

# Download and Set Up (Cooperative Runtime)

HJlib provides support for multiple runtimes. In this article, we will discuss how to set up the JDK11 compatible cooperative runtime. Instructions on the [download and set up of the thread-blocking runtime is available in another article](#).

Here are the steps for running HJlib on Java 11 using an IDE:

## **Step 1: Java 11 Installation**

You will also need a Java 11 installation on your machine and have your JAVA\_HOME and PATH point to the new installation. Java 11 can be downloaded and installed from the [Oracle website](#).

For e.g., I have the following on my Mac machine's .bash\_profile:

```
JAVA11_HOME=/Library/Java/JavaVirtualMachines/jdk11.0.13.jdk/Contents/Home
export JAVA_HOME=${JAVA11_HOME}
export PATH=$JAVA_HOME/bin:$PATH
```

On Windows the environment variables have to be set up differently, refer to this [stackoverflow question](#) to see how it can be done.

## **Step 2: HJlib JAR File**

[Download the JDK11 compatible HJlib jar](#) file and save it to a local directory.

## **Step 3: IDE like IntelliJ or Eclipse**

HJlib is a simple Java library (jar file) and can be used in any Java project. As such a simple text editor can be used to write programs that use HJlib.

However, we recommend using an IDE like IntelliJ to do the Java development using HJlib in the labs and assignments.

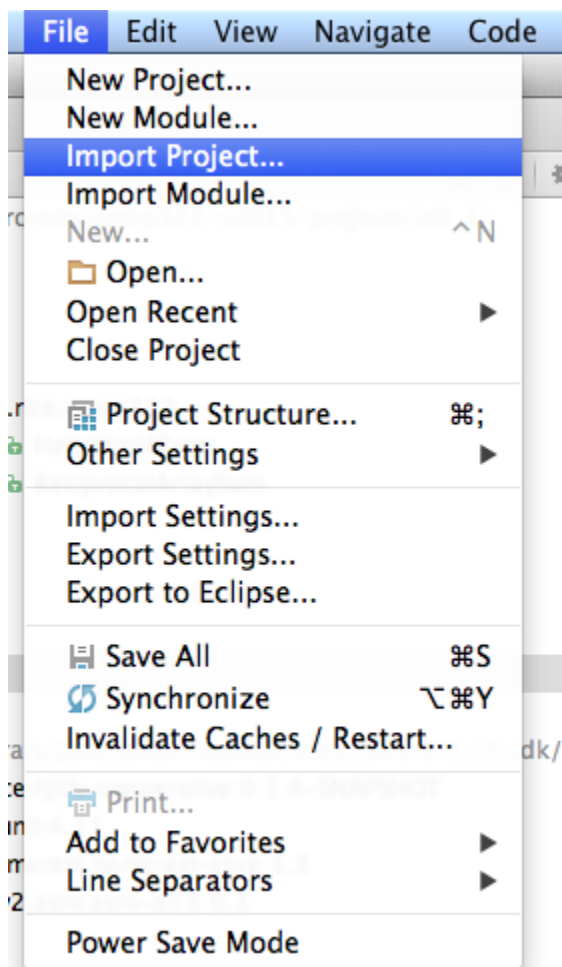
A free version of IntelliJ (Community Edition) can be downloaded and installed from the [Jetbrains website](#).

## **Step 4: Your first project**

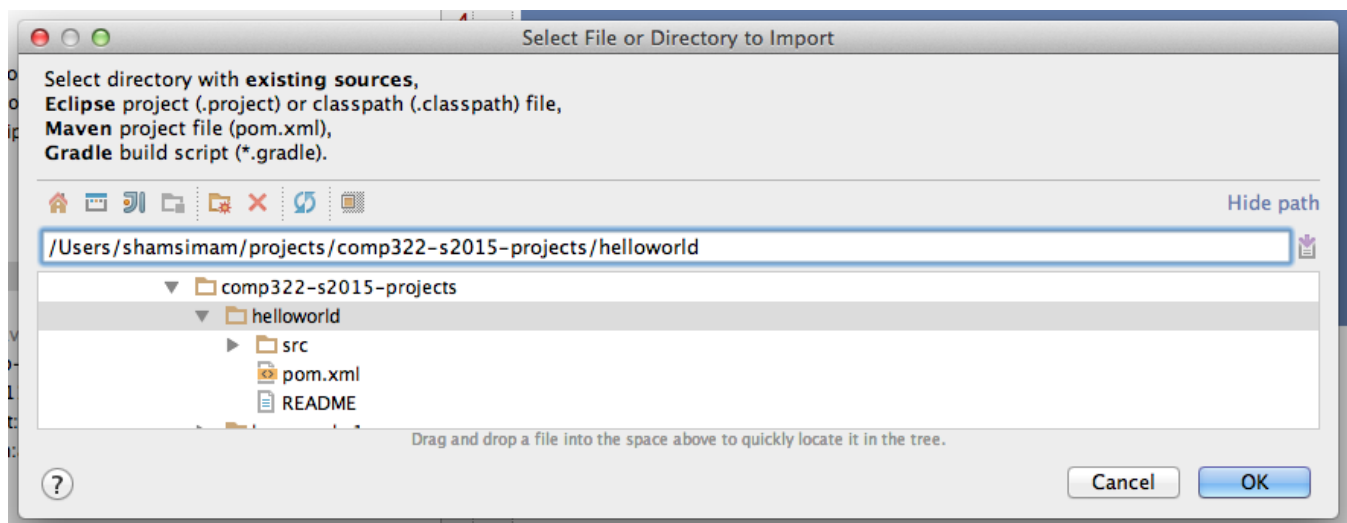
We show how to set up a simple Maven project with a HJlib HelloWorld program in IntelliJ. Similar steps can be followed for other IDEs like Eclipse.

