

CS5004, Spring 2021

Lab1: Setting Up and Getting Started

Therapon Skoteiniotis¹, Tamara Bonaci 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](#)
- [4. Creating more classes](#)
- [5. Javadoc](#)
- [6. Classes again!](#)
- [7. Resources](#)

1. Summary

In today's lab, we will:

- Configure our machines to use Java and IntelliJ
- Show how to create a Gradle project, and update build.gradle file
- 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 on the GitHub course organization.

The deadline to push your lab work appropriate is by 11:59pm on Tuesday, January 20, 2021.

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

2. Setup

2.1. Java Tools

Download and install the Java SE Development Kit 11

2.2. IntelliJ

The teaching staff will be using IntelliJ in class and labs. You do not have to install IntelliJ, but if you do not have an IDE already available, you may choose to download and use IntelliJ for this course. Here are the steps:

- Navigate to the IntelliJ website: <https://www.jetbrains.com/idea/>
- As a student, you are eligible for a free full version (Ultimate version) of JetBrains products, but you need to apply for a free student license: <https://www.jetbrains.com/student/>
- While waiting for your student license is approved, download IntelliJ IDEA Ultimate: <https://www.jetbrains.com/idea/download/download-thanks.html?platform=mac>
- **Important: please download and use IntelliJ IDEA Ultimate edition, not the Community edition.**
- Install IntelliJ IDEA Ultimate on your machine using the process that you typically use for your operating system to install new software, and follow the steps by the IDE to validate your account (typically, just sign in with your JetBrains username and password when prompted by IntelliJ.)

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.

2.3. Introduction to Git and GitHub

Prerequisite: for this part of the lab, you will need your **Khoury (CCIS) account**. If you don't already have a Khoury account, please create one using the following link: <https://my.ccs.neu.edu/account/apply>

2.3.1. 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 Khoury Enterprise GitHub (a private GitHub instance) to submit homework and lab assignments.

The URL to the course organization on Khoury GitHub is:

<https://github.ccs.neu.edu/cs5004-spr21-sea>

On the course organization, you should have access to two repositories (repos):

- Repo accessible to all students, named **lecture-code**
- Repo accessible to you only

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.

2.3.2. Using Khoury GitHub

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 this semester. Here are the instructions on how to get started with command line:

Connecting to the Khoury GitHub server Using Command Line

- **Step 1: Clone your personal repository**
 - You only need to complete this step once. If you've already cloned your repository, skip to step 3.
 - First, find the URL for your personal repository. In your browser, login to Khoury GitHub and open your personal repository. Click the *Clone or download* button and copy the URL.
 - Open up 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 may be prompted to login to Khoury GitHub. 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
```

- **Step 3: Create a folder for the assignment**

- If you are just getting started on the assignment, create a new folder in your local repository using Finder/File Explorer. Please name your folder something obvious, like “HW1”. Course staff will access your repository to review your submissions so make sure it’s easy for them to find your submissions.

- **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 Khoury 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 your summary message.

Connecting to the Khoury GitHub server GitHub GUI App

In this section, we outline steps to clone a GitHub repo using GitHub Desktop app:

<https://desktop.github.com>

It will be your decision to make whether you want to use command line or GUI app.

If you choose to use GUI app, steps are as follows:

Step 1: Download and install GitHub GUI app

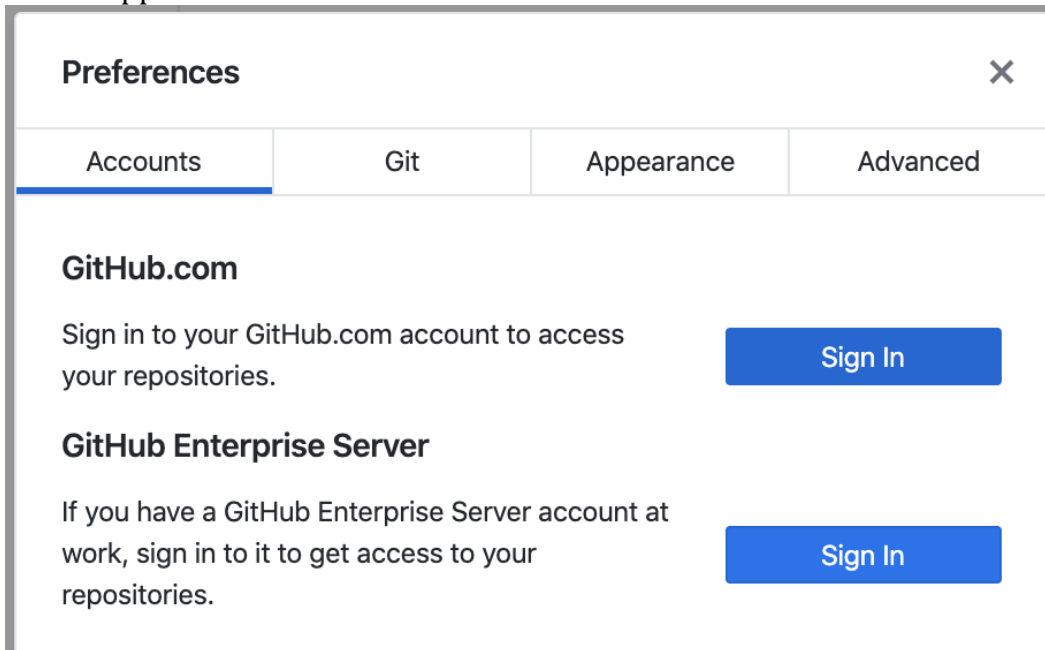
Download an appropriate version of GitHub desktop app: <https://desktop.github.com>, and install it on your machine using the process that you typically use for your operating system to install new software.

Step 2: Connect to the Khoury GitHub server

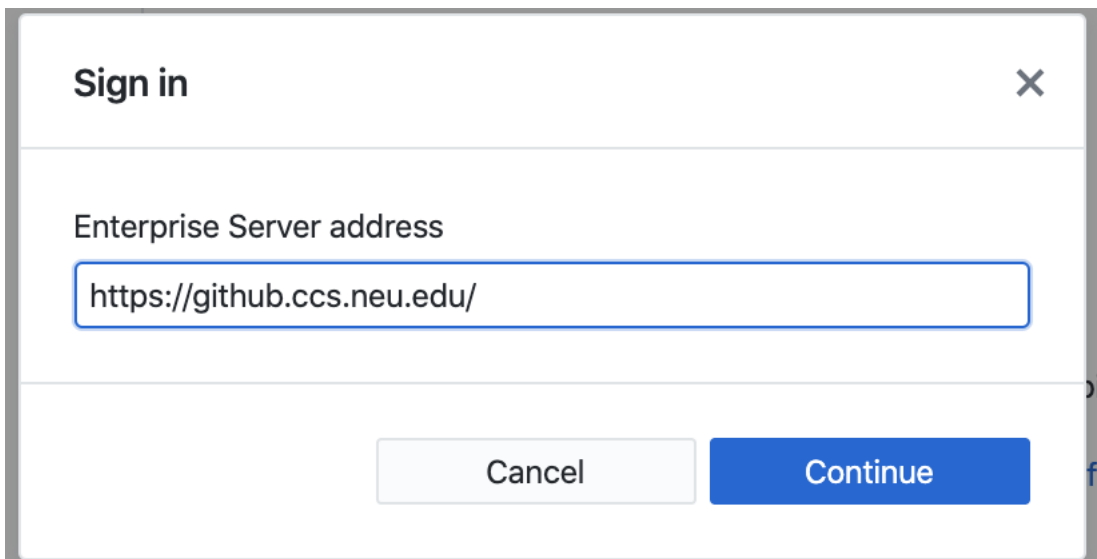
These instructions assume you are using the GitHub Desktop application. The following instructions were written using version the Mac GitHub Desktop application.

- Start the application.

