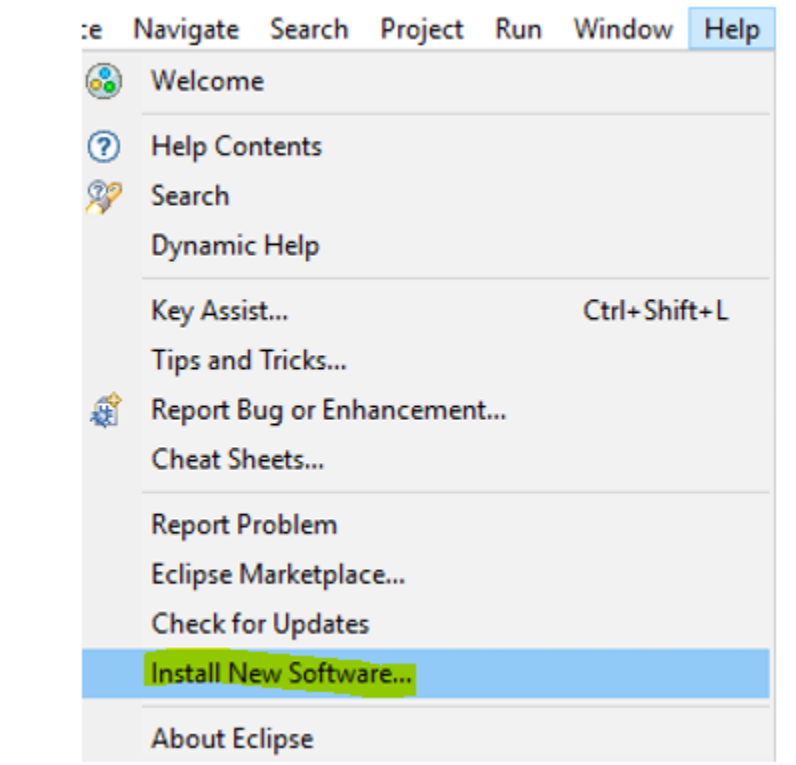
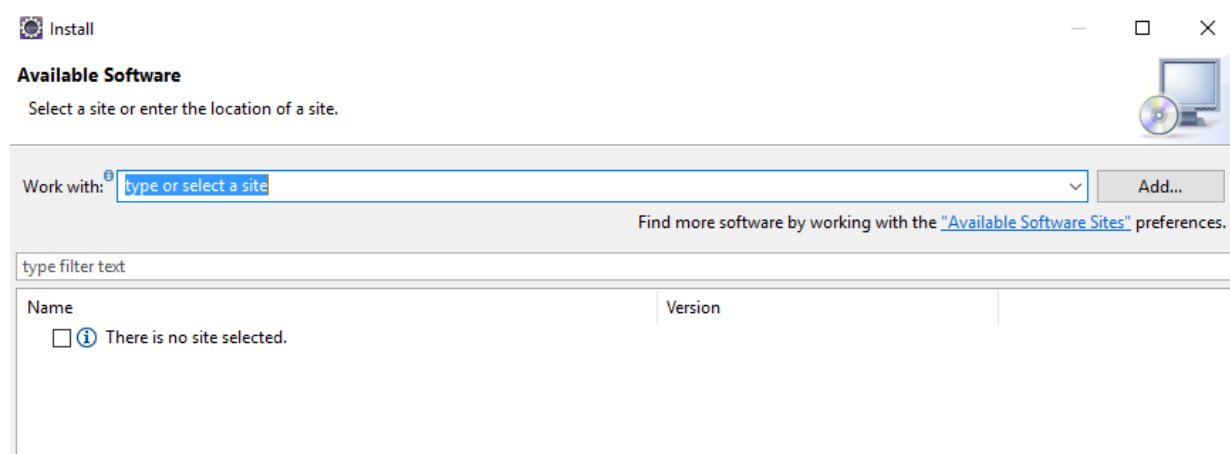


Scala IDE setup with Eclipse

Step 1 : Open the Eclipse editor and from Help menu, select “Install New Software”



Step 2 : Click on Add button to add new repository for Scala IDE



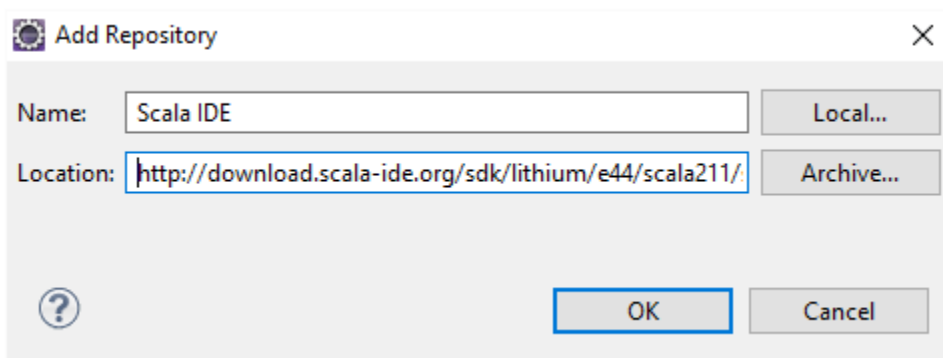
Step 3: Add Scala IDE update site

You can use the latest update site from Scala site:

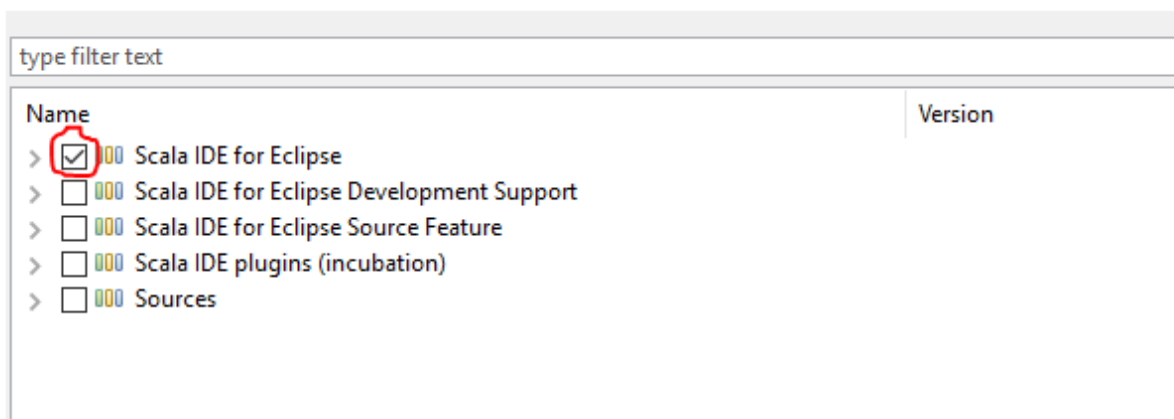
<http://scala-ide.org>

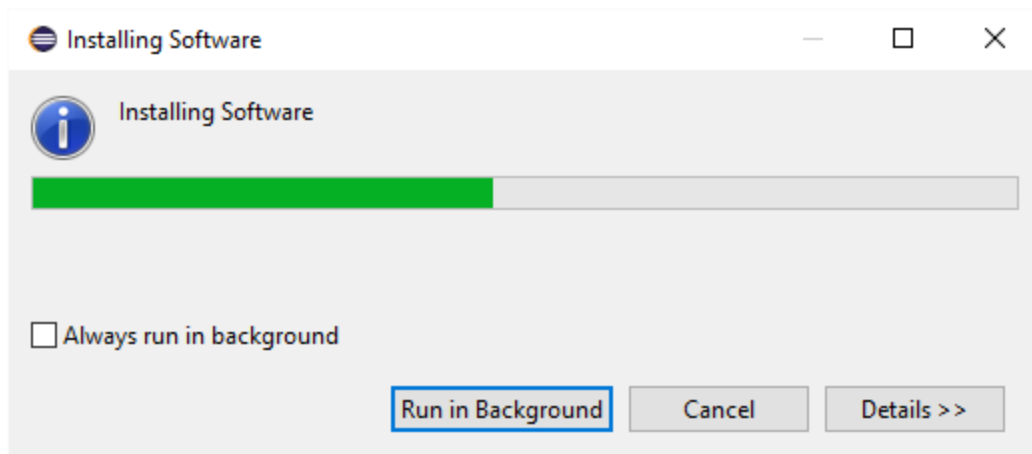
For this setup, we have used :

<http://download.scala-ide.org/sdk/lithium/e47/scala212/stable/site>

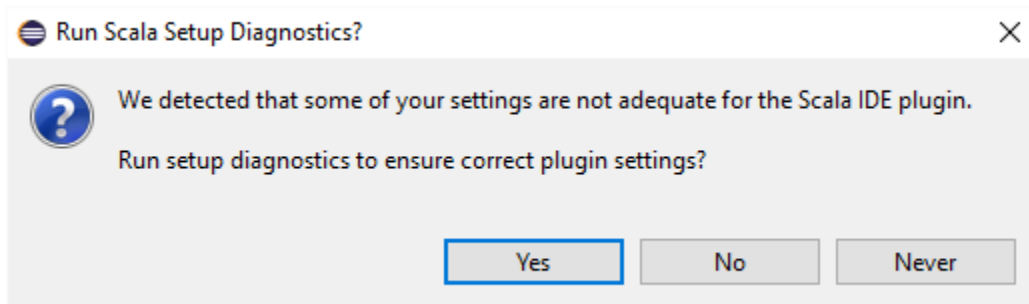
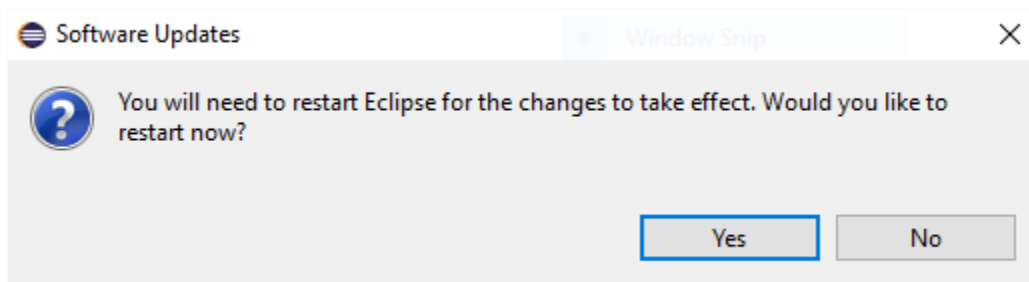


Step 4 : Select Scala IDE for Eclipse to download it





Step 5 : Restart Eclipse for the changes to take effect



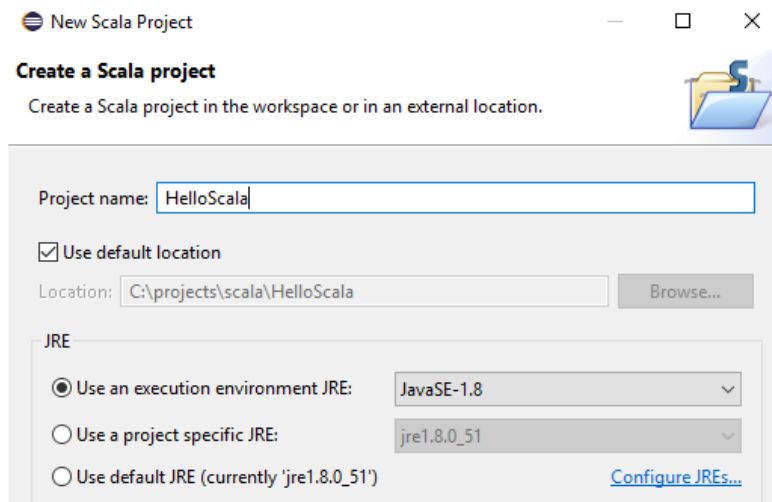
If you get this message, that means you need to allocate more memory to Eclipse. Follow the instructions on screen for the same.

