Lab 0: Infrastructure Setup

Instructor: Mackale Joyner

Course Wiki: http://comp322.rice.edu

Goals for this lab

- Piazza Setup
- Java 8 installation
- Maven installation
- IntelliJ installation
- GitHub Classroom integration
- Introduction to HJlib Installation and Setup

Important tips and links

NOTE: The instructions below are written for Mac OS and Linux computers, but should be easily adaptable to Windows with minor changes. For example, you may need to use \ instead of / in some commands.

Note that all commands below are CaSe-SeNsItIvE. See FAQ section if you have setup issues.

Piazza site: https://piazza.com/rice/spring2021/comp322/home

Java 8 Download: https://www.oracle.com/technetwork/java/javase/downloads/index.html

Maven Download: http://maven.apache.org/download.cgi

IntelliJ IDEA: http://www.jetbrains.com/idea/download/

1 Piazza Setup

We will use COMP 322's Piazza site, https://piazza.com/rice/spring2021/comp322/home, for all discussions and Q&A. Please only register on this site with your email address of the form, *your-netid*@rice.edu.

If you can access the Q&A tab in this Piazza site, you are done and you can move on to the next section. If not, please send email to the instructors with a request to add your email address to the Piazza enrollment list for this class.

2 Java 8 Setup

You may already have Java 8 installed on your computer. To check whether you have Java 8 installed on your machine, go to the command line and type the following `/usr/libexec/java_home -V`. If you see something as follows:

```
 $ /usr/libexec/java_home -V
Matching Java Virtual Machines (2):
    11.0.3, x86_64: "Java SE 11.0.3" /Library/Java/JavaVirtualMachines/jdk-11.0.3.jdk/Contents/Home
    1.8.0_102, x86_64: "Java SE 8" /Library/Java/JavaVirtualMachines/jdk1.8.0_102.jdk/Contents/Home
```

you already have Java 8 installed on your machine. *If you do have Java 8 already installed on your machine, please skip to section 3.*

If you do not already have it installed, you will need a Java 8 installation on your machine. Java 8 can be downloaded and installed from the Oracle website.

On Windows, the environment variables (e.g. JAVA_HOME, PATH) have to be set up differently. Please refer to the stackoverflow question to see how it can be done.

3 Maven Setup

Maven is a build automation tool used primarily for Java projects. Projects using Maven or other build systems are easiest to use as they simplify compiling, building, and testing the project. In addition, major IDEs like IntelliJ and Eclipse have excellent support via maven plugins which simplifies the development process. Dependency management is one of the areas where Maven excels; for our purposes, this means that we will save a lot of effort in configuring the projects to set up dependencies on HJlib, JUnit, and other jars. Hence, for the labs and assignments in COMP 322, we will distribute maven project templates to be used by students to complete their work.

Maven is a Java tool, so you must have Java installed in order to proceed. Next, follow the instructions on the maven site to install maven. Please remember to install a version that is 3.3.3 or later. Detailed instructions are also available on the COMP 322 wiki at `https://wiki.rice.edu/confluence/display/PARPROG/Using+Maven+for+HJlib+projects`. During installation, you may need to have your `M2_HOME`, `MAVEN_HOME` and `PATH` point to the new installation. For example, in the following `.bash_profile` file on a Mac:

```
export M2_HOME=/Users/mjoyner/apache-maven-3.6.3
export MAVEN_HOME=/Users/mjoyner/apache-maven-3.6.3
export PATH=$PATH:$MAVEN_HOME/bin
```

Once installed, open a new command prompt and run `mvn --version` to verify that it is correctly installed. If you see something as follows:

```
 $ mvn --version
Apache Maven 3.6.3 (7994120775791599e205a5524ec3e0dfe41d4a06; 2020-01-05T06:57:37-05:00)
Maven home: /Users/mjoyner/apache-maven-3.6.3
...
```

Maven has been successfully installed on your machine.

If you have unzipped the contents of the Maven zip file and notice an output similar to the following:

```
$ mvn --version
-bash: mvn: command not found
```

it means that your `PATH` variable has not been configured correctly.

4 IntelliJ Installation

An IDE is not strictly required for this course since Java programs can also be written using a text editor and then compiled using the command-line. However, we strongly recommend using an IDE as it simplifies writing, compiling, running, and debugging your programs. A good IDE gives you error warnings, code completion and navigation, refactoring, syntax highlights, etc. We recommend using the IntelliJ IDE to do the Java development in the labs and assignments for this course. A free version of IntelliJ (Community Edition) IDE can be downloaded and installed from the Jetbrains website.

