基于Spark的K-mer计数

08012101 张泽渊 1120200325

目录

[1环境搭建 2](#_Toc135668052)

[1.1安装Scala插件 2](#_Toc135668053)

[1.2用idea打开项目spark-yunjisuan 3](#_Toc135668054)

[1.3将以下文件导入为ScalaSDK 3](#_Toc135668055)

[1.4 maven引包 4](#_Toc135668056)

[1.5将src.main.java文件标记为Sources Boot 7](#_Toc135668057)

[1.6环境搭建完成，右键创建Scala类即可 7](#_Toc135668058)

[1.7参考资料 7](#_Toc135668059)

[2程序编写 8](#_Toc135668060)

[2.1题目要求 8](#_Toc135668061)

[2.2解题代码 8](#_Toc135668062)

[2.3运行截图 10](#_Toc135668063)

[3出现的问题及解决方案 12](#_Toc135668064)

[3.1出现大量警告 12](#_Toc135668065)

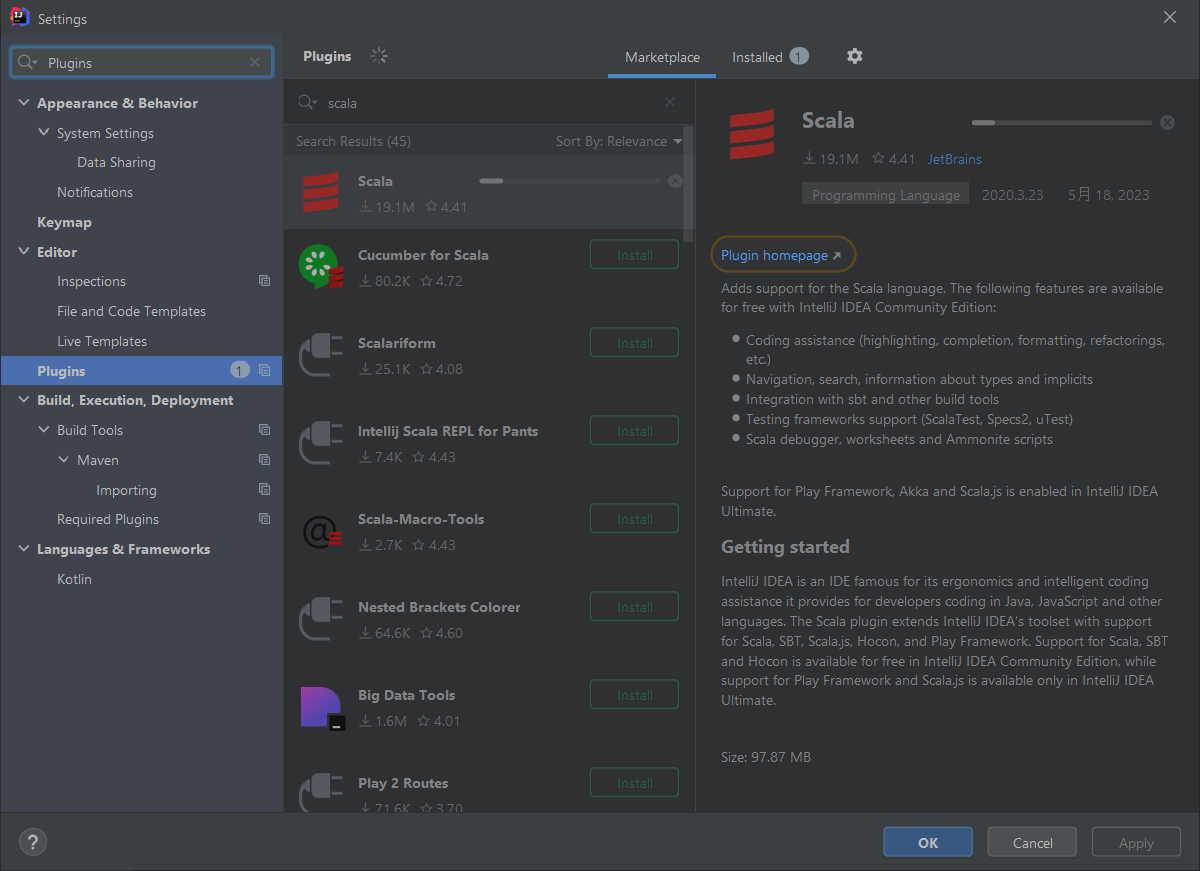
[3.2控制台出现大量红色信息 12](#_Toc135668066)

[3.3出现大量报错 14](#_Toc135668067)

