به نام خدا

درس برنامه‌سازی پیشرفته

بانک

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۹۹ـ۹۸

اساتید:

مهدی مصطفی‌زاده، ایمان عیسی‌زاده، امیر ملک‌زاده، علی چکاه

نگارش و تهیه محتوا:

محسن دهقان کار، علیرضا شاطری و صابر ظفرپور

تنظیم داک:

امیرمهدی نامجو

فهرست

اجرای بانک

برای اجرای فایلی که در اختیارتان قرار می‌دهیم در ترمینال یا (در ویندوز cmd) این دستور را وارد کنید:

```
java -jar bank.jar "port" "debug"
```

Bank.jar اسم فایل با پسوند jar است که در اختیارتان قرار داده می‌شود و در قسمت port پورت مورد نظر و در قسمت debug عدد ۰ یا ۱ بنویسید. (۰ یعنی سرور در مود عادی اجرا شود و ۱ یعنی سرور با مود دیباگ اجرا شود و تفاوت این دو مود در پایین توضیح داده شده است.)

- برای اجرای سرور باید ورژن جاوا شما حداقل ۱۱ باشد. برای اینکه ورژن جاوا خود را ببینید دستور زیر را در ترمینال یا cmd ویندوز وارد کنید:

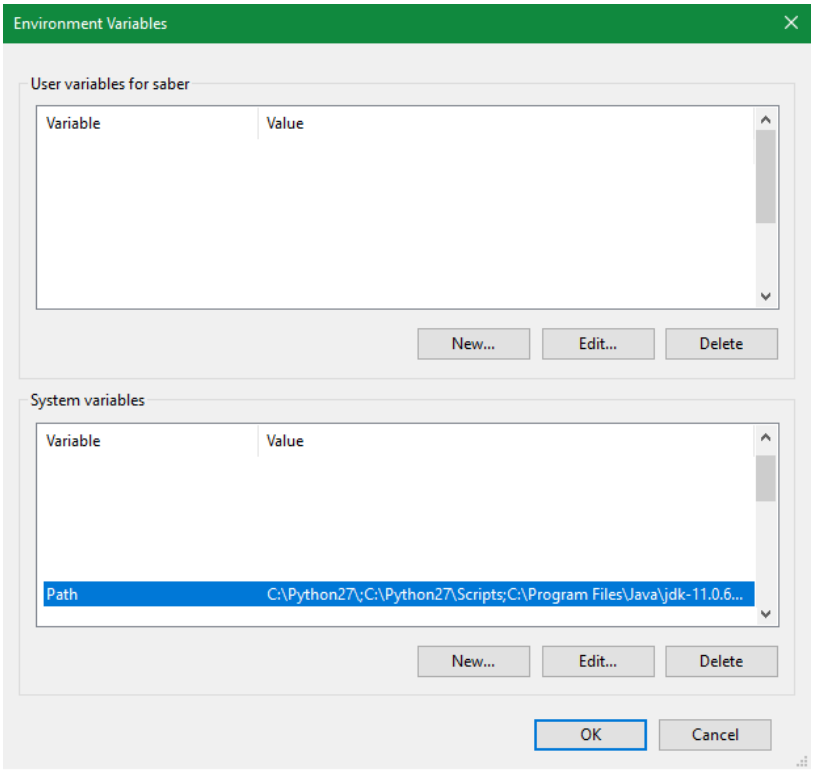
```
java --version
```

⋆⋆⋆ در صورتی که نسخه جاوا شما قدیمی است:

اگر **سیستم عامل ویندوز** دارید ابتدا از این لینک فایل نصبی جاوا را دانلود و نصب کنید و پس از نصب وارد تنظیمات Edit the system environment variables شوید و در قسمت environment variables طبق تصویر صفحه بعد path را بیابید و ادیت کنید. حال در بین مسیرهای موجود سعی کنید مسیر قبلی java را بیابید و آن را با مسیر پوشه bin جاوای نسخه جدید خود عوض کنید.

(C:\Program Files\Java\jdk-11.0.6\bin)

مشکل نسخه جاوای شما با این روش حل می‌شود و شما می‌توانید سرور بانک خود را به سادگی با همان دستور قبلی اجرا کنید.



