

IntelliJ, Maven, JUnit Submitting Assignments

CSC416/565: Design and Implementation of Compilers

Utilize our CSC416/565 discord for your questions!

Installing IntelliJ

Platforms and Versions

- [IntelliJ](#) can be installed on:

- Windows
- Mac
- Linux



- Versions:

- **Ultimate** – free for students
- **Community Edition (CE)** – open source, and free for all, **this is what I have installed**
 - Scroll down on the download page to find CE

Experimenting w/ IntelliJ and Maven

IntelliJ

- IntelliJ is just like any other full-featured IDE (Eclipse, NetBeans, Visual Studio, Visual Studio Code w/ plugins), but there is a little bit of a learning curve as to where things are
 - *The more you use a new IDE, the more familiar you will be!*
 - There's no short cut here except to spend time playing around.
- We'll also be utilizing **Maven** project management and **JUnit** in our projects

Maven

- [Maven](#) helps control the lifecycles of Java projects, and standardizes where source files, compiled files, libraries, etc., are all kept



Maven – pom.xml configuration file

- A single configuration file that contains project **properties** and **dependencies**
- I'll be writing this file for you. You won't have to modify it. But feel free to look at it to learn more about Maven!

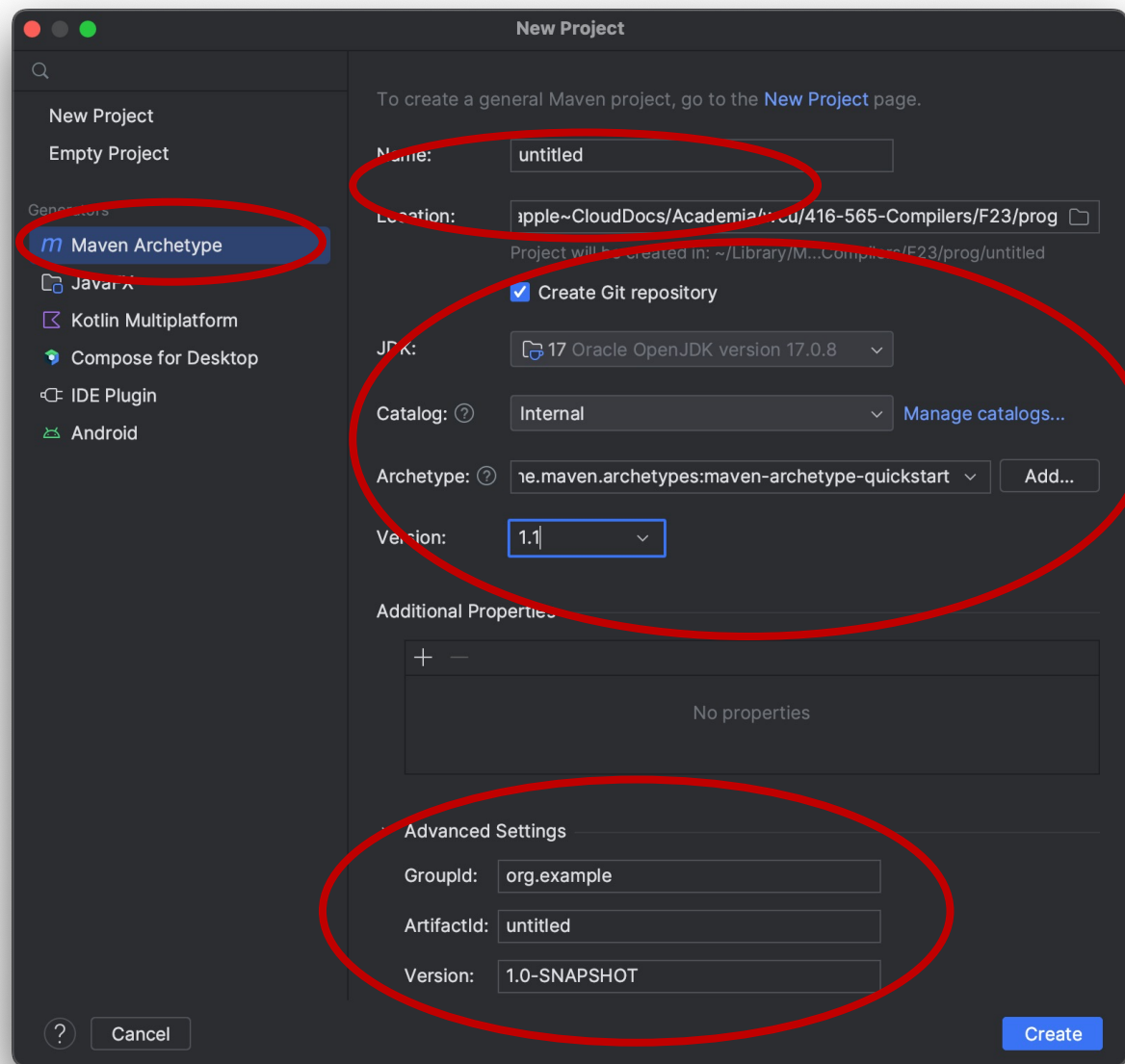
	<pre>1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 2. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> 3. <modelVersion>4.0.0</modelVersion> 4. 5. <groupId>com.mycompany.app</groupId> 6. <artifactId>my-app</artifactId> 7. <version>1.0-SNAPSHOT</version> 8. 9. <properties> 10. <maven.compiler.source>1.7</maven.compiler.source> 11. <maven.compiler.target>1.7</maven.compiler.target> 12. </properties> 13. 14. <dependencies> 15. <dependency> 16. <groupId>junit</groupId> 17. <artifactId>junit</artifactId> 18. <version>4.12</version> 19. <scope>test</scope> 20. </dependency> 21. </dependencies> 22. </project></pre>
Name of project	
Which java version is required?	
This project uses Junit 4.12 which is also needed.	

