Kotlin is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of Kotlin's standard library depends on the Java Class Library, but type inference allows its syntax to be more concise

/In this chapter, we will use IntelliJ , which is free to download from https://www.jetbrains.com/idea/download/.

**Good To Know:** In Kotlin, code statements do not have to end with a semicolon (;) (which is often required for other programming languages, such as [Java](https://www.w3schools.com/java/default.asp), [C++](https://www.w3schools.com/cpp/default.asp), [C#](https://www.w3schools.com/cs/default.asp), etc.).

The fun keyword is used to declare a function. A function is a block of code designed to perform a particular task. In the example above, it declares the main() function.

The main() function is something you will see in every Kotlin program. This function is used to execute code. Any code inside the main() function's curly brackets {} will be executed.

## What is the definition of Extension Function?

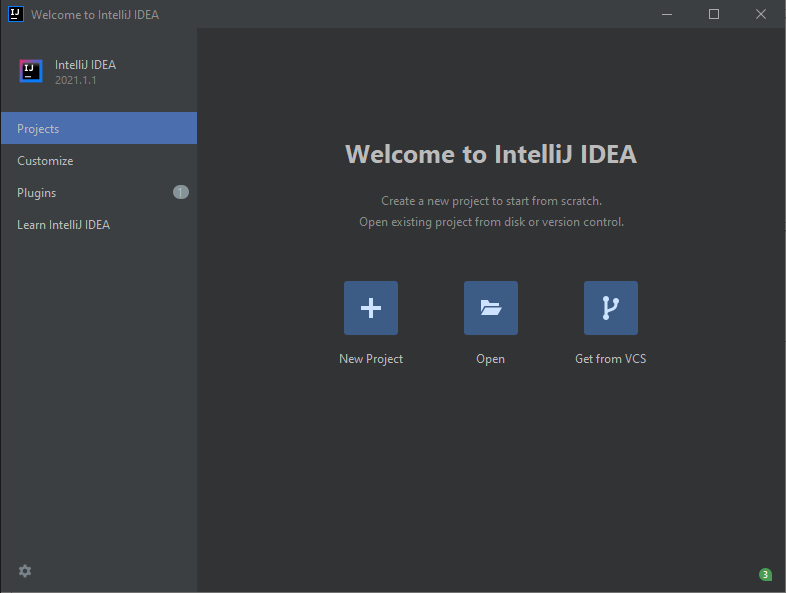
Fundamentally, an extension function is a member function of a class, which is defined outside the class. For instance, if you need to use a method to the String class that returns a new string with first and last character removed, you can write an extension method for it. In fact, this method is not already available in String class.

