PART B EXPERIMENT 4

TITLE: Write Simple Program in Scala Using Apache Spark Framework

PROBLEM STATEMENT: -Write Simple Program in Scala Using Apache Spark Framework.

OBJECTIVE: Learn how to create and develop Scala program using Apache spark framework.

THEORY:

1. What is Scala

Scala is an acronym for "Scalable Language". It is a general-purpose programming language designed for the programmers who want to write programs in a concise, elegant, and type-safe way. Scala enables programmers to be more productive. Scala is developed as an object-oriented and functional programming language.

2. Scala is pure Object-Oriented programming language

Scala is an object-oriented programming language. Everything in Scala is an object and any operations you perform is a method call. Scala, allow you to add new operations to existing classes with the help of implicit classes.

One of the advantages of Scala is that it makes it very easy to interact with Java code. You can also write a Java code inside Scala class. The Scala supports advanced component architectures through classes and traits.

3. Scala is a compiler-based language (and not interpreted)

Scala is a compiler-based language which makes Scala execution very fast if you compare it with Python (which is an interpreted language). The compiler in Scala works in similar fashion as Java compiler. It gets the source code and generates Java byte-code that can be executed independently on any standard JVM (Java Virtual Machine).

4. Prerequisites for Learning Scala

Scala being an easy to learn language has minimal prerequisites. If you are someone with basic knowledge of C/C++, then you will be easily able to get started with Scala. Since Scala is developed on top of Java. Basic programming function in Scala is similar to Java. So, if you have some basic knowledge of Java syntax and OOPs concept, it would be helpful for you to work in Scala.

5. Choosing a development environment
Once you have installed Scala, there are various options for choosing an environment. Here are the 3 most common options:
Terminal / Shell based
Notepad / Editor based
IDE (Integrated development environment)-IntelliJ IDE

6. Scala Basics Terms
Object: An entity that has state and behaviour is known as an object. For example: table, person, car etc.
Class: A class can be defined as a blueprint or a template for creating different objects which defines its properties and behaviour.
Method: It is a behaviour of a class. A class can contain one or more than one method. For example: deposit can be considered a method of bank class.
Closure: Closure is any function that closes over the environment in which it's defined. A closure returns value depends on the value of one or more variables which is declared outside this closure.
Traits: Traits are used to define object types by specifying the signature of the supported methods. It is like interface in java.

7. Things to note about Scala

- It is case sensitive
- If you are writing a program in Scala, you should save this program using ".scala"
- Scala execution starts from main() methods
- Any identifier name cannot begin with numbers. For example, variable name "123salary" is invalid.
- You can not use Scala reserved keywords for variable declarations or constant or any identifiers.

8. Variable declaration in Scala

- In Scala, you can declare a variable using 'var' or 'val' keyword. The decision is based on whether it is a constant or a variable. If you use 'var' keyword, you define a variable as mutable variable. On the other hand, if you use 'val', you define it as immutable. Let's first declare a variable using "var" and then using "val".

8.1 Declare using var

```
var Var1 : String = "Ankit"
```

In the above Scala statement, you declare a mutable variable called "Var1" which takes a string value. You can also write the above statement without specifying the type of variable. Scala will automatically identify it. For example:

```
var Var1 = "Gupta"
```

8.2 Declare using val

```
val Var2 : String = "Ankit"
```

In the above Scala statement, we have declared an immutable variable “Var2” which takes a string “Ankit”.

9. Operations on variables

You can perform various operations on variables. There are various kinds of operators defined in Scala. For example: Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, Assignment Operators.

Lets see “+” , “==” operators on two variables ‘Var4’, “Var5”. But, before that, let us first assign values to “Var4” and “Var5”.

```
scala> var Var4 = 2
Output: Var4: Int = 2
scala> var Var5 = 3
Output: Var5: Int = 3
```

Now, let us apply some operations using operators in Scala.