- From the main application menu, find *Preferences* and select it. A new window should appear.



- In the new window, find the section titled *GitHub Enterprise* and click *Sign In*.
- For the *server address* field, enter <https://github.ccs.neu.edu> then click *Continue*.



- Enter your Khoury account username and password to login.
- Navigate to the CS 5004 course organization, directly accessible here:
- <https://github.ccs.neu.edu/cs5004-spr21-sea>
- Navigate to your personal repository:

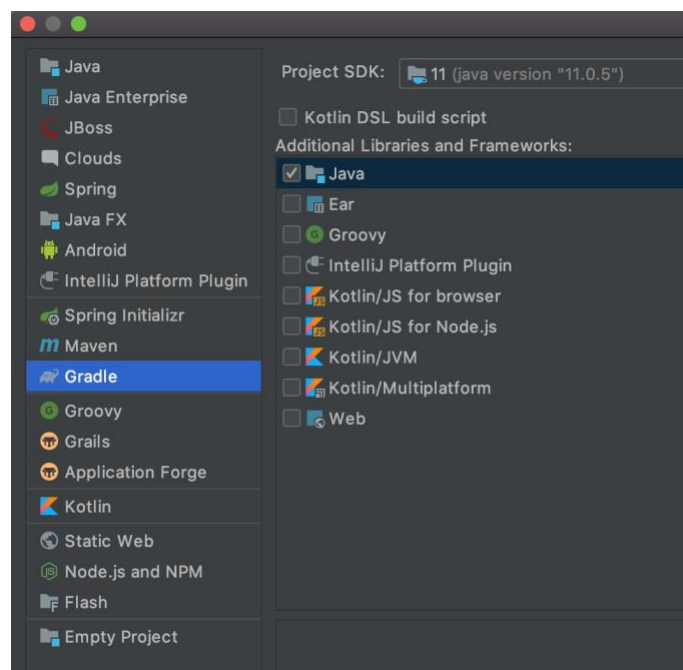
Step 3: Clone your personal repository

Cloning a repository copies a complete remote repository on the server to your personal machine. ***You only need to clone the first time you use the repository.***

- From the application dashboard, click *Clone a Repository*.
 - An overlay window will open, listing all the repositories available to you. If you don't see any repositories, check that the *Enterprise* tab is selected.
- Click on your personal repository to select it.
- Check the *Local Path* field—this is where the cloned repository will live on your machine, so make sure it's where you want it to be. Use the *Browse* button to change the location if needed. When you're happy with the location, click *Clone*.
- In Finder (Mac) or File Explorer (PC), navigate to the repository on your local machine. You should see a new folder/directory that matches the remote repository you cloned.

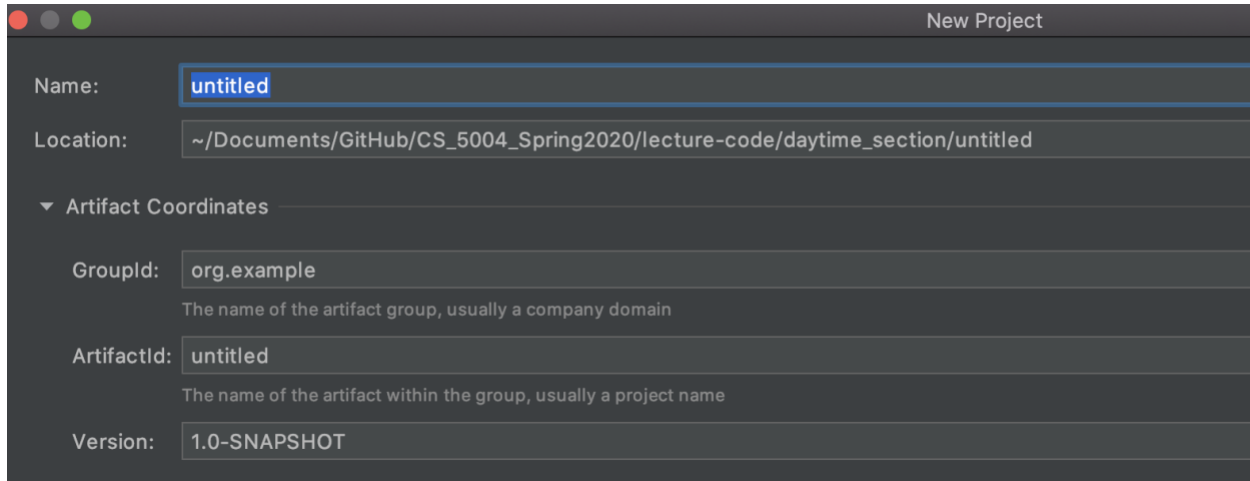
2.4. Creating Your First Gradle Project

1. Start IntelliJ IDEA, and create a new project.
2. In the left pane of the selection window, select Gradle.



3. Make sure "Java" is selected in the "Additional Libraries and Frameworks" list in the right pane (see above).
4. Make sure the Project SDK is 11 (see screenshot above). If you can't see 11 as an option under Project SDK, you will need to add the SDK before continuing.
