به نام خدا

پروژه میان‌ترم درس برنامه‌نویسی پیشرفته

پاییز ۱۴۰۰

۱. تمامی فایل‌های کد را به همراه فایل های تنظیمات به صورت یک فایل آرشیو zip (rar =! zip) که به قالب زیر نام‌گذاری شده است، بارگذاری نمایید.

AP-Project-FirstName_LastName-StudentNumber.zip

مثال: AP-Project-Bardia_Ardakanian-9831072.zip

۲. در صورت مشاهده هرگونه تقلبی، طبق موارد گفته شده در قوانین درس برخورد خواهد شد.

۳. در صورت وجود هرگونه ابهام می‌توانید از طریق گروه تلگرامی با تدریس‌یاران در ارتباط باشید.

مهلت تحویل فاز اول: تا جمعه ۱۹ آذر ۱۴۰۰ ساعت ۲۳:۵۵ شب

مهلت تحویل فاز دوم: تا جمعه ۲۶ آذر ۱۴۰۰ ساعت ۲۳:۵۵ شب

مهلت تحویل فاز سوم: تا جمعه ۳ دی ۱۴۰۰ ساعت ۲۳:۵۵ شب

پیش‌گفتار

استفاده مناسب از مفاهیم تدریس شده ضروری است. طراحی خوب و منطقی کلاس‌ها و اینترفیس‌ها باید متناسب با اصول برنامه نویسی شی‌گرا باشد. رعایت اصول پنهان‌سازی اطلاعات (information hiding)، سلسله مراتب ارث‌بری جهت استفاده مجدد از کدها (code reusability)، استفاده از چندریختی و سایر نکات تدریس شده الزامی است. سعی کنید پیش از شروع پیاده سازی، تعداد و نام کلاس‌ها، فیلدها و متدهای مورد نظر برنامه را تحلیل کرده و مطابق با تحلیل و طراحی انجام‌شده، برنامه را پیاده سازی کنید.

مستندسازی به کمک Javadoc، کامنت گذاری و رعایت اصول کد نویسی خوانا برای همه کلاس‌های پیاده سازی شده الزامی است.

برنامه را پیش از بارگذاری به خوبی تست و اشکال زدایی کنید ! همه کلاس‌ها و متدهای موجود در برنامه‌ها کاملا مورد بررسی و آزمون قرار بگیرند تا از درستی عملکرد برنامه اطمینان حاصل کنید. حالت‌های مختلف ورودی توسط کاربر باید بررسی شود و در صورت لزوم، پیغام خطای مناسب نمایش داده شود.

در قسمت‌های مختلف پروژه باید خطاهای مختلف بررسی شوند و در قبال آن رفتار مناسبی از برنامه دریافت شود. پس شما باید برای تمامی قسمت‌ها عملیات Exception Handling را برای استثناهای Checked و همچنین در مواقع لزوم برای Unchecked پیاده سازی کنید.

در این پروژه شما می‌بایست کد خود را در گیت نیز قرار دهید. توجه شود صرف قرارگیری و آپلود پروژه در گیت کفایت نمی‌کند و شما باید کامیت‌های مناسب و مستمر داشته باشید.

به عنوان مثال اگر ویژگی جدیدی به کد قبل اضافه کرده‌اید می‌توانید مانند زیر پیام کامیت را مشخص کنید:

[feature] users can follow eachother

یا اگر ایرادی را برطرف کرده‌اید:

[bug] client can't connect to server

یا اگر اصلاحات یا اضافاتی بر ویژگی‌های قبلی داشته‌اید:

[enhancement] speed of tweet posting optimized

روش‌هایی که برای کامیت ذکر شده‌اند کاملا قراردادی بوده و شما از هر روش خوانای دیگری می‌توانید استفاده کنید.

لطفا از ساخت مخزن (Repository) به صورت عمومی (Public) قبل از تمام شدن مهلت ارسال خودداری کنید

پروژه دارای تحویل آنلاین (از طریق اسکایپ) است و تسلط بر آن تاثیر زیادی در نمره شما دارد.

تعریف پروژه

پروژه پیش‌رو شامل 4 بخش اساسی است که به ترتیب مباحث تدریس شده، تعریف شده است. طبیعتا تمام مباحث این پروژه تا این لحظه به شما تدریس نشده است، اما برنامه‌ریزی شده با توجه به برنامه درس، ددلاین‌های فازهای پروژه عملیاتی شوند.