Apply ‘+’ operator

```
Var4+Var5
Output:
res1: Int = 5
```

Apply “==” operator

```
Var4==Var5
Output:
res2: Boolean = false
```

10. The if-else expression in Scala

In Scala, if-else expression is used for conditional statements. You can write one or more conditions inside “if”. Let’s declare a variable called “Var3” with a value 1 and then compare “Var3” using if-else expression.

```
var Var3 =1
if (Var3 ==1){
 println("True")}else{
 println("False")}
Output: True
```

In the above snippet, the condition evaluates to True and hence True will be printed in the output.

11. Iteration in Scala

Like most languages, Scala also has a FOR-loop which is the most widely used method for iteration. It has a simple syntax too.

```
for( a <- 1 to 10){
 println( "Value of a: " + a );
 }
Output:
Value of a: 1
Value of a: 2
Value of a: 3
Value of a: 4
Value of a: 5
Value of a: 6
Value of a: 7
Value of a: 8
Value of a: 9
Value of a: 10
```

Scala also supports “while” and “do while” loops.

12. Declare a simple function in Scala and call it by passing value

You can define a function in Scala using “def” keyword. Let’s define a function called “mul2” which will take a number and multiply it by 10. You need to define the return type of function, if a function not returning any value you should use the “Unit” keyword.

In the below example, the function returns an integer value. Let’s define the function “mul2”:

```
def mul2(m: Int): Int = m * 10
Output: mul2: (m: Int)Int
```

Now let’s pass a value 2 into mul2

```
mul2(2)
Output:
res9: Int = 20
```

13. Few Data Structures in Scala

Arrays
Lists
Sets
Tuple
Maps
Option

13.1 Arrays in Scala

In Scala, an array is a collection of similar elements. It can contain duplicates. Arrays are also immutable in nature. Further, you can access elements of an array using an index:

Declaring Array in Scala

To declare any array in Scala, you can define it either using a new keyword or you can directly assign some values to an array.
Declare an array by assigning it some values

```
var name = Array("Faizan","Swati","Kavya", "Deepak", "Deepak")
Output:
name: Array[String] = Array(Faizan, Swati, Kavya, Deepak, Deepak)
```

In the above program, we have defined an array called name with 5 string values.
Declaring an array using “new” keywords

The following is the syntax for declaring an array variable using a new keyword.

```
var name:Array[String] = new Array[String](3)
or
var name = new Array[String](3)
Output:
name: Array[String] = Array(null, null, null)
```

Here you have declared an array of Strings called “name” that can hold up to three elements.

You can also assign values to “name” by using an index.

```
scala> name(0) = "jal"
scala> name(1) = "Faizy"
scala> name(2) = "Expert in deep learning"
```

Let’s print contents of “name” array.

```
scala> name
res3: Array[String] = Array(jal, Faizy, Expert in deep learning)
```

Accessing an array

You can access the element of an array by index. Lets access the first element of array “name”.

By giving index 0. Index in Scala starts from 0.

```
name(0)
Output:
res11: String = jal
```

13.2 List in Scala
Lists are one of the most versatile data structure in Scala. Lists contain items of different types in Python, but in Scala the items all have the same type. Scala lists are immutable.
Here is a quick example to define a list and then access it.
Declaring List in Scala
You can define list simply by comma separated values inside the “List” method.

```
scala> val numbers = List(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
numbers: List[Int] = List(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
```

You can also define multi dimensional list in Scala. Lets define a two dimensional list:

```
val number1 = List( List(1, 0, 0), List(0, 1, 0), List(0, 0, 1) )
number1: List[List[Int]] = List(List(1, 0, 0), List(0, 1, 0), List(0, 0, 1))
```

Accessing a list

Let's get the third element of the list "numbers" . The index should 2 because index in Scala start from 0.

```
scala> numbers(2)
res6: Int = 3
```

We have discussed two of the most used data Structures.

14. Writing & Running a program in Scala using an editor

Let us start with a "Hello World!" program. It is a good simple way to understand how to write, compile and run codes in Scala. No prizes for telling the outcome of this code!

```
object HelloWorld {
 def main(args: Array[String]) {
 println("Hello, world!")
 }
 }
```

As mentioned before, if you are familiar with Java, it will be easier for you to understand Scala. If you know Java, you can easily see that the structure of above "HelloWorld" program is very similar to Java program.