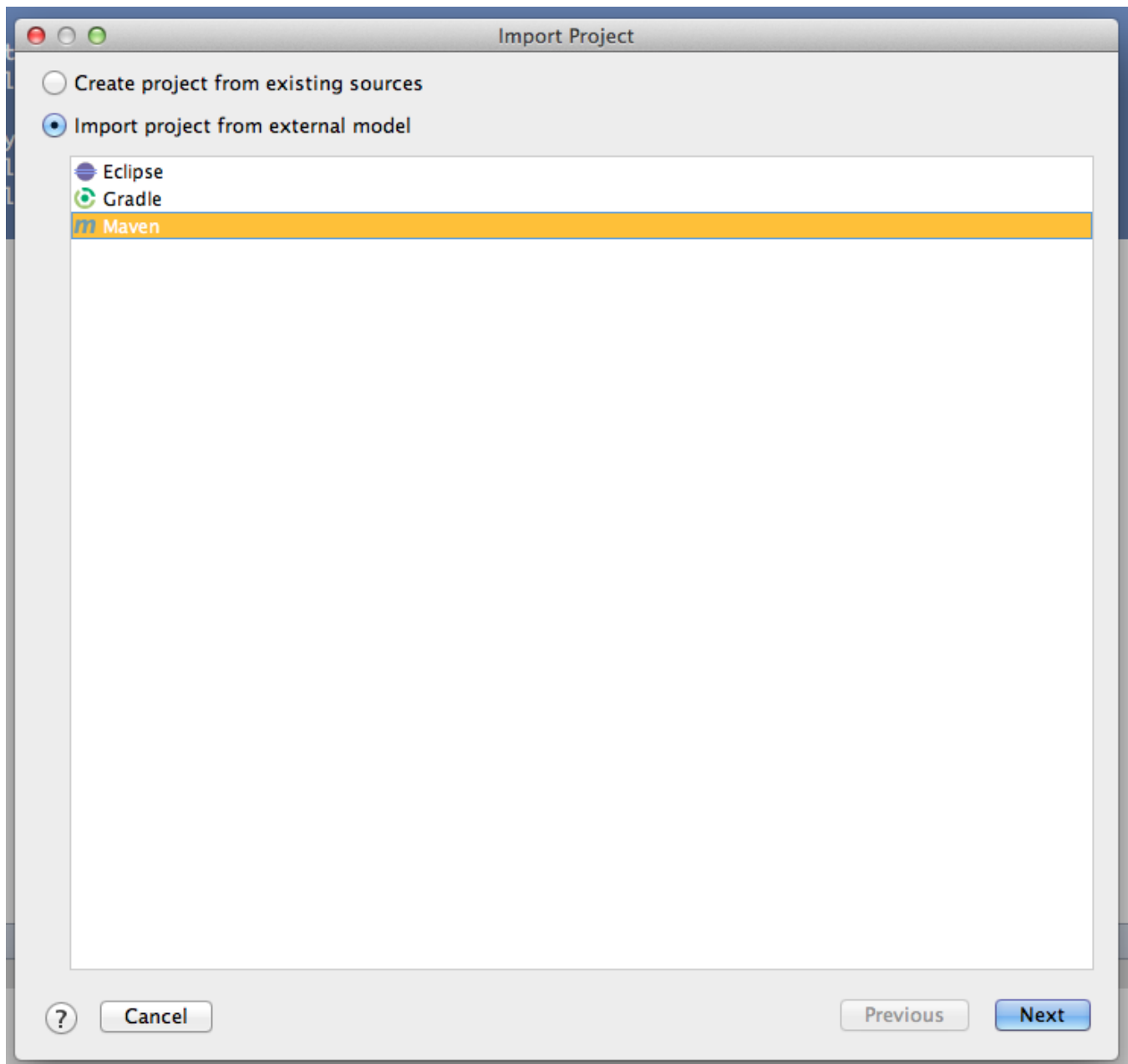
First, download the [helloworld.zip](#) file and unzip it into a directory of your choice (e.g. /Users/shamsimam/projects/comp322-s2015-projects/helloworld).

Next, import this project into IntelliJ using the File -> Import Project menu option.



This should show the following series of popups:





Import Project

Root directory:  ...

☐ Search for projects recursively

Project format:

☐ Keep project files in:  ...

☐ Import Maven projects automatically

☒ Create IntelliJ IDEA modules for aggregator projects (with 'pom' packaging)

☐ Create module groups for multi-module Maven projects

☒ Keep source and test folders on reimport

☒ Exclude build directory (%PROJECT\_ROOT%/target)

☒ Use Maven output directories

Generated sources folders:

Phase to be used for folders update:

IDEA needs to execute one of the listed phases in order to discover all source folders that are configured via Maven plugins.  
Note that all test-\* phases firstly generate and compile production sources.

