HJlib (Thread-blocking Runtime) Download and Set Up

HJlib provides two jars for JDK8 and JDK7 compatibility. Installation instructions for both these are similar, except that the HJlib jar file needs to be different.
JDK8 installation instructions are mentioned first followed by instructions for JDK7.

JDK8 compatible HJlib

Here are the steps for running HJlib on Java 8 using an IDE:

Step 1: Java 8 Installation

You will also need a Java 8 installation on your machine and have your JAVA_HOME and PATH point to the new installation. Java 8 can be downloaded and installed from the Oracle website.

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

```
  JAVA8_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home
  export JAVA_HOME=${JAVA8_HOME}
  export PATH=$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 JDK8 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.

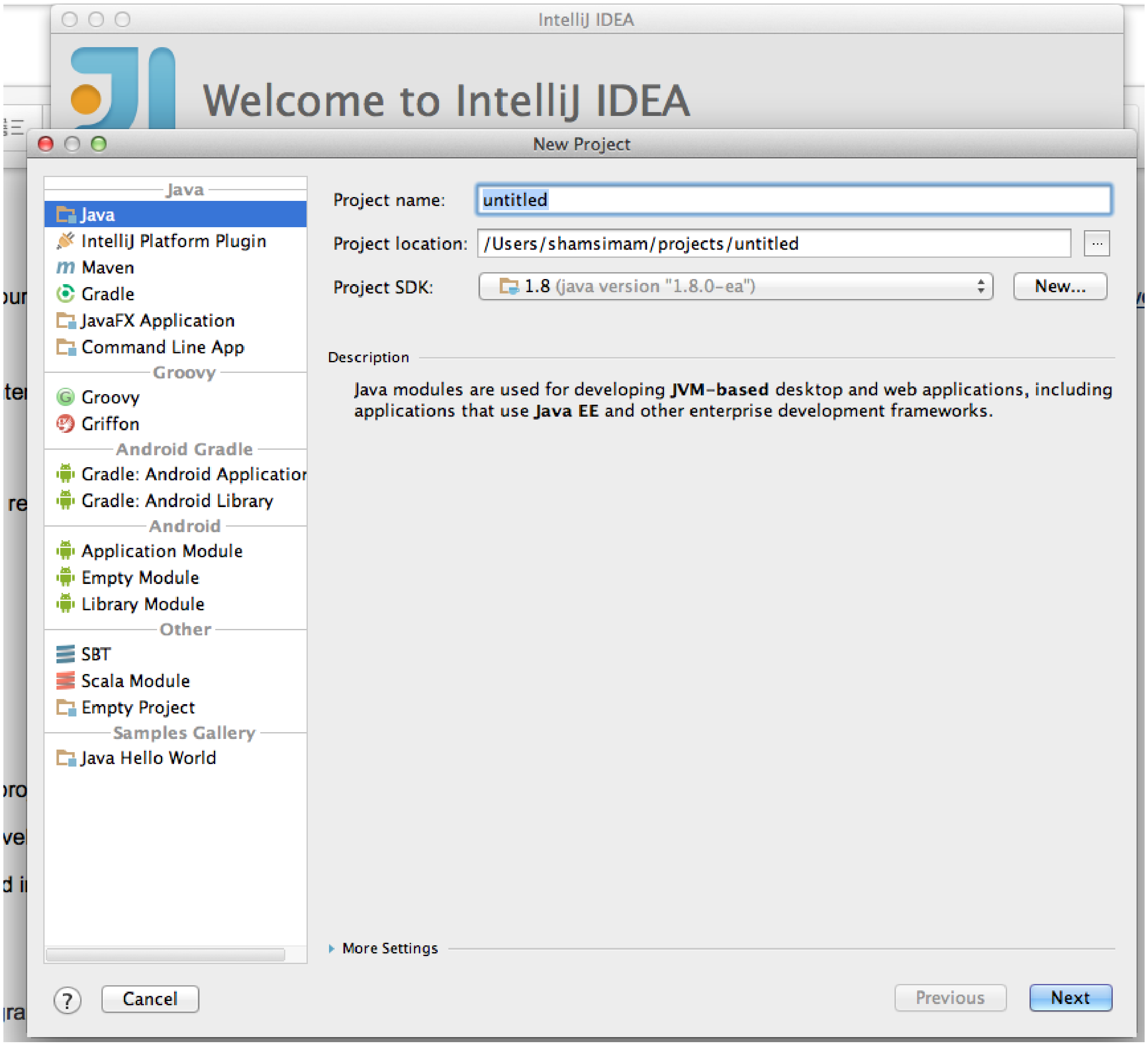
However, we recommend using an IDE like IntelliJ to do the Java development using HJlib in the labs and assignments.

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 project with a HJlib HelloWorld program in IntelliJ. Similar steps can be followed for other IDEs like Eclipse.

