abcfinlab – Interview exercise
Anagram
Documentation

Milson António

02/05/2021

Contents

1- Build and Run Instructions

The first step to run or test the Application in your own machine is to download the project, unzip the files at their destination. The folder should look like the in the Figure below.

Options:

Link 1 – download

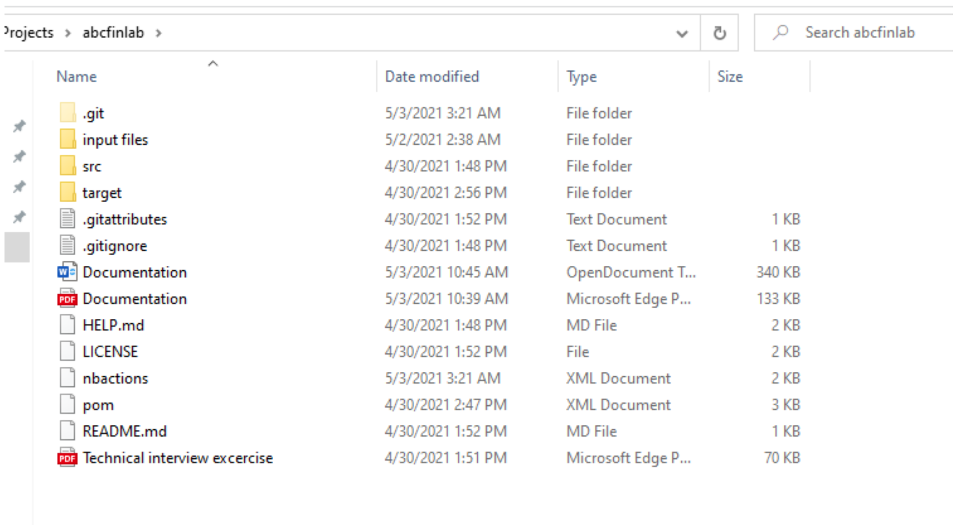
https://furb-my.sharepoint.com/:f:/g/personal/milsona_furb_br/Es26v8RA39FPsw8Le9HQ4kEB27FW2C6g6bTreYX0ziIv3w?e=khTv5L

Link 2 – Git Hub repository

https://github.com/Milsondepaz/abcfinlab

Link 2 – Run directly in Heroku

https://anagramabcfinlab.herokuapp.com/



Name	Date modified	Type	Size
.git	5/3/2021 3:21 AM	File folder	
input files	5/2/2021 2:38 AM	File folder	
src	4/30/2021 1:48 PM	File folder	
target	4/30/2021 2:56 PM	File folder	
.gitattributes	4/30/2021 1:52 PM	Text Document	1 KB
.gitignore	4/30/2021 1:48 PM	Text Document	1 KB
Documentation	5/3/2021 10:45 AM	OpenDocument T...	340 KB
Documentation	5/3/2021 10:39 AM	Microsoft Edge P...	133 KB
HELP.md	4/30/2021 1:48 PM	MD File	2 KB
LICENSE	4/30/2021 1:52 PM	File	2 KB
nbactions	5/3/2021 3:21 AM	XML Document	2 KB
pom	4/30/2021 2:47 PM	XML Document	3 KB
README.md	4/30/2021 1:52 PM	MD File	1 KB
Technical interview excercise	4/30/2021 1:51 PM	Microsoft Edge P...	70 KB

After finishing the download, you can import the project to your IDE preferably as a MAVEM project. In NetBeans, you can open it directly because the project was created from this IDE. after having the project open inside your IDE, just "Build" and then you can Run, if all goes well, in your browser type the address: **http://localhost:8080/** you will see the application in your browser.

2- Design rationale for solution

Technically, the solution for this type of problem is based on manipulation of Strings, in a summary way is to compare if two “Strings” have the same size first, if this check is true then the next one is to verify if both have exactly the same characters. This is the central and main idea.

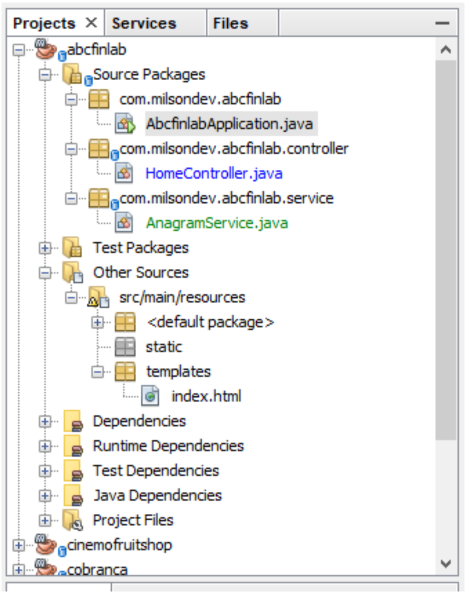
To solve this solution, I chose to use the following tools and technologies:

- Java 8
- Spting Boot 2.4.5
- Thymeleaf
- Bootstrap 4
- HTML & CSS
- NetBeans 8.2

Through a GUI, intuitive, simple and easy to handle, the user selects the directory file, only with the extension “.txt”, once this is done, click on an “Upload & Process” button for the application to check the file and return a result, which is returned in a non-editable "text area", with a label "Output", the user can also receive a simple message, about the success or not of his request.