توجه داشته باشید که این پروژه برای تمام ترم است و پروژه پایانترم مجزا تعریف نخواهد شد

در تعریف این پروژه هر رنگ در متن (و نه در عکس‌ها) معنی خاص خود را دارند.

رنگ نارجی تیره به معنی پیاده‌سازی اجباری است.

رنگ سبز به معنی پیاده‌سازی امتیازی است.

رنگ آبی‌کمرنگ به معنی توضیحات بیشتر است.

این پروژه 3 فاز دارد که در ادامه توضیحات هر فاز آمده است.

در صورت وجود هرگونه ابهام در تعریف پروژه می‌توانید با تدریس‌یارها در ارتباط باشید.

شبکه اجتماعی توییتر (Twitter)

در این پروژه ما سعی در پیاده سازی شبکه اجتماعی توییتر به صورت کنسولی (درابتدا) و با رابط کاربری گرافیکی در (انتها) را داریم.

توضیحات اولیه

توییتر (به انگلیسی: Twitter) یک شبکه اجتماعی و سرویس ارائه دهنده میکروبلاگ است که به کاربران اجازه می‌دهد تا ۲۵۶ حرف، پیام متنی را که **توییت** (به انگلیسی: Tweet) نامیده می‌شود، ارسال کنند.

کاربرانی که در توییتر ثبت‌نام کرده‌اند می‌توانند توییت ارسال کنند، به توییت‌ها جواب بدهند، یک توییت را دوباره بازنشر کنند (به انگلیسی: Retweet) و همچنین آنها را لایک کنند.

به‌علاوه موارد ذکر شده کاربرانی که ثبت نام شده‌اند می‌توانند کاربران دیگر را دنبال کنند و توییت‌های آنان را نیز مشاهده کنند، به یکدیگر پیام متنی ارسال کرده و با هم صحبت کنند.

فاز اول: پیاده سازی موجودیت‌ها و سرویس‌ها

این پروژه شامل دو برنامه است، برنامه سرور (Server Side) و برنامه کاربر (Client Side) است. برنامه سمت سرور وظیفه پردازش ورودی‌هایی که از سمت کلاینت می‌آید را دارد مانند ثبت‌نام و احراز هویت کاربر، ارسال توییت و ... و سمت کلاینت صرفا یک ابزار برای استفاده از سرور است همانگونه که ما با استفاده از برنامه نصبی در تلفن‌همراه خود از توییتر استفاده می‌کنیم که در این حالت تلفن‌ما کلاینت و برنامه توییتر که به وسیله اینترنت به آن متصل می‌شویم سرور است. در این فاز به پیاده‌سازی موجودیت‌ها و سرویس‌هایی که سرور پروژه ارائه می‌کند می‌پردازیم.

توییتر شامل تعدادی موجودیت اصلی می‌باشد که به شرح برخی از آنها می‌پردازیم. توجه داشته باشید به همه موجودیت‌ها به صورت مستقیم اشاره نمی‌شود. در صورت نیاز با تعریف موجودیت جدید برنامه خود را کامل‌تر کنید.

حساب کاربری:

کاربران برای به اشتراک گذاشتن توییت، لایک کردن و مابقی مکانیزم‌های توییتر نیاز به حساب‌کاربری دارند. هر حساب کاربری نماینده یک کاربر در سامانه است. موجودیت کاربر شامل صفاتی من‌جمله نام، نام‌خانوادگی، نام کاربری، گذرواژه، تاریخ تولد، تاریخ عضویت، بیوگرافی است.

- نام‌کاربری هر کاربر باید از بقیه کاربران متمایز باشد.
- بیوگرافی باید به صورت متن و با محدودیت حداکثر ۲۵۶ کاراکتر باشد.
- گذرواژه باید به صورت رمزنگاری شده (Hash) ذخیره شده باشد.

برای مطالعه بیشتر در خصوص رمزنگاری متن در جاوا از این لینک کمک بگیرید.

توییت:

هر کاربر می‌توانند تفکرات خود را به صورت محتوای متنی در قالب توییت به اشتراک بگذارد. هر توییت شامل صفاتی من‌جمله ارسال کننده توییت،تعداد لایک‌ها، محتوا و تاریخ ارسال می‌باشد. صفت محتوا به صورت متنی و طول حداکثر ۲۵۶ کاراکتر می‌باشد و صفت تاریخ ارسال به صورت تاریخ ذخیره می‌شود.