First, create a new project using the File -> New Project Menu. This should show a popup window as follows:



Project name: untitled

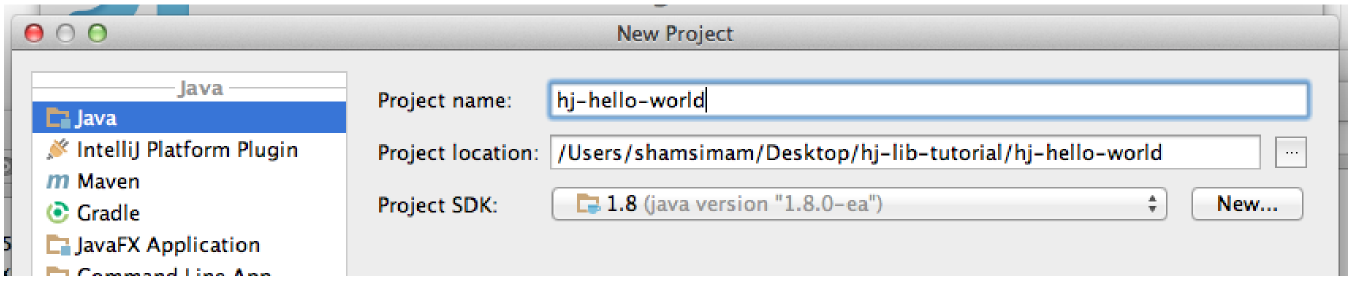
Project location: /Users/shamsimam/projects/untitled

Project SDK: 1.8 (java version "1.8.0-ea") New...

Description

Java modules are used for developing **JVM-based** desktop and web applications, including applications that use **Java EE** and other enterprise development frameworks.

Now, choose a name for your project and a directory location for the project. Remember to select the Java 8 SDK and click Next:

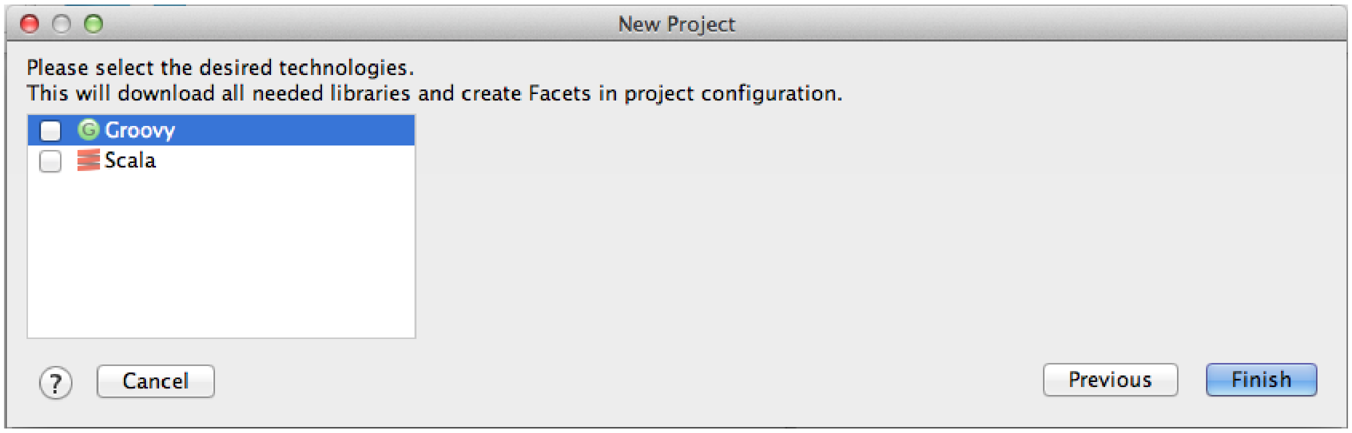


Project name: hj-hello-world

Project location: /Users/shamsimam/Desktop/hj-lib-tutorial/hj-hello-world

Project SDK: 1.8 (java version "1.8.0-ea") New...