This program contains a method "main" (not returning any value) which takes an argument – a string array through command line. Next, it calls a predefined method called "Println" and passes the argument "Hello, world!".

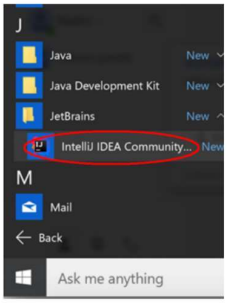
You can define the main method as static in Java but in Scala, the static method is no longer available. Scala programmer can't use static methods because they use singleton objects.

In this assignment, we will use IntelliJ IDE to create your first Scala application. If you have done previous programming in other languages, you will certainly have come across the infamous Hello World program.

Steps

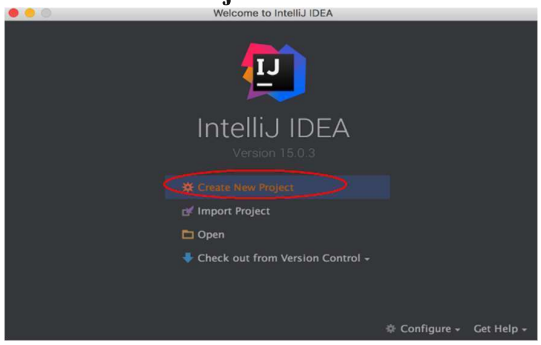
1. Start IntelliJ

On **Windows**, click on **IntelliJ** menu item from your **Windows Start** menu:

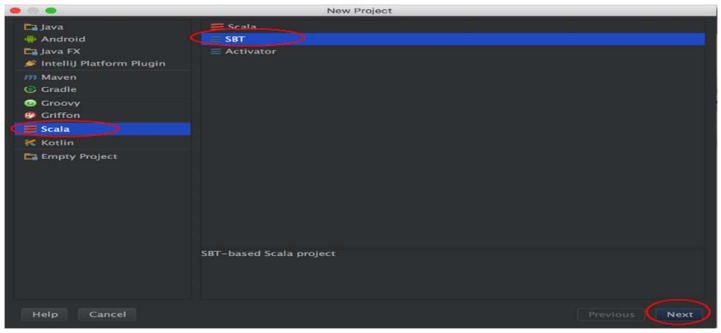


2. Create new project

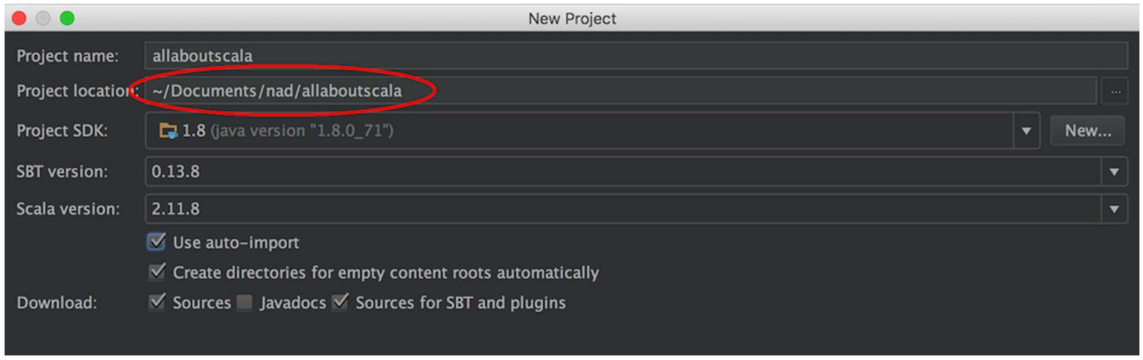
From Step 1, you will now see the IntelliJ welcome screen, so go ahead and click menu "**Create New Project**".



Next, click on **Scala** in the left panel and then click on **SBT** in the right panel. Next, click on the Next button in the bottom right corner.

