IFOR-Entrega

Author: Sergio Esteban Pellejero.

This overview is in English because while I was developing the software I wrote the README.md of the repo. As the jobs needs a PDF specifying an overview i convert the README to a PDF. If you see the README of this project on github, you'll find out some images in the doc, which aren't here because in the conversion to PDF they were lost.

Making a small GUI oriented to a little suite of forensics applications. With this applications you'll be able to see the diferences between files, directories.

Table of Contents

Objectives

The main goal of this repository is to bring some software build from scratch to the forensics field into the software development. The software shown in this repository is extremely simple but could be a proof of concept or a seed to make bigger projects if the community wants to.

Tools used in the project

Main tools used are:

- **Visual Studio Code**: main IDE to develop the software. Very useful due to the integration with SCV tools as git and all the useful plugins. You can obtain more information here VSCode
- **Gradle**: for build the project and make all the folders and settings to the project. Combined with intellij for download dependencies, it is so powerful. You can get more information at Gradle
- **Github**: to have a control version and share knowledge with rest of the world. You can find more software projects in my Github account.
- **JavaFX**: the future of graphics in Java, it combine the power of CSS and the simplicity of awt to make bright and colourful gui's. See JavaFX documentation to get more information.

Running the software

There is only **one condition required** to run this software. You must **have installed JDK 11**. Be sure that your enviromental variable for java is pointing to the JDK 11 distribution, if not, the software will not run. To check your java version you need to run the command `java --version` if your are in Linux, or `java -version` if your're a windows user. Then the output should be something like this:

As said before, make sure that you're enviromental variable is pointing to the right way. You can follow the next tutorial: How to set up ENV variable for java in windows or if you are in Linux How to set up ENV variable for java in Linux to set up or check the enviromental variable. Also, sometimes is needed to reboot the computer to update the value of the variable to the correct one.

In theory you don't need gradle, because the gradle wrapper makes all the work. In case the gradle wrapper fails in the execution, you must install gradle in your computer and, as told before, make sure the envormental variable for gradle is poiting to the last version. For this software you need Gradle 4.10.2.

To run this software, you have to specify in the build.gradle file the Program to run. You must change there the variable mainClassName. The values accepted are:

- `mainClassName = 'Program1'` : to run the program related to search diferences.
- `mainClassName = 'Program2'` : to run the program related to search files between date ranges.
- `mainClassName = 'Program3'` : to run the program related to searach files by the extension or magic numbers.

Program 1

This program has two diferent functionalities:
- **Search for differences between two files**: given two files, selected with the known path or the gui, the program search for differences in these files and show them in the text area of the program.
- **Search for differences between two directories**: given two directories, selected writing the path or with the gui, the program search for differences in these directories and show you the files that are different.

This program is based on the `diff file1 file2` command. This command is supported in a native way by `Bash shell` and also in the `Windows Power Shell`.
Basically you select the folders or directories, passed them as a string in the command executed by Java.

In the two cases, if the program doesn't find any diferences, show a message lie No differences were found. You can select which version of the program you want to run by clicking in Tools menus in the upper bar.

Program 2

This program has three diferent functionalities:
- **Search for created files between a date range**: given a date range in the format "10 Dec 2018", without quotes. Also able to parse dates like "10 December 2018", "10 dec 2018" or "10-12-2018" or "2018/12/12". Also you

need to select the folder you want to look for and then click in Look for! button. The the software shows you the results in a table. In this table you are able to select some files and copy them into a given directory previously selected, once the file was copy is removeed from the table and the software shwos you a dialog.

To run this program on linux, Java is executing the following command: `find inPath -newerct "YYYY-MM-DD" ! -newerct "YYYY-MM-DD" -ls`. This command looks for all the files in the path `inPath` created (-newerCT, C=creation, T=timeStamp) between the date range provided. This command is not supported in a native way on windows. Because of this in windows Java runs: `Get-Child rest of the command`. **Doesn't adapted for windows**.

- **Search for modified files between a date range**: if you go to the Tools menu in the upper bar, your're able to select the functionality which you have to run. This allows you to look for by the modified files within a date range into a folder.

To run this program on linux, Java is executing the following command: `find inPath -newermt "YYYY-MM-DD" ! -newermt "YYYY-MM-DD" -ls`. This command looks for all the files in the path `inPath` modified (-newerMT, M=modified, T=timeStamp) between the date range provided. This command is not supported in a native way on windows. Because of this in windows Java runs: `Get-Child rest of the command`. **Doesn't adapted for windows**.

- **Search for accessed files between a date range**: if you go to the Tools menu in the upper bar, your're able to select the functionality which you have to run. This allows you to look for by the accessed files within a date range into a folder.

To run this program on linux, Java is executing the following command: `find inPath -newerat "YYYY-MM-DD" ! -newerat "YYYY-MM-DD" -ls`. This command looks for all the files in the path `inPath` modified (-newerAT, A=accessed, T=timeStamp) between the date range provided. This command is not supported in a native way on windows. Because of this in windows Java runs: `Get-Child rest of the command`. **Doesn't adapted for Windows**. I know the gci command works fine to do this job but I don't have time to implement it.

The software allows you to select the date range by a DatePicker, which is better than writting the date in the text field. Also when the software copy the selected file into the selected directory, informs the user with an information dialog.

Program 3

This program has two diferent functionalities:
- **Search files filtering by extension**: given a file extension and a root directory, the user is abel to look for all the files tino the given directory. Once the files are found, then the user is able to copy them into a diferent folder, previously selected by the user.

To run this program on linux, Java is executing the following command: `find inPath -type f -name '*.extension'`. This command look for all the files (-type f), which which names (-name) matches with the next regular expression ('*.extension') where extension is the extension provided by the User in the GUI's textfield. The user doesn't need to add the `.` to the extension, just typing `java` or `py` instead of `.java` or `.py` works fine. This command is not supported by WIndows, because of that, Java runs the following command `Find-command in windows`. **Doesn't adapted for Windows**. There is a command like find in the power shell but I don't have time to find it.

- **Search files filtering by magic numbers**: this function is not implemented yet. But basically the user should be able to look for files by some magic unmbers provided by an external program.