اسپاتیفای

- سطح سوال : آسان
- طراح : شایان حقیقت

شما به عنوان **"Music Manager"** در شرکت **Spotify** مشغول به کار هستید. مسئولیت شما شامل مدیریت لیست‌های پخش موسیقی، ثبت تغییرات و هماهنگی با تیم‌های مختلف است. برای انجام این کار، باید از ابزار Git استفاده کنید تا کارهای مختلف خود را به صورت سازمان‌یافته انجام دهید.

۱. برای شروع کار، باید یک **"PlayList Project"** جدید راه‌اندازی کنید.
۲. نام خود را به عنوان مدیر تنظیم کنید (به نام دلخواه) و Config پروژه را بررسی کنید .
۳. با دستور touch شش تا پلی لیست با نام های playlist1.txt, playlist2.txt, playlist3.txt, playlist4.txt, playlist5.txt , playlist6.txt ایجاد کنید .
۴. با کمک دستور echo , پلی لیست ها را نام گذاری کنید .

```
"Top Hits of 2024" > playlist1.txt
"Chill Vibes" > playlist2.txt
"Workout Jams" > playlist3.txt
"Classical Favorites" > playlist4.txt
"Indie Discoveries" > playlist5.txt
```

۵. پلی لیست های playlist2.txt و playlist3.txt و playlist4.txt و playlist5.txt را به گیت اضافه کنید سپس با پیغام "Added initial playlists" کامیت کنید .
۶. اطلاعات درون دو فایل playlist2.txt و playlist5.txt را بدین صورت تغییر بدید .

```
"Updated Chill Vibes" > playlist2.txt
"Updated Indie Discoveries" > playlist5.txt
```

۷. فایل playlist5.txt را به گیت افزوده به و سپس وضعیت را **مانند untracked, modified,...** بررسی کنید .
۸. تغییرات خود را کامیت کرده (با پیغام "Updated playlists") و تاریخچه تغییرات را در یک خط مشاهده کنید.
۹. تغییرات در **"لیست پخش Chill Vibes"** یا playList2 را بازگردانی کنید .

۱۰. یک تگ با نسخه 1.0v ایجاد کرده با پیغام "Final version of playlists" برچسب گذاری کنید.

۱۱. فایل gitignore. ایجاد کرده و playlist6.txt را از نادیده بگیرید .

۱۲. دو **"شاخه لیست پخش جدید"** ایجاد کرده با نام های playlistUpdateA و playlistUpdateB ایجاد کنید .

۱۳. وارد پلی لیست playlistUpdateA شوید و playlist7.txt را با محتوای "Summer Hits 2024" ایجاد کنید سپس این فایل را به گیت اضافه و سپس کامیت کنید (با پیغام : "Added new summer hits playlist") .

۱۴. به شاخه main بازگردید و شاخه قبلی را ترکیب کنید .

۱۵. تمامی این عملیات را در محیط‌های توسعه مدرن مانند **VS Code** یا **IntelliJ IDEA** انجام دهید و **تصویر نمودار GitGraph** را به مستندات خود اضافه کنید.

۱۶. در ترمینال کد های خود را اجرا کنید و کد ها را در فایل جدا ارسال کنید و فایل های ایجاد شده را به همراه عکس **Gitgraph** در قالب zip ارسال کنید.

E-commerce app

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: متوسط
- طراح: آرمان اسدی

یک **E-commerce app** برنامه ای کاربردی است که به کاربران اجازه می دهد محصولات و خدمات را به صورت آنلاین خریداری کنند و بفروشند. در این سوال قصد داریم یک E-commerce app ساده را برای مدیریت یک فروشگاه پیاده‌سازی کنیم. ابتدا فایل خام پروژه را از این لینک دانلود کنید. سپس کلاس های زیر را همانطور که از شما خواسته شده پیاده‌سازی کنید.

کلاس Product

فیلدها

این کلاس دارای فیلدهای زیر است:

```
private int id;
private String name;
private double price;
private int quantityInStock;
```

- `id` : شناسه یکتای محصول.
- `name` : نام محصول
- `price` : قیمت محصول.
- `quantityInStock` : تعداد موجودی محصول در انبار.