❖ تاریخ ارسال توییت می‌بایست از پکیج java.time باشد. برای مطالعه بیشتر به این لینک مراجعه کنید.

توییتر شامل تعدادی سرویس اصلی می‌باشد که به شرح کلی آن‌ها می‌پردازیم. توجه داشته باشید به همه سرویس‌ها به صورت مستقیم اشاره نمی‌شود. در صورت نیاز سرویس‌هایی به این سامانه اضافه کنید.
منظور از سرویس قابلیت‌هایی است که برنامه باید داشته باشد. این قابلیت‌ها در قالب کلاس‌هایی که پسوند Service در نام آن‌ها آمده است پیاده‌سازی می‌شوند تا از سایر کلاس‌ها متمایز باشند.

سرویس احراز هویت یا Authentication Service

همانطور که در ابتدا ذکر شد یکی از قابلیت‌های سرور توییتر، ثبت‌نام اعضا و یا ورود افرادی است که قبلا ثبت‌نام کرده‌اند. بنابراین برای پیاده‌سازی این سرویس باید ساختاری در نظربگیرید که در درون خود توابعی برای انجام این عملیات‌ها را داشته باشد.

سرویس ارسال توییت Tweeting Service

در این سرویس باید توابعی پیاده‌سازی شود که با استفاده از آن بتوان یک توییت اضافه یا حذف کرد و یا یکی از توییت‌های قبلی را بازنشر یا لایک (Like) کند.

قابلیت پاسخ دادن به یک توییت (Reply)

هر کاربر باید بتواند زیر توییت دیگران نظر بدهد و ریپلای کند. همچنین کاربران می‌توانند روی ریپلای‌های یکدگر نیز ریپلای کنند. به مثال زیر توجه کنید: (این مثال برای پیاده‌سازی در حالت کنسولی برنامه است)

```
| Ted                              11/24/2021 12:24 |
| Hey, what's up?                                   |
----| Barney                                 11/24/2021 18:32 |
----| Suit up!!!                                                |
--------| Ted                                        11/24/2021 18:37 |
--------| I'm not suiting up Barney                                   |

----| Robin                              11/24/2021 13:11 |
----| Same as yesterday                                   |
```

در مثال بالا توییت اول ۲ پاسخ دارد و پاسخ اول نیز ۱ پاسخ دیگر دارد.

موجودیت ریپلای همان توییت است با این تفاوت که به یک شی دیگر اشاره دارند.

سرویس دنبالگر یا Observer Service

در این سرویس باید توابعی پیاده‌سازی شود که یک کاربر بتواند کاربر دیگری را دنبال و یا خود را از لیست دنبال کنندگان یک کاربر حذف کند و همچنین باید تابعی داشته باشد که تمام توییت‌های کاربرانی را که دنبال کرده‌است را برگرداند.

کلیه سرویس‌ها باید بگونه‌ای پیاده‌سازی شوند که در برابر هر ورودی خواه صحیح و خواه ناصحیح خروجی متناسب را داشته باشند.

همانطور که قبلا ذکر شد این سرویس‌ها کامل نیستند و درصورت نیاز می‌توانید به آن‌ها اضافه کنید و همچنین جزئیات پیاده سازی و توابع آن‌ها نیز قابل تغییر است ولی این تغییرات نباید به‌گونه‌ای باشد که ماهیت سرویس و کاربرد آن را تغییر دهد.

سرویس جدول زمانی یا Timeline Service

در این سرویس باید توابعی پیاده‌سازی شود که یک کاربر بتواند توییت‌ها، لایک‌ها، بازنشرها و ریپلای‌های کاربرانی را که دنبال کرده است را مشاهده کند. همچنین این توییت‌ها باید با توجه به زمان انتشار مرتب شوند. نمایش و پیاده‌سازی پاسخ به توییت‌ها (ریپلای‌ها) اختیاری است.

فاز دوم: **شبکه**

در این فاز باید با استفاده از مفاهیمی که در مباحث شبکه و Socket Programming آموخته‌اید برنامه‌ای را بنویسید (درسمت سرور) که ابتدا کاربرها (Client) به آن متصل می‌شوند و سپس درخواست‌های خود را مانند مشاهده توییت‌ها و یا ارسال توییت به سرور می‌دهند و این سرور با استفاده از سرویس‌هایی که در فاز قبلی برای آن پیاده‌سازی کرده‌اید عملیات را انجام داده و پاسخ متناسب را به کاربر ارسال می‌کند. باید سرور را به‌گونه‌ای طراحی کنید که همیشه آماده اتصال کلاینت باشد و برای هر کلاینت متصل شده یک نشست (Session) ایجاد کند.