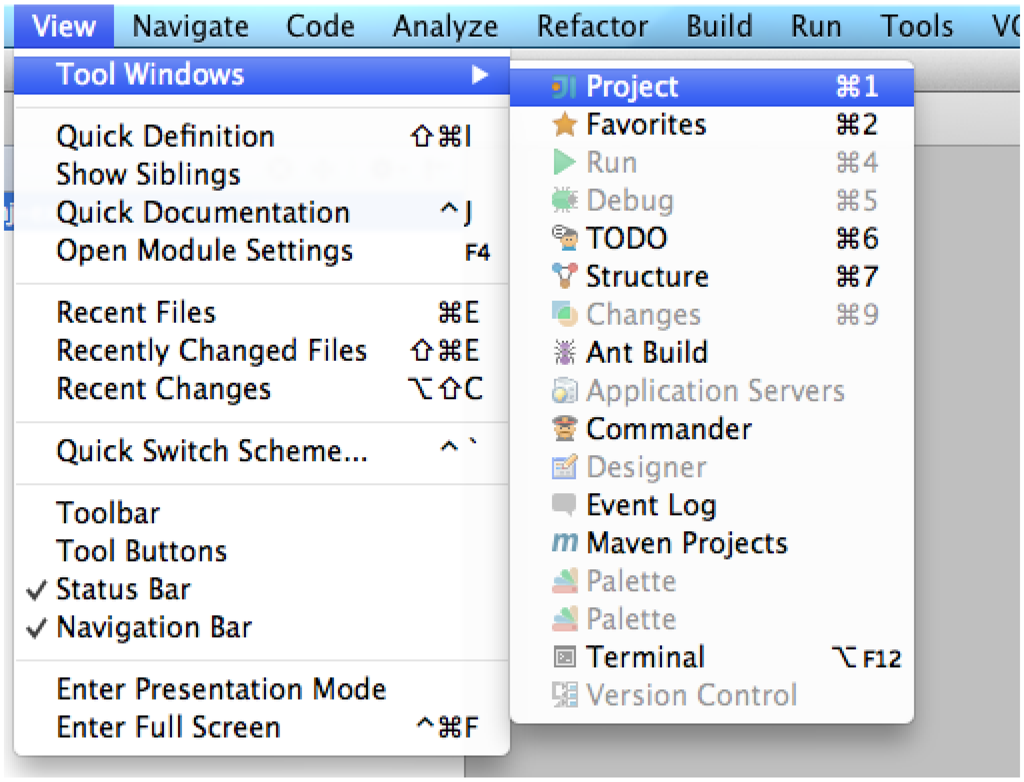
On the next window click Finish, we do not need dependencies on any other technologies.

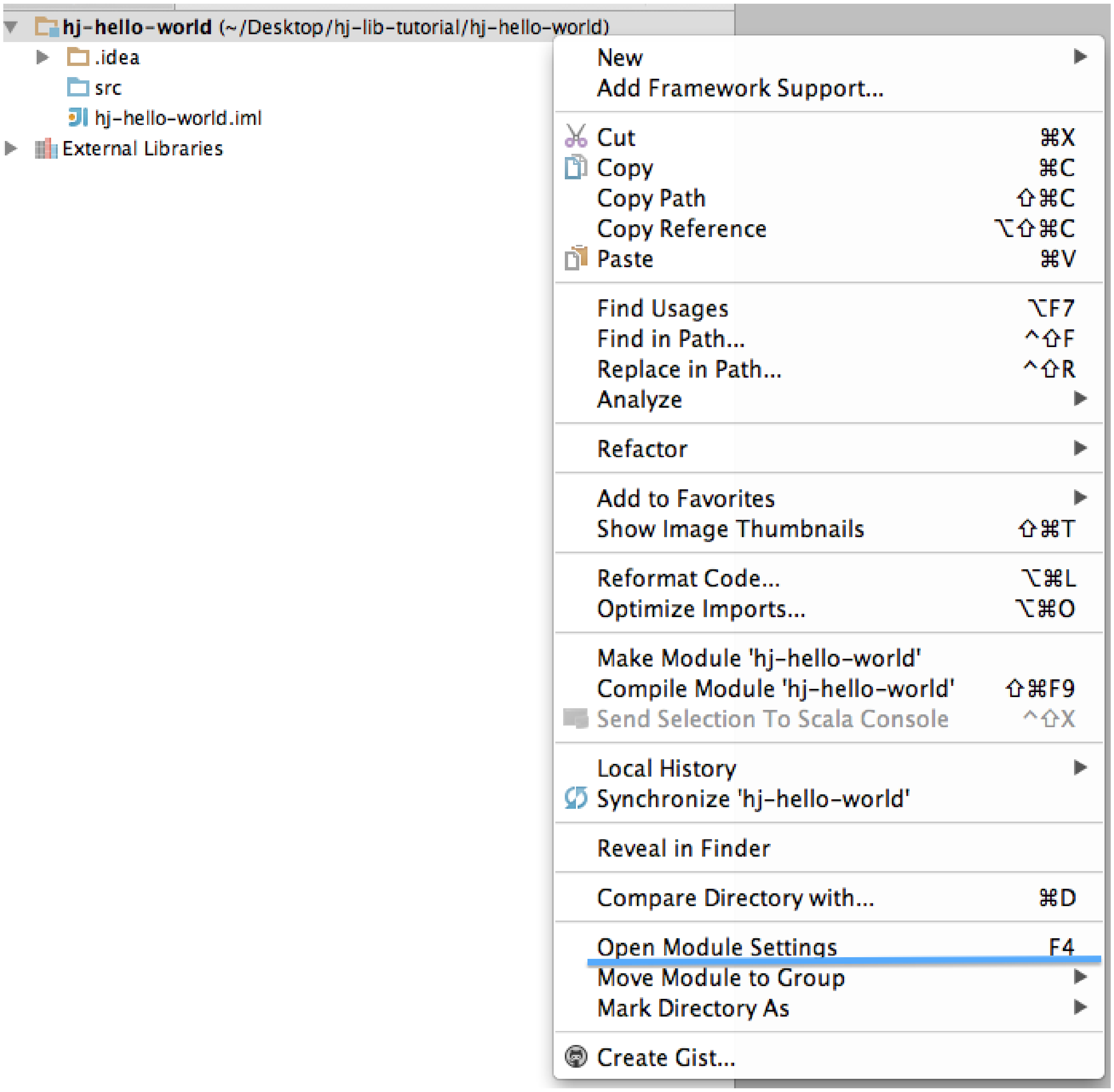

New Project

Please select the desired technologies.