Maven – Standard Project Structure

Under this directory you will notice the following [standard project structure](#).

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   |-- java
    |   |   |-- com
    |   |   |   |-- mycompany
    |   |   |   |   |-- app
    |   |   |   |   |   App.java
    |-- test
    |   |-- java
    |   |   |-- com
    |   |   |   |-- mycompany
    |   |   |   |   |-- app
    |   |   |   |   |   AppTest.java
```

Creating a New Maven Project in IntelliJ



The screenshot shows an IDE interface with a dark theme. The top bar includes window controls, a tab labeled 'U untitled', and a dropdown menu set to 'main'. The right side of the top bar contains icons for 'Current File', a play button, a bug icon, and a settings icon. Below the top bar, the 'Project Explorer' on the left shows a project structure: 'untitled' (path: ~/Library/Mobile Documents/com~apple~CloudDocs/Aca...), containing '.idea', 'src' (with 'main' and 'java' subfolders), 'test' (with 'java' subfolder), '.gitignore', and 'pom.xml'. The 'App' class is selected in the 'main/java/org.example' path. The main editor displays the code for 'App.java' (package org.example; /** * Hello world! */ new * public class App { new * public static void main(String[] args) { System.out.println("Hello World!"); } }). The bottom panel shows a 'Run' tab with a successful execution of 'org.apache.maven.plugins:maven-archetype-plugin:RE...' (12 sec, 255 ms). The console output includes several '[WARNING]' messages about the maven-archetype-plugin version (3.2.1) and a final message 'Process finished with exit code 0'. The bottom status bar shows the current file path 'untitled > src > main > java > org > example > App', the time '7:14', and encoding 'LF UTF-8 4 spaces'.

Project Explorer (highlighted):

- untitled ~/Library/Mobile Documents/com~apple~CloudDocs/Aca...
- .idea
- src
 - main
 - java
 - org.example
 - App
 - test
 - java
 - org.example
 - AppTest
 - .gitignore
 - pom.xml- External Libraries
- Scratches and Consoles

Code Editor (highlighted):

