In-Class Exercise 1: Infrastructure Setup

Instructor: Mackale Joyner

Course Website: https://www.clear.rice.edu/comp504

Goals for this exercise

- Piazza Setup
- Java 11 installation
- Maven installation
- IntelliJ installation
- GitHub classroom integration
- Introduction to Spark Java

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.

Piazza site: https://piazza.com/rice/fall2020/comp504/home

Java: https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html

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

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

1 Piazza Setup

We will use COMP 504's Piazza site, https://piazza.com/rice/fall2020/comp504/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 instructor with a request to add your email address to the Piazza enrollment list for this class.

2 Java 11 Setup

You may already have Java 11 installed on your computer. To check whether you have Java 11 installed on your machine, go to the command line and type the following `java -version`. If you see something as follows:

```
 $ java -version
java version "11.0.3" 2020-04-16 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.3+12-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.3+12-LTS, mixed mode)
```

where the Java version shows 11.*, you already have Java 11 installed on your machine. *If you do have Java 11 already installed on your machine, please skip to section 3.*

If you do not already have it installed, you will need a Java 11 installation on your machine. You may need to have your `JAVA_HOME` and `PATH` point to the new installation. Java 11 can be downloaded and installed from the Oracle website.

For example, if Java 11 was not installed in the /usr/bin or /usr/local/bin directory on a Mac, you may need to something like the following:

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk11.0.3.jdk/Contents/Home
export PATH=$JAVA_HOME/bin:$PATH
```

On Windows, the environment variables 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 JUnit and other jars. Hence, for the exercises and assignments in COMP 504, 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 3.3.9 or later. During installation, you may need to have your `MAVEN_HOME` and `PATH` point to the new installation. For example, on my Mac machine, I can set up the following:

```
export MAVEN_HOME=/Users/mjoyner/apache-maven-3.6.1
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 3.6.1 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2020-08-27T02:58:13-05:00)
Maven home: /Users/mjoyner/apache-maven-3.6.1
Java version: 11.3.0, vendor: Oracle Corporation
...
```

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

We strongly recommend using the IntelliJ IDE for all software development in the exercises and assignments for this course. A free version of IntelliJ (Ultimate Edition) IDE can be downloaded and installed from the Jetbrains website if you use your Rice email account. You should not download the Community Edition. It has some missing features that you will need for you exercises and homework assignments.

5 Use of JUnit in COMP 504

JUnit is a unit testing framework for the Java programming language. We will use JUnit for all programming 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 4.12 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 504, we will use your private repository to distribute code to you. You can always examine the most recent contents of your GitHub classroom exercise 1 repository by visiting GitHub ex1. You'll be asked to select your student id (netid) the first time you visit the GitHub classroom. If you are unable to access the above URL or locate your netid, please send email to the course instructor and request that your netid 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 504: `checkout`, `add`, `commit`, and `push`. A git `checkout` downloads files from the Git cloud to your local laptop, like downloading the provided source code template for exercises and homeworks from your private git repository. A git `commit` saves the staged changes to your local repository. 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 subversion 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 In-class Exercise

Visit the url at GitHub exercise 1. Select your netid if this is your first time visiting the GitHub classroom. Ensure that the files for this exercise are available. The directory structure for the exercise should look like the following:

```
pom.xml
src
    main
        java
            edu
```

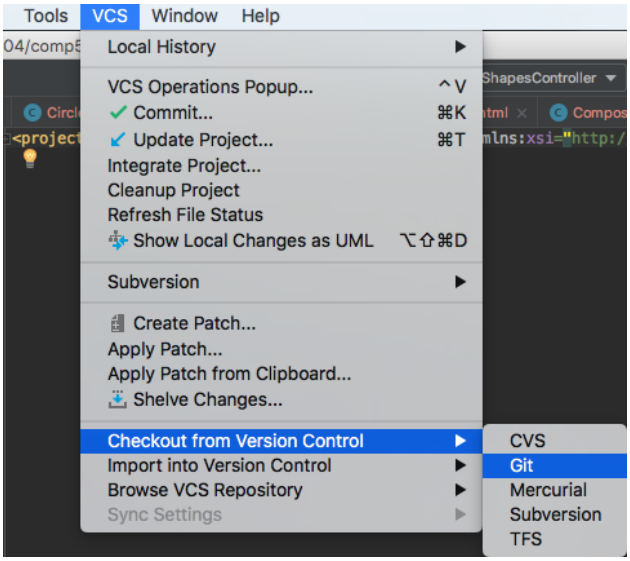
```
rice
    comp504
        HelloWorld.java
```

Note that you will need to have Java 11, Maven, and IntelliJ installed for this step. You can download the `ex1` 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 exercise 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:

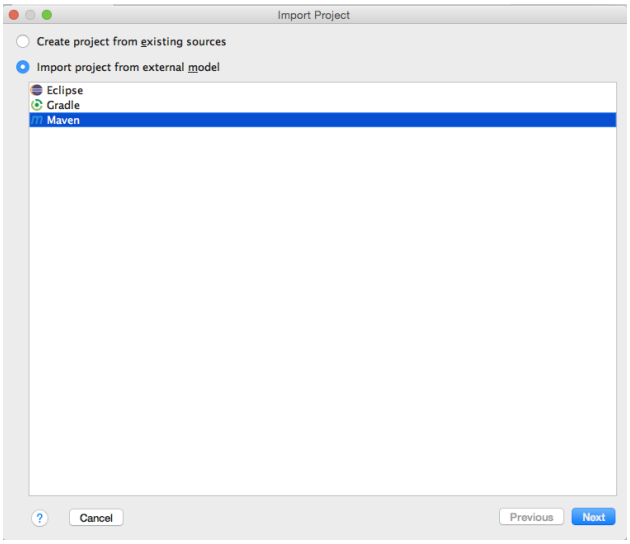
```
https://github.com/RiceGradOOCourse/ex1-gitID.git
```

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 Git 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 Subversion?". 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 exercise 1 on the command line, navigate in a terminal to the directory you would like to checkout into and issue the following command:

```
$ git checkout https://github.com/RiceGradOOCourse/ex1-GITID.git
```

Replace GITID with your GitHub account name. You should see a ex1 directory created in your working directory with the source code for this exercise.

7.3 Hello World in Spark Java

The exercise for today is to output "Hello, World!" on your browser. I recommend using Chrome to test your solution since we will use Chrome when grading all your homework assignments. You should see the `HelloWorld.java` source file in the `src/main/java` directory. You'll notice that `HelloWorld.java` contains 3 TODO comments that indicate what you need to do to complete this exercise. The TODOs are the following:

1. Add the Spark Java `get` request method for the "/" endpoint. The "/" endpoint should return "Hello, World!" using a lambda function. You'll want to create a Run configuration to test your solution. To create a run configuration, select Run on the menu and click on "Edit Configurations...". You'll click on the `+` button to add a new configuration. The configuration type will be `Application`. Give the configuration a name. The main class should be `edu.rice.comp504.HelloWorld`. Click on the `OK` button.

   You should now see the run configuration you just created appear on a dropdown list of run configurations. Hit the run button to start up the local server at `http://localhost:4567/`. Your browser should output "Hello, World!" for the url `http://localhost:4567/`.

2. Create a redirect from the endpoint "/hello" to "/". You should not create a `get` request method for the "/hello" endpoint.

3. Add a `System.out.println` to output the method request and the url each time you change the url in the browser. Now, each time you type a url in the browser, you should see both a method request and the url output in the console.

Please don't forget to add, commit, and push your work to your GitHub classroom cloud (remote) repository.