درس برنامه‌سازی پیشرفته

تمرین سوم بخش دوم

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۹۹ـ۹۸

مبحث:
شبکه و ترد

مهلت ارسال:
۲۱ خرداد
ساعت ۲۳:۵۹

ویراستار فنی:
صابر ظفرپور و محمّد فراهانی

به موارد زیر توجه کنید:

* فایل برنامه‌ی خود با پسوند zip. را در بخش مربوط به سوال بارگذاری کنید.
* پس از ارسال فایل مربوط به هر سوال، سامانه‌ی کوئرا به‌صورت لحظه‌ای برنامه‌ی شما را داوری کرده و نمره‌ی آن سوال را به شما اعلام می‌کند که در صورت کم بودن نمره‌تان، می‌توانید آن را تصحیح کرده و دوباره ارسال کنید.
* هم‌فکری و هم‌کاری در پاسخ به تمرینات اشکالی ندارد و حتی توصیه نیز می‌شود؛ ولی پاسخ ارسالی شما باید حتما توسط خود شما نوشته شده‌باشد. در صورت هم‌فکری در مورد یک سوال، نام افراد دیگر را به‌صورت کامنت در ابتدای کد هر سوال بنویسید. این نکته رو در نظر بگیرید که هم‌فکری تنها مربوط به بخش ایده سوال هست نه پیاده‌سازی آن و در صورت محرز شدن تقلب برای فرد خاطی بدون مسامحه منفی نمره تمرین منظور می‌گردد.
* شما می‌توانید تمامی سوالات و ابهامات خود را در سایت کوئرا در بخش مشخص‌شده برای این تمرین بپرسید.
* به‌ازای هر روز تاخیر در ارسال پاسخ هر سوال، ۳۰ درصد از نمره‌ی کسب‌شده‌ی شما در آن سوال کم می‌شود. به عنوان مثال اگر پاسخ یک سوال را با دو روز تاخیر ارسال کنید، فقط ۴۰ درصد از نمره‌ای که برای آن سوال گرفته‌اید برای شما لحاظ خواهد شد.
* در کل شما می‌توانید سه روز تاخیر بدون کسر نمره داشته باشد.
* مهلت ارسال تمرین تا ساعت ۲۳:۵۹ روز ۲۱ خرداد ۱۳۹۹ است.

۱ شبکه شتاب

در این تمرین قرار است یک شبکه‌ی بانکداری را پیاده سازی کنید:
این شبکه متشکل از سه بخش است:

۱. یک سرور DNS (برای اطلاع بیشتر درباره‌ی DNS ها می‌توانید از این لینک استفاده کنید.)

۲. تعدادی بانک که هر کدام یک سرور مختص به خود دارند.

۳. تعدادی عابربانک که هر کدام با سرور بانک خود در ارتباط هستند.

- **سرور بانک:** در هر بانک تعدادی حساب وجود دارد. هر حساب توسط شماره حساب آن، که یک عدد یکتاست، شناخته می‌شود . همچنین هر حساب مقداری موجودی دارد. تنها عملیاتی که بر روی یک حساب انجام می‌شود، واریز یا برداشت وجه است.
  در هر عملیات واریز، اگر شماره حساب مقصد موجود نبود، ابتدا یک حساب جدید با آن شماره حساب و موجودی صفر ساخته شده،سپس واریز صورت می‌گیرد. همچنین هر عملیات برداشت در صورتی انجام می‌گیرد که آن شماره حساب موجود باشد و مقدار برداشتی حداکثر برابر موجودی حساب باشد.در غیر این صورت این عملیات نادیده گرفته می‌شود. در ابتدای کار هیچ حسابی در بانک ها وجود ندارد و حساب ها تنها توسط عملیات واریز ساخته می‌شوند. همچنین حساب‌ها هیچ گاه از بین نمی‌روند.
  زمانی که یک بانک جدید ایجاد می‌شود، ابتدا شماره‌ی port خود را به همراه نام بانک برای سرور DNS می‌فرستد. سپس منتظر اتصال عابر بانک ها می‌شود.
- **عابر بانک:** زمانی که یک عابر بانک جدید ایجاد می‌شود، ابتدا نام بانک خود را برای سرور DNS می‌فرستد. سپس سرور DNS در پاسخ، شماره‌ی port سرور آن بانک را می‌فرستد.در نهایت عابر بانک با استفاده از آن port به سرور بانک خود متصل می‌شود و از آن پس تراکنش‌ها را برای سرور بانک خود ارسال می‌کند تا بررسی و اعمال شوند.
- **سرور DNS :** کار آن دادن آدرس سرورها به عابربانک ها است. خود این سرور نیز شماره‌ی port ثابتی دارد که از ابتدا مشخص است و هم بانک‌ها و هم عابر بانک ها از آن اطلاع دارند و با استفاده از آن با سرور DNS ارتباط برقرار می‌کنند.

