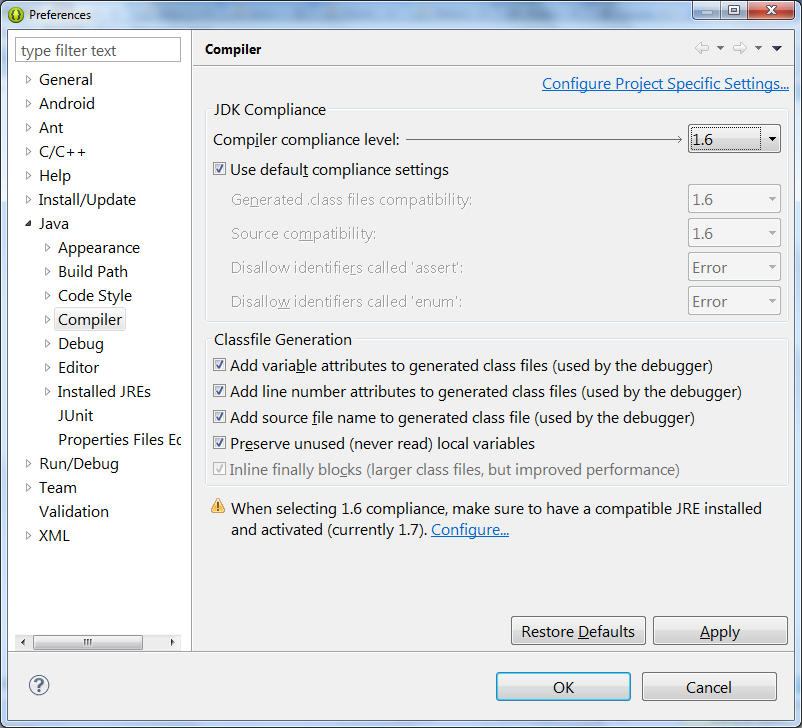
**Eclipse Users:**

For old developers who prefer Eclipse, [google stops support Eclipse Android Developer tools](https://developer.android.com/sdk/installing/installing-adt.html)

if you installed Java 8 JDK, then give it a try, if any problems appears try to set the compiler as 1.6 in Eclipse from window menu → **Preferences** → **Java** → **Compiler**. Java 7 will works too:



Java 7 or higher is required if you are targeting Android 5.0 and higher.

install multiple JDK and try.

[share](https://stackoverflow.com/a/23318269)[improve this answer](https://stackoverflow.com/posts/23318269/edit)

[edited Jun 21 '17 at 22:22](https://stackoverflow.com/posts/23318269/revisions)

answered Apr 27 '14 at 1:10

[[https://www.gravatar.com/avatar/31090e3d12227ea3361a13262030dc56?s=32&d=identicon&r=PG](https://stackoverflow.com/users/1924299/eng-samer-t)](https://stackoverflow.com/users/1924299/eng-samer-t)

[Eng. Samer T](https://stackoverflow.com/users/1924299/eng-samer-t)

**4,645**32631

* 2

Did Java 8 change the way the bytecode works? If not, it should be fine to use any version of Java, theoretically, as long as it compiles. – [Ben Leggiero](https://stackoverflow.com/users/3939277/ben-leggiero) [Mar 12 '15 at 14:13](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment46269176_23318269)

* 6

the correct question is: what if you use new features in java 8 not exists in java 7 to build android app, is it ok? to be on safe side follow official instructions. – [Eng. Samer T](https://stackoverflow.com/users/1924299/eng-samer-t) [Mar 12 '15 at 20:12](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment46282472_23318269)

* 12

@BenC.R.Leggiero there's no new bytecode in Java 8, but the instructions' meanings and structures were changed [stackoverflow.com/questions/28228450/…](https://stackoverflow.com/questions/28228450/java-is-backward-compatible-but-why-we-need-to-upgrade-many-libraries-when-we-u?lq=1) [stackoverflow.com/questions/16143684/…](https://stackoverflow.com/questions/16143684/can-java-8-code-be-compiled-to-run-on-java-7-jvm) – [phuclv](https://stackoverflow.com/users/995714/phuclv) [Mar 14 '15 at 19:13](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment46343772_23318269)