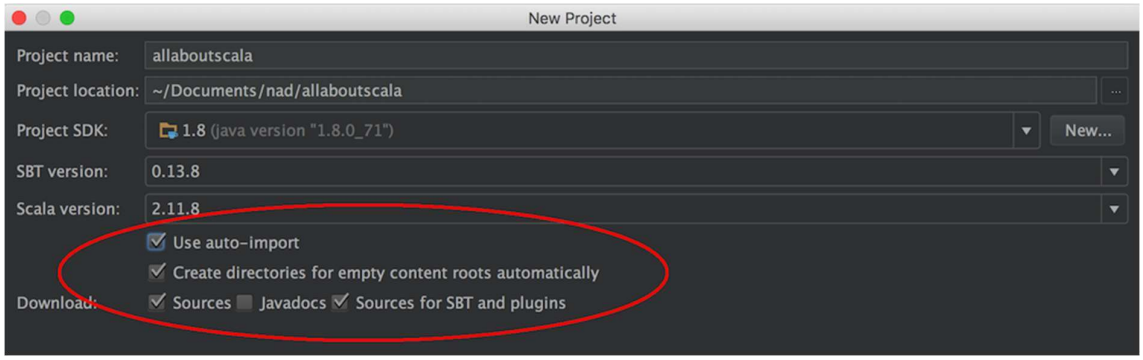


Enter your project name and location which in this tutorial will be called: **allaboutscala**



At the bottom of the screen, make sure the following are checked:

- Auto-import
- Create directories for empty content roots automatically
- Sources
- Source for SBT and plugins



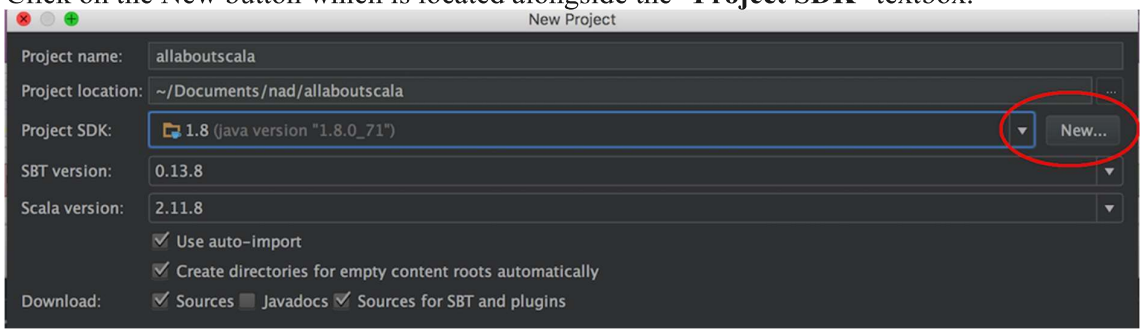
At this point, do **NOT** hit finish button yet! But don't worry we are nearly there :) Because it's your first Scala project, you have to tell IntelliJ which **JDK** to use as shown in Step 3 below.

NOTE:

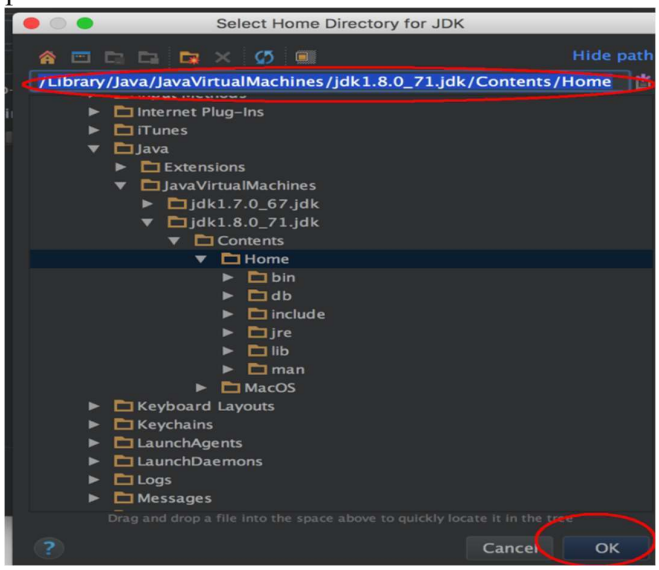
- If you do not have a JDK installed, you can follow the instructions from the tutorial on Installing JDK.