This will download all needed libraries and create Facets in project configuration.

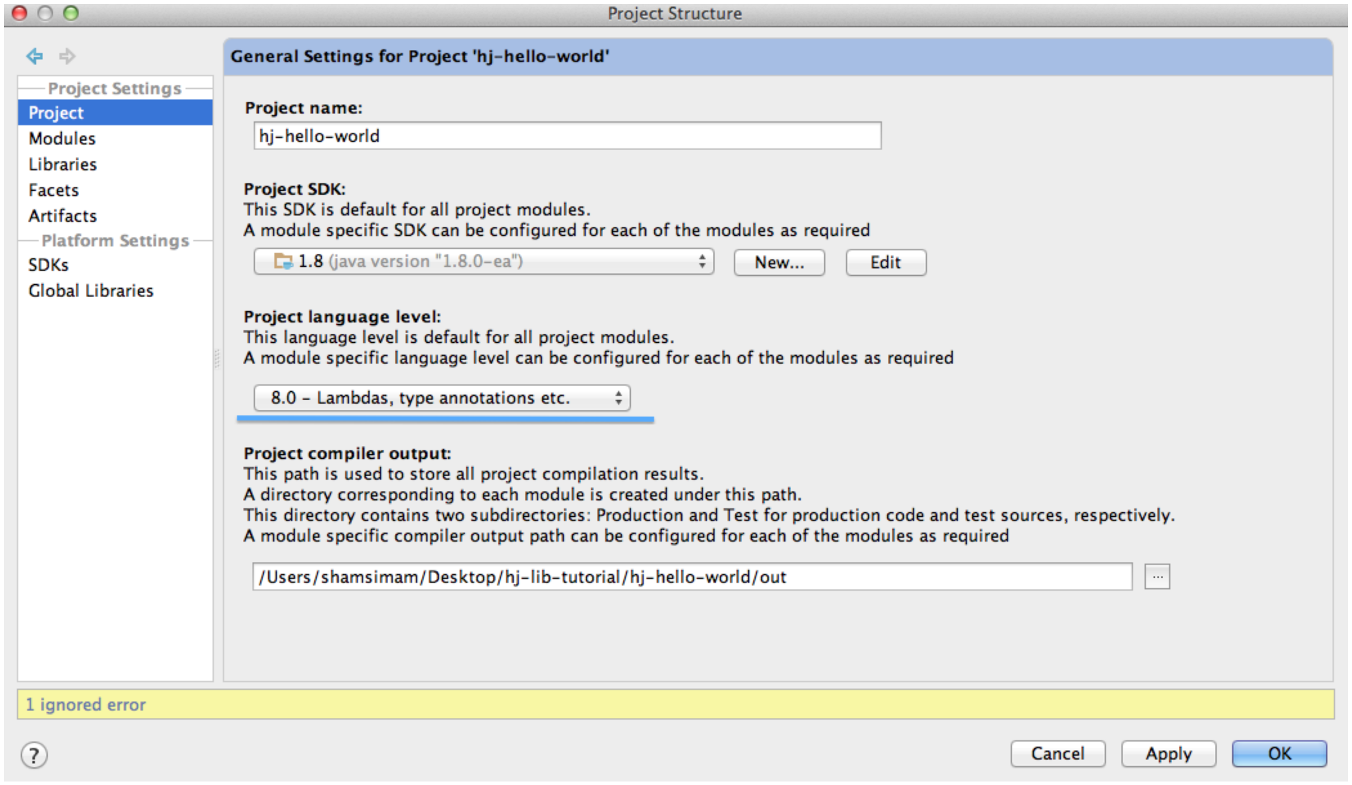
We now need to add the previously downloaded HJlib jar from Step 2 as a dependency to this project. We need to do this from the module settings.