* 15

Please do not post "Update N" in your answer. Rewrite it to have up-to-date information. Otherwise, SO will in a few years be as difficult to get information from as the forums it superseded. – [Aleksandr Dubinsky](https://stackoverflow.com/users/1151521/aleksandr-dubinsky) [Mar 7 '16 at 21:44](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment59374928_23318269)

* 1

:( Instant Run does NOT work with Jack Options !!! – [mythicalcoder](https://stackoverflow.com/users/2369867/mythicalcoder) [Feb 27 '17 at 22:47](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment72133620_23318269)

[show **6** more comments](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development)

341

**UPDATE 2017/11/04 - Android Studio 3.0 now has native support for Java 8.** gradle-retrolambda is now no longer needed. See <https://developer.android.com/studio/write/java8-support.html>

The above link also includes migration instructions if you are using gradle-retrolambda. Original answer below:

Android does not support Java 8. It only supports up to Java 7 (if you have kitkat) and still it doesn't have invokedynamic, only the new syntax sugar.

If you want to use lambdas, one of the major features of Java 8 in Android, you can use [gradle-retrolamba](https://github.com/evant/gradle-retrolambda). It's a gradle build dependency that integrates retrolambda, a tool that converts Java 8 bytecode back to Java 6/7. Basically, if you set the compiler in Android Studio to compile Java 8 bytecode, thus allowing lambdas, it'll convert it back to Java 6/7 bytecode which then in turn gets converted to dalvik bytecode. It's a hack for if you want to try out some JDK 8 features in Android in lieu of official support.

[share](https://stackoverflow.com/a/23318644)[improve this answer](https://stackoverflow.com/posts/23318644/edit)

[edited Nov 3 '17 at 22:30](https://stackoverflow.com/posts/23318644/revisions)

answered Apr 27 '14 at 2:14

[[https://www.gravatar.com/avatar/acca3130f907f6e48e0380285b4127bf?s=32&d=identicon&r=PG&f=1](https://stackoverflow.com/users/3577235/ekcr1)](https://stackoverflow.com/users/3577235/ekcr1)

[ekcr1](https://stackoverflow.com/users/3577235/ekcr1)

**4,264**2610

* 13

It was a bit of a struggle to get it working, but once I actually followed the right instructions it worked nicely. Note to Eclipse users: Eclipse doesn't work well with this, I suggest migrating to Android Studio (I will miss Eclipse, but that's the way it has to be for now!) – [Simon Forsberg](https://stackoverflow.com/users/1310566/simon-forsberg) [Apr 28 '14 at 10:49](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment35739317_23318644)

* 2

Nitpick: multi-catch clauses from Java 7 are NOT supported by Android. – [csvan](https://stackoverflow.com/users/1155721/csvan) [Jun 5 '14 at 9:10](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment37091394_23318644)

* 19

Multi-catch is working for me. – [Daniel Ryan](https://stackoverflow.com/users/515455/daniel-ryan) [Jun 24 '14 at 23:39](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment37738766_23318644)

* 11

Multi-catch is supported, only JDK7 feature not supported by Android is try-with-resources – [Matthieu Harlé](https://stackoverflow.com/users/1424030/matthieu-harl%c3%a9) [Jul 23 '14 at 10:31](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment38696799_23318644)

* 12

Java 7 working with all versions of Android. One exception: try-with-resources works only since Kitkat. – [Dmitry Zaytsev](https://stackoverflow.com/users/926907/dmitry-zaytsev) [Aug 21 '14 at 10:20](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development#comment39661097_23318644)

[show **10** more comments](https://stackoverflow.com/questions/23318109/is-it-possible-to-use-java-8-for-android-development)

99

You can indeed use gradle-retrolamba gradle build dependency to use Java 8 for Android Development.