5 Use of JUnit in COMP 322

JUnit is a unit testing framework for the Java programming language. As in COMP 215, we will use JUnit for all labs and programming assignments. You will be provided basic JUnit tests for your labs and assignments. Note that, in Maven projects, there are separate directories for the source code and unit tests. Since Maven will handle our dependencies, it will automatically download the necessary JUnit jars. We will be using version 4.13 of JUnit, this can be determined by checking the version in the `pom.xml` file of each individual Maven project.

6 GitHub Classroom Setup

GitHub is a software versioning and revision control system. This allows you to recover older versions of your data or examine the history of how your data changed. During COMP 322 we will use your private repository to distribute code to you. You can always examine the most recent contents of your GitHub classroom lab1 repository by visiting *GitHub lab1*. You'll be asked to select your name the first time you visit the GitHub classroom. If you are unable to access the above URL or locate your name, please send email to comp322-staff@mailman.rice.edu and request that your name be added to the roster. After that, you can ignore this section for now (till you get access) and move on to the next section.

There are primarily four git commands you will need to be familiar with for COMP 322: `clone`, `add`, `commit`, and `push`. A git `clone` downloads files from the Git cloud to your local laptop, like downloading the provided source code starter code for labs and homework from your private git repository. A git `commit` saves the staged changes to your local repository. Provide a description of the change using the `-m` option. A git `push` uploads files from a checked-out version of the git repo on your local laptop back to the git cloud. It will only upload the staged changes you have made to those files. You can look back through your commits to see the changes you have made over time. The git `commit` updates the cloud GitHub classroom repo with changes you made to your local copy. If you create or modify a file inside a local, checked-out Git repo, you'll first have to perform a Git `add` to inform git that there is a new file or modified changes it should add to the staging area.

There are a few ways you can interact with your git repository: use git with IntelliJ projects, use the EGit plugin for Eclipse, use a standalone Git client like Tortoise Git, or use the command line.

7 Lab 1 Exercises

Visit the url at *GitHub lab1* to ensure that the files for this lab are available. Select your name if this is your first time visiting the GitHub classroom. The directory structure for the lab should look like the following:

```
pom.xml
src
```

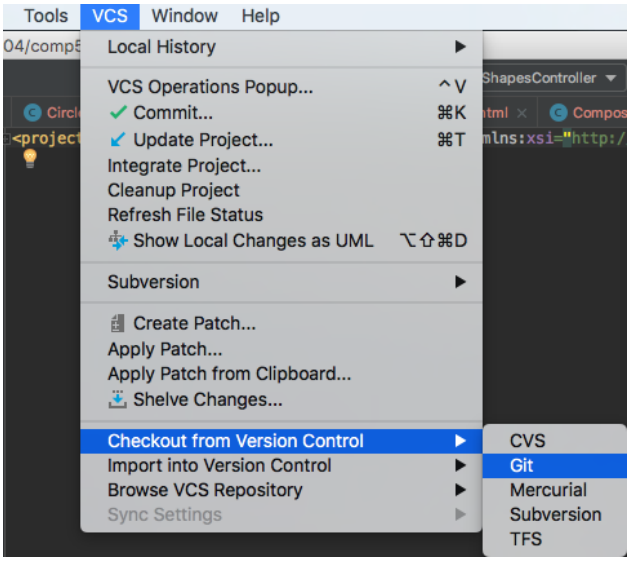
```
main
   java
       edu
           rice
               comp322
                   HelloWorldError.java
                   ReciprocalArraySum.java
test
    java
        edu
            rice
                comp322
                    Lab1CorrectnessTest.java
```

Note that you will need to have Java 8, Maven, and IntelliJ installed for this step. You can download the `lab1` project on your machine through either the command line or using IntelliJ's Git support. The two sections below contain instructions on both approaches, so you only need to follow one.

