Middle East Technical University | Department of Computer Engineering

CENG 443

Introduction to Object Oriented Programming Languages and Systems

Spring 2019-2020

Homework 3 - Covid.java

Due date: May 27, 2020, Wednesday, 23:59

1 Introduction

In this assignment, you will get familiar with Java 8 streams and lambda functions. You are asked to show some statistical results from the data provided to you with these features of Java. In these days, because the main topic of our lives is coronavirus, of course your data is also about coronavirus and its spread all around the world.

2 Problem Definition

First of all, it is asked to you to solve the given problem with using especially streams and lambda functions. Problems can be solved in different ways, however, the aim of this homework is to make you familiar with these concepts.

"covid19.csv" file was provided to you with homework text in ODTUclass. As it can be seen, in this file, there are 7 columns. The column headers are "Country Code", "Location", "Date", "Total Cases", "New Cases", "Total Deaths", "New Deaths" respectively. The data provided to you was eliminated version of the real data which is updated by the information that comes from many country all around the world (Original version can be found at this link https://github.com/owid/covid-19-data/tree/master/public/data). In the csv file, there are some dates for each location and some statistical information for each date of that location, as we can see in the news every day. The statistical results asked from you will be generated from this data, so it is really important to understand the data format in csv file.

With the homework files a Covid.java file was provided to you in ODTUclass. In this Covid class, as you can see declarations of 10 methods which need to be completed and one constructor and main method. Constructor reads the "covid19.csv" file line by line and write it to "rows" variable in this format [["ABW", "Aruba", "2020-03-13", "2", "2", "0", "0"], ["ABW", "Aruba", "2020-03-20", "4", "2", "0", "0"], ..., ["ZWE", "Zimbabwe", "2020-04-26", "31", "2", "4", "0"], ["ZWE", "Zimbabwe", "2020-04-27", "31", "0", "4", "0"]]. It is provided to you and you must not change the constructor and "rows" variable because only your methods will used while evaluating your codes not your constructor or "rows" variable. To be sure you are in the right way, please use our constructor and "rows" variable while testing your codes. However, you can add your own variables to the place which is shown in the Covid.java file or you can add your own methods and files. In total, from your file, everything will be taken for testing, except constructor method, main method and "rows" variable. Constructor method and "rows" variable will be provided by us as in the Covid.java file.

In the Covid.java file, the methods that you have implemented will be tested in a way that they will be called from main method and the exact output will be expected. In the implementation part, for each question, there is

a "Sample Method Call & Output" part, you can see the exact output that we expect. The main method which is provided in the Covid.java file will be replaced our test main and so, it does not matter what you have written for the main method in the file that you have uploaded.

3 Implementation

- In this section, you can see the methods that we expect you to implement. For each method in this section, you can see the empty versions or templates of them in the Covid.java file.
- Please use "System.out.printf()" method while printing to screen.
- The date format which is to print or to take as an argument will be the same as in the csv file.

3.1 void printOnlyCases(String location, String date) (6 Points)

- This method must print (Total Case Count - Total Death Count) for given specific location and date.
- Expected output is in the format of "Result:<<one space character>><<number>>".

3.1.1 Sample Method Call & Output

- Sample Method Call is: `printOnlyCases("Turkey", "2020-03-20")`
- Output(Screen Output) is: `Result:  355`

- Sample Method Call is: `printOnlyCases("United States", "2020-02-25")`
- Output(Screen Output) is: `Result:  53`

3.2 long getDateCount(String location) (6 Points)

- This method must return how many dates are there for the given location, in other words, it returns the row count for a location in csv file.
- It does not print anything on the screen, it only returns the count.
- Expected output is in the format of "<<row count>>".

3.2.1 Sample Method Call & Output

- Sample Method Call is: `getDateCount("Turkey")`
- Output(Returned Output) is: `45`

- Sample Method Call is: `getDateCount("Italy")`
- Output(Returned Output) is: `119`

3.3 int getCaseSum(String date) (6 Points)

- This method returns the sum of "New Cases" count which is collected from every location at a given date.
- If the date is not given for a location in the csv file, that location must be ignored for the sum.
- Expected returned output is in the format of <<sum>>.

3.3.1 Sample Method Call & Output

- Sample Method Call is: `getCaseSum("2020-03-05")`
- Output(Returned Output) is: `2239`

- Sample Method Call is: `getCaseSum("2020-04-05")`
- Output(Returned Output) is: `86484`

3.4 long getZeroRowsCount(String location) (6 Points)

- This method returns the row count of the location where rows have 0 values for these columns "Total Cases", "New Cases", "Total Deaths", "New Deaths".
- Expected returned output is in the format of <<row count>>.