Below is the complete guide that I have recently followed to run lambda expressions for Android development. The original source of this guide is mentioned at the end.

In this guide, a method for bringing some **Java 8** features into Android Development Tools will be demonstrated, specifically aiming at *Eclipse IDE*. However, steps which will be described throughout this guide might also be adapted to Google’s new uprising development environment, Android Studio. It is based on the community edition of popular IntelliJ Idea IDE by JetBrains and it has recently been upgraded to its ‘beta’ version by Google in early July 2014, slightly before this guide was written. Eclipse will remain as the prominent development environment, at least for a while, and considering the fact that most Android projects have been developed using Eclipse, a method for bringing new Java 8 features like lambda expressions into ADT seems to be quite useful for developers.

Android Development is based on a custom Java implementation called Apache Harmony Project which was terminated back in 2011. The most commonly used Java syntax in Android Development is Java 6 (v1.6) and Java 7 (v1.7) is also partially supported on the KitKat edition (Android 4.4.+). Therefore, Java 8 features like lambda expressions cannot be used directly in the Android App Development without applying some tweaks into the development tools. Luckily, these constructs are basically some ‘syntactic sugar’ enhancements which give developers the shortcomings of things like ‘anonymous classes’ and they can be translated into Java 6 or Java 7 classes.

A recent approach for translating a Java 8 source code into lower Java versions is called **RetroLambda**. This library makes developers run Java 8 code with lambda expressions on Java 7 or even lower. Unfortunately, Java 8 features other than lambda expressions are not supported by RetroLambda for now but the concept of lambda expressions is the biggest leap on Java 8 platform and it’s a great tool for Android developers anyway.

Details about this library can be found on its GitHub page:

<https://github.com/orfjackal/retrolambda#getting-started>

Also, a Gradle plugin for RetroLambda created by another developer allows Gradle-based builds to be implemented in Java or Android Projects. However, the developer only mentions about integrating this plugin into Android Studio environment. Details can be found on its GitHub page:

<https://github.com/evant/gradle-retrolambda>

Using these infrastructures within an Eclipse-based development environment cannot be approached directly but it’s doable and will be demonstrated throughout this guide.

**Preparation**

This guide assumes that the reader has a basic understanding of Android Development and it is based on ADT version 22.6.2 because recent ADT version 23.0.2 seems to have problems like layout folder creation. Details about this issue can be found under the following link:

<http://code.google.com/p/android/issues/detail?id=72591>

Steps in this guide will be given for a *Windows 8.1, 64-bit* development machine but they can easily be adapted to other platforms. The new build system Gradle will be used for build/clean processes and its installation procedure will also be provided. Also, both JDK 8 and JDK 7 must coexist on the development machine. Steps given below must be followed to install them:

* Go to JDK 8 early access preview page [http://jdk8.java.net](http://jdk8.java.net/)
* Download JDK 8u20 and install it. JRE 8 installation is not necessary and it can be skipped
* Go to JDK 7 latest stable release page <http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>
* Download JDK 7u65 and install it. JRE 7 installation is again not necessary and it can be skipped
* Add JDK 8 home folder and JDK 8 bin folder to your %PATH% variable
* Create a new environment variable JAVA\_HOME with the value of the path of JDK 8 home folder
* Create a new environment variable JAVA8\_HOME again with the value of the path of JDK 8 home folder
* Create a new environment variable JAVA7\_HOME with the value of the path of JDK 7 home folder
* Open a terminal window and run java -version command and verify that Java 8 is up and running
* Run javac -version command in the same window and verify that JDK 8 Java compiler is also up and running

Now, ADT-22.6.2 must be downloaded from the following link:

<http://dl.google.com/android/adt/22.6.2/adt-bundle-windows-x86_64-20140321.zip>

* Download ADT and unzip its contents into a folder, e.g. D:\adt
* Define a new environment variable called ANDROID\_HOME with the value of the path of your ADT installation folder, e.g. D:\adt\sdk
* Add your Andoid SDK Platform Tools and Andoid SDK Tools folders, e.g. D:\adt\sdk\toolsand D:\adt\sdk\platform-tools, to your %PATH% variable
* Create a shortcut to Eclipse IDE if you like. It is located under your ADT installation folder, e.g. D:\adt\eclipse
* Run Eclipse IDE and create a workspace, e.g. D:\adt\workspace
* Click on the Android SDK Manager button which is located on the toolbar
* Select Android SDK Build tools Rev. 19.1 and Android Support Library only. Un-select everything else and install these two packages.