اما اگر سیستم عامل لینوکس دارید با توجه به توزیع مورد استفاده می‌توانید از گوگل بهره مند شوید:) (این روش برای اوبونتو تست شده است.)

نکات

- پس از اجرای سرور، در صورت نبود مشکل، آی پی و پورتی که سرور روی آن اجرا شده است در کنسول چاپ می‌شود. (اگر سرور و کلاینت روی یک کامپیوتر اجرا می‌شوند، آی پی 127.0.0.1 قابل استفاده خواهد بود.)
- در صورت انتخاب مود دیباگ برای اجرای سرور، دستوری که هر کلاینت به سرور می‌فرستد چاپ می‌شود و همچنین هنگام قطع شدن ارتباط کلاینت از سرور تعداد کلاینت‌های آنلاین چاپ می‌شود.
- همچنین در مود دیباگ، پس از وصل شدن هر کلاینت به سرور یک پیام از سرور به کلاینت به شکل زیر فرستاده می‌شود. این پیام نشان دهنده این است که کلاینت با موفقیت به سرور متصل شد (به هر کلاینت متصل به بانک یک آی دی نسبت داده می‌شود که در فرایند دیباگ کردن کاربرد خواهد داشت (طبق آن چه در بالا گفته شد))

```
hello "clientID"
```

- ممکن است هنگام اجرای سرور خطاهای دیگری نظیر خطای مربوط به پورت و وصل شدن به کلاینت و ... نیز چاپ شوند.

API

API یا Application Programming Interface در واقع همان واسطی (interface) است که به وسیله ی آن با سرور بانک ارتباط برقرار می‌کنید. در اینجا استفاده از API به وسیله ارسال یک رشته صورت می‌گیرد و شما می‌توانید با فرمتی که در ادامه داک توضیح داده می‌شود از آن استفاده و نیازهای مالی فروشگاه خود را برطرف کنید. همچنین یک کد آماده در کنار فایل jar قرار داده شده است تا شما به صورت عملی نیز با استفاده از API بانک داده شده آشنا شوید.(پیشنهاد می‌شود از متدهای writeUTF و readUTF برای ارسال و دریافت دستورات استفاده کنید)

ساخت اکانت

```
create_account "firstname" "lastname" "username" "password"
 "repeat_pasword"
```

- این متد برای ساختن حساب بانکی است.
- ورودی اول و دوم اسم کاربر هستند.
- ورودی سوم نام کاربری برای حساب است.
- ورودی چهارم پسورد اکانت است.
- ورودی آخر تکرار پسورد است.
- خروجی این متد یک عدد به عنوان شماره حساب است.
- یک مثال

```
create_account Bob Bobian ImBob bobword bobword
```

- خطاها:
  - ”passwords do not match” = پسورد و تکرار آن هم خوانی ندارند.
  - ”username is not available” = این نام کاربری قبلا استفاده شده است.

دریافت توکن

```
get_token "username" "password"
```

- این متد برای دریافت توکن از سرور استفاده می‌شود. توکن دریافتی تا ۱ ساعت اعتبار دارد و پس از آن منقضی می‌شود، در نتیجه نیاز است که دوباره این متد صدا زده شود تا توکن تازه دریافت گردد.
- ورودی اول نام کاربری حساب بانکی است.
- ورودی دوم پسورد حساب بانکی است.

- خروجی این متد توکن است که بصورت یک رشته داده می‌شود.
- خطاها:
  - ”invalid username or password”

ایجاد فیش بانکی

```
create_receipt "token" "receipt_type" "money" "sourceID" "destID"
 "description"
```

- این متد برای ساختن فیش بانکی است. برای هر عملیات بانکی ابتدا یک فیش باید ساخته شود، سپس آن فیش را باید پرداخت کرد (عملیات آن را انجام داد).
- ورودی اول آن (یک فاصله بعد از create_receipt) ، توکن دریافتی از سرور است.
- ورودی دوم آن یکی از رشته‌های زیر است:
  - ”deposit” = واریز پول به حساب
  - ”withdraw” = برداشت پول از حساب
  - ”move” = انتقال وجه
- ورودی سوم یک عدد است که مبلغ را تعیین می‌کند.
- ورودی چهارم شماره حساب مبدا است.
  - برای عملیات ”deposit” این عدد ۱ ـ وارد شود.
- ورودی پنجم شماره حساب مقصد است.
  - برای عملیات ”withdraw” این عدد ۱ ـ وارد شود.
- ورودی آخر که اختیاری است، توضیحی در مورد تراکنش است.
- خروجی خروجی این متد یک عدد است که بیانگر ID فیش تولید شده است.