5. Click "Next". You will see a dialog box asking you to enter "GroupId" and "ArtifactId".
 - Ignore the GroupId box.

- ArtifactId: enter the project name. For assignments, this will be "AssignmentX", where X is the assignment number.
- Adjust the project location if needed but leave everything else as-is. Click "Finish".



New Project

Name:

Location:

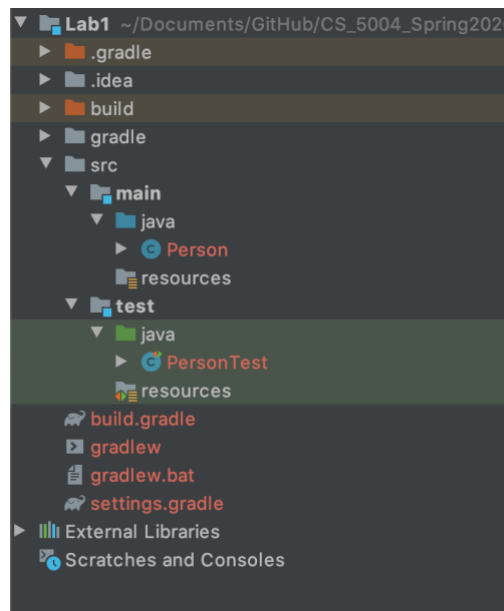
▼ Artifact Coordinates

GroupId:
The name of the artifact group, usually a company domain

ArtifactId:
The name of the artifact within the group, usually a project name

Version:

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:

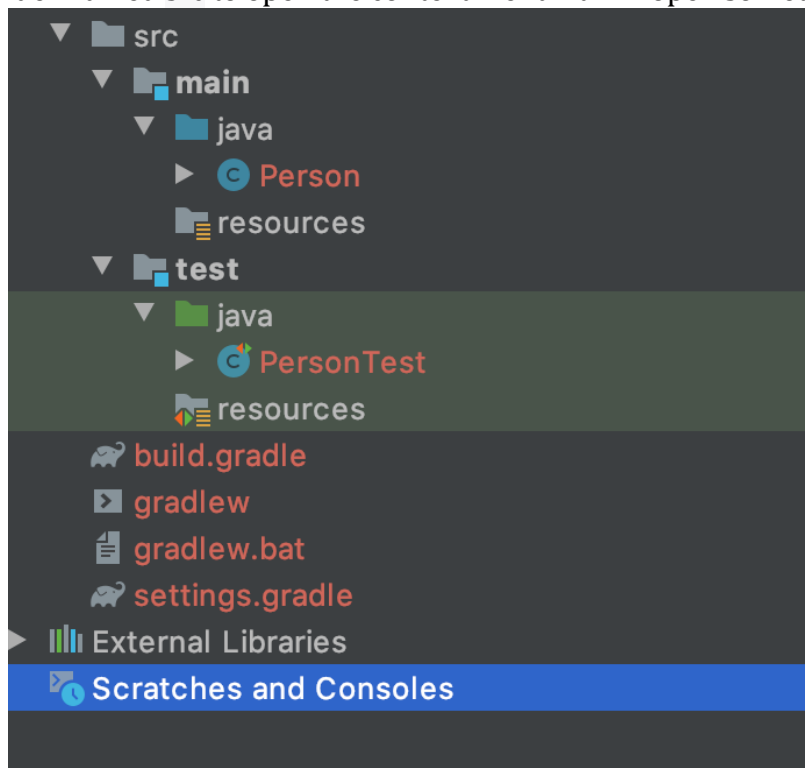


The last step is to configure your **build.gradle** file so that it does everything we want it to in this course. The easiest way to do this is to download the course **build.gradle**, and replace the **build.gradle** in your project folder (top level of your project folder) with the downloaded file.

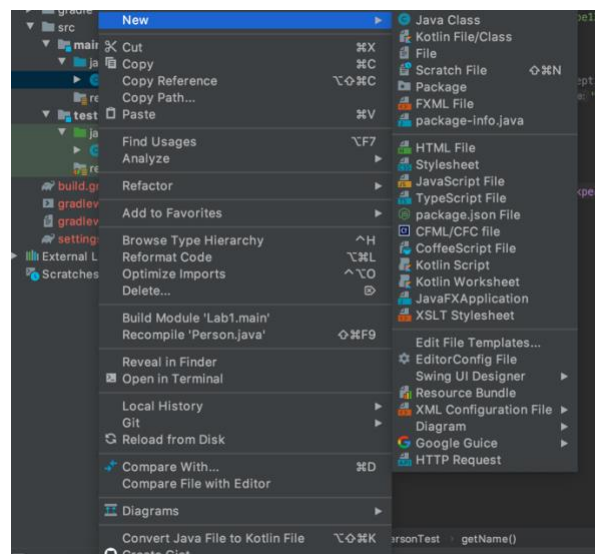
A message will pop up in IntelliJ, warning you that Gradle projects need to be imported. Click "import changes" (or "Enable Auto-Import" if you don't want to bother with this step in future).

2.5. 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.



8. A small window will pop-up asking you to provide a name for your class. Input the name `Author` and click `OK`
9. IntelliJ will detect that you are inside a git repository and will ask you if you would like to add the file to your repo. Please click `No`
10. The right tab of your IntelliJ window should now contain a minimal Java class with
 - a Java comment as the first line
 - an empty Java class definition
11. Notice that in the Project Explorer tab there is now a new file named `Author` under the folder named `main/java`.
12. To test that your setup is working as expected, replace all the contents of the file `Author.java` with the following code

Author.java

```
/**
 * Represents an Author with their details--name, email and physical address
 *
 * @author theapon
 *
 */

public class Author {

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

    /**
     * Creates a new author given the author's name, email and address as strings.
     *
     * @param name the author's name
     * @param email the author's email address
     * @param address the authors physical address
     */
    public Author(String name, String email, String address) {
        this.name = name;
        this.email = email;
        this.address = address;
    }

    /**
     * @return the name
     */
    public String getName() {
        return this.name;
    }

    /**
```

```

    * @return the email
    */
    public String getEmail() {
        return this.email;
    }

    /**
     * @return the address
     */
    public String getAddress() {
        return this.address;
    }
}

```

19. 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 : `Alt` + `Enter`
 - Mac : `Option` + `Enter`
20. Click on the yellow light bulb icon and a menu should appear.
21. From the menu select `Create Test`. This action will cause a new pop-up window to appear.
22. 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
23. For "**Testing library**" select `JUnit 4`
24. Click the `Fix` button to add `JUnit4` to your project. A new pop-up window will appear.
25. Select the first option with the title "**Use JUnit4 from IntelliJ IDEA distribution**"
26. Click `OK`. This will close this pop-up window and takes us back to continue setting up our test for `Author`.
27. Back in the pop-up for setting up our test for `Author` leave "**Superclass**" empty.
28. Leave "**Destination Package**" empty
29. Select the check mark with the title "**setUp/@Before**" only.
30. 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.
31. Click `OK`
32. Select `JUnit Test Case` from the pop-up menu. This will create a new Java class (and a new file) called `AuthorTest`. If IntelliJ asks you to add this new file to your repo click `No`.
33. The Package Explorer should now have a new file with the name `AuthorTest` under the folder `src`
34. The editor should now display

AuthorTest.java

```
public class AuthorTest {
    @org.junit.Before
    public void setUp() throws Exception {

    }

    @org.junit.Test
    public void getName() throws Exception {

    }

    @org.junit.Test
    public void getEmail() throws Exception {

    }

    @org.junit.Test
    public void getAddress() throws Exception {

    }
}
```

35. Notice the red lines in the right margin of your editor. Use your mouse to hover over the red lines and see the issue. The issue here is that we have not added the JUnit4 library's code so that IntelliJ can find it. Move your cursor to the second line in the file *AuthorTest* that contains the string `@org.junit.Before` and force the IntelliJ assistant (see step 19).
36. From the assistant's pop-up menu select **"Add junit.jar to classpath"**. The red marks in the right margin should now disappear.
37. Observe that our test methods have no code in them.
38. Add the following line to **each** of the method bodies in *AuthorTest* except for the method `setUp`
 - `TestCase.fail("Not yet implemented");`The addition of `TestCase.fail` will cause an issue, use the assistant to resolve the issue. Your file should now look like this

AuthorTest.java

```
public class AuthorTest {
    @org.junit.Before
    public void setUp() throws Exception {

    }

    @org.junit.Test
    public void getName() throws Exception {
        TestCase.fail("Not yet implemented");
    }
}
```

```

@org.junit.Test
public void getEmail() throws Exception {
    TestCase.fail("Not yet implemented");
}

@org.junit.Test
public void getAddress() throws Exception {
    TestCase.fail("Not yet implemented");
}
}

```

39. Our tests now will fail because we explicitly added `TestCase.fail("Not yet implemented");` to each test method. To run our tests and see the failures

- 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 {`

40. 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.

3. 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`.