COMP3111: Software Engineering

Introduction to GitHub and Starting a Gradle Project

Learning Outcomes

- Be familiar with the steps of creating a local repository of a Java project via Eclipse
- Be able to learn how to commit changes and other Git operations
- Be able to create a GitHub account and track changes in the remote repository

Supervised Lab Exercises

Environment: Eclipse (Version: Photon RC3 (4.8.0RC3)) with Java Development Kit (JDK 10 64-bits) installed on a Windows 10. The steps may be slightly difference if you are using other versions of Eclipse or Mac

There are four guided exercises in this lab:

1. Create a Gradle Java Application project in Eclipse;
2. Create a local Git repository;
3. Commit to your local Git repository;
4. Setup a remote Git repository on github and push to it.

After finished these four exercises, you need to work on some unguided tasks. Demo the result to your TA and your lab is done. (If you missed the lab due to add/drop or whatever reason, submit it on Canvas.)
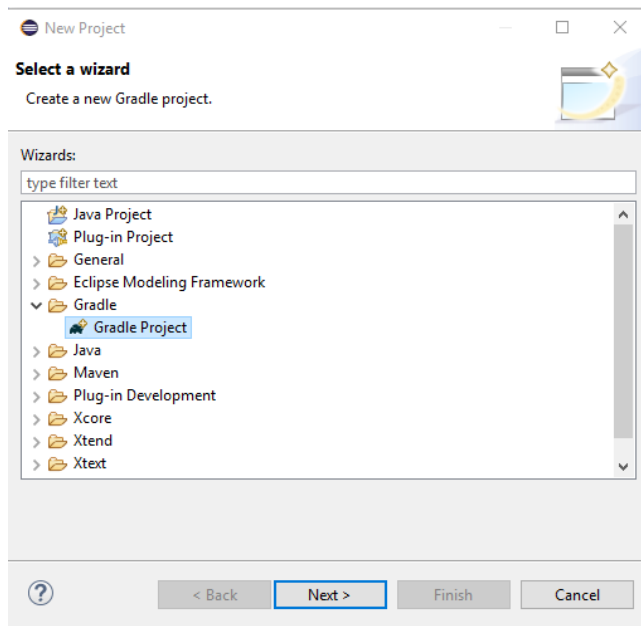
Submission/Demo – Submit To Canvas

Exercise 1: Create a new Gradle Java Application project
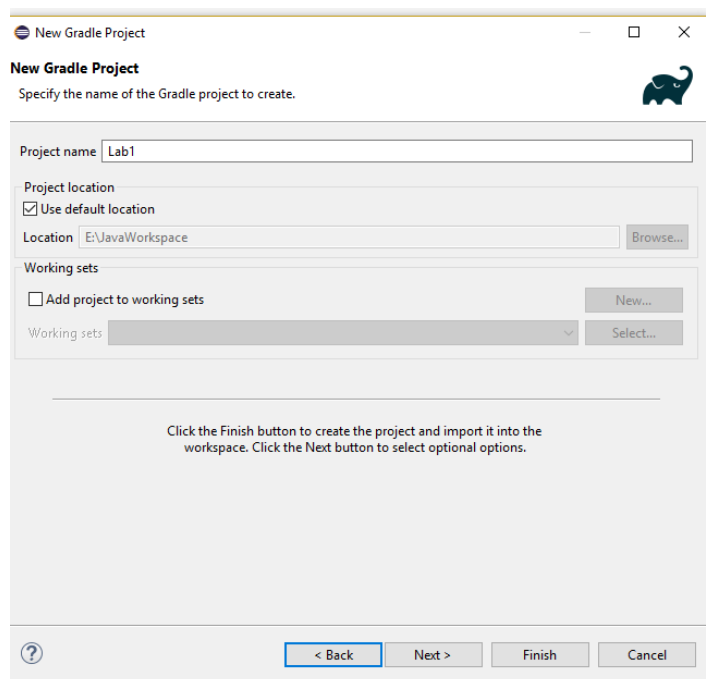
Step 1.1: From the toolbar, **select** Project
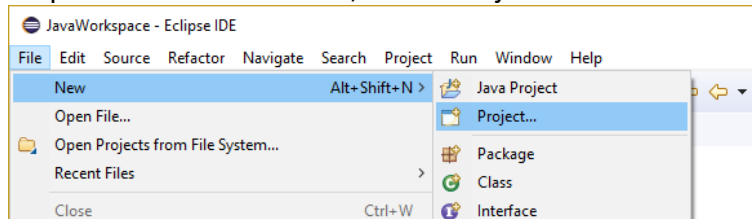
Step 1.2: **Click** Gradle > Gradle Project.

Note: Gradle works like Makefile in C++ so that we can streamline a lot of tasks using Gradle.

Step 1.3: **Type** "Lab1" for the project name and **click** "Finish".

Step 1.4: A new Gradle project should be created for you. It will generate two example java files for you. They are located at src/main/java and src/test/java. Apparently the files stored at src/test/java are for testing purpose. At this moment we should ignore this folder and the files inside. **Compare** your project against the figure.
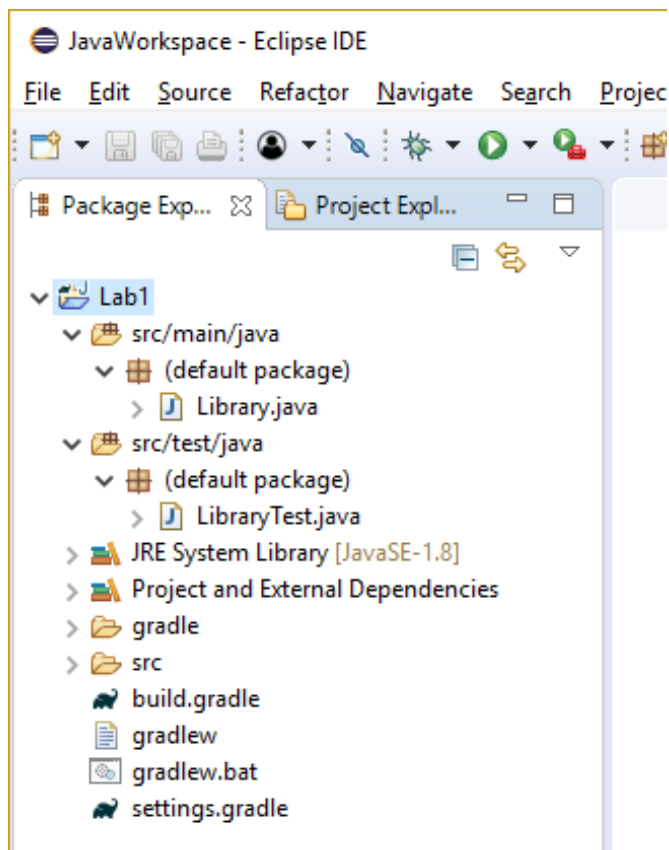
Step 1.5: **Open** Library.java and **add** a function
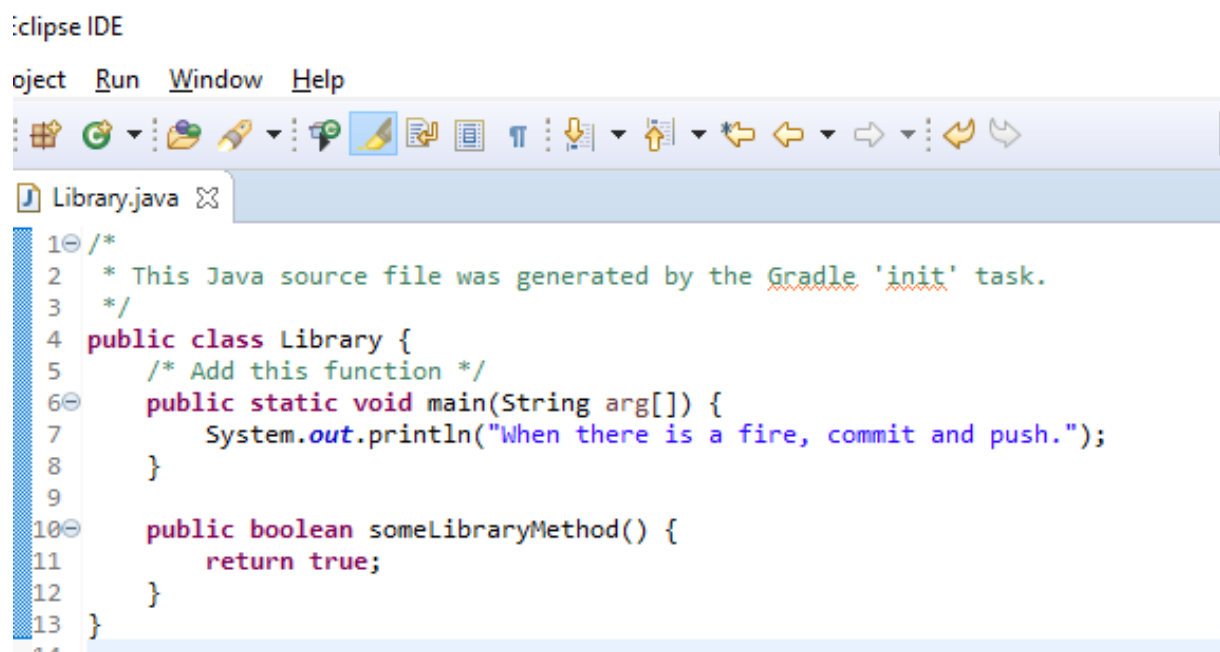
Step 1.6: **Edit** the file build.gradle. Make sure you have **save** both Library.java and build.gradle.

```
 1 plugins {
 2     id 'java'
 3     id 'application'
 4 }
 5
 6 repositories {
 7     jcenter()
 8 }
 9
10 dependencies {
11     // Use JUnit test framework
12     testImplementation 'junit:junit:4.12'
13 }
14
15 mainClassName = 'Library'
```

Step 1.7 **Click** Windows > Show View > Other from the menu bar. **Open** both Gradle Executaions and Gradle Tasks. (Note: you can use the shift key on your keyboard to select two items at a time)
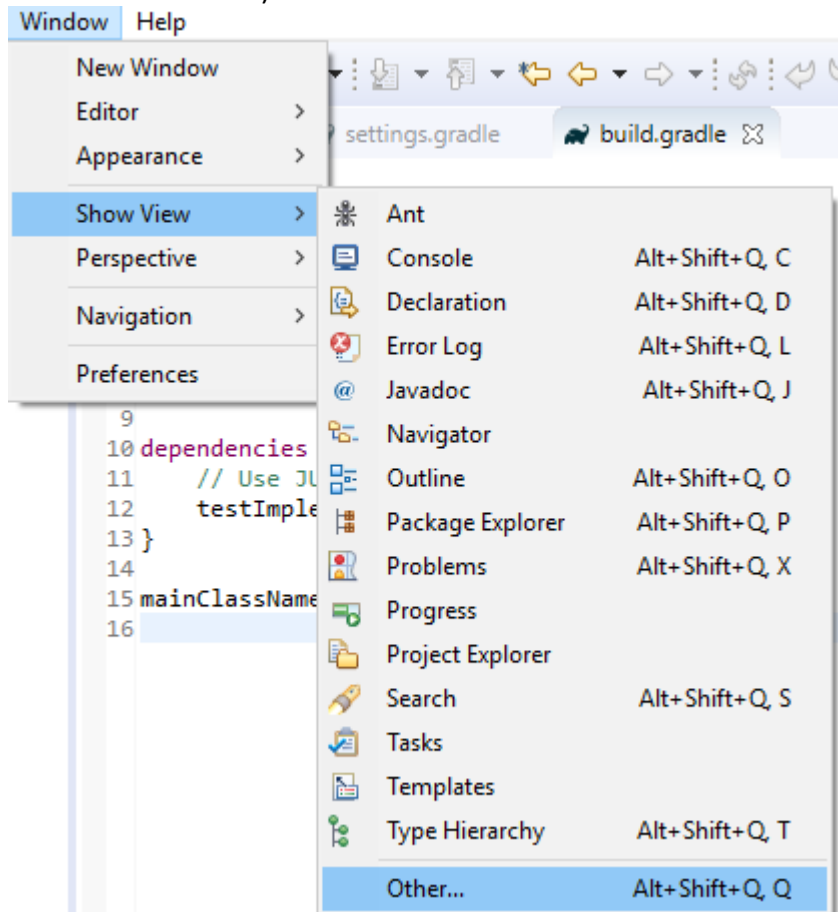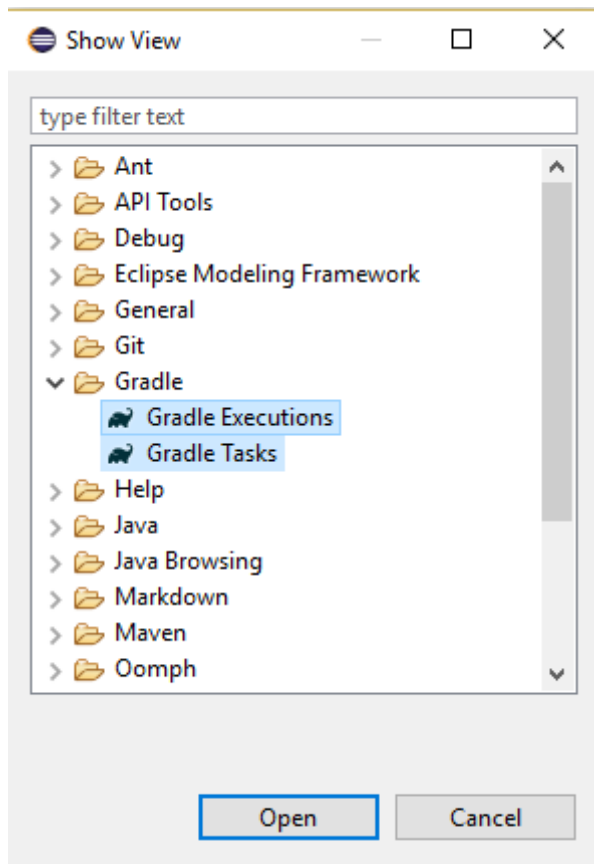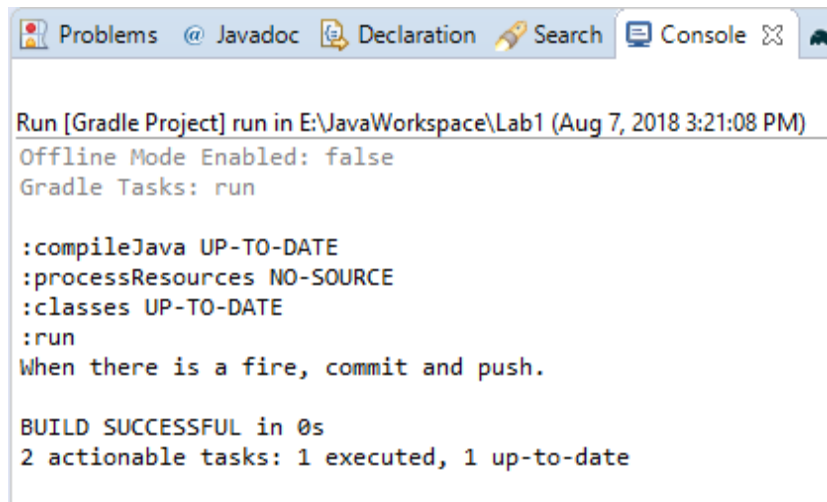
Step 1.7 On the tab Gradle Tasks **click** Application > Run. If you cannot find Application, **click** the refresh button and do it again.
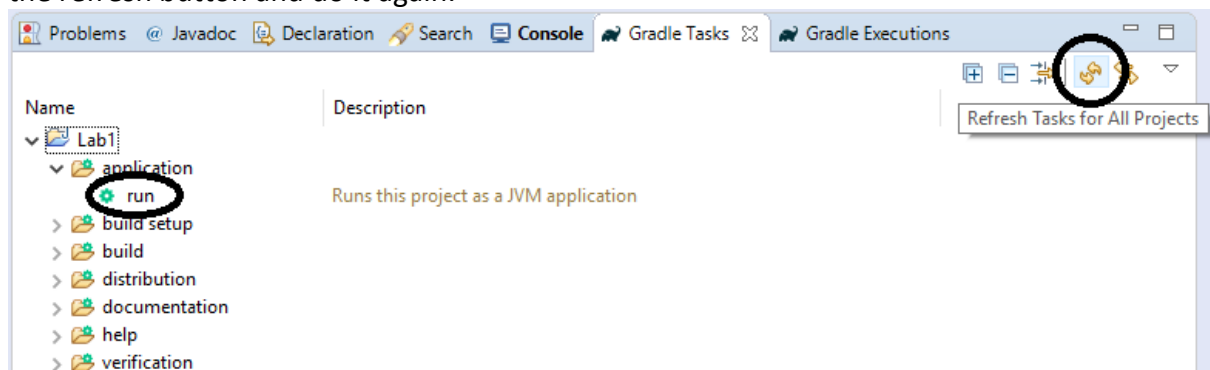
Step 1.8 **Check** your Gradle Executions tab and your Console tab. They should look like:
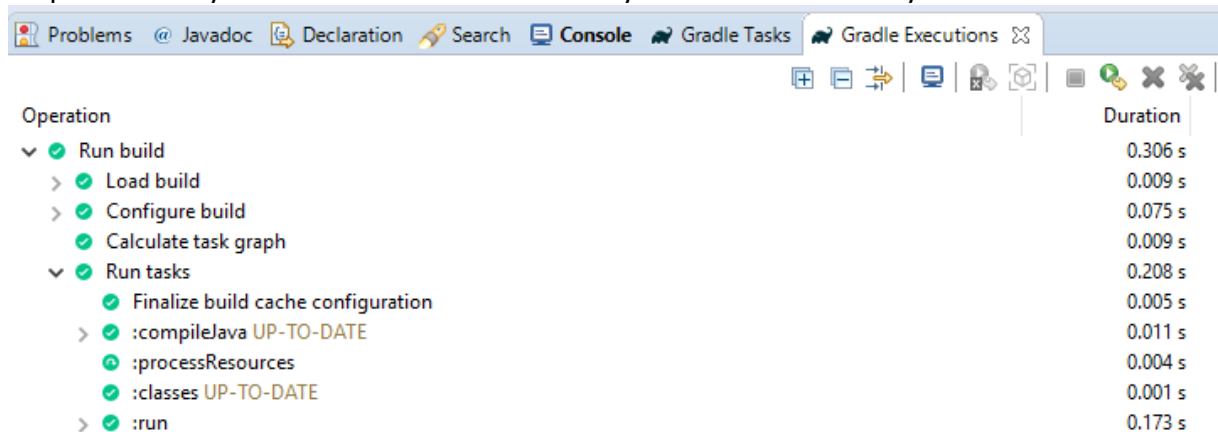
```
Problems  @ Javadoc  Declaration  Search  Console ✕

Run [Gradle Project] run in E:\JavaWorkspace\Lab1 (Aug 7, 2018 3:21:08 PM)
Offline Mode Enabled: false
Gradle Tasks: run

:compileJava UP-TO-DATE
:processResources NO-SOURCE
:classes UP-TO-DATE
:run
When there is a fire, commit and push.

BUILD SUCCESSFUL in 0s
2 actionable tasks: 1 executed, 1 up-to-date
```

Exercise 2: Setup a local Git repository

Step 2.1: From the menu bar, **select** Window > Show View > Other…
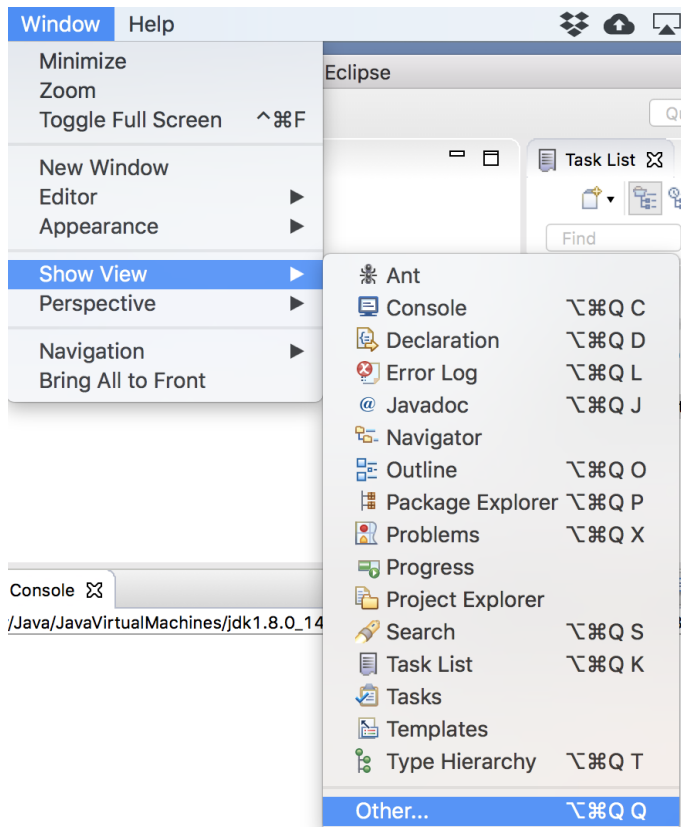
Step 2.2: **Select** "Git Repositories" and "Git Staging". After that, **click** "Open"
(*Hint: You can click the "SHIFT" key to select multiple items*)
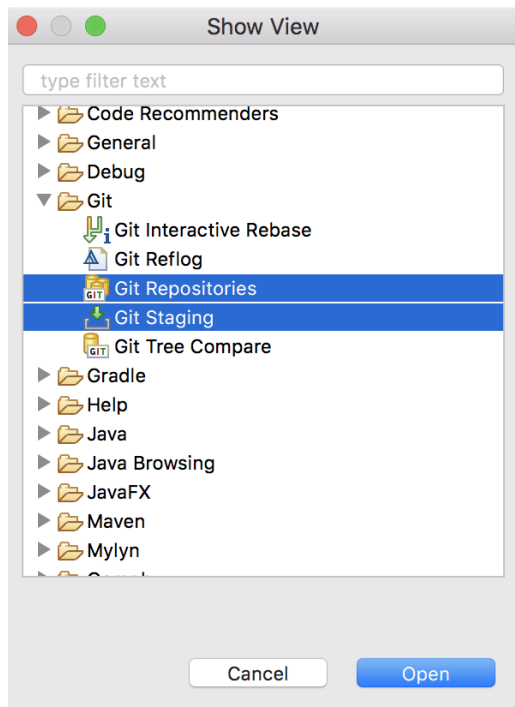
Verification: You are expected to see the following screenshot:
Note: If you accidentally close "Git Repositories" and "Git Staging", follow the above steps to restore the windows
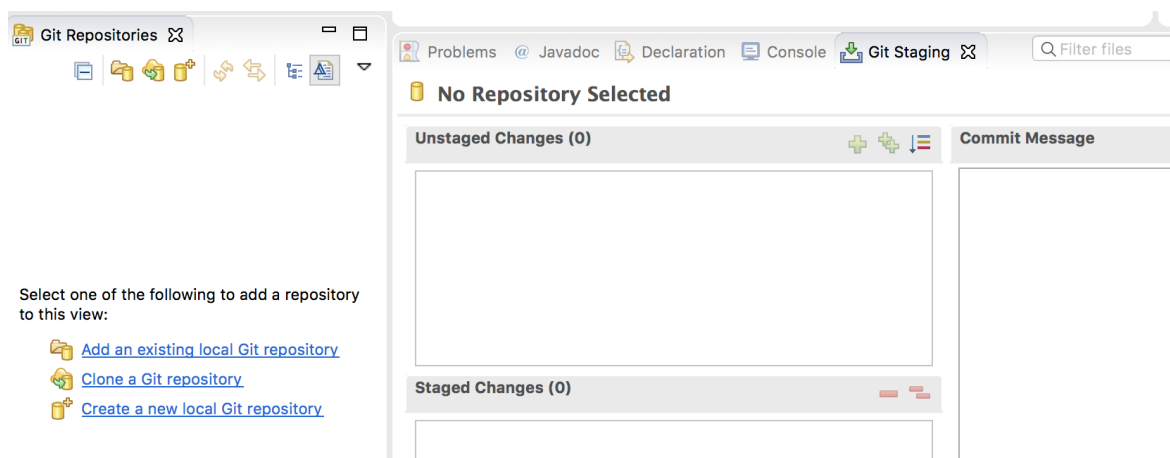
Step 2.3 **Right-click** the project folder, **select** Team > Share Project…
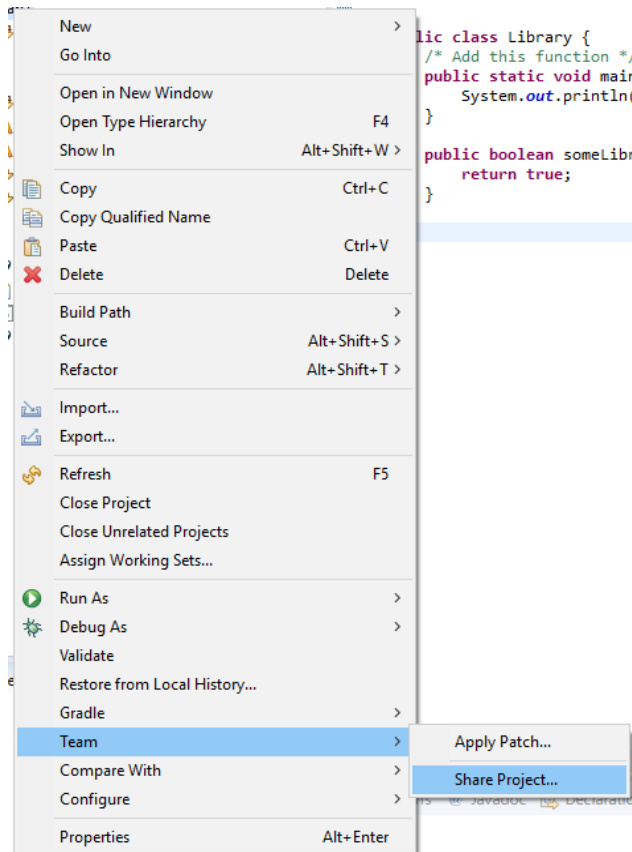
Step 2.4: First, **check** the "Use or create repository in parent folder of project". Second, **click** "Create Repository". A hidden ".git" folder will be created in the project directory with some configuration files. After that, you can **click** "Finish" button
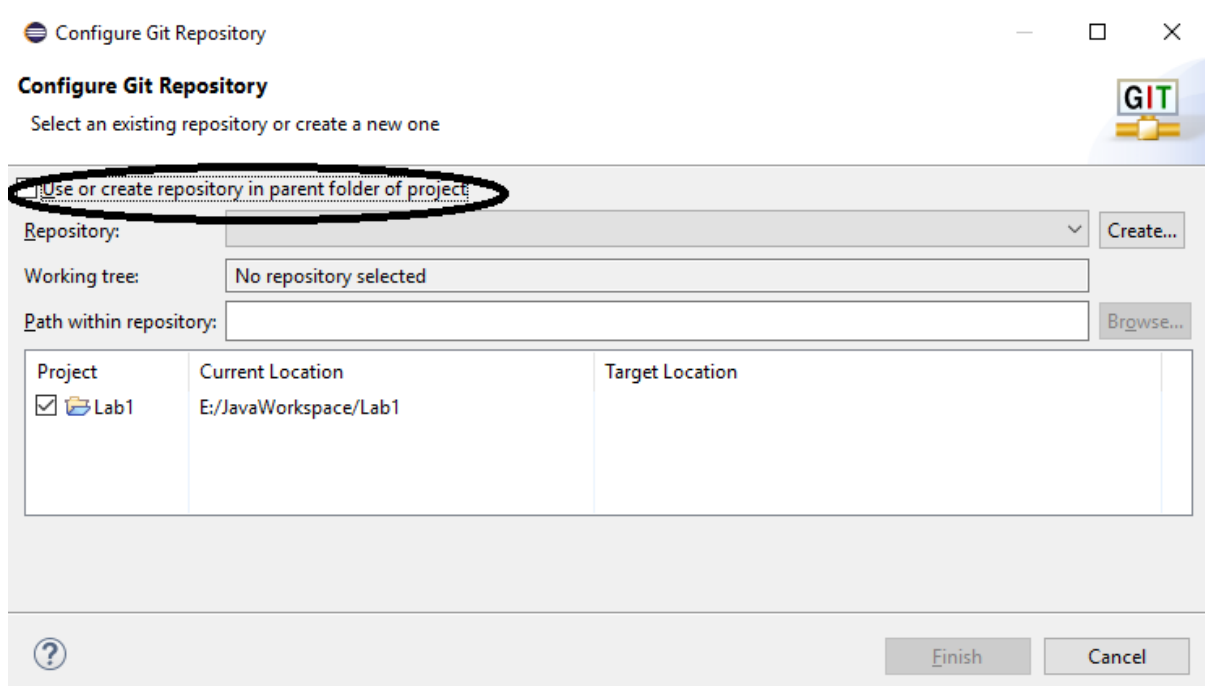
Note: You must click "Create Repository" button first. Otherwise, the "Finish" button will be disabled.
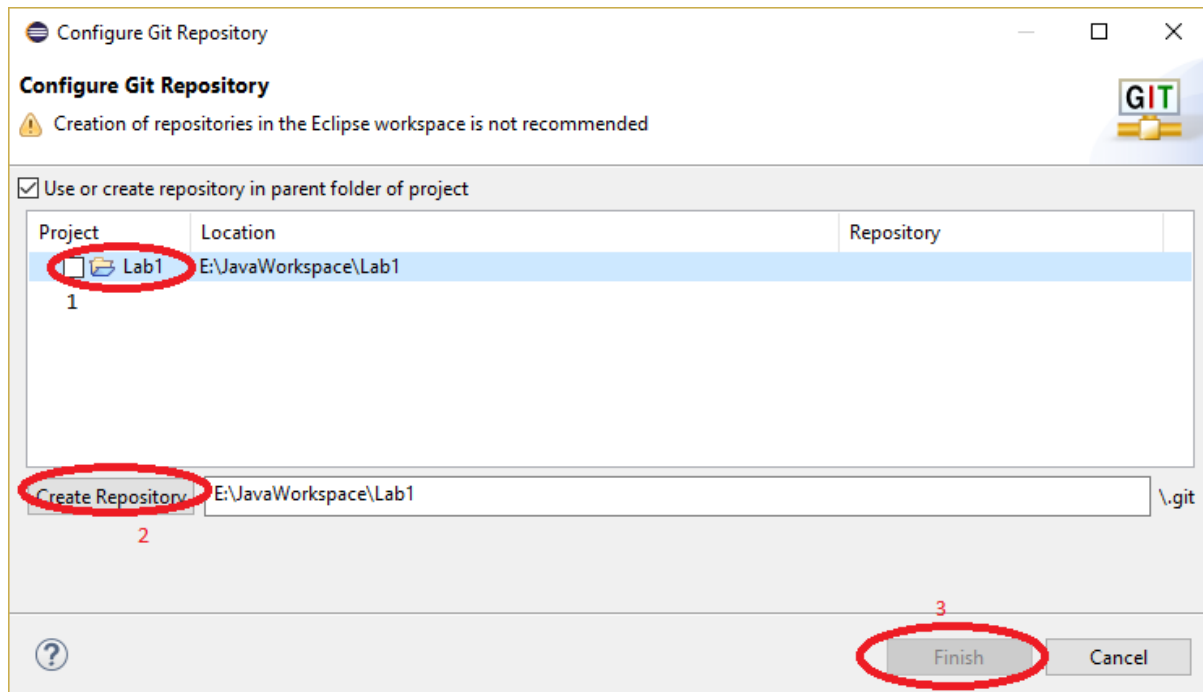
Step 2.5: **Select** Library.java from the Package Explorer. Compare your "Git Staging" windows with the figure below:
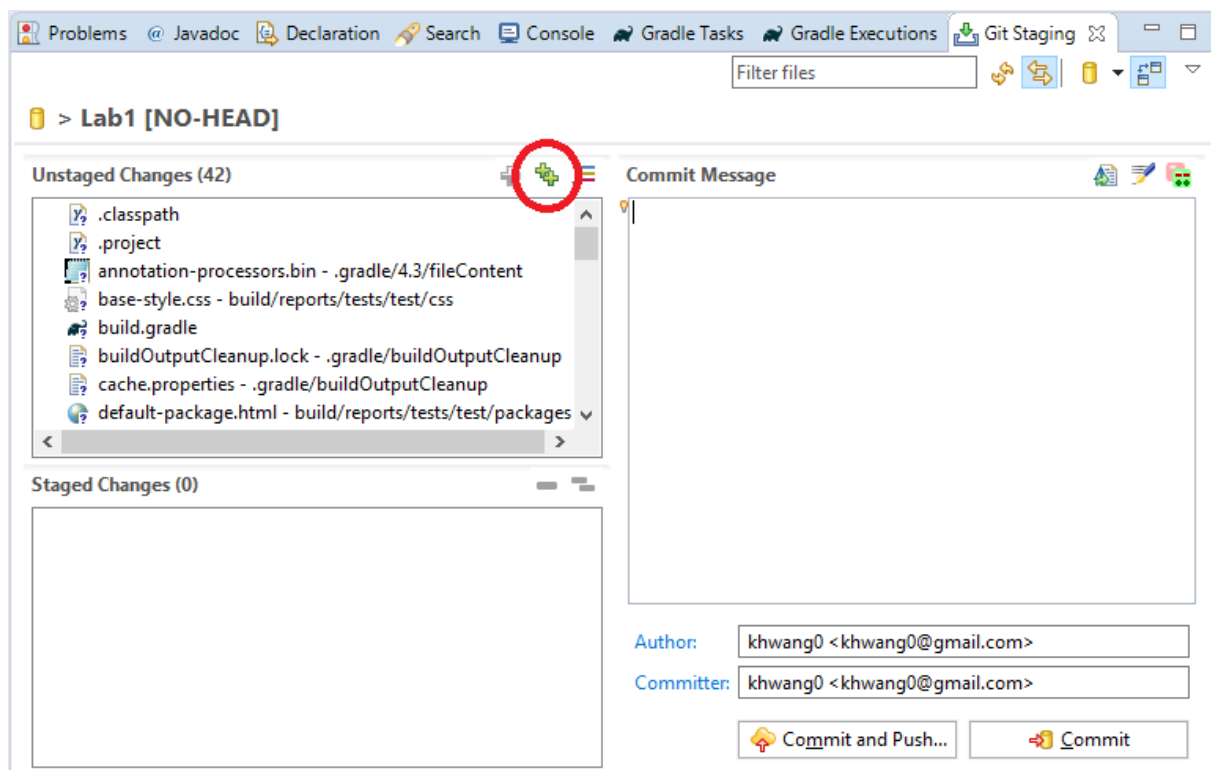
Further explanation:

- *Un-staged changes: Files that are already changed, but not yet ready to be tracked by the repository*
- *Staged Changes: Files that are changed and be ready to be pushed to the repository*
- *Sample usage: For example, you modified 10 files. You are 100% confident on the changes on 9 files, but not very confident the 10th file. In the next commit, you can stage the first 9 files, and not to stage the 10th file.*

Step 2.6: **Click** "Add all files" button (as circled above) on the "Unstaged Changes" tab of the "Git Staging" window. All files should now be staged. **Type** in a meaningful commit message (e.g. Initial project setup). After that, **click** the "Commit" button

Further explanation: Developers should write a short, precise, and meaningful commit message. Otherwise, it will be hard for other developers to understand the commit log.
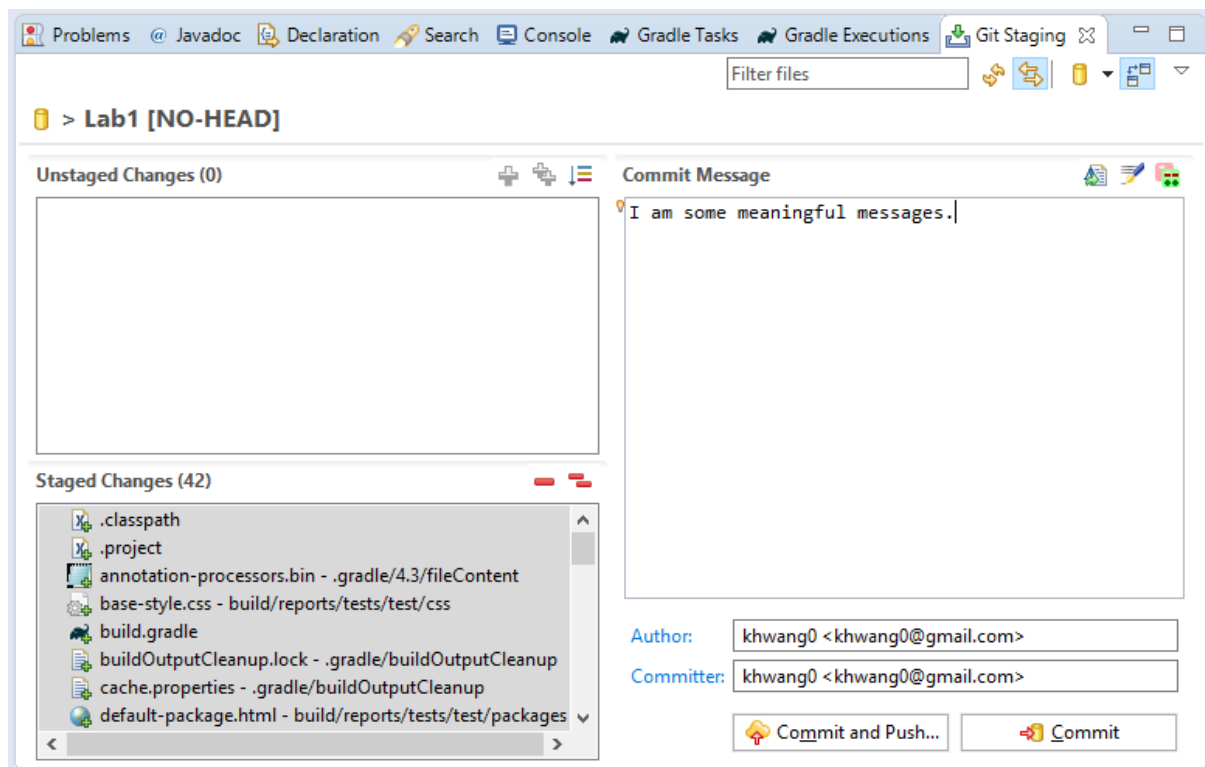
Step 2.7: From Git Repositories, **right-click** the repository and **select** Show In > History
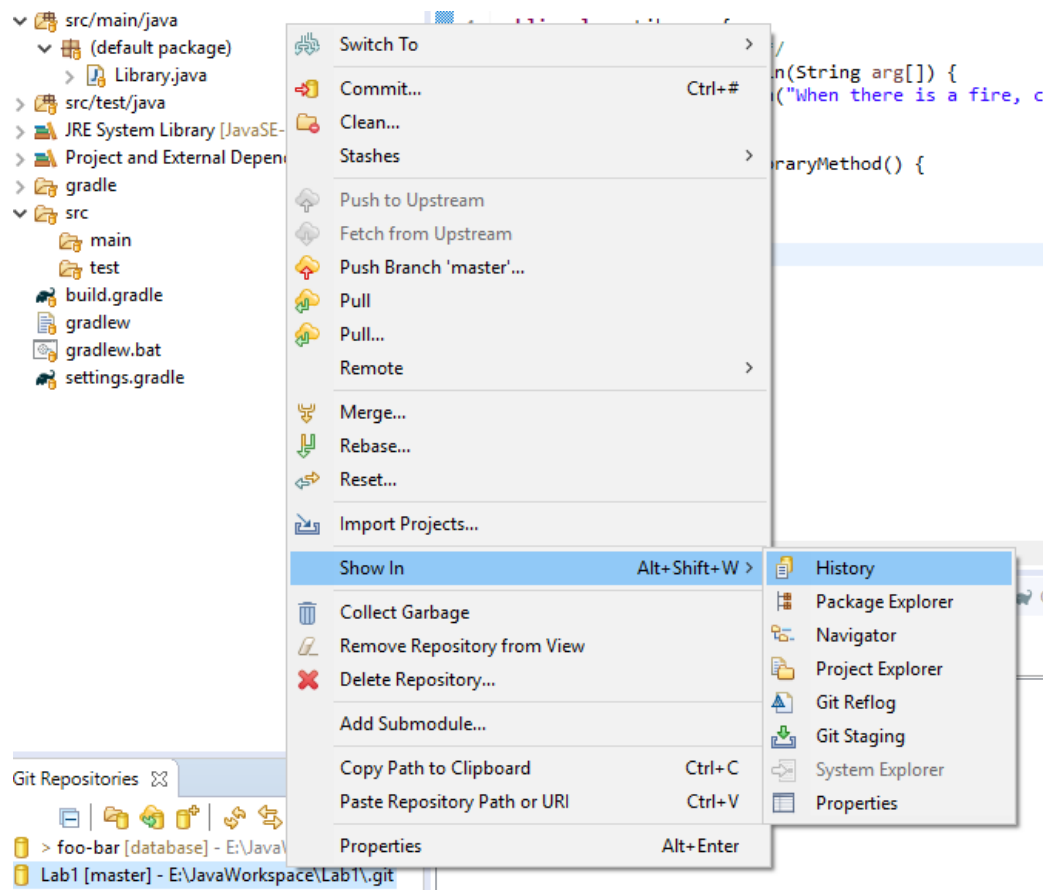
Verification: The following commit log message should be displayed in the History window

Exercise 3: Make and commit changes to the repository

Step 3.1: **Modify** Library.java as follows.

```
 1⊖ /*
 2    * This Java source file was generated by the Gradle 'init' task.
 3    */
 4  public class Library {
 5      /* Add this function */
 6⊖     public static void main(String arg[]) {
 7          Library lib = new Library();
 8          if (lib.someLibraryMethod())
 9              System.out.println("When there is a fire, commit and push.");
10      }
11
12⊖     public boolean someLibraryMethod() {
13          return true;
14      }
15  }
```

Step 3.2: At your Git Staging Tab, **stage** Library.java (by clicking the add button). **Type** in a meaningful commit log message and then **click** "Commit"
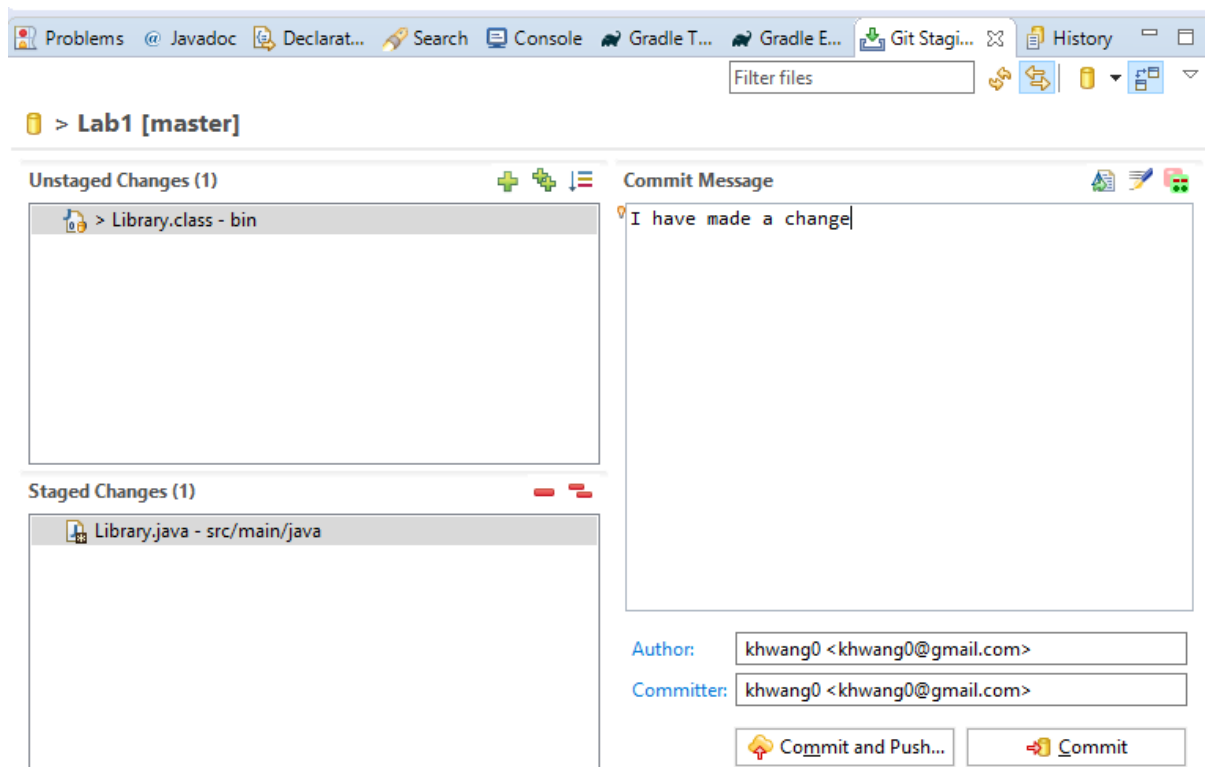
Note: it is not required to stage the file Library.class. It is ok if you have staged it.

Exercise 4: Create a GitHub account and track changes

Step 4.1: **Register/Login** your GitHub account

GitHub (https://www.github.com) is a web-based hosting service for version control using git. It is free for hosting open source projects. Developers may need to pay for a monthly subscription to create private repositories.

For university students, you can apply for GitHub Education (https://education.github.com/) using your university email account. It takes several work days for approval. It provides you the ability to create free private repositories at GitHub. Please apply the education account AFTER the lab.

We recommend students to use GitHub. There exist other web-based Git services (e.g. GitLab, BitBucket). You may need to self-study the setup instructions for these alternative services.

Step 4.2: **Click** + button to create a "New Repository" at GitHub. **Name** the repository and **click** "Create repository"

Note: You should choose a meaningful name for this remote repository

Step 4.3: In the next page, **copy** the link circled.

Step 4.4: **Back** to Eclipse and **right-click** the project folder. **Select** Team > Remote > Push…
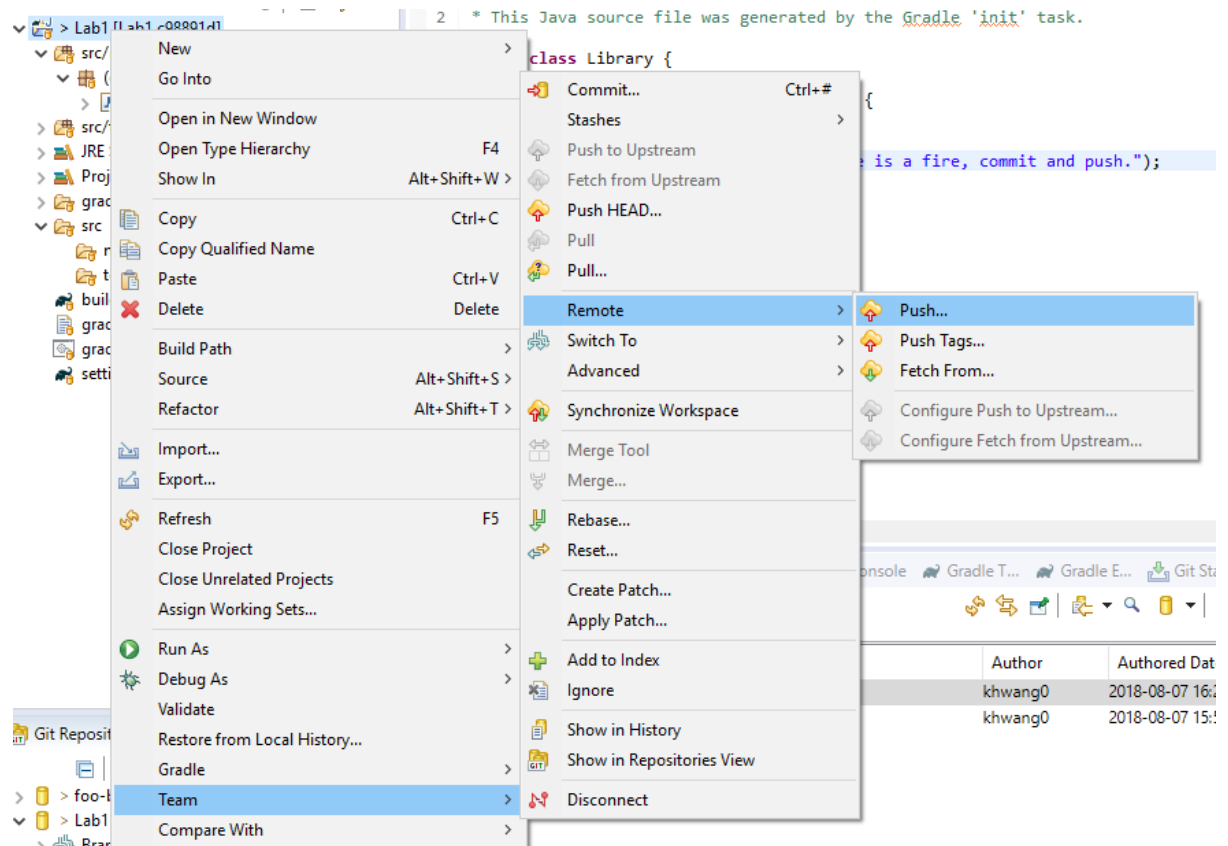
Step 4.5: **Paste** the link in the box URI. **Select** "https" protocol. **Enter** the username and password of your GitHub account and click "Next"

Step 4.7: **Select** the master branches from source and destination ref. **Click** "Add All Branches Spec" (to enable the "Finish" button). After that, click the "**Finish**" button.
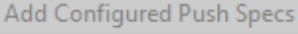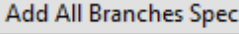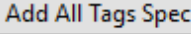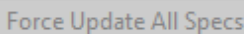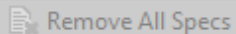
Step 4.8: **Reload** your GitHub webpage. At this point your project has been linked to a remote repository. You don't need your USB thumb drive anymore!

khwang0 / comp3111-lab1-demo

Watch ▾ 0   ★ Star 0   Fork 0

‹› Code   ⊙ Issues 0   Pull requests 0   ▥ Projects 0   ▦ Wiki   Insights   ⚙ Settings

No description, website, or topics provided.   Edit

Add topics

| ⊙ **2** commits | ⸕ **1** branch | ⬡ **0** releases | ⚄ **1** contributor |

Branch: master ▾   New pull request   Create new file   Upload files   Find file   Clone or download ▾

khwang0 I have made a change   Latest commit 89e3da6 24 minutes ago

📁 .gradle	I am some meaningful messages.	an hour ago
📁 .settings	I am some meaningful messages.	an hour ago
📁 bin	I am some meaningful messages.	an hour ago
📁 build	I am some meaningful messages.	an hour ago
📁 gradle/wrapper	I am some meaningful messages.	an hour ago
📁 src	I have made a change	24 minutes ago
📄 .classpath	I am some meaningful messages.	an hour ago
📄 .project	I am some meaningful messages.	an hour ago
📄 build.gradle	I am some meaningful messages.	an hour ago
📄 gradlew	I am some meaningful messages.	an hour ago
📄 gradlew.bat	I am some meaningful messages.	an hour ago
📄 settings.gradle	I am some meaningful messages.	an hour ago

Help people interested in this repository understand your project by adding a README.   Add a README

Lab Activity and Assessment

Lab Activity, you might need to do some research to learn how to do it.

1. **Add** a file called readme.md in your project root.
2. Inside this file, **write down** your name, your student id.
3. Commit with the following message: "Add new readme.md"
4. Push it to github.
5. Delete the file src/test/java/LibraryTest.java
6. Commit with the message: "delete LibraryTest.java"
7. Screen cap the "Git History" panel in your eclipse to reflect the change.
8. Attached the captured screen in the repository and show it in readme.md
9. Commit your captured screen and readme.md and push them to github.
10. Locate the deleted LibraryTest.java on github.

Note
1. To screen cap on Windows, press **PrtScn** button on your keyboard. To do it in mac, press cmd-shift-3 and screenshot file will be saved on your desktop.
2. The captured screen should show four commits. But your github may contain more than that.
3. Do some research to find out how to insert an image in readme.md. Keyword: "markdown image"

Assessment
1. The front page of your Github should be
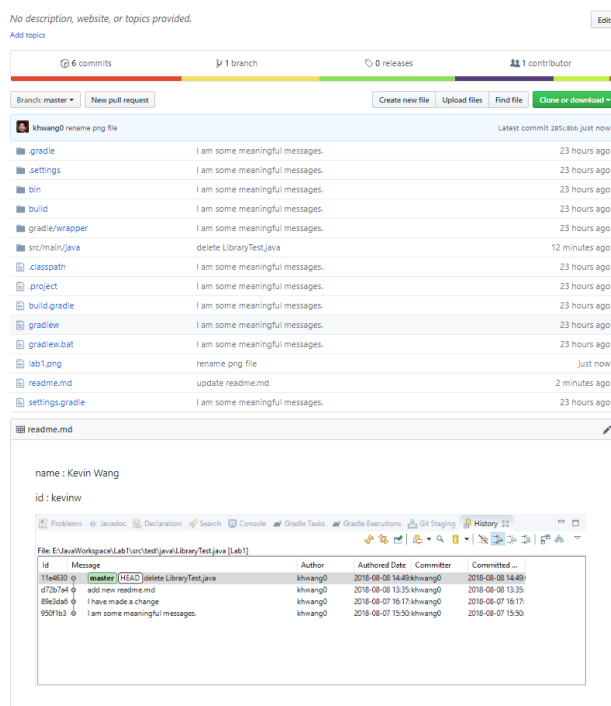
2. There should be no LibraryTest.java on your github folder src/test/java (deleted already). However, try to navigate your github webpage to locate the deleted LibraryTest.java.

Submission

Submit the followings to Canvas by the end of Week 1.

1. Your github repo URL
2. The link to where the deleted LibraryTest.java is located on github.