# فاز سوم پروژه درس برنامه نویسی پیشرفته

مهلت ارسال : 17 تیر 1398

به موارد زیر توجه کنید:

- بعد از اتمام این فاز پروژه تگی در گیت خود به نام phase_3 بزنید. در روز تحویل حضوری این tag بررسی خواهد شد.
- در روز تحویل حضوری مشارکت تمام اعضای تیم در پروژه بررسی خواهد شد و در صورت عدم مشارکت بعضی از اعضا نمره آنها برای آن فاز 0 منظور خواهد شد.
- در هر فاز میتوانید سه روز تاخیر به ازای کسر نمره داشته باشید. (به ازای هر روز 10 درصد از نمره شما کسر خواهد شد.)
- در مجموع سه فاز سه روز تاخیر بخشیده خواهد شد.
- درصورت کشف تقلب از هر یک از تیم ها، برای بار اول نمره منفی فاز برای آن تیم ثبت می شود و برای بار دوم، نمره منفی کل پروژه برای تیم ثبت خواهد شد که معادل با مردود شدن در درس می باشد.

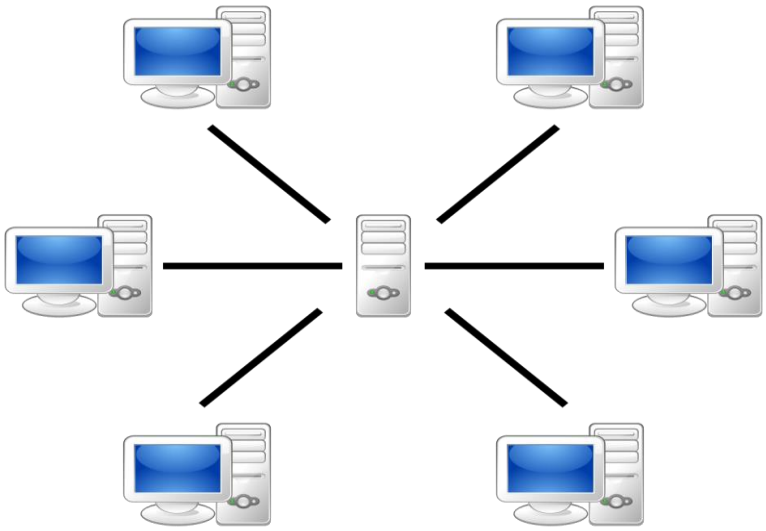
مسوول پروژه: سید علیرضا حسینی

طراحان پروژه: سید علیرضا حسینی، سید مهدی صادق شبیری، سجاد رضوانی، امید مسگرها، مهدی جوانمردی

## فاز سوم - شبکه

در فاز سوم به پیاده سازی شبکه و مدیریت چندین پروسه از بازی بدین وسیله می‌پردازید. باید در انتهای این فاز بتوانید تعداد زیادی بازی به سرور متصل شوند. ابتدا کمی به آشنایی معماری شبکه‌ای که در این فاز پیاده خواهید کرد می‌پردازیم.

## معماری *Client Server* و ارتباطات اصلی



معماری شبکه استفاده شده در این فاز از نوع کلاینت-سرور (در مقابل می‌توان به معماری peer-to-peer اشاره کرد که برخلاف این معماری decentralized است) است. در این معماری یک کامپیوتر نقش کارگزار (سرور) را برعهده می‌گیرد و سایر کامپیوترها با استفاده از آی‌پی سرور به آن متصل می‌شوند. در نتیجه شما در این فاز باید بخش سرور را آماده کرده و برخی وظایف بازی را به سرور منتقل کنید.

## نحوه ارتباط کلاینت و سرور

از آنجایی که مدیریت حساب کاربری به سرور منتقل می‌شود، بازی تنها به صورت آنلاین امکان پذیر خواهد بود.

ارتباط کلاینت ها و سرور از طریق سوکت برقرار خواهد شد. این ارتباط می‌تواند از نوع TCP و یا UDP باشد. ( برای مطالعه بیشتر هرکدام به این و این و این مراجعه نمایید)

برای اتصال کافیست سرور روی یک پورت خاص گوش کند و کلاینت ها با اتصال به آن پورت به سرور متصل شوند.

توجه کنید که سرور و کلاینت به صورت جداگانه اجرا می‌شوند و پورت مذکور را از یک فایل config مشخص شده می‌خوانند. علاوه بر این یک پورت به صورت دیفالت (8000) در برنامه خواهد بود که در صورت عدم تامین پورت توسط فایل، مورد استفاده قرار می‌گیرد.
شما در این فاز باید به پیاده سازی بخش های زیر با استفاده از معماری کلاینت سرور بپردازید.