```
1 package org.example;
2
3 /**
4  * Hello world!
5  *
6  */
7 new *
8 public class App
9 {
10     new *
11     public static void main( String[] args ) { System.out.println( "Hello World!" ); }
12 }
13
14
```

Run Tab:

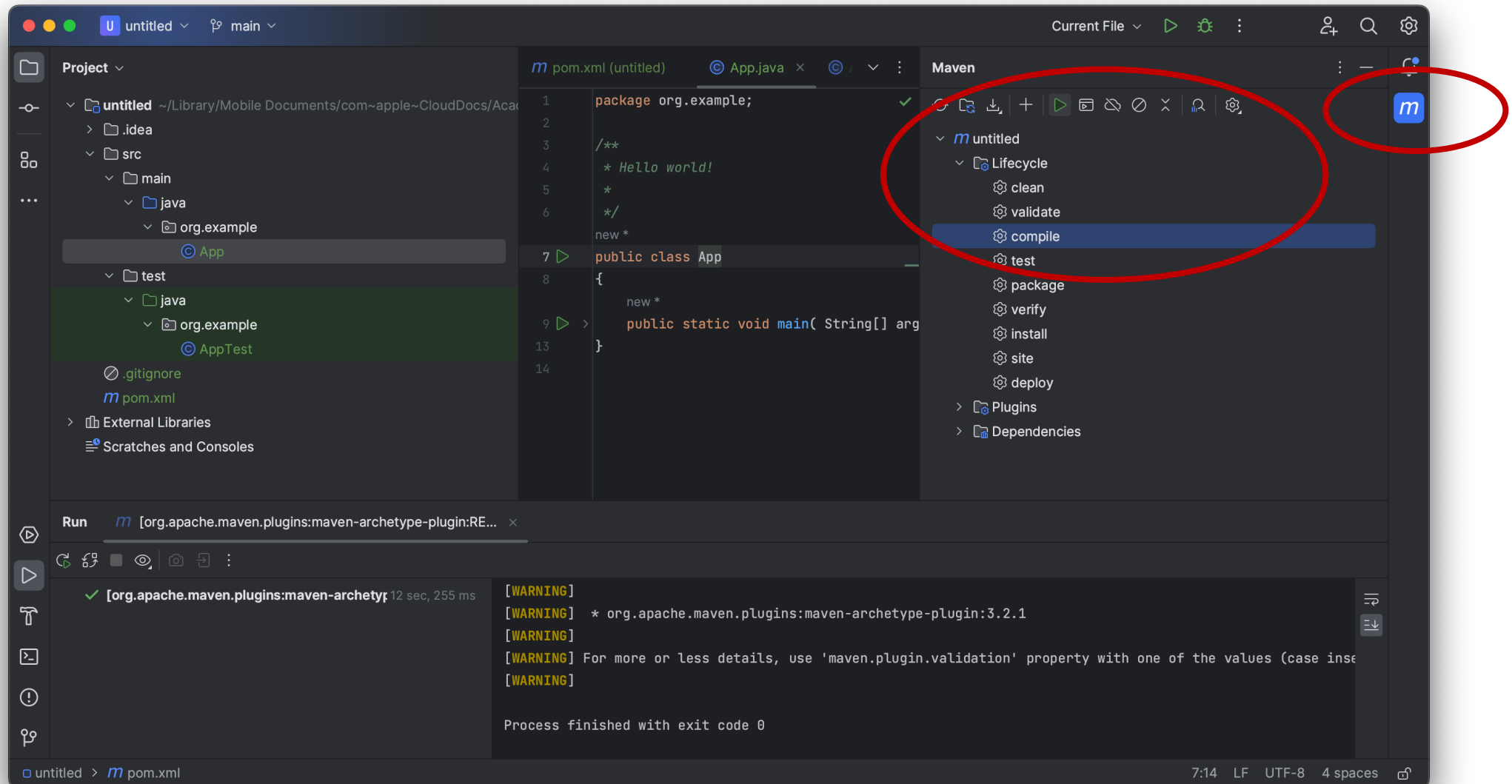
Run [org.apache.maven.plugins:maven-archetype-plugin:RE...]

Console Output:

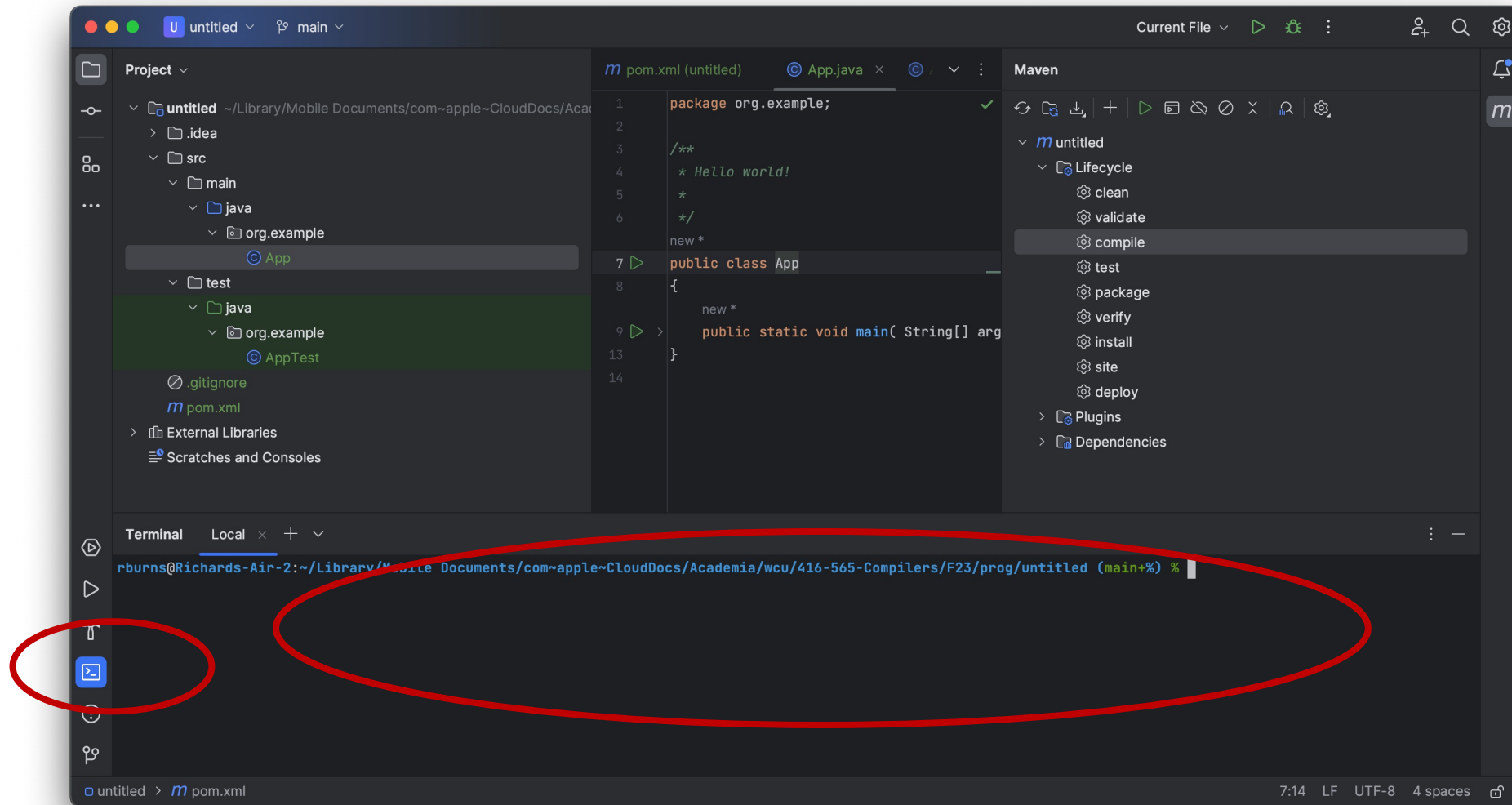
```
[WARNING]
[WARNING] * org.apache.maven.plugins:maven-archetype-plugin:3.2.1
[WARNING]
[WARNING] For more or less details, use 'maven.plugin.validation' property with one of the values (case insensitive)
[WARNING]