متد کانستراکتور

متد کانستراکتور این کلاس به شکل زیر است:

```java
Product(int id, String name, double price, int quantityInStock);
```

متدها

این کلاس دارای متدهای زیر است:

```java
public boolean isInStock();
```

این متد بررسی می کند که آیا محصول در انبار موجود است یا خیر. در صورتی که محصول موجود بود `true` و در غیر این صورت `false` بر می‌گرداند.

```java
public boolean reduceStock(int quantity);
```

این متد تعداد موجودی محصول را به میزان داده شده (quantity) کاهش می‌دهد و `true` بر می‌گرداند. در صورتی که تعداد موجودی محصول در انبار ناکافی باشد این متد `false` بر می‌گرداند و موجودی را تغییر نمی دهد.

```java
public void increaseStock(int quantity);
```

این متد تعداد موجودی محصول رابه میزان داده شده افزایش می دهد.

```java
public String getDetails();
```

این متد جزئیات محصول شامل نام، قیمت و موجودی محصول را در قالب یه رشته بر می‌گرداند. رشته خروجی باید به شکل زیر باشد:

```
name: Energy Drink - Redbull 355ml
price: $25.84
quantity: 10
```

کلاس ProductItem

فیلدها

فیلدهای این کلاس به شکل زیر است:

```
private Product product;
private int quantity;
```

- `product` : محصول انتخاب شده توسط مشتری.
- `quantity` : تعداد واحدهای انتخاب شده از محصول.

متد کانستراکتور

متد کانستراکتور این کلاس به شکل زیر است:

```
ProductItem(Product product, int quantity);
```

کلاس Customer

فیلدها

این کلاس دارای فیلدهای زیر است:

```
private int id;
private String name;
private String email;
private double wallet;
private ProductItem[] shoppingCart;
```

- `id` : شناسه‌ی یکتای مشتری.
- `name` : نام مشتری.
- `email` : ایمیل مشتری.
- `wallet` : کیف پول مشتری.

- `shoppingCart` : یک آرایه از `ProductItem` ها که اقلام موجود در سبد خرید مشتری را نشان می‌دهد.

متد کانستراکتور

متد کانستراکتور این کلاس به شکل زیر است:

```
Customer(int id, String name, String email, double wallet);
```

متدها

متدهای این کلاس به صورت زیر است:

```
public void addToCart(ProductItem item);
```

این متد یک آیتم به سبد خرید مشتری اضافه می کند. دقت کنید در صورتی که محصول مربوط به آیتم داده شده از قبل در سبد خرید مشتری وجود داشته باشد، صرفا باید تعداد واحدهای انتخاب شده از محصول (quantity) افزایش یابد.

```
public boolean removeFromCart(ProductItem item);
```

این متد یک آیتم را از سبد خرید مشتری حذف می‌کند. در صورتی که آیتم داده شده در سبد خرید مشتری وجود داشته باشد، باید از سبد خرید مشتری حذف کند و مقدار `true` را بر می‌گرداند. اگر آیتم داده شده در سبد خرید مشتری وجود نداشت، مقدار `false` را بر می‌گرداند.

```
public double getTotalPrice();
```

این متد کل قیمت محصولات در سبد خرید را محاسبه و بر می‌گرداند.

```
public boolean checkout();
```

این متد فرآیند پرداخت را شبیه سازی می‌کند و پس از پرداخت، سبد خرید را خالی و قیمت نهایی خرید را از کیف پول مشتری کم می‌کند و `true` بر می‌گرداند. در صورتی که موجودی مشتری برای این خرید ناکافی باشد، مشتری نمی‌تواند این خرید را انجام دهد و این متد باید مقدار `false` را بر گرداند.

کلاس StoreManager

این کلاس برای مدیریت مشتری ها و محصولات به کار گرفته می‌شود.

فیلد ها

این کلاس شامل فیلد های زیر است:

```
private Product[] products;
private Customer[] customers;
```

- `products` : آرایه ای از محصولاتی که توسط فروشگاه ارائه می‌شود.
- `customers` : آرایه ای از مشتریان فروشگاه.