توجه نمایید که در این فاز شما موظف به پیاده‌سازی دو کد مجزا برای سرور ( که وظایفی همچون ارتباط بین کلاینت ها، authentication و ... دارد ) و کلاینت ( که همان کد فاز دوی شما با کمی تغییر و اضافه شدن ارتباط با سرور است ) هستید.

## موارد اجباری فاز سوم

### Accounts:
شما باید در این قسمت، ورود و خروج را به هاست بازی منتقل کنید. یعنی اینکه با اجرا شدن کلاینت، در صفحه اول باید به صورت خودکار به هاست بازی متصل می شود. شما باید سیستم مدیریت اکانت ها را به سرور بازی منتقل کنید. سپس در کلاینت بتوان اکانت ساخت یا به اکانت خود وارد شد. در واقع تمام اکانت ها و رمز آنها باید به یک سرور منتقل شوند.
مراحل ساخت اکانت یا ورود و خروج به اینصورت است که ابتدا پیامی از کلاینت بازی به هاست ارسال میشود که بیانگر عملیات مورد نظر است و هاست بازی نتیجه عملیات (ورود موفقیت آمیز یا ...) را برای کلاینت ارسال میکند. شما باید در سرور بازی، بعد از ورود موفقیت آمیز، یک کلید برای کلاینت بفرستید که کلاینت در تمام پیام های خود به سرور بازی این کلید را در فیلدی تحت عنوان AuthToken، بفرستد.
در واقع هاست با دریافت AuthToken تشخیص خواهد داد که کاربر لاگین کرده است یا خیر. درصورتی که کلاینت لاگین نکرده باشد این فیلد باید نال باشد.
تمامی اطلاعات گفته شده در فاز اول که باید در اکانت ذخیره می شدند اکنون نیز باید در اکانت و در داخل هاست ذخیره شوند.
لذا بعد از بستن کلاینت و بازکردن دوباره آن، و ورود دوباره به حساب کاربری خود، باید تمام اطلاعات در آن موجود باشد.

### GlobalChat:
در قسمت مولتی پلیر بازی چت رومی به نام چت همگانی ( Global Chat ) پیاده سازی کنید که در آن تمامی بازیکن هایی که آنلاین هستند می توانند گفت و گو کنند .

### Multiplayer:
در منوی مولتی پلیر بعد از آنکه مد بازی و دکی که بازیکن میخواهد در بازی استفاده کند ، انتخاب شد ، قسمتی را تعبیه کنید که بازیکن ها در آن به هاست درخواست میدهند تا بازی کنند . هاست در صورتی که بازیکن دیگری را که درخواست بازی داده است را پیدا کرد ، بازی بین این دو بازیکن شروع می شود( در این قسمت هاست اولین بازیکنی که درخواست بازی داده است را پیدا می کند ) ؛ پیش از اینکه بازی شروع شود صفحه باید قفل شود و گزینه ای برای انصراف وجود داشته باشد . بنابراین بازیکن تنها می تواند منتظر حریف خود بماند یا از درخواست بازی خود انصراف دهد. اینکه هاست بتواند همزمان چند بازی مشترک را پشتیبانی کند امتیازی محسوب میشود.
بازی هم به این صورت شروع می شود که به صورت تصادفی نوبت اول بازی به یکی بازیکن ها داده می شود . سپس بازیکن نوبت خود را بازی می کند و پس از اتمام نوبت ، بازیکن دیگر نوبت خود را بازی می کند و همین طور ادامه یابد . ( قوانین بازی تغییر نمی کند و سیاست های تغییر مانا و حمله و دفاع و .. همان هایی است که در فازهای قبل استفاده می کردید )

بازی های این قسمت در مد های مختلف و با توجه به پایان بندی هر مد تمام می شوند و در آخر نتایج بازی از جمله جوایز در هاست بر روی دارایی های هر بازیکن ( یا ویژگی های دیگر مثل تعداد برد و باخت ) تاثیر می گذارد .
اسکوربوردی را که پیش تر پیاده سازی کرده بودید بگونه ای تغییر دهید که براساس تعداد برد و باخت در قسمت مولتی پلیر بازیکن ها را رتبه بندی کند و نشان دهد . ( آنلاین یا آفلاین بودن بازیکن در این رتبه بندی تاثیر گذار نیست بنابراین چنین اسکوربوردی باید در هاست پیاده سازی شود.)