پیاده سازی

برای این منظور کد خام و تست های این تمرین در اختیار شما قرار می‌گیرد که شما می‌بایست کد خام را به گونه ای تکمیل کنید که تست ها را پاس کنید.

- **کلاس DNS :**

```
import java.io.IOException;
import java.util.HashMap;

public class DNS {
    private final HashMap<String, Integer> bankPorts = new
HashMap<>();

    public DNS(int dnsPort) throws IOException {

    }

    public int getBankServerPort(String bankName) {
        return -1;
    }
}
```

این کلاس سرور DNS را پیاده سازی می‌کند.

```
public DNS(int dnsPort)
```

کانستراکتور کلاس DNS است. dnsPort هم شماره‌ی port سرور است.
بعد از ایجاد تنها شی این کلاس ، باید سرور DNS فعال شده و بر روی پورت dnsPort منتظر اتصال بانک‌ها وعابر بانک‌ها شود.

```
public int getBankServerPort(String bankName)
```

باید port سرور بانک با اسم bankName را برگرداند.
این تابع صرفا از قسمت Test Unit صدا زده خواهد شد و هیچ کدام از کلاس ها و توابعی که شما پیاده سازی می کنید حق صدا زدن این تابع را ندارند.

- **کلاس BankServer :**

```
import java.io.IOException;
import java.util.HashMap;

public class BankServer {
    private final HashMap<Integer, Integer> accounts = new
HashMap<>();
    public BankServer(String bankName, int dnsPort) throws IOEx-
ception {

    }

    public int getBalance(int userId) {
        return 0;
    }

    public int getNumberOfConnectedClients() {
        return 0;
    }
}
```

این کلاس، بانک ها را پیاده سازی می‌کند.

```
public BankServer(String bankName, int dnsPort)
```

کانستراکتور بانک است. برای ایجاد یک بانک جدید با نام bankName و port با مقدار dnsPort این تابع صدا می‌شود. هر بانک تنها یک سرور دارد و بعد از صدا شدن کانستراکتورش، باید با استفاده از dnsPort به سرور DNS متصل شود و شماره‌ی port خود را به همراه اسم بانک اعلام کند. پس از آن بانک ساخته شده منتظر اتصال عابر بانک ها می‌ماند و بعد از اتصال هر عابر بانک، تراکنش های آن را انجام می‌دهد.

```
public int getBalance(int userId)
```

موجودی حساب با شماره‌ی userId را برمی‌گرداند.

این تابع صرفا از قسمت Test Unit صدا زده خواهد شد و هیچ کدام از کلاس ها و توابعی که شما پیاده سازی می کنید حق صدا زدن این تابع را ندارند.

```
public int getNumberOfConnectedClients()
```

تعداد عابر بانک های متصل شده یک بانک را برمی گرداند.
این تابع صرفا از قسمت Test Unit صدا زده خواهد شد و هیچ کدام از کلاس ها و توابعی که شما پیاده سازی می کنید حق صدا زدن این تابع را ندارند.

