

CS5004, Object-Oriented Design and Analysis

Lab1: Setting Up and Getting Started

Therapon Skoteiniotis¹, Tamara Bonaci, Chris Geeng, and Abigail Evans
skotthe@ccs.neu.edu, t.bonaci@northeastern.edu, a.evans@northeastern.edu

Table of Contents

- [1. Summary](#)
- [2. Setup](#)
 - [2.1. Java Tools](#)
 - [2.2. IntelliJ](#)
 - [2.3. Test your Setup](#)
- [3. Creating new classes](#)

1. Summary

In today's lab, we will:

- Configure our machines to use Java and IntelliJ/VSCode
- Show how to use Github Classroom to get assignment code.
- Walk through a simple example to create a class in Java
- Practice designing simple classes
- Practice designing classes that contain other classes

Note 1: Labs are intended to help you get started, and to give you some practice while the course staff is present, and able to provide assistance. You are not required to finish all the problems in this lab assignment, but you are expected to push your lab work to your individual repo/group repo on the GitHub course organization.

Please check course website for Lab 1 deadline.

¹ Assignment modified from the original version prepared by Dr. Therapon Skoteiniotis.

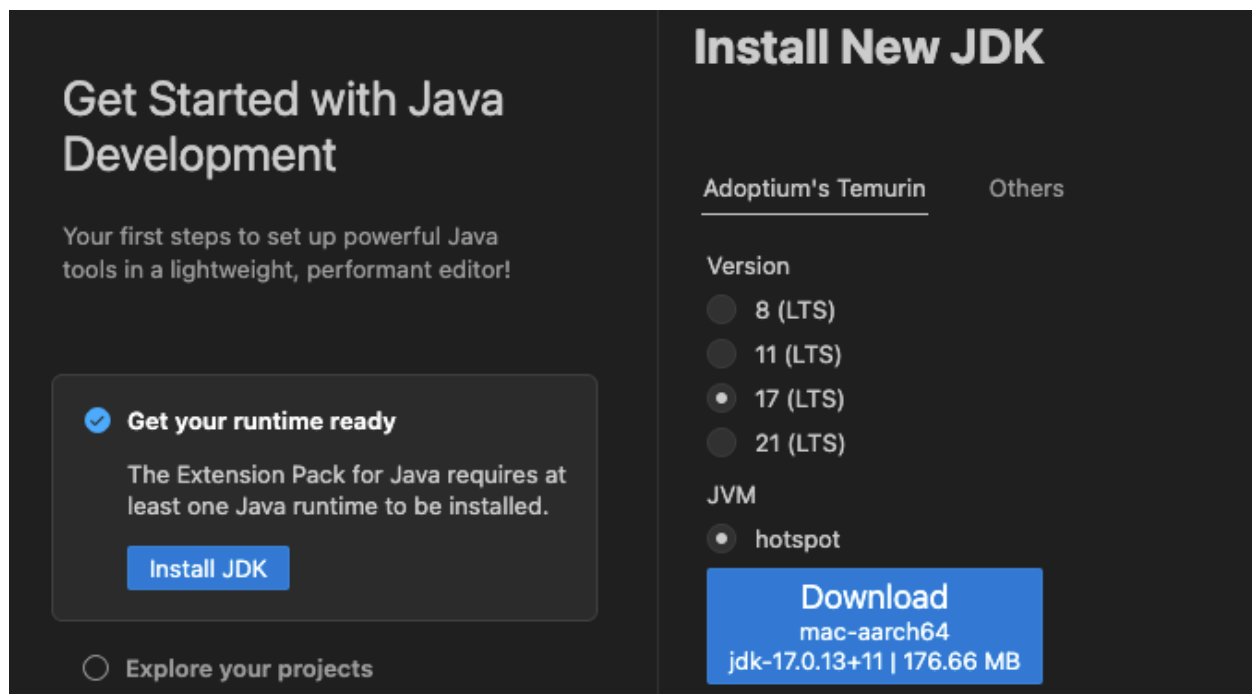
2. Setup

We will be using **Java 17, Gradle, and the IntelliJ IDE/VSCode**. **Note:** Course staff are more familiar with IntelliJ.

