Installing and Verifying Java Guide (Zulu OpenJDK 11 with JavaFX)

Installation

Windows

- Uninstall any existing versions of Java (if any) on your PC.
- Go to https://www.azul.com/downloads/?version=java-11-lts&os=windows&architecture=x86-64-bit&package=jdk-fx#zulu, download the `.msi` from the "Java 11 (LTS)" section, and install it.

Mac

- Go to the URL, download the `.dmg` from the "Java 11 (LTS)" section, and install it:
  - For Apple Silicon Macs ("M" Series processors): https://www.azul.com/downloads/?version=java-11-lts&os=macos&architecture=arm-64-bit&package=jdk-fx#zulu
  - For Intel-based Macs (not "M" Series processors): https://www.azul.com/downloads/?version=java-11-lts&os=macos&architecture=x86-64-bit&package=jdk-fx#zulu
- Run the following command (copy and paste the entire highlighted box in your terminal and hit enter):

```
printf "\nexport JAVA_HOME=/Library/Java/JavaVirtualMachines/zulu-11.jdk/Contents/Home/\n" | tee -a ~/.bash_profile ~/.zshenv > /dev/null && export JAVA_HOME=/Library/Java/JavaVirtualMachines/zulu-11.jdk/Contents/Home
```

- After completing the Verification, Part 1 step, if the JDK version doesn't match the expected, follow these steps: https://medium.com/@devkosal/switching-java-jdk-versions-on-macos-80bc868e686a
  - Include step 5, but instead of adding the required text to just `~/.bash_profile` add it to the file (create the file if it doesn't exist) `~/.zshenv` too.

Linux

- See https://www.azul.com/downloads/?version=java-11-lts&package=jdk-fx for an appropriate version in the "Java 11 (LTS)" section. Do not change the filters for "Java Version" and "Java Package" (should be autocompleted from the link)
  - For the "Operating System" field, we recommend choosing specific Linux distributions, like "Ubuntu," instead of the generic "Linux" option (unless the generic "Linux" is the only one that is available for your computer architecture).

Verification, Part 1

If you use the command `javac --version`, you should get an output like this (the version might vary):

```
javac 11.0.23
```

If you use the command `java --version`, you should get an output like the one below:

```
openjdk 11.0.23 2024-04-16 LTS
OpenJDK Runtime Environment Zulu11.72+19-CA (build 11.0.23+9-LTS)
OpenJDK 64-Bit Server VM Zulu11.72+19-CA (build 11.0.23+9-LTS, mixed mode)
```

Important notes to review the information for these commands:

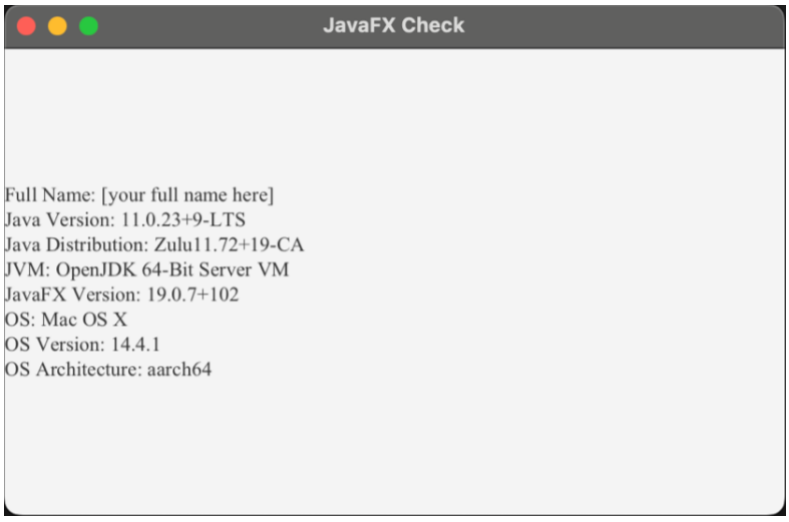
- They might show a different version of OpenJDK 11, like `11.0.24`, but the first version number must be 11, and it should be at least `11.0.23`. For the java command, it should also be marked "`LTS`" after a date. The version in the `javac` and `java` commands must match.
- The distribution version might be different, but it must start with "`Zulu11`", end with "`-CA`" and should be at least `Zulu11.72+19-CA`.

If these steps didn't work (either the command is not showing the Zulu installation or the version is not correct), review that you followed the installation steps correctly. For Mac, there is a troubleshooting step to try before reviewing the steps.

Verification, Part 2

To verify that your installation is fully complete, you will need to run a simple JavaFX application that tests that your JDK is capable of executing JavaFX applications. Follow these steps:

- Copy the code on the next page into a file named `JavaFXCheck.java` file.
- Using the terminal, go to the folder you saved the file in using the `cd` command.
- Compile and run the application with the commands `javac JavaFXCheck.java` and `java JavaFXCheck`
- When you run the application, you will type your name and hit enter before the GUI appears. After that, verify that the shown GUI looks like the one below. It should have your name and the correct information about the JDK and your computer. The JavaFX version should be one starting in 19.



```
import java.util.Scanner;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.stage.Stage;

public class JavaFXCheck extends Application {

    private static final String COULD_NOT_DETERMINE = "could not determine";
    private static String studentName = COULD_NOT_DETERMINE;

    private static String getProperty(String key) {
        try {
            String property = System.getProperty(key);
            return (property != null && !property.isEmpty()) ? property : COULD_NOT_DETERMINE;
        } catch (Throwable t) {
            System.err.println("Could not retrieve property for key: " + key);
            return COULD_NOT_DETERMINE;
        }
    }

    private static String getInfo(String propertyFriendly, String propertyKey) {
        return propertyFriendly + ": " + getProperty(propertyKey) + "\n";
    }

    private static String getLabelText(String studentName) {
        String text = "Full Name: " + studentName + "\n";
        text += getInfo("Java Version", "java.runtime.version");
        text += getInfo("Java Distribution", "java.vendor.version");
        text += getInfo("JVM", "java.vm.name");
        text += getInfo("JavaFX Version", "javafx.runtime.version");
        text += getInfo("OS", "os.name");
        text += getInfo("OS Version", "os.version");
        text += getInfo("OS Architecture", "os.arch");
        return text;
    }

    private static void getStudentName() {
        try {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter your full name: ");
            JavaFXCheck.studentName = sc.nextLine();
            sc.close();
        } catch (Throwable t) {
            System.err.println("Could not retrieve full name");
        }
    }

    @Override
    public void start(Stage stage) {
        Label info = new Label(getLabelText(studentName));
        info.setStyle("-fx-font-family: 'serif'");
        stage.setScene(new Scene(info, 500, 300));
        stage.setTitle("JavaFX Check");
        stage.show();
        stage.requestFocus();
    }

    public static void main(String[] args) {
        JavaFXCheck.getStudentName();
        Application.launch(args);
    }
}
```