miniproject_0.zip

Project Goals & Outcomes

This project is intended to test your environment and give you a chance to work out any kinks in your development setup prior to starting work on a graded mini-project. The provided code includes a complete (but trivial) application with tests and source code. You will use this application to verify that you are able to compile it in your development environment and try out submitting it to the Coursera autograder. No code modifications are required or permitted for this project; this project is intended to be compiled as-is.

The following steps present the process your should follow for all projects in this course, which includes local development and testing followed by uploading your submission to the Coursera autograder.

Project Setup

Start by downloading and unzipping the project files attached to this description using the gray button labeled miniproject_0.zip at the top of this description. After that, you should see the main project source code file at:

`miniproject_0/src/main/java/edu/coursera/parallel/Setup.java`

and a single test file at:

`miniproject_0/src/test/java/edu/coursera/parallel/SetupTest.java`

The hands-on mini-projects in this course will be provided as fully functioning Maven projects. Maven is a build tool which automatically manages dependencies, source code compilation, test compilation, and test execution. You can use Maven manually from the command line or from the Eclipse/IntelliJ IDEs. While Maven is the most convenient way to build and run projects in this course, you are not required to use Maven if you prefer not to. To that end, we will also provide the necessary JARs with each mini-project if you would rather compile manually or use a different build tool.

The following sections will briefly describe how to both compile and test the mini-projects in this course in four different ways: with Maven on the command line, with Maven through IntelliJ, with Maven through Eclipse, and with javac on the command line. Note that you only need to use one approach (and it need not even be one of these). Choose whatever approach to compiling Java programs and running JUnit tests you are most comfortable with.

Compiling with Maven from the Command Line

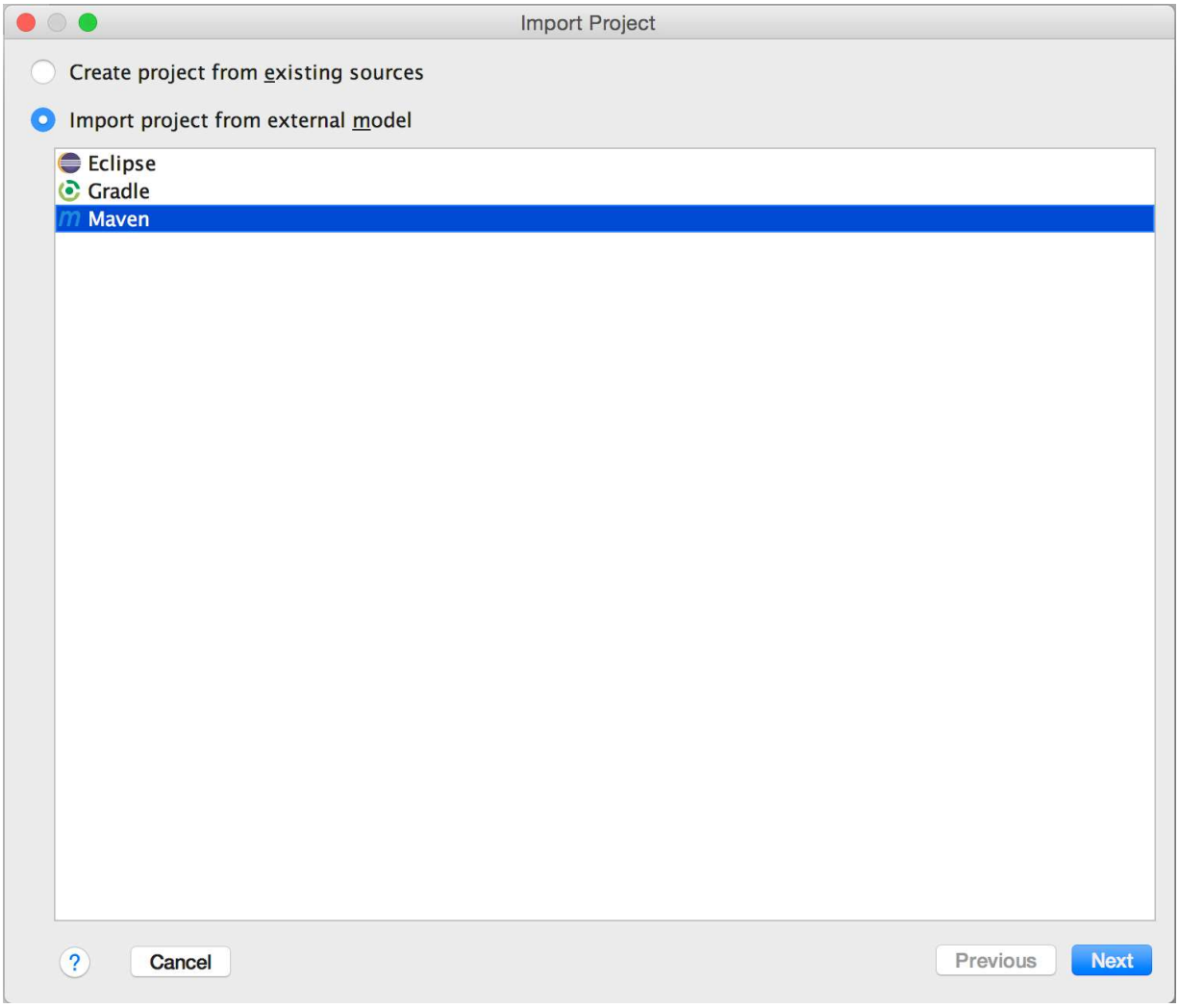
If you wish to compile the source code using Maven from the command line, simply issue the following command from the command line from inside the root miniproject_0/ directory:

```
1  $ mvn compile
```