- Download [IntelliJ IDEA](https://www.jetbrains.com/idea/download/#section=mac) (<https://www.jetbrains.com/idea/download/#section=mac>). As a student, you can get a free license of IntelliJ IDEA Ultimate. **Ultimate is preferred over Community.**
- Or Download [VSCode](#)
- Download **Java 17** through your IDE

IntelliJ comes ready to use Gradle so you don't need to download anything else.

Note 2: If you select an IDE other than the one used in class, we will do our best to help you, but please be prepared to do extra work on your own. We will not give you extra time or amend your grade due to any possible issue that you might face with your IDE's configuration. VSCode will need the [Extension Pack for Java](#) (turn off IntelliCode).



2. Introduction to Git and GitHub

Prerequisite: for this part of the lab, you will need your **Github account**. If you don't already have one, sign up here <https://github.com/>.

6.1. Introduction

Git is a widely used, open source software for file management and version control. GitHub is an online code hosting platform that includes Git version control. For our course, we will be using the class organization in Github Classrooms to submit homework and lab assignments.

The URL to the course organization on Github Classrooms is:

<https://classroom.github.com/classrooms/179882046-5004-sea-sp25-geeng-classroom> (this link may not work until you clone your first assignment repository).

On the course organization, you should have access to repositories for Labs and Assignments. These are visible only to students in the organization.

- Assignment repos accessible to all students
- Assignment repo accessible to you only. You create a repo for each assignment/lab by clicking on the join link and signing up as a student from the roster. For example, Lab 1 is <https://classroom.github.com/a/5e8WDZ8r>.

You will need to access your individual repo, and clone it in order to connect a working directory on your own machine (local copy) with your repo on remote server.

6.2. Using Github Classrooms

When it comes to using git on your machines, you have many different options, such as:

- the Git command line tools,
- the GitHub GUI for Mac, or
- the GitHub GUI for Windows.

We will show how to use git on command line, and how to use GitHub GUI, but this semester, **we recommend using the command line**. Here are the instructions on how to get started with command line:

Getting Personal Access Token to Clone Github Repositories

Github has deprecated the usage of passwords. You must now use tokens to enable repository permissions. Follow instructions here:

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens#creating-a-personal-access-token-classic>.

1. Generate a personal access token (classic)

2. Set Note to be 5004 SP25.
3. Set expiration to **no expiration**.
4. Select **scopes** to be **repo**.
5. **Generate token**, and then **SAVE THE TOKEN SOMEWHERE ON YOUR COMPUTER**. You will not be able to access that token through Github again.

Connecting to the Classroom Github server Using Command Line

- **Step 1: Clone your assignment repository**
 - Go to Lab 01: <https://classroom.github.com/a/5e8WDZ8r>. Accept the assignment and click the repo link.
 - Open the **terminal** or equivalent. Navigate to the local folder where you want to store your files by typing:

```
cd <path to folder>
```

- <path to folder> is the full path of the folder on your machine.
- cd stands for “change directory”.

- Then, type the following:

```
git clone <remote repo URL>
```

- <remote repo URL> is the URL you copied earlier.
- Hit Return/Enter. You will be prompted for your username and password. Enter your **Personal Access Token** as your password.
- Once the repo has been cloned, cd into the newly created repo folder:

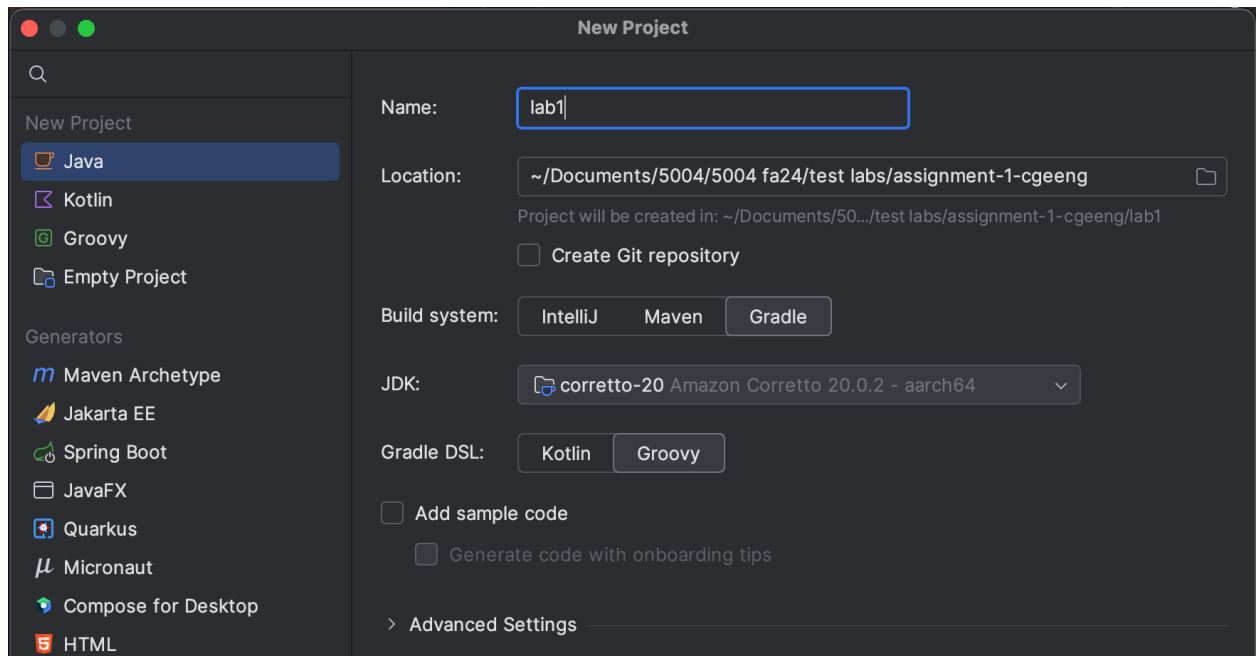
```
cd <folder name>
```

- **Step 2: Fetch changes from the remote repository**
 - This step only applies once your repository is active. Open Terminal or equivalent, and navigate to your local repository. Type:

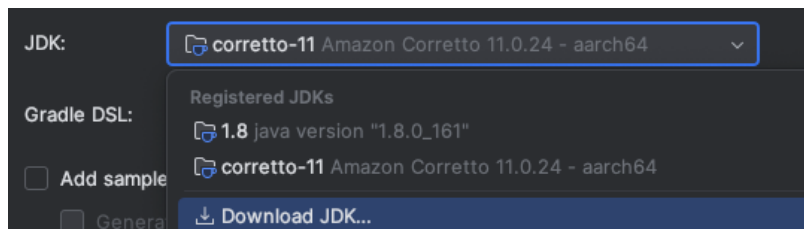
```
git pull
```

3. Creating Your First Gradle Project

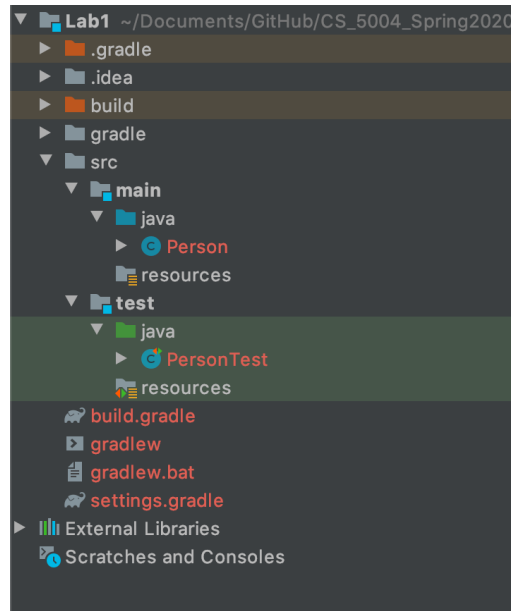
1. Start IntelliJ IDEA, and create a new project. Name it something relevant, like Lab1. Set its location to your new created repository.



2. Under Build system, select **Gradle**. (If this is not an option, download and enable Gradle under **Plugins**.)
3. Make sure the Project SDK (JDK) is 17 If you can't see 17 as an option under Project SDK, you will need to download the SDK before continuing.