- **کلاس BankClient :**

```
import java.io.File;
import java.io.IOException;

public class BankClient {
    public static final String PATH = "./tests/";

    public BankClient(String bankName, int dnsServerPort) throws
IOException {

    }

    public void sendTransaction(int userId, int amount) {

    }

    public void sendAllTransactions(String fileName, int timeBetween-
Transactions) {

    }
}
```

این کلاس یک عابر بانک را پیاده سازی می‌کند.

```
public BankClient(String bankName, int dnsPort)
```

کانستراکتور آن است. برای ایجاد یک عابر بانک جدید برای بانکی با نام

bankName یک شی از این کلاس ساخته می شود. هر بانک می‌تواند چند عابر بانک داشته باشد. بعد از صدا شدن کانستراکتور، عابر بانک باید با استفاده از dnsPort به سرور DNS متصل شود و بعد از ارسال اسم بانک، شماره‌ی port بانک خود را دریافت کند سپس به سرور بانک خود متصل شود.

```
public void sendTransaction(int userId, int amount)
```

توسط این تابع، یک تراکنش برای شماره حساب userId به سرور فرستاده می شود.
اگر amount نامنفی بود، مقدار amount واریز می‌شود.
در غیر این صورت مقدار lamountl واحد برداشت می‌شود.
دقت کنید که پردازش این عملیات باید در سرور اتفاق بیفتد و کلاینت صرفا درخواست را می فرستد.

```
public void sendAllTransactions(String fileName, int timeBetween-
Transactions)
```

این تابع باید از فایل fileName لیست تراکنش‌ها را بخواند و آن ها را به ترتیب اجرا کند. بین اجرای هر دو تراکنش متوالی هم باید حداقل timeBe- tweenTransactions میلی‌ثانیه فاصله باشد. در هر خط از فایلfileName ،مشخصات یک تراکنش آمده است.
هر تراکنش با دو عدد نمایش داده شده است. عدد اول شماره‌ی حساب و عدد دوم مقدار تراکنش است. همانند بالا علامت عدد دوم نمایانگر نوع تراکنش است.

قوانین

۱. توجه کنید تمام تراکنش ها در سرور بانک صورت می‌گیرند و عابر بانک فقط اطلاعات تراکنش را برای سرور بانک می‌فرستد.(هیچ قسمت از پردازش نباید در کلاینت اتفاق بیفتد.)

۲. دقت کنید که برای آنکه بتوانید تست ها را پاس کنید لازم است بعضی توابع (یا constructor ها) را به صورت Blocking پیاده سازی کنید و بعضی دیگر را به صورت Non-Blocking.

- تابع Blocking : تا زمانی که عملیات خواسته شده به صورت کامل انجام نشده است از تابع خارج نمی شود و در نتیجه برنامه اصلی(caller) متوقف می ماند.
- تابع non-blocking : عملیات خواسته شده را در یک ترد دیگر آغاز می کند و از تابع خارج شده و برنامه اصلی(caller) به اجرایش ادامه می دهد.

۳. لازم است که در ابتدای توابعی که پیاده سازی میکنید نوع آن تابع (Blocking / Non-Blocking) را ذکر کنید.

۴. در صورتی که کلاس های شما علاوه بر بستر شبکه از راه های دیگری با هم ارتباط داشته باشند (مثلا ارتباط از طریق صدا زدن تابع های همدیگر و یا ارتباط از طریق اشتراک گذاری فایل یا ...) نمره تان ۰ خواهد شد.

۵. تنها می توانید توابع private به کلاس هایتان اضافه کنید.

۶. پاسخ ارسالی شما باید تنها حاوی ۳ فایل BankClient.java و BankServer.java و DNS.java باشد. دقت کنید که مقدار PATH را به درستی انتخاب کرده باشید.

فایل خام تمرین