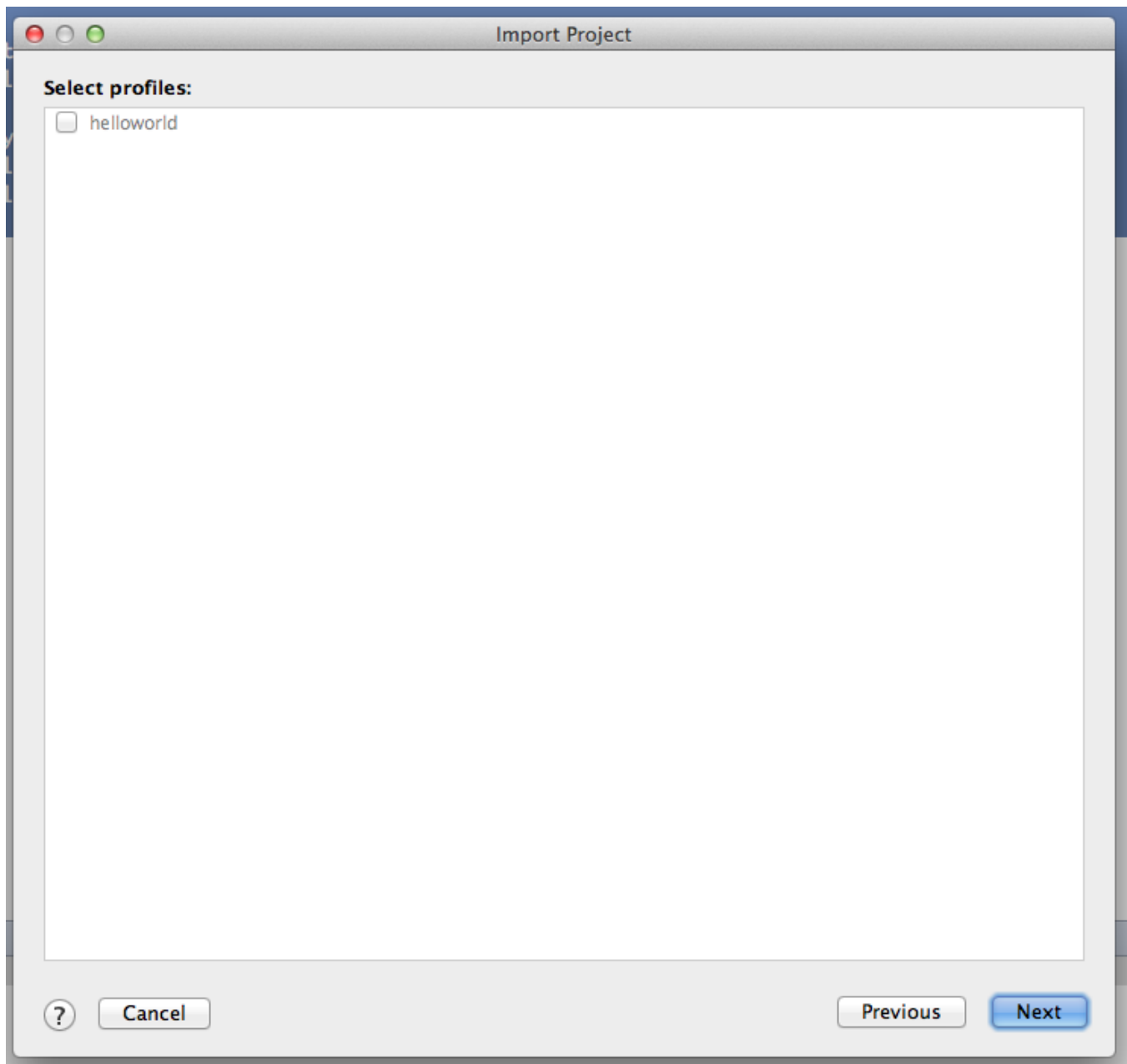
Automatically download: ☐ Sources ☐ Documentation

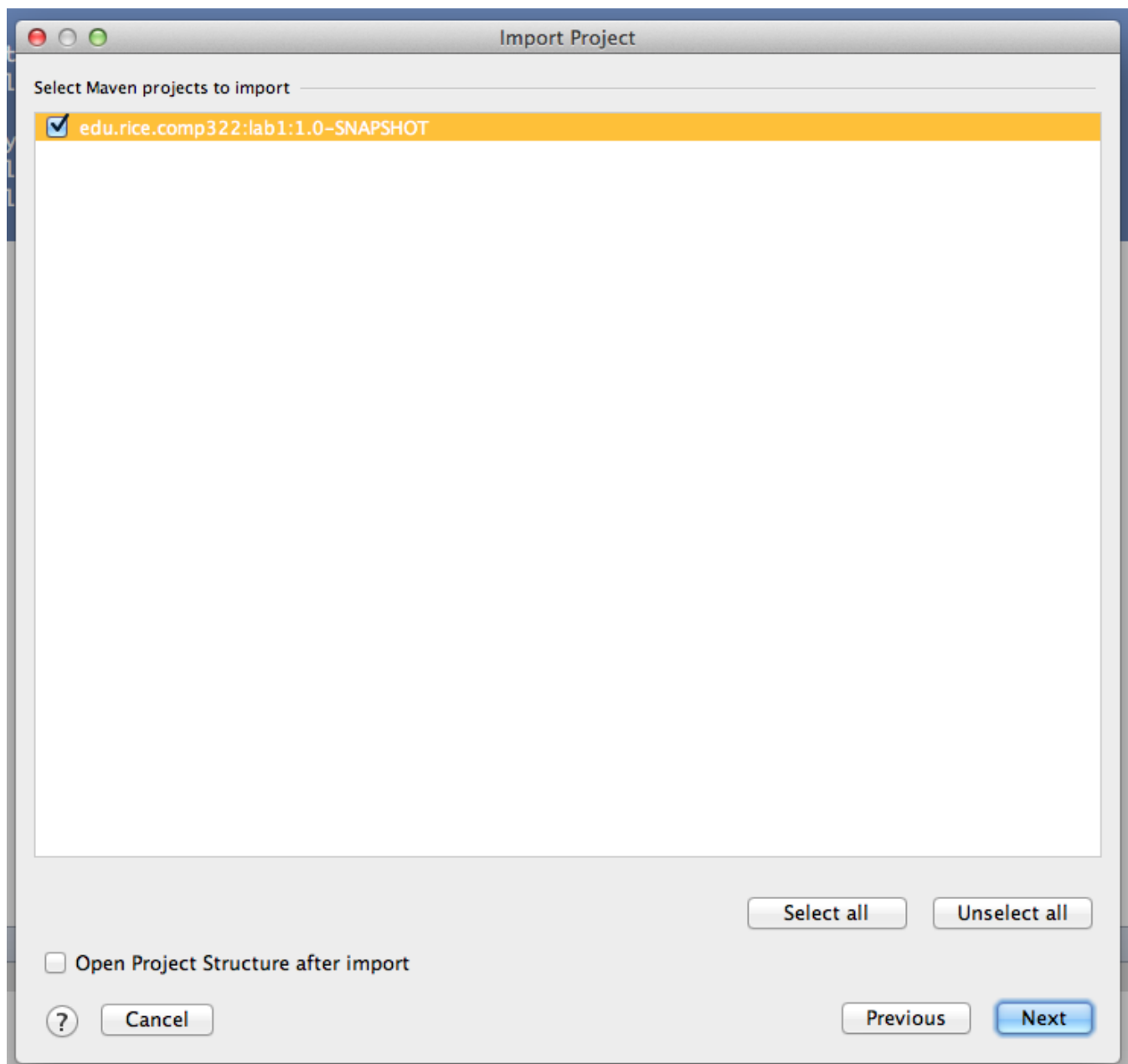
Dependency types:

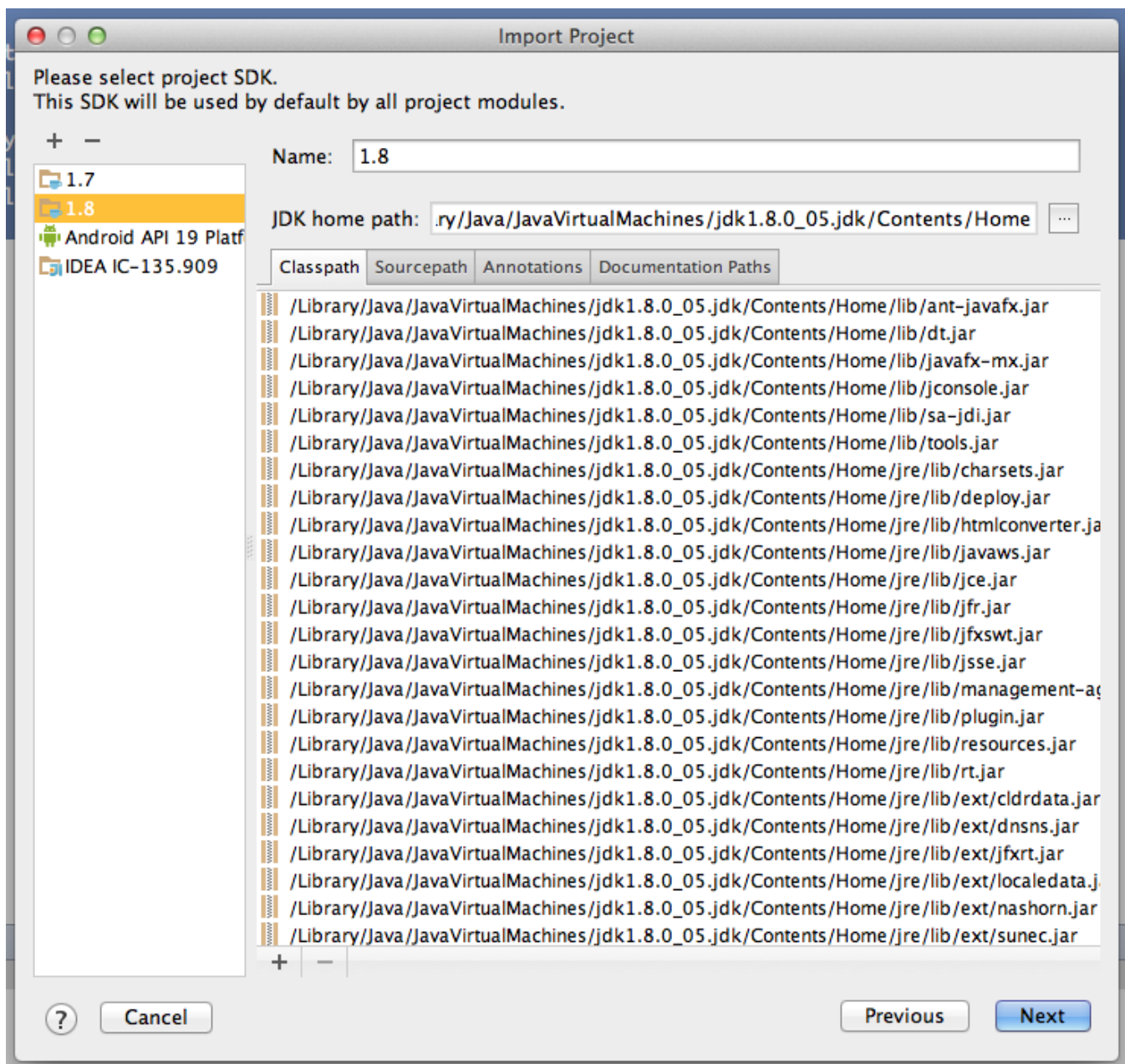
Comma separated list of dependency types that should be imported