At this point, you should have a Gradle project created. In IntelliJ, open the project tab to see the file structure. It will look something like this:



3.1 Adding and committing changes.

- **Step 4: Create/edit your files**
 - Please make sure you create all files for your assignment/lab inside the assignment folder you created.
- **Step 5: Add newly created files to Git tracking**
 - When you create a new file, you need to tell Git to keep track of it. Type the following:

```
git add <file-path>
```

- <file-path> is the path to the new file relative to the folder/directory you're currently in. So, if you're in your top level repository folder, which is called repo, and you want to commit a new file called hello.java that's in a sub-folder called HW1, you would enter

```
git add HW1/hello.java
```

- You can also easily add all untracked files at once by typing

```
git add .(including the period)
```

- **Step 6: Commit your files**

- A commit saves a local snapshot of your file for version tracking. It doesn't overwrite any previous versions, so if something goes wrong, you can always revert back to a previous version. Commit often!
- In Terminal or equivalent, navigate to your local repository folder. Type the following:

```
git commit -m "Your commit summary-should be short and descriptive"
```

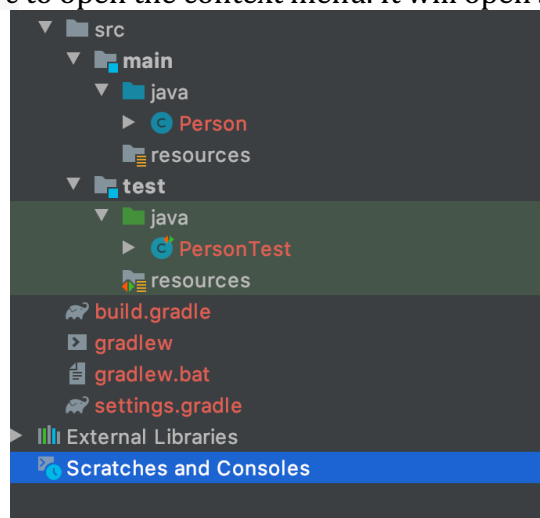
- **Step 7: Push your commit to the remote repository**
 - You do not need to push every time you commit, but it is good practice to push often. Type:

```
git push
```

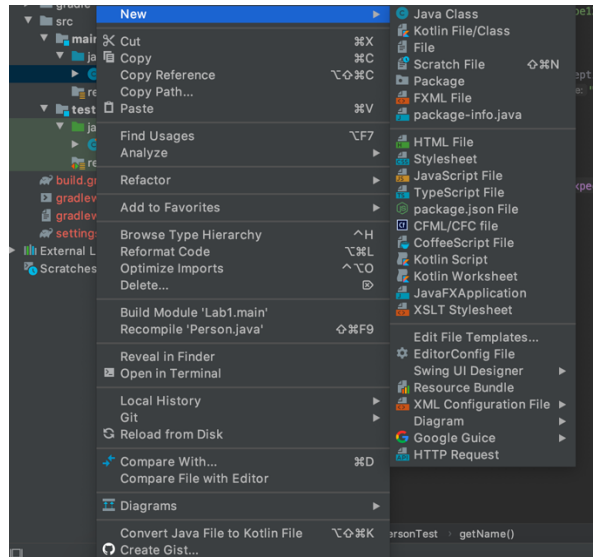
- You can check that your commit made it to the server by opening the repository in your browser.
- Login to Classroom Github in your browser, and open up your repository. You should see your changes. To make sure, click on the tab that says "X commits" and look for the "submitted".

3. Creating Your First Java Class

Click on the folder named src to open the context menu. It will open something like this:



Right-click on package main/java, to open context menu, and select `New → Class` to create a new Java Class.



A small window will pop-up asking you to provide a name for your class. Input the name `Author` and click `OK`

The right tab of your IntelliJ window should now contain a minimal Java class with

- a Java comment as the first line (if you don't have this, you can add it manually later. You can change Java class template to add it automatically, instructions [here](#).)
- an empty Java class definition

Notice that in the Project Explorer tab there is now a new file named `Author` under the folder named `main/java`.

To test that your setup is working as expected, replace all the contents of the file `Author.java` with the following code


Author.java


```

public class Author {

    private String name;
    private String email;
    private String address;

    public Author(String name, String email, String address) {
        this.name = name;
        this.email = email;
        this.address = address;
    }

    public String getName() {
         return name;
    }





    public String getEmail() {
        return email;
    }

    public String getAddress() {
        return address;
    }
}

```

4. Creating Your First Java Test Class

IntelliJ tries to assist you while you code by popping up an image of a small light bulb inside your editor (the right tab). Move your cursor to be inside the name of class, i.e., the word `Author` in the class header `public class Author`. Wait for a couple of seconds and the yellow light bulb image should appear at the start of that line. You can force the IntelliJ assistant to open using the following keystrokes

