1 New requirements for assignment 2A

As mentioned in my announcement on Blackboard for assignment 2A, I will introduce new requirements as part of the scrum software development process. The customer (that is the instructor), has now introduced the following new items.

1. You are now asked to implement the `find` command. The syntax of the `find` command is as follows: `find path ... -type [f|d] -name expression`. So here are some examples on how the `find` command may be used:

   - `find /users/Desktop -type f -name "xyz"`. This command will search the directory `Desktop` and find all files (indicated by `type f`) that have the name exactly `xyz`.
   - `find /users/Desktop -type d -name "abc"`. This command will search the directory `Desktop` and find all directories (indicated by `type d`) that have the name exactly `abc`.
   - `find /users/Desktop`. This command will result in an error because it has missing arguments. Note: For any missing arguments or incorrect argument parameters, your program must not crash and result in error.
   - `find /users/Desktop /users/Desktop1 -type d -name "abc"`. This command will search the directory `Desktop` and `Desktop1` and find all directories (indicated by `type d`) that have the name exactly `abc`.
   - `find /users/Desktop /users/Desktop1 -type f -name "abc"`. This command will search the directory `Desktop` and `Desktop1` and find all directories (indicated by `type f`) that have the name exactly `abc`.
   - If at any point one of the path in this command is invalid, you must print out an error for that path, however, you must continue searching for any other files or directories that may exist in other valid paths.
   - The `find` command must accept the path just as other commands in your assignment handout, as relative or in absolute path.
   - You can display the output of the `find` command in any format that you wish.

2. You are now asked to implement the `tree` command. The the `tree` command takes in no input parameter.

   - When the user types in the `tree` you must starting, from the root directory ('\') display the entire file system as a tree. For every level of the tree, you must indent by a tab character.
     - For instance if the root directory contains two subdirectories as 'A' and 'B,' then you will display the following:

```
\
  A
  B
```

     - For instance if the root directory contains two sub directories as 'A', 'B', 'C' and 'A' in turn contains 'A1' and 'A2', then you will display the following:

```
\
  A
   A1
   A2
  B
  C
```

     - When the user types in `tree` and the only directory present is the root directory, then you simply show the root directory.

```
\
```

2 Some more things to keep in mind

1. With these new requirements, you will first modify your product backlog and make sure that you have added in the new requirements in there as user stories. Once you have completed this, you will then create your sprint 2 backlogs. Then, you will continue working as normal on your sprint backlog.

2. Each one of you will be asked to perform a demo starting in the week of 25th June. More details of the demo will be made available over the weekend sometime on Piazza.

3. You can use any libraries within the Java JDK; however, you cannot use the inbuilt File and Directory class. Remember, you are creating a mock file system. At the very least, you are expected to create these classes yourself and not use the inbuilt one. Further, the inbuilt classes of File and Directory will interact with the real file system, and you do not want this behaviour in your assignment.

4. Do not forget to watch my video on the use cases that we will use to test your assignment. This video contains a similar type of use cases that me and the TA will use when testing your assignment. These video links are also available on Piazza.

   - Part 1 of the video is here `https://youtu.be/qjF4SBUuj_c`
   - Part 2 of the video is here `https://youtu.be/BazNwN8syoo`

5. You are responsible for the commands that are mentioned in the assignment two handout and in this handout. Do not add extra commands that the customer has not asked for. Make sure the commands work as mentioned in the handout. You will be penalized if your commands do not meet this specification and will be marked as a failure on validation by the customer.

6. When creating directories, the only valid characters are from lowercase `a` to `z` and upper case `A` to `Z` and numbers `0` to `9`. Anything other than these characters are considered invalid. You must give an error when any of the invalid characters are part of the `mkdir` command.

7. You must have guessed from the above item, that file names and directory names in `JShell` do not contain a space. Space is considered as an invalid character.

8. The '⟩⟩' in the echo command when appending or the '⟩,' file creation command, must only redirect the actual non-error component of the string into the file. For some reason, if this command were to generate an error, the error must always be shown on the console and never be part of the file that gets created. This is important because you must ensure somewhere in your program that you have a way to separate out the error portion of the string from the non-error portion of the string. In the BASH shell, this is typically done by having the STDOUT and the STDERR. If you are not familiar with the STDOUT and the STDERR, I suggest you to read this `https://en.wikipedia.org/wiki/Standard_streams`.

9. For the `man` command, I expect to see some documentation (not extensive), but enough that allows some user of your JShell to get familiar with the commands. You must ensure that you have few examples in the provided description, guiding the user on how to use the commands correctly. You are free to use any format that you wish when displaying the output.

10. The `STRING` in the `echo` command can contain a space. As a matter of fact, you can have any character in the `STRING`.

    - When writing or appending the file contents to the `FILE` in the `echo` command, the internal contents of the `FILE` must have some new line separator, so that your `CAT` command can then use this new line character to figure out how to break the contents into separate lines when displaying back on the console of the `JShell`.

11. When using the `CAT` command with multiple files, it is OK if you were to separate the contents of one file from the other file with some set of separator characters. Whatever this is, make sure to document it in the manual of the `CAT` command that can then be used by the user by typing in `man cat`.

    - if any of the `FILE` in the `CAT` command has an invalid file contents, make sure to show the error for that file on the console, however, you must proceed with the other `FILE` and if these other `FILE` are valid, display the contents of these `FILE` on the console.