دسته‌بندی
                                                                          // "Clothing"
this.categories.get(index:1).addSubcategory(new Category(name:"Pants")); // اضافه کردن دسته‌بندی "Pants" به دسته‌بندی
                                                                         // "Clothing"
this.categories.get(index:2).addSubcategory(new Category(name:"Novels")); // اضافه کردن دسته‌بندی "Novels" به دسته‌بندی
                                                                          // "Books"
this.categories.get(index:2).addSubcategory(new Category(name:"Poems")); // اضافه کردن دسته‌بندی "Poems" به دسته‌بندی
                                                                         // "Books"
this.categories.get(index:2).getSubcategories().get(index:1).addSubcategory(new Category(name:"Poems_1")); // اضافه کردن
                                                                                                           // دسته‌بندی "Poems_1"
                                                                                                           // به دسته‌بندی فرعی
                                                                                                           // "Poems"

this.categories_name = new ArrayList<>(); // مقداردهی اولیه به لیست نام دسته‌بندی‌ها
collectCategoryNames(categories, categories_name); // جمع‌آوری نام دسته‌بندی‌ها و اضافه کردن به لیست نام
                                                   // دسته‌بندی‌ها
```

در کد بالا مشاهده میشود که چه کتگوری ها و زیر دسته هایی ایجاد شده است در اخر همه را صورت یک لیست در می آوریم تا فروشنده بتواند از بین آن ها هرکدام را که میخواهد انتخاب کند.

در هنگام انتخاب کتگوری هم لیست زیر به فروشنده نمایش داده میشود

```
Select the category of the product:
0. Electronics
1. Electronics - Laptops
2. Electronics - Phones
3. Clothing
4. Clothing - Shirts
5. Clothing - Pants
6. Books
7. Books - Novels
8. Books - Poems
9. Books - Poems - Poems_1
```

دسترسی کاربر ها :

برای جلوگیری از دسترسی کاربر به قابلیت های اکانت ادمین یا فروشنده به قابلیت های اکانت ادمین تمام دسترسی ها و عملیات ها را در قسمت مربوطه برای هر کاربر میگذاریم:

وقتی کاربری وارد شد برسی میکنیم که اکانت متعلق به چه کلاسی هست

```
// حساب کاربری عادی (User)
if (account instanceof User) {…

// حساب فروشنده
else if (account instanceof Seller) {…

// حساب مدیر
else if (account instanceof Admin) { // اگر حساب کاربری از نوع Admin باشد…
```

با این کار اجازه دسترسی غیر مجاز به افراد داده نمیشود.

تایید خرید توسط ادمین:

یکی از مشکلات رایج تایید خرید توسط ادمین بود زمانی که 2 تا کاربر همزمان خرید را انجام داده بودند

مثلا در انبار 20 عدد سیب وجود داشت

کاربر 1 درخواست خرید 15 سیب را داده بود اما چون خرید تایید نشده بود تعداد کم نشده بود

کاربر 2 درخواست خرید 8 سیب را داده بود اما چون خرید تایید نشده بود تعداد کم نشده بود

حال وقتی ادمین میخواست هر 2 را تایید کند در صورت وجود پول کافر در حساب هر کاربر تعداد سیب درانبار منفی میشد

برای جلو گیز یاز این اتفاق مجدد شرط کافی بودن در انبار را برسی میکنیم .

افزایش حجم کد ها:

با توجه به اینکه تعداد خط ها و حجم کد ها افزایش یافت نوشتن و کامل کردن پروژه بسیار پیچیده شد.

برای مطمئن شدن از پیشرفت درست پروژه با تست کردن مداوم قسمت های پروژه تلاش شد تا از نبود خطا مطمئن شویم.

پیشنهاد هایی برای بهبود پروژه:

بخش نظرات و همین طور افزودن ویژگی اضافه به محصولات ساختار اولیه ای نیاز داشتند که پیاده شده

برای بهبود بیشتر میتوان این قسمت ها را نیز تکمیل کرد

استفاده از فایل

برای ذخیره اطلاعات میتوان از فایل برای ذخیره اطلاعات استفاده کرد تا زماتی که برنامه اجرا میشود نیاز به وارد کردن اطلاعات از ابتدا نباشد.
