Setting up Eclipse and Git

You will need a Java Virtual Machine (JVM) and a Java Integrated development Environment (IDE). We recommend Eclipse, but you may choose to use a different development environment. If you are running any Java apps, you already have the Java Runtime Environment (JRE) installed. If not, you can obtain it from https://java.com/en/download/ or you can download the full Java Development Kit (JDK) from https://www.oracle.com/technetwork/java/javase/downloads/index.html.

Download and install the Java IDE Eclipse. An installer and installation instructions are available here. Your Eclipse installation should include Java, JUnit, and an Eclipse Team provider for the Git version control system.

Earlier versions of Eclipse should work fine. Be sure that the Eclipse Team Provide for Git is installed.

If you installed the JDK and receive an error message from Eclipse about a missing JRE when attempting to run a Java program, follow the instructions here to tell Eclipse to use the JDK instead of the JRE.

For all assignments in this class we will be using Java SE 8 and JUnit 4. Do not use any Java SE 9, 10, or 11 specific features or JUnit 5 syntax or features, as it would cause your code to fail to compile on Submitty, resulting in a grade of 0 for the corresponding autograded components.

Cloning the Homework Repository in Eclipse

In the instructions below,

- substitute **HW_Folder** with hw00, hw01, etc.;
- substitute **HW_Name** with hw0, hw1, etc.;
- RCSID with your RCS ID, i.e., the part of your RPI e-mail address before @rpi.edu, e.g., kuzmik2. **RCS ID is NOT YOUR RIN! Your RIN is a series of numbers, e.g. 12345678**

Note: Package Explorer and Project Explorer are two different windows in Eclipse. Pay attention to the text to make sure that you are trying to access the correct window.

Clone a homework repository in Eclipse:

1. Create a directory to hold your CSCI-2600 projects. For example, in Windows you might choose something like C:\Users\Konstantin Kuzmin\Documents\CSCI-2600\Spring2020\csci2600.
2. Start Eclipse and select the path you created as a workspace.
3. Go to **File** ⟶ **Import**.
4. Select **Git** ⟶ **Projects from Git**.
5. Select **Clone URI** and click **Next**.
6. Enter the URI: `https://submitty.cs.rpi.edu/git/s20/csci2600/HW_Folder/RCSID`.

   Important: `HW_Folder` is a Submitty repository folder name such as hw00, as shown above. RCSID is your RCS ID.

7. Enter your RCS ID and password then click **Next**.

8. In the `Branch Selection` dialog, click **Next** (`master` should be selected).
9. In the `Local Destination` dialog, select a target directory for the local Git repo (e.g., in Windows it might be something like C:\Users\Konstantin Kuzmin\Documents\CSCI-2600\Spring2020\csci2600\git\kuzmik2), then click **Next**.

   Important: Remember the target directory! You will need to type it in when creating the Java project in Eclipse.

10. In the dialog that appears, select `Import using the New Project wizard`, then click **Finish**.
11. In the `New Project` dialog, select **Java** ⟶ **Java Project**, then click **Next** to start creating a new Java project.
12. In `Project Name`, type `csci2600-HW_Name (e.g. csci2600-hw0)`, uncheck `Use Default Location` in `Location`, enter the target directory where the local Git repo resides (in our example, C:\Users\Konstantin Kuzmin\Documents\CSCI-2600\Spring2020\csci2600\git\kuzmik2), then click **Finish**.

    Important: Make sure the project name is `csci2600-HW_Name. HW_NAME` is a homework name such as hw0, as shown above. It is not the same as the Submitty repo folder. For example, the project name for the first assignment should be `csci2600-hw0`.

13. At this point, you have created a Java project under Git version control. In *Package Explorer* you should see a project named `csci2600-HW_Name [RCSID master]` with the folder structure that matches the description below. Typically, you will see the JRE System Library as well. Open the description of the homework from the docs folder. You may find it convenient to open an HTML document by right-clicking on it in the *Package Explorer* and selecting **Open With** ⟶ **Web Browser** from the context menu.
14. Last, add JUnit to the project. Right-click on the project name in *Package Explorer*, then select **Properties**. In the Properties dialog box, select **Java Build Path.** Select the **Libraries** tab. If ClassPath appears in the window, Select ClassPath and then the Add **Library…**. Button, otherwise, just select the Add **Library…**. button. In the Add Library dialog box select **JUnit**, click **Next**, select **Junit 4** from the JUnit library version list, then click **Finish**. Do not chose Junit 5.

    To run JUnit tests, right-click on a test file, e.g., FibonacciTest.java, or on an entire test directory, then select **Run As** ⟶ **JUnit Test**.

Make sure that your directory structure is correct. If not, compilation on Submitty will fail resulting in a grade of 0. You must have project `csci2600-HW_Name` with subfolders `src`, `docs`, and `answers`. Folder `src` must have subfolders `main` and `test`. All Java source code files (both classes created for the assignment and their corresponding JUnit classes for test cases) are assigned to the same package with the name *HW_Name* causing those files to be placed in the *HW_Name* folder, which can be verified in *Package Explorer*.

```
project
        src
                main
                        java            // Java source files
                        resources       // xml, images, properties, etc.
                test
                        java            // Java source files for JUnit test cases
                        resources       // xml, images, properties, etc.
        answers                         // text files with answers to assignment problems
        docs                            // assignment description and other documentation
```

Modify the code and create new files, as required to complete the assignment

Perform whatever actions are needed to complete the assignment.

Using Git:

1. To commit a file that is already under version control, e.g., the starter files that you cloned from the Submitty repository, right-click on the file, select **Team** ⟶ **Commit**, type a commit message, then click **Commit**. This commits changes to your local repository.
2. To add a new file, e.g., a text file with assignment problem answers that you created in the `answers` folder, to version control, right-click on the file, then select **Team** ⟶ **Add to Index**. Then commit.
   Important: if you do not add a new file to index, it will not be tracked by Git and therefore neither committed to your repositories, nor submitted to the Submitty server. You will receive a grade of 0 for any code or responses that have not been submitted to Submitty, regardless of whether they are present in your local or remote repositories. Every time you submit to Submitty, it records the date and time, and it is the only method that we will be using to ensure that submissions are made before the deadline. Be sure to add the files that you create to your Git repo index.
3. To push local commits into the remote repository on the server, right-click on the project name in *Package Explorer*, e.g., `csci2600-HW_Name`, then select **Team** ⟶ **Push to Upstream**. When committing, you may click the **Commit and Push** button in the `Git Staging` dialog box to commit and then immediately push the new commit to the remote repository. In this case, there is no need to perform the push operation separately.

   You may commit and push your changes to the repository as often as you deem appropriate. We highly recommend at least committing and pushing to Submitty after finishing a major part of the assignment (e.g., a particular problem) or before experimenting with the code after you reached some stable, but not necessarily complete, state of the project. In general, try to commit and push early and often, not just when you are done with the assignment! There is no penalty, nor limit on the number of times you may commit and push. In all the cases, when you commit, write a meaningful commit message describing the changes that were made to the project since the previous commit. **Commit early and often.**

Turn in the assignment:

1. The Submitty page for a particular assignment becomes available a few days before the assignment due date. Test your code thoroughly on your local machine. Do not use Submitty to

test or debug your code. You should be testing code on Submitty that you reasonably believe to be functional and well tested. Start early, preferably as soon as the assignment is released!

2. Once you are done enough with the assignment that you want to have it graded, push the repo to the remote Git repository on the Submitty server, click **Submit** for the corresponding assignment in Submitty (authenticate with your RCS credentials), and then click **Grade My Repository**. Submitty retrieves your Git repository and autogrades parts of the assignment. It also shows you which files it retrieved. Make sure to check this list and confirm that it contains all the files that should be submitting for the assignment.

   For the autograded part of a homework assignment, Submitty will show the points earned. If errors have occurred, click the Details links to see the error message.

   Important: Your homework will not be graded unless you click **Grade My Repository** the button.

   Important: Submitty is set up to allow up to 20 submissions (clicks of the **Grade My Repository** button) graded without a penalty. At the 21st submission, there will be a 1 point penalty deducted from your overall score. At 30 submissions, there will be an additional 1 point penalty and at every 10 submissions afterwards up to a maximum of 5 points.

   Insurance Against Computer Crashes

   In a class of this size, on average one student will have a serious computer crash during the semester. Commit and push your work to your repository often. Ideally, push your files each time you exit an Eclipse session. This guarantees that your work has been saved. If you do have a crash, at least you can recover your homework. Also, get in the habit of backing up your files. Use a cheap USB drive or online system to back up your important files.

   "There are two kinds of people in the computer business: those who have had a crash and lost valuable data, and those to whom it hasn't happened YET."