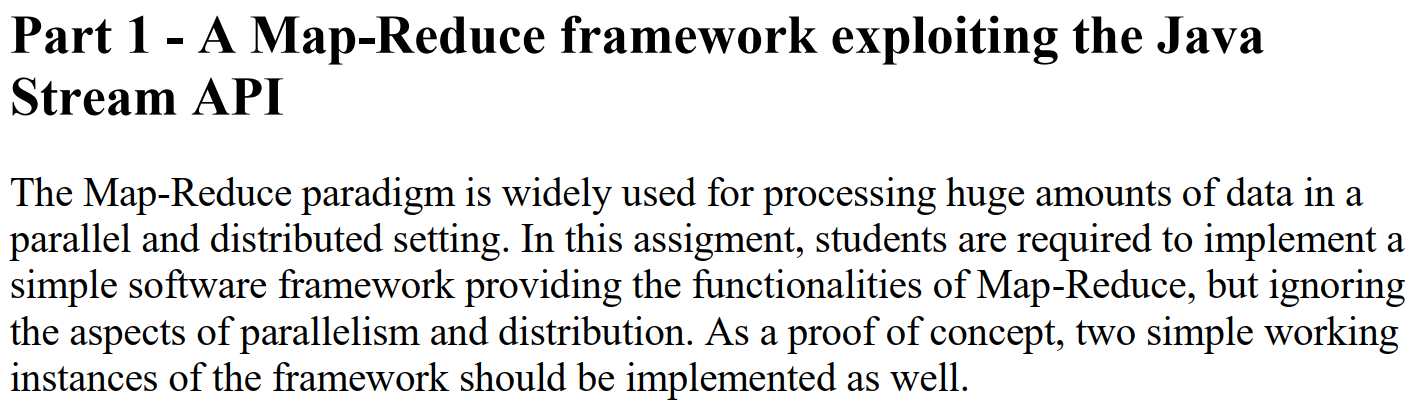
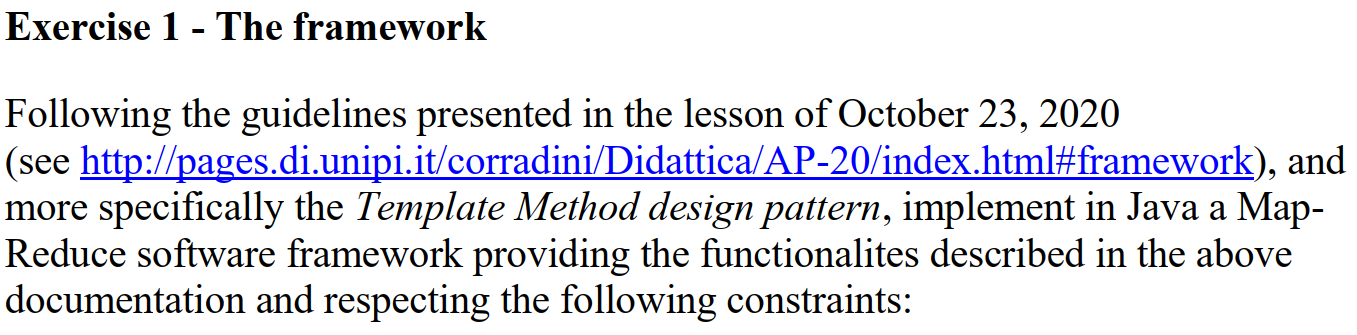
بسم الله الرحمن الرحیم

**حل تمرین جاوا استریم API**



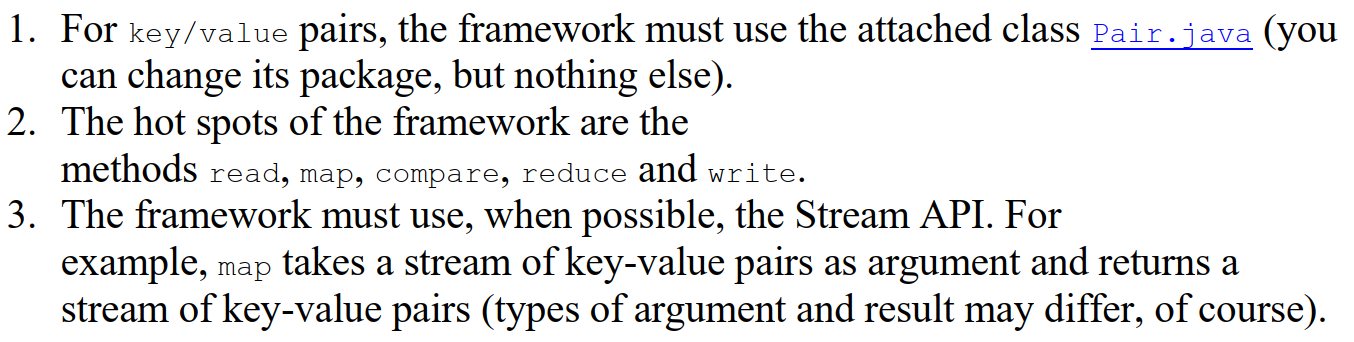
به طور کلی در زبان برنامه نویسی جاوا برای این که یک تابع یا عملگر خاص را بر روی تک‌تک المنت های یک iterable مانند لیست، کالکشن یا ... انجام دهیم، ابتدا باید آن iterable را استریم کرده و سپس به ترتیب توابع و عملگر های مورد نیاز روی آن اجرا کنیم. این کار به وسیله یک قابلیت جدید در جاوا 8 به بعد قابل انجام است به نام جاوا استریم API .