If everything goes well, ADT will be up and running.

**The installation of the following tools is also highly recommended:**

* Eclipse Kepler Java 8 Support: It makes Eclipse recognize new Java 8 syntax extensions and makes you get rid of annoying red dots in your Java code editor. It might be installed through Help -> Install New Software in Eclipse. Enter <http://download.eclipse.org/eclipse/updates/4.3-P-builds/> into the Work with field and continue to install it.
* Nodeclipse/Enide Gradle: It is mainly used to highlight Groovy language keywords. Groovy is used as the DSL for Gradle build scripts. This plugin can be installed through Eclipse Marketplace. However, Eclipse within ADT-22.6.2 does not come along with Eclipse Marketplace Client. Therefore, you will first need to install Eclipse Marketplace Client by means of Install New Software tool in Eclipse. Enter http//:download.eclipse.org/mpc/kepler/ into the Work with field and continue to install it. After installing Eclipse Marketplace Client, you may search for Nodeclipse/Enide Gradle in the Eclipse Marketplace Client and install it.
* Genymotion Virtual Device: It is a great replacement of the default Android Virtual Devicewhich comes along with ADT. AVD is annoyingly cumbersome and it keeps on crashing for no reason. Genymotion makes you prepare Android VD's using CyanogenMod images which are executed by *Oracle VirtualBox*. Its single user license is for free and it can be downloaded from [http://www.genymotion.com](http://www.genymotion.com/). Only a login is required and it can also be integrated into Eclipse. Details can be found under:

<https://cloud.genymotion.com/page/doc/#collapse8>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

It might be considered as a fully-fledge Android device running on a x86 or x64 based personal computer. In order to use Google services like Google PlayStore on this virtual device, a gappsimage for the Android version that it uses must be flashed onto the device. A proper gapps image for the device might be downloaded from CyanogenMod website:

<http://wiki.cyanogenmod.org/w/Google_Apps>

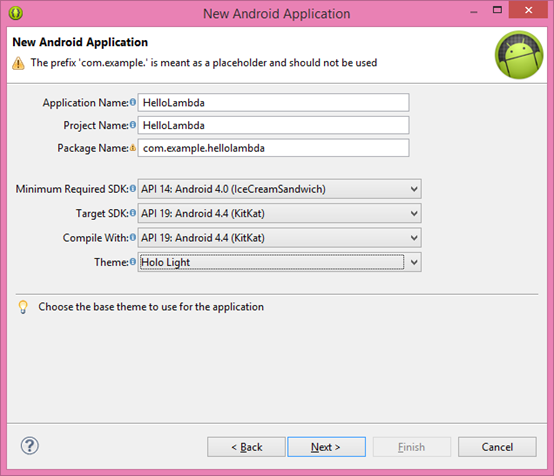
Gradle installation is optional since it is also provided by Android SDK itself but its separate installation is highly recommended. Installation of it might be conducted by following these steps:

* Go to Gradle web site: <http://www.gradle.org/>
* Click Downloads
* Under Previous Releases choose version 1.10 and download either gradle-1.10-all.zip or gradle-1.10-bin.zip
* Unzip its contents into a folder, e.g. D:\adt\gradle
* Define a new environment variable called GRADLE\_HOME with the value of the path of your Gradle installation folder, e.g. D:\adt\gradle
* Add your Gradle binaries folder, e.g. D:\adt\gradle\bin, to your %PATH% variable
* Open a terminal window and run gradle -v command and verify that it`s up and running If you have come up to this point successfully then it means that you are ready to create your first Android App using Java 8 features.