متد کانستراکتور

کانستراکتور این کلاس به شکل زیر تعریف می‌شود:

```
StoreManager(Product[] products, Customer[] customers);
```

متد ها

این کلاس شامل متد های زیر است:

```
public void addProduct(Product newProduct);
```

این متد یک محصول را به عنوان ورودی دریافت می‌کند و آنرا به محصولات فروشگاه اضافه می‌کند.

```
public void addCustomer(Customer newCustomer);
```

این متد یک مشتری جدید را به مشتریان فروشگاه اضافه می‌کند.

```
1 | public Product[] searchProductByName(String name);
```

این متد آرایه ای از محصولاتی را که رشته ورودی `name` پیشوندی از نام آنهاست به عنوان خروجی بر می‌گرداند. **این متد به بزگی و کوچی حروف حساس است (به صورت case sensitive عمل می‌کند).**

```
1 | public Customer[] searchCustomerByName(String name);
```

این متد آرایه ای از مشتریانی را که رشته ورودی `name` پیشوندی از نام آنهاست به عنوان خروجی بر می‌گرداند. **این متد به بزگی و کوچی حروف حساس است (به صورت case sensitive عمل می‌کند).**

```
1 | public void updateProductPrice(int productId, double newPrice);
```

این متد قیمت یک محصول مشخص را آپدیت می‌کند. این متد قیمت محصولی را که شناسه آن برابر با `productId` است، به `newPrice` تغییر می دهد.

چند نکته مهم

۱. تسک شما در این سوال پیاده‌سازی کلاس ها و متد های ذکر شده در صورت سوال است.
۲. شما باید کلاس ها را دقیقا همانطور که در صورت سوال آمده پیاده‌سازی کنید. به عنوان مثال نام کلاس ها، نام فیلد ها و سطح دسترسی آن ها و امضای متد ها را همانطور که در صورت سوال گفته شده بنویسید. برای این منظور پیشنهاد می شود فایل خام پروژه را دانلود کنید و آنرا توسعه دهید.
۳. شما باید برای تمامی فیلد ها **متد های گتر و ستر** بنویسید. دقت کنید که نام متدی گتر یا ستری که برای هر فیلد می‌نویسید باید مطابق الگوی زیر و به صورت استاندارد باشد:

$$(get|set) + fieldName$$

به عنوان مثال متد گتری که برای فیلد `name` می‌نویسید باید به شکل زیر باشد:

```
1 | public String getName();
```

آنچه باید آپلود کنید

شما باید یک فایل zip آپلود کنید که شامل کلاس های Product.java, ProductItem.java, Customer.java, StoreManager.java باشد. **دقت کنید فایل های کلاس ها باید در root فایل زیپی که آپلود می‌کنید باشد. یعنی زمانی که فایل زیپ را باز می کنیم، بتوانیم فایل های کلاس ها را مشاهده کنیم.**

باغ‌وحش من

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: متوسط
- طراح: سید محمد حسینی

مهیار به‌تازگی مدیر یک باغ‌وحش شده‌است. مدیر سابق این باغ‌وحش به شیوه‌ای سنتی این باغ‌وحش را مدیریت می‌کرد اما مهیار می‌خواهد به شیوه‌ای مدرن این باغ‌وحش را مدیریت کند. او شمارا استخدام کرده‌است تا برنامه‌ای در جهت مدیریت باغ‌وحش برای او بنویسید. ابتدا فایل خام پروژه را از این لینک دانلود کنید.

توجه: کلاس `Employee` و `Animal` تکمیل شده‌اند و شما می‌بایست کلاس `MyZoo` را تکمیل کنید.

کلاس Employee:

فیلدها

این کلاس دارای فیلدهای زیر است:

```
private String name;
private int property = 0;
private Degree degree;
private boolean isVip=false ;
private boolean isLoanGiven = false;
private boolean isFire =false;
private boolean isShift =false;
```

- `name` : نام کارمند استخدام شده (یکتا است و کارمندی با نام تکراری استخدام نمی‌شود).
- `property` : دارایی کارمند (میزان پول).
- `degree` : درجه آن کارمند که یک نوع enum است و جلوتر آن‌را خواهیم دید .
- `isVip` : آیا جز کارمندان ویژه است ؟

- `isLoanGiven` : آیا کارمند در حال حاضر وام گرفته است یا خیر.
- `isFire` : وضعیت کارمند که اخراج شده‌است یا خیر.
- `isShift` : آیا کارمند در حال حاضر شیفت است یا خیر.