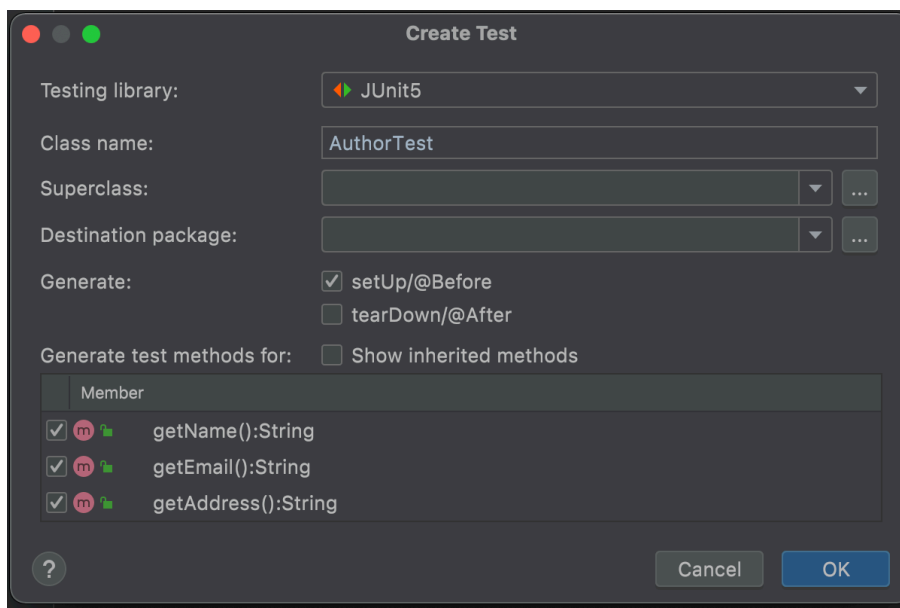
- Windows/Linux :  + 
- Mac :  + 

Click on the yellow light bulb icon and a menu should appear.

From the menu select `Create Test`. This action will cause a new pop-up window to appear.

In the new pop-up window, we will configure and create a test for our `Author` class. Starting from the top of the window going down:

- For "**Testing library**" select JUnit 5
- Leave "**Superclass**" empty.
- Leave "**Destination Package**" empty
- Select the check mark with the title "**setUp/@Before**" only.
- At the bottom of this pop-up window you should see the list of methods that are available in class `Author` for testing. Select **all** of them.
- Click `OK`
- Select JUnit Test Case from the pop-up menu. This will create a new Java class (and a new file) called `AuthorTest`.
- The Package Explorer should now have a new file with the name `AuthorTest` under the folder `src`



- The editor should now display something like the class presented below:

AuthorTest.java

```
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class AuthorTest {

    @BeforeEach
    void setUp() {
    }

    @Test
    void getName() {
    }

    @Test
    void getEmail() {
    }

    @Test
    void getAddress() {
    }
}
```

Observe that our test methods have no code in them. Let's implement them.

We will need to create a test object, and then we will want to use it, to test our three getters.

Your test class should now look something similar to this:

AuthorTest.java

```

import static org.junit.jupiter.api.Assertions.*;

class AuthorTest {

    private Author testAuthor;

    @BeforeEach
    void setUp() {
        testAuthor = new Author( name: "Jane Doe", email: "jane@paerson.com",
                                address: "5th Avenue N, San Francisco, CA");
    }

    @Test
    void getName() {
        assertEquals( expected: "Jane Doe", testAuthor.getName());
    }

    @Test
    void getEmail() {
        assertEquals( expected: "jane@paerson.com", testAuthor.getEmail());
    }

    @Test
    void getAddress() {
        assertEquals( expected: "5th Avenue N, San Francisco, CA", testAuthor.getAddress());
    }
}

```

To run our tests:

- press the green "play" button at the top of the IntelliJ window
- or press the green "play" button on the **left** margin on the line that contains the Java class definition header, e.g., `public class AuthorTest`.

IntelliJ will run your tests and the results of the test run are displayed in a new tab that opens at the bottom of the IntelliJ window.

5. Creating new classes

We decided to update our `Author` class. Instead of holding an author's name as a string, we would like to create a new class called `Person` that will hold

- a person's first name, and,
- a person's last name.

Practice Exercises

- 1) Create the class Person.
- 2) Update your Author class to use Person instead of String for an author's name.
 - a. Make sure to update any other parts of your code, e.g., getter methods etc.
- 3) Create an appropriate test class for your new class Person.