7.1 Using IntelliJ to checkout your Git repo

Checking out your Lab 1 Git folder from the remote Git server is straightforward in IntelliJ. You can start from either the "Check out from Version Control" option on the welcome screen:

or use the "Checkout from Version Control" option accessible under the top menu's VCS tab:



Use the following GitHub classroom url:

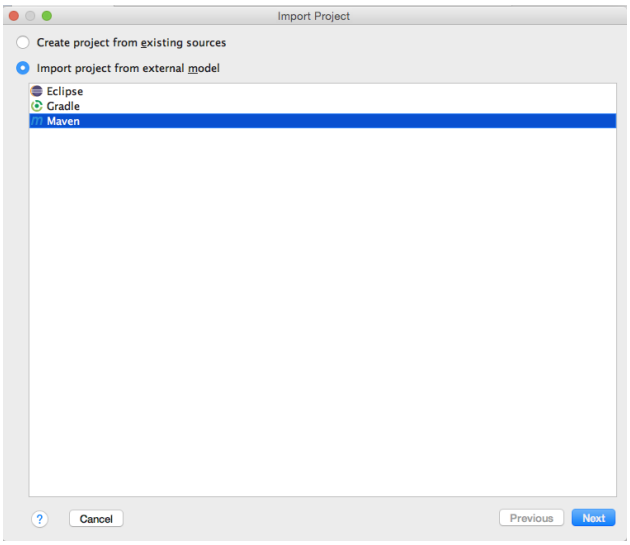
`https://github.com/RiceParProgCourse/lab1-GITID.git`

with GITID replaced by your GitHub account name. Click the Checkout button to start downloading this repo.

The next prompt will query you for the folder name to checkout the local copy of the GitHub folder to. The choice of folder is up to you. Once you have made a choice, click OK.

IntelliJ will now ask if you would like to open the newly checked-out project. Hit Yes:

On the next screen, you may be asked if you would like to import this project as a Maven project (your IDE may also skip this step and auto-detect it as a Maven project, in which case you can skip ahead):



Finally, IntelliJ should open your project and may ask you if you would "like to schedule the following file for addition to Git?". Select No. You are now ready to get started!

7.2 Using the command line to checkout your Git repo

Note that you do not need to follow these instructions if you have already checked the project out through IntelliJ. These instructions are for those who would prefer to work on the command line. The command-line Git client can be installed by following the instructions at Git. Once installed, you can verify that the command works by running the following command:

```
$ git --version
  git version 2.10.1 (Apple Git-78)
```

To checkout your Git repo for lab 1 on the command line, navigate in a terminal to the directory you would like to checkout into and issue the following command:

```
$ git clone https://github.com/RiceParProgCourse/lab1-GITID.git
```

Replace GITID with your GitHub account name. You should see a lab1 directory created in your working directory with the source code for lab1.

8 FAQ for Setup Issues

- **Question:** My InteliJ gave me an error message when I tried to check out the project.

  It said "Cannot load supported formats: Cannot run program "git" (in directory "..."): CreateProcess error=2, The system cannot find the file specified"

  How should I fix this?

  Answer: This probably means you need to install the Git command-line client. For Windows, I recommend Tortoise Git - make sure you install "command line client tools", which is not installed by default. On Linux, sudo apt-get install git. On OS X, I believe you get git with XCode (let us know if you're on a Mac and having this problem).

  You need to restart cmd after installation, before typing "git" so cmd will reload the PATH environment. You also need to restart IntelliJ.

  To check whether git is installed, open a terminal (or cmd, on Windows) and type "git". You should see "Type 'git help' for usage" or similar. Let us know if git is installed and you're still having problems.

- **Question:** One of the first few things the lab asks you to done is run "mvn clean compile", but where is this?

  Answer: There are a few options here. Be sure you actually have Maven installed first.

  1. You could use the builtin terminal in IntelliJ (should have a button on the bottom bar).
  2. You could use whatever command line interface is on your machine (Command Prompt on Windows, Terminal on Mac, etc.)
  3. IntelliJ has a handy Maven Projects tool window (should be a button on the right when you have a project open) where you can execute the commands (under "Lifecycle", double click "clean" and then "compile").

- **Question:** How do I fix the exception in thread main() when running HelloWorldError.main()?

  Exception in thread "main" java.lang.Error: C:/Program Files/JetBrains/IntelliJ IDEA Community Edition 2017.3.2/lib/idea_rt.jar=52189:C:/Program Files/JetBrains/IntelliJ IDEA Community Edition 2017.3.2/bin

  Answer: Go to "Run" then "Edit Configurations" and paste the following line into VM options:

-javaagent:"C:/Users/your_user/.m2/repository/edu/rice/hjlib-cooperative/0.1.13-SNAPSHOT/hjlib-cooperative-0.1.13-20170111.022154-1.jar"

Be sure to change your_user to your own username. Remove the C:/ if you're using a mac.

- **Question:** I can commit to git through the command line, but IntelliJ says that there are no updates detected. How do I solve this issue?

  Answer: Make sure to add the specified files for tracking before attempting to commit