**Demo App**

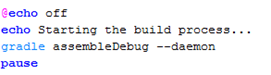
A simple app will be created to demonstrate the usage of the tools which were described in the previous section. You may simply follow the steps given below to get an insight on using lambda expressions in Android Developer Tools:

* Run Eclipse IDE and create a new Android App by selecting File -> New -> Other -> Android -> Android Application Project
* Fill in the form that comes up as shown below:

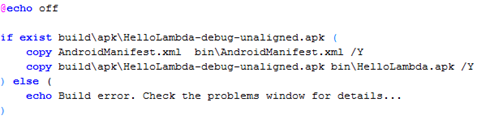


* Simply click the Next button on the following forms and click the Finish button on the last one. Wait till ADT finishes loading up the project
* Right-click on the project and select New -> Folder and name it builders
* Right-click on the gen (Generated Java Files) folder and delete it. Gradle will generate the same files for us soon and we will add them into the projects build path. Thegen` folder created by the default Ant builder is no longer needed and the artifacts under that folder will be obsolete
* Create following batch files under the builders folder:
  + gradle\_build.cmd
  + gradle\_post\_build.cmd
  + gradle\_clean.cmd
* Fill in these batch files as follows:

**gradle\_build.cmd:**



**gradle\_post\_build.cmd:**

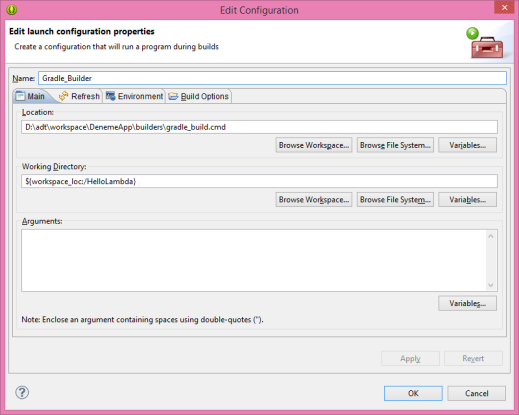


**gradle\_clean.cmd:**

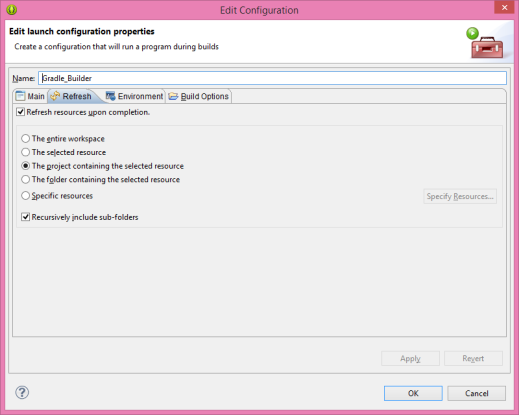
enter image description here

* Un-select Project -> Build Automatically menu option
* Right-click on the project and select Properties -> Builders and un-select all default builders provided by ADT
* Click the New button in the same window and select Program and click OK
* New builder configuration window will appear. Fill in its tabs as follows:

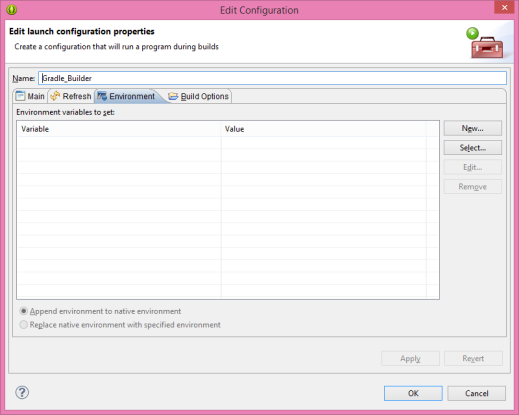
**Main Tab of the new Builder Configuration**