You should be ready now to create your Scala projects in Eclipse.

Scala Hello World program using Eclipse IDE

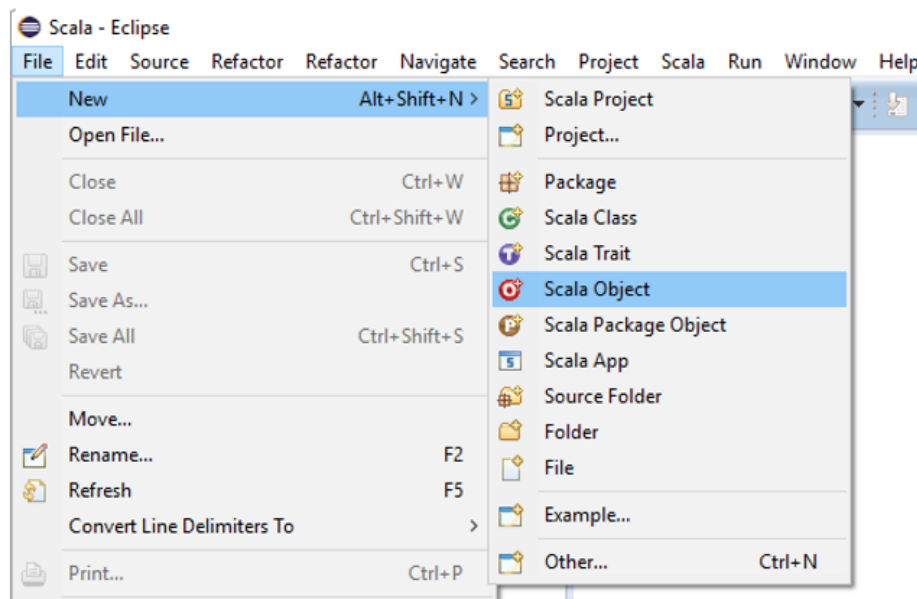
Step 1 : Create new Scala project in Eclipse

From File menu, select New -> Scala project and provide the project a name

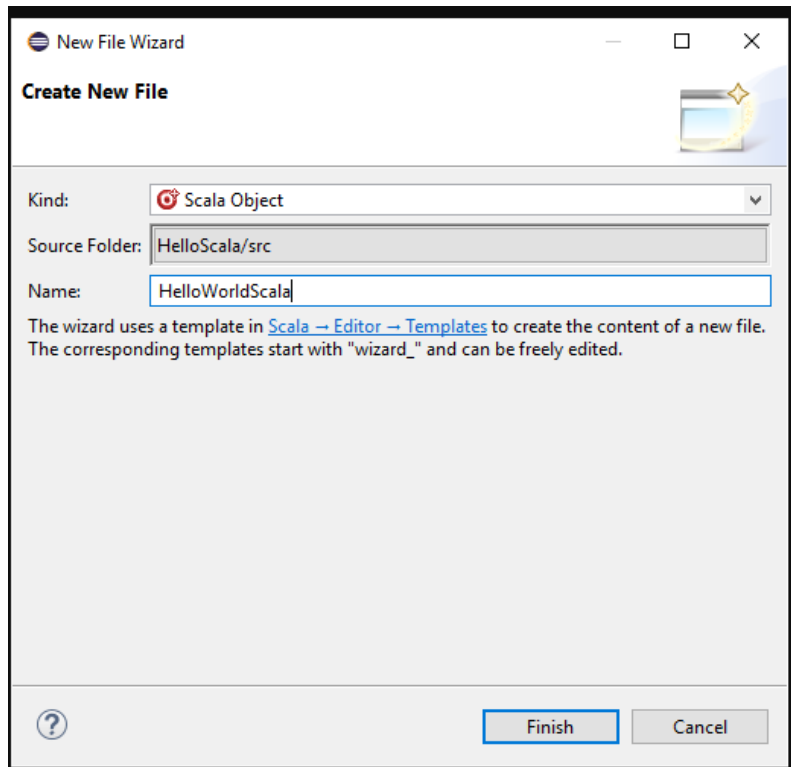


Step 2 : Create a Scala object

Create a new Scala Object using File -> New -> Scala Object :



Provide a name for your Scala application and click Finish to create the file.



Step 3 : Add code that prints Hello Scala

Add the following code in HelloWorld.scala :

```
// Scala program to print Hello World!
```

