مدرس: دکتر احسان خامس‌پناه سنا ساری نوایی، میثاق محقق

دستگرمی

مقدمه

هدف از این پروژه، آشنایی شما با Git، Java، Maven و Unit Testing است. در این راستا شما به توسعه یک سیستم ساده خواهید پرداخت تا با مفاهیم مذکور آشنا شوید. سیستم در نظر گرفته شده، یک سیستم رزرو هتل است که شامل مدیریت رزروها، اعتبارسنجی تاریخ‌ها و استفاده از فایل‌های JSON می‌باشد.

گام‌های پیاده‌سازی

نصب Java

در ابتدا، ابزارهای توسعه جاوا را بر روی سیستم خود نصب کنید. JDK یا Java Development Kit شامل همه ابزارهای مورد نیاز برای توسعه و اجرای یک برنامه جاوا است. باینری‌های JDK توسط جاهای مختلفی عرضه می‌شوند که می‌توانید در اینجا به طور خلاصه تفاوت آنها را ببینید و بهترین انتخاب را طبق نظر خود نصب کنید.
برای انجام پروژه‌های درس، توصیه می‌شود که نسخه JDK استفاده شده حداقل 17 و ترجیحا 21 باشد.

ایجاد پروژه Maven

در اکوسیستم Java، دو انتخاب معمول برای استفاده به عنوان Build System پروژه خود داریم. این دو انتخاب شامل Maven و Gradle می‌شوند که برای پروژه‌های این درس، از Maven استفاده می‌کنیم.
ابتدا یک پروژه Maven بسازید و با ساختار ایجاد شده توسط آن آشنا شوید. همچنین، فایل pom.xml و محتوای داخل آن را مشاهده کنید. می‌توانید این لینک را جهت آشنایی با ساختار Maven مطالعه کنید.
پیشنهاد ما برای انجام پروژه‌های این درس، استفاده از ابزار IntelliJ IDEA می‌باشد. نسخه رایگان (Community Edition) این ابزار برای انجام پروژه‌های این درس کافی است. برای ایجاد پروژه Maven در محیط توسعه Intellij IDEA، می‌توانید از این لینک استفاده کنید.

منطق برنامه

شما باید یک سیستم رزرو هتل را پیاده‌سازی کنید. این سیستم از کلاس‌های زیر تشکیل شده است:

کلاس Customer

- این کلاس تنها شامل داده‌های زیر بوده و منطق خاصی ندارد.
- اطلاعات مشتری هتل شامل کد ملی، نام، شماره تماس و سن او می‌باشد.

کلاس Room

- این کلاس نیز فقط شامل داده‌های زیر بوده و منطق خاصی ندارد.
- اطلاعات اتاق شامل شماره اتاق و ظرفیت هر اتاق می‌باشد.

کلاس Booking

- این کلاس مسئول مدیریت اطلاعات رزرو را بر عهده دارد.
- این اطلاعات شامل آیدی رزرو، فردی که رزرو را انجام داده، اتاق رزرو شده، تاریخ ورود و تاریخ خروج می‌باشد.
- تابع getStayDurationInDays() مدت زمان اقامت را حساب می‌کند.

کلاس Hotel

- این کلاس مدیریت اشیاء سیستم را بر عهده دارد و شامل لیست‌هایی از مشتریان، اتاق‌ها و رزروها می‌باشد.
- تابع getRooms(minCapacity) لیست اتاق‌هایی که ظرفیت آنها برابر یا بیشتر از مقدار مشخصی باشد را برمی‌گرداند.
- تابع getOldestCustomerName() نام مسن‌ترین مشتری هتل را برمی‌گرداند.
- تابع getCustomerPhonesByRoomNumber(roomNumber) شماره تلفن تمام مشتریانی که اتاق مشخصی را رزرو کرده‌اند را برمی‌گرداند.

دو تابع اول گفته شده در این کلاس، باید به سبک فانکشنال و بدون استفاده از چرخه‌های دستی نوشته شوند. به این منظور با امکانات stream در جاوا آشنا شوید و از عبارات lambda استفاده کنید. می‌توانید از این لینک جهت آشنایی با آنها استفاده کنید.

خواندن اطلاعات از فایل

داده‌های اولیه در قالب یک فایل JSON در اختیار شما قرار می‌گیرند. برای خواندن این فایل می‌توانید از کتابخانه Jackson استفاده کنید. ابتدا آخرین نسخه این کتابخانه را با استفاده از Maven و فایل pom.xml به پروژه خود اضافه کنید. توجه کنید که همه نسخه‌های کتابخانه‌ها را می‌توانید در Maven Repository مشاهده کنید. برای اطلاعات بیشتر در مورد این کتابخانه و نحوه استفاده از آن، می‌توانید به این لینک مراجعه کنید.
یک فایل به نام data.json را در کنار فایل‌های برنامه خود قرار دهید و در شروع برنامه، محتوای آن را خوانده و در کلاس‌های پیاده‌سازی شده بریزید. یک نمونه از Schema این فایل JSON در زیر آورده شده است:

```
{
  "customers": [
    {
      "ssn": 12345,
      "name": "Example Name",
      "phone": "09101231212",
      "age": 18
    },
    {
      ...
    }
  ],
  "rooms": [
    {
      "id": 1,
      "capacity": 3
    },
    {
      ...
    }
  ],
  "bookings": [
    {
      "id": 1,
      "room_id": 1,
      "customer_id": 1,
      "check_in": "2024-01-01 08:00:00",
      "check_out": "2024-01-02 16:00:00"
    },
    {
      ...
    }
  ]
}
```

نوشتن اطلاعات به فایل

در این قسمت، یک تابع به کلاس هتل به نام logState() اضافه کنید که با فراخوانی آن، استرینگ JSON وضعیت هتل با Schema زیر ریترن می‌شود. سپس در Main برنامه پس از خواندن اطلاعات از فایل ورودی و وارد کردن آنها به کلاس‌ها، تابع جدید را فراخوانی کنید و خروجی آن را به فایل state.json بریزید.

