Task 2: File system manager

You have been tasked with creating a JAVA console application that can be used to manage and manipulate files.

Requirements

You are required to create a proper folder structure that matches conventions. It should separate your JAVA source code from your non-source code (the provided files and ones you generate). You are not to use an IDE (Intellij) at this point, we are compiling the old-fashioned way like was shown in Lesson 2. The commands will be a challenge to get with all the classes and folders, but that is the point.

The program should contain several classes, the number of classes and how to group (package) them is up to you – it is part of the convention mark.

The program contains several parts:

1. User interaction and input
2. File manipulation
3. Logging

User interaction and input

This is the main loop of the system where it will provide the user with various options – these options are to match the functionality of your system. This menu should persist, meaning that the user can make a choice, see what happens, and them come back to the menu. They should have an option to exit the program from here.

File manipulation

You are to create a File service which has the single responsibility of gathering information about files in the project. The provided files are to be used for this, but feel free to add any other files and expand on this functionality. All these files should be placed in a resources folder, and that's where your application should look (see project structure for an idea where to put this folder).

The **first** requirement is a method that lists all the file names in the resources directory.

The **second** requirement is a method to get specific files by their extension (all .txt files, all .png files, ect…). You can present the user with all the available extensions and allow them to choose.

The **third** requirement is a several methods which manipulate the provided .txt. They should:

- The name of the file.
- How big the file is.
- How many lines the file has.
- Let me search if a specific word exists in the file.
- How many times the specific word is found in the file.
- The search must ignore case.

Logging

You are to create a Logging service which has the single responsibility of logging (writing) strings to a file. This file should be placed in an appropriate folder and be named appropriately.

You are to log all the results from the “file manipulation” functionality (in addition to displaying the message to the console – you can just log the message with a timestamp).

The duration of each of the function calls must be logged as well. You could use the system time to do this. i.e. Get system time, execute function, get system time, subtract the two times as duration.

An example of a log could be something like:

9-9-2020T14:00: The term “hello” was found 1000 times. The function took 157ms to execute.

Provided files

- A txt file for the manipulation of contents. Most of the “file manipulation” functionality is performed in this file.

- Some images (.png, .jpeg,…), these are used for one or two smaller requirements in “file manipulation”.
- Some other files for requirements 1 and 2 in “file manipulation”.

Output

Your application should be compiled into a single .jar file placed in /out/, as is convention.

Provide screenshots (in a folder and shown in your REAME) with the following images:

- The “javac” command run to compile all your .java files into .class files in /out/.
- The command to create a single .jar file from your /out/.
- Running the .jar file (java -jar MyJarFile.jar) in a terminal showing the menu being presented, i.e. the application is working.

NB: You will notice that you cant move your .jar file from where you create it and have it work properly (resources won’t be found), this is because there is a path to resources that is pointing to the resources outside your .jar file. It is okay to have this “issue” – if you want you can look how to bundle resources folder into your jar file so that it moves around with it, go ahead 😊.

Optional upgrade

You can have two versions of searches. The first one you can make as unoptimized as you want, then you can have fastSearch where you try optimize it (have a look at various search algorithms – like binary on sorted lists and others). Due to us logging the time it takes to execute you can try see how much faster your optimized search is that the straight forward method.

Submission

Provide a git repo link as submission. This repo should contain a well formatted (its written in markdown, so look how to make headings,

#Heading1, …) README explaining what your project does. Some extra inspirations could be provided if you want.

Remember, you can use your tasks as a portfolio, so try and make it as attractive as possible.

Comments

Provide comments to explain functionality. More comments are better and will result in a better mark. Be aware not to over comment just for the sake of placing a comment on every line. Comments should server the purpose of explaining – I should be able to come into the application and understand exactly what it does and how it works from reading comments, variable names, and function names.

Error handling

Proper error handling is expected throughout the application. These are most found in user inputs and file manipulations.