First enable the project view, befretrying 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:


Project Structure

General Settings for Project 'hj-hello-world'

Project name:
hj-hello-world

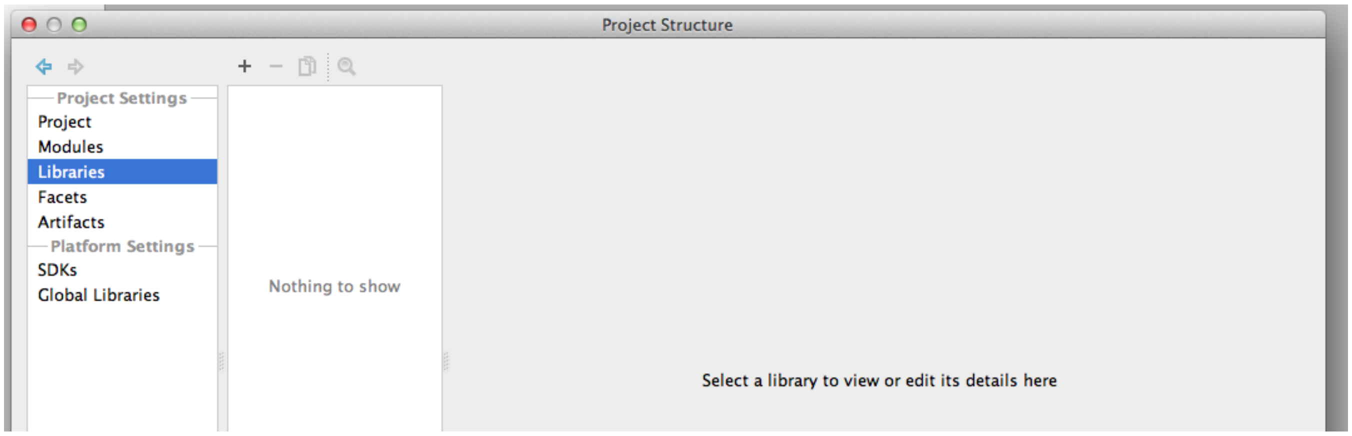
Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required
1.8 (java version "1.8.0-ea")

Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required
8.0 - Lambdas, type annotations etc.

Project compiler output:
This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.
This directory contains two subdirectories: Production and Test for production code and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required
/Users/shamsimam/Desktop/hj-lib-tutorial/hj-hello-world/out

1 ignored error

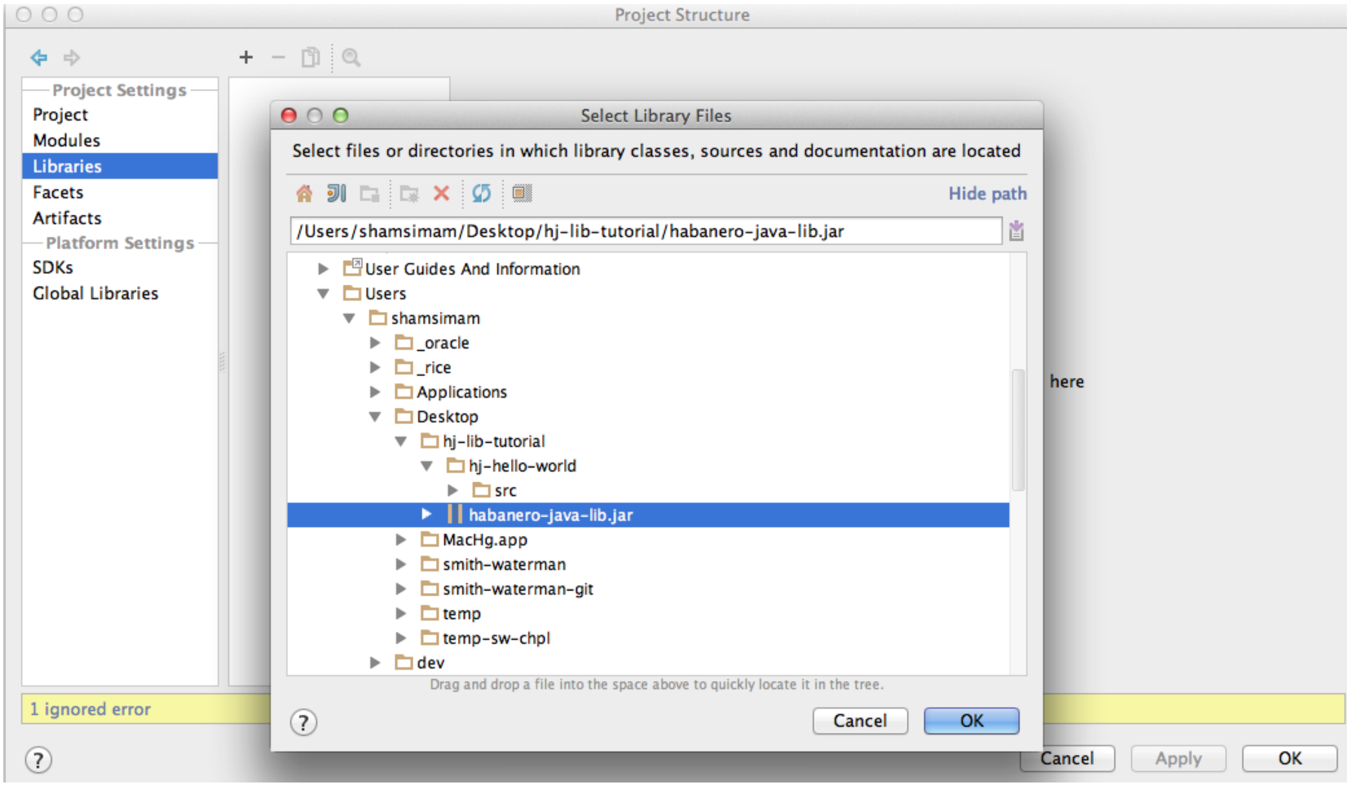
Next, we need to add the library dependency:


Project Structure