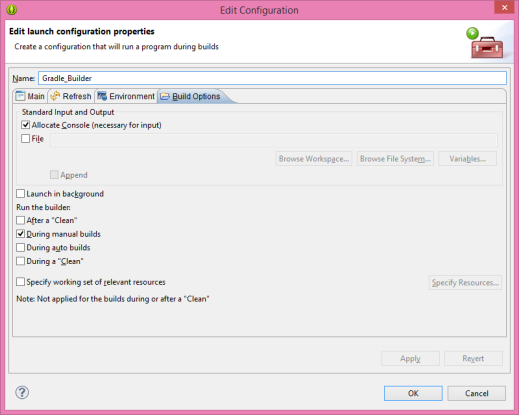
**Refresh Tab of the new Builder Configuration**



**Environment Tab of the new Builder Configuration**

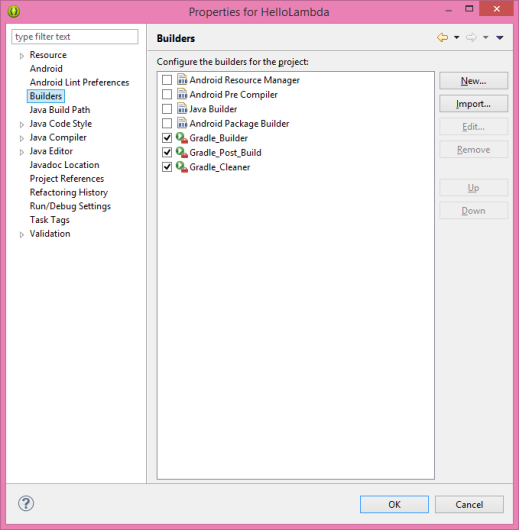


**Build Options Tab of the new Builder Configuration**



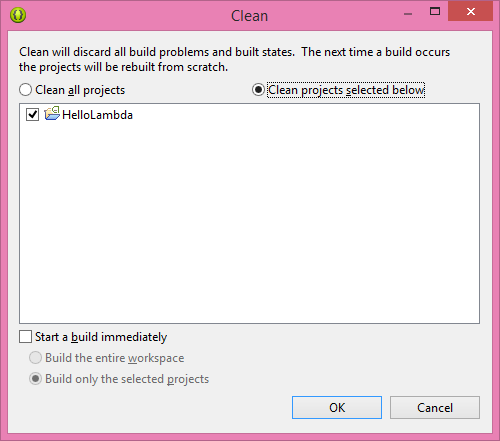
* Create the second builder called Gradle\_Post\_Build that uses gradle\_post\_build.cmd as its program. All other settings of this builder must exactly be the same with the previously created builder. This builder will be responsible for copying the artifacts created by the build process into the bin folder.
* Create the third builder called Gradle\_Cleaner that uses gradle\_clean.cmd as its program. Only Run the builder setting in the final tab must be set as During a Clean. All other settings of this builder must exactly be the same with the first builder. This builder will be responsible for cleaning the artifacts created by the build process as the name suggests.

**New Builders of the HelloLambda Project**

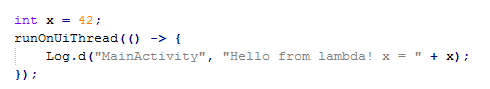


* Right-click on the project and select Export
* Select Android -> Generate Gradle Build Files and click Next
* Select the project in the next window and click Finish
* Go to your project's root folder and delete the files gradlew and gradlew.bat. Also delete gradle folder
* Switch back to Eclipse and select Project -> Clean menu option. Fill in the form that shows up as follows:

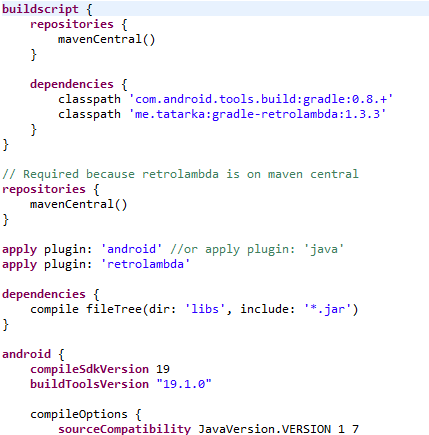
**Clean Project Window**



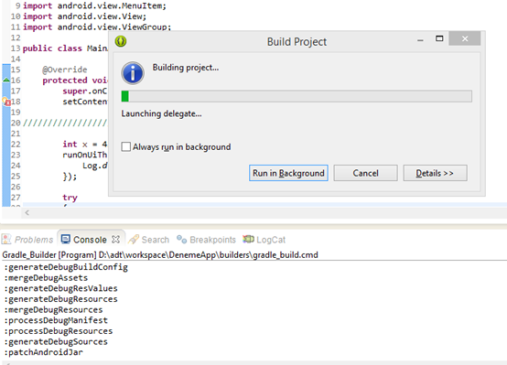
* Click OK and wait till the cleaning process completes
* Add the Java code snippet given below right after the call to setContentView function in your MainActivity class:



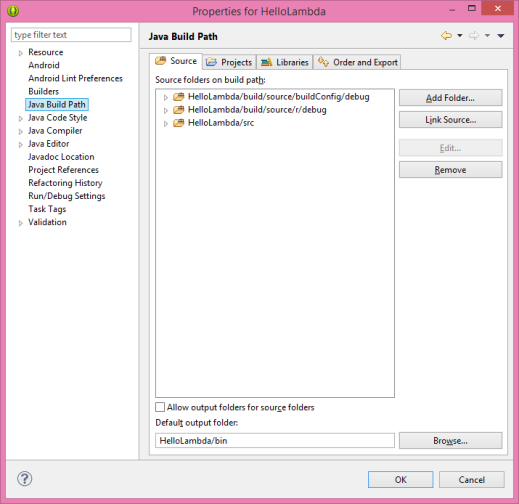
* Change the beginning of the build.gradle file till the sourceCompatibility section as follows:



* Right-click on the project and select Properties -> Java Compiler option and set all compliance levels to Java 8. This will make Eclipse recognize new Java 8 constructs like lambda expressions.
* Click No in the notification window
* Right-click on the project and select Build project. Eclipse will start building the project.

**Build Process** 

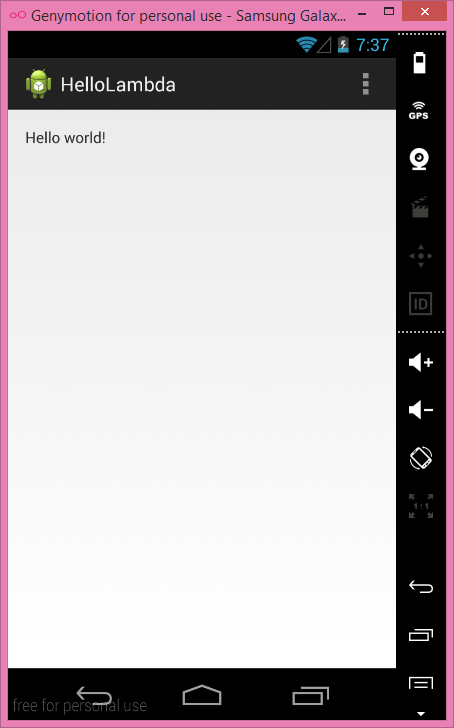
* Right-click on the project and go to Properties -> Java Build Path. Add the following folders to the build path (also shown in below image):
  + build\source\buildConfig\debug
  + build\source\r\debug



Eclipse will now be able to recognize R.java and buildConfig.java files and it will not display any red dots which denote errors related to the resource files of the project.

* Create a new Run Configuration for your Android target platform by right-clicking on the project and then selecting Run As -> Run Configurations. For instance, this demo application looks like shown below on the Genymotion VD:

**HelloLambda Application**



You may observe in the LogCat window that the code snippet with a simple lambda expression works properly