# Homework 2 – Implementation of a simple maintenance task in an unfamiliar Open Source Software

# Software system and version: muCommander 0.9.3

# Change Request*:*

Add my full name to the middle-top of the initial splash window of the application, and make sure the splash window is visible for at least 5 seconds. Do NOT add your name in the bottom status bar of the splash window.

# Detailed Environment Description*:*

Eclipse Version: Oxygen.2 Release(4.7.2)

OS/Machine: Mac OS High Sierra (10.13.3) / MacBook Air

Working Directory: /Users/jguevara/CS-471/mucommander

Gradle User Home: /Users/jguevara/.gradle

Gradle Distribution: Gradle wrapper from target build

Gradle Version: 4.8

Java Version: Java JDK 1.8.0

Java Home: /Library/Java/JavaVirtualMachines/jdk1.8.0\_161.jdk/Contents/Home

JVM Arguments: None

Program Arguments: None

Build Scans Enabled: false

Offline Mode Enabled: false

Gradle Tasks: <run default tasks>

# Compiling or Running Issues:

No compilation issues were encountered. However there were a few issues when running because of how I originally intended to implement this feature: using layered panes. For some reason my indexing was not correct and kept overshooting resulting in a NullPointerException. In order to go around this, since how this was to be implemented was not specified, I decided to go with repainting my name along with the Loading Message and Version of muCommander. So while I would have liked to not repaint my name each time, and leave it static and laid over the imageIcon, the repaint work around works just as well at the moment.

# Identifying the Relevant Code Location:

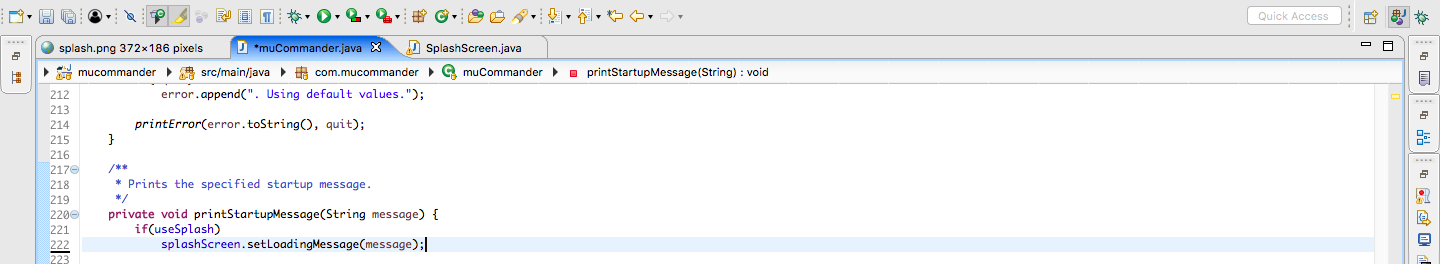
In order to identify where the changes need to be implemented I had to step back and think about how other programs that I’ve worked on have been usually “laid” out. By this I mean that I thought of how the splash page must’ve needed an image to set the splash background. Therefore there must be a resource folder (which would usually be in the **src** directory). Luckily enough there is such a folder exists in muCommander. Within this folder the splash image name is given. From there the next point for me to go was to the main muCommander file. This I found by going to **src/main/java** (I figured it’d be here since we’re editing in Java, and it’s the main source folder). Next I figured **muCommander.java** was the driver for the whole project, and decided to search for anything called ‘Splash’. Found that the SplashScreen class was being called here, and from there I looked in the header for the specific class/package that this was being called from: **com.mucommander.ui.main/SplashScreen.java**. It’s here where I decided to implement my change under the overridden paint() function. After that all it took was some fiddling/guess-work to get my name positioned correctly.

As for getting the splash screen to remain for longer than five seconds, all I did was go back to the main driver and insert a sleep(), for 500 milliseconds, call inside the printStartupMessage() function (figured this is where all the splash screen messages were coming from when it’s loading up) this can be found from lines 224-229 of my modified muCommander.java.

# Successful Implementation Video:

https://youtu.be/a-pSGpikvg4

# Highlighted Source Code of the Implementation:



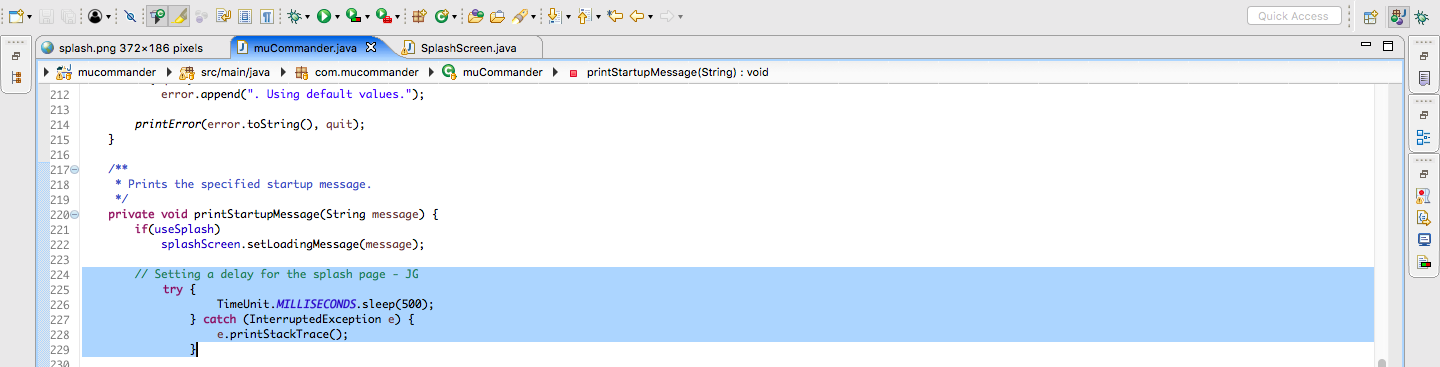


Figure 1 Call to sleep() for 500 milliseconds, in order to have the splash page stay longer as opposed to waiting for each process to finish before continuing. Pre-implementation (top), implementation (i.e. diff, bottom).

# 

Figure 2. Highlighted section is the implementation of adding my name to the pane during each call to the overridden paint() from repaint(). Since the code was added at the end of the original, I did not see a need to include an image of the code prior to the change. (i.e. the original code is just the same file sans the part highlighted in blue)

# Time required for completing this assignment:

|  |  |
| --- | --- |
| **Activity** | **Time Required (in minutes)** |
| How much time did it take you to clone the repository, read the documentation about, etc. before being able to successfully compile and run the software? | *< 10 min.* (I’m used to cloning repositories and getting code to run on my machine) |
| How much time did it take you to understand the code that you need to modify, to implement the change request, and to test it? | **~ 15 – 20 minutes** to read/understand what I was doing.  **~ 4 hours** to implement and test (I wanted to do it a different way, but couldn’t get it quite right. So I went with what I thought was good, and it could’ve taken less). Testing as I went. |