مدرس: **احسان خامس‌پناه**
طراحان: **شهریار سلطان‌پور، امیرمحمد رنجبر پازکی**

مهلت تحویل: یکشنبه ۲۰ بهمن ۱۳۹۸، ساعت ۲۳:۵۵

لقمه!

مقدمه

هدف از این فاز، آشنایی با git، maven و unit testing است که در تمام فازهای بعدی به آن‌ها نیاز خواهید داشت. همچنین، بخشی از منطق اصلی را نیز پیاده‌سازی می‌کنید.

توجه داشته‌باشید که در این فاز نیازی به پیاده‌سازی مفاهیم وب ندارید و در فازهای بعدی به این مفاهیم پرداخته می‌شود.

کلیت پروژه

سیستمی که در طی درس به توسعه آن خواهید پرداخت، یک سیستم سفارش غذاست. در این برنامه، کاربر وارد می‌شود و نام رستوران یا غذای مورد نظر خود را جستجو می‌کند. غذای خود را انتخاب می‌کند و به سبد[1]ش اضافه می‌کند. پس از نهایی کردن خرید و پرداخت، سفارش غذا نهایی می‌شود و کاربر منتظر دریافت غذا نیز می‌ماند. همچنین، روزانه تعدادی پیشنهاد ویژه به عنوان جشن غذا[2] نیز گذاشته می‌شود که تعداد محدودی با قیمت بسیار پایینتر دارند.

گام‌های پیاده‌سازی

ایجاد پروژه‌ی Maven[3]

ابتدا یک پروژه‌ی maven بسازید و با ساختار ایجاد شده توسط آن آشنا شوید. همچنین، فایل pom.xml و اطلاعات داخل آن را مشاهده کنید.

پیشنهاد ما برای انجام پروژه‌های این درس، استفاده از Intellij IDEA می‌باشد. برای ایجاد پروژه maven در فضای IntelliJ، می‌توانید از این لینک استفاده کنید.

اضافه کردن وابستگی[4]های موردنیاز

داده‌های مورد نیاز برای انجام این فاز، در قالب json به شما داده می‌شود. برای خواندن و تجزیه json و تبدیل آن به فرمت موردنیاز از پکیج های مخصوص کار با json در java استفاده کنید. در اینترنت جستجو کنید و بهترین پکیج موجود را به وابستگی‌های پروژه خود اضافه کنید.

---

[1] Cart

[2] Food Party

[3] برای اطلاعات بیشتر در مورد Maven می‌توانید به لینک‌های زیر مراجعه کنید:
https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

https://maven.apache.org/guides/getting-started

[4] Dependency

پیاده‌سازی منطق برنامه

در این فاز نیاز است تا تعدادی از عملیات‌های پایه‌ای برنامه خود را پیاده‌سازی کنید. این عملیات در قالب دستوراتی در خط فرمان[5]، به برنامه شما داده‌خواهند شد. هر دستور در یک خط از ورودی شکل کلی زیر را دارد:

```
commandName <JsonData>
```

commandName نشان‌دهنده نام دستور و JsonData داده‌ی مربوط به آن دستور به شکل serialize[6] شده در قالب json است. برای استفاده از این داده، باید با استفاده از کتابخانه‌ای که در فاز قبل به پروژه اضافه کردید، آن را deserialize کنید. همچنین، قابل ذکر است که JsonData برای همه دستورات وارد نمی‌شود.

تمامی موجودیت‌هایی خود را برای راحتی در حافظه اصلی نگه دارید.

دستوراتی که باید در این فاز پیاده‌سازی کنید، در ادامه آمده‌اند.

۱. اضافه‌کردن رستوران

این دستور به شکل زیر است.

```
addRestaurant <restaurantInfo>
```

restaurantInfo اطلاعات رستوران است که شامل نام(name)، توضیحات(description)، موقعیت جغرافیایی(Location) و منوی(Menu) آن می‌شود. منو لیستی از غذاهای رستوران است.

به عنوان مثال، با وارد کردن دستور زیر،در صورت عدم وجود Hesturan، رستورانی با نام Hesturan، توضیحات luxury و موقعیت جغرافیایی با طول ۱ و عرض ۳ به رستوران‌ها اضافه می‌شود. این رستوان دو غذای gheime با قیمت ۲۰ هزار تومان و Kabab با قیمت ۳۰ هزار تومان در منوی خود دارد.

```
addRestaurant {"name": "Hesturan", "description": "luxury", "location": {"x": 1, "y": 3},
"menu": [{"name": "Gheime", "description": "it's yummy!", "popularity": 0.8, "price":
20000}, {"name": "Kabab", "description": "it's delicious!", "popularity": 0.6, "price":
30000}]}
```

۲. اضافه‌کردن غذا

این دستور به شکل زیر است.

```
addFood <foodInfo>
```

foodInfo اطلاعات غذا است که شامل نام(name)، توضیحات(description)، محبوبیت(popularity)، قیمت (price) و نام رستوران (restaurant) می‌شود.

به عنوان مثال، با وارد کردن دستور زیر، در صورت عدم وجود gheime، غذایی با نام gheime، توضیحات it's yummy و محبوبیت 0.8 به منوی رستوران Hesturan اضافه می‌شود.

اگر رستورانی با این نام وجود نداشت، صرفاً یک پیام مبنی بر عدم وجود رستوران به کاربر نمایش داده می‌شود.

---

[5] CLI (Command Line Interface)

[6] https://stackoverflow.com/a/3316779/7621505

```
addFood {"name": "gheime", "description": "it’s yummy!", "popularity": 0.8,
"restaurantName": "Hesturan", "price": 20000}
```

۳. گرفتن لیست رستوران‌ها

این دستور به شکل زیر است.

```
getRestaurants
```

برای خروجی این دستور، لیست تمام رستوران‌ها (صرفاً نام هر رستوران) در چاپ می‌شود. (لزومی ندارد که خروجی این بخش در قالب Json باشد.)

۴. گرفتن اطلاعات یک رستوران خاص

این دستور به شکل زیر است.

```
getRestaurant <restaurantInfo>
```

برای خروجی این دستور، رستوران مورد نظر به همراه ویژگی‌ها (نام، توضیحات، مکان و منوی رستوران) در قالب json چاپ می‌شود.
اگر رستورانی با این نام وجود نداشت، صرفاً یک پیام مبنی بر عدم وجود رستوران به کاربر نمایش داده می‌شود.
برای مثال، جستجوی Hesturan به صورت زیر است.

```
getRestaurant {"name": "Hesturan"}
```

۵. گرفتن اطلاعات یک غذای خاص

این دستور به شکل زیر است.

```
getFoods <foodInfo>
```

برای خروجی این دستور، غذای مورد نظر به همراه ویژگی‌ها (نام، قیمت، توضیحات و محبوبیت) در قالب json چاپ می‌شود.
اگر غذایی با این مشخصات وجود نداشت، صرفاً یک پیام مبنی بر عدم وجود این غذا به کاربر نمایش داده می‌شود.
برای مثال، جستجوی Kabab در Hesturan به صورت زیر است.

```
getFood {"foodName": "Kabab", "restaurantName": "Hesturan"}
```

۶. اضافه کردن غذا به سبد خرید

این دستور به شکل زیر است.

```
addToCart <foodName>
```

پس از وارد کردن این دستور، باید غذای وارد شده به سبد خرید مشتری اضافه شود. دقت کنید که اگر مشتری غذایی از رستوران دیگری را در سبد خرید داشته باشد، باید با خطای مناسب او را آگاه کنید و غذا را به سبد خرید اضافه نکنید.
اگر غذایی با این مشخصات وجود نداشت، صرفا یک پیام مبنی بر عدم وجود رستوران به کاربر نمایش داده می‌شود.

```
addToCart {"foodName": "Kabab", "restaurantName": "Hesturan"}
```

۷. دیدن سبد خرید

این دستور به شکل زیر است.

```
getCart
```

در خروجی این دستور، لیستی از نام غذاهای سفارش داده شده و تعداد آن‌ها در قالب Json نمایش داده می‌شود.

۸. نهایی کردن سفارش

این دستور به شکل زیر است.

```
finalizeOrder
```

پس از وارد کردن این دستور، باید لیستی از نام غذاهای سفارش داده شده و تعداد آن‌ها در قالب Json نمایش داده شود و پیامی مبنی بر ثبت شدن سفارش به کاربر نمایش داده شود و سبد خرید او خالی شود.

۹. گرفتن لیست رستوران‌های پیشنهادی

این دستور به شکل زیر است.

```
getRecommendedRestaurants
```

پس از وارد کردن این دستور، مقدار محبوبیت برای هر رستوران محاسبه می‌شود و نام سه رستورانی که بالاترین میزان محبوبیت را دارند بازگردانده می‌شود. (لزومی ندارد خروجی در قالب Json باشد.)
مقدار محبوبیت هر رستوران از رابطه‌ی زیر محاسبه می‌شود:

$$Average(foodsPopulations) * distanceFromUser$$

که در آن distanceFromUser فاصله‌ی اقلیدسی رستوران با کاربر است. (فرض کنید کاربر در مبدا مختصات قرار دارد.)

آزمون واحد[7]

در این قسمت باید با استفاده از کتابخانه‌ی Junit برای سناریوی موفقیت آمیز آیتم‌های گرفتن لیست رستوران‌های پیشنهادی و نهایی کردن یک سفارش تست بنویسید. تست‌های شما باید ساختار مناسب Setup، Test و Teardown را رعایت کنند.

افزودن پروژه به گیت

ابتدا در سایت gitlab.com عضو شوید و یک مخزن خصوصی ایجاد کنید. سپس، اکانت ieSpring99 را به پروژه خود اضافه کنید. تمامی تغییرات خود را به گیت اضافه کنید و در نهایت، در مخزن خود بارگذاری کنید. توجه کنید که پروژه شما پس از clone شدن باید به راحتی قابل اجرا باشد. برای تمرین نحوه‌ی کار با گیت توصیه می‌شود که این لینک و برای آشنایی با شیوه‌ی مناسب کامیت توصیه می‌شود که این لینک را مطالعه کنید.

[7] Unit Testing

نکات پایانی

- کافی است که یکی از اعضای گروه Hash مربوط به آخرین کامیت پروژه را در سایت درس آپلود کند. در هنگام تحویل، پروژه روی این کامیت مورد ارزیابی قرار می‌گیرد.
- متاسفانه سایت گیت‌لب برای ایرانیان در دسترس نیست و برای دسترسی به آن می‌توانید از شکن یا FoD استفاده کنید.
- ساختار صحیح و تمیزی کد برنامه بخشی از نمره‌ی این فاز پروژه‌ی شما خواهد بود. بنابراین در طراحی ساختار برنامه دقت به خرج دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده‌ی مشابهت بین کدهای دو گروه، ۵۰٪ نمره‌ی کل پروژه‌ها از گروه متقلب و تقلب‌دهنده کسر خواهد شد و در صورت تکرار، این روند ادامه خواهد داشت.
- سوالات خود را تا حد ممکن در فروم درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آن‌ها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این فاز پروژه ارتباط برقرار کنید. توجه داشته باشید که دیگر شبکه‌های اجتماعی مانند تلگرام راه ارتباطی رسمی با دستیاران آموزشی نیست و دستیاران آموزشی موظف به پاسخگویی در محیط‌های غیررسمی نیستند.
- ایمیل طراحان پروژه:
  - shahryar.soltanpour@gmail.com
  - iamirranjbar@gmail.com