Compiling with Maven from IntelliJ

Assuming you already have IntelliJ downloaded and installed on your laptop, adding the Maven plugin can be done from Preferences > Plugins. Simply type "Maven" in the search bar at the top of the Plugins window, select the "Maven Integration" plugin, and hit OK to install it.

Next, the provided Maven project must be imported to IntelliJ. To do so, go to File > New > Project from Existing Sources... and select the root miniproject_0/ directory extracted from the provided miniproject_0.zip. You should receive an Import Project prompt like the one shown below. Select the "Import project from external model" radio button, select Maven from the list below, and click through the following prompts to complete the import.

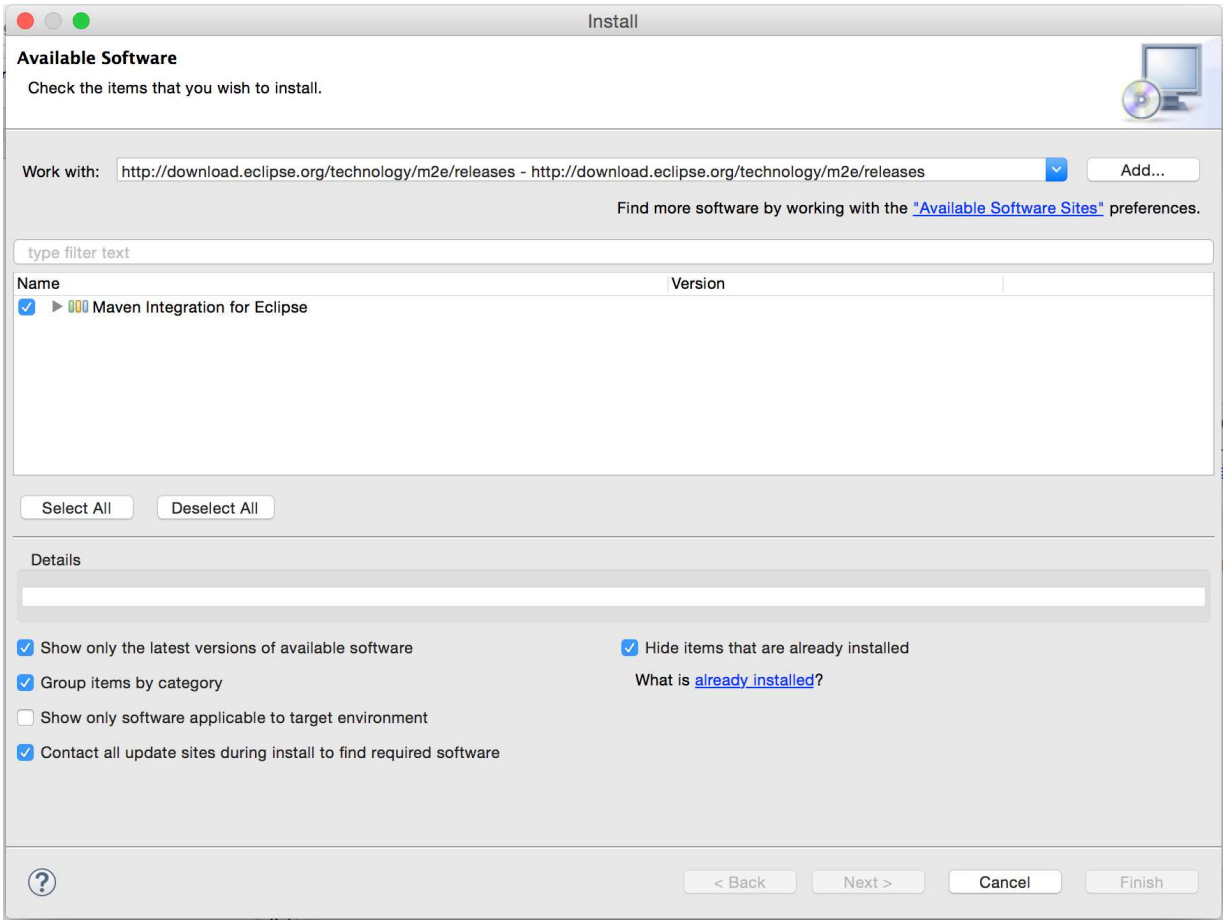


Compiling with Maven from Eclipse

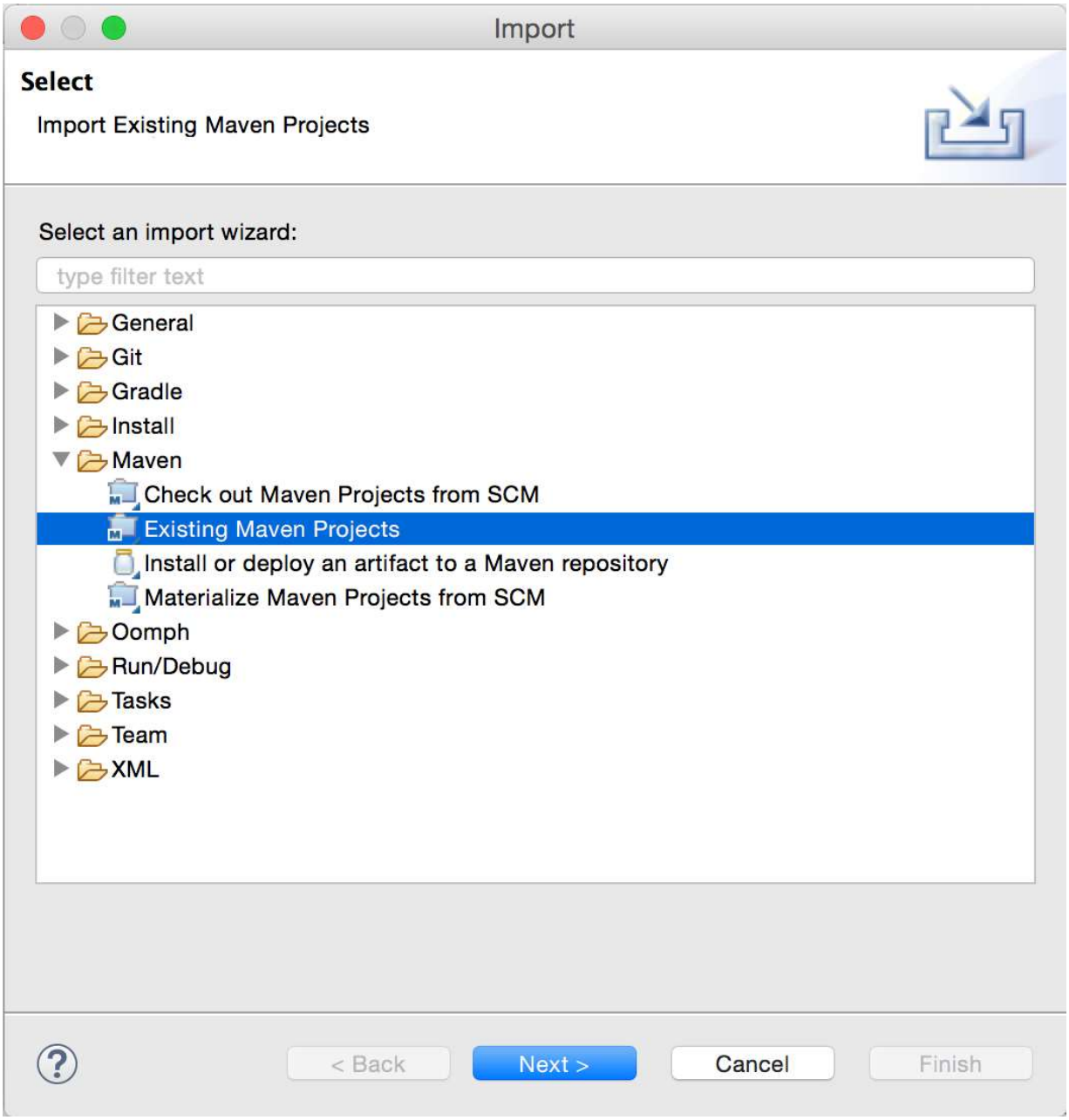
You can also use the Eclipse IDE to compile Maven projects. Assuming you already have Eclipse downloaded and installed, navigate to Help > Install New Software..., click the Add... button at the top right, and use the following URL to add a new Eclipse repository:

http://download.eclipse.org/technology/m2e/releases

With that repository selected you should see the "Maven Integration for Eclipse" appear in the central pane of the installation window (shown below). Select the checkbox to its left, click Next, and follow the prompts to complete installation of the Maven plugin for Eclipse.



Next, to import this project navigate to File > Import and in the Import wizard expand the Maven section to select Existing Maven Projects (shown below).



After clicking the Next button, select the root directory for this mini-project in the Root Directory text box at the top and hit Finish. You should now have a completely imported Maven project.

Compiling Manually Using Javac

If you wish to compile manually using the command line (without Maven), you will need to add the following JARs to your classpath while building both the provided source and test files using javac: hamcrest-core-1.3.jar, junit-4.12.jar, and pcdp-core-0.0.4-SNAPSHOT.jar. One possible way to invoke the javac command from the root folder is as follows:

```
1  $ javac -cp ./hamcrest-core-1.3.jar:./junit-4.12.jar:./pcdp-core-0.0.4-SNAPSHOT
     .jar:target/classes/:target/test-classes/ src/main/java/edu/coursera/parallel
     /Setup.java src/test/java/edu/coursera/parallel/SetupTest.java
```

Project Instructions

Similar to compiling mini-projects, testing of mini-projects in this course can generally be done either with Maven (through the command line, Eclipse, or IntelliJ) or manually.

