Compulsory assignment 3 - INF 102 - Autumn 2017

Deadline: **10 November, 16:00**

Organizational notes

This compulsory assignment is an individual task, however you are allowed to work together with at most 1 other student. If you do, remember to write down both your name and the name of team member on everything you submit (code + answer text). In addition to your own code, you may use the entire Java standard library, the booksite and the code provided in the github repository associated with this course.

The assignments are pass or fail. If you have made a serious attempt but the result is still not sufficient, you get feedback and a new (short, final) deadline. You need to pass all (3) assignments to be admitted to the final exam.

Your solutions (including all source code and textual solutions in the 'hand-in.pdf') must be submitted to the automatic submission system accessible through the course before 10 November, 16:00. Independent of using the submission system, you should always keep a backup of your solutions for safety and later reference.

The project will be a maven project (and not Eclipse project) so that students can use their IDE of choice. If you're using Eclipse, you should either clone the repository first using terminal and then import it as *existing maven project*, or import it directly using this guide.

The link to your git repositoriy is: https://retting.ii.uib.no/<username>/inf102h17_oblig3.git

Some of the problems are covered by tests. Be aware that they are just an indication of how the structure of your code should look like. Passed tests does not necessarily mean that you have solved the problem. Unpassed tests does not necessarily mean that you haven't solved the problem. Though the TAs won't take your own written tests into account, we highly recommend you to write tests as you write code.

A note regarding placement of files: If you're making data files (etc. input files for your programs), use the resource folder. Don't put these in the root folder or in the java packages. Your pdf-document should be in the root folder of the repository, named `hand-in.pdf` as usual.

If you have any questions related to the exercises, send an email to:
`knut.stokke@student.uib.no`
In addition you have the opportunity to ask some questions in the Tuesday review session and at the TA sessions.

1 Printing tree-structured directories

In this exercise you have to make a program that takes in a file containing information about a folder system and prints out the files and the folders in a specific way. We'll call both files and folders *items*. Each item should be printed as "`'-itemname`" after the correct amount of indentation. The items inside a folder should be indented by 2 spaces more than their parent folder, and the root folder should have zero indentation. Hint: a file can be treated as an empty folder.

As an optional refinement (not compulsory, only for those students who want some extra challenge): print the content of each folder in alphabetical order.

The input file will consists of three parts:

- The first line is an integer telling how many items there are in total.
- The next line contains all the item names separated by spaces. The first item has id 0, the next item has id 1, etc. Item 0 is always the root folder.
- The rest of the lines are in the format `"folderX : itemY...itemZ"` which means that the folder with id `folderX` contains the items with ids `itemY, ..., itemZ`.

Here is an example of an input file and output file:

```
7
root folder1 folder2 folder3 file1.txt file2.txt file3.txt
0 : 1 2
1 : 4
2 : 3 5
3 : 6
```

```
'-root
  '-folder1
    '-file1.txt
  '-folder2
    '-file2.txt
    '-folder3
      '-file3.txt
```

You are given a base class `PrintTree` with a main method and an unfinished method `formatTree` that takes a filename and returns a string containing the formatted file structure.

2 Finding the shortest path in a maze

A maze consist of paths, a start point and an endpoint. For this exercise you are given two classes:

- Point with the integer fields x and y.
- Maze with the width and height for the maze (as integers), a point for the start position, the end position, and a set of points representing the walls in the maze. Origin (0, 0) is the upper left point. This class also reads a text file into a maze object.

In the class MazeSolver, implement the solve method to find the shortest path between the start position and the end position. Only horizontal or vertical steps within the boundaries of the maze are allowed, and of course only if the endpoint is not a wall. The method should return a list of points starting from the first position after the start, ending with the position at the goal. The list of points must be in the same order as the player would walk.

I.e. if a maze has `width=2`, `height=3`, `start=(0,2)`, `end=(0,0)`, `walls=[(0,1)]`, then the solution is `[(1,2), (1,1), (1,0), (0,0)]`.

You may use data structures from the course repository or *algs4*, but you are to implement the algorithm yourself.

If you solve the problem correctly you should see an image of the path you found in your default browser.

3 Renewing the sewage system

Bergen is renewing parts of their sewage system, mainly the pipes between the public toilets. They need to connect all the toilets together, and they want to use as few meters pipe as possible. You have been given the responsibilty to find the best way to connect the toilets.

You'll use the classes provided in the package *connectingToilets*. You are to implement the method *connectToilets*. This method gets a set of toilets as input, where a toilet has a name, an x coordinate and an y coordinate. A pipe between two toilets is represented by an object of the class Edge, which contains the names of two toilets and the euclidean distance between them. The method should return a set of these edges, such that the total length of the pipes is the smallest possible—while all the toilets are connected.

Run the program, and you'll see a map of the connections you chose. Your solution should also work for the map of toilets in Australia and the map of random toilets.

Tips: Does this look like a well-known graph problem? If you don't know where to start: What if all toilets were connected to each other in the first place?

Sources:

- *https://data.norge.no/data/bergen-kommune/dokart-bergen-sentrum*
- *http://www.civicdata.com/dataset/national-public-toilet-map-data/resource/8a7cc861-b49b-4506-872e-2153aff575c6*