کانستراکتور

کانستراکتور این کلاس به شکل زیر است:

```
public Employee(String name, Degree degree);
```

در سازنده کلاس کارمند از یک **enum** به‌نام Degree استفاده شده‌است که به شکل زیر تعریف می‌شود:

```
enum Degree{
    Fire, Worker, Foreman, Supervisor, Senior, Leader, Retire;
}
```

در اینجا سلسله مراتب کارمندان نشان داده شده‌است که پایین‌ترین مقام Fire و بالاترین مقام Retire می‌باشد.

این کلاس setter و getter هایی برای هریک از فیلدهای خود دارد.

کلاس Animal:

فیلدها

این کلاس دارای فیلدهای زیر است:

```
private String name ;
private int value ;
private Rarity rarity;
private boolean isStolen =false;
```

- `name` : نام حیوان.
- `value` : ارزش حیوان که یک integer است.

- `rarity` : کمیابی حیوان.
- `isStolen` : آیا حیوان دزدیده شده‌است یا خیر.

کانستراکتور

کانستراکتور این کلاس به شکل زیر است:

```
public Animal(String name, int value, Rarity rarity);
```

در اینجا هم از یک enum به‌نام Rarity استفاده شده‌است که به صورت زیر تعریف می‌شود:

```
enum Rarity{
  Common, Rare, SuperRare
}
```

این کلاس فقط شامل متدهای getter , setter است.

کلاس MyZoo:

فیلدها

این کلاس دارای فیلدهای زیر است:

```
private static Employee[] employees = new Employee[200];
private static Animal[] animals = new Animal[100];
private static int employeeCount = 0;
private static int animalCount = 0;
```

- `employees` : لیستی از کارمندان است که اگر مدیر کسی را اخراج یا بازنشسته کند، کسی را جایگزین او نمی‌کند.
- `animals` : لیستی از حیوانات باغ وحش است که در صورت دزدیده شدن حیوانی، مدیر حیوانی را جایگزین او نمی‌کند.
- `employeeCount` : تعداد کارمندان در **کل زمان ریاست**.

- `animalCount` : تعداد حیوانات در **کل زمان ریاست**.

توجه: تعداد کل کارمندان حداکثر ۲۰۰ و تعداد کل حیوانات حداکثر ۱۰۰ خواهد بود.

متدها

متدهای این کلاس به صورت زیر است:

```
public static void hire(String name, Degree degree);
```

این متد نشان‌دهنده استخدام یک کارمند جدید است. برای استخدام یک کارمند جدید ذکرکردن نام و درجه او کافیست.

```
public static void paySalary(String name);
```

این متد نشان‌دهنده پرداخت حقوق به کارمندان است که با دریافت نام کارمند حقوق او را بسته به درجه به فیلد property اضافه می‌کند. کارمندان با توجه به درجه خود حقوق دریافت می‌کنند:

۱. درجه‌ی *Worker*: حقوق ۲۰۰ تومان
۲. درجه‌ی *Foreman*: حقوق ۴۰۰ تومان
۳. درجه‌ی *Supervisor*: حقوق ۶۰۰ تومان
۴. درجه‌ی *Senior*: حقوق ۸۰۰ تومان
۵. درجه‌ی *Leader*: حقوق ۱۰۰۰ تومان

توجه: کسانی که اخراج یا بازنشسته می‌شوند حقوقی دریافت نمی‌کنند.

```
public static boolean withdraw(String name, int value);
```

متاسفانه کارمندان مجوز دسترسی به حساب خود را ندارند و برای برداشت باید به مدیرعامل نامه بزنند تا پول را از حسابشان برداشت کند و به آنها بدهد. این متد با دریافت نام کارمند و میزان پولی که کارمند میخواهد برداشت کند، آن مقدار پول را از حساب کاربر برداشت می‌کند.

توجه: اگر نام ورودی درمیان کارمندان نبود یا موجودی کاربرد از میزان درخواستی کمتر بود این متد `false` برمی‌گرداند و مقداری از حساب کاربر کسر نمی‌گردد. در غیر اینصورت مقدار `true` را برمی‌گرداند.

```
1 | public static boolean setVip(String name)
```

این متد با دریافت یک نام ، آن کارمند را به کارمند ویژه تبدیل می‌کند. اگر نام دریافتی درمیان نام کارمندان نباشد مقدار `false` و در غیر اینصورت مقدار `true` را برمی‌گرداند.