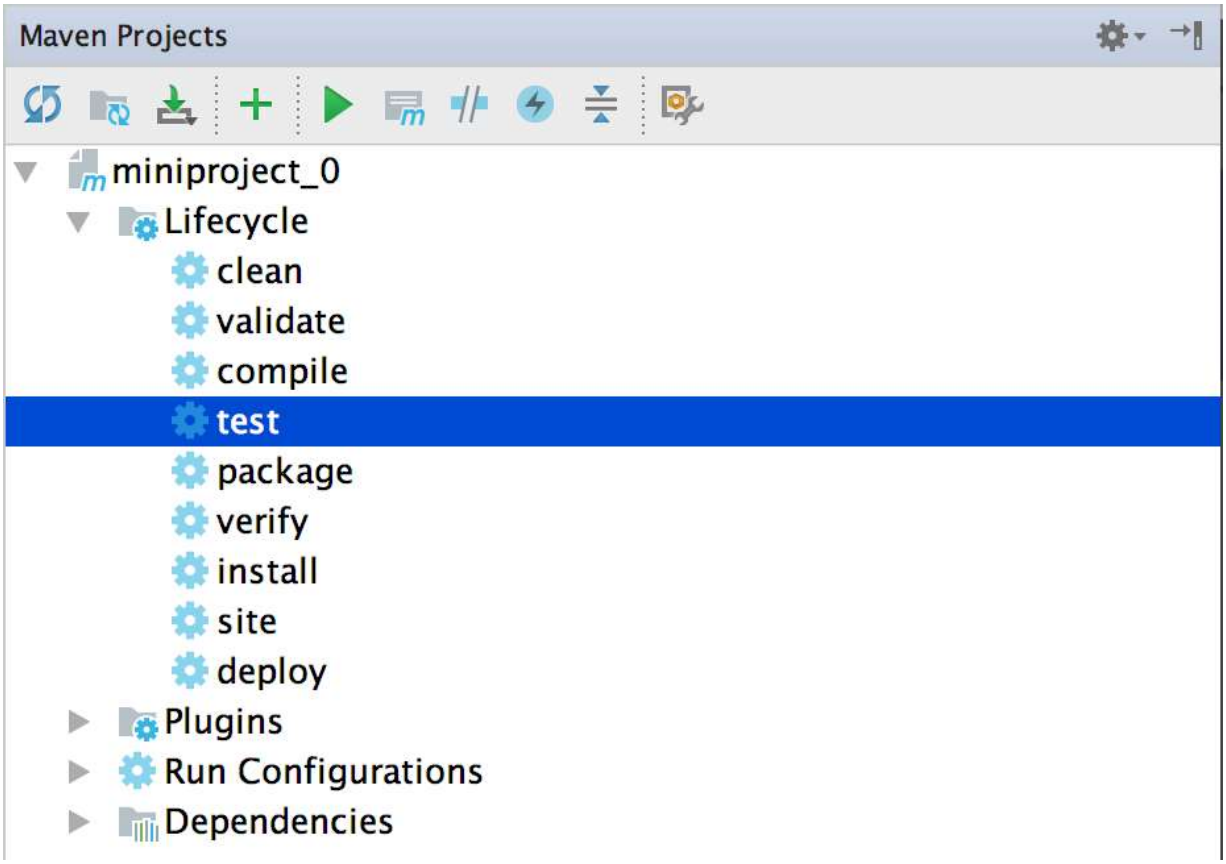
Testing with Maven from the Command Line

Maven projects can be tested from the command line by simply using the following command:

```
1  $ mvn test
```

