Assignment 2: Interfacing Linux with a Java GUI (7%)

SoftEng 206, Semester 2, 2020

Due: Monday 7th September, 3pm

Learning Outcomes

The purpose of this assignment is to target the following learning outcomes, as listed in the SoftEng 206 course outline. In particular, they are:

Learning Outcome #1: *be able to explain the common issues that arise in the construction of software.*

Learning Outcome #2: *be comfortable using the command line to perform file processing, writing/running scripts, and various system calls.*

Learning Outcome #3: *be able to explain the purpose of, and how to use, a number of the tools commonly used in the construction of software.*

Introduction

This assignment contains similar requirements as Assignment 1, except you now develop a GUI interface for your program instead of requiring the user to interface via the command line. You will ultimately be using the same Linux commands under the hood, however the user should never know about this. It will allow the user to view the game board, ask a question, view the current winnings and reset the game. Similar to Assignment 1, your GUI program here should also take into consideration *ease of use* and *error handling*, for example:

- Provide a simple GUI to help the user achieve their tasks.
- You should be thinking more about **usability** and giving **useful error/confirmation messages** to the user.

The Main Menu

Similar to Assignment 1, you should provide a main GUI menu that allows the user to perform all of the following operations. This can either be in one GUI menu, or on different tabs. Naturally, as it is a GUI, the "quit" functionality will be when the user closes the main GUI window.

"Frozen" GUI

You may notice that the GUI will "freeze" during long operations, such as when you run a command that takes a long time. You need to avoid this by using the correct mechanisms taught in class (e.g. using JavaFX Tasks or SwingWorker). In particular, you need to ensure the thread-safety correctness of your program (i.e. ensuring that GUI components are only accessed from the GUI thread).

Technologies to use

If you need to make calls to BASH from within the Java application, you can use the approach taught in class. Other than that, you can use anything you want to write to files (i.e. more Linux BASH calls, or just use Java File I/O). You are welcome to use either JavaFX or Swing, but make sure it works in the VirtualBox image and that you take GUI concurrency into consideration.

Frequent Submissions (GitHub)

You **must** use GitHub as version control for all assignments in this course, including this assignment. To get the initial repository for this assignment, you must accept the GitHub Classroom invitation that will be shared with you. As you work on your assignment, **you must make frequent git commits**. If you only commit your final code (or too infrequently), it will look suspicious and you will be penalised. You can check your commits in your GitHub account to make sure the frequent commits are being recorded. Using GitHub to frequently commit your changes is mandatory in this course.

Final Submission (Canvas) and Assessment

You will submit via Canvas (final submission) and GitHub (frequent snapshot commits). **Make sure you can get your program running with the VirtualBox image shared by the lecturers**. On Canvas, submit the following, in a single ZIP archive file:

- This assignment is worth 7% of your course mark. Your submission must include the following files submitted via Canvas. You should use ZIP to archive your submission – **do not use any other archive format.**
  - The entire project source (all your *.java files and any other files necessary), and
  - An easy to run "executable" with the README.md file how to run your project (without requiring the code to be compiled). Either include an executable jar file, or a script file that runs it. Remember that this executable will run on the VirtualBox image shared in the course.
  - A **signed and dated Cover Sheet** stating that you worked on the assignment independently, and that it is your own work. Include your name, ID number, the date, the course and assignment number. You can generate and download this in Canvas, see the Cover Sheet entry.

You must double check that you have uploaded the correct code for marking! In general, there will be no exceptions if you accidentally submitted the wrong files. Should you happen to mess up, your GitHub repository will be the only acceptable form of evidence to prove you did not modify your submission since the deadline. No other exceptions. Get into the habit of downloading your submission from Canvas, and double-checking all was submitted correctly.

All programs will be executed on the Linux image specified. Be sure to test *all* your work using the command line, as all programs will be executed this way. You will lose a lot of marks if the markers cannot get your program to run on specified image, this is especially important if you developed your code via Cygwin or the likes.

Late submissions

Late submissions incur the following penalties:

- 15% penalty for zero to 24 hours late
- 30% penalty for 25 to 48 hours late
- 100% penalty for over 48 hours late (dropbox automatically closes)

You must double check that you have uploaded the correct code for marking. There will be no exceptions if you accidentally submitted the wrong files, regardless if you can prove you did not modify them since the deadline. No exceptions.