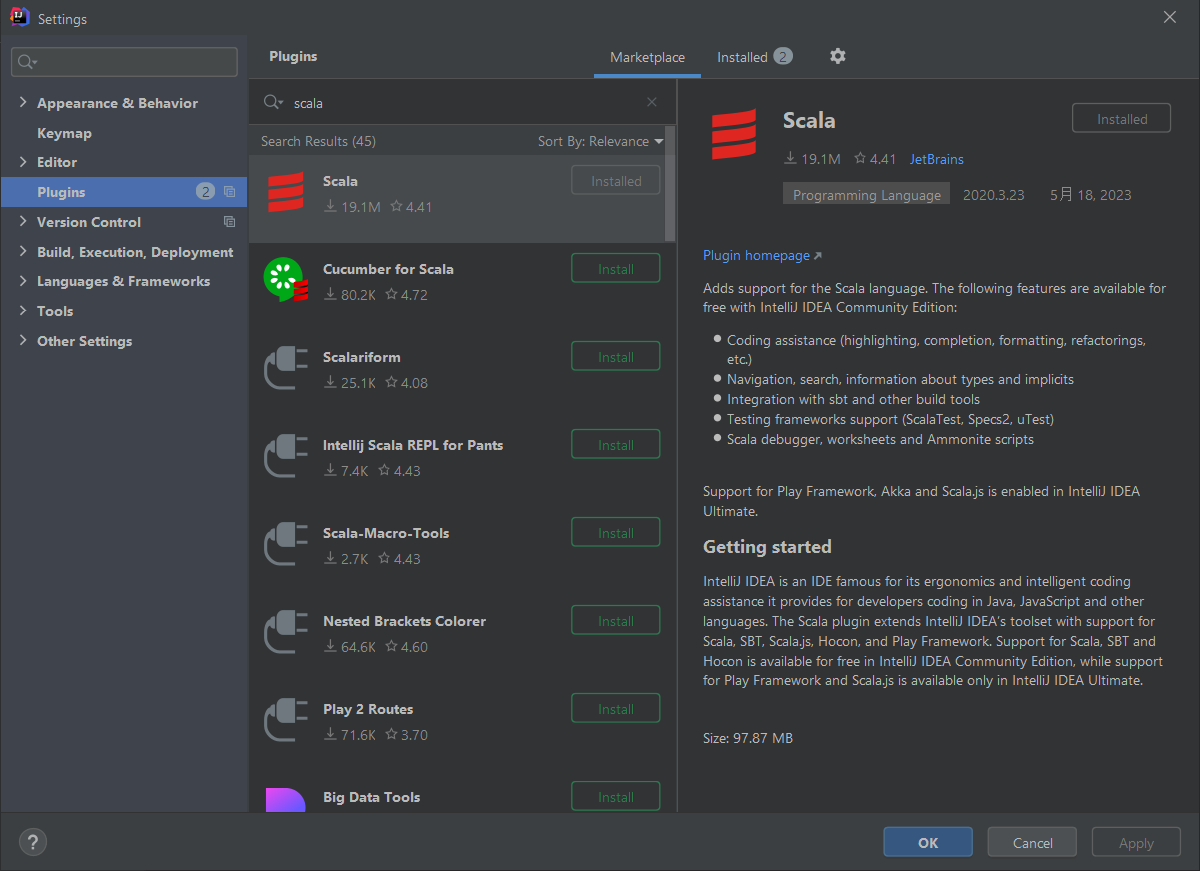
[备注 15](#_Toc135668068)

# 1环境搭建

## 1.1安装Scala插件



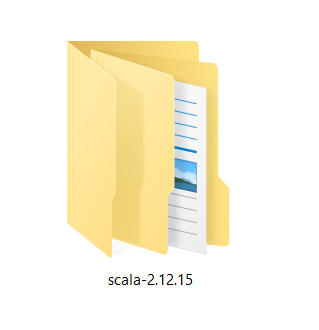
安装完成：

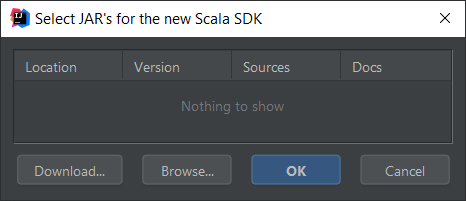


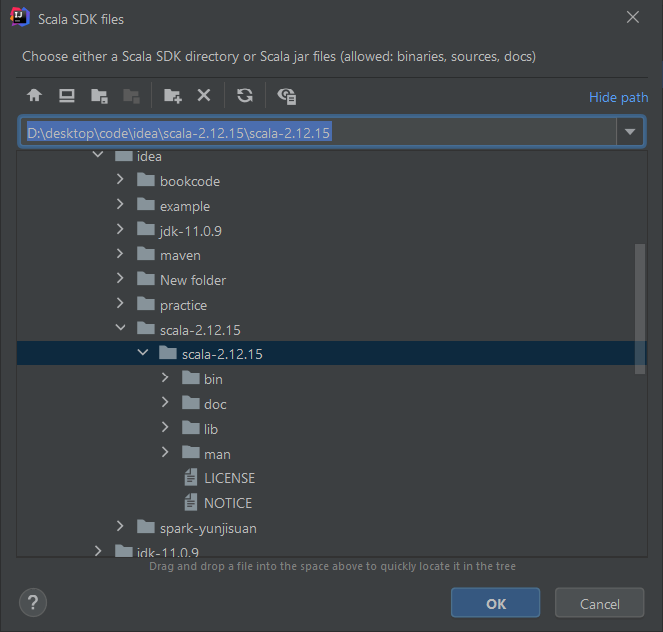
## 1.2用idea打开项目spark-yunjisuan