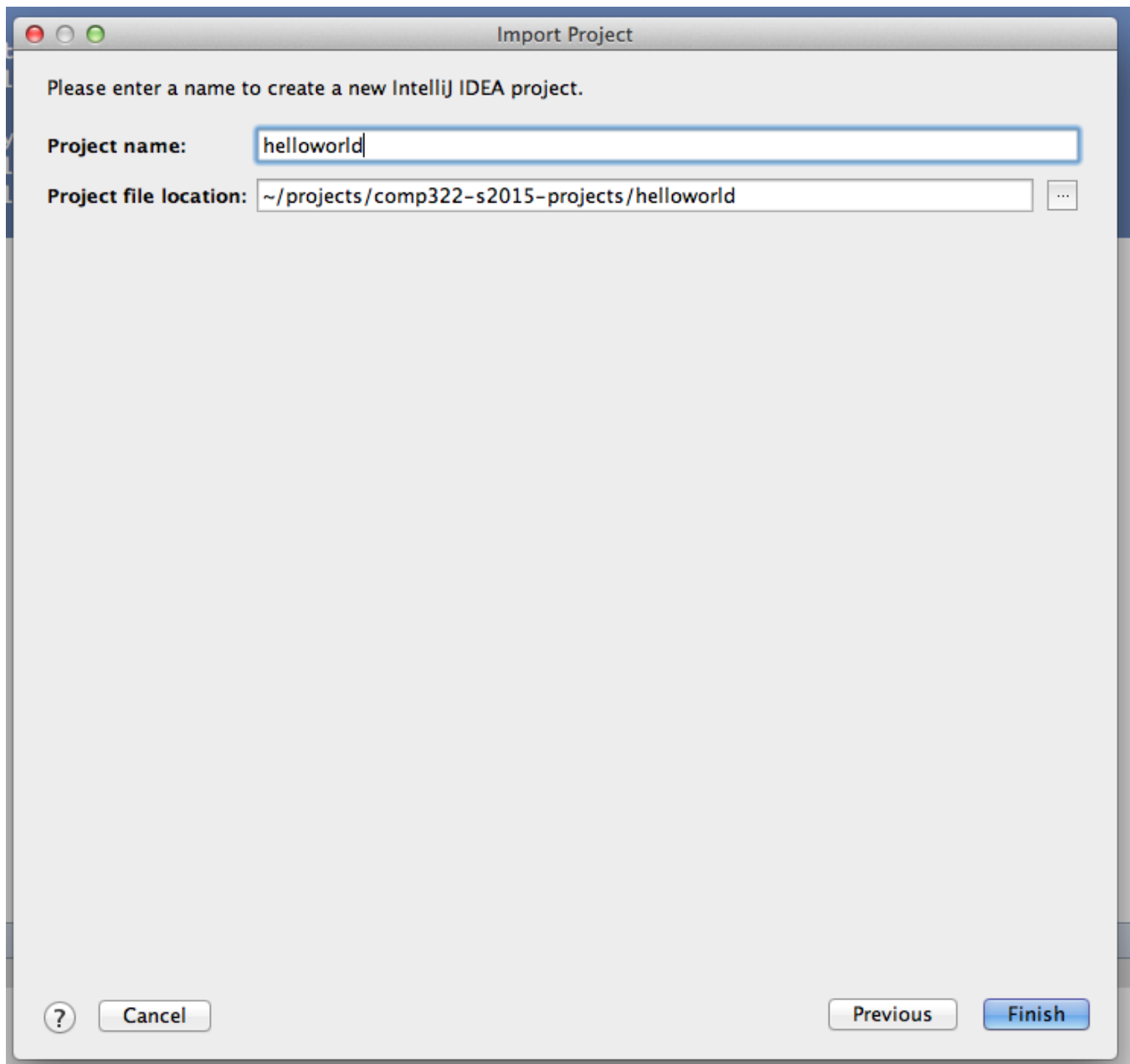
Environment settings...