**Shop:**

در این فاز فروشگاه بازی باید به هاست منتقل شود و تمامی کارت هایی که در شاپ وجود دارند از یک ظرفیت محدود برخوردار باشند. اگر بازیکنی کارتی را خرید ، ظرفیت آن در شاپ یکی کم می شود. در صورتی که ظرفیت کارتی تمام شد هیچ بازیکنی ، دیگر نمی تواند از آن کارت بخرد . تمامی تغییراتی که روی اقلام شاپ اتفاق می افتد باید در سرور ذخیره شود.
از طرفی بازیکن ها می توانند کارت هایی که دارند را بفروشند و در این صورت ظرفیت آن کارت در شاپ یکی بیشتر می شود.

توضیحاتی درباره **Auth Token:** یک کلید برای ارتباط بین کلاینت و سرور است. سرور برای هر کاربر یک Auth Token مختص آن می سازد و با ورود یک کلاینت، این کلید را برای آن کلاینت میفرستد. با ورود دوباره هر کاربر، این فیلد باید دوباره آپدیت شود. سپس کلاینت در هر پیامی که برای هاست میفرستد این Auth Token را می فرستد و هاست از طریق این کلید متوجه می شود که پیام برای چه کسی است.

**موارد امتیازی : (نیازی نیست همه موارد را پیاده کنید. با دریافت 60 درصد نمرات امتیازی، شما تمام نمره امتیازی پروژه را دریافت می کنید. اگر بیش از 60 درصد نمرات امتیازی کسب کنید مازاد آن تقسیم بر 4 شده و به نمرات فازهای پروژه شما اضافه می شود.)**

1- **نمایش وضعیت آفلاین یا آنلاین بودن کاربر ها در score board**
2- **مزایده :**
   - در فروشگاه (Shop) گزینه‌ای وجود داشته باشد برای اینکه کاربر بتواند کارت خود را به مزایده بگذارد.
   - مزایده اینگونه است که فرد فروشنده کارت خود را به مزایده می‌گذارد و دیگران ۳ دقیقه وقت دارند که قیمت‌های پیشنهادی خود را بدهند و بعد از ۳ دقیقه کارت به کسی که بیشترین قیمت را داده می‌رسد و پول خریدار به فروشنده.
   - هرکاربر باید بتواند بالاترین قیمت را در لحظه ببیند.
   - در فروشگاه قیمت خرید از قیمت فروش باید بیشتر باشد.
   - در فروشگاه از هر کارت به تعداد محدود باشد. (مثلا ۱۰ تا)
3- در بازی آنلاین زمانی که هر بازیکن در هر Turn دارد، محدود باشد و در صورتی که زمانش تمام شد، Turn به پایان می‌رسد و نوبت بازیکن مقابل شود.
4- **ایجاد کارت سفارشی در سرور:**

در این قسمت، با انتقال فروشگاه به هاست بازی، باید بتوان کارت سفارشی جدیدی مطابق با موارد مطرح شده در فاز دوم، در برنامه host اضافه نمود به صورتی که بعد از ایجاد کارت، اطلاعات این کارت به تمام کلاینت ها منتقل شود و تمام کاربران بتوانند این کارت را بخرند و از آن استفاده کنند.
انتقال اسپرایت کارت، (اپلود در سرور و انتقال آن اسپرایت به کلاینت ها از طریق سوکت) دارای نمره اضافی بیشتری نیز است.

توجه کنید تنها درصورتی نمره این قسمت را دریافت می کنید که بتوان از کارت خریداری شده در بازی استفاده نمود.و در غیر اینصورت هیچ نمره ای دریافت نمی کنید.

5- **مشاهده زنده بازی در حال انجام:**

در تمام کلاینت های متصل به یک هاست، بتوان بازی درحال انجام بین دو نفر را مشاهده کرد.
در لیست بازی های در حال انجام، گزینه ای وجود داشته باشد که با انتخاب هر کدام بتوانیم بازی را به صورت زنده، مشاهده کنیم.
برای پیاده سازی این مورد ممکن است با توجه به حجم داده هایی که انتقال می دهید، از طریق پروتکل های tcp بسیار کند باشد.
برای اینکار میتوانید از پروتکل های udp استفاده کنید.

6- **استفاده از دیتابیس داده شده:**

در آخر این داک توضیحات پایگاه داده ای که تیم پروژه برای شما آماده کرده است آمده شده است. استفاده از این پایگاه داده برای ذخیره اطلاعات امتیازی است. توضیحات بیشتر در آخر داک آمده است.
این دیتابیس را میتوانید از این لینک دانلود کنید.