## KEY DIFFERENCES between java and koltin:

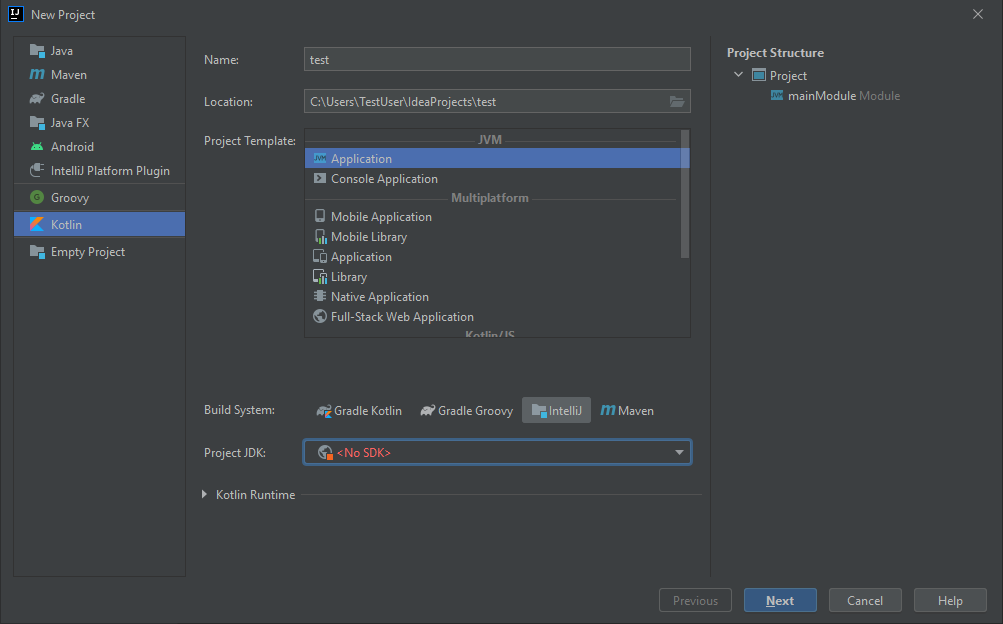
* Kotlin combines features of both object-oriented and functional programming, whereas Java is limited to object-oriented programming.
* Kotlin allows users to create an extension function while Java doesn’t offer any extension functions.
* Kotlin doesn’t offer implicit conversions, and Java supports implicit conversions.
* There are no null variables or objects in Kotlin on the other hand, Null variable or objects are part of Java language.
* Kotlin doesn’t support static members, while Java uses static members.
* In, Kotlin variables of a primitive type are objects while in Java Variables of a primitive type aren’t objects
* Kotlin supports Lambda Expression whereas Java doesn’t support Lambda expression.
* Kotlin doesn’t require any variable datatype specifications while Java requires variable datatype specifications.
* Kotlin programs don’t require semicolons in their program while Java program does need a semicolon.
* Language scripting capabilities allow you to use Kotlin directly in your Gradle build scripts while Java does not offer language scripting capabilities.

Kotlin Install

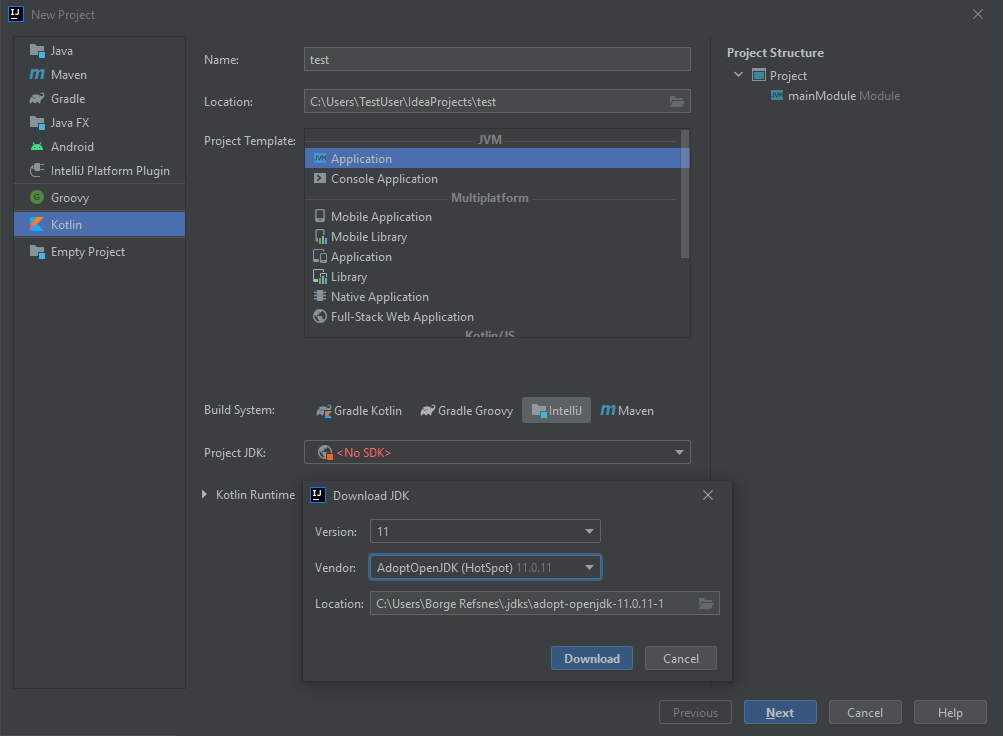
Once IntelliJ is downloaded and installed, click on the **New Project** button to get started with IntelliJ:



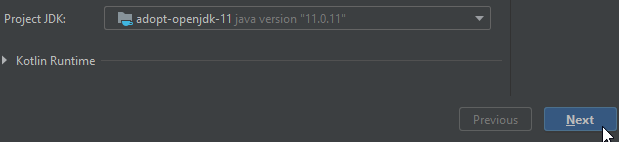
Then click on "Kotlin" in the left side menu, and enter a name for your project:



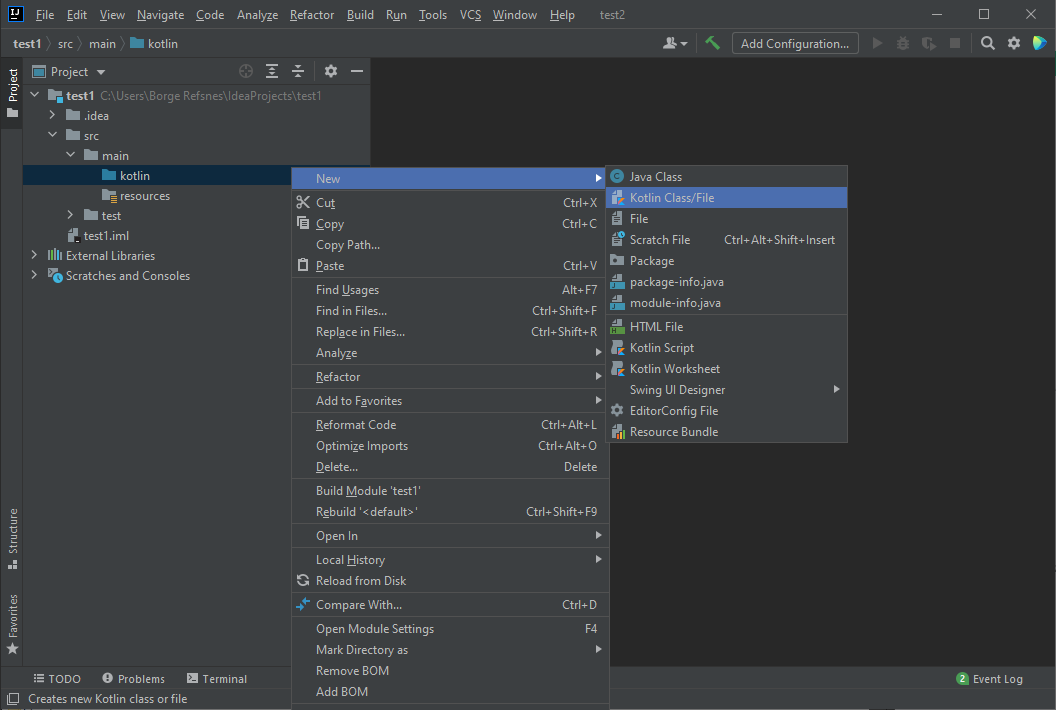
Next, we need to install something called JDK (Java Development Kit) to get our Kotlin project up and going. Click on the "Project JDK" menu, select "Download JDK" and select a version and vendor (e.g. AdoptOpenJDK 11) and click on the "Download" button:



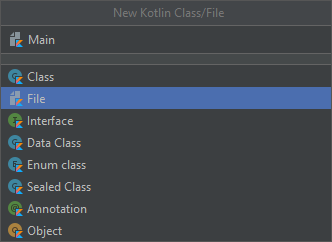
When the JDK is downloaded and installed, choose it from the select menu and then click on the "Next" button and at last "Finish":



Now we can start working with our Kotlin project. Do not worry about all of the different buttons and functions in IntelliJ. For now, just open the src (source) folder, and follow the same steps as in the image below, to create a kotlin file:



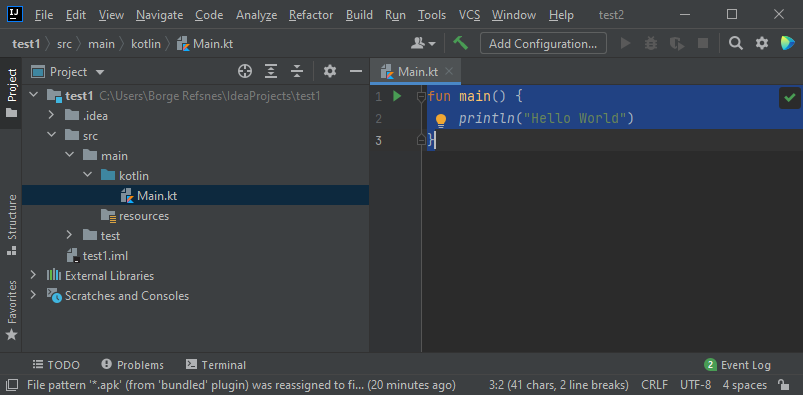
Select the "File" option and add a name to your Kotlin file, for example "Main":



You have now created your first Kotlin file (Main.kt). Let's add some Kotlin code to it, and run the program to see how it works. Inside the Main.kt file, add the following code:

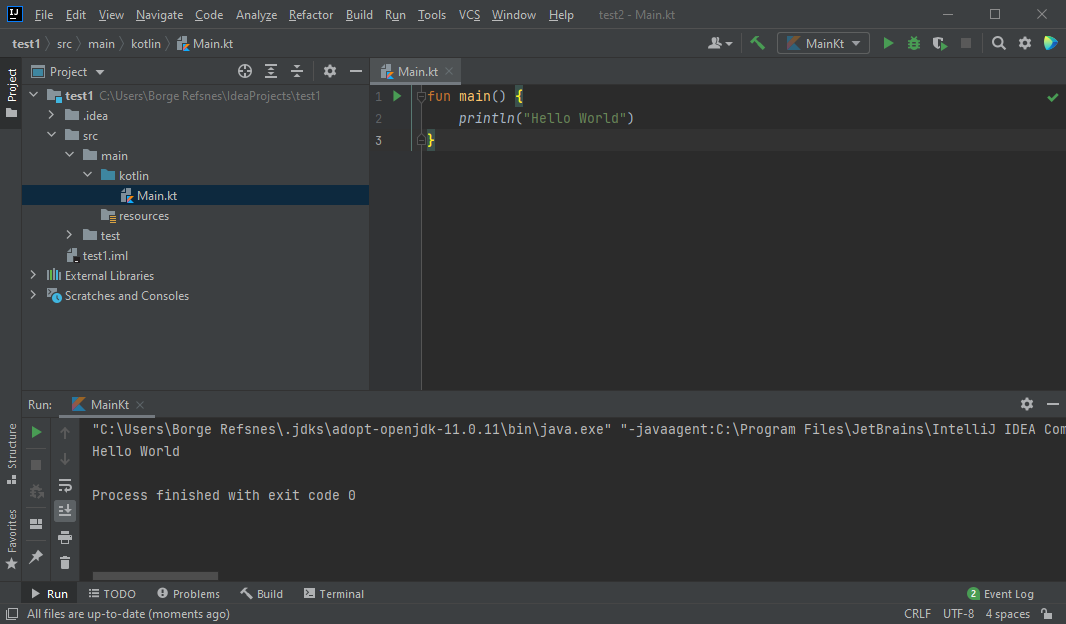
Main.kt

fun main() {  
  println("Hello World")  
}



Don't worry if you don't understand the code above - we will discuss it in detail in later chapters. For now, lets focus on how to run the code. Click on the Run button at the top navigation bar, then click "Run", and select "Mainkt".

Next, IntelliJ will build your project, and run the Kotlin file. The output will look something like this:



As you can see, the output of the code was "Hello World", meaning that you have now written and executed your first Kotlin program!