3. Setup JDK

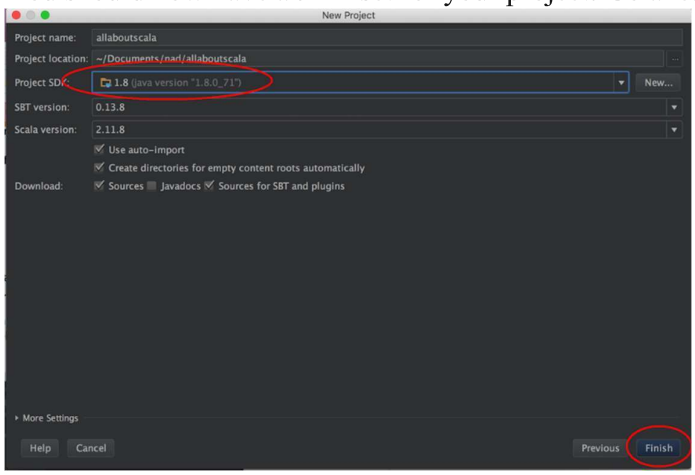
Click on the New button which is located alongside the "**Project SDK**" textbox.



A new window will open which should show you the location of the JDK. Check that the path is correct and hit the **OK** button.



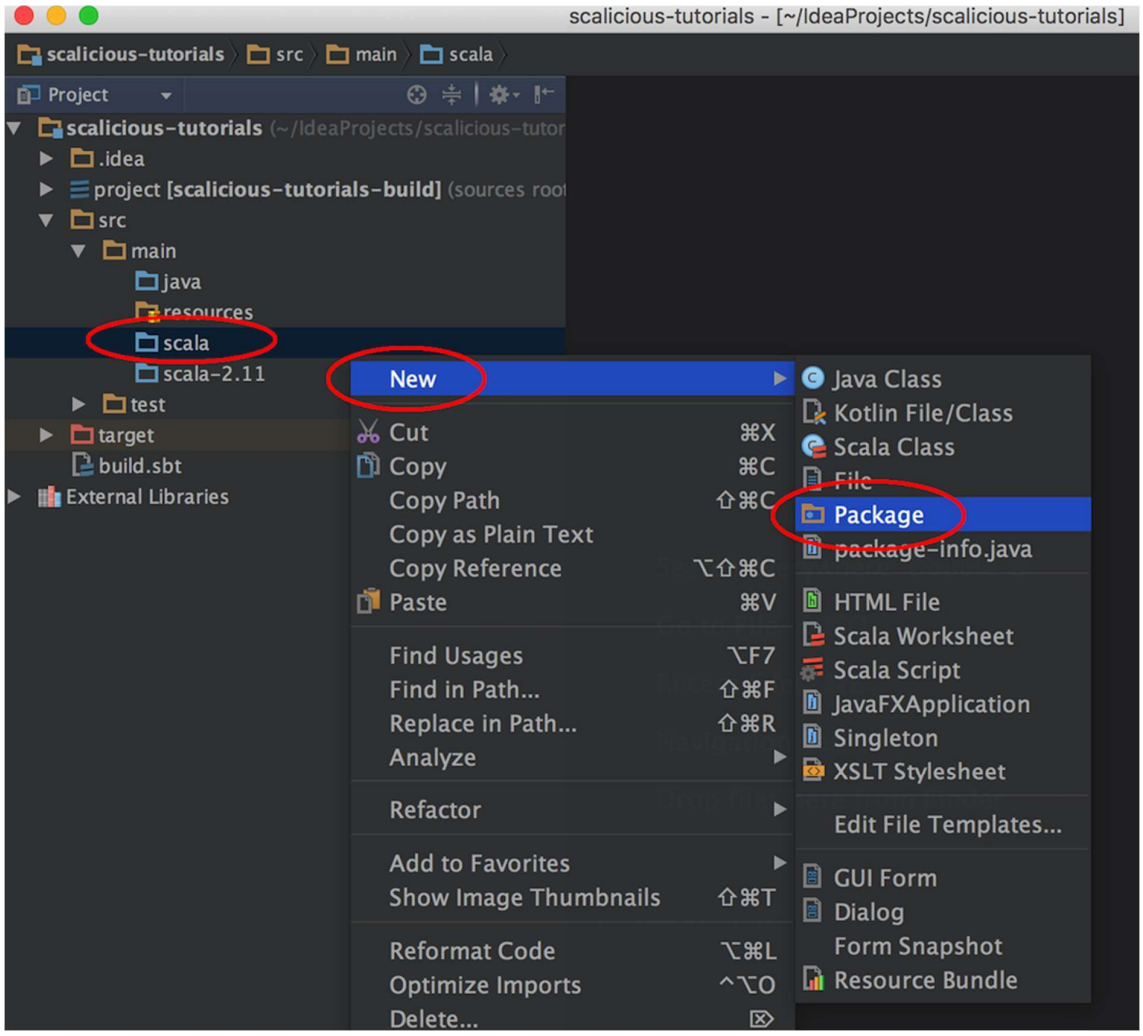
You should now have a JDK set for your project. Go ahead and hit the finish button