در تمرین Part-1 مجموعاً 2 تمرین اجباری و 1 تمرین اختیاری وجود دارد که هر سه این تمرین ها در ادامه یکدیگر بوده و هر یک کامل کننده تمرین قبلی است.

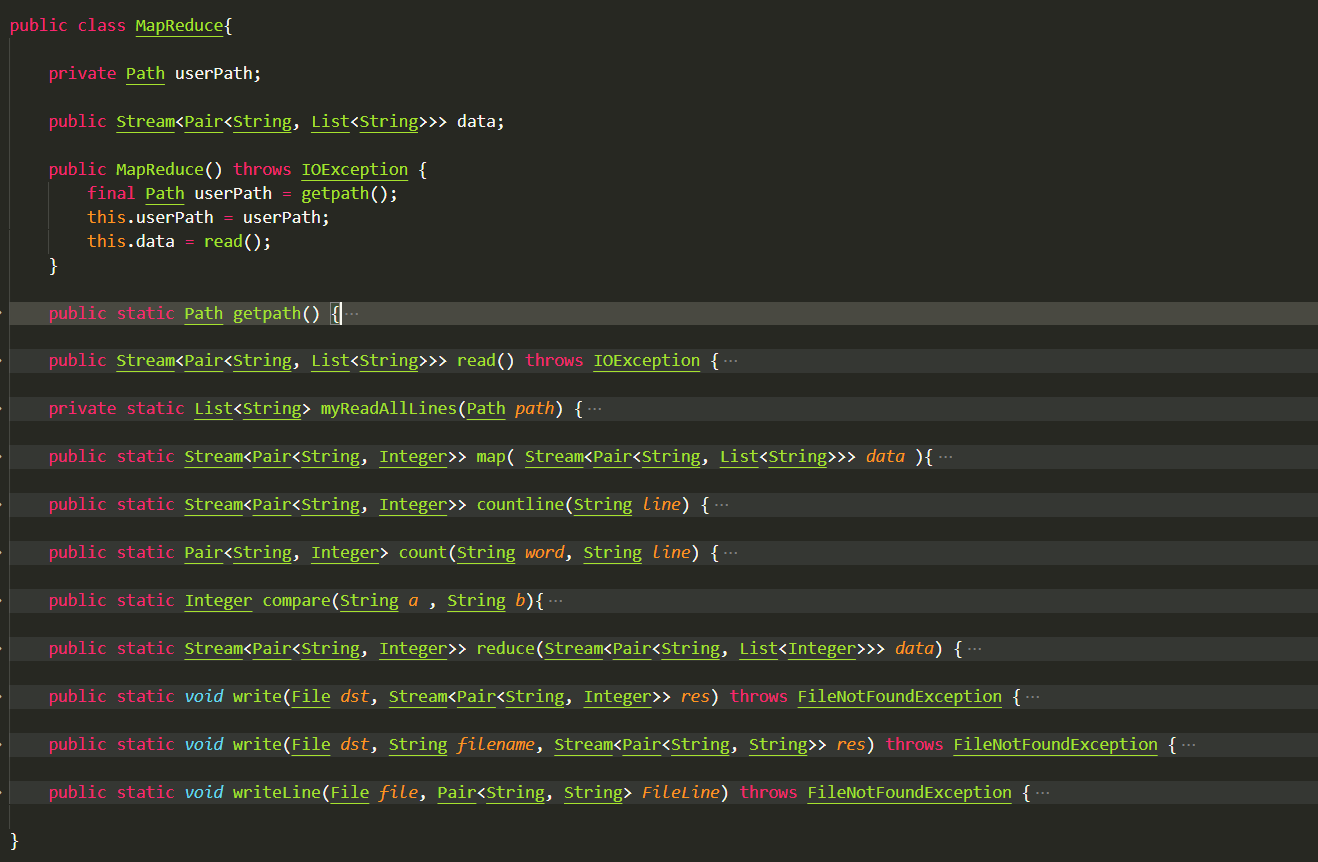
در اولین تمرین از ما طراحی یک فریم‌ورک خواسته شده که شامل یک سری توابع باشد. این فریم ورک باید بر اساس Template Method design pattern باشد. این نوع الگوی طراحی الگوریتم زمانی به کار می‌رود که مثلا قرار است برای چند بار، چند کد مختلف بزنیم که همه آنها با هم متفاوت اند و در عین حال در یکسری الگو ها مشترک اند. مثلا فرض کنید برای طراحی پوستر تبلیغاتی چند کاندید مختلف در انتخابات، با وجود اینکه همه این پوستر ها با هم متفاوت اند اما اما در یکسری الگو ها مشترک اند: مثلا احتمالا در بالای همه آنها اسم کاندید سپس عکس بزرگی از او و در نهایت توضیحاتی در مورد برنامه ها و ... وجود دارد.