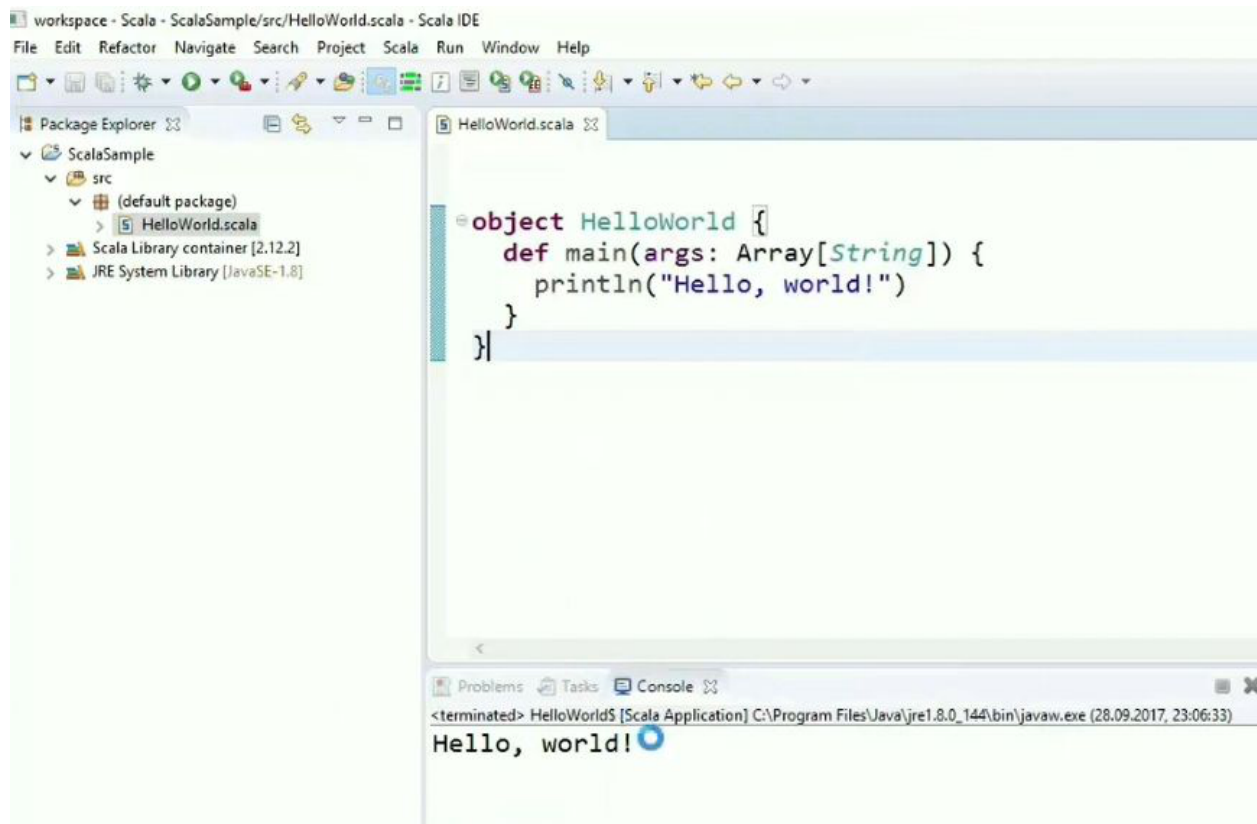
```
object Helloworld
{
    // Main Method
    def main(args: Array[String])
    {
        // prints Hello World
        println("Hello World!")
    }
}
```

Step 4 : Run the Scala program

Click on Run button or Run menu to run this as a Scala application.

Set the Run configuration if needed.

Output 



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project named 'ScalaSample' with a source folder 'src' containing a file 'HelloWorld.scala'. The main editor displays the code for 'HelloWorld.scala':

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

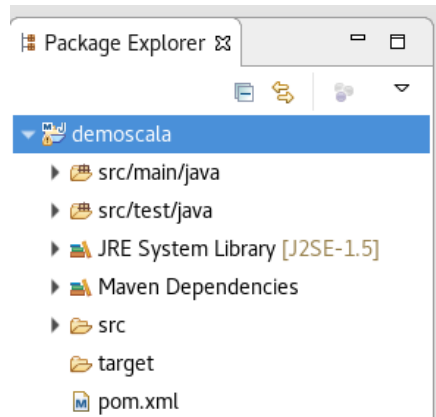
The Console window at the bottom shows the output of the program: 'Hello, world!'.

Apache Spark Scala program for Wordcount using Eclipse IDE

Step 1: create a new Maven Project by going to

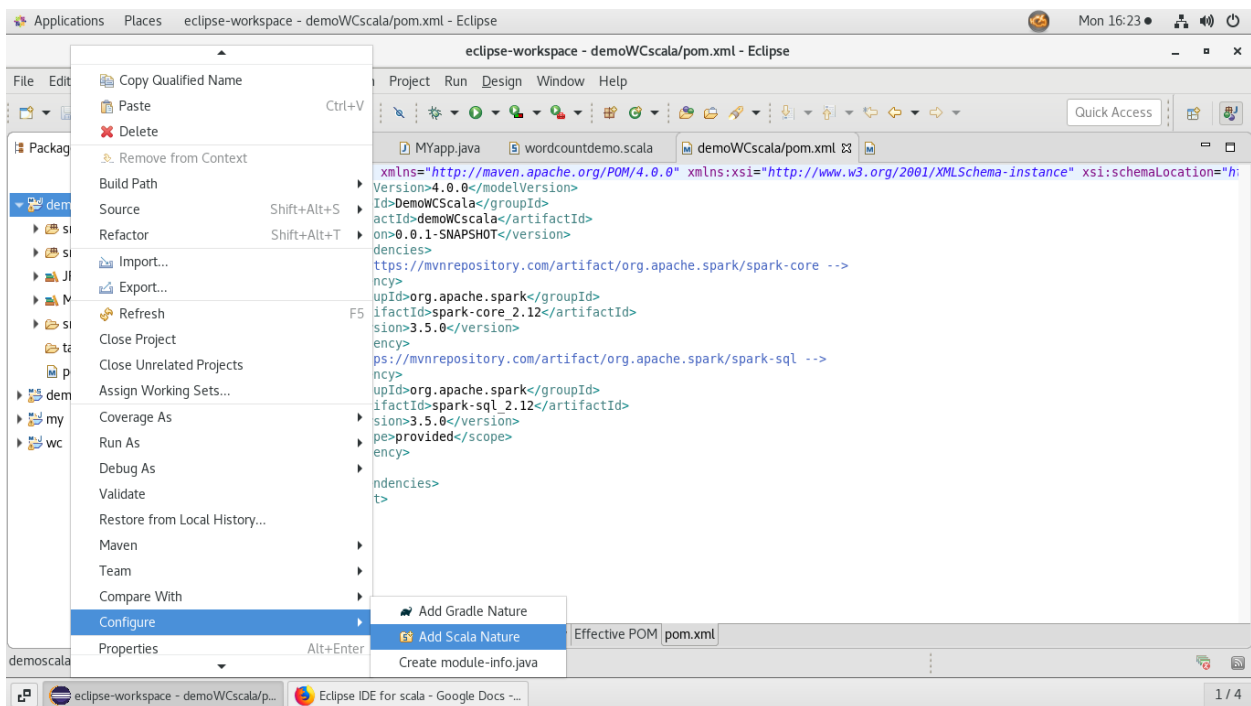
File -> new -> Project-> Maven-> Maven Project