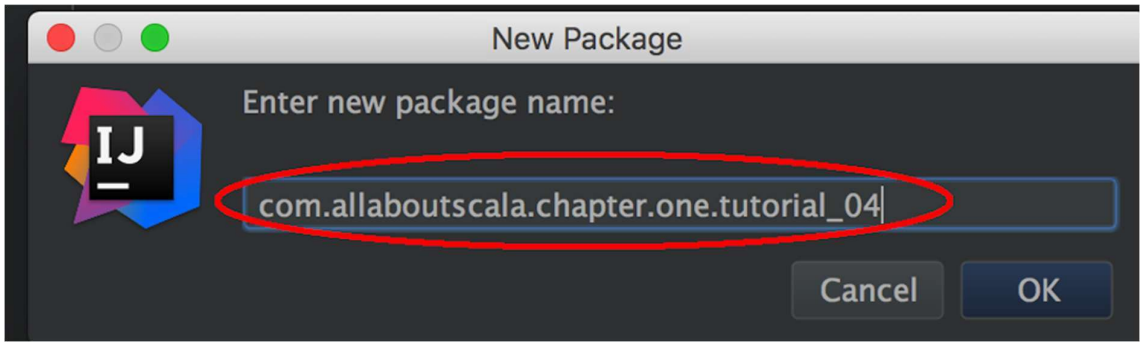
4. Create first package

You should now see a Project panel on the left hand side which shows the project structure for your Scala project named "**allaboutscala**".

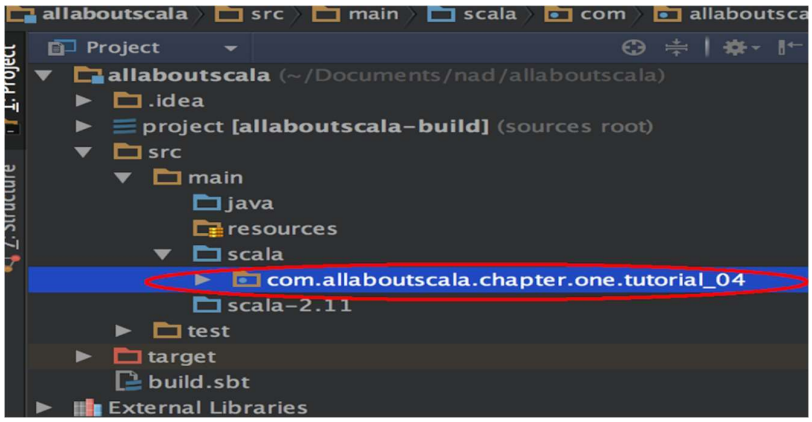
Next expand the folder "**src**", then "**main**" and then right click on the "**scala**" folder. A popup menu will open where you can go ahead and select "**New**" and then click on "**Package**"



