به نام خدا

تمرین سری دوم

مائده بادان فیروز - ۹۸۲۲۲۰۰۹

بهار ۱۴۰۲

مقدمه:

۱- این سری از تمرینات با هدف آشنایی با Maven و Gradle و نحوه‌ی خواندن اطلاعات با استفاده از JSON طراحی شده است.
۲- نحوه‌ی گرفتن اطلاعات از یک API مورد نیاز است که کد آن داده شده و ما صرفا باید کد را بررسی کرده و یاد بگیریم.

طراحی و پیاده سازی:

۱- در ابتدا در Github پروژه را fork کردم. برای clone کردن یک SSH Key ساختم و آن را وارد کردم. برای این امر باید Git Bash را هم نصب می‌کردم که به خوبی انجام شد. (سیستم عامل من ویندوز است).

برای استخراج اطلاعات از JSON نیاز بود که ابتدا اطلاعات دریافت شده از سایت که در یک String ذخیره می‌شدند به JSONObject تبدیل شوند تا بتوان از method های کتابخانه‌ی JSON برای استخراج اطلاعات از آن، استفاده کرد. پس برای آن یک متد به نام jsonParser نوشتم که JSON موجود در یک String را به JSONObject تبدیل می‌کند. برای این تبدیل نوع یک parser از نوع JSONParser ساختم. سپس با متد ()parse. JSON string گرفته شده را به JSONObject تبدیل کردم. هردوی JSONParser و ()parse. در کتابخانه‌ی Json-simple قرار دارند که آن را از سایت mvnrepository دریافت و به Gradle اضافه کردم. خروجی این تابع برای ورودی دو تابع getTemperature و getHumidity استفاده می‌شود.
در دو تابع getTemperature و getHumidity با استفاده از متد ()get. به شیء‌های داخل JSONObject دسترسی پیدا کردیم. متد get به گونه‌ای است که نوع خروجی‌اش را می‌توانیم خودمان تعیین کنیم. به عنوان مثال این بخش از کد:

```
(JSONObject) weatherJson.get("current"))
```

درweatherJson که یک JSONObject است به ما شیء current را می‌دهد که خودمان با نوشتن (JSONObject) تعیین کرده‌ایم به صورت یک JSONObject داده شود. بعد با اضافه کردن ("get("temp_c. به کد قبلی و به دست آوردن

(JSONObject) weatherJson.get("current")).get("temp_c") به مقدار عددی temp_c دست می‌یابیم که اینجا هم لازم است نوع را تعیین کنیم؛ که برای آن (double) را در ابتدای خط اضافه می‌کنیم. به این ترتیب به سادگی مقدار دما بر حسب درجه سانتی گراد به دست می‌آید.

برای رطوبت هم به همین ترتیب است تنها با این تفاوت که نمی‌شد مستقیما نوع خروجی get را int تعریف کرد. از ارور خود برنامه ابتدا نوع را به long تبدیل کردم و سپس برای کمتر حافظه گرفتن به int تبدیل کردم.

در تابع main هم که با چند خط کد ابتدایی و ساده اسم شهر گرفته شده و به تابع گرفتن وضعیت آب و هوا داده شده است. بعد اطلاعات دریافت شده از API با تابع فرعی از JSON string به JSONObject تبدیل شده و به توابعی که دما و رطوبت را می‌دهند، داده شده و خروجی‌شان چاپ شده است.

۲- حال به تابع getWeatherData می‌رویم. اول آدرسی که باید در اینترنت سرچ شود ساخته می‌شود که یک شیء از نوع URL است؛ که برای آن کتابخانه‌ی net.URL اضافه شده. بخشی از آن همان apiKey است که با وارد شدن در سایت مورد نظر و ساخت اکانت دریافت و در تابع main در متغیر apiKey ذخیره کردیم. در خط بعدی با استفاده از تابع openConnection(). یک شیء از نوع HttpURLConnection ساخته شده که به خاطر آن هم کتابخانه‌ی net.HttpURLConnection اضافه شده. این کلاس فقط با پروتکل‌های HTTP کار می‌کند. با استفاده ار این کلاس می‌توانیم اطلاعاتی مانند استاتوس کد، هدر اینفورمیشن و ... را از هر HTTP URL دلخواهی بگیریم. در خط بعد گفته شده connectionی که ساختیم نوع درخواستی که به URL می‌دهد "گرفتن" است. چون وقتی به URL درخواست فرستاده‌ایم باید نوع این درخواست هم مشخص شود. تابع setRequestMethod(). هم در کتابخانه‌ی net.HttpURLConnection است.

در خط بعدی با استفاده از InputStreamReader رشته‌ای را که با connection.getInputStream() دریافت کرده و به صورت byte است به رشته‌ای از کاراکترها تبدیل می‌کند. برای آن کتابخانه‌ی io.InputStreamReader گرفته شده. این رشته‌ی ساخته شده، در یک BufferedReader به نام reader نگهداری می‌شود.

BufferedReader بخشی در حافظه برای نگهداری داده‌ای است، که قرار است بعدا جای دیگری برود.