منظور از نشست کلاسی است که در آن ورودی‌های کاربر بررسی شده و با استفاده از سرویس‌های سرور به درخواست‌ها پاسخ داده شود. باید توجه کنید که این نشست‌ها برای هر کاربر منحصربه‌فرد است و همزمان با یکدیگر اجرا می‌شوند. پس برای پیاده‌سازی آن باید از مطالب Multithreading استفاده کنید.

فاز سوم: برنامه کاربر به همراه رابط کاربری کنسولی

در این فاز شما باید برنامه‌ای بنویسید که با استفاده از شبکه به سرور متصل شود و ورودی‌هایی را که کاربران به برنامه می‌دهد را برای سرور ارسال و پاسخ دریافتی را در کنسول نمایش دهد.

برنامه کاربر شامل سرویس های زیر است:

این سرویس‌ها کامل نیستند و در صورت لزوم می‌توانید به آن‌ها اضافه کنید.

سرویس تجزیه کننده ورودی کاربر یا Command Parser Service

در این سرویس باید توابعی پیاده‌سازی شوند که ورودی‌های کاربر را پردازش و با استفاده از دیگر سرویس‌ها درخواست کاربر را به سرور ارسال کند و همچنین به کمک سایر سرویس‌ها پاسخ سرور را نمایش دهد. به بیان دیگر این سرویس مانند یک کنترلگر اجزار مختلف برنامه client را مدیریت می‌کند.

سرویس شبکه یا Connection Service

این سرویس وظیفه برقراری ارتباط با سرور و فرستادن و دریافت پیام را دارد.

سرویس نمایشگر کنسول یا Console view Service

این سرویس وظیفه نمایش پاسخ سرور در کنسول را دارد که باید بر اساس نوع پاسخ خروجی متناسبی را در کنسول چاپ کند به مثال‌های زیر توجه کنید:

Log-In

در این صفحه باید اطلاعات حساب کاربری از کاربر گرفته شود و با موفق بودن احراز هویت وارد حساب کاربری خود شود.

```
Attempt Log-In
username= RichardHendricks
password= *********

Logging in...
Logging in...
Logging in...

Log-In attempt successful, welcome back!
```

Timeline

در این صفحه باید تمام توییت‌های کاربرانی را که کاربر وارد شده به برنامه دنبال کرده است به ترتیب زمان انتشار نمایش داده شود: نمایش و پیاده‌سازی پاسخ به توییت‌ها اختیاری است.

```
| Ted                          11/24/2021 12:24 |
| Hey, what's up?                               |
| 0 Retweets, 2 Likes                           |
----| Barney                                11/24/2021 18:32 |
----| Suit up!!!                                             |
----| 0 Retweets, 0 Likes                                    |
--------| Ted                                      11/24/2021 18:37 |
--------| I'm not suiting up Barney                                 |
--------| 0 Retweets, 0 Likes                                       |

----| Robin                           11/24/2021 13:11 |
----| Same as yesterday                                |
----| 0 Retweets, 0 Likes                              |

> Liked by Ted
| Marshall                          12/24/2021 1:33 |
| Vikings ruleeee!!                                 |
| 13 Retweets, 158 Likes                                    |
----| Lily                                  12/24/2021 1:34 |
----| They haven’t won in the past 10 years                 |
----| 0 Retweets, 13 Likes                                  |

| Barney                            11/24/2021 14:24 |
| It’s gonna be Legen-WAIT FOR IT-Dary               |
| 137 Retweets, 999 Likes                            |
```

توجه کنید فرمت پیاده سازی رابط کاربری با کنسول تماما به عهده دانشجو است.

فاز چهارم: پیاده‌سازی رابط کاربری گرافیکی

توضیحات مربوط به این فاز در آینده نزدیک به شما ارائه خواهد شد.

