Task 7

Introduction

In this assignment you will use JavaFX to create a graphical user interface (GUI) for the labyrinth program you created in task 5. Before you start programming, you should make a plan for how the user interface can be structured advantageously both in terms of logic and visual design. It may be helpful to draw some drafts by hand.

Part A: Presentation of the exit-way

Write the program so that you use the class *javafx.stage.FileChooser* to let the user find/select the labyrinth file.

Once the user has selected the labyrinth file to open, the labyrinth should be displayed graphically. For the graphic representation of the labyrinth, the program will use the class

javafx.scene.layout.GridPane.

When the user clicks on a white route, use the program from task 5 to find all the solutions from this route. You should take one solution from the container of solutions and show it in the labyrinth. If there is more than one solution, you can view the first one, with information about how many solutions were found in total.

Conversion of solution strings

In the program from task 5, each solution should be provided in a such format:

```
(1, 1) --> (1, 2) --> (0, 2)
```

It may be difficult to retrieve relevant information from this string to highlight the correct routes in the interface. If you have followed the format above, you can therefore choose to use already made method (sent a java-file for this) to convert a solution string to a two-dimensional array with boolean values. Values that are true indicate that the route is part of a solved exit-way. Values that are *false* indicates that the route is not part of the solution.

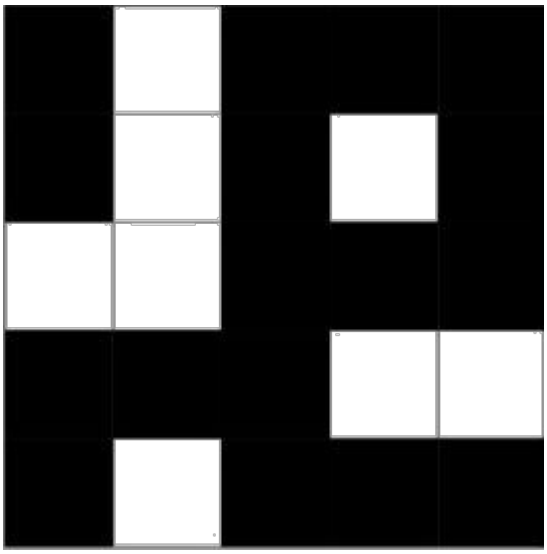

Figure 1: The labyrinth in file 7.

An example call for solutionStringToTable in file 7 (from the test files of task 5) looks like this:

```
String solution = "(1, 1) --> (1, 2) --> (0, 2)" ; // possible exit-way from (1,1) in file 7
solutionStringToTable (solution, 5, 5);
```

This will return a boolean [] [] object with the following values (T represents true and .(dot) represents false):

```
. . . . .
. T . . .
T T . . .
. . . . .
. . . . .
```

You can find the method (sent u a java-file)

Summary

You must submit all classes that requires for the main program to work, including required classes from task 5 and any list classes that you have used.
Avoid using packages (especially relevant when using IDEs like IntelliJ).