بنابراین طبق این الگوی طراحی الگوریتم لازم نیست 10 بار مثلا برای 10 پوستر مختلف کد بزنیم؛ برای این کار یک بار Template یا قالب پوستر را طراحی میکنیم و سپس برای هر پوستر جدید فقط قسمت مختلفش را به آن می‌چسبایم.

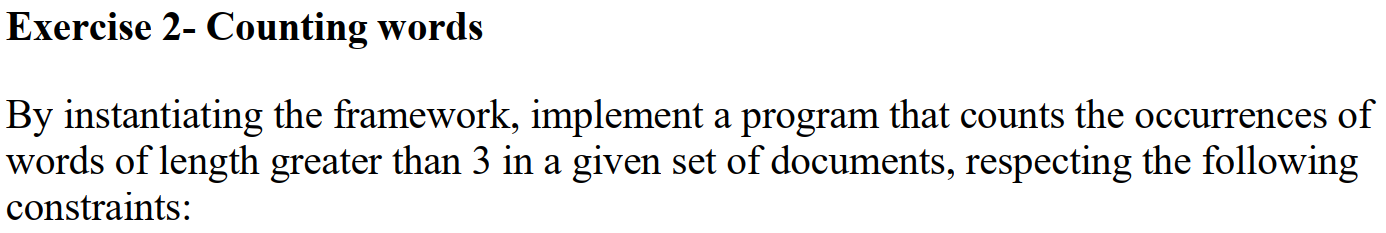
فریم ورک ما باید شامل توابع زیر باشد:

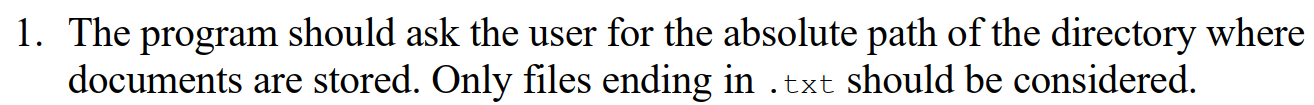
 در قدم اول Pair.java را دانلود می‌کنیم و نام پکیج آن را به نام دلخواه تغییر می‌دهیم سپس آن را در پوشه ای با همان نام پکیج می‌گذاریم. با این کار می‌توانیم آن را در کد های دیگرمان import کنیم.

سپس درون فایل فریم‌ورک توابع ذکر شده را تعریف می‌کنیم؛ در انتها هم سعی می‌کنیم در هر جایی که ممکن است از Stream API استفاده کنیم.

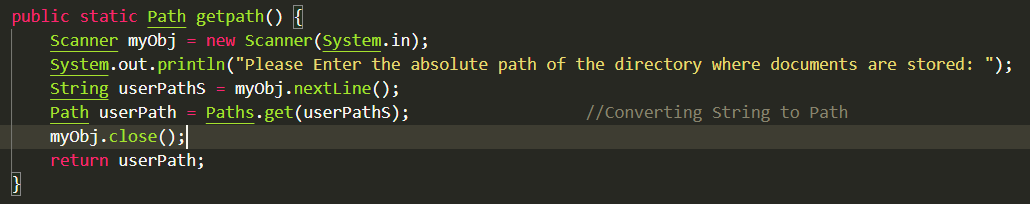
****فریم‌ورک ما به همراه توابع:

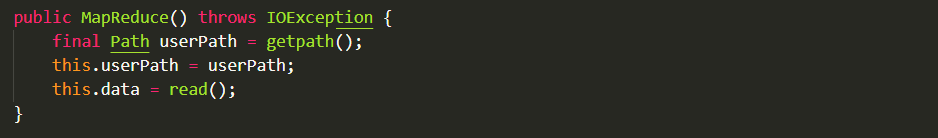
**تمرین دوم(ادامه تمرین اول):**

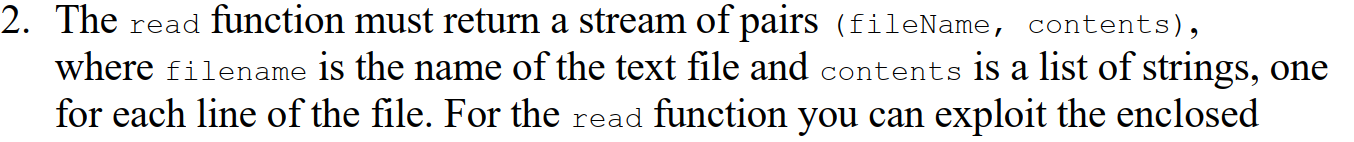
باید برنامه ای بنویسیم که با استفاده از فریم‌ورک تعداد تکرار های کلمات بزرگ‌تر از 3 حرف را به ما نشان دهد.

برای این کار باید 1- آدرس پوشه ای که فایل ها درون آن قرار دارند را از کاربر دریافت کنیم.

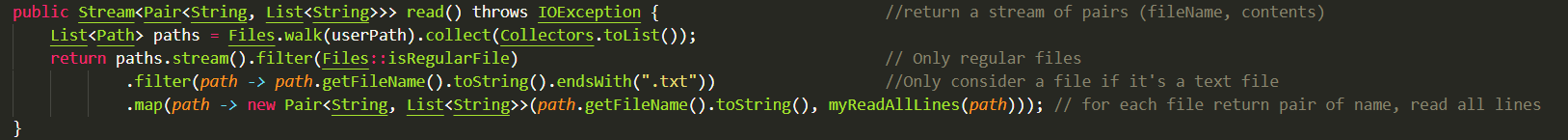
تابع getpath() به همین منظور نوشته شده است، این تابع به عنوان خروجی Path کاربر را به ما برمی‌گرداند:



همچنین نیاز داریم که این تابع به صورت خودکار در هر بار که یک instance جدید از کلاس می‌سازیم اجرا شود، بنابراین آن را درون متدی با همان اسم کلاس می‌گذاریم. (همانند متد \_\_init()\_\_ در پایتون):

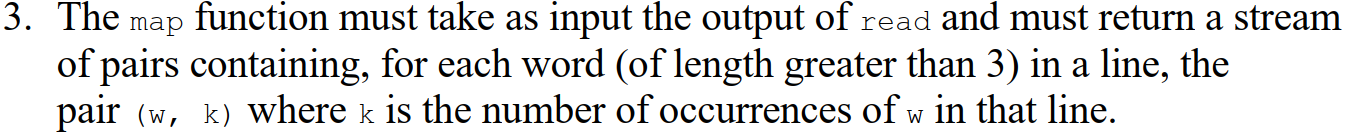
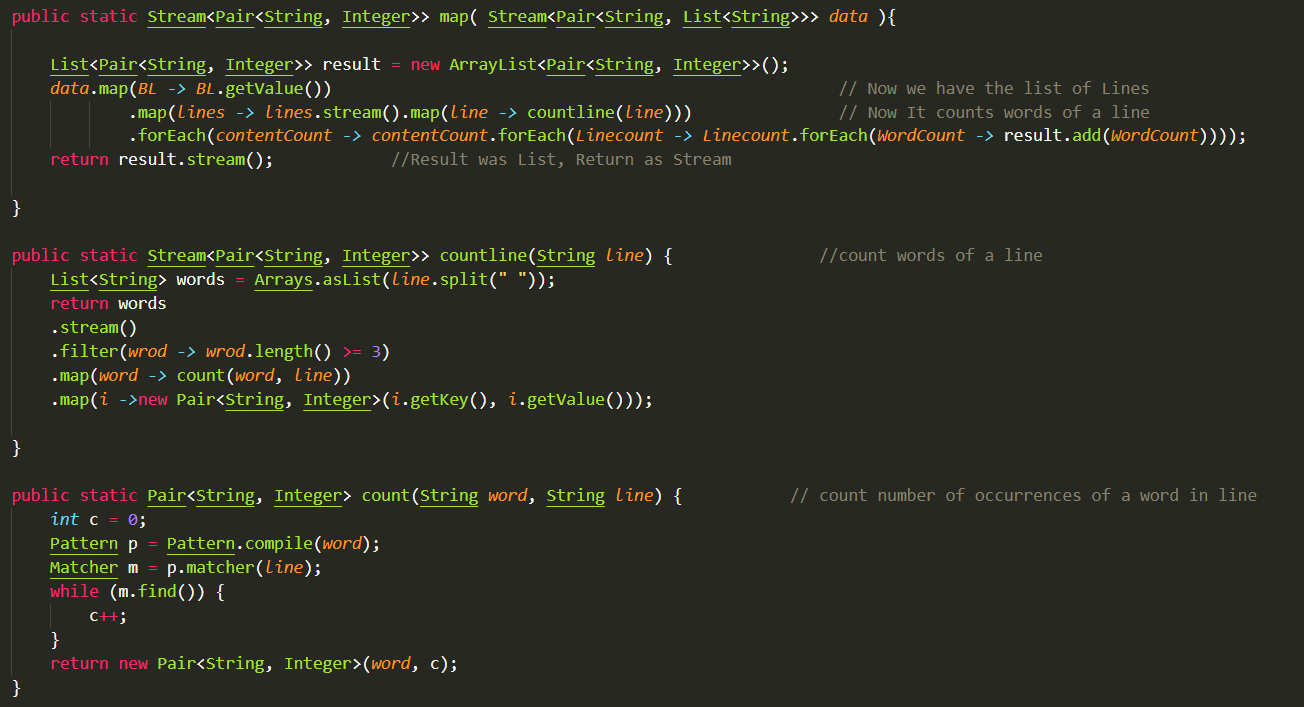
همچنین این متغیر را به وسیله کلمه کلیدی this متغیری از همین کلاس میکنیم تا در تمام توابع قابل استفاده باشد.

2- تابع read باید با استفاده از آدرس پوشه کاربر تمام اطلاعات فایل های txt را خارج کرده و به صورت استریمی از String(filename) و List<String>(content) نمایش دهد.



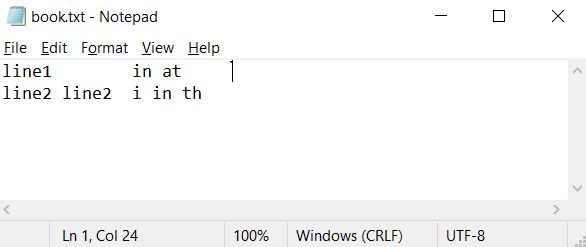
برای این کار با استفاده از فایل reader.java متد read را استخراج می‌کنیم که ابتدا تمام فایل های دایرکتوری را استریم می‌کند سپس به ترتیب فایل های غیر regular و txt را فیلتر کرده و در نهایت کانتنت فایل را به صورت pair خواسته شده بر می‌گرداند.

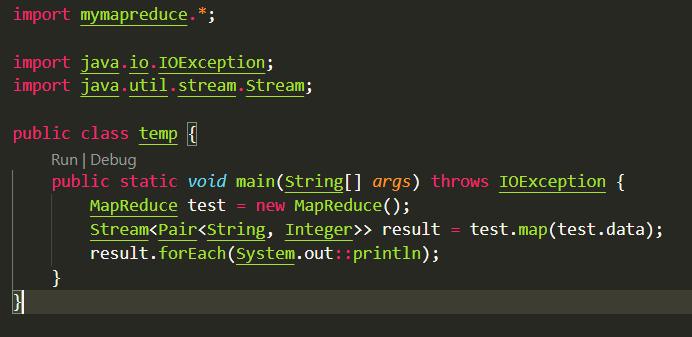
این تابع از متد myReadAllLines برای خواندن محتویات یک فایل txt استفاده می‌کند. خروجی این تابع به صورت لیستی از استرینگ خطوط می‌باشد.

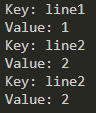
قسمت اصلی سوال در واقع از اینجا شروع می‌شود باید تابعی به نام map تعریف کنیم که به عنوان ورودی، خروجی تابع read را دریافت کند و به عنوان خروجی لیستی از Pair های کلمه و تعداد تکرارش در آن خط را برگرداند.

بدین منظور تابع map از تابع برای هر خط از تابع countline استفاده می‌کند. تابع countline هم برای مشخص کردن تعداد تکرار هر کلمه در خط، تابع count برای هر کلمه در خط invoke می‌کند. برای درک بهتر این 3 تابع بهتر است از آخرین تابع یعنی count شروع کنیم. این تابع به عنوان ورودی یک کلمه و یک خط را دریافت کرده و تعداد تکرار آن کلمه در آن خط را به صورت جفت آن کلمه و تعداد تکرار بر می‌گرداند. تابع countline یک خط را به عنوان ورودی گرفته و برای تک‌تک کلمات آن خط تابع count را invoke می‌کند. خروجی این تابع به صورت استریمی از جفت کلمه و تعداد تکرار خواهد بود و در نهایت تابع map برای هر خط موجود درون فایل ها تابع countline را invoke می‌کند. خروجی این تابع همانطور که خواسته شده بود به صورت استریمی از جفت کلمه و تعداد تکرار است.

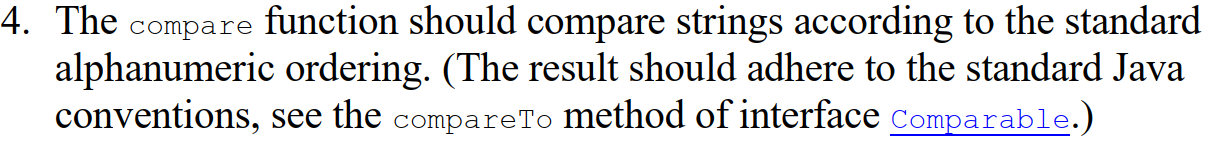
برای امتحان کردن تابع map یکی یک فایل ساده .txt با دو خط مینویسم:

برای بررسی نتیجه یک اینستنس از تابع به نام تست در فایل جداگانه تعریف می‌کنیم که پکیج به آن ایمپورت شده:

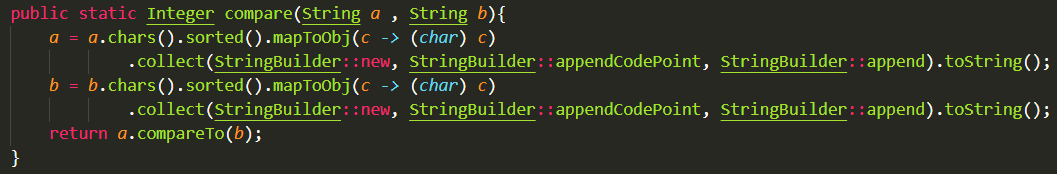
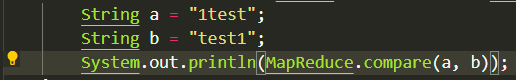
مشاهده می‌کنید که با استفاده از template design methode pattern چقدر کد کوتاه شده است.



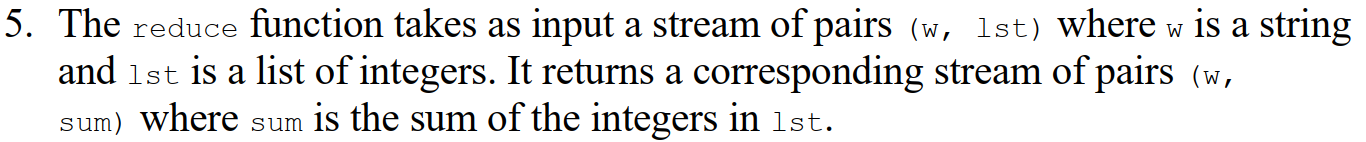
نتیجه به صورت روبرو خواهد بود که مطابق انتظار ماست:

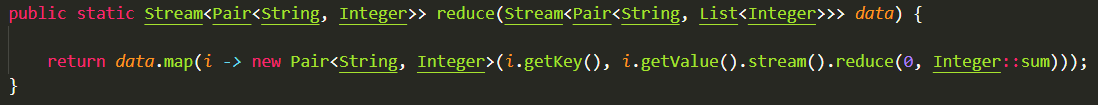


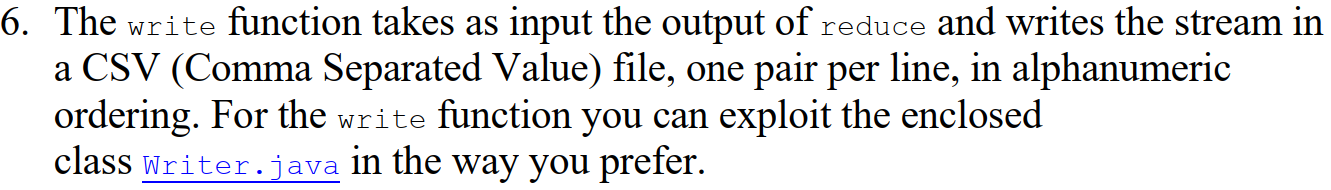
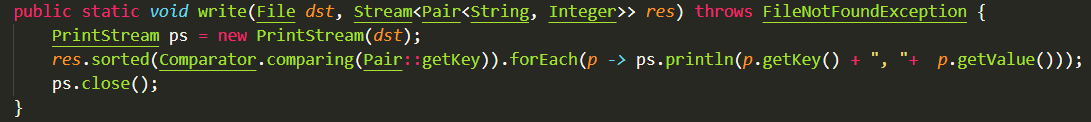
این تمرین هم البته تمرین مبهمی هست اما اون چیزی که واضح هست اینه که باید به عنوان ورودی دو تا String بگیره و اونها رو با هم مقایسه کنه. این مقایسه باید با دو شرط انجام بشه اول اینکه باید بنا به ترتیب alphanumeric باشه و دوم اینکه نتایج پایبند کانونشن های جاوا باشند. برای اینکار تابع compare به صورت زیر تعریف شده است:



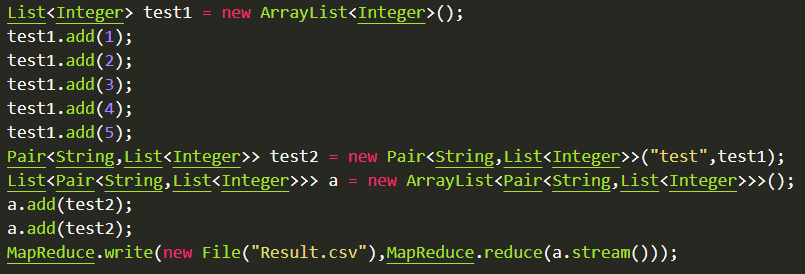
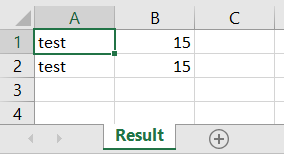
همانطور که مشاهده می‌کنید نتیجه به صورت خواهد بود. یعنی ترتیب آلفانومریک a و b تفاوتی ندارند.

تمرین شماره 5 هم نسبتاً تمرین ساده تری محسوب می‌شود و صرفاً برای آشنایی با reduce در Stream API است. تابع خواسته شده استریمی از جفت های استرینگ و لیست اینتجر دریافت می‌کند و به عنوان خروجی لیست اینتجر را به مجموع لیست کاهش می‌دهد.



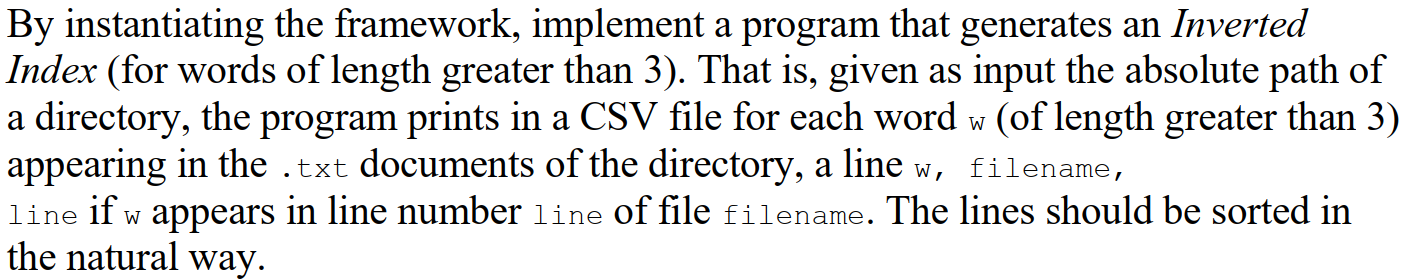
این تمرین هم در ادامه تمرین 5 بوده و نتیجه تمرین 5 را به ترتیب آلفانومریک و به صورت فایل csv ذخیره می‌کند.

نتیجه تمرین 5 و 6 :

در این کد هم تا خط آخر، لیستی از جفت استرینگ و لیست عدد ایجاد کردیم و در خط آخر هم با فراخوانی دو تابع reduce و write نتیجه را بررسی می‌کنیم:

که مطابق انتظار مجموع لیست به درستی در فایل csv ذخیره شد .

تمرین اختیاری سوم:

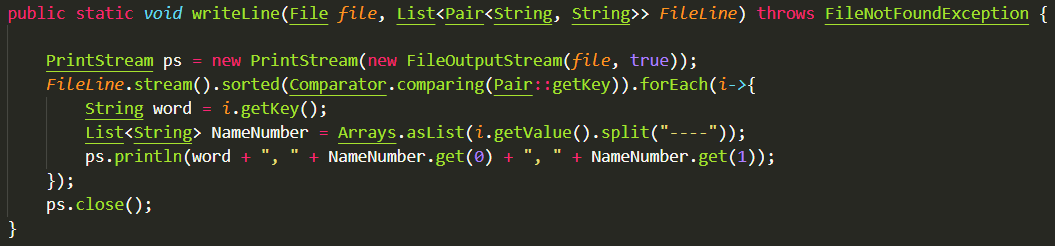
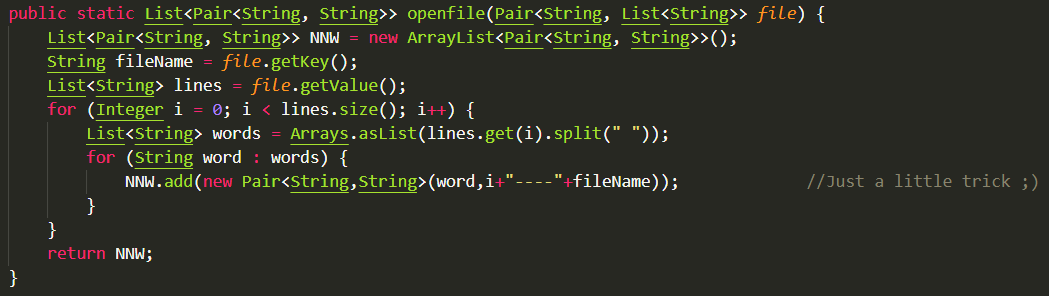
 البته در متن سوال ابهام هایی وجود دارد اما در مجموع به نظر می‌رسد که سوال، همانند تمرین های بالا بعد گرفتن آدرس یک دایرکتوری از کاربر و خواندن فایل های txt آن به ازای هر کلمه درون متن یک خط در فایل csv شامل همان کلمه، نام فایل و شماره خط آن کلمه ذخیره کند. البته خطوط باید بر اساس ترتیب آلفانومریک مرتب شوند.

این تمرین هم اندکی چالشی به حساب می‌آید اما با خرج کردن مقداری خلاقیت و حوصله حل شدنیست:

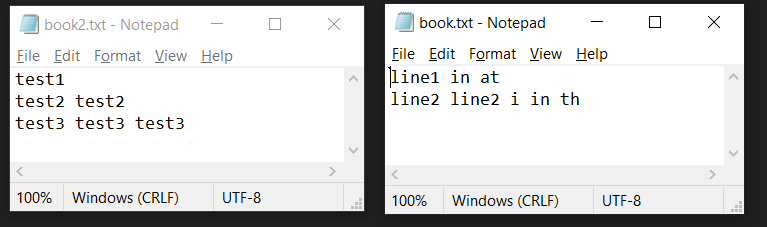
در واقع چالش اصلی این سوال مرتب کردن لاین های فایل csv می‌باشد زیرا جمع کردن همه آنها آن هم به همراه دو متغیر دیگر نام فایل و شماره خط مشکل است برای حل دو راهکار وجود دارد راهکار اول مقداری خلاقانه تر است و راهکار دوم که فقط آن را توضیح می‌دهیم اصولی تر:

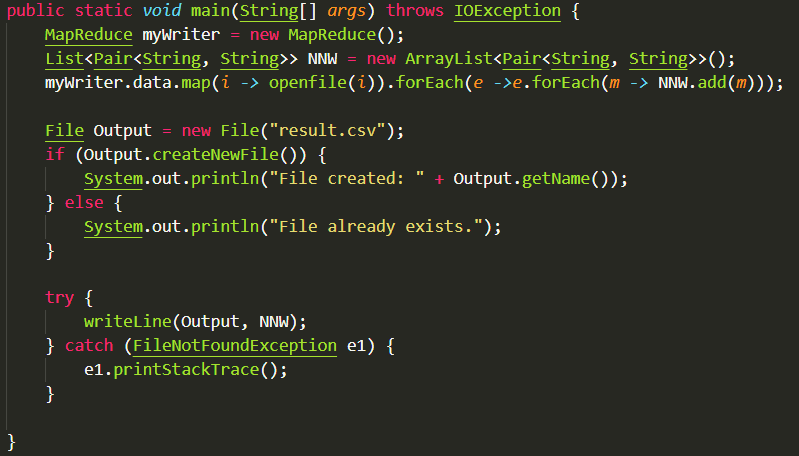
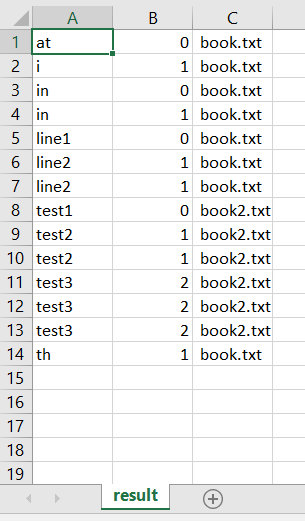
راه اول برای ذخیره سه متغیر کلمه، نام فایل و خط درون جفت این است که ما یک Pair<String,String> تعریف کرده و درون قسمت اول، کلمه را ذخیره کنیم تا بعدا جفت ها را بر اساس key مرتب کنیم؛ سپس استرینگ شماره خط و استرینگ نام فایل را به وسیله یک Seperator دلخواه مثلا "----" به هم وصل کرده و استرینگ حاصل را درون قسمت دوم جفت ذخیره کنیم. در صورت لزوم و بعد از مرتب کردن جفت ها این دو بخش را به وسیله همین seperator ، split می‌کنیم.

راه دوم برای این کار تعریف Pair<String,Pair<String,String>> است بدین صورت که درون قسمت اول، کلمه را ذخیره کنیم تا بعدا جفت ها را بر اساس key مرتب کنیم؛ سپس استرینگ شماره خط و استرینگ نام فایل را درون جفت قسمت دوم ذخیره می‌کنیم.

 برای این کار ابتدا تابع writeline را بررسی می‌کنیم که با گرفتن آدرس فایل csv مقصد که باید در آن اطلاعات را ذخیره کند و لیستی از جفت کلمه و (نام----شماره) طبق راه حل اول دریافت می‌کند و به وسیله split کردن قسمت دوم نام فایل و شماره خط را به دست می‌آورد و اطلاعات مربوط به هر کلمه را درون خط مربوط به خود ذخیره می‌کند. 

تابع openfile هم طراحی شده تا خروجی تابع read یعنی جفت استرینگ و لیست استرینگ را به عنوان ورودی تابع writeline آماده کند.

حال همه چیز برای امتحان کردن توابع آماده است، فقط قبل از شروع دو فایل txt کوچک با محتویات 2 یا 3 خطی آماده می‌کنیم تا تابع روی آنها ران شود.



بعد از جمع آوری دیتا ها ایجاد کردن فایل مقصد تابع writeline را کال می‌کنیم:

نتایج همانطور که انتظار داشتیم مرتب شده و درست قابل مشاهده است