نکات پیاده‌سازی

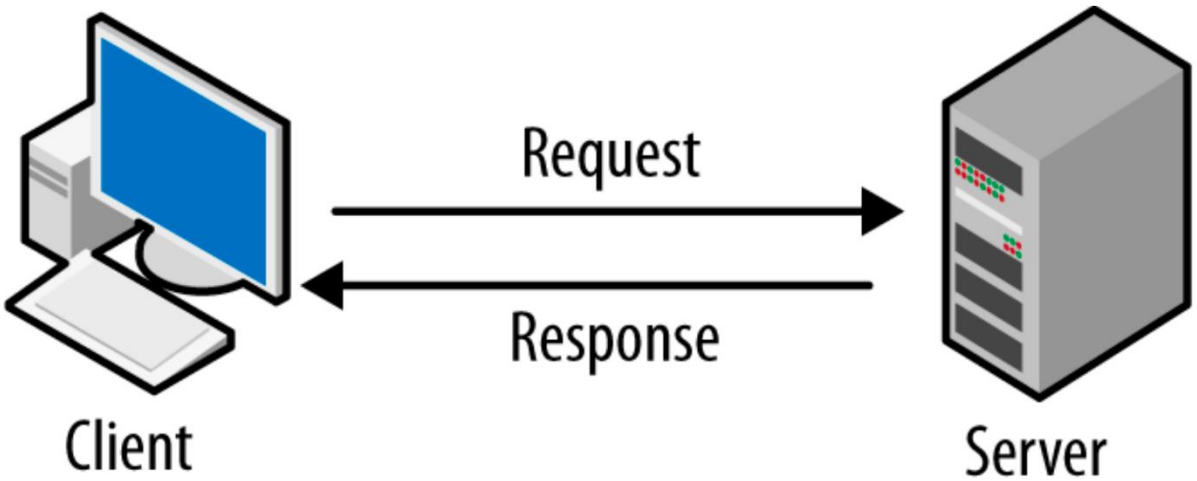
برای پیاده‌سازی این سامانه ملزم به استفاده از شبکه و Socket هستید.

در پیاده سازی این برنامه باید یک سرور و یک کلاینت پیاده‌سازی کنید. به این صورت که یک سرور و چند کلاینت باید ساخته شود و تمامی سرویس‌های برنامه در سمت سرور انجام شود.

کلاینت به سرور درخواست (Request) می‌فرستد. سرور هم متناسب با درخواست فرستاده شده پاسخی (Respond) ارسال می‌کند.

برای این منظور راه‌های مختلفی وجود دارد مانند استفاده از:

1. Multi-threaded Client/Server
2. HTTP Server-Client



فرمت Request و Responseها (حالت اجباری)

فرمت تمامی Request و Responseها باید یا متنی یا به صورت ارسال شیء باشد.

به عنوان مثال؛ اگر کاربری بخواهد توییتی را به اشتراک بگذارد، می‌تواند متن زیر از طرف کلاینت برای سرور فرستاده شود و متناسب با آن پاسخی از سرور دریافت کند.

Request

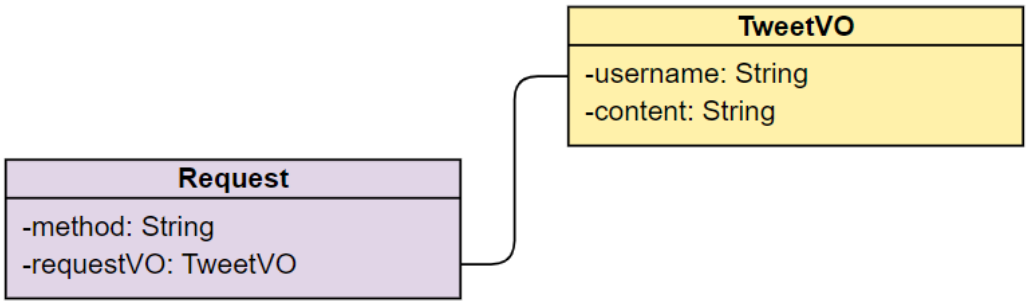
```
-SendTweet {username="RichardHendricks", content="Hello, World!"}
```

Response

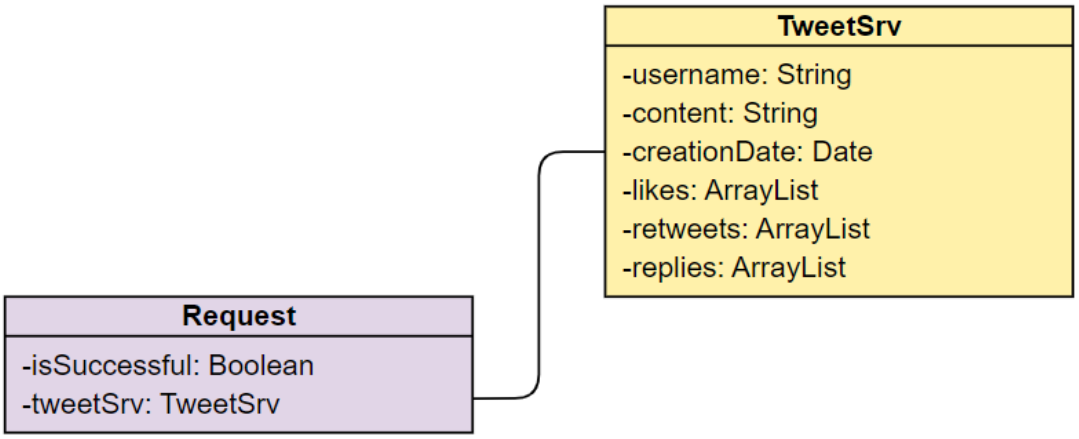
```
[Successful] -SendTweet {username="RichardHendricks", content="Hello,
World!", creationDate="11/24/2021 2:39 PM", likes=[], retweets=[],
replies=[]}
```

یا می‌تواند شیء زیر از طرف کلاینت برای سرور فرستاده شود و متناسب با آن پاسخی از سرور دریافت کند.

Request



Response



لزوما نیاز نیست فرمت متن یا شیء ارسال شده توسط سرور و کلاینت مانند مثال قبل باشد. فرمت متن یا شیء ارسال شده تماما به عهده دانشجو است.

توجه! همانطور که در مثال قبل مشاهده کردید برای لایک، ریتوییت و ریپلای‌های توییت‌ها در پاسخ ارسالی توسط سرور باید از ارایه استفاده کنید. صرفا فرستادن تعداد آنها کفایت نمی‌کند. همچنین اگر توییتی توسط کسانی که دنبال می‌کنید لایک شده باشد، باید در پاسخ فرستاده شود.

فرمت Request و Responseها به صورت فایل JSON (حالت امتیازی)

فرمت تمامی Request و Response ها به صورت فایل JSONباشد.

به عنوان مثال; اگر کاربری بخواهد توییتی را به اشتراک بگذارد، باید فایل JSONزیر از طرف کلاینت برای سرور فرستاده شود و متناسب با آن پاسخی از سرور دریافت کند.

Request:

```
{
  "method": "sendTweet",
  "description": "Validate tweet and send it",
  "parameterValues": {
    "content": "Hello, World!",
    "username": "RichardHendricks"
  }
}
```

Response:

```
{
  "hasError": false,
  "errorCode": 0,
  "count": 1,
  "results": [
      {
        "tweetID": 1,
        "content": "Hello, World!",
        "username": "RichardHendricks",
        "creationDate": "11/24/2021 2:39 PM",
        "likes": [],
        "retweets": [],
        "replies": []
      }
    ]
}
```

همانطور که گفته شد برای رد و بدل Request و Responseها بین کلاینت و سرور باید از فایل JSON استفاده کنید. لیست زیر به توضیح مختصری از پارامترهای ارسالی می‌پردازد.

فرمت درخواست‌ها (Requests)

هر Request از ۳ بخش اصلی تشکلی شده است که به توضیح هر یک می‌پردازیم:

1. Method
2. Description
3. ParameterValues

1. نام متد (Method): این فیلد به سرور می‌فهماند کدام متد را در برنامه خود صدا بزند تا درخواست کاربر را به انجام برساند. به عنوان مثال در مثال صفحه قبل متد sendTweet باید صدا زده شود. (نام و عملکرد متدها می‌تواند متفاوت باشد.)
2. توضیحات (Description): این فیلد توضیحات متدی که قرار است در سرور صدا زده شود شرح می‌دهد.
3. پارامترهای ورودی (ParameterValues): این فیلد که ارایه‌ای از اشیاست تمامی پارامترهای ورودی متدی که قرار است در سرور صدا زده شود را در خود دارد.

توجه شود اگر در پارامترهای ورودی نیاز به ارسال ارایه داشتید باید به این صورت ارایه فرستاده شود:

```
{
  "method": "something",
  "description": "Does something cool",
  "parameterValues": {
    "foo": "bar",
    "somethingArray": [
        "something1",
        "something2"
      ]
  }
}
```

فرمت پاسخ‌ها (Responses)

هر Response از ۴ بخش اصلی تشکلی شده است که به توضیح هر یک می‌پردازیم:

1. HasError
2. ErrorCode
3. Count
4. Results

1. HasError: اگر در صدا زدن متد ذکر شده در Request خطایی باشد (مانند طولانی تر بودن توییت از ۲۵۶ کاراکتر) این فیلد true برگردانده می‌شود در غیر این صورت false است.
2. ErrorCode: اگر HasError صحیح باشد باید یک کد برای خطا مورد نظر ارسال شود (مانند ارسال کد ۹۹۹ اگر توییت از ۲۵۶ کاراکتر طولانی تر باشد) در غیر این صورت ۰ می‌باشد.
3. Count: اندازه ارایه Results را مشخص می‌کند. در مواردی که لیستی از اشیا در پاسخ فرستاده می‌شود این مقدار از ۱ بیشتر خواهد بود.
4. Results: ارایه‌ای از اشیا که به عنوان پاسخ فرستاده می‌شود.

همچنین می‌توانید در صورت بروز خطا بجای HasError و ErrorCode یک ErrorObject بفرستید. در این صورت دیگر نیازی به فرستادن Count و Results در صورت بروز خطا ندارید.

به عنوان مثال در صورت موفق بودن سرویس، پاسخ (Response) زیر فرستاده شود:

```
{
  "count": 1,
  "results": [
      {
       "results": "Something"
      }
    ]
}
```

و در صورت بروز خطا ErrorObject زیر فرستاده شود:

```
{
  "errorType" : "validation" ,
  "errorCode" : "InvalidUsernameException" ,
  "errorParams" :[]
}
```

توجه! تعریف هر ErrorCode و معنی آن در کلاسی معنی دار مانند ErrorCode.java الزامی است.

به عنوان مثال اگر کاربر قصد دریافت تایم‌لاین خود را داشته باشد Request و Response این‌چنین خواهند بود:

Request

```
{
  "method": "timeline",
  "description": "Fetch timeline",
  "parameterValues": {
    "username": "Ted"
  }
}
```

Response

```
{
  "hasError": false,
  "errorCode": 0,
  "count": 2,
  "results": [
      {
        "tweetID": 3,
        "content": "I took 12 sleeping pills and I’m still awake.",
        "username": "Robin",
        "creationDate": "11/24/2021 12:33 AM",
        "likes": [
            "Ted",
            "Barney",
            "Blah blah"
          ],
        "retweets": [],
        "replies": [
            {
              "username": "Lily",
              "content": "That’s a bummer"
            }
          ]
      },
      {
        "tweetID": 17,
        "content": "I’m out of ideas.",
        "username": "James",
        "creationDate": "11/24/2021 18:01 PM",
        "likes": [
            "Ted",
            "Barney",
            "Marshall",
```

```
            "Lily",
            "Tom"
        ],
        "retweets": [
            "Peter"
        ],
        "replies": []
    }
  ]
}
```

توجه! همانطور که در مثال قبل مشاهده کردید برای لایک، ریتوییت و ریپلای‌های توییت‌ها در پاسخ ارسالی توسط سرور باید از ارایه استفاده کنید. صرفا فرستادن تعداد آنها کفایت نمی‌کند. همچنین اگر توییتی توسط کسانی که دنبال می‌کنید لایک شده باشد، باید در پاسخ فرستاده شود. (توییت دوم مثال فوق، پارامتر **status**)

در درخواست (**Request**) و پاسخ (**Response**)های ارسالی می‌توانید پارامترهای بیشتری بفرستید و به عهده دانشجو است.

کار با فایل

در هر لحظه این امکان وجود دارد که سرور متوقف و دوباره روشن شود. لذا برای این کار نیاز دارید اطلاعات سرور را در فایلی ذخیره کنید تا بعدا بتوان از آن اطلاعات استفاده کرد و به ادامه کار پرداخت.

اطلاعات سرور باید روی فایل و فولدر بندی متمایزی صورت گیرد.

```
%POJECT_PATH%\
              |
              |files\
              |     |log/
              |     |
              |     |model\
              |     |     |
              |     |     |%SELF_MADE_OBJECT/
              |     |     |tweets/
              |     |     |users\
              |     |     |     |richardHendricks.txt
              |     |     |     |Ted.txt
              |
```

همانطور که مشاهده می‌کنید در فولدر users که مربوط به کاربران ثبت‌نام شده در سامانه است برای هر کاربر فایل جدا در نظر گرفته شده که نام فایل همان نام کاربری آنها است.

این فرمت را برای تمامی موجودیت‌های دگر حفظ کنید. لزوما نیاز نیست از نام کاربری استفاده کنید و این موضوع تماما به اختیار دانشجو است.

ثبت لاگ توسط سرور

سرور باید بتواند تمامی فعالیت‌های سرویس‌های خود را در فایلی به ادرس files/log ثبت کند. این فعالیت‌ها می‌تواند شامل صدا زده شدن سرویس‌ها، موفق یا غیر موفق بودن انجام عملیات مختلف من جمله ورود کاربران به سامانه، ثبت کاربر جدید، توییت زدن و ... باشد.

برای سادگی، لاگ مشابه پیام هایی است که در کنسول چاپ می‌شود تا متوجه شویم برنامه چگونه عمل می‌کند. با این تفاوت که در فایل سیو می‌شود و ساختار مشخصی دارد.

به مثال زیر توجه کنید:

```
[INFI] Client@12345, -Attempt LOG_IN
[ERROR] Client@12345, -Wrong Password
```

در این مثال کاربری تلاش کرده تا با کمک **سرویس احراز هویت** وارد اکانت خود شود ولی با وارد کردن گذرواژه اشتباه موفق به وارد شدن نشده است.

نظارت فعال گزارش سرور می تواند به جلوگیری از نفوذ تصادفی و مخرب به سیستم کمک کند.

چگونگی و فرمت ثبت لاگ توسط سرور به عهده دانشجو است.

- **می توانید برای ثبت کردن لاگ یک سرویس مجزا درست کنید.**

ساختار فولدرهای پروژه

برای معماری بهتر پروژه، از فولدر بندی زیر استفاده کنید. هریک از پکیج‌ها به اختصار شرح داده شده است.

```
%POJECT_PATH%\
              |
              |files/
              |
              |out/
              |
              |src/main\
                      |
                      |java/java/org/ce/ap\
                      |                    |
                      |                    |server/
                      |                    |     |
                      |                    |client/
                      |
                      |
                      |
                      |recourses\
                            |client-application.properties
                            |
                            |server-application.properties
```

این ساختار در پیوست پروژه موجود است.

1. محتویات فولدر client ممکن است مشابه فولدر server باشند.
2. مقادیر داخل فایل‌های properties کاملا قابل تغییر هستند.
   - مقادیر پیش‌فرض در فایل پیوست صرفا برای نمایش مثال است و استفاده از مقادیر آن الزامی نیست. شما می‌توانید مقادیر دلخواه را وارد کنید.
3. هر کلاسی که نقش سرویس را ایفا می‌کند باید ابتدا متدهای آن به صورت abstract و در یک interface تعریف شود و سپس پیاده سازی آن در پکیجی به‌نام impl قرار بگیرد.
   - این کار به این منظور است که وظایف یک سرویس کاملا مشخص است ولی پیاده سازی در ورژن‌های مختلفی از برنامه می‌تواند تغییر کند.
4. نحوه اسم‌گذاری: اگر interface داشته باشیم به اسم TestController، کلاسی که آن را پیاده‌سازی می‌کند باید نام کلاس TestControllerImpl باشد.
5. اگر در قسمت‌های دیگر پروژه از interface یا abstract class استفاده کرده‌اید باید مانند مثال‌های قبل عمل کنید. (نباید interface و کلاس پیاده سازی آن دقیقا در یک package باشند).

6. باید تمامی تنظیمات قابل تغییر مانند درگاه (Port) سرور، محل ذخیره شدن فایل‌ها و ... را چه در برنامه سرور و چه در کلاینت از فایل تنظیمات (Properties) بخوانید. در این خصوص ویدئوی آموزشی در اختیار شما قرار می‌گیرد.

نکات فوق اصول اولیه طراحی با زبان جاوا هستند و رعایت آن‌ها الزامی است

امتیازی

1. قابلیت پاسخ دادن به یک توییت (**Reply**) (۵ امتیاز)
2. فرمت Request و Responseها به صورت فایل JSON (۲۰ امتیاز)
3. پیاده‌سازی چت‌روم برای کاربران تا با یکدیگر مکالمه داشته باشند. (۱۵ امتیاز)
4. استفاده از هر Relational Database برای ذخیره اطلاعات موجودیت‌ها. (۵ امتیاز)

توجه! بیشینه نمره امتیازی ۳۵ نمره می‌باشد. در صورت کسب نمره امتیازی بیشتر در نمره شما لحاظ نمی‌گردد.