? Cancel Previous Next



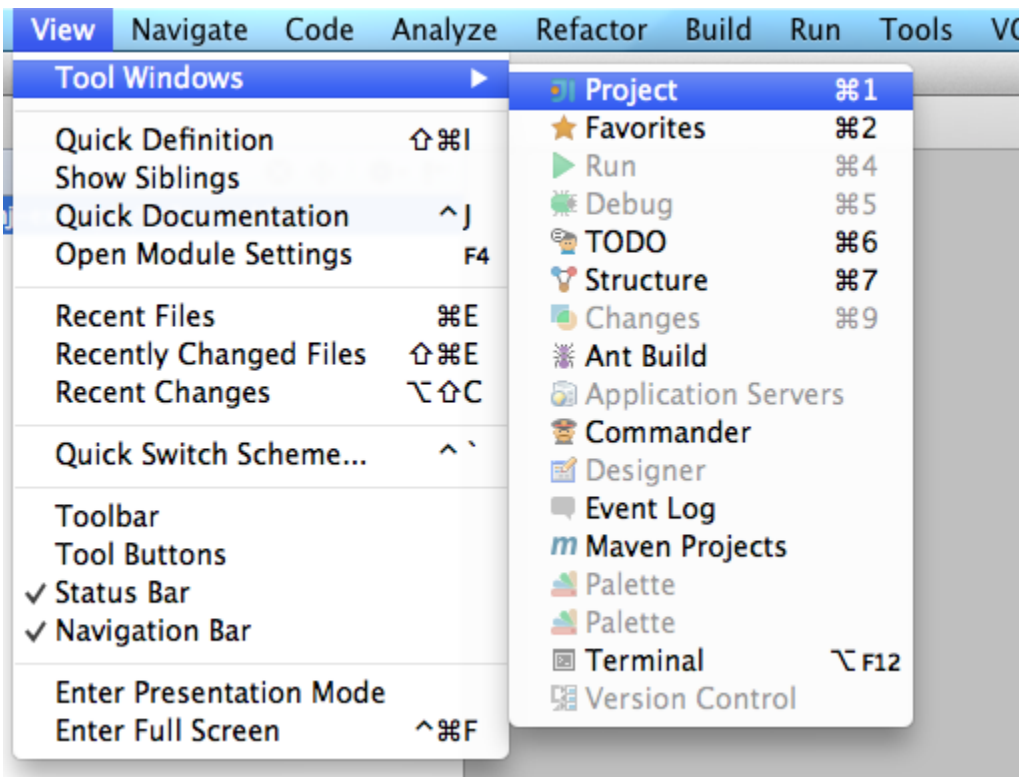


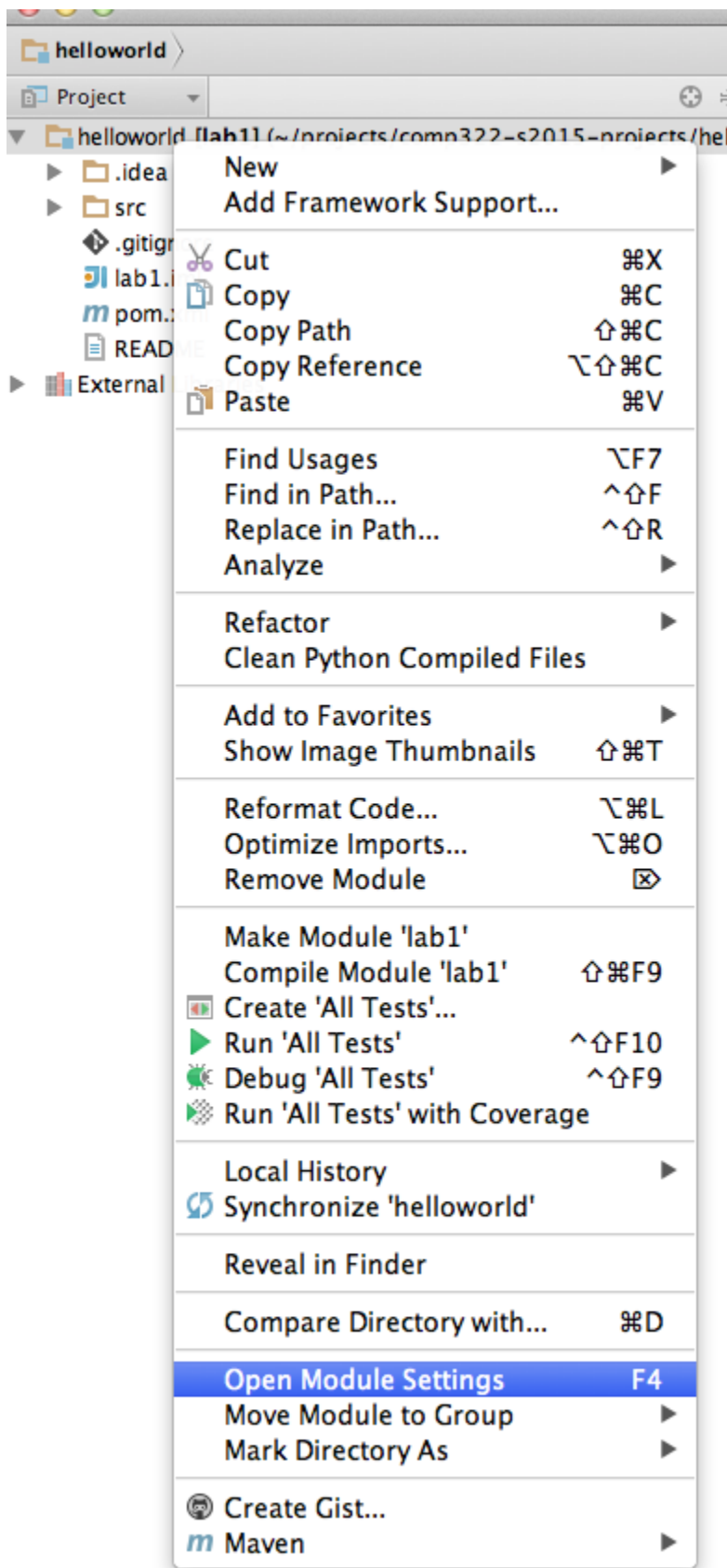




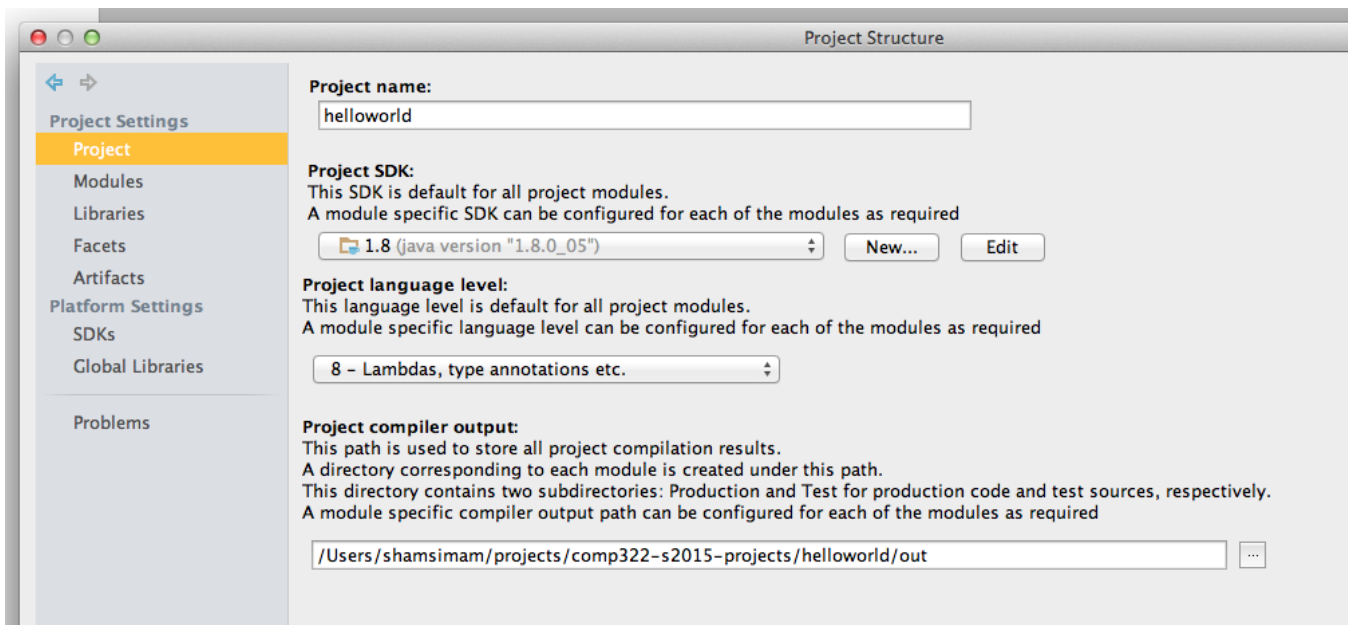
First enable the project view, then right-clicking on the root folder to change the module settings.



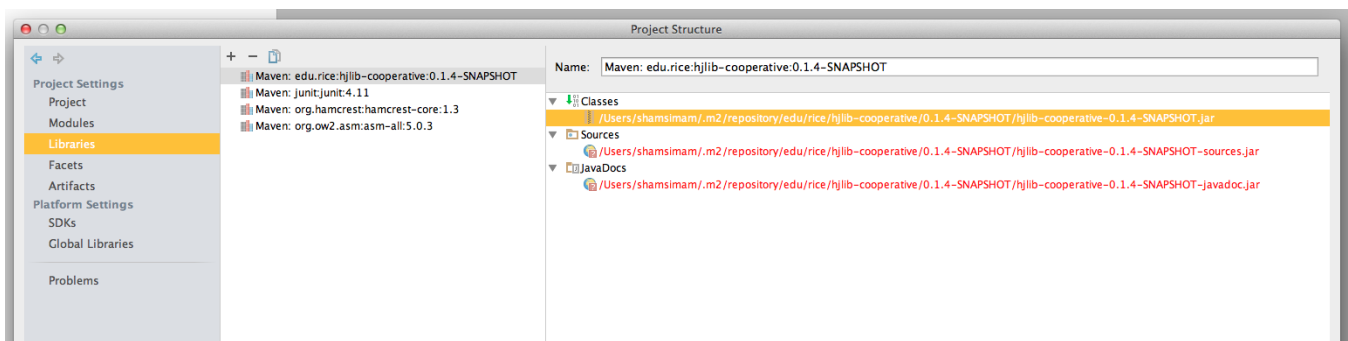




We will be using Java 8 lambdas while developing with HJlib, so we need to ensure the language level settings in the project:

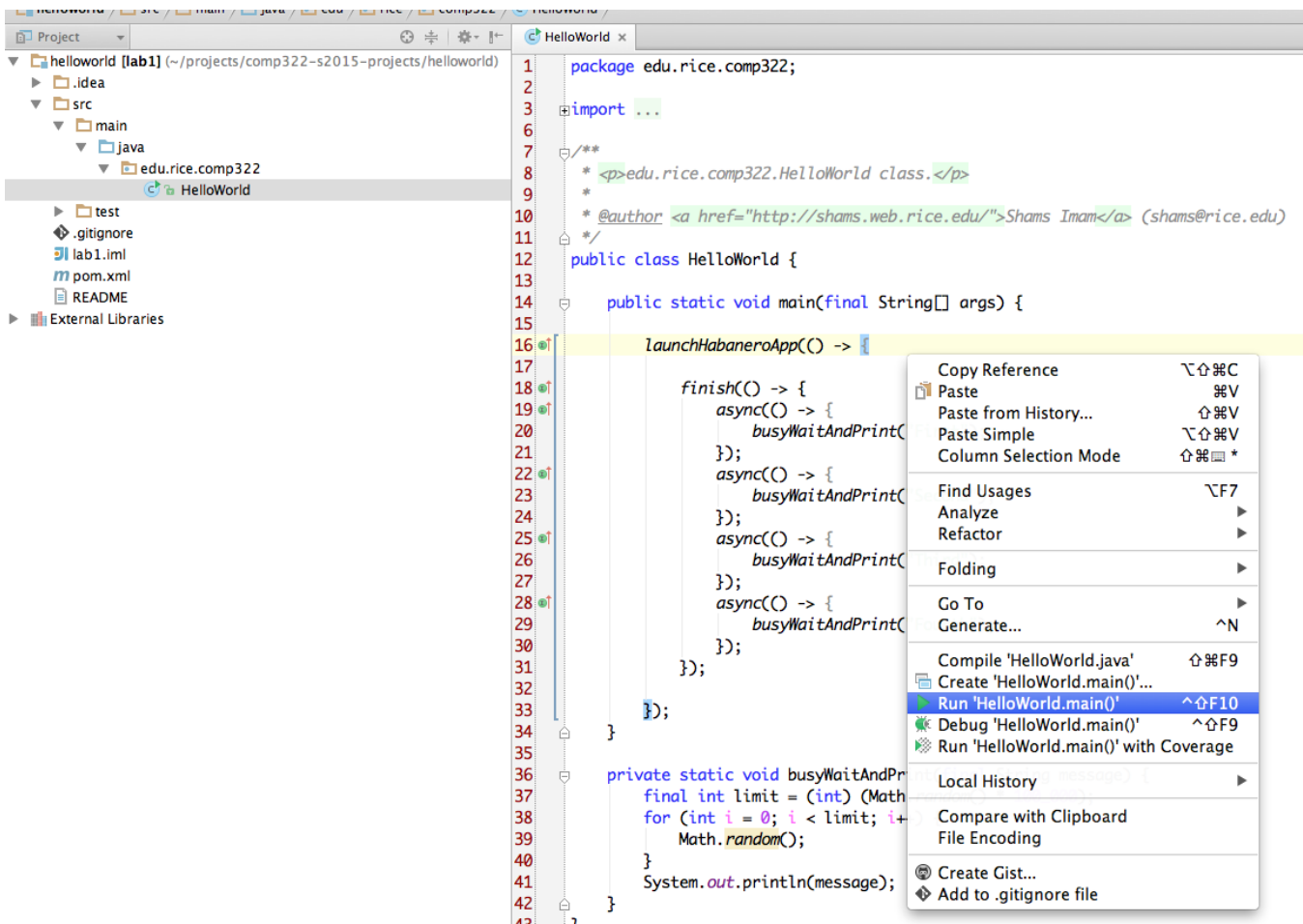


Notice, the library dependencies have already been resolved thanks to maven:

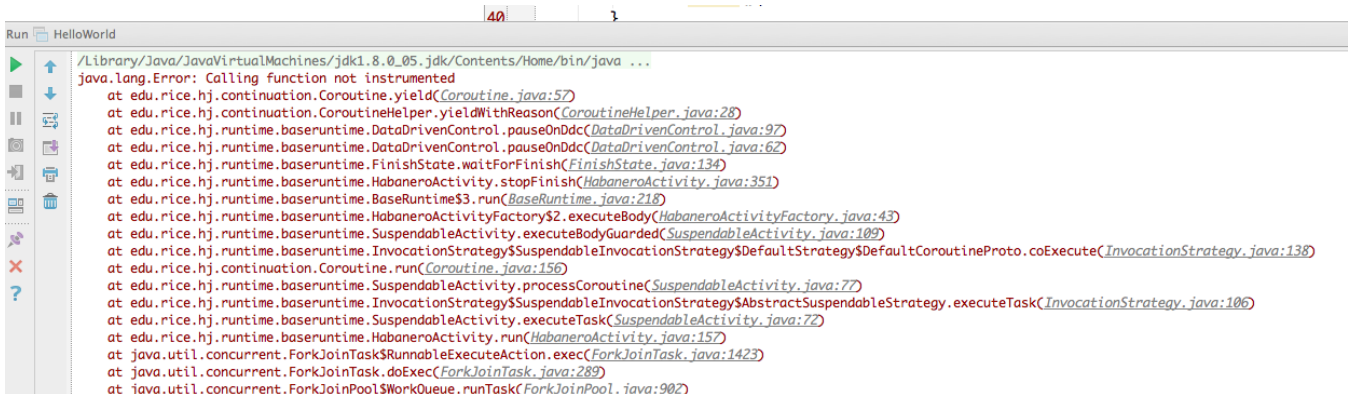


You will need to note the path of the HJlib jar file as we will need it to be able to run the `HelloWorld.java` program from IntelliJ. On my machine it is at `/Users/shamsimam/.m2/repository/edu/rice/hjlib-cooperative/0.1.4-SNAPSHOT/hjlib-cooperative-0.1.4-SNAPSHOT.jar`

Open the `HelloWorld.java` file in the editor and attempt to run it by right clicking on it and choosing the Run option:



Since we are running HelloWorld.java without configuring the javaagent option, it is likely to intermittently fail with an error as follows: java.lang. Error: Calling function not instrumented



Or the following error:

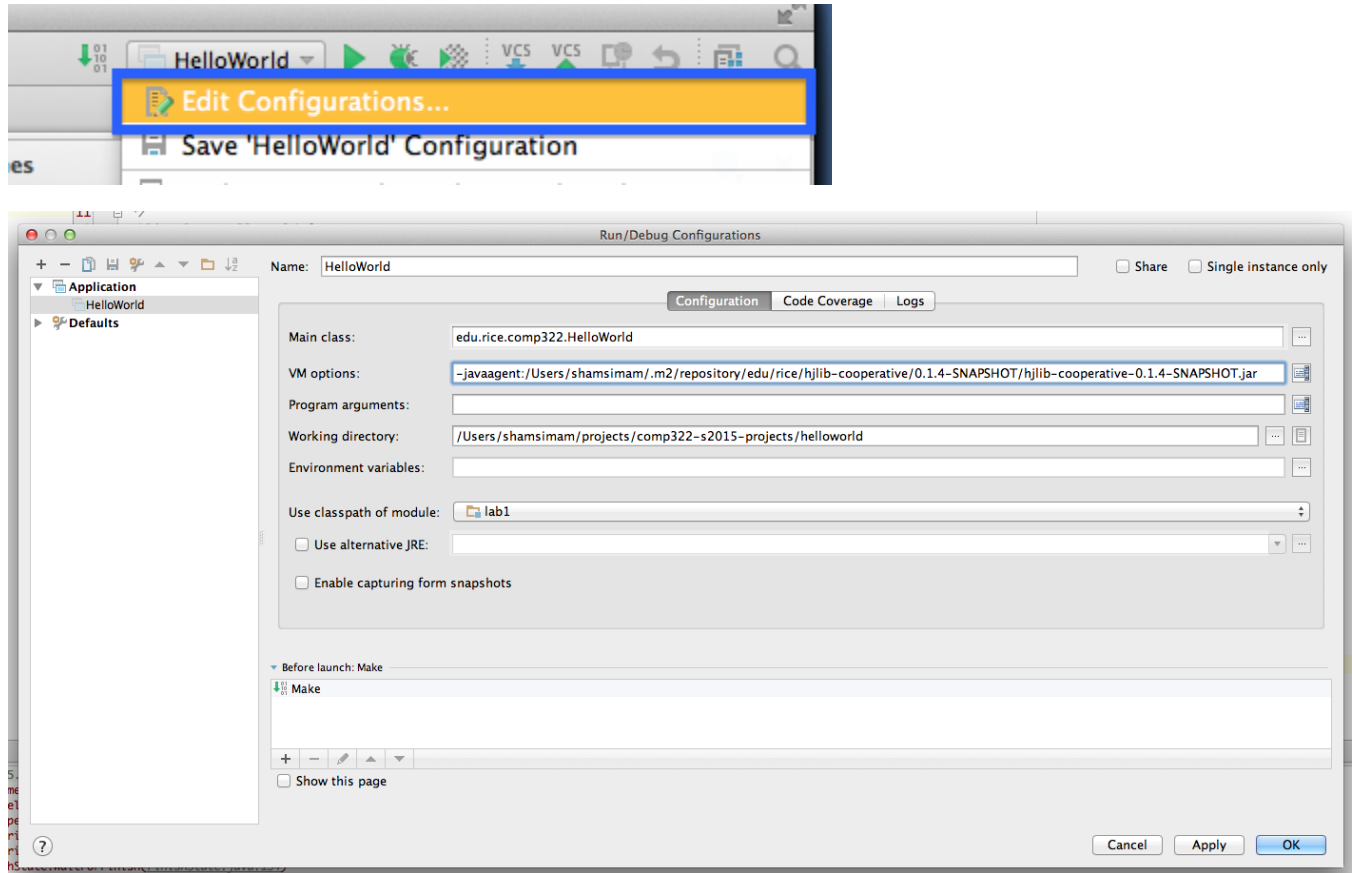
```
Error occurred during initialization of VM
agent library failed to init: instrument
```

To fix this error, we need to correct set up the javagent JVM option before attempting to run HelloWorld.java.

Java agents provide services that allow Java programming language agents to instrument programs running on the JVM. To run HJlib programs under the cooperative runtime we need to configure the JVM to use agents available in the HJlib jar. An agent is started by adding this option to the command-line:

```
-javaagent:jarpath[=options]
```

We need to configure this in IntelliJ to be able to successfully run HJlib programs. Below is an image of what the configuration looks like after editing the run configuration:



Now we can run the program by clicking on the green play button.  
Running the file on my machine produces the following output for example:

