۲ تردپول

همانطور که می‌دانید، اجرای کارهای همزمان در جاوا با مفهومی به نام **ترد** انجام می‌شود. مشکلی که در تردها وجود دارد این است که، ساختن و نابود کردن یک ترد از سیستم حافظه و زمان نسبتا زیادی می‌برد، به همین علت هنگامی که می‌خواهیم تعدادی کار را داخل یک صف قرار دهیم و آن‌ها را به ترتیب به کمک تعداد مشخصی ترد اجرا کنیم، از سرویسی به نام ThreadPool استفاده می‌کنیم. برای حل این سوال، شما باید این سرویس را در جاوا پیاده سازی کنید. برای اینکه بیشتر با این مفهوم آشنا شوید، به این لینک مراجعه کنید.

بانکداری هوشمند

در این سوال شما باید سامانه ATM (یا خودپرداز) یک بانک را طراحی کنید. برای شروع، پروژه اولیه را که به پست مربوط به این تمرین پیوست شده است دانلود کنید. ساختار فایل های این پروژه به صورت زیر هستند:

```
src
├── Bank
│   ├── ATM.java
│   ├── Bank.java
│   ├── Card.java
│   └── Handler.java
├── Exceptions
│   ├── InvalidCardNoException.java
│   ├── InvalidCashAmountException.java
│   ├── NoCardInsertedException.java
│   ├── NoFreeAtmException.java
│   ├── NotEnoughBalanceException.java
│   └── WrongPasswordException.java
├── Results
│   ├── AccountCreatedResult.java
│   ├── BalanceMovedResult.java
│   ├── BalanceReturnedResult.java
│   ├── CardInsertedResult.java
│   ├── CardRemovedResult.java
│   ├── CashDepositedResult.java
│   ├── PasswordChangedResult.java
│   └── Result.java
└── Tasks
    ├── ChangePasswordTask.java
    ├── CreateAccountTask.java
    ├── DepositCashTask.java
    ├── GetBalanceTask.java
    ├── InsertCardTask.java
    ├── MoveBalanceTask.java
    ├── RemoveCardTask.java
    └── Task.java
```

روند کلی کار به این صورت است که در ابتدای اجرای برنامه یک instance از کلاس بانک ایجاد می‌شود. این بانک تعداد مشخصی خودپرداز دارد که هنگام ساختن بانک مشخص می‌شود. هر خودپرداز در هرلحظه می‌تواند حداکثر به یک کاربر خدمات ارائه کند، اما خودپردازها می‌توانند به صورت همزمان با هم فعالیت کنند؛ یعنی اگر پنج خودپرداز داشته باشیم، می‌توانیم هم‌زمان به پنج کاربر خدمات ارائه کنیم. کارهایی که هر خودپرداز می‌تواند انجام دهد، به عنوان یک دسته کلاس در پکیج Tasks تعریف شده‌اند و جواب‌هایی هم که هر Task برمی‌گرداند در پکیج Results تعریف شده. توضیحات بیشتر در مورد کلاس‌ها را در ادامه می‌بینید.

چند نکته

۱. در این سوال ما هیچ ورودی یا خروجی نداریم! (فکر کنم خبر خوبی براتون باشه :D)

۲. هرکدام از کلاس‌های Task یک تابع به اسم run دارند که هیچ ورودی دریافت نمی‌کند و تعدادی عملیات انجام می‌دهد. اگر انجام این عملیات با خطا مواجه شود، متناسب با ارور پیش آمده، یک Exception پرتاب می‌کند. پرت کردن این استثنا نباید منجر به کشته شدن (اصطلاحا Kill شدن) ترد شود. در نتیجه هندل کردن و return کردن جواب Task یا استثناء پرت شده به عهده شماست.

۳. جوابی که آپلود می‌کنید نباید تابع main داشته باشد. برای تست کردن کد خود می‌توانید یک تابع main بنویسید اما موقع آپلود باید این تابع را حذف کنید.

کلاس Bank

همانطور که بالاتر بیان شد، در کل فرایند اجرای برنامه فقط یک آبجکت از این کلاس ایجاد می‌شود. کانستراکتور این کلاس تعداد خودپردازهای این بانک را در ورودی دریافت می‌کند و به همان تعداد ATM ایجاد می‌کند و تا قبل از کال شدن تابع runATM، این کلاس کار دیگری انجام نمی‌دهد. برای اینکه خودپردازهای داخل این بانک قابلیت اجرای همزمان داشته باشند، باید موقع ساخته شدن این کلاس، یک ThreadPool با اندازه مناسب ایجاد کنید و تسک‌هایی که در ادامه داده می‌شوند را به این کلاس بدهید تا هرکدام در تردی جداگانه اجرا شوند. توابع داخل کلاس Bank به شرح زیر هستند:

- Bank(int ATMCount) : کانستراکتور کلاس که تعداد خودپردازهای این بانک را ورودی دریافت می‌کند. تضمین می‌شود که در تست کیس‌ها این عدد، یک عدد مثبت است.
- runATM(ArrayList<Task> tasks, Handler handler) : این تابع تسک‌ها را دریافت می‌کند، اگر یک ATM خالی (تردی که مشغول اجرای دستوری نباشد) وجود داشت، دستورات را توسط آن ترد اجرا می‌کند. در غیر این صورت، دستور وارد صف تردپول می‌شود و تا زمان اجرا شدن در آن صف باقی می‌ماند. پس از پایان یافتن کار ترد، باید تابع ()done از آبجکت Handler داده شده کال شود؛ با صدا کردن این تابع کد شما اعلام می‌کند که نتیجه اجرای این Task آماده می‌باشد.

نکته مهم: این تابع نباید به صورت blocking پیاده سازی شود، یعنی برنامه تا قبل از اجرای کامل ترد و آماده شدن نتیجه، در این تابع باقی نمی‌ماند.

کلاس ATM

این کلاس معادل مدل خودپرداز می‌باشد. همواره یک instance از این کلاس باید به Task در حال اجرا داده شود.

کلاس Card

این کلاس نشان دهنده یک کارت بانکی می‌باشد. هر کارت یک شماره کارت، یک پسورد و یک موجودی دارد.

پکیج Tasks

کلاس های داخل پکیج Tasks به صورت کامل پیاده سازی شده‌اند و شما نباید در آنها تغییری ایجاد کنید. هر کدام از این کلاس‌ها یک کانستراکتور دارند که فیلدهای مورد نیاز را دریافت می‌کند. یک تابع run هم در این کلاس‌ها تعریف شده که هیچ ورودی دریافت نمی‌کند و خروجی آن یک آبجکت از کلاس Result می‌باشد. اگر در حین پردازش و اجرای این تابع خطایی بوجود بیاید، یک Exception توسط این کلاس پرتاب می‌شود. شما باید داخل تردی که تعریف می‌کنید، تابع run تسک های داده شده را کال کنید، اکسپشن احتمالی پرتاب شده را catch کنید و یا جواب ریترن شده را بگیرید و داخل یک ArrayList بریزید و در نهایت ArrayList تولید شده را ریترن کنید. توجه کنید که این ArrayList باید یک ArrayList<Object> و متشکل از Result ها و Exception های تولید شده باشد.

کاری که شما باید انجام دهید

ابتدا باید پروژه را به طور کامل دانلود کنید. وظیفه اصلی که بر عهده شماست، پیاده کردن کلاس ThreadPool می‌باشد. این کلاس باید در کانستراکتور خود یک عدد به عنوان ورودی دریافت کند، و به همان تعداد ترد بسازد. همچنین باید تابعی داشته باشد که یک مجموعه دستور دریافت کند و آن‌ها را داخل یک صف قرار دهد و به محض اینکه کار یک ترد به پایان رسید، دستور بعدی را از صف برداشته و اجرا کند.

نکته: ترد ها باید همزمان اجرا شوند، اما ترتیب خروجی‌ها باید به همان ترتیب ورودی‌ها باشد. همچنین صفی که تعریف می‌کنید باید سینکرونایزد (Synchronized) باشد تا وقتی که همزمان چند ترد از آن داده می‌خوانند، مشکلی بوجود نیاید.

آنچه باید آپلود کنید

تکمیل شده‌ی همه فایل‌هایی که دانلود کرده‌اید + کلاس‌هایی که ایجاد کرده‌اید را آپلود کنید. دقت کنید که تابع main نباید در برنامه وجود داشته باشد.

راهنمایی

صرفا جهت روشن‌تر شدن اینکه چگونه باید کلاس ThreadPool را پیاده سازی کنید: می‌توانید یک property از جنس Queue در این کلاس تعریف کنید که مجموعه تسک‌های داده شده را نگهداری کند:

```
Queue<ArrayList<Task>> queue = new Queue<>();
```

موقع ساخته شدن هر ترد هم، می‌توانید در تابع run که برای آن ترد تعریف می‌کنید، یک حلقه بی‌نهایت قرار دهید که تا زمانی که دیتا در این صف وجود دارد، اطلاعات را بخواند و آن‌ها را اجرا کند:

```
new Thread(new Runnable() {
    @Override
  public void run() {
        while (true) {
            //fetch tasks and execute...
  }
    }
}).start();
```