Step 2: The maven Project with language java will be created by default

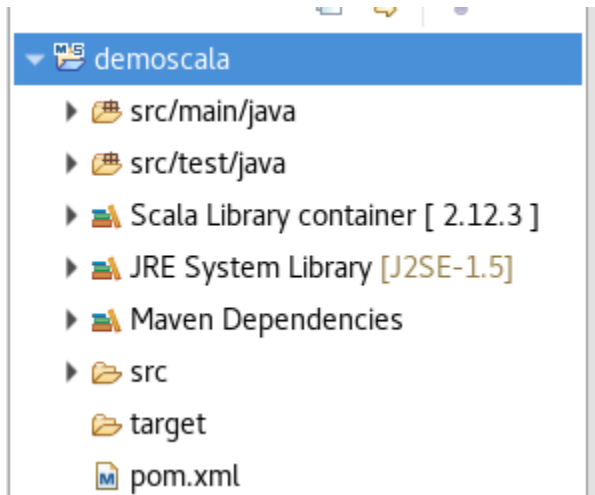


Step 3: configure the project with scala

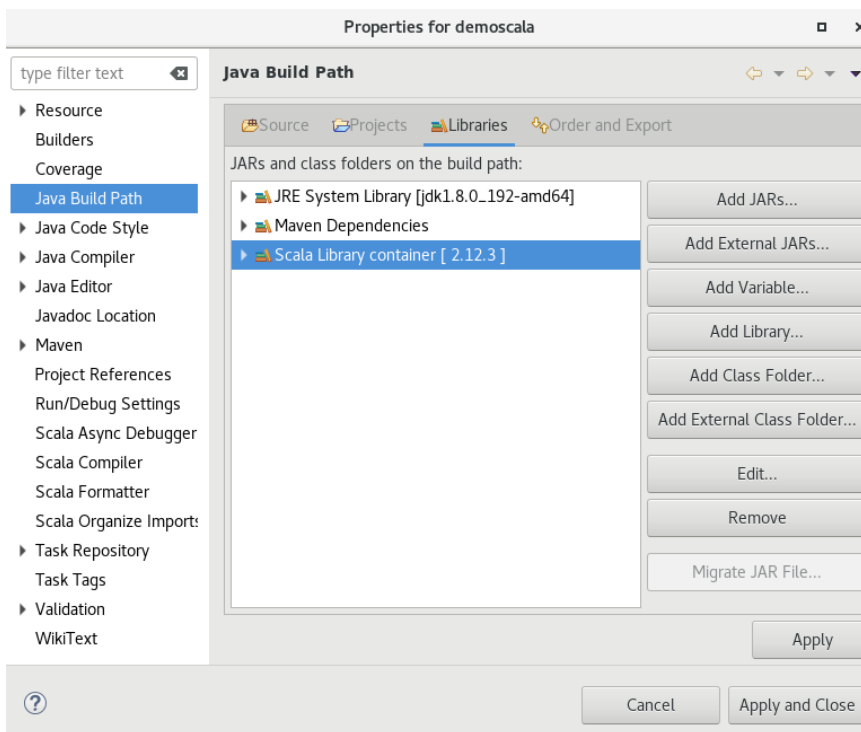
Right click on the project name -> configure-> add scala nature



Once the Maven project is configured as scala nature it will show **scala** on project name .



Step 4 : Change appropriate JRE version(minimum JRE =1.8)
Scala version should be compatible with spark scala version



Step 5 : Add the dependencies for spark core and spark sql

```

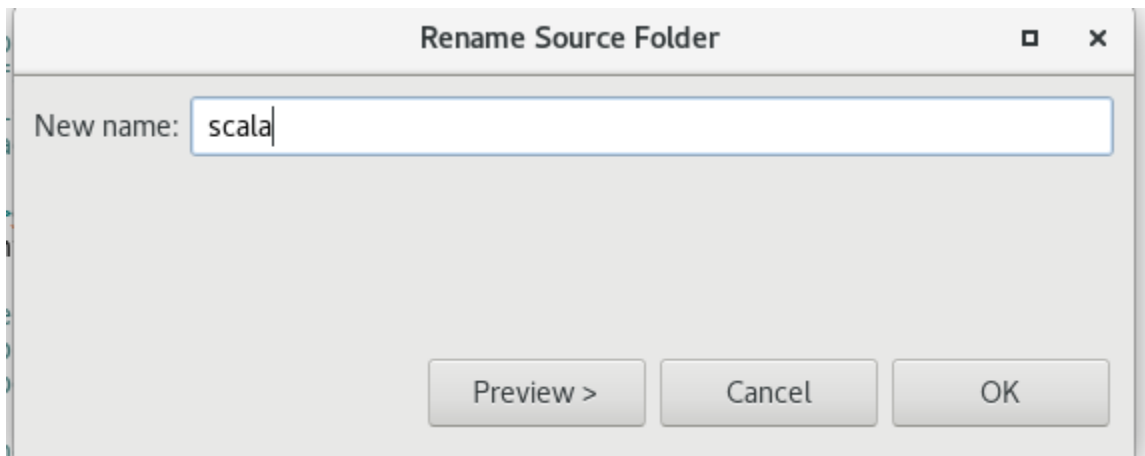
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>DemoWCScala</groupId>
4   <artifactId>demoWCScala</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <dependencies>
7     <!-- https://mvnrepository.com/artifact/org.apache.spark/spark-core -->
8     <dependency>
9       <groupId>org.apache.spark</groupId>
10      <artifactId>spark-core_2.12</artifactId>
11      <version>3.5.0</version>
12    </dependency>
13    <!-- https://mvnrepository.com/artifact/org.apache.spark/spark-sql -->
14    <dependency>
15      <groupId>org.apache.spark</groupId>
16      <artifactId>spark-sql_2.12</artifactId>
17      <version>3.5.0</version>
18      <scope>provided</scope>
19    </dependency>
20  </dependencies>
21 </project>

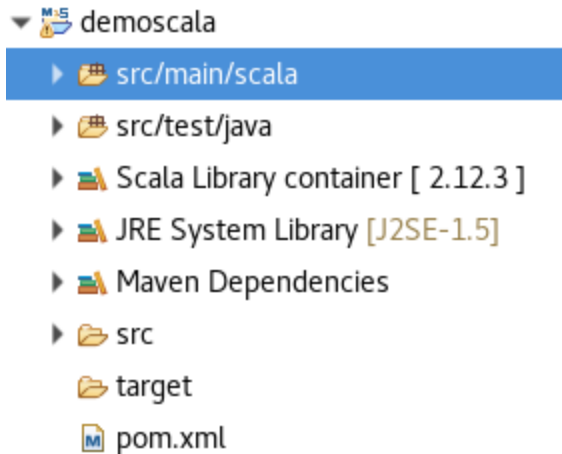
```