توجه:اگر کارمندی جز کارمندان ویژه باشد و این متد برای او فراخوانی شود اتفاق خاصی نخواهد افتاد.

```
1 | public static boolean giveLoan(String name);
```

این متد به کارمند مشخص شده وام می‌دهد. شرایط برای دریافت وام به صورت زیر است:

۱. کارمند وام نگرفته یا اگر گرفته آن را بازگردانده است.

۲. کارمند در درجه `Worker` نباشد.

۳. کارمند موردنظر جز کارمندان ویژه باشد.

۴. کارمند اخراج نشده باشد. در صورت داشتن شرایط زیر به میزان ۶ برابر مقدار حقوق دریافتی‌شان پول دریافت خواهند کرد.

```
1 | public static boolean payBackLoan(String name);
```

این متد برای بازگرداندن وام است. اگر کارمندی وام گرفته باشد و موجوی حساب او از دوبرابر میزان حقوق دریافتی او بیشتر باشد، آنگاه از موجودی حساب او به اندازه دوبرابر حقوق دریافتی او کسر می‌شود و او می‌تواند دوباره وام بگیرد.

```
1 | public static boolean shift(String name)
```

مدیر برای مدت مشخصی عده‌ای را شیفت برای نگهداری از حیوانات قرار می‌دهد. این متد با دریافت نام دریافتی اگر نام جز لیست نام کارمندان نباشد مقدار `false` را برمی‌گرداند. در غیر اینصورت اگر کارمند

شیفت باشد به او استراحت می‌دهد و اگر کارمند در بازه استراحت باشد، او را به حالت شیفت در می‌آورد.

```
1 | public static boolean promote(String name)
```

این متد با دریافت نام کارمند دریافتی، او را ترفیع می‌دهد. بالاترین درجه یک کارمند `Retire` است و اگر کارمندی که بازنشسته شده یا اخراج شده ترفیع بگیرد اتفاقی نخواهد افتاد. در این متد در صورتی که نام دریافتی در میان لیست کارمندان نباشد مقدار `false` و درغیر اینصورت مقدار `true` را برمی‌گرداند.

```
1 | public static boolean demotion(String name)
```

این متد با دریافت نام کارمند دریافتی او را تنزل مقام می‌دهد. اگر کارمندی در حالت بازنشستگی یا اخراج شده باشد اتفاقی برای او نخواهد افتاد. در این متد در صورتی که نام دریافتی در میان لیست کارمندان نباشد مقدار `false` و درغیر اینصورت مقدار `true` را برمی‌گرداند.

```
1 | public static void purchase(String name, int value, Rarity rarit
```

اگر مدیرعامل یک حیوان جدید بخرد این متد فراخوانی می‌شود.

```
1 | public static void steal(String name)
```

اگر حیوانی دزدیده شود این متد فراخوانی می‌شود. این متد با دریافت نام حیوان دزدیده شده تمامی کارمندانی که در آن لحظه شیفت بودند را به این شکل مجازات خواهد کرد: اگر موجودی کارمند برابر و یا بیشتر از دوبرابر ارزش آن حیوان بود، مدیرعامل این مبلغ را از موجوی کارمند کسر خواهد کرد، در غیر اینصورت آن کارمند را تنزل مقام خواهد داد. تبعا هیچ‌گاه افراد بازنشسته و اخراج شده مجازات نخواهند شد.

آنچه باید آپلود کنید

شما باید یک فایل zip آپلود کنید که شامل کلاس های Animal.java, Employee.java, MyZoo.java, و enum های Rarity و Degree است.

چاپخانه

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: متوسط
- طراح: امیرعلی وکیلی

امیرعلی به تازگی صاحب یک چاپخانه شده است و به دنبال راهکاری است تا بتواند چاپخانه خود را بهتر مدیریت کند. او در این زمینه با برنامه نویسی مشورت میکند و از آن میخواهد که شبیه سازی برای مدیریت چاپخانه او پیاده سازی کند. در این تمرین، شما به شبیه سازی این چاپخانه میپردازید.

ابتدا پروژه اولیه را از این لینک دانلود کنید.

توجه داشته باشید که می‌توانید پراپرتی و متدهای مورد نیاز خودرا اضافه کنید. تمامی پراپرتی های هر کلاس را به صورت private تعریف کنید.

