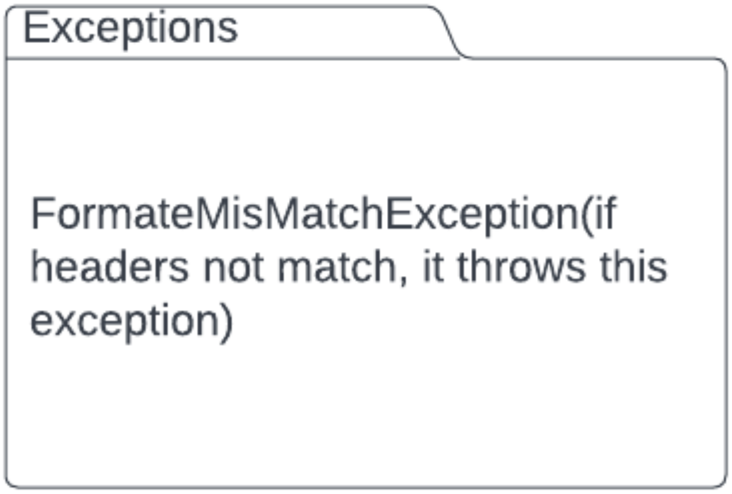
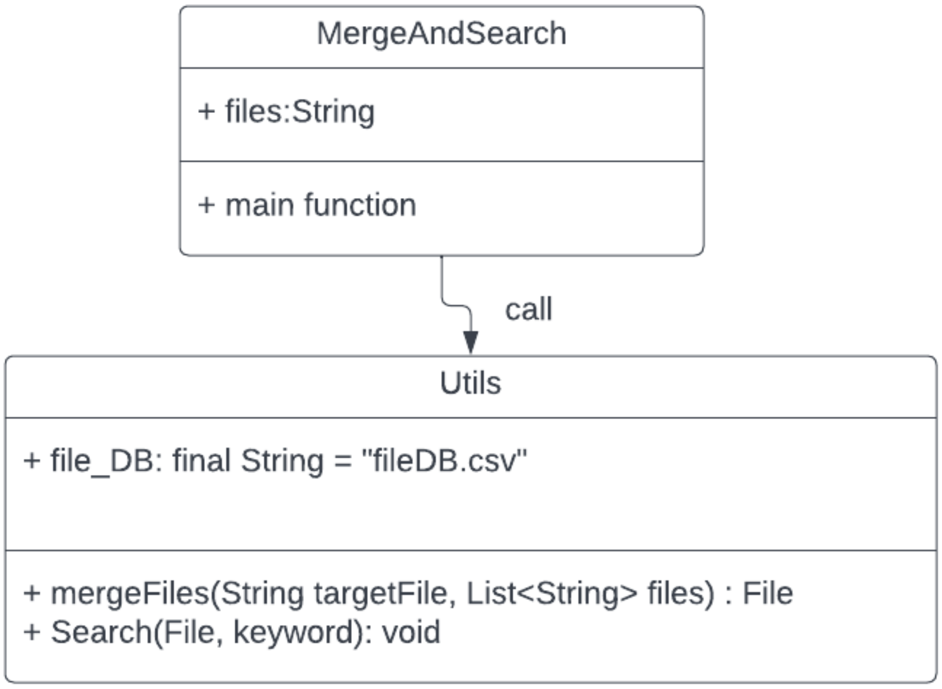
Assignment 2: Merge and Search files

Author: Xiaonan Liu

My assignment has two parts,

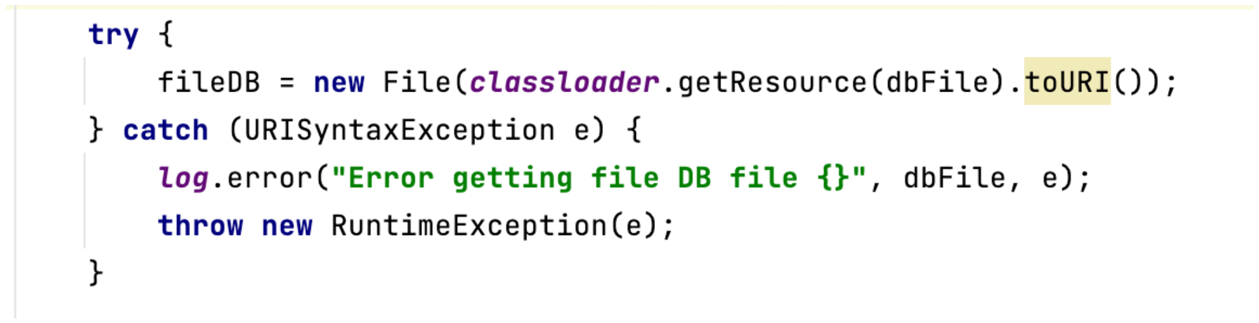

Exceptions
FormateMisMatchException(if headers not match, it throws this exception)


MergeAndSearch
+ files:String
+ main function
call
Utils
+ file_DB: final String = "fileDB.csv"
+ mergeFiles(String targetFile, List<String> files) : File
+ Search(File, keyword): void

In the exceptions package, I define an exception class named “FormatMisMatchException” when the headers are not match, it will throw this exception, and log the info

In the “MergeAndSearch”, it only contains the main function to run this application, the functions we need to use are defined in the “Utils”, it also contains a string we use to merge the files.

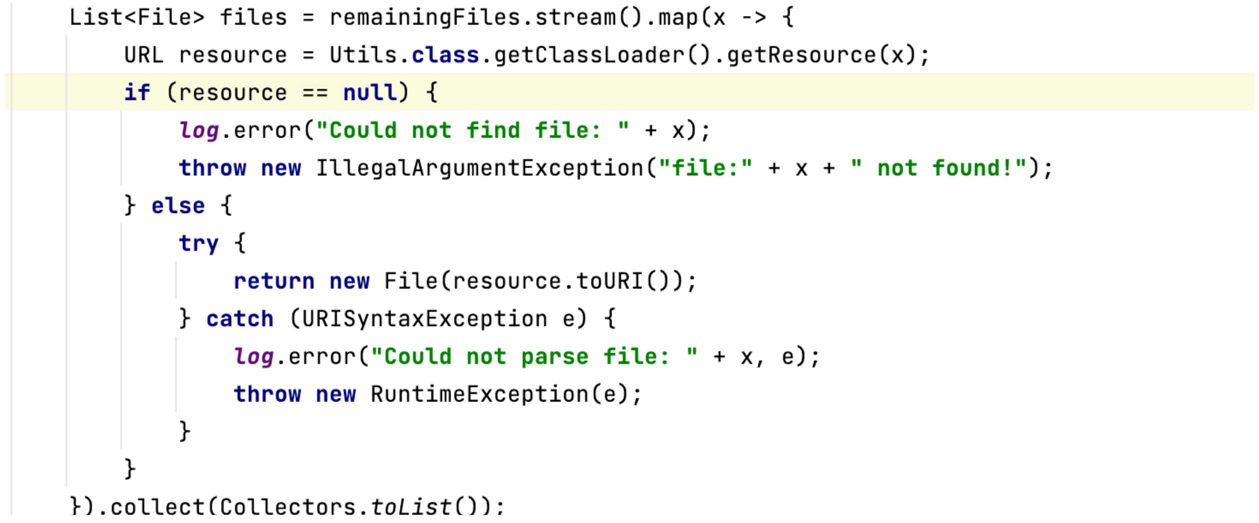
For merge the files, call “mergeFiles” function
It will first check whether the given files exists or not, if exist it will generate the absolute path for this file, but if not exist, it will catch the “ URISyntaxException”



```
try {
    fileDB = new File(classloader.getResource(dbFile).toURI());
} catch (URISyntaxException e) {
    log.error("Error getting file DB file {}", dbFile, e);
    throw new RuntimeException(e);
}
```

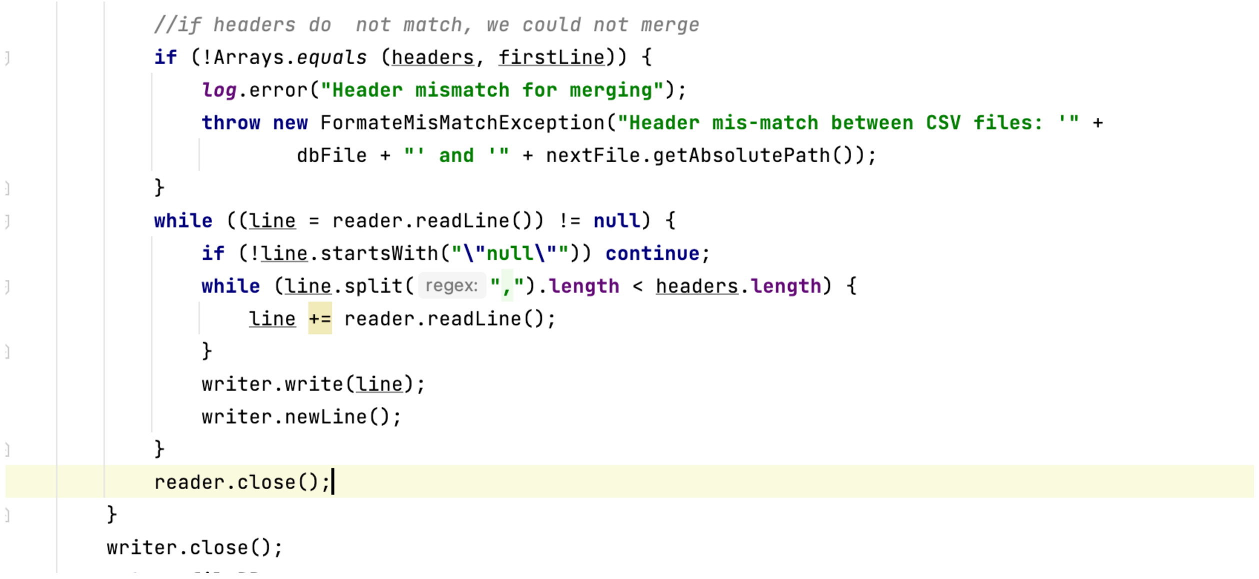
After this, the system will scan the headers of the target files, if exist, we merged according into these schemas, if not exist, we will exact from the files we need to merge.

Then we begin to merge the given files, first, we need to check does these files exist, if exist, generate the path for these files, “URISynataxException” could been generate, if the file could not parse, if the file is not exist, it throw the exception indicates that the file is not found .



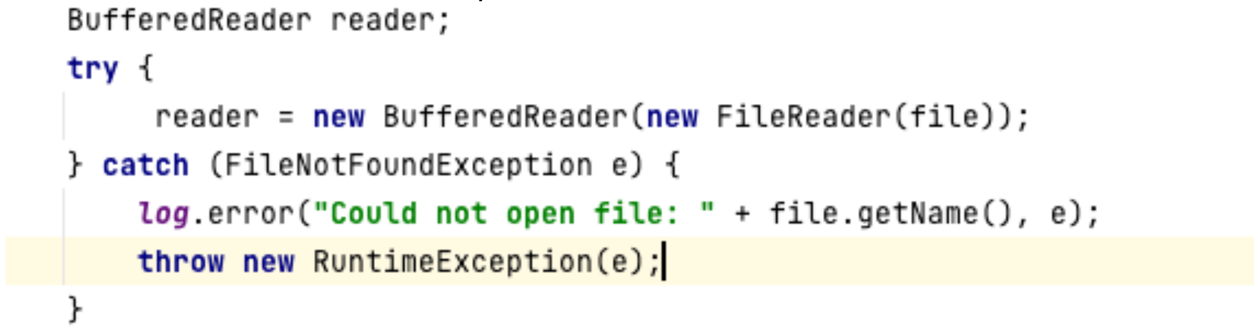
```
List<File> files = remainingFiles.stream().map(x -> {
    URL resource = Utils.class.getClassLoader().getResource(x);
    if (resource == null) {
        log.error("Could not find file: " + x);
        throw new IllegalArgumentException("file:" + x + " not found!");
    } else {
        try {
            return new File(resource.toURI());
        } catch (URISyntaxException e) {
            log.error("Could not parse file: " + x, e);
            throw new RuntimeException(e);
        }
    }
}).collect(Collectors.toList());
```

Next step: we will merge according to the headers
Firstly, if headers are null or empty now, we will try to scan from files we need to merge.
When we have the headers,we check it will the first line of the file we are now reading, if matched, we write in the new file, otherwise, the FormatMisMatchException” will be throwed.



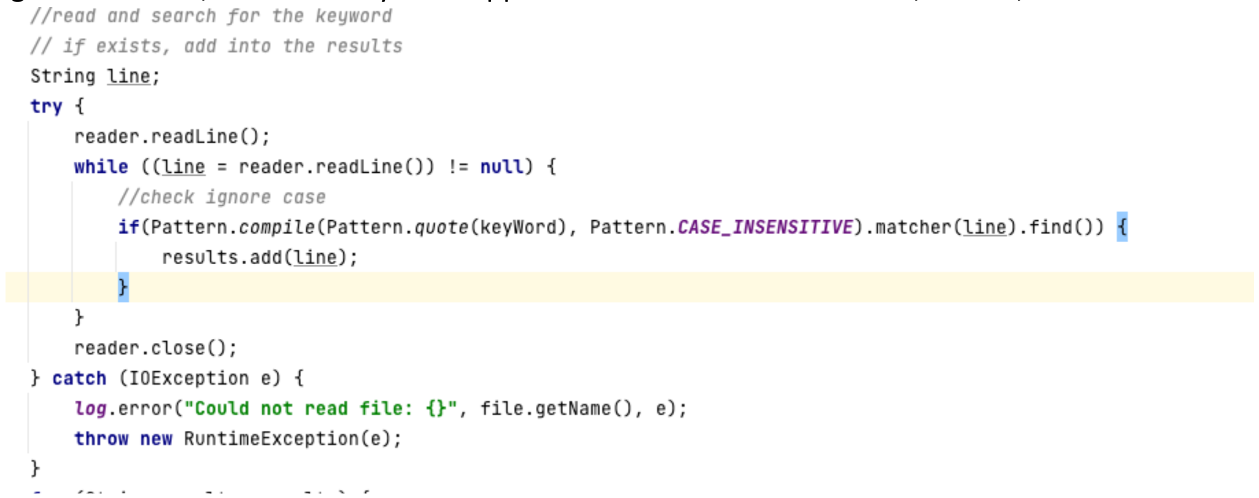
```
            //if headers do  not match, we could not merge
            if (!Arrays.equals (headers, firstLine)) {
                log.error("Header mismatch for merging");
                throw new FormateMisMatchException("Header mis-match between CSV files: '" +
                        dbFile + "' and '" + nextFile.getAbsolutePath());
            }
            while ((line = reader.readLine()) != null) {
                if (!line.startsWith("\"null\"")) continue;
                while (line.split( regex: ",").length < headers.length) {
                    line += reader.readLine();
                }
                writer.write(line);
                writer.newLine();
            }
            reader.close();
        }
        writer.close();
```

In the search function:
We use a container named results,
Firstly, we will read the file, if the file is not exist, we will throw the runtime exception, otherwise we will read the file line by line



```
BufferedReader reader;
try {
    reader = new BufferedReader(new FileReader(file));
} catch (FileNotFoundException e) {
    log.error("Could not open file: " + file.getName(), e);
    throw new RuntimeException(e);
}
```

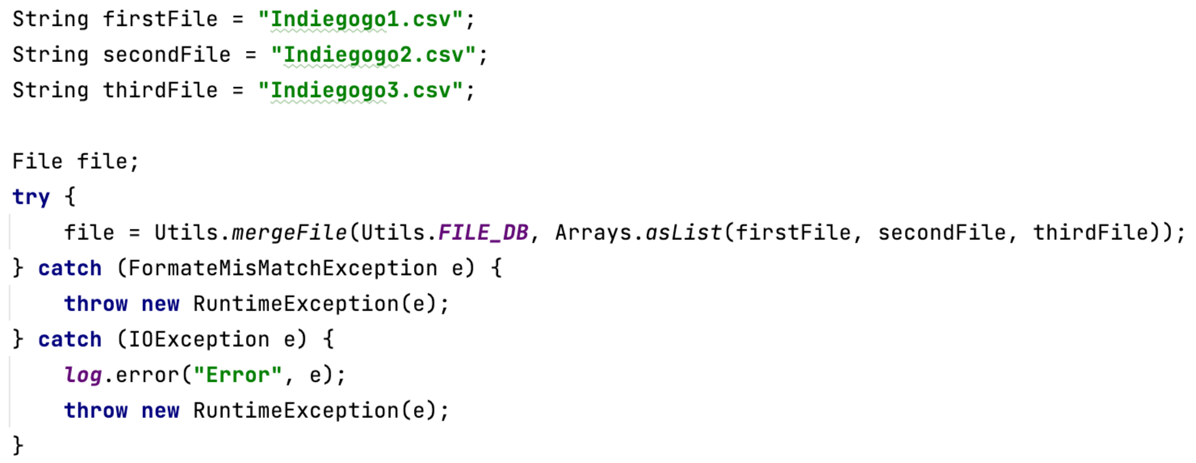
Secondly, we will check whether the line contains the given keyword or not, and the check will ignore the case, whether they have upper case or lower case or mixed, if finds, add to result



```
//read and search for the keyword
// if exists, add into the results
String line;
try {
    reader.readLine();
    while ((line = reader.readLine()) != null) {
        //check ignore case
        if(Pattern.compile(Pattern.quote(keyWord), Pattern.CASE_INSENSITIVE).matcher(line).find()) {
            results.add(line);
        }
    }
    reader.close();
} catch (IOException e) {
    log.error("Could not read file: {}", file.getName(), e);
    throw new RuntimeException(e);
}
```

Thirdly : print with the given format

In the main function:
The files we need to merge will be given in the main function, no matters how much are them, we could all add in the list, and merge them by functions, in this example , we only use three files, due to the test efficiency.

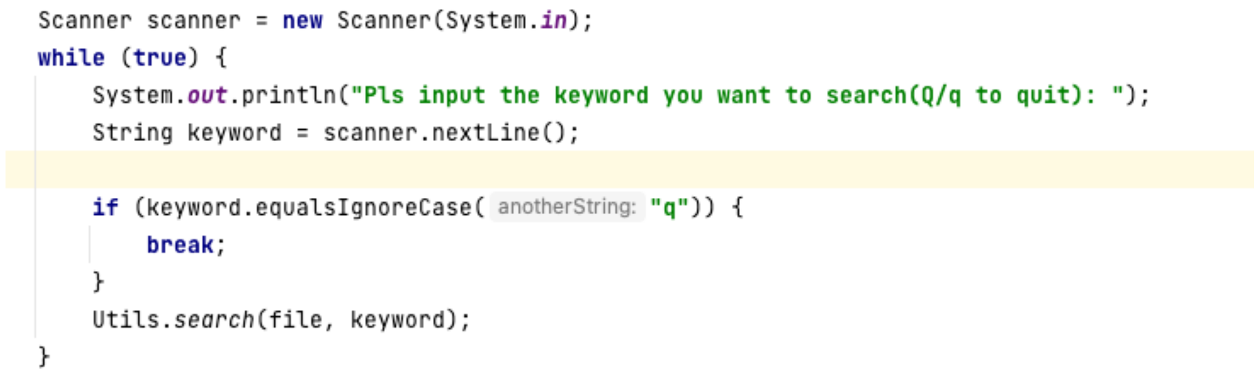


```java
String firstFile = "Indiegogo1.csv";
String secondFile = "Indiegogo2.csv";
String thirdFile = "Indiegogo3.csv";

File file;
try {
    file = Utils.mergeFile(Utils.FILE_DB, Arrays.asList(firstFile, secondFile, thirdFile));
} catch (FormateMisMatchException e) {
    throw new RuntimeException(e);
} catch (IOException e) {
    log.error("Error", e);
    throw new RuntimeException(e);
}
```

This program could handle more cases, if you want to merge more files.

To call the search function, it will scan the user's input as the keyword, only when user input "Q" or" q", it will quit this program, otherwise we will search use user's input, no matter the case what user input, the program will handle use regex in search function. And print the results in console.



```java
Scanner scanner = new Scanner(System.in);
while (true) {
    System.out.println("Pls input the keyword you want to search(Q/q to quit): ");
    String keyword = scanner.nextLine();

    if (keyword.equalsIgnoreCase( anotherString: "q")) {
        break;
    }
    Utils.search(file, keyword);
}
```

In the unit Test:
We check the mergeFiles function and search function , and all tests had past, you could run mvn clean tests to easy see the results.