Nothing to show

Select a library to view or edit its details here

Clicking on the plus icon and selecting a Java library allows us to choose the location for the jar file:



Select files or directories in which library classes, sources and documentation are located

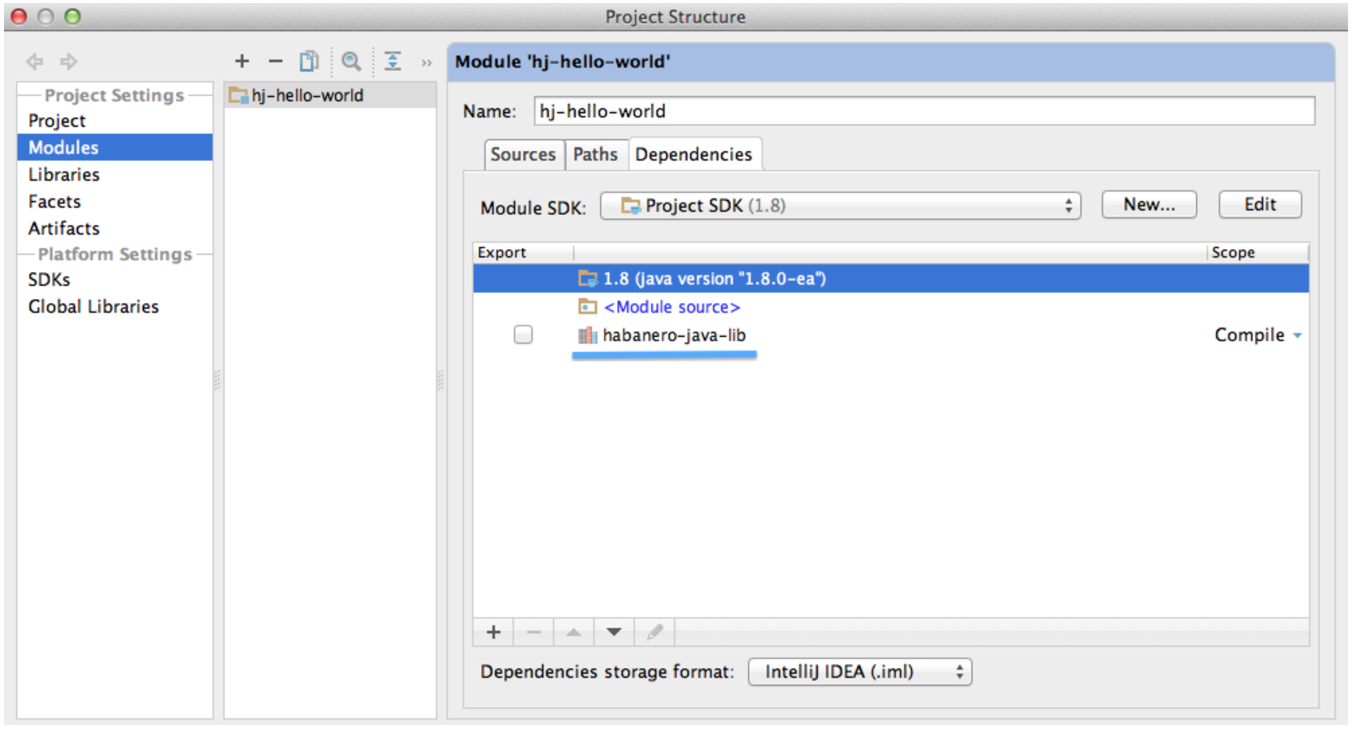
/Users/shamsimam/Desktop/hj-lib-tutorial/habanero-java-lib.jar

```
▶ User Guides And Information
▼ Users
  ▼ shamsimam
    ▶ _oracle
    ▶ _rice
    ▶ Applications
    ▼ Desktop
      ▼ hj-lib-tutorial
        ▼ hj-hello-world
          ▶ src
        ▶ habanero-java-lib.jar
      ▶ MacHg.app
      ▶ smith-waterman
      ▶ smith-waterman-git
      ▶ temp
      ▶ temp-sw-chpl
    ▶ dev
```

Drag and drop a file into the space above to quickly locate it in the tree.

Once added, we can click 'Apply' to save our changes.