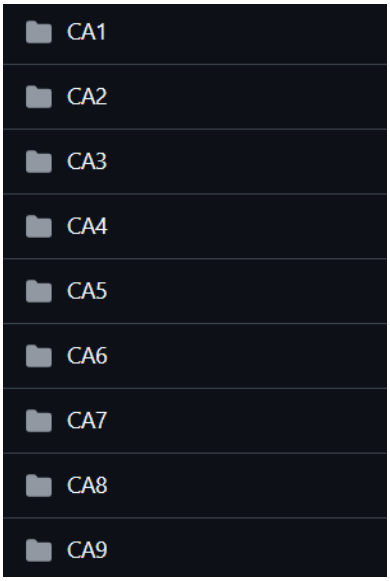
```
[
  {
    "room_id": 1,
    "capacity": 3,
    "bookings": [
      {
        "id": 1,
        "customer": {
          "ssn": 12345,
          "name": "Example Name",
          "phone": "09101231212",
          "age": 18
        },
        "check_in": "2024-01-01 08:00:00",
        "check_out": "2024-01-02 16:00:00",
      },
      {
        ...
      }
    ]
  },
  {
    ...
  }
]
```

آزمون واحد

در این قسمت باید با استفاده از چارچوب JUnit 5 برای سناریوهای مختلف مانند مقداردهی نامعتبر (مثلا تاریخ خروج قبل از ورود باشد) یا بررسی عملکرد متدهای مختلف، آزمون واحد بنویسید. آزمون‌های شما باید ساختار مناسب Setup، Test و Teardown را رعایت کنند. برای آشنایی با رویکرد JUnit، از این لینک استفاده کنید.

افزودن پروژه به گیت‌هاب

ابتدا در GitHub یک مخزن خصوصی (Private) به نام Internet-Engineering-Course-Projects ایجاد کنید. سپس کاربر IE-S04 را به پروژه خود اضافه کنید. تمامی تغییرات خود را به گیت اضافه کنید و در نهایت، در مخزن خود بارگذاری کنید. ساختار مخزنی که می‌سازید به این صورت باشد که یک مخزن کلی برای فازهای مختلف پروژه ایجاد کنید و برای هر فاز یک دایرکتوری جداگانه اختصاص دهید. به طور مثال:



Best Practice-ها

این بخش از صورت پروژه، به معرفی برخی از Best Practice-ها در حوزه مهندسی نرم افزار می‌پردازد. در هر فاز از پروژه‌های این درس، اصولی بر اساس ماهیت پروژه بیان می‌شوند که رعایت آنها حائز اهمیت می‌باشد.

Git Commit

کامیت‌ها بخش جدایی ناپذیر از سیستم‌های Version Control مانند Git هستند و نقش مهمی در توسعه پروژه‌های نرم‌افزاری دارند. به عبارتی کامیت‌ها snapshot-هایی از Repository شما در زمان‌های خاص هستند که بر اساس یک واحد تغییر منطقی در کد می‌باشند. با گذشت زمان، کامیت‌ها باید داستانی از تاریخچه Repository شما و نحوه تبدیل آن به ورژن فعلی را بیان کنند. در ادامه به دو ویژگی مهم یک کامیت خوب می‌پردازیم:

- *هر کامیت باید Atomic باشد.* به این معنی که هر کامیت باید نماینده یک مجموعه تغییر باشد که کمترین سایز ممکن را دارد. هر کامیت یک کار ساده انجام می‌دهد که می‌تواند در یک جمله ساده خلاصه شود. توجه داشته باشید که مقدار تغییر کد مهم نیست. می‌تواند یک حرف یا صد هزار خط باشد، اما شما باید بتوانید تغییر را با یک جمله ساده و کوتاه توصیف کنید. همچنین این تغییر نیز باید کامل باشد.

- *هر کامیت باید message معناداری داشته باشد.* commit message-ها راهی برای ارتباط بین اعضای تیم هستند و نوشتن commit message معنادار می‌تواند در زمان پاسخگویی به بسیاری از "چرا؟"ها صرفه‌جویی کند. فرض کنید یک اشکال در برنامه وجود دارد که قبلا وجود نداشت؛ برای اینکه بفهمید چه چیزی باعث این مشکل شده است، خواندن commit message-ها می‌تواند بسیار مفید باشد. راه‌های زیادی برای نوشتن یک commit message خوب وجود دارد. **در طول پروژه‌های این درس از این استاندارد برای نوشتن commit message-های خود استفاده کنید.**

نکات پایانی

- این تمرین در گروه‌های حداکثر دو نفره انجام می‌شود و 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.
- حتما کاربر IE-S04 را به پروژه خود اضافه کنید.
- ساختار صحیح و تمیزی کد برنامه، بخشی از نمره این فاز پروژه شما خواهد بود. بنابراین در طراحی ساختار برنامه و همچنین خوانایی کد دقت به خرج دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده شباهت بین کدهای دو گروه، از نمره هر دو گروه مطابق سیاستی که در کلاس گفته شده است کسر خواهد شد.
- سوالات خود را تا حد ممکن در گروه 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 تمرین ارتباط برقرار کنید.