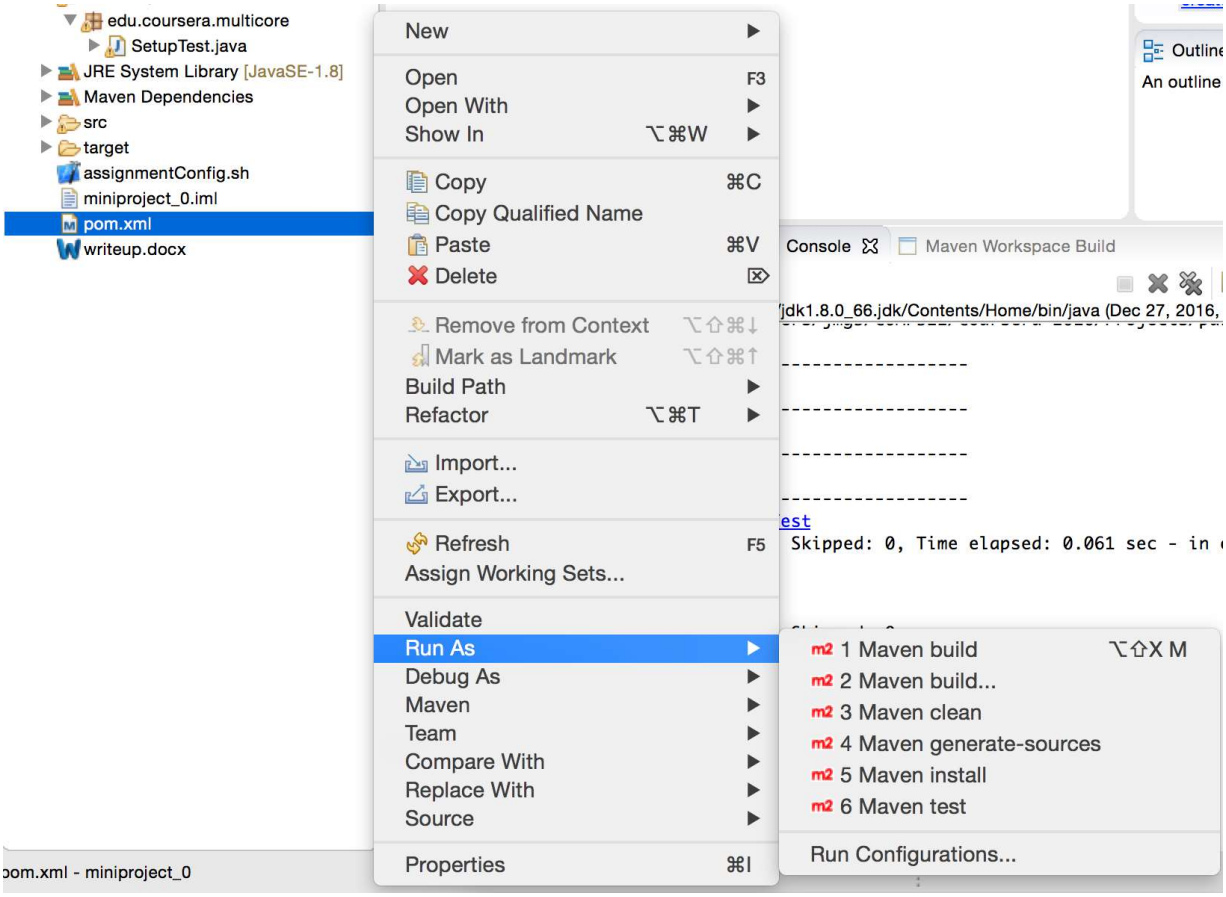
Testing with Maven from IntelliJ

Running the JUnit tests provided with this project is very simple to do from IntelliJ. Open the Maven Projects pane by navigating to View > Tool Windows > Maven Projects. From the Maven Projects pane, expand the Lifecycle section and double-click "test" to automatically run the tests (shown below). The output of the tests will be displayed in a separate text pane.



Testing with Maven from Eclipse

Running Maven tests through Eclipse simply requires right-clicking the pom.xml file in Eclipse's Package Explorer, selecting Run As, and selecting "Maven test" (shown below). A new text pane should open with the results of the tests. If you do not see the "Maven test" option, try refreshing your project by right-clicking on the project name in the Package Explorer and selecting Refresh.



Testing Manually from the Command Line

For testing manually, you will need the following JARs on your classpath as well as the folder containing your compiled class files: hamcrest-core-1.3.jar, junit-4.12.jar, and pcdp-core-0.0.4-SNAPSHOT.jar. You can then run the provided tests manually from the command line using the following command:

```
$ java -cp ./hamcrest-core-1.3.jar:./junit-4.12.jar:./pcdp-core-0.0.4-SNAPSHOT
  .jar:target/classes/:target/test-classes/ org.junit.runner.JUnitCore edu
  .coursera.parallel.SetupTest
```

If you are unfamiliar with the -cp option and the term, classpath, or not accustomed to running JVM programs on the command line, we strongly recommend that you use Maven (manually or through an IDE).

Project Evaluation

Once you are able to compile this project locally and run the provided test locally, you should test submitting it to the Coursera autograder through the assignment page for this mini-project. You can do so by only uploading the provided source code file:

`miniproject_0/src/main/java/edu/coursera/parallel/Setup.java`

After that, the Coursera autograder will take over and assess your submission, which includes building your code and running it on one more tests. Please give it a few minutes to complete the grading. Once it has completed, you should see a score of 100/100 appear in the “Score” column of the “My submission” tab, and see “Yes” appear in the “Passed?” column of the same tab. You can also click on

"Show grader output" to see the details of your test results. Note that for all assignments in this course you are free to resubmit as many times as you like. Congratulations on completing your first mini-project!

✓ Complete