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

SoftEng 206, Semester 2, 2017

Due: Monday 4th September, 1pm

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 create new creations, and then list/view, play and delete existing creations. 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 interact with the application.
- You should be thinking more about usability and giving useful error/confirmation messages to the user.

Assessment

- 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 README file how to run your project (without compiling). Either include an executable jar file, or a script file that runs it.
  - A **signed Cover Sheet** (found on Canvas) clearly stating the date, your name, the course name, the assignment number confirmed that the submission is all your own work that does not include any academic misconduct. Your submission will not be marked without this declaration.
- **All programs will be executed on the ECE Linux image.** Be sure to test *all* your work on that image before submission, otherwise you will lose a lot of marks if markers cannot run it.

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.

"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 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 EDT/GUI thread).

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. The functionality includes: **View, Play, Delete, Create** creations. Naturally, as it is a GUI, the "quit" functionality will be when the user closes the main GUI window.

Create a new creation

Similar to Assignment 1, this includes creating a new creation by entering a name and recording their voice for 3 seconds. However, the user might attempt to type any character on the keyboard (not only `a-z` and `A-Z`) – so keep this in mind for possible error messages and how your program should respond. While you can reject certain characters, you should at least allow letters, digits, hypens, underscores and hyphens as part of the creation names. You are welcome to reject other characters. Similar to Assignment 1, offer the user the chance to hear the recording before finalising the creation, and the name should not conflict with an existing creation. If the user insists, you might want to offer them the chance to overwrite an existing creation (e.g. they provide the same name, and you double-check they know they are overwriting).

View creations

This will be very similar to Assignment 1. However, it is up to you how to display these and in what order (alphabetical? creation time?). You might decide to combine the View with Delete and Play functionality (e.g. a table view with buttons for actions to Play or Delete the creation). It otherwise may appear pointless having 3 different places where you display the creations. In the end, it comes down to your design.

Play creations

Allows the user to play an existing creation. Simple enough. While it might be tempting to call ffplay to do this, how would this impact the look and feel of the application? Chances are it would look weird playing the creation in an external application. You should therefore be embedding the video in the GUI.

Delete creations

While this could be easy to implement, consider showing a thumbnail/preview of the creation somehow non-intrusively. This could either be in the confirmation message when the user tries to delete it, or maybe selecting a creation in a list view shows a mini thumbnail/preview of the video (different from an actual Play of that creation which opens it full-size with sound). Start thinking about the user experience.

Technologies to use

You will need to make use of FFMPEG to create the videos, so you should use the approach taught in class to make calls to BASH from within the Java application. 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 all works in the labs and that you take GUI Concurrency into consideration.