## Using Kotlin for Android Development

Kotlin is a great fit for developing Android applications, bringing all of the advantages of a modern language to the Android platform without introducing any new restrictions:

**Compatibility:** Kotlin is fully compatible with JDK 6, ensuring that Kotlin applications can run on older Android devices with no issues. The Kotlin tooling is fully supported in Android Studio and compatible with the Android build system.

**Performance:** A Kotlin application runs as fast as an equivalent Java one, thanks to very similar bytecode structure. With Kotlin's support for inline functions, code using lambdas often runs even faster than the same code written in Java.

**Interoperability:** Kotlin is 100% interoperable with Java, allowing to use all existing Android libraries in a Kotlin application. This includes annotation processing, so databinding and Dagger work too.

**Footprint:** Kotlin has a very compact runtime library, which can be further reduced through the use of ProGuard. In a real application, the Kotlin runtime adds only a few hundred methods and less than 100K to the size of the .apk file.

**Compilation Time:** Kotlin supports efficient incremental compilation, so while there's some additional overhead for clean builds, incremental builds are usually as fast or faster than with Java.

**Learning Curve:** For a Java developer, getting started with Kotlin is very easy. The automated Java to Kotlin converter included in the Kotlin plugin helps with the first steps. Kotlin Koans offer a guide through the key features of the language with a series of interactive exercises.

## Advantages and Disadvantages

Following are some of the advantages of using Kotlin for your application development.

**Easy Language** - Kotlin is a functional language and very easy to learn. The syntax is pretty much similar to Java, hence it is very easy to remember. Kotlin is more expressive, which makes your code more readable and understandable.

**Concise** - Kotlin is based on JVM and it is a functional language. Thus, it reduce lots of boiler plate code used in other programming languages.

**Runtime and Performance** - Better performance and small runtime.

**Interoperability** - Kotlin is mature enough to build an interoperable application in a less complex manner.

**Brand New** - Kotlin is a brand new language that gives developers a fresh start. It is not a replacement of Java, though it is developed over JVM. It is accepted as the first official language of android development. Kotlin can be defined as - **Kotlin** = **JAVA** + extra updated new features.

## Following are some of the disadvantages of Kotlin.

**Namespace declaration** - Kotlin allows developers to declare the functions at the top level. However, whenever the same function is declared in many places of your application, then it is hard to understand which function is being called.

**No Static Declaration** - Kotlin does not have usual static handling modifier like Java, which can cause some problem to the conventional Java developer.

## Kotlin - Environment Setup

However, if you still want to use Kotlin offline in your local system, then you need to execute the following steps to configure your local workspace.

**Step 1** - Java 8 installation.

Kotlin runs on JVM, hence. it is really necessary to use JDK 8 for your local Kotlin development. Please refer to the official website of oracle to download and install JDK 8 or an above version. You might have to set the environment variable for JAVA such that it can work properly. To verify your installation in Windows operating system, hit "java –version" in the command prompt and as an output it will show you the java version installed in your system.

**Step 2** - IDE installation.

There are a number of IDE available over the internet. You can use any of your choice. You can find the download link of different IDE in the following table.

**IDE Name  Installation Link**

1) NetBeans

   https://netbeans.org/downloads/

2) Eclipse

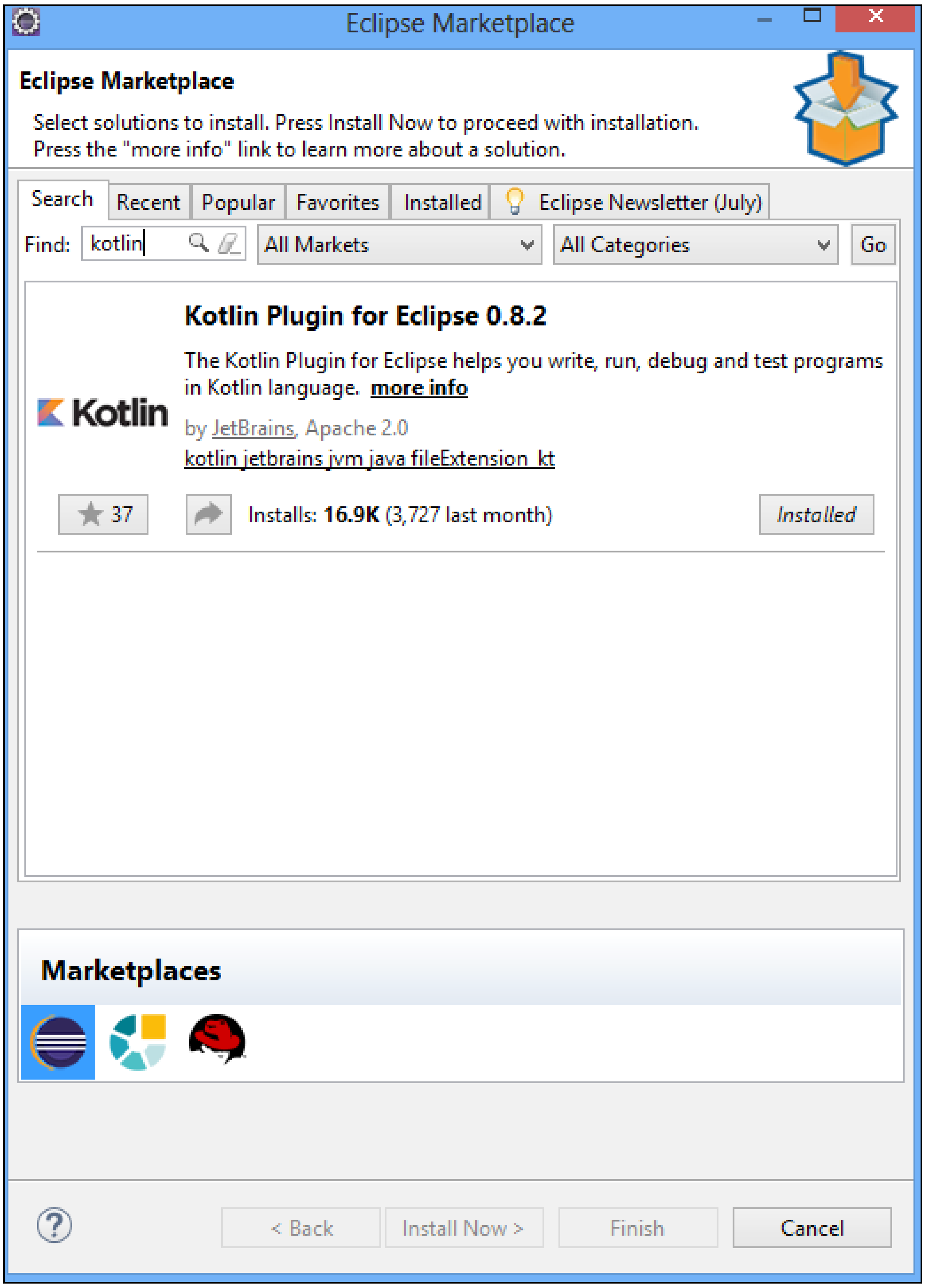
   https://www.eclipse.org/downloads/

3) Intellij

   https://www.jetbrains.com/idea/download/#section = windows

**Step 3** - Configuring Eclipse.

Open Eclipse and go to "Eclipse Market Place". You will find the following screen.



Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

Find: kotlin

Kotlin Plugin for Eclipse 0.8.2

The Kotlin Plugin for Eclipse helps you write, run, debug and test programs in Kotlin language. more info