Maven will automatically download the dependencies for your project.

Step 6 : Rename the project source folder name to scala.

right click on the project name-> Refactor->> Rename

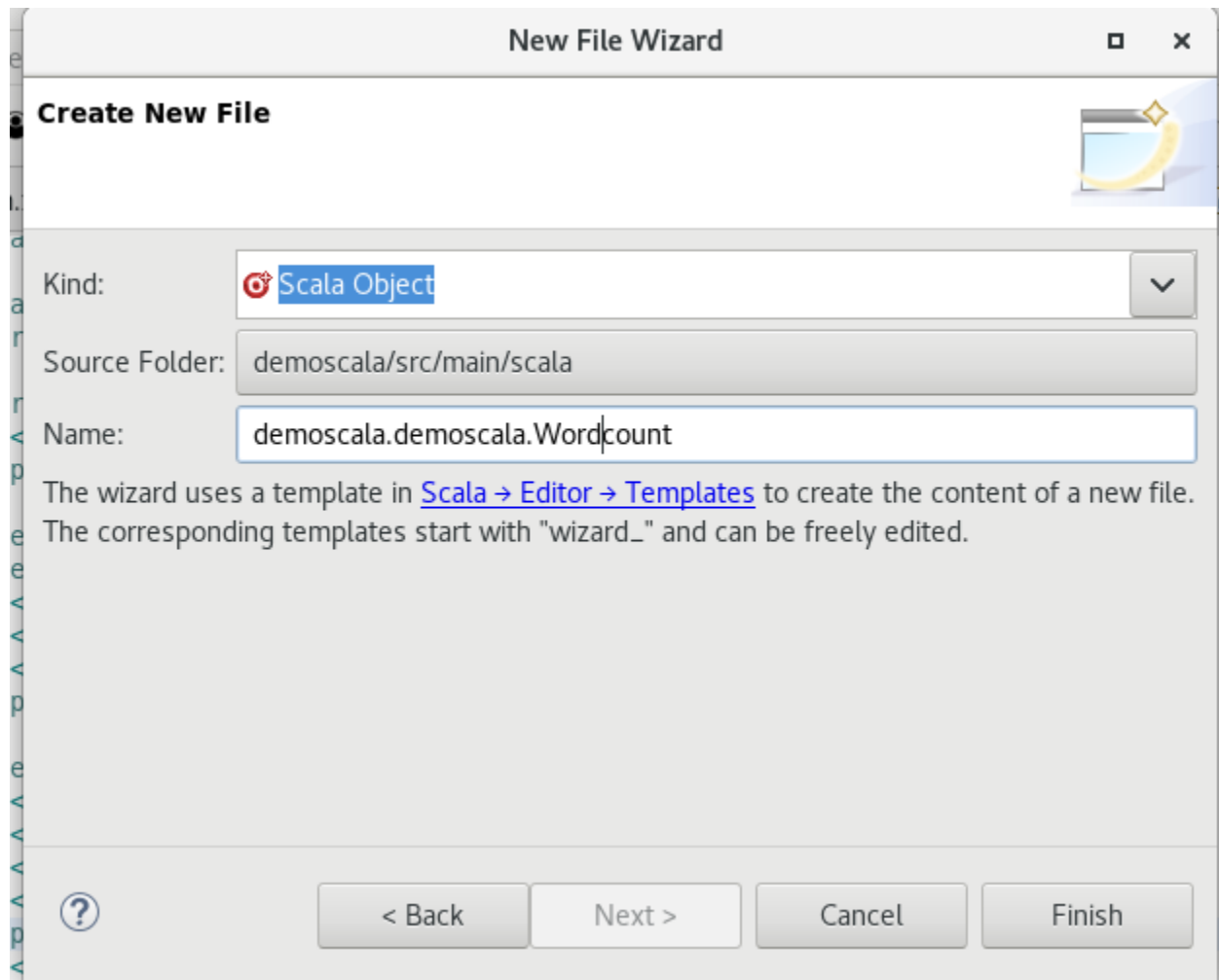




Update Scala compiler version for Spark:-

Scala IDE by default uses the latest version(2.11) of Scala compiler, however Spark uses version 2.10. So we need to update appropriate version for IDE. Right click on project- > Go to **properties** -> **Scala compiler** -> **update Scala installation** version to 2.12.x

Step - 7: create new object as wordcount



The image shows a 'New File Wizard' dialog box with a title bar containing a maximize button and a close button. The main area is titled 'Create New File' and features a decorative icon of a folder with a star and a crescent moon. Below the title, there are three input fields: 'Kind' with a dropdown menu showing 'Scala Object', 'Source Folder' with the text 'demoscala/src/main/scala', and 'Name' with the text 'demoscala.demoscala.Wordcount'. A paragraph of text explains that the wizard uses a template from 'Scala → Editor → Templates' and that templates start with 'wizard_'. At the bottom, there is a help icon (a question mark in a circle) and four buttons: '< Back', 'Next >', 'Cancel', and 'Finish'.

New File Wizard

Create New File

Kind: Scala Object

Source Folder: demoscala/src/main/scala

Name: demoscala.demoscala.Wordcount

The wizard uses a template in [Scala → Editor → Templates](#) to create the content of a new file. The corresponding templates start with "wizard_" and can be freely edited.

? < Back Next > Cancel Finish

Add the following code into file

```
package com.spark.firstapp
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD.rddToPairRDDFunctions
object wordcountdemo {
  def main(args: Array[String]) = {

    //Start the Spark context
    val conf = new SparkConf()
    .setAppName("WordCount")
    .setMaster("spark://localhost:7077") // master on standalone mode

    val sc = new SparkContext(conf)

    //Read some example file to a test RDD
    val test = sc.textFile(args(0));

    // for each line split the line in word by word.
    val words = test.flatMap(line => line.split(" "));

    //for each word Return a key/value tuple, with the word as key and 1 as
    value

    val counts = words.map(word => (word,1))

    //Sum all of the value with same key
    val wordcount = counts.reduceByKey(_+_);
```

```
val wordcount_sorted = wordcount.sortByKey()

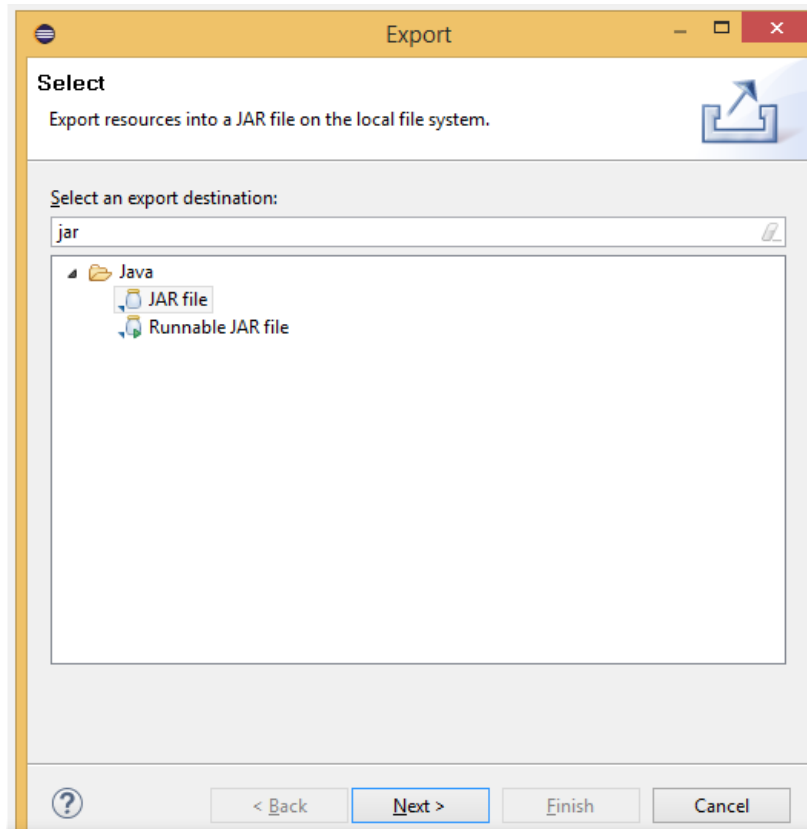
wordcount_sorted.foreach(println)

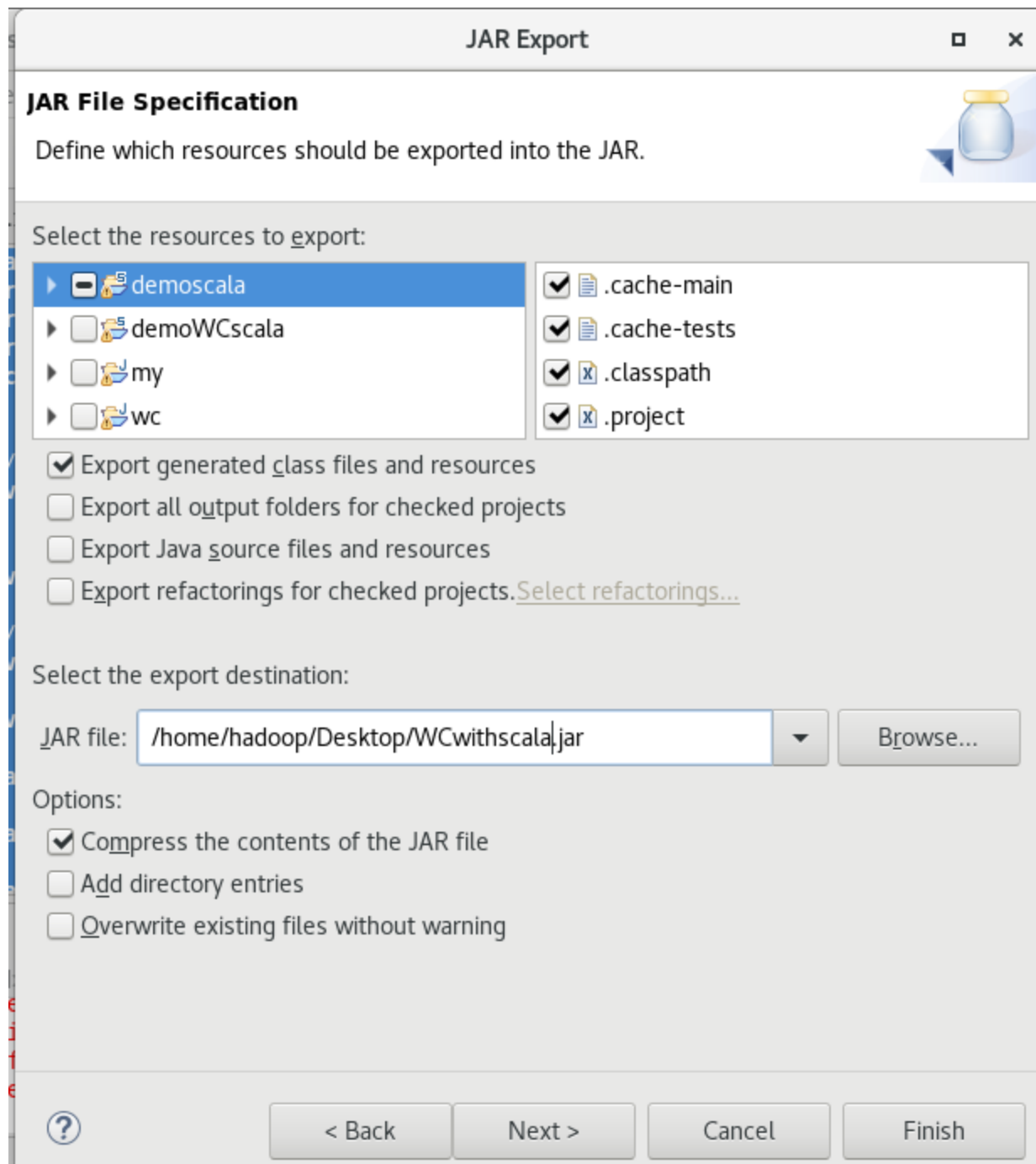
wordcount_sorted.saveAsTextFile(args(1)) //Save to a text file


//Stop the Spark context
sc.stop
}
```