3.4.1 Sample Method Call & Output

- Sample Method Call is: `getZeroRowsCount("Turkey")`
- Output(Returned Output) is: `0`

- Sample Method Call is: `getZeroRowsCount("Australia")`
- Output(Returned Output) is: `25`

3.5 double getAverageDeath(String location) (6 Points)

- This method returns the average of "New Deaths" count at every date of the given location.
- In order to parse your double value to return, one line code was provided to you in this function. Please use this function before returning your value.
- Expected returned output is in the format of <<average>>.

3.5.1 Sample Method Call & Output

- Sample Method Call is: `getAverageDeath("Turkey")`
- Output(Returned Output) is: `62.33`

- Sample Method Call is: `getAverageDeath("Italy")`
- Output(Returned Output) is: `223.9`

3.6 String getFirstDeathDayInFirstTenRows(String location) (10 Points)

- This method must return the first date that a death happened in that location in first ten rows of that location.
- Your code should not look the other rows if the expected row was found before the tenth row or not found. For example, if the expected row is the fifth row (In the fifth row, the first death happened, which we look for.), when your code see this is a matching row, it should finish the search and print, which means it should not look for the remaining rows. Otherwise, there will be penalty for this method.

- In the case at least a death occured in the first ten rows, expected output to return is in the format of "<<date(as in the csv format)>>".
- If no deaths occured in that location in the first ten rows, expected output to return is "Not Found".

3.6.1 Sample Method Call & Output

- Sample Method Call is: `getFirstDeathDayInFirstTenRows("Turkey")`
- Output(Returned Output) is: `"2020-03-19"`

- Sample Method Call is: `getFirstDeathDayInFirstTenRows("Italy")`
- Output(Returned Output) is: `"Not Found"`

3.7 String[ ] getDateCountOfAllLocations() (15 Points)

- This method returns the how many dates are there for all locations, in other words, it returns the row count for every location.
- To complete this method, you should use the method you have written before named "getDateCount(String location)".
- This method will be tested with different csv data to be sure that your code is working properly.
- It does not print anything on the screen, instead, it must return a string array where every string has an information of country code of a location and the row count of that location in the format of "<<country code>>:<<one space character>><<row count>>"

3.7.1 Sample Method Call & Output

- Sample Method Call is: `getDateCountOfAllLocations()`
- Output(Returned Output) is: `["ABW: 36", "AFG: 109", "AGO: 37", "AIA: 32", "ALB: 50", ..., "ZMB: 40", "ZWE: 38"]`

3.8 List<String> getLocationsFirstDeathDay() (15 Points)

- This method must find the locations and their date which the first death occured if at least a death occured in that location. If there is no death occured in that location, it ignores that location.
- This method will be tested with different csv data to be sure that your code is working properly.
- Tip: For a location, the day when "Total Deaths" equals to "New Deaths" is the first death day.
- It does not print anything on the screen, instead, it must return a list of strings where every string has an information of location and first death date in the format of "<<location name>>:<<one space character>><<date(as in the csv format)>>"

3.8.1 Sample Method Call & Output

- Sample Method Call is: `getLocationsFirstDeathDay()`
- Output(Returned Output) is: `["Aruba: 2020-04-16", "Afghanistan: 2020-03-24", "Angola: 2020-03-30", ..., "Zimbabwe: 2020-03-24"]`

3.9 String trimAndGetMax(String location, int trimCount) (15 Points)

- This method must firstly sort the rows of the given location by "New Cases" count in ascending order. Then, it trims rows up to trimCount from the head and from the end. After trimming, it returns the information about the day which has maximum number of New Deaths.
- If there are more than one max rows in the trimmed list of rows, in other words, if more than one rows have same number of New Deaths and it is the max death count in the rows, then in the ascending order it must return the top row (Tip: max() method gives the top row if the rows are in ascending order).
- Number of rows for a location will be always bigger than 2 × trimCount, so any erroneous input about trimCount will not be tested.
- The expected returned output format is "<<date(as in the csv format)>>:<<one space character>><<max number of New Deaths>>"

3.9.1 Sample Method Call & Output

- Sample Method Call is: `trimAndGetMax("Turkey", 5)`

```
[TUR, Turkey, 2020-03-12, 1, 1, 0, 0]
[TUR, Turkey, 2020-03-13, 2, 1, 0, 0]
[TUR, Turkey, 2020-03-16, 18, 16, 0, 0]
[TUR, Turkey, 2020-03-17, 47, 29, 0, 0]
[TUR, Turkey, 2020-03-18, 98, 51, 0, 0]
[TUR, Turkey, 2020-03-19, 191, 93, 1, 1]
[TUR, Turkey, 2020-03-20, 359, 168, 4, 3]
[TUR, Turkey, 2020-03-22, 947, 277, 21, 12]
[TUR, Turkey, 2020-03-23, 1236, 289, 30, 9]
[TUR, Turkey, 2020-03-24, 1529, 293, 37, 7]
[TUR, Turkey, 2020-03-21, 670, 311, 9, 5]
[TUR, Turkey, 2020-03-25, 1872, 343, 44, 7]
[TUR, Turkey, 2020-03-26, 2433, 561, 59, 15]
[TUR, Turkey, 2020-03-27, 3629, 1196, 75, 16]
[TUR, Turkey, 2020-03-31, 10827, 1610, 168, 37]
[TUR, Turkey, 2020-03-29, 7402, 1704, 108, 16]
[TUR, Turkey, 2020-03-30, 9217, 1815, 131, 23]
[TUR, Turkey, 2020-03-28, 5698, 2069, 92, 17]
[TUR, Turkey, 2020-04-02, 15679, 2148, 277, 63]
[TUR, Turkey, 2020-04-27, 110130, 2357, 2805, 99]
[TUR, Turkey, 2020-04-03, 18135, 2456, 356, 79]
[TUR, Turkey, 2020-04-01, 13531, 2704, 214, 46]
[TUR, Turkey, 2020-04-04, 20921, 2786, 425, 69]
[TUR, Turkey, 2020-04-26, 107773, 2861, 2706, 106]
[TUR, Turkey, 2020-04-05, 23934, 3013, 501, 76]
[TUR, Turkey, 2020-04-23, 98674, 3083, 2376, 117]
[TUR, Turkey, 2020-04-24, 101790, 3116, 2491, 115]
[TUR, Turkey, 2020-04-25, 104912, 3122, 2600, 109]
[TUR, Turkey, 2020-04-06, 27069, 3135, 574, 73]
[TUR, Turkey, 2020-04-07, 30217, 3148, 649, 75]
[TUR, Turkey, 2020-04-19, 82329, 3783, 1890, 121]
[TUR, Turkey, 2020-04-08, 34109, 3892, 725, 76]
[TUR, Turkey, 2020-04-20, 86306, 3977, 2017, 127]
[TUR, Turkey, 2020-04-10, 42282, 4056, 908, 96]
[TUR, Turkey, 2020-04-15, 65111, 4062, 1403, 107]
[TUR, Turkey, 2020-04-14, 61049, 4093, 1296, 98]
[TUR, Turkey, 2020-04-09, 38226, 4117, 812, 87]
[TUR, Turkey, 2020-04-16, 69392, 4281, 1518, 115]
[TUR, Turkey, 2020-04-18, 78546, 4353, 1769, 126]
[TUR, Turkey, 2020-04-22, 95591, 4611, 2259, 119]
[TUR, Turkey, 2020-04-21, 90980, 4674, 2140, 123]
[TUR, Turkey, 2020-04-11, 47029, 4747, 1006, 98]
[TUR, Turkey, 2020-04-13, 56956, 4789, 1198, 97]
[TUR, Turkey, 2020-04-17, 74193, 4801, 1643, 125]
[TUR, Turkey, 2020-04-12, 52167, 5138, 1101, 95]
```

The rows of "Turkey" in ascending order

→

```
[TUR, Turkey, 2020-03-19, 191, 93, 1, 1]
[TUR, Turkey, 2020-03-20, 359, 168, 4, 3]
[TUR, Turkey, 2020-03-22, 947, 277, 21, 12]
[TUR, Turkey, 2020-03-23, 1236, 289, 30, 9]
[TUR, Turkey, 2020-03-24, 1529, 293, 37, 7]
[TUR, Turkey, 2020-03-21, 670, 311, 9, 5]
[TUR, Turkey, 2020-03-25, 1872, 343, 44, 7]
[TUR, Turkey, 2020-03-26, 2433, 561, 59, 15]
[TUR, Turkey, 2020-03-27, 3629, 1196, 75, 16]
[TUR, Turkey, 2020-03-31, 10827, 1610, 168, 37]
[TUR, Turkey, 2020-03-29, 7402, 1704, 108, 16]
[TUR, Turkey, 2020-03-30, 9217, 1815, 131, 23]
[TUR, Turkey, 2020-03-28, 5698, 2069, 92, 17]
[TUR, Turkey, 2020-04-02, 15679, 2148, 277, 63]
[TUR, Turkey, 2020-04-27, 110130, 2357, 2805, 99]
[TUR, Turkey, 2020-04-03, 18135, 2456, 356, 79]
[TUR, Turkey, 2020-04-01, 13531, 2704, 214, 46]
[TUR, Turkey, 2020-04-04, 20921, 2786, 425, 69]
[TUR, Turkey, 2020-04-26, 107773, 2861, 2706, 106]
[TUR, Turkey, 2020-04-05, 23934, 3013, 501, 76]
[TUR, Turkey, 2020-04-23, 98674, 3083, 2376, 117]
[TUR, Turkey, 2020-04-24, 101790, 3116, 2491, 115]
[TUR, Turkey, 2020-04-25, 104912, 3122, 2600, 109]
[TUR, Turkey, 2020-04-06, 27069, 3135, 574, 73]
[TUR, Turkey, 2020-04-07, 30217, 3148, 649, 75]
[TUR, Turkey, 2020-04-19, 82329, 3783, 1890, 121]
[TUR, Turkey, 2020-04-08, 34109, 3892, 725, 76]
[TUR, Turkey, 2020-04-20, 86306, 3977, 2017, 127]
[TUR, Turkey, 2020-04-10, 42282, 4056, 908, 96]
[TUR, Turkey, 2020-04-15, 65111, 4062, 1403, 107]
[TUR, Turkey, 2020-04-14, 61049, 4093, 1296, 98]
[TUR, Turkey, 2020-04-09, 38226, 4117, 812, 87]
[TUR, Turkey, 2020-04-16, 69392, 4281, 1518, 115]
[TUR, Turkey, 2020-04-18, 78546, 4353, 1769, 126]
[TUR, Turkey, 2020-04-22, 95591, 4611, 2259, 119]
```

Same rows after trimming five rows from head and end

Then, it must find the row that has maximum New Deaths count and return(This is just an illustration to explain better, please do not print anything on the screen).

- Output(Returned Output) is: `"2020-04-20:  127"`

3.10 List<List<String>> getOnlyCaseUpDays(String location) (15 Points)

- This method returns the rows of the given location where the rows do not have 0 value for "New Cases" column. It also prints the number of rows which are returned.
- Expected output on the screen is in the format of "Result:<<one space character>><<row count>>".
- Expected returned output is like [["TUR", "Turkey", "2020-03-12", "1", "1", "0", "0"], ["TUR", "Turkey", "2020-03-13", "2", "1", "0", "0"], ...]. As it can be seen, putting the rows to a different List<List<String>>

variable is enough, you do not have to change the row information that you can get it from "private List<List<String>> rows" variable.

3.10.1 Sample Method Call & Output

- Sample Method Call is: `getOnlyCaseUpDays("Turkey")`
- Output(Screen Output) is: `Result: 45`
- Output(Returned Output) is: `[["TUR", "Turkey", "2020-03-12", "1", "1", "0", "0"], ["TUR", "Turkey", "2020-03-13", "2", "1", "0", "0", ..., ["TUR", "Turkey", "2020-04-27", "110130", "2357", "2805", "99"]]`

- Sample Method Call is: `getOnlyCaseUpDays("Aruba")`
- Output(Screen Output) is: `Result: 22`
- Output(Returned Output) is: `[["ABW", "Aruba", "2020-03-13", "2", "2", "0", "0"], ["ABW", "Aruba", "2020-03-20", "4", "2", "0", "0"], ..., ["ABW", "Aruba", "2020-04-23", "100", "3", "2", "0"]]`

4 Specifications

- **This is an individual assignment. Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the Internet. The violators will be punished according to the department regulations.**
- **Late Submission: As indicated in the syllabus, $5day^2$ for at most 3 days.**
- Your code must be written in Java.
- Efficiency is not important, however, it is expected that your programs will finish in a reasonable amount of time.
- Your homeworks will be evaluated with **black box testing**, however, in order to be sure that you used Java 8 streams and lambda functions **where possible**, **white box testing** will also be used and a penalty will be applied for those who do not use them.
- covid19.csv file will also be changed during the tests. However, its format will be the same.
- Follow the course page on ODTUClass for any updates and clarifications. Please ask your questions on ODTUClass instead of e-mailing if the question does not contain code or solution.
- Your codes will be tested by calling from main method and as an example,

```
Covid covid = new Covid();
covid.getZeroRowsCount();
System.out.printf(covid.getDateCount("Turkey"));
```

5 Submission

Submission will be done via ODTUClass. You will submit a single tar.gz file called "**e1234567.tar.gz**". In this archive, there should be Covid.java, all the files that you have created and not covid19.csv file. Your homework should compile with

```
> tar -xf e1234567.tar.gz
> cd e1234567
> javac *.java
```