

Version Control with Git

Objectives:

- Create a repository to hold versioning history of your Java code
- Commit changes to a repository
- Provide descriptive information about each revision
- Clone a repository from an external source

Overview:

If you've ever worked on a big project (either in programming or anywhere outside of programming), you have certainly made modifications to the project along the way and saved those changes. Chances are likely that you've even saved changes and then later wished you had the ability to revert back from those changes. You might even have done something to the project to render it useless and wish that you had at least the last working version before these changes were made. If you haven't experienced this yet in our Computer Science courses, you are likely to do so at some point in the future.

To avoid this problem of losing work we're going to start practicing something called version control. There are many different tools out there to help automate this process, but the one we use here in this department is called Git and there is an Eclipse Plugin called, Git. This exercise will teach you the basics of version control – something that we'll be doing in every exercise for the remainder of the semester.

Installing Git

1. Nothing – it's already a part of the basic Eclipse download. Yea!

Using Git: Creating and Using a Repository

2. Download the RepoLab.zip file from the Homework page on Moodle. Although there are many ways to get your files into Eclipse, the following way is recommended as it will maintain the repository information, if present (this project does not have an initial repository on it, but future exercises may).

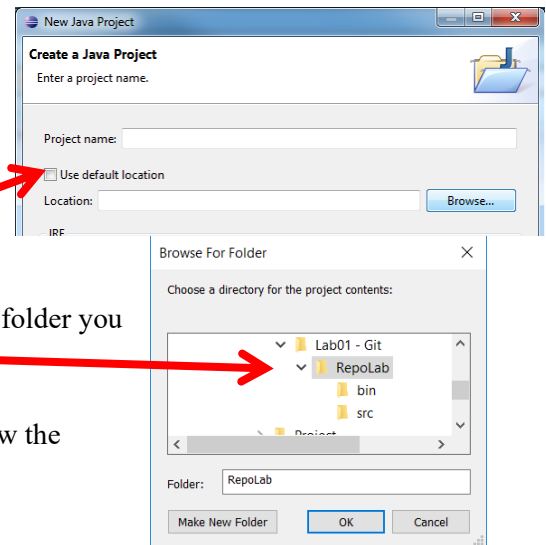
Please be sure to always follow these procedures in this course to avoid problems later.

3. Use 7-Zip to extract the project files from RepoLab.zip.
4. Start Eclipse and open the RepoLab solution files:
File > New > Java Project
5. Uncheck the box 'Use default location' and then Browse inside the folder you just unzipped. Select the RepoLab folder and click **OK**
6. Back in the New Java Project dialog, the Location should now show the folder you just selected. Click **Finish** to open the project.
7. Create package `edu.westga.cs6312.newrepo`
8. In the package, create a class named `Demo` with a `main` method.
9. To be sure that you are using Java 1.9, be sure to place the following line of code inside `main` to create a `List` and immediately initialize it with some values and then display them on the console with a `foreach` loop:

```
List<String> myList = List.of("one", "two", "three");
for (String current : myList) {
    System.out.println(current);
}
```

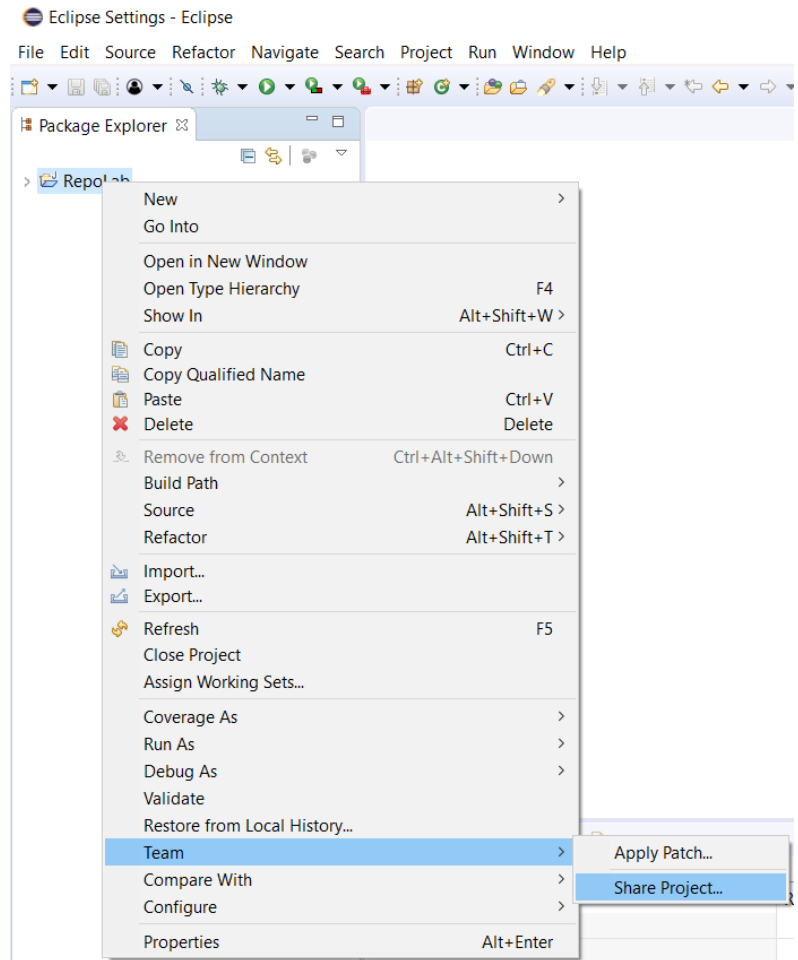
In order to use `List.of`, you'll need to be sure that you have Java 1.9 installed and that Eclipse is set up to use it. If Eclipse is showing you a compiler error, then you'll need to double-check your installation of Eclipse to be sure that (a) you have Java 1.9 installed and (b) you are set up to use Java 1.9 by default. Get this working correctly before moving on.

10. Be sure to add appropriate Javadoc comments and make sure that the code passes Checkstyle before continuing.



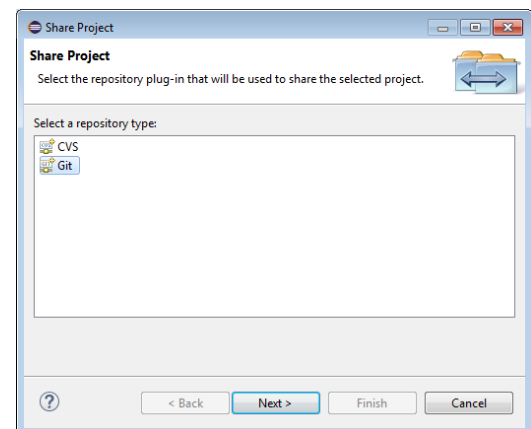
11. Now it's time to create the repository:

- a. Right-click RepoLab in the Package Explorer.
- b. In the popup menu, select **Team** > **Share Project...**

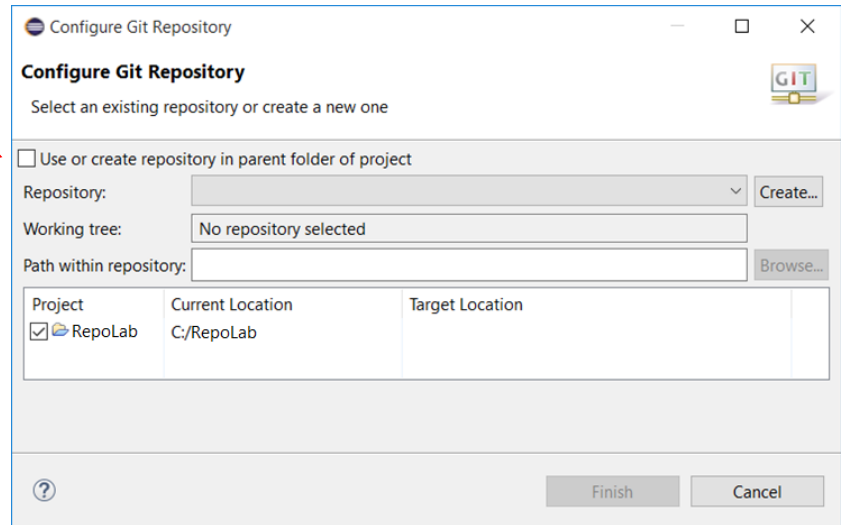


- c. Select **Git** as the repository type, and click **Next** >

(this may not be necessary – your download may not have included CVS, which is fine, you'll just go right to the next step)



- d. In the next dialog, click the checkbox at the top that says "Use or create repository in parent folder of project".



Configure Git Repository

Select an existing repository or create a new one

☐ Use or create repository in parent folder of project

Repository: Create...

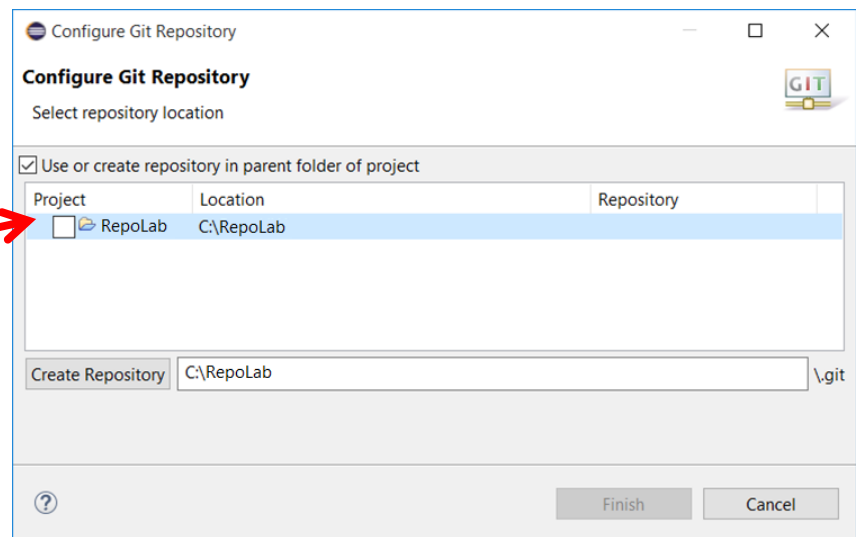
Working tree:

Path within repository: Browse...

Project	Current Location	Target Location
<input checked="" type="checkbox"/> RepoLab	C:/RepoLab	

Finish Cancel

- e. After checking this box, the dialog will change and you'll now see the name of your project and the location. Click to select your Project (note, the Checkbox here won't show a check)



Configure Git Repository

Select repository location

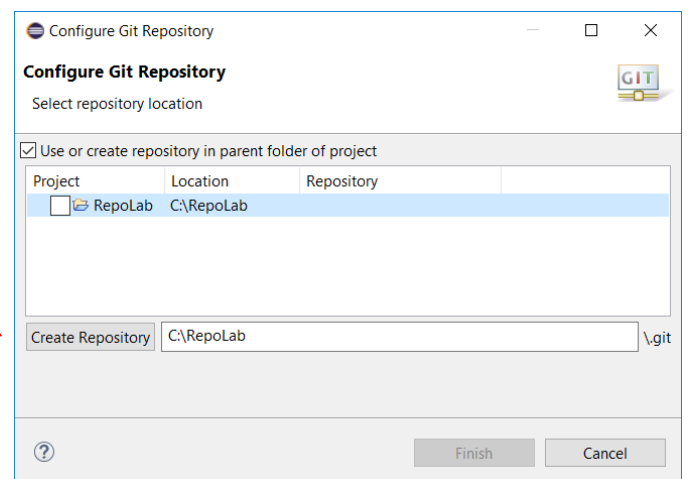
☒ Use or create repository in parent folder of project

Project	Location	Repository
<input type="checkbox"/> RepoLab	C:\RepoLab	

Create Repository \.git

Finish Cancel

- f. After selecting your project, the dialog will change button toward the bottom that says, **Create Repository** should become enabled. Go ahead and click it to create your Repository



Configure Git Repository

Select repository location

☒ Use or create repository in parent folder of project

Project	Location	Repository
<input type="checkbox"/> RepoLab	C:\RepoLab	

Create Repository \.git

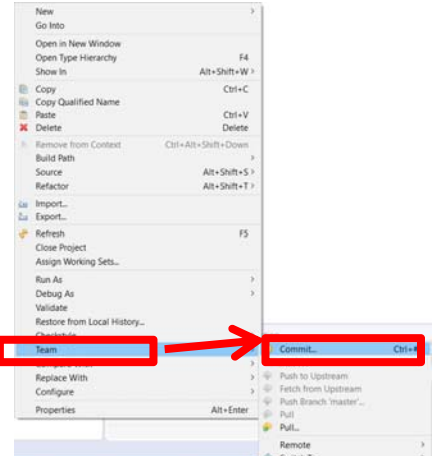
Finish Cancel

- g. Once the repository is created, the checkbox next to the Project name should be checked and the Finish button will become enabled. Complete things by clicking **Finish**

12. The Package Explorer will show a [] tag and icons to indicate that the repository has been created, but that Demo.java has not been added to it.

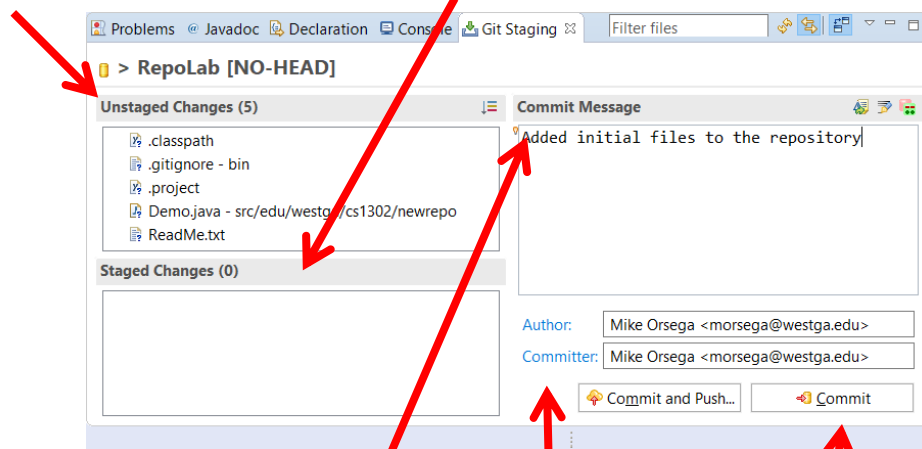
13. Add the files to the repository and commit the changes.

- a. Right-click the project.
On the **Team** menu, select **Commit...**



This will open a new tab in the lower section of the screen called 'Git Staging'

Highlight the files you want to add to the repository (here all of them) and drag them from the Unstaged Changes box to the Staged Changes box.

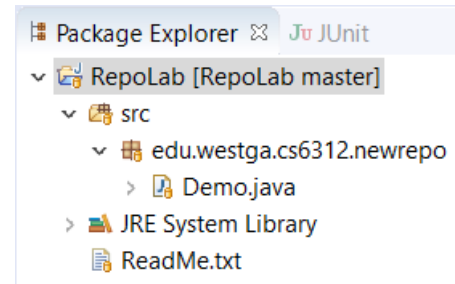


- b. Enter a commit message that explains the changes you've made. In this situation, for example, the message could be **Added initial files to the repository.**

Be sure your first and last name shows in the 'Author' and 'Committer' areas

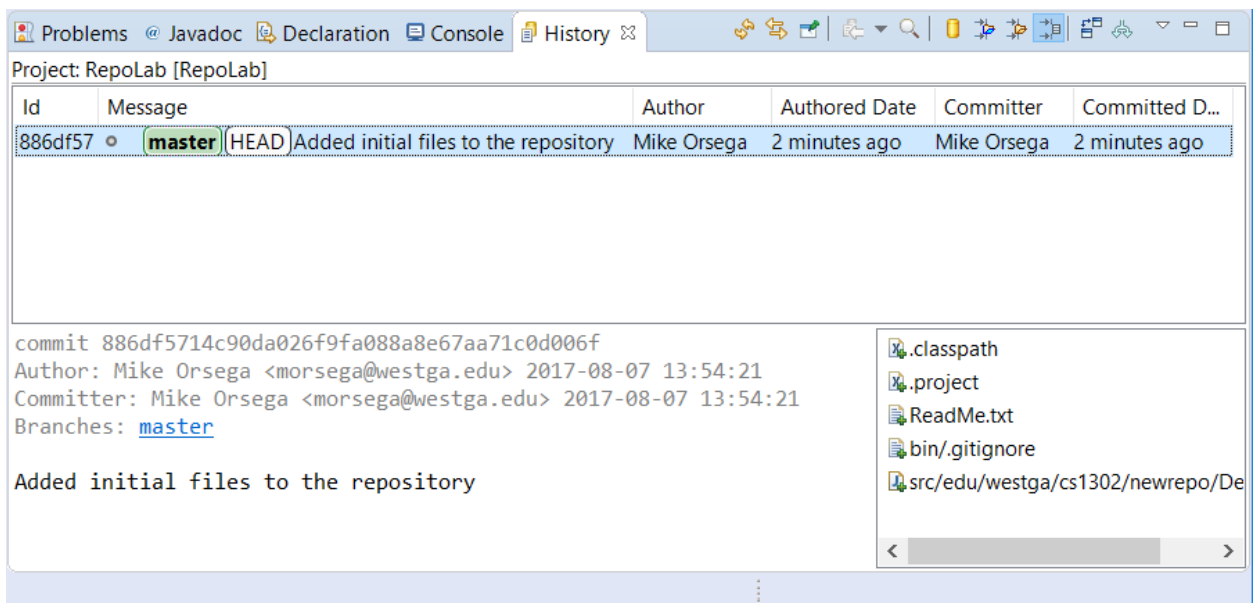
- c. Click **Commit**.

- d. Package Explorer will change to show that the files have been committed to the repository.



14. Look at the history of the repository:

- a. Right-click the **RepoLab** project and select **Team > Show in History**. (near the bottom)
- b. The changeset shows in the *History panel*.



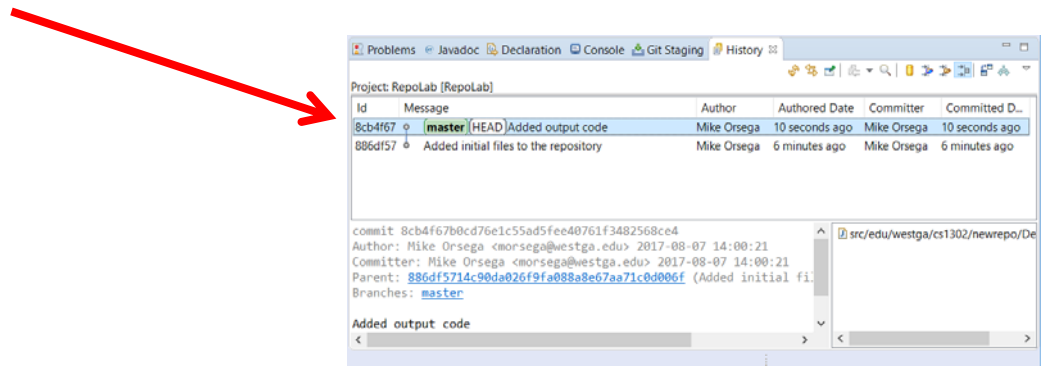
- c. Make a change to class Demo. In main, add a line of code to call to `System.out.println` to display "I love Java!"

15. Save Demo.java.

16. Commit the change to the repository (HINT: if you forgot where this is located, look at Step 8a).

- a. Since it is already in the repository from Step 9, the changed file should be selected. If not, please be sure to check its checkbox.
- b. Write a meaningful commit message to explain the purpose of the commit.
- c. Click **OK**

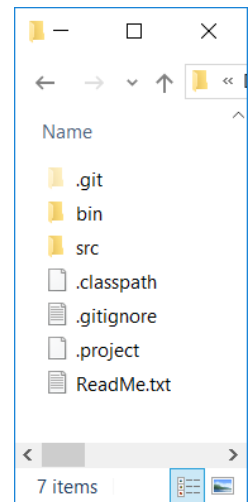
17. Show the repository *History panel* again and select the new changeset.



18. Finally, you might be wondering where all of that data (the old files) are being stored. If you'll minimize Eclipse and open the folder where your Java Project is located, you might see something like:

The folder called `.git` has all of the details. This means that this folder is critical to your repository (essentially, it *is* your repository), so you must be careful to include this whenever you make backups – or submit your work online.

Please know that the `.git` folder is a 'hidden' folder (look at the icon above to see that it's a lighter shade of yellow). It's showing on the screen shot above because I have my machine configured to display hidden files and folders. So if yours isn't showing, that could be why. Please see note in box at the end of this document for tips on checking to make sure repository information is present (with or without being able to see this folder).



19. When you are finished, close Eclipse and use 7-Zip to Zip this folder:

Please be aware that these steps may be different than what you have done in the past. Please follow these steps (all semester long) as other ways of creating a Zip file may actually remove the Repository from your Zip file!

- Use Windows Explorer to locate the Project folder
- Right-click the folder
- Choose **7-Zip > Add to "HW.zip"**

(Mac users will not be able to use 7-Zip, but that's OK because the compression utility built-in to the Mac OS will work just fine here.)

- Rename this new Zip file `6312YourLastNameLab01.zip`

20. Upload your Zip file to the appropriate link in Moodle.

NOTE: It is your responsibility to make sure that the repository information is stored in the Zip file appropriately. To confirm that it is there, after uploading, you should download this file to another location on your machine (or possibly use a different machine) and opening it in Eclipse to confirm that the repository information is there. If you can't see any repository information in this newly unzipped file, then we won't be able to see it when we grade your work (**and you will NOT receive credit for it**).

Also be aware that for the rest of this semester, *all* grades will have a component based on your repository. Be sure you're able to get this working and/or ask for help right away so you don't lose points.

HINT: In order to avoid losing points in this and future exercises, please be sure to:

- Make a repository at the beginning. Version control is a critical part of a professional software developer's life. Please get in the habit of doing this from now on, without being asked.
- Use descriptive, professional-sounding language in your commit messages. This is going to help anyone who looks at your repository to figure out what happened and where it happened.
- Make commits at appropriate times. Certainly we don't need a commit after each line of code is written. But we also don't want to have only two commits (like this exercise) with one at the start and one at the end. The idea is that once you get a piece of code written and tested, that's a good time for a commit. Again, we want to practice iterative development: write just a little code, test it, make sure it works, commit, and then move ahead to the next little piece.
- Use 7-Zip to compress/decompress your Zip files. Yes, I know there are easier ways to get this stuff into Eclipse, but I need to be sure that Repository survives the process. 7-Zip is guaranteed to do this for you.
- When re-opening an Eclipse project that already has a repository, you will need to follow the directions that we've been teaching all along (use Eclipse's menu system to choose **File > New > Java Project** and then browsing to the location where the *unzipped* file is located. Again, we're aware that there are other methods for opening Eclipse projects – the way we teach *will* work, others may or may not. Please be careful.