You will see a popup textbox and type in "**com.allaboutscala.chapter.one. tutorial_04**" and click the **OK** button

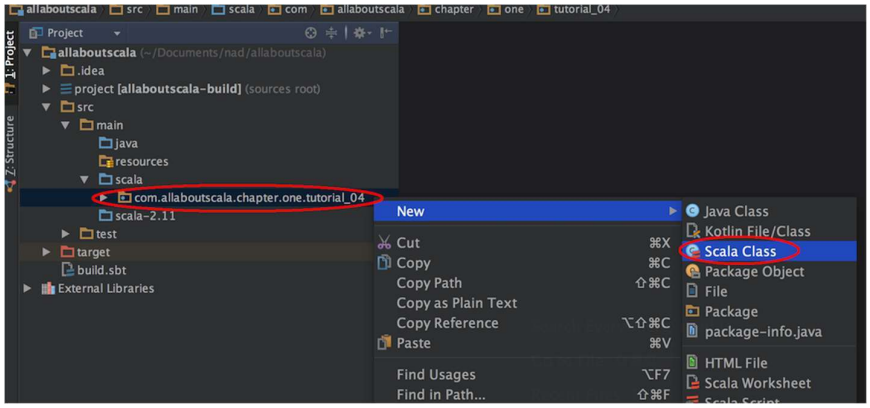


In the **Project Panel**, you should now see your package created as follows:

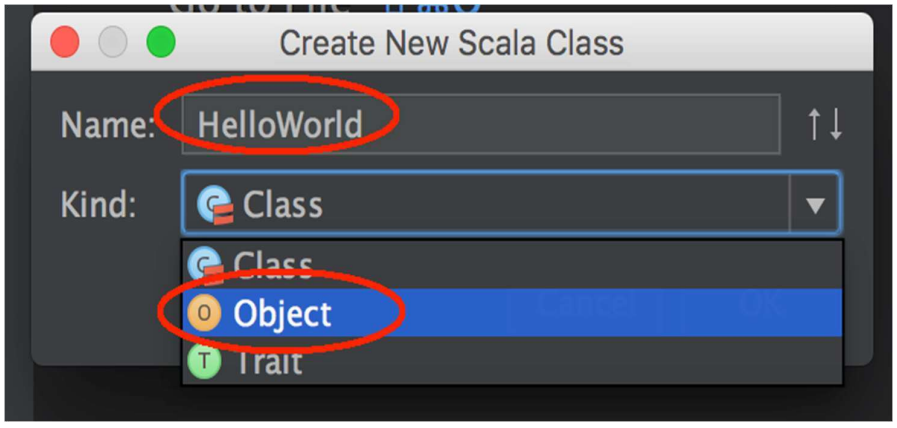


5. Create HelloWorld object

Next, right click on the package which you created in Step 4 and select "New" and then "Scala class".

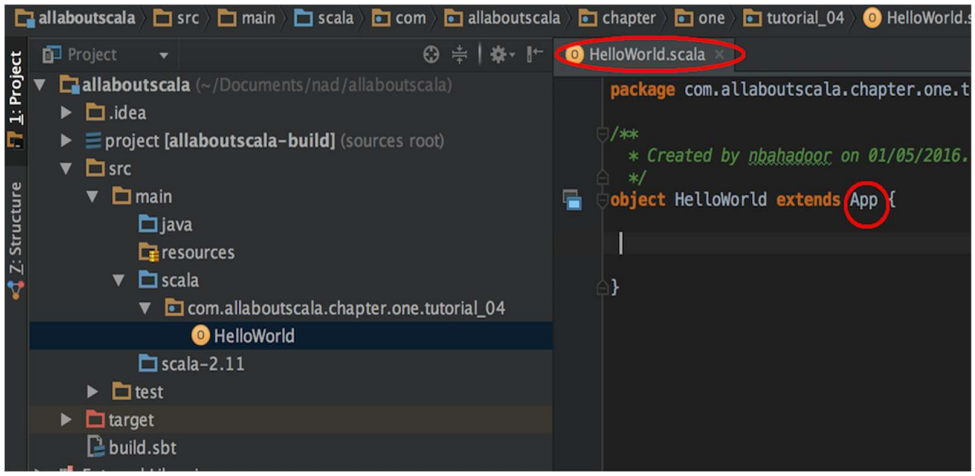


A popup window will open and type in "**HelloWorld**" in the Name textbox. Then, select "**Object**" from the **Kind drop down** and click the **OK** button.