کلاس Printer:

این کلاس دارای `property` های زیر می‌باشد.

```
1 private int id;
2 private int amountOfInk;
```

هر پرینتری آیدی دارد تا بتوان آن را از بقیه پرینتر ها موقع کار کردن متمایز کرد. پراپرتی دوم مقدار جوهر مانده پرینتر را مشخص میکند که در ابتدا برابر با 200گرم است (مقدار جوهر هر پرینتر در بازه 0 تا 200 گرم است) .

سازنده‌ی این کلاس به شکل زیر است.

```
1 Printer(int id);
```

این کلاس دارای متدهای زیر است:

```
boolean print(Order order);
```

اگر عملیات چاپ موفقیت آمیز باشد و پرینتر اگر به مقدار سفارش ثبت شده برای چاپ جوهر داشته باشد، تابع مقدار true برمیگرداند در غیر این صورت false.

```
void setAmountOfInk(int amountOfInk);
```

مقدار جوهر پرینتر را از 0 تا 200 گرم تنظیم میکند.

```
int getAmountOfInk(int amountOfInk);
```

مقدار جوهر موجود در پرینتر را برمیگرداند.

کلاس Order:

این کلاس دارای `property` های زیر می‌باشد.

```
private int pages;
private int id;
```

هر سفارشی آیدی دارد و باید تعداد صفحات برای چاپ را مشخص کند به ازای هر صفحه 10 گرم جوهر نیاز است.

سازنده‌ی این کلاس به شکل زیر است.

```
Order(int id, int pages);
```

کلاس Management:

این کلاس دارای `property` های زیر می‌باشد.

```
private Printer[] printers;
private Order[] orders;
```

این کلاس دارای متدهای زیر است:

```
void addPrinter(Printer printer);
```

پرینتری به لیست پرینتر ها اضافه میکند.

```
void addOrder(Order order);
```

سفارش را به سفارشات موجود اضافه میکند.(سفارشی اگر توسط پرینتر انجام و چاپ شد باید از این لیست خط بخورد)

```
void reFill(Printer printer);
```

این متد مقدار جوهر پرینتر ورودی را پر میکند(برابر با 200 قرار میدهد).

```
boolean assignOrder();
```

این متد نقش مدیریت چاپخانه را دارد و هنگامی که فراخوانی شده اولین سفارشی که ثبت شده را با پرینتری که در بین پرینتر های موجود چاپخانه بیشترین جوهر را دارد چاپ میکند و اگر این عملیات با موفقیت انجام شد، true برمیگرداند در غیر این صورت false برمیگرداند.

آن‌چه باید آپلود کنید

ساختار فایل `zip` ارسالی باید به صورت زیر باشد:

```
<zip_file_name.zip>
    ├── Printer.java
    ├── Order.java
    └── Management.java
```

Risk

- سطح سوال : سخت
- طراحان : آریا شاکو و برنا ماهرانی

ابتدا پروژه اولیه را از این لینک دانلود کنید.

ریسک یک بازی معروف است که برای آشنایی بیشتر از ان میتوانید به لینک مراجعه کنید

https://en.wikipedia.org/wiki/Risk_(game)

قرار است اینجا این بازی را پیاده سازی کنیم.

کلاس Country

این کلاس دو فیلد دارد

```java
public class Country{
    String owner;
    int troops;
    public String getOwner{
        \\TODO
    }
    public int getTroops{
        \\TODO
    }
    public void setOwner(String owner){
        \\TODO
    }
    public void setTroops(int troops){
        \\TODO
    }
}
```

کلاس Player

این کلاس 5 فیلد دارد ستر و گتر های مناسب برای این فیلد هارا پیاده سازی کنید. بولین strike مشخص میکند که پلیر در حالت attack هست یا defend برای ستر گتر ها اولین اسم متغیر را بزرگ بذارید به طور مثال setStrike

```
public class Player{
    boolean strike;
    int dice;
    boolean tag;
    String name;
    Country[] countries;
}
```

کلاس Risk

کلاس ریسک این کلاس مهم ترین کلاس این سوال است در این کلاس 1 تابع داریم که منطق بازی را پیاده سازی میکنیم. و یک فیلد 2 تایی از 2 تا بازیکن داریم تا با انها بتوانیم بازی کنیم .