In “Source Package”, I created two sub-packages “controller” with the “HomeController” class, this class receives two requests, a “Get” through the default address **http://localhost:8080/** and the other “Post” type, through the address **http://localhost:8080/upload** both of which later return a single "View" the "index" with the difference that in the second request some attributes are sent to be rendered to the user.

This is the structure of the project:



```
abcfinlab
    Source Packages
        com.milsondev.abcfinlab
            AbcfinlabApplication.java
        com.milsondev.abcfinlab.controller
            HomeController.java
        com.milsondev.abcfinlab.service
            AnagramService.java
    Test Packages
    Other Sources
        src/main/resources
            <default package>
            static
            templates
                index.html
    Dependencies
    Runtime Dependencies
    Test Dependencies
    Java Dependencies
    Project Files
cinemofruitshop
cobranca
```

2.1- The AnagramService class

The AnagramService Class, has 1 main public method "ProcessAnagram" that receives the file as parameter and returns a String with the processed anagram groups. It also has other 3 private auxiliary methods that help solve this task.

The “ProcessAnagram” method passes the file to be processed to the “TextContentToArray” method, which in turn aims to read all the “Strings” contained in the file and add them to an ArrayList <String> that it could also be some other object but after some analysis, he realizes that there is no such need, it is only possible to work with "Strings".

After all the words in the file are inserted into the ArrayList <String>, it is time to go through this data set and find out which words are part of an Anagram group.

The logic at this point was to take an element in the “i = 0” position of the ArrayList <String> and the element in the following position, “i++” and then send both “Strings” to the “isAnagrama (String pr_str1, String pr_str2)” method which returns a boolean value. When the result is true, the element “i++” is removed from the list, since a word cannot be an anagram of two words that are not anagrams, removing that element “i++” from the list also helps to reduce the algorithm cost. as a whole, since in the next rounds, the list will have a smaller set of elements.

The method "isAnagrama (String pr_str1, String pr_str2)" as seen, receives two "Strings" and returns a boolean value. More details about the operation of this method are explained in the section **“3.1- How do you verify your solution?”**.

3- Questions

3.1- How do you verify your solution?

When we think in solution to solve Anagram's problem, the first idea that comes to mind is that we must compare the size of two words, if they are different, they are not anagrams, if they have the same size, then we must check if they have exactly the same characters, usually in this step, we can face a problem, the order of the characters, and to solve this, It will be necessary that the character set of the two words, before being ordered for later comparison, between the characters of both words in the same position.

The method I adopted to solve is based on a logical operator xor “^”, as we know all characters are part of the ASCII table where they are represented by numbers. Therefore, the strategy was to go through a word, converting each character into the equivalent bit and save it in a variable using the operation "xor" "^" successively, in the end we will have the bits equivalent to every word in the variable. And the same process is repeated for the second word, and the value of this second operation must be saved in the same variable successively, if the characters of two words are equivalent, the bits will be equivalent, and the xor operation will reset the variable value to 0 bit.

Answering the question, I think my solution is computationally more efficient compared to the example above, since this has to sort two collections that will make the algorithm slower. There are several algorithms for sorting collections and each one with its cost and efficiency, if we were to implement an algorithm to sort the collections, we would lose more time but we can use Java methods to sort collections without implementing it with our hands.

3.2- How complex is your program? How will it perform on large datasets?

I don't consider my solution complex, just use the xor operator and most of us are no longer used it on a day-to-day, we only study once in University and forget, but the solution is simple and easy to understand, also it's faster with the time complexity of O(n).

For now, I don't see anything that I can change to perform it or turn more efficient if I get an large dataset.

3.3- Imagine you have a large input file (e.g. multiple gigabytes) and only limited memory (100MB). How would your program perform? Please outline if and how you would change your solution (no need to implement this)

With a large input file and little memory, I am not absolutely sure but I believe that even before we think about algorithm efficiency, we would have trouble uploading the file. I could think of dividing the file before working on it, but we would still fall into the same problem because before I divide this file into small pieces to work with, I must be able to load it completely into memory first.

But if we can split the file into pieces unless our machine can load it, I think one way that would help to make the application faster and have a good performance would be to implement Threads, and make each Thread do some part of the work.