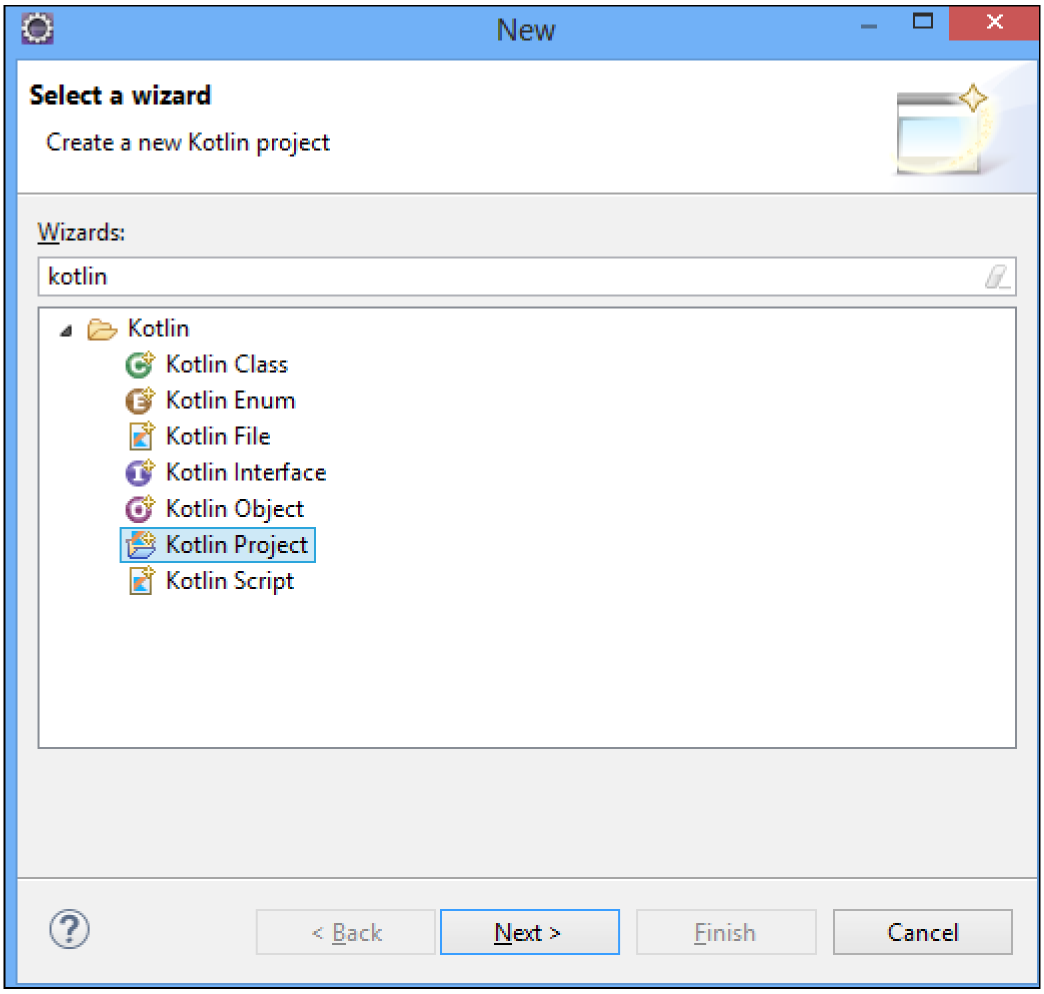
by JetBrains, Apache 2.0
kotlin jetbrains jvm java fileExtension_kt

Installs: 16.9K (3,727 last month)

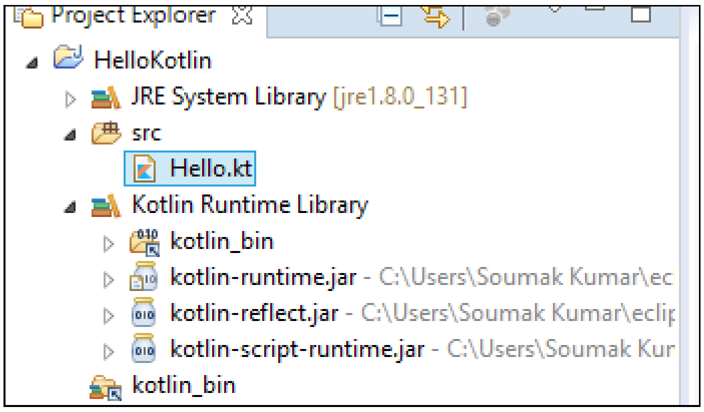
Search for Kotlin in the search box and install the same in your local system. It might take some time depending on the internet speed. You may have to restart your Eclipse, once it is successfully installed.

**Step 4 - Kotlin Project.**

Once Eclipse is successfully restarted and Kotlin is installed, you will be able to create a Kotlin project on the fly. Go to **File → New → Others** and select "**Kotlin project**" from the list.



Once the project setup is done, you can create a Kotlin file under "SRC" folder. Left-click on the "Src" folder and hit "new". You will get an option for Kotlin file, otherwise you may have to search from the "others". Once the new file is created, your project directory will be look like the following.


Project Explorer
HelloKotlin
JRE System Library [jre1.8.0_131]
src
Hello.kt
Kotlin Runtime Library
kotlin_bin
kotlin-runtime.jar - C:\Users\Soumak Kumar\ec
kotlin-reflect.jar - C:\Users\Soumak Kumar\eclip
kotlin-script-runtime.jar - C:\Users\Soumak Kur
kotlin_bin

Your development environment is ready now. Go ahead and add the following piece of code in the "Hello.kt" file.

```
 fun main(args: Array<String>) {
  println("Hello, World!")
}
```

## Working with the Command Line Compiler

## For Linux – UBUNTU

1) **Snap** package

If you're on Ubuntu 16.04 or later, you can install the compiler from the command line:

```
$ sudo snap install --classic kotlin
```

2) Create a simple application in Kotlin that displays **Hello, World!.**

Using our favorite editor, we create a new file called hello.kt with the following:

```
fun main(args: Array<String>) {
    println("Hello, World!")
}
```

3) Compile the application using the Kotlin compiler

```
$ kotlinc hello.kt -include-runtime -d hello.jar
```

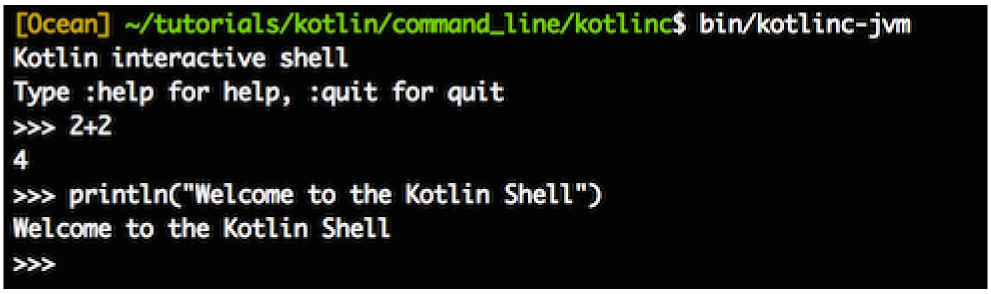
4) Run the application.

```
$ java -jar hello.jar
```

## Running the REPL

We can run the compiler without parameters to have an interactive shell.

We can type any valid Kotlin code and see the results.



```
[Ocean] ~/tutorials/kotlin/command_line/kotlinc$ bin/kotlinc-jvm
Kotlin interactive shell
Type :help for help, :quit for quit
>>> 2+2
4
>>> println("Welcome to the Kotlin Shell")
Welcome to the Kotlin Shell
>>>
```

## Kotlin Architecture:

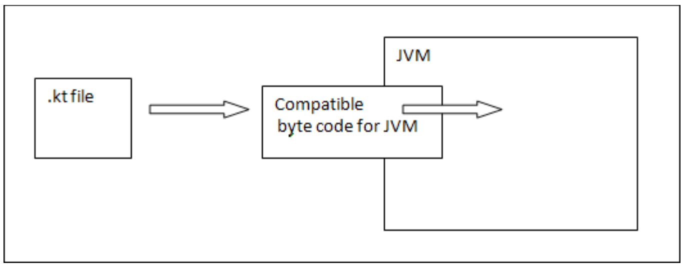
Kotlin is a programming language and has its ***own architecture*** to allocate memory and *produce a quality output* to the end user.

Following are the different scenarios where Kotlin compiler will work differently, whenever it is targeting different other kind of languages such as Java and JavaScript.

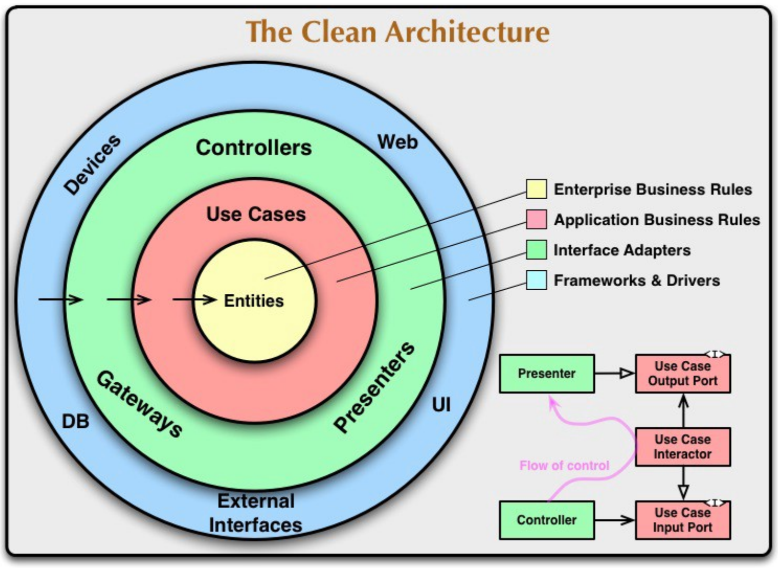
Kotlin compiler creates a byte code and that byte code can run on the JVM, which is exactly equal to the byte code generated by the Java .class file.

➔ Whenever two byte coded file runs on the JVM, they can communicate with each other and this is how an **interoperable feature** is established in Kotlin for Java.

➔ Whenever Kotlin targets JavaScript, the Kotlin compiler converts the .kt file into **ES5.1** and generates a compatible code for JavaScript.



Kotlin compiler is capable of creating platform basis compatible codes via LLVM.



https://kotlinlang.org/docs/reference/basic-syntax.html

https://codelabs.developers.google.com/codelabs/build-your-first-android-app-kotlin/index.html#0