## آشنایی با مفهوم : REST framework

REST مخفف عبارت Representational State Transfer است. این پروتکل سال هاست که به عنوان عضوی از دنیای وب شناخته میشود که توسط آن با هر زبان برنامه نویسی‌ای می‌توان API هایی را برای دسترسی آسان به اطلاعات و یا بارگذاری آنها در وب ساخت.
معماری REST دارای یکسری ویژگی‌ها است که شاخص ترین آن‌ها عبارتند از:
- ثبات و یکنواختی این معماری در API
- به‌کارگیری از کدهای وضعیت اچ تی تی پی
- استفاده از url ها برای مشخص ساختن مسیرهای مد نظر

## دیتابیس پیش‌ساخته miniDB:

یکی از موارد امتیازی شما استفاده از این دیتابیس آماده برای ذخیره کردن اطلاعات خود است.
در این لینک یک فایل jar قرار دارد که با اجرا کردن آن در کنار فایل jar یک دیتابیس به طور خودکار ایجاد می‌شود که شما میتوانید با تعامل کردن با این دیتابیس( شیوه تعامل کردن توضیح داده خواهد شد) از نمره این بخش برخوردار شوید.در این دیتابیس اطلاعاتی از قبیل کارت های کاستوم ساخته شده، پروفایل های ساخته شده بازی، تاریخچه بازی های انجام شده و ... را میتوان ذخیره کرد که نمره آن با توجه مقدار استفاده شما از این دیتابیس تعلق خواهد گرفت.

## فراخواندن توابع REST در جاوا :

برای صدا کردن توابع این فریمورک راه های بسیار زیادی هست که یکی از آسان ترین آنها میشود به کتابخانه unirest اشاره کرد که خودش تمام کار های برقراری ارتباط و گرفتن پیغام های سرور را انجام میدهد.برای استفاده میتوانید از اینجا فایل های jar مورد نیاز را دانلود کنید و به پروژه خود اضافه کنید و یا اگر از maven در پروژه خود استفاده میکنید این ها را در فایل pom پروژه درج کنید :

```
<!-- Pull in as a traditional dependency -->
```

```
<dependency>
    <groupId>com.konghq</groupId>
    <artifactId>unirest-java</artifactId>
    <version>2.3.08</version>
</dependency>

<!-- OR as a snazzy new standalone jar with shaded dependencies -->
<dependency>
    <groupId>com.konghq</groupId>
    <artifactId>unirest-java</artifactId>
    <version>2.3.08</version>
    <classifier>standalone</classifier>
</dependency>
```

بعد از اضافه کردن این کتابخانه ها میتوانید به این صورت از آنها استفاده کنید :

```
final String baseAddress = "http://127.0.0.1:8080/";
final String path = "get_all_keys";
HttpResponse<String> response= null;
HashMap<String, Object> parameters = new HashMap<>();
parameters.put("name", "DB_name");
parameters.put("key", "key for adding to a DB or deleting from it");
parameters.put("value", "your content to save");
try {
  response = Unirest.post(baseAddress + path)
        .fields(parameters)
        .asString();
} catch (UnirestException e) {
  e.printStackTrace();
  //do something
}
```

**استفاده از دیتابیس :**

**این دیتابیس دارای تعدادی تابع با شیوه های فراخوانی مختلف هست که آنها را به اختصار توضیح میدهیم :**

**-وضعیت درخواست:**

شما میتوانید موفق بودن درخواست خودتان را با تابع()response.getStatus چک کنید.اگر این عدد ۲۰۰ بود به معنای درخواست موفق است در غیر این صورت درخواست با خطا مواجه شده است.

از اینجا به بعد بعد برای فراخواندن هر کدام از توابع از توابع شیوه درست انتخاب کردن متغیر ها را می گوییم :

تابع ساختن دیتابیس جدید :

path = “init_DB”
Parameter to add : name.

تابع اضافه کردن یک مقدار به یک دیتابیس :

path = “put”
Parameters to add : name(DB’s name), key, value

تابع گرفتن تمام کلید های یک دیتابیس:

path = “get_all_keys”
Parameter to add : name

تابع گرفتن تمام مقدار های یک دیتابیس:

path = “get_all_values”
Parameters to add : name

تابع گرفتن یک مقدار خاص از دیتابیس:

path = “get”
Parameters to add : name, key

تابع پاک کردن یک مقدار :

path = “del_from_DB”
Parameters to add : name, key