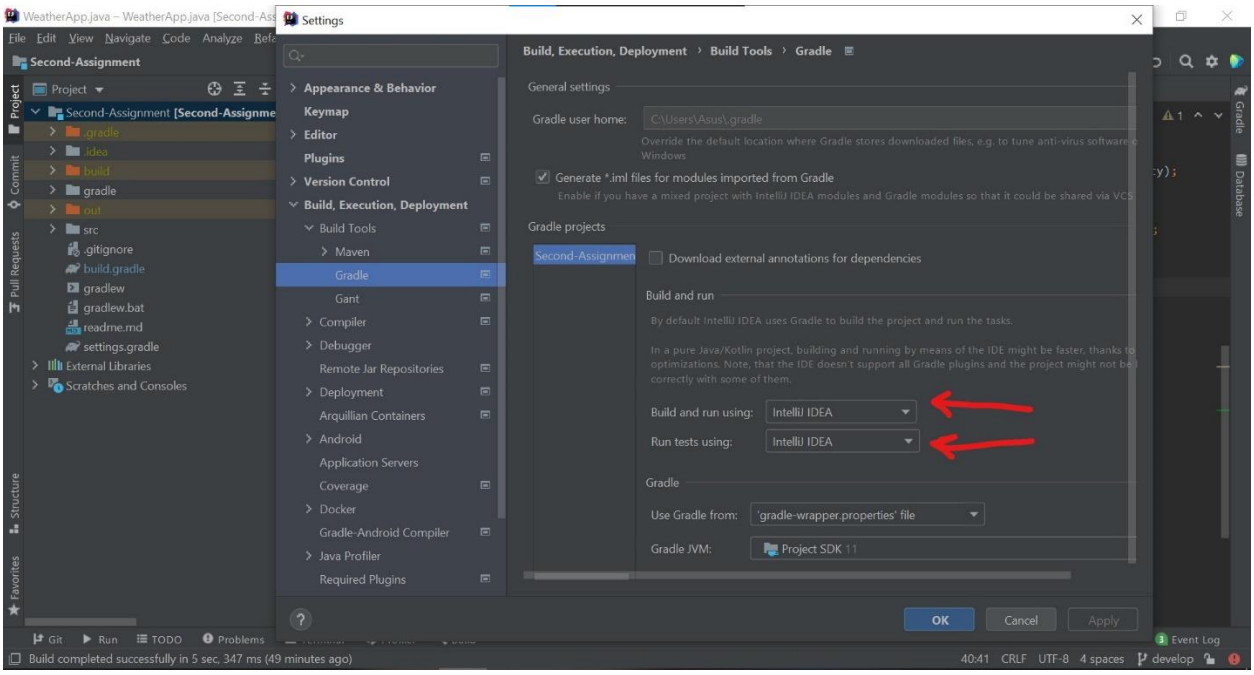
در خط بعدی یک StringBuilder به نام stringBuilder ساخته می‌شود که بعدا خط به خط اطلاعات را به خود اضافه می‌کند و String نهایی که تمام اطلاعات را در خود دارد، در آن نگهداری خواهد شد. StringBuilder یک کلاس است که این قابلیت را دارد که یک رشته کاراکتر پشت سر هم را به نوعی که می‌خواهیم تبدیل کند. مثلا می‌توان بسته به نیاز، در انتها رشته کاراکتر را به عدد یا به String تبدیل کرد. اینجا در کد ما، در انتها با تابع ()toString. به String تبدیل می‌شود.

در خط بعد یک متغیر String به نام line ساخته می‌شود که در حلقه‌ی while هر خط از اطلاعات هواشناسی، موقع خوانده شدن توسط reader در این متغیر ریخته می‌شود و بعد به stringBuilder اضافه می‌شود. با توجه به شرط قید شده در حلقه‌ی while تا وقتی که اطلاعاتی برای خواندن موجود باشد و به انتها نرسیده باشیم که reader خالی باشد، حلقه ادامه می‌یابد. متد ()readLine. که در شرط while نوشته شده یک متد از کلاس Console است که نه پارامتری می‌گیرد و نه چیزی برمی‌گرداند، تنها یک خط از صفحه‌ی کنسول را می‌خواند. اینجا به reader وصل شده و ما خودمان خطی که می‌خواند را در line می‌ریزیم.

بعد از حلقه‌ی while و خواندن تمام اطلاعات، با استفاده از متد ()close. جریان reader بسته می‌شود. سپس نتیجه که در stringBuilder ذخیره شده به String تبدیل شده و به عنوان خروجی تابع getWeatherData، return می‌شود. در کل برای ErrorHandlig کل این بخش کد در قسمت try از یک try catch گذاشته شده است. در بخش catch برای Exception نام e قرار داده شده و در بدنه‌ی catch با استفاده از متد ()printStackTrace. ارور به همراه شماره‌ی خط کد دارای ارور، و یا نام تابعی که در آن قرار دارد چاپ می‌شود. در این حالت چون برنامه به ارور برخورد کرده و اطلاعاتی نخوانده و چیزی برای ارائه به عنوان خروجی نداریم null را return می‌کنیم.

سنجش و ارزیابی:

۱- در ابتدا Gradle مشکل داشت که با سرچ در اینترنت و اعمال این تغییرات درست شد:



۲- در تابع geHumidity برای تبدیل نوع به int ارور می‌داد و می‌گفت که باید long باشد که برای رفع آن ابتدا نوع را به long تبدیل کردم و بعد به int.

نتیجه‌گیری:

۱- JSON یک فرمت تبادل داده است. یک JSONObject یک مجموعه‌ی غیر مرتب از جفت key و value هاست. Keyها حتما باید Stringهایی ناتهی باشند. Valueها می‌توانند JSONArray ,Array, Boolean, Number, String و یا حتی JSONObject باشند.
برای دستیابی به این اطلاعات کافی است بسته به نوع اطلاعات از JSONArray, JSONObject و... استفاده کنیم تا هر قسمت از JSON را به قسمت‌های کوچکتر بشکنیم تا بتوانیم به مقادیر داخل آن‌ها دست یابیم. یا اینکه اگر آرایه و نوع‌های پیچیده‌تر نباشند به سادگی با تابع get(). به دستشان بیاوریم.