Process finished with exit code 0
```

Running Maven Lifecycle Phases



Opening a Terminal

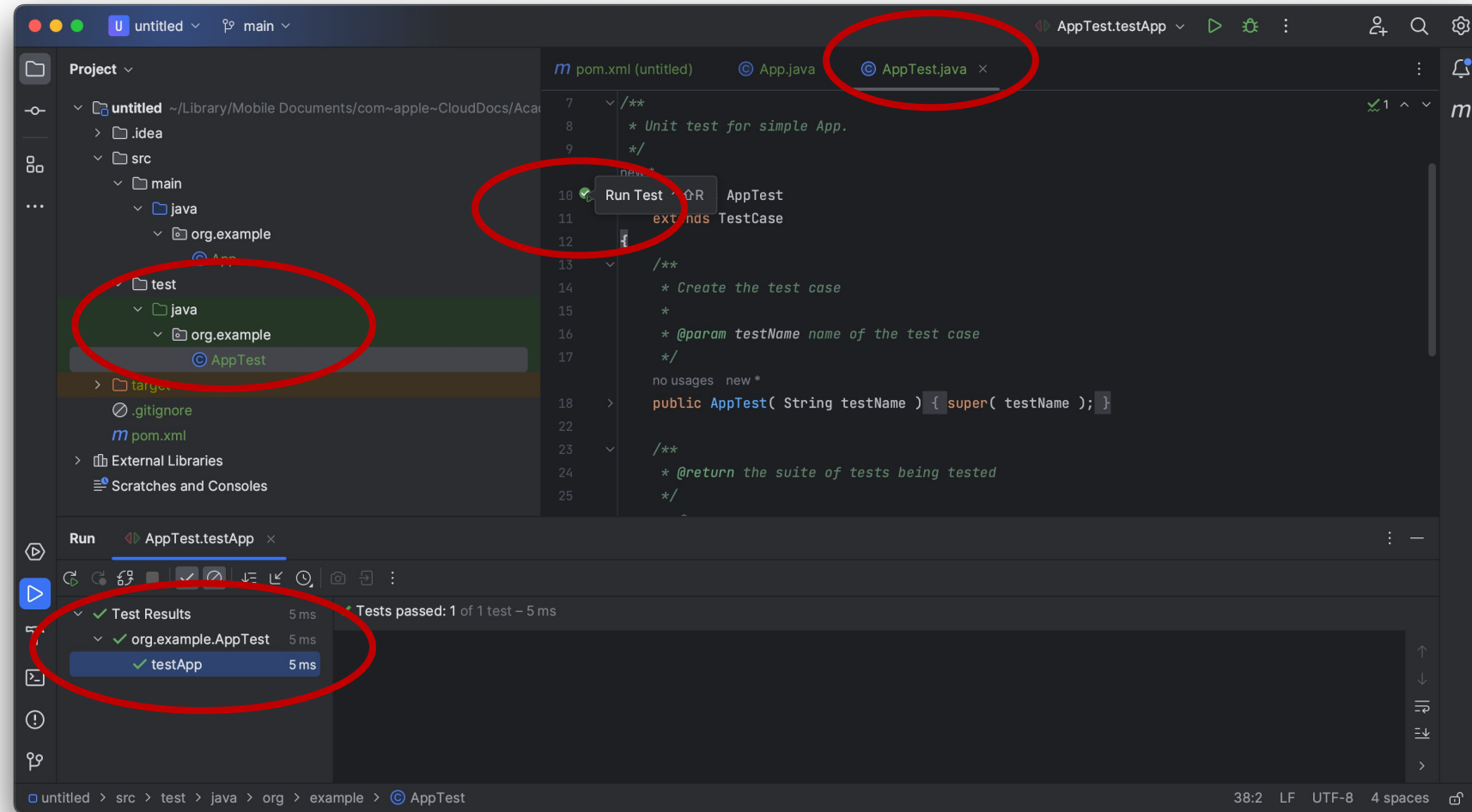




JUnit

- One of many popular **unit testing frameworks** for Java
- Will be using [JUnit 4.13](#) in this course
 - Specified and configured in the Maven `pom.xml` file
 - No need for you to manually download and install JUnit
- *Note:* JUnit 5 also exists
- I will be writing your unit tests, so you do not create JUnit code yourself

Using JUnit in IntelliJ



The screenshot displays the IntelliJ IDEA IDE interface. The left sidebar shows the Project view with a tree structure: `untitled` (root) → `.idea` → `src` → `main` → `java` → `org.example` → `App`. The `test` directory is also visible under `src`. The central editor shows the `AppTest.java` file, which contains the following code:

```
7  /**
8   * Unit test for simple App.
9   */
10 new *
11 public class AppTest
12     extends TestCase
13 {
14     /**
15      * Create the test case
16      *
17      * @param testName name of the test case
18      */
19     public AppTest( String testName ) {
20         super( testName );
21     }
22
23     /**
24      * @return the suite of tests being tested
25      */
26     public static Test suite() {
27         return new TestSuite( AppTest.class );
28     }
29 }
```

The right sidebar shows the Maven build lifecycle, with the `test` goal selected. The bottom status bar shows the Run configuration `AppTest.testApp` and the Test Results panel, which indicates that 1 of 1 tests passed in 5 ms.

Using JUnit in IntelliJ with Maven

Submitting Assignments

GitHub Classroom



Prerequisite:

- A GitHub account – sign up for one if needed!
 - Probably not necessary, but consider adding your WCUPA email address as one of the emails associated with the account

No prior Version Control experience necessary!

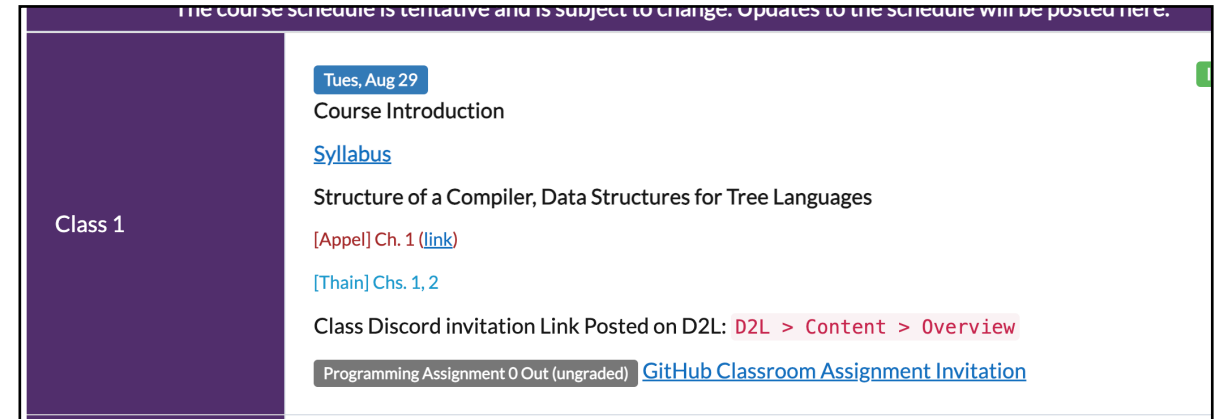
In CSC416/565:

- You will accept “assignments” that are hosted in GitHub Classroom.
- Immediately GitHub Classroom will prompt you to link your GitHub account with your identifier in the class roster
 - *(Thus, I will know who you are, despite whatever your GitHub is.)*

“Programming Assignment 0”

- Before we use GitHub in real, graded assignments, we’ll practice it in a “Programming Assignment 0”
- Like a real HW:
 - There are small tasks to do
 - There is a deadline