Create the Spark Scala Program Jar File

Before running created **Spark word count application** we have to create a jar file. Right click on **project >> export**





Save the jar file on any location

Submit Spark Application using spark-submit script

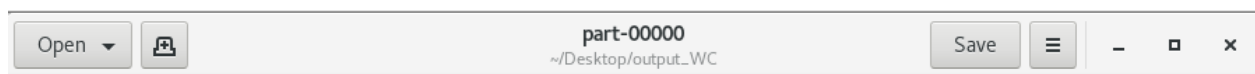
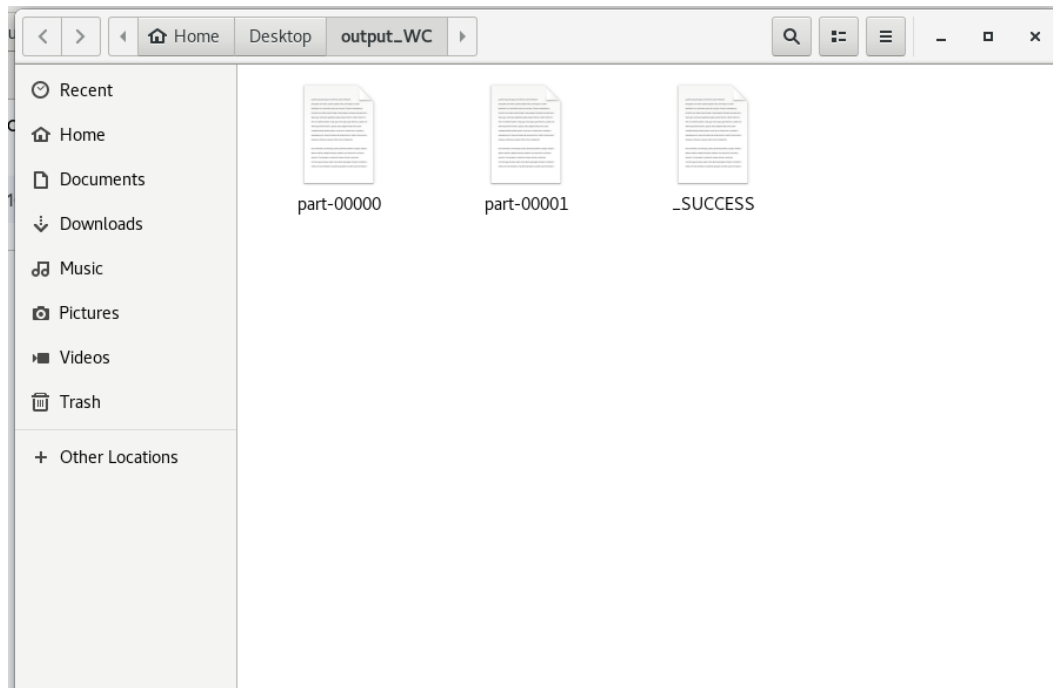
```
bin/spark-submit --class <Qualified-Class-Name> --master <Master>  
<Path-Of-Jar-File> <Input-Path> <Output-Path>
```

```
bin/spark-submit --class demoscala.demoscala.Wordcount  
--master spark://localhost:7077  
--deploy-mode=cluster  
/home/hadoop/Desktop/input/inputdata.txt  
/home/hadoop/Desktop/Output
```

Let's understand above command:

- **bin/spark-submit:** To submit Spark Application
- **--class:** To specify the class name to execute
- **--master:** Master (local / <Spark-URI> / yarn)
- **<Jar-Path>:** The jar file of application
- **<Input-Path>:** Location from where input data will be read
- **<Output-Path>:** Location where Spark application will write output
- **Deploymode** = cluster or yarn(bydefault local)

Output of reducer as shown below



```
(bear,4)
(deer,1)
(river,3)
(apple,4)
(dear,4)
(cat,2)
```