Ensure that habanero-java-lib has been added as a dependency to the module:



Module 'hj-hello-world'

Name: hj-hello-world

Module SDK: Project SDK (1.8)

Export		Scope
	1.8 (java version "1.8.0-ea")	
	<Module source>	
☐	habanero-java-lib	Compile

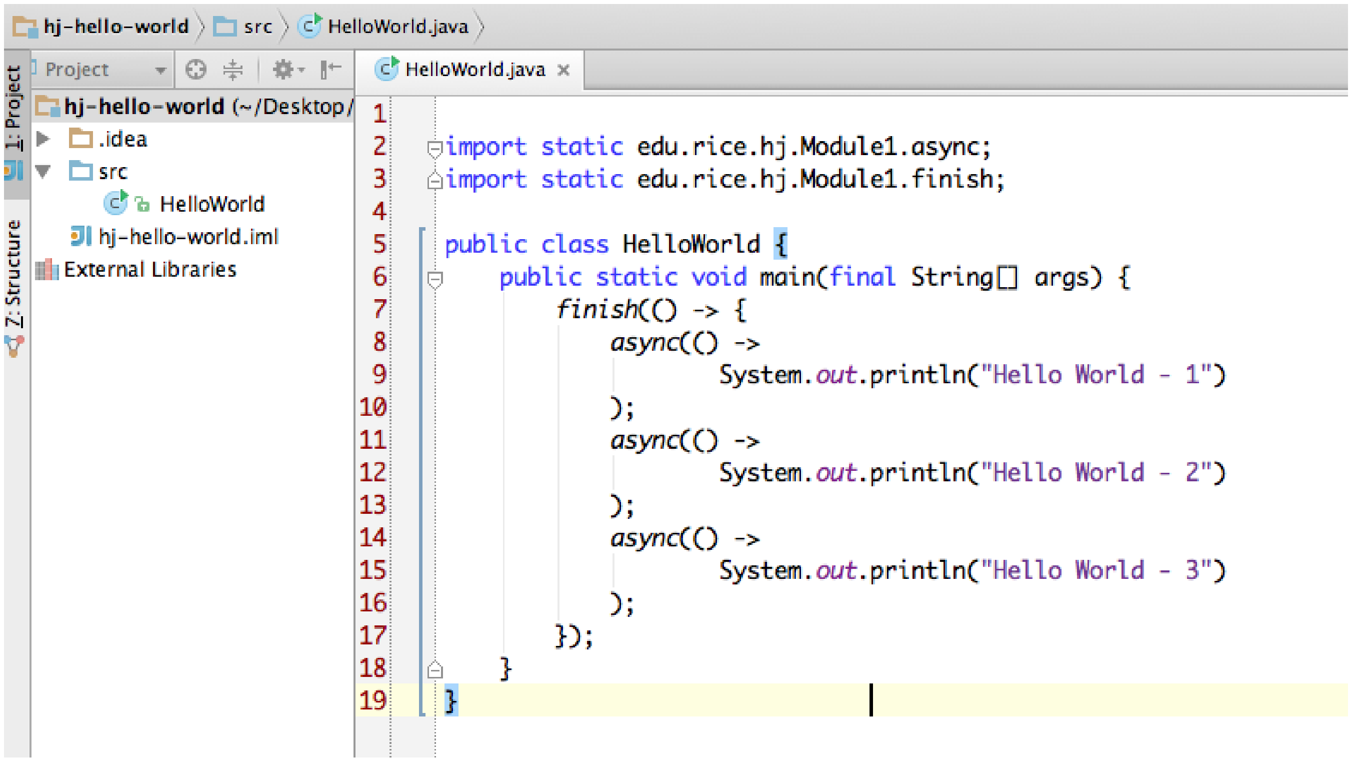
Dependencies storage format: IntelliJ IDEA (.iml)

After confirming, click "OK" to save your changes.

We are now ready to write our first HJ application using HJlib.

To create a java file, right click on the 'src' folder and select New -> Java Class.