- خطاها:
  - "invalid receipt type" = ورودی دوم نامعتبر است.
  - "invalid money" = عدد وارد شده برای پول صحیح نیست. (عدد مثبت و صحیح باید باشد)
  - "invalid parameters passed" = ورودی‌های داده شده از فرمت بالا پیروی نمی‌کنند.
  - "token is invalid" = توکن نامعتبر است. (اگر مثلا اقدام به ساخت فیش برداشت از حسابی غیر از حساب مربوط به این توکن کنید نیز این خطا داده می‌شود و بطور مشابه برای انتقال وجه)
  - "token expired" = توکن داده شده منقضی شده است.
  - "source account id is invalid"
  - "dest account id is invalid"
  - "equal source and dest account"
  - "invalid account id" = به جای شماره حساب معتبر، ۱ ـ وارد شده است.
  - "your input contains invalid characters" = بخش توضیحات، دارای یکی از کاراکترهای خاص است (مثلا *).

دریافت گزارش تراکنش‌ها

```
get_transactions "token" "type"
```

- این متد برای دریافت تاریخچه‌ی تراکنش‌ها است.
- ورودی اول توکن دریافتی از سرور است.
- ورودی دوم نوع تاریخچه را مشخص می‌کند:
  - "+" برای دریافت تاریخچه تراکنش‌هایی که مقصد آنها حساب شما (حساب مربوط به این توکن) است.

- ”-” برای دریافت تاریخچه‌ی تراکنش‌هایی که مبدا آن‌ها حساب شما است.
- ”*” برای دریافت تاریخچه‌ی همه‌ی تراکنش‌های مربوط به حساب شما.
- این ورودی همچنین می‌تواند یک عدد، بیانگر ID فیش باشد که در آن صورت آن فیش خروجی داده می‌شود.

- خروجی این تابع آرایه‌ای از فیش‌های سریالایز شده (json) خواهد بود که با ”*” از هم جدا شده‌اند. مثلا:

```
{"receiptType":"deposit",
"money":20,
"sourceAccountID":-1,
"destAccountID":10001,
"description":"",
"id":0,
"paid":1}*{"receiptType":"deposit",
"money":2000,
"sourceAccountID":-1,
"destAccountID":10001,
"description":"",
"id":0,
"paid":1}
```

⋆ به فیلدهای یک فیش دقت کنید.

- خطاها
  - ”token is invalid”
  - ”token expired”
  - ”invalid receipt id” = اگر این فیش وجود نداشته باشد یا مربوط به این حساب نباشد.

انجام تراکنش

```
pay "receiptID"
```

- این متد برای انجام عملیات یک فیش است (واریز ، برداشت یا انتقال وجه).
- ورودی این متد ID فیش است.
- خروجی این متد در صورت موفق بودن عملیات ”done successfully” است.
- خطاها:
  - ”invalid receipt id”
  - ”receipt is paid before” = این فیش قبلا پرداخت شده است.
  - ”source account does not have enough money”
  - ”invalid account id” = شماره حساب درج شده در فیش اشتباه است.

دریافت موجودی حساب

```
get_balance "token"
```

- این متد برای گرفتن موجودی حساب است.
- ورودی آن توکن دریافتی از سرور است.
- خروجی آن موجودی حساب است.
- خطاها:
  - ”token is invalid”
  - ”token expired”

قطع ارتباط با سرور

```
exit
```

- ارتباط کلاینت با سرور قطع می‌شود.

خطاهای کلی

- "invalid input" = ورودی وارد شده صحیح نیست.
- "database error" = مشکلی در ارتباط با دیتابیس بانک وجود دارد.