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
- JDK7 compatible HJlib

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 <u>Oracle website</u>.

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:

	000		IntelliJ IDEA	
1-	Welco	ome to l	ntelliJ IDEA	
- (00		New Project	
1	Java Java ∭ Intellij Platform Plugin Maven	Project name: Project location: Project SDK:	Untitled /Users/shamsimam/projects/untitled	
our	Gradie	Description		
tei	 Groovy Griffon Android Gradle 	Java modules a applications th	are used for developing JVM-based desktop and web applications, ir hat use Java EE and other enterprise development frameworks.	icluding
re	Gradle: Android Library Android Android			
ve ve				
Ira	? Cancel	• More Settings	Previous	Next

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

	00	New Project	
	Java	Project name: hj-hello-world	
	💉 IntelliJ Platform Plugin	Project location: /Users/shamsimam/Desktop/hj-lib-tutorial/hj-hello-world	
	<i>m</i> Maven ⓒ Gradle	Project SDK: 🔁 1.8 (java version "1.8.0-ea") 🗘 New	
5	JavaFX Application		

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

00	New Project		
Please select the desired technologie This will download all needed librarie	s. is and create Facets in project configuration.		
🔲 🌀 Groovy			
📄 📕 Scala			
? Cancel		Previous	Finish

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.

View Navigate Code	Analyze	Refactor Build	Run Tools V
Tool Windows	•	I Project	% 1
Quick Definition Show Siblings	企業I	★ Favorites ▶ Run	₩2 ₩4
Quick Documentation Open Module Settings	^ J F4	₩ Debug ™ TODO Structure	第5 第6 第7
Recent Files Recently Changed Files Recent Changes	¥E ዕжE ጋዕፓ	Changes Ant Build Application Se	#9 ervers
Quick Switch Scheme	^`	Commander Designer	
Toolbar Tool Buttons ✓ Status Bar ✓ Navigation Bar		 Event Log Maven Project Palette Palette 	5
Enter Presentation Mode Enter Full Screen	^₩F	Version Contr	ol

hj-hello-world (~/Desktop/hj-lib-tutorial/hj-hello-world)			
 idea src 	New Add Framewo	ork Support	•
External Libraries	 ✓ Cut ☑ Copy Copy Path Copy Referent ☑ Paste 	ice	X光 つ光 つ光ひ つ光ひブ V光 V光
	Find Usages Find in Path Replace in Pa Analyze	th	℃F7 ^☆F ^☆R
	Refactor		►
	Add to Favor Show Image	ites Thumbnails	► ک#T
	Reformat Coo Optimize Imp	de oorts	て第L て第O
	Make Module Compile Mod Send Selection	e 'hj-hello-world' lule 'hj-hello-world' on To Scala Console	ዕଞ୍ଜF9 ^습X
	Local History Ø Synchronize	'hj-hello-world'	•
	Reveal in Fine	der	
	Compare Dire	ectory with	жD
	Open Module	Settings	F4
	Move Module Mark Directo	e to Group ry As	* *
	Create Gist		

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

 $\Theta \cap \Theta$

Project Structure

存 ⇒	General Settings for Project 'hj-hello-world'
Project Settings Modules Libraries Facets Artifacts Platform Settings SDKs Global Libraries	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 Image: I.8 (java version "1.8.0-ea") + New Edit Project language level: This slaguage 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 office compiler output path can be configured for each of the modules as required /Users/shamsimam/Desktop/hj-lib-tutorial/hj-hello-world/out Image:
1 ignored error	
?	Cancel Apply OK

Next, we need to add the library dependency:

00		Project Structure	
(= ⇒	+ - 🖻 🔍		l
Project Settings Project Modules Libraries Facets Artifacts Platform Settings SDKs Global Libraries	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:



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

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

$\Theta \cap \Theta$		Project Structure	
	+ - 🖻 🔍 호 »	Module 'hj-hello-world'	
Project Settings Project Modules Libraries Facets Artifacts Platform Settings SDKs Global Libraries	▶ hj-hello-world	Name: hj-hello-world Sources Paths Dependencies Module SDK: Project SDK (1.8) + New Export Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source > Image: All of the source >	Edit Scope Compile •

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:



Here is the source code:

```
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.

🔻 🗖 src	World	3 Gimpo	rt static	edu.rice.	hj
J hj-hello-w	New				£
▶ 💼 External Libra	Cut Copy Copy Path Copy Refe Paste Jump to Sc	rence ource		第X 第C 企業C て企業C て公業C 業V F4	ia :e
	Find Usage Analyze	es)	℃F7 ►	e
	Refactor			•	e
	Add to Fav	orites		►	
	Browse Ty Reformat (Optimize I Delete	pe Hierarch Code mports	y	H^ 」第了 〇第了 図	
	Make Mod Compile 'He Create 'He	ule 'hj-hello IelloWorld.ji IloWorld.ma	o-world' ava' ain()'	☆業F9	
	Run 'Hello'	World.main	0'	^	٩.
	Send Selec	tion To Sca	la Console	· ^ 쇼F9 · ^ 쇼X	l
	Local Histo	ory ze 'HelloWo	rld.java'	Þ	
	Reveal in F	inder			
	Create Gis	t			

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



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