SIMON FRASER UNIVERSITY
Department of Computing Science
CMPT 276: Assignments

Assignment 3

1 Considerations

This is an individual assignment. Each person has to complete the assignment separately.

The TA will mark the uploaded files on Canvas on the specified deadline. **You will upload one submission for assignments 2 and 3.** See the Schedule page on Canvas for deadlines.

You should work on the same project you created for assignment 2 (Maven project + all code in Java + all tests in JUnit format.

2 Creating your File System from User Input

For this assignment, you will extend the capabilities of your model file system and add the new features to the same code you implemented for assignment 2. In this assignment in particular, we will practice reading and parsing a text file, and creating and traversing and searching the tree structure representing your file system.

2.1 Import a File System from a File

We want to add a feature to our existing file system model that allows a client (the tests suite in this case) to automatically import a file system from its description in a file, without needing to manually create and connect all objects representing all files and directories. The input file is a "real" file on your machine, not a model File object in your program.

Add a `FileSystem` class to your project that is responsible for importing and maintaining the file system in your execution, using the classes you created for assignment 2. This class should keep a reference to the head (root) of your file system, which is a directory. Note that having the root, you can access all subdirectories and files by traversing your data structure. This should be easy now that you have designed and implemented your file system using the composite design pattern, and all files and directories are subclasses of a `Node` class (Component participant), which represents their common behaviour.

Next, add an `import()` method to the `FileSystem` class, which takes as input a relative path (String) to a file. The file is a real text file on your machine, which contains the specs of a file system that a client wants to import. Include a sample input file with your submission that is used by your tests.
Here is an example of the input file:

```
d:root/
d:root/dir1/
d:root/dir2/
f:root/file1.txt
f:root/dir1/file2.pdf
d:root/dir2/dir3/
f:root/dir2/dir3/file3.png
```

Each entry is located on a new line in the file, showing the path to a file or a directory. The first letter of a line determines whether that line contains a file (f) or a directory (d). That letter is followed by a colon (`:`), and then the path to the file/directory. The path separator is a forward slash (`/`).

Update your tests. Implement the new feature and verify its behaviour by running the new tests. When invoking the `import()` method from your tests, make sure that the path to the input file is **relative** to the `$basedir` of your project. Do not use **absolute** paths. The tests should be executable by anyone.
Make sure you have not introduced breaking changes by running your old (regression) tests.

As always, pay attention to the quality of your code. E.g., the assignment asks you to create an `import()` method, but it doesn't mean all your new code should be physically located in that method. Create new/helper methods/fields when needed, and refactor when you see your code is starting to smell.

3 Deliverables

You should upload a zip file containing the code for assignment 2 and 3 "Assignment 2 + 3" on Canvas, named "<Student ID>.zip'.' In addition to the report for assignment 2, the submission should contain:

- Updated code for both assignments, in src/ directory.
- Updated tests that verify the new feature. Include a sample input file that you use in your tests with the submission.
- POM file.