منطق بازی اینطور است که پس از اعلام حمله و دفاع باید بازی شروع شود که این بولین در کلاس پلیر وجود دارد. اگر کشوری به کشور دیگر حمله میکند باید تعداد نیرو ها از یکدیگر کم شود و اگر اختلاف تعداد کشور های بازیکنی از بازیکن دیگر بزرگتر مساوی 2 باشد بازی تمام می‌شود.

تاس را در یک حلقه مقدار دهی کنید تا عددی از 0 تا 2 باشد و طبق ان کشور یک بازیکن رندوم انتخاب میشود. برای هر بازیکن 3 تا کشور در نظر بگیرید و به ترتیب به هر کشور 5 10 15 تا نیرو بدهید.

حالا که وارد حلقه بازی شدیم با توجه به کشور انتخابی باید تعداد نیرو ها از هم کم شود و توجه داریم که اگر کشوری از یک بازیکن نیروی بیشتری از کشور بازیکن دیگر دارد کشور با نیروی ماکسیمم آن یکی کشور را تصاحب میکند و به بازیکن داده می‌شود.

در همین حلقه بازی بولین strike باید برای بازیکن اول به attack ست شود و بازیکن دوم به defense ست شود و در دور بعدی تغییر میکند و بازیکن دوم حمله می‌کند و بازیکن اول دفاع می‌کند.

و در نهایت تابعی تعریف کنید که تعداد سربازان باقی مانده بازیکنان رو می‌شمارد اگر بازیکنی در انتهای بازی بیشتر مساوی 20 سرباز داشت بولین تگ برای آن بازیکن true می‌شود. پس از اتمام بازی یک بولین quit فعال شود.

```java
public class Risk{
    Player[] players = new Player[2];
    public void game(){
        // TODO
    }
}
```

آنچه باید آپلود کنید

یک فایل zip حاوی 3 کلاس Player , Country , Risk

فروشگاه عباس(امتیازی)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- طراح: مهرسا سمیع‌زاده

عباس که متوجه شد از دانشگاه چیزی عایدش نمی‌شود، تصمیم گرفت به تجارت و فروشندگی روی بیاورد. او که در ابتدای حرفه‌ی خویش است نیاز دارد که برای تحلیل وضعیت فروشگاهش و نظارت بر عملکرد صحیح آن، برنامه‌ای طراحی کند تا در کمترین زمان پاسخ‌گوی نیازهایش باشد.

ابتدا پروژه اولیه را این لینک دانلود کنید.

توجه داشته باشید که می‌توانید پراپرتی و متدهای مورد نیاز خودرا اضافه کنید.

تمامی پراپرتی های هر کلاس را به صورت private تعریف کنید.

و برای استفاده و یا تغییر هرکدام از پراپرتی ها نیز از توابع getter و setter استفاده کنید.

کلاس Customer :

این کلاس دارای `property` زیر می‌باشد.

```
1 private String customerName;
2 private int phoneNumber;
3 private Product[] savedProducts;
4 private Product[] purchasedProducts;
```

سازنده‌ی این کلاس به شکل زیر است.

```
1 Customer(String customerName, int phoneNumber);
```

این کلاس دارای متد های زیر است:

```
boolean saveProduct(Product product);
```

که با اجرای این متد محصول مورد نظر مشتری، ذخیره می‌شود. در نهایت با موفقیت‌آمیز بودن این عملیات مقدار true برگردانده می‌شود.

```
boolean purchaseProduct(Product product);
```

با اجرای این متد مشتری می‌تواند محصول مورد نظر خود را خریداری کند ، و در این متد نیز در صورت موفقیت آمیز بودن true برگردانده می‌شود.

```
Product[] discountOffers();
```

در این متد محصولاتی که برای کاستومر شامل تخفیف می‌شوند را برمیگرداند. که این محصولاتی، آنهایی اند که هم مشتری آنهارا را سیو کرده و هم در لیست محصولات پیشنهادی وی (با ماکسیمم تعداد 10) قرار داردند.

```
int calculateProductPrice(Product product, Store store);
```

قیمت نهایی محصول را برای کاستومر برمی‌گرداند. باید توجه داشت که آیا محصول ورودی، شامل تخفیف برای کاستومر می‌شود. توجه داشته‌باشید، میزان تخفیف اعمالی 10 درصد است.

کلاس Product:

این کلاس دارای `property` زیر می‌باشد.

```
private String productName;
private int price;
private int stockRemaining;
private int numSaved;
private int numSold;
```

سازنده‌ی این کلاس به شکل زیر است.

```
Product(String productName, int price, int initialStock);
```

کلاس Store :

این کلاس دارای property زیر می‌باشد.

```
private int productCount ;
private int customerCount ;
private Product[] productList;
private Customer[] customerList;
```

سازنده‌ی این کلاس به شکل زیر است.

```
Store();
```

این کلاس دارای متد های زیر است.

```
void addProduct(Product product);
```

```
void addCustomer(Customer customer);
```

```
int findProductPrice(String productName);
```

این متد نام محصول را می‌گیرد، و قیمت آن را برمی‌گرداند.

```
Product mostPopularProduct();
```

محبوب‌ترین محصول را برمی‌گرداند. اولویت محبوبیت ابتدا با میزان فروش آن محصول، و سپس میزان ذخیره آن محصول می‌باشد.

```
Product[] mostBought(int n);
```

آرایه‌ای مرتب شده از تاپ n محصول پرفروش را برمی‌گرداند.

```
Product[] mostSavedProducts(int n);
```

آرایه‌ای مرتب شده از n محصول با بیشترین میزان ذخیره را برمی‌گرداند.

```
public Product[] recommendProducts(Customer customer, int maxRec
```

این متد آرایه ای از محصولات پیشنهادی برای مشتری مورد نظر را تا حداکثر عدد ورودی را برمی‌گرداند. محصولات پیشنهادی هر مشتری، بر اساس سابقه خرید او و میزان شباهتش با دیگر مشتریان به دست آورده می‌شود. میزان شباهت دو کاستومر، با تعداد خرید های مشابه آن دو، محاسبه می‌شود.

اولویت انتخاب محصولات پیشنهادی به ترتیب، ابتدا با میزان شباهت سابقه خرید مشتری و سپس با فروش کلی محصول میباشد.

اگر شباهتی بین خرید مشتری مورد نظر ما و دیگر مشتریان نبود آرایه خالی برگردانده می‌شود.

ابتدا محصولاتی که توسط کاستومر های مشابه با کاستومر ورودی، خریداری شده را پیدا می‌کنیم.سپس بین این محصولات، اولویت بندی می‌کنیم و آنهارا مرتب می‌کنیم.

توجه داشته باشید محصولاتی که از سابقه‌ی خرید مشتری، با شباهت بالاتر با مشتری ورودی ما، به دست می‌آیند دارای اولویت بالاتری می‌باشند. و بین محصولاتی که در این مورد دارای شرایط یکسان میباشند، محصولاتی دارای اولویت هستند که به طور کلی و بین تمامی مشتریان فروش بیشتری داشته باشند.

مثال

پروژه را به گونه ای پیاده سازی کنید که با اجرای **Main** زیر:

```
public class Main {
public static void main(String[] args) {
// Create store
Store store = new Store();
// Create products
Product p1 = new Product("Laptop", 1000, 10);
Product p2 = new Product("Phone", 500, 20);
Product p3 = new Product("Tablet", 300, 15);
// Add products to the store
store.addProduct(p1);
```

```
store.addProduct(p2);
store.addProduct(p3);
// Create customers
Customer c1 = new Customer("A", 123456);
Customer c2 = new Customer("B", 789012);
// Customers buy products
c1.purchaseProduct(p2);
c2.purchaseProduct(p2);
c2.purchaseProduct(p1);
store.addCustomer(c1);
store.addCustomer(c2);
c2.saveProduct(p2);
c1.saveProduct(p2);
c2.saveProduct(p3);
System.out.println(store.mostSavedProducts(2)[1].getProductName(
Product[] recommendations = store.recommendProducts(c1,2);
System.out.println("Recommended Products for " + c1.getCustomerN
for (int i = 0; i < recommendations.length; i++)
System.out.println(recommendations[i].getProductName());
}
System.out.println("Most Popular Product:\n"+store.mostPopularPr
}
```

خروجی به شکل زیر باشد:

```
Tablet
Recommended Products for A:
Laptop
Most Popular Product:
Phone
```

آنچه باید آپلود کنید

یک فایل Zip آپلود کنید که شامل سه فایل تکمیل شده Customer.java , Product.java, Store.java باشد.