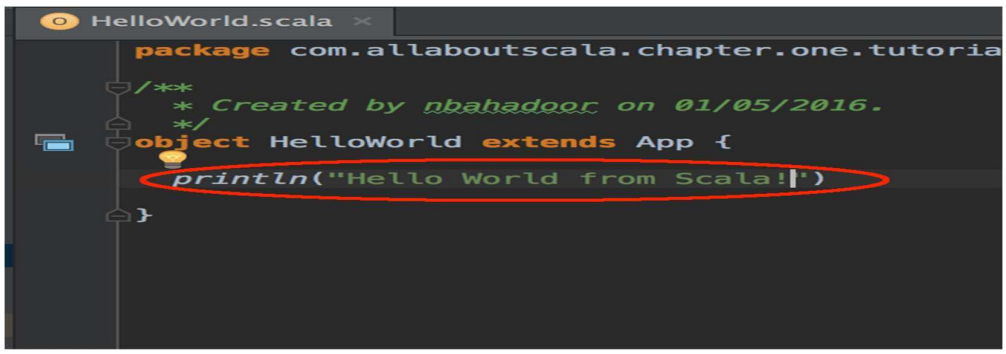


On the right panel, you should now see a "**HelloWorld.scala**" file created for you. At the very top of the file you will see the package name which we created in Step 4.

In the next line, right after **object HelloWorld**, type in **extends App as** follows:

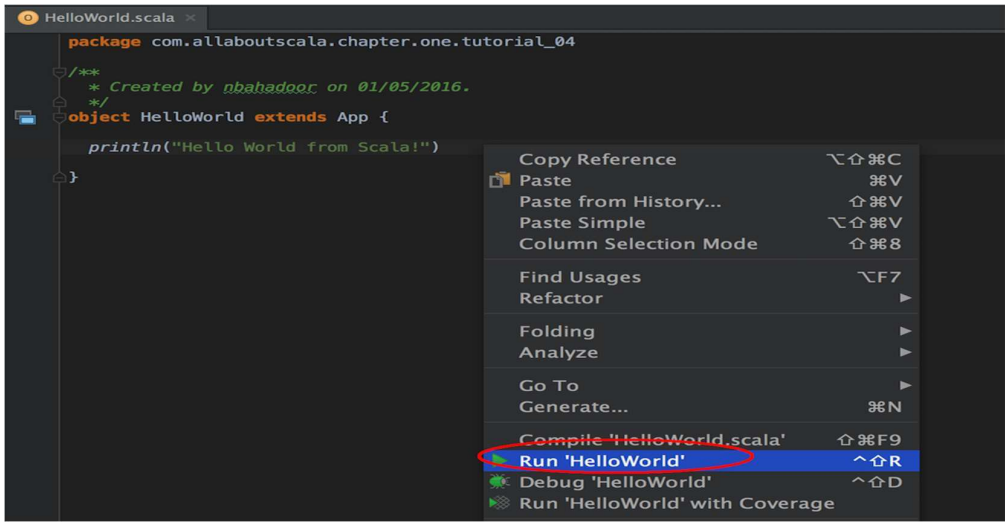


Then, inside our **HelloWorld** file, following the first curly braces, type in **println("Hello World from Scala!")**



6. Run HelloWorld

Next, it is time to run your first Scala application. Right click anywhere in the "**HelloWorld.scala**" file and select "**Run HelloWorld**"



In the **bottom panel**, you will see that IntelliJ will first compile your code using the Scala compiler. Finally, you should see a new panel open with the result of your application which simply prints "**Hello World from Scala!**"

And that's it! You've just created and run your first Scala application within IntelliJ.

This concludes our assignment on **Our first Scala Hello World application.**