 - The assignment is auto marked correct or incorrect w/ unit tests
 - There is Java/IntelliJ/Maven/JUnit/Git/GH practice
- But:
 - It’s not graded. (Doesn’t count for anything.)

Step 1: Login to GitHub



Step 2: Follow “GitHub Classroom Assignment Invite” link

Join the classroom:

WCU-CSC345-S22-classroom

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? [Skip to the next step →](#)

WCU-CSC345-S22-classroom

Accept the assignment —

Homework 0

Once you accept this assignment, you will be granted access to the `homework-0-burnsrichard-student` repository in the **WCU-CSC345-S22** organization on GitHub.

[Accept this assignment](#)



You accepted the assignment, **Homework 0** . We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates.

📅 Your assignment is due by **Feb 3, 2022, 23:59 EST**



You're ready to go!

You accepted the assignment, **Homework 0**.

Your assignment repository has been created:

📄 <https://github.com/WCU-CSC345-S22/homework-0-burnsrichard-student>

We've configured the repository associated with this assignment ([update](#)).

📅 Your assignment is due by **Feb 3, 2022, 23:59 EST**

main 1 branch 0 tags

Go to file

Add file

Code



burnsrichard tests updates

✖ a16a543 2 hours ago 14 commits

.github	tests updates	2 hours ago
src	Update App.java	14 hours ago
.gitignore	Initial commit	14 hours ago
README.md	Initial commit	14 hours ago
pom.xml	Initial commit	14 hours ago

☰ README.md



Programming Assignment 0

Date Out: Tuesday, August 29

Due Date: Tuesday, September 12 (start of class)

This introductory programming assignment will help familiarize you with the development environment of this course:

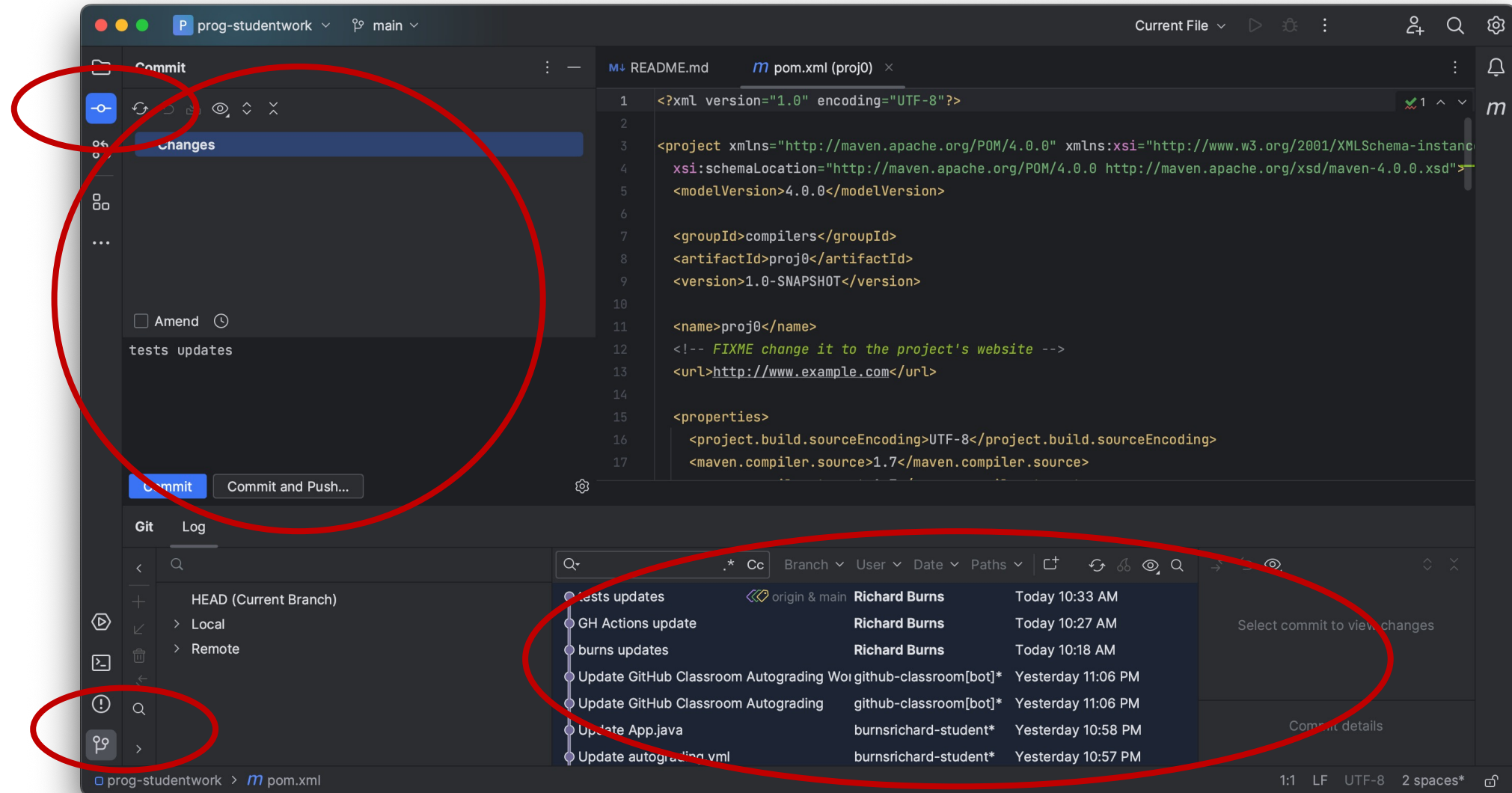
- Java for programming
- IntelliJ CE (community edition) IDE for coding, debugging, and more!
- Maven, a project manager for specifying project structure and dependencies

Next Steps

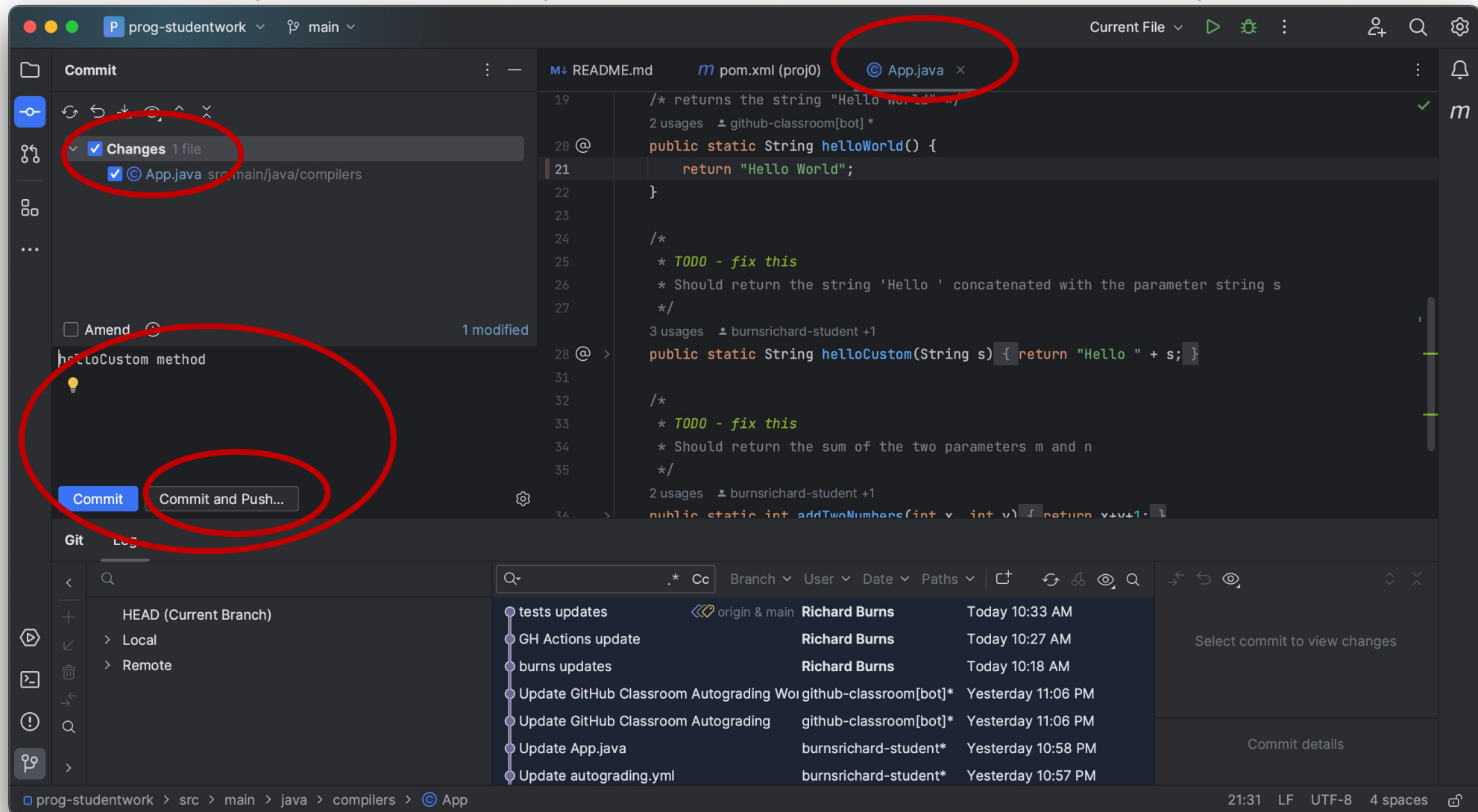
See the README.md (assignment description)

1. Open IntelliJ. Create a “New” -> “Project from Version Control”.
2. This will *clone* your *remote GH repo* onto your computer.
3. Write / test the code.
4. Upload/commit the code to your GitHub Classroom repository.
5. Look at and inspect the results of your GitHub auto-graded actions.

Support for Git and version control is already integrated with IntelliJ



To save changes, commit to your local repo, and also push to your remote GH repo

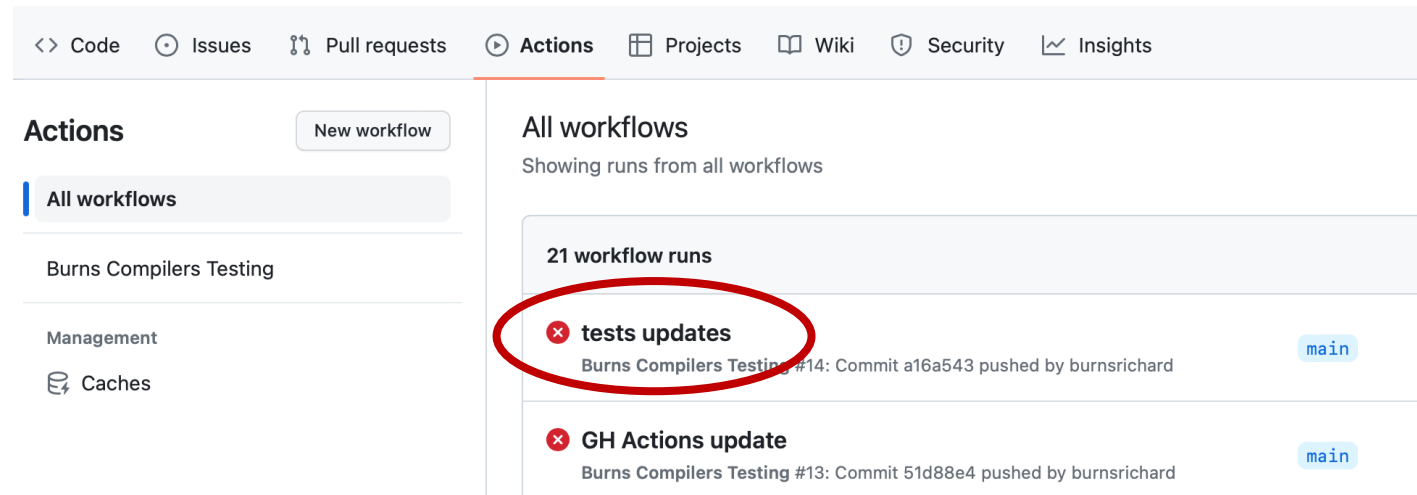


Autograded Tests and GitHub **Actions**



 **programming-assignment-0-burnsrichard-student** (Private)

- Initially, GH tests your starter code when it is created and some unit tests fail
- Expand the workflow by clicking on “GitHub Classroom Autograding Workflow”



- Then click on “Autograding”

✖ GitHub Classroom Autograding Workflow GitHub Classroom Workflow #1

🏠 Summary

Jobs

✖ Autograding

Triggered via push 12 minutes ago

Status

🖥️ github-classroom pushed -o- d63b195 **master**

Failure

classroom.yml

on: push

✖ Autograding

14s

Click this!

Autograding Output

There is a lot of info here, but if you scroll down:

- Some tests may **pass**
- Some tests may **fail**
- **Overall score**

```
142  
143 ✓ test_helloArgOutput  
144  
145 📝 test_addTwoNumbers()  
146
```

```
183 Error: R] Failures:  
184 Error: R]   AppTest.addTwoNumbers:36 expected:<5> but was:<6>  
185 [INFO]  
186 Error: R] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0
```

```
205  
206 ✗ test_addTwoNumbers()  
207 ::error::Error: Exit with code: 1 and signal: null  
208  
209 ::***::  
210 Points 3/4
```

Ideally, code/test your code locally first. But then verify that after pushing that your GH tests are also passing.

Consult the testing directory in Maven which contains the unit tests.

Be Patient!

Autograding sometimes takes a few minutes

- Under “Actions”, your latest commit should may say “Queued” or “In progress”
 - This means your code will be auto unit tested very soon

2 workflow runs		Event ▼	Status ▼	Branch ▼	Actor ▼
🟡	fixed the two methods, I think GitHub Classroom Workflow #2: Commit b3762af pushed by burnsrichard-student			master	📅 17 seconds ago 🕒 In progress ...

2 workflow runs		Event ▼	Status ▼	Branch ▼	Actor ▼
✅	fixed the two methods, I think GitHub Classroom Workflow #2: Commit b3762af pushed by burnsrichard-student			master	📅 2 minutes ago 🕒 45s ...

Scores are automatically populated to me in my admin view on GitHub Classroom

Homework 0

👤 Individual assignment 📅 Due 02/04/22 ● Active

<https://classroom.git>



✎ Edit

⬇ Download

Rostered students	70
Added students	0
Accepted students	1
Assignment submissions	0

🔍 Search by GitHub username or student identifier

Classroom roster

Unlinked accounts

Accepted

Submitted

Passing

Sort



Burns,Richard-Student

@burnsrichard-stude...

✖ Latest commit failed

2/6

1 commit