Use 'HelloWorld' as the name of the class and type in the sample program:

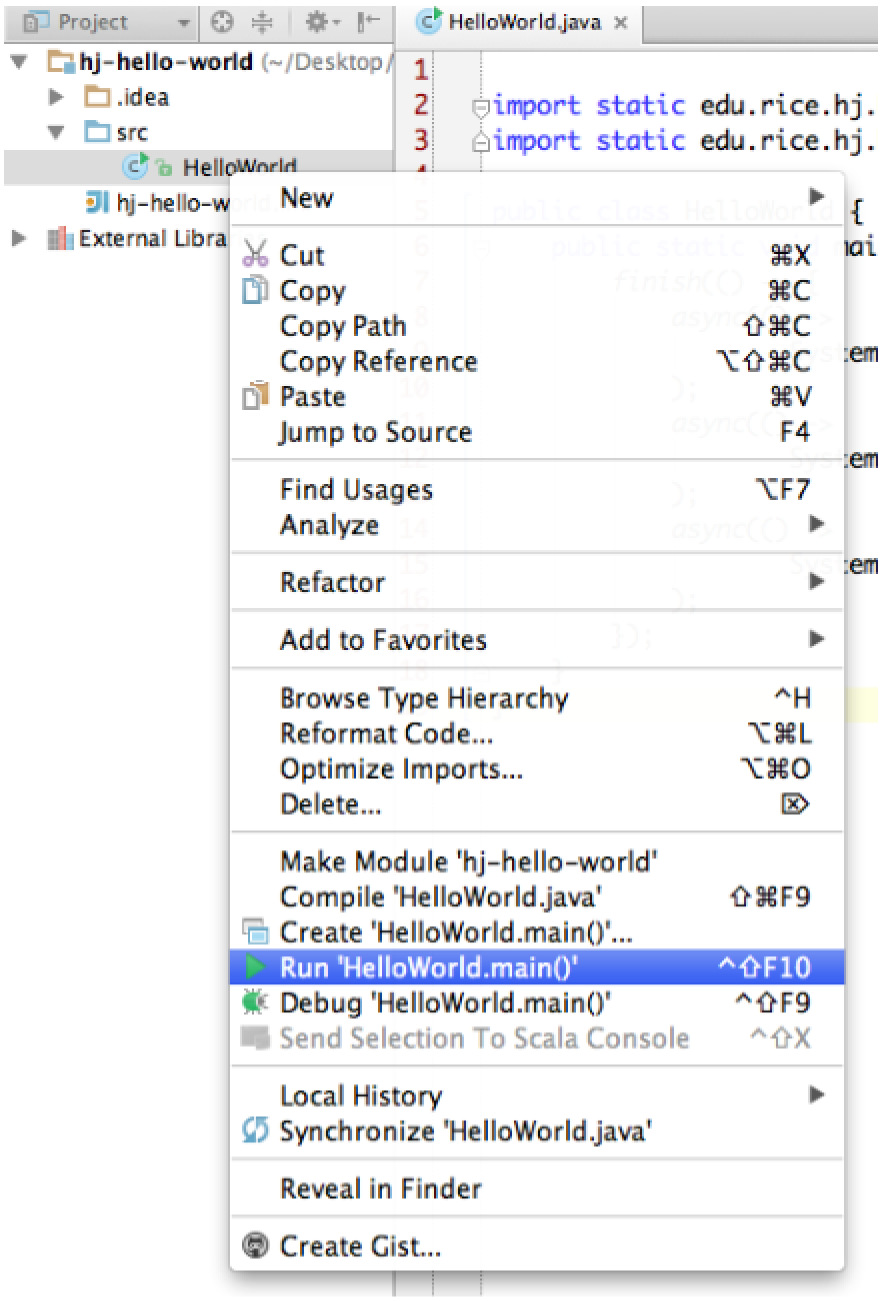


```java

import static edu.rice.hj.Module1.async;
import static edu.rice.hj.Module1.finish;

public class HelloWorld {
    public static void main(final String[] args) {
        finish(() -> {
            async(() ->
                    System.out.println("Hello World - 1")
            );
            async(() ->
                    System.out.println("Hello World - 2")
            );
            async(() ->
                    System.out.println("Hello World - 3")
            );
        });
    }
}
```

Here is the source code:

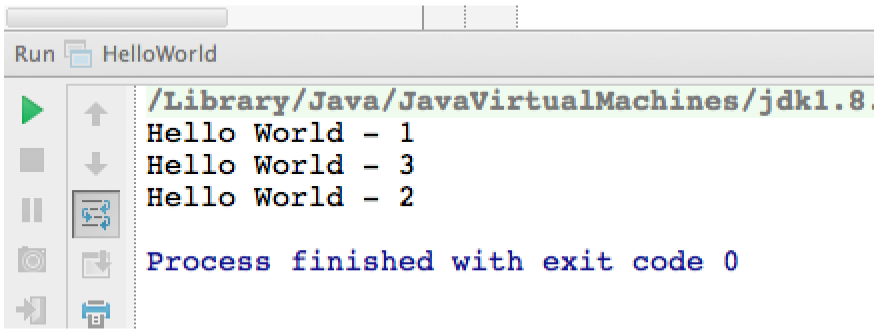
```java
import static edu.rice.hj.Module1.async;
import static edu.rice.hj.Module1.finish;

public class HelloWorld {
    public static void main(final String[] args) {
        finish(() -> {
            async(() ->
                    System.out.println("Hello World - 1")
            );
            async(() ->
                    System.out.println("Hello World - 2")
            );
            async(() ->
                    System.out.println("Hello World - 3")
            );
        });
    }
}
```

Now we can run this program by right clicking on the HelloWorld item on the project explorer and clicking run.



```
import static edu.rice.hj.
import static edu.rice.hj.
```

Running the file on my machine produces the following output for example:



```
/Library/Java/JavaVirtualMachines/jdk1.8.
Hello World – 1
Hello World – 3
Hello World – 2

Process finished with exit code 0
```

JDK7 compatible HJlib

A Java 7 compatible jar is available for download here: habanero-java-lib-0.1.2-SNAPSHOT-jdk7.jar

An IntelliJ project can be set up similar to the above instructions and the language level set to Java 7.

Code examples of HJlib with JDK7 compatibility are available here: examples directory