Programming Test

**Task description**

Your main task is to create a system for visualisation of trips where a user can specify a driver for which he or she wants to see all trips. A trip consists of the points recorded from the moment when the driver starts driving until the moment he or she stops. As input, the user will have a file with trips and points. The task must cover both SQLite database and Excel formats, but a special attention should be paid to the extensibility of the solution. The content of the input file are two tables. The first table contains trips and the second one contains points. The user then enters a driver ID, type of the output file with visualised trips, and file name. The final requirement is to visualise trips and store the result to the output file.

The format of the output file should be .txt and PNG.

Besides the main functionality, you need to design and implement a GUI desktop application through which a user can choose the driver ID, output file type, output file name and generate the result.

**Database model**

As previously stated, the database/Excel file should have two tables.

The first table is TRIPS and it has four columns:

TRIP\_ID: string

DRIVER\_ID: string

START\_TS: string(dd-MM-yyyy HH:mm:ss)

END\_TS: string(dd-MM-yyyy HH:mm:ss)

The second table is POINTS and it has five columns:

ID: integer

TRIP\_ID: string

X: integer

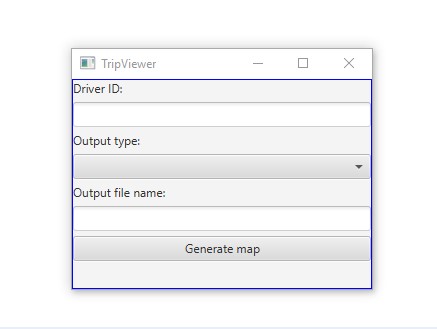
Y: integer

TS: string(dd-MM-yyyy HH:mm:ss)

You can assume that values of X and Y columns are in the range from 0 to 50 (including these two values). These columns represent a position in a 2D coordinate system.

**Requirements**

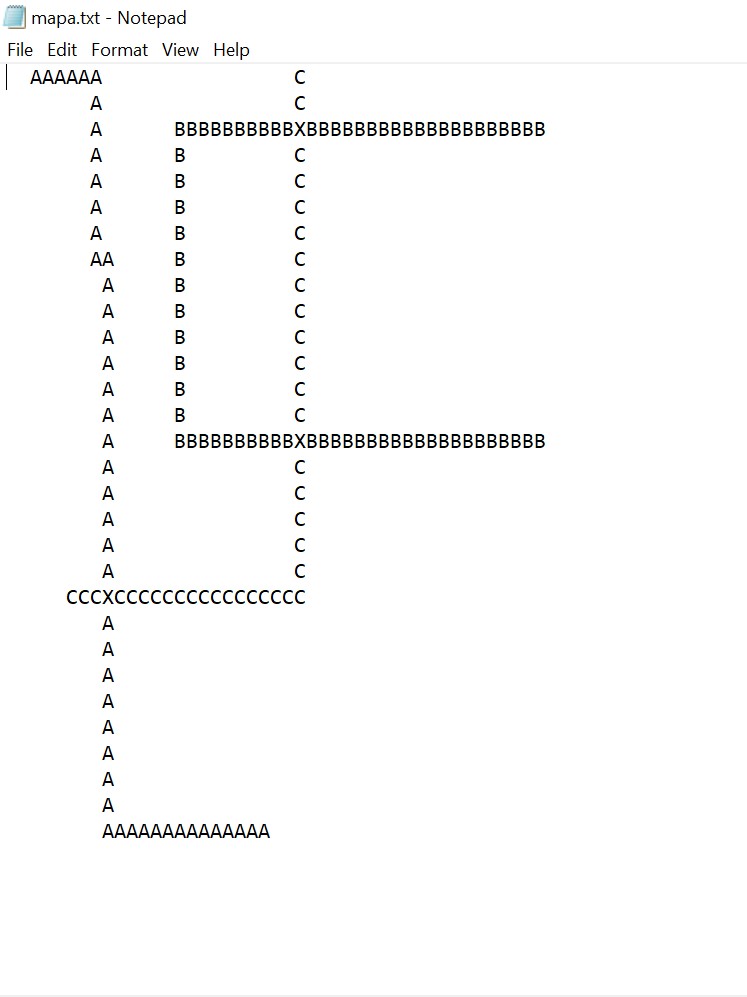
You should create a Java Desktop Application whose task is to perform the jobs described in the previous section. The application will get the file with trips and points and provide a way to search this content by driver ID. Also, the application gets a directory where it should store the output files. After input data is loaded, your application must show a simple GUI similar to the picture below.



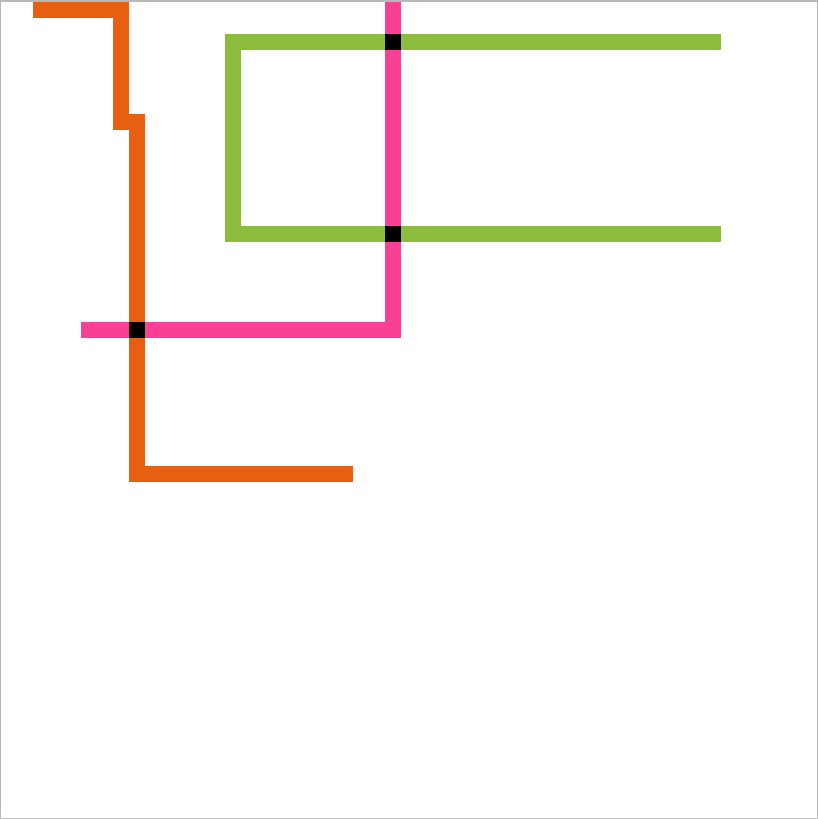
User then enters a driver ID, chooses the output type from the dropdown list (.txt or PNG), types the name of the output file and clicks the “Generate map” button. In the directory passed as the second command line argument, a file with the given name should be created.

Content of that file depends on the selected output type.

In case of a text file, points from the same trip should be marked with the same letter. If in some position multiple trips have a common point, then that position should be marked with ‘X’. If there is no point on some position, mark that position with the whitespace character (‘ ‘). No trip can be presented with ‘X’, all other letters are available. The trip with the earliest start timestamp should be presented with ‘A’, then next with ‘B’, etc. Example of a text output file is given in the picture below.



In the case of a PNG file, points from the same trip should be marked with the same colour. If in some position multiple trips have a common point, that position should be marked in black colour. If there is no point in some position, that position should be marked in white colour. So, the use of black and white is restricted and no trip can be presented in these colours. All other colours are available for you to choose for your trips. An example of a PNG output file is given in the picture below.



In both cases, the top-left corner is the starting position (0,0) and going from left to right, the value of the x coordinate rises, while going top-down the value of the y coordinate rises. Size of a point in the text file is one character. Size of a point in the PNG file is one pixel. Also, you can assume that there will never be more than 25 trips for one driver (it is equal to the number of alphabetic characters without ‘X’).

Outputs from provided files are shown on two previous pictures.

**File format**

The Excel file will contain **two** sheets, one for each table described in the section *Database model*. SQLite database will have **two** tables described in the section *Database model*.

**Calling the application**

Pack the application/program as an executable jar file named ttp.jar. The jar file needs to be named exactly this way (using all lowercase letters), because of the automated tests used later. Running of the program will be executed from the console (command line) with the following command:

java -jar ttp.jar pathToInputFile pathToOutputDir

After executing this command, the application GUI must be shown.

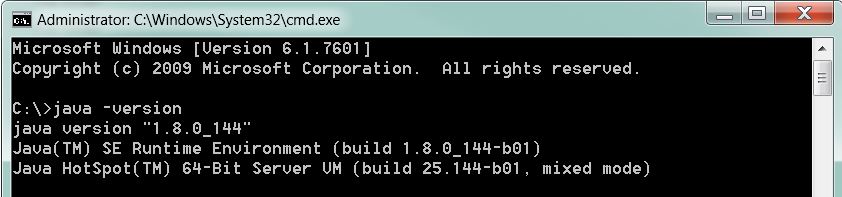
Make sure that your jar file contains all necessary libraries. We recommend executing this jar on a different machine before submission.

As you can see, launching of the application is done by two input parameters:

* pathToInputFile – The string containing the path to the input file(e.g. ./trips.xlsx). The input file will **always** be provided.
* pathToOutputDir – The string containing the path to the output directory(e.g. ./outputDir). The output directory will **always** be provided. You should save files with visualised trips in this directory.

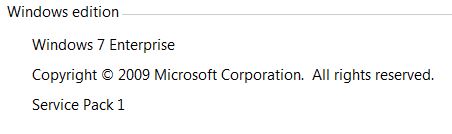
**Testing**

Microsoft Windows operating system with installed Java8 is going to be used for testing. After installing Java and running the command in the console (command line) java –version, the following result should be received:

****

The information about the operating systems can be found in the following picture:

Windows 7

****

Windows 10



When launching the application, two input parameters are set. They represent the path to the input file and path to the output directory.

When testing, we will select parameters and execute multiple test cases by clicking the **EXPORT** button. After exporting the files, we will compare them with expected results.

**Submission**

Your zip/rar solution needs to be uploaded to the link provided in the e-mail.

This zip/rar file needs to contain a folder named **firstname\_lastname**.

The **firstname\_lastname** folder should contain jar files (created as described above), the UML diagrams and the folder containing the source code and all additional libraries.

The UML class diagrams should be at the high-level abstraction and we will use them for a better understanding of your solution.

For example, after completing the assignment, the candidate Petar Petrovic should submit his solution in the following way:

|  |
| --- |
| petar\_petrovic.zip //or petar\_petrovic.rar |
| + diagram.png //UML diagram |
| + ttp.jar //executable jar file |
| + ttp-petrovic //java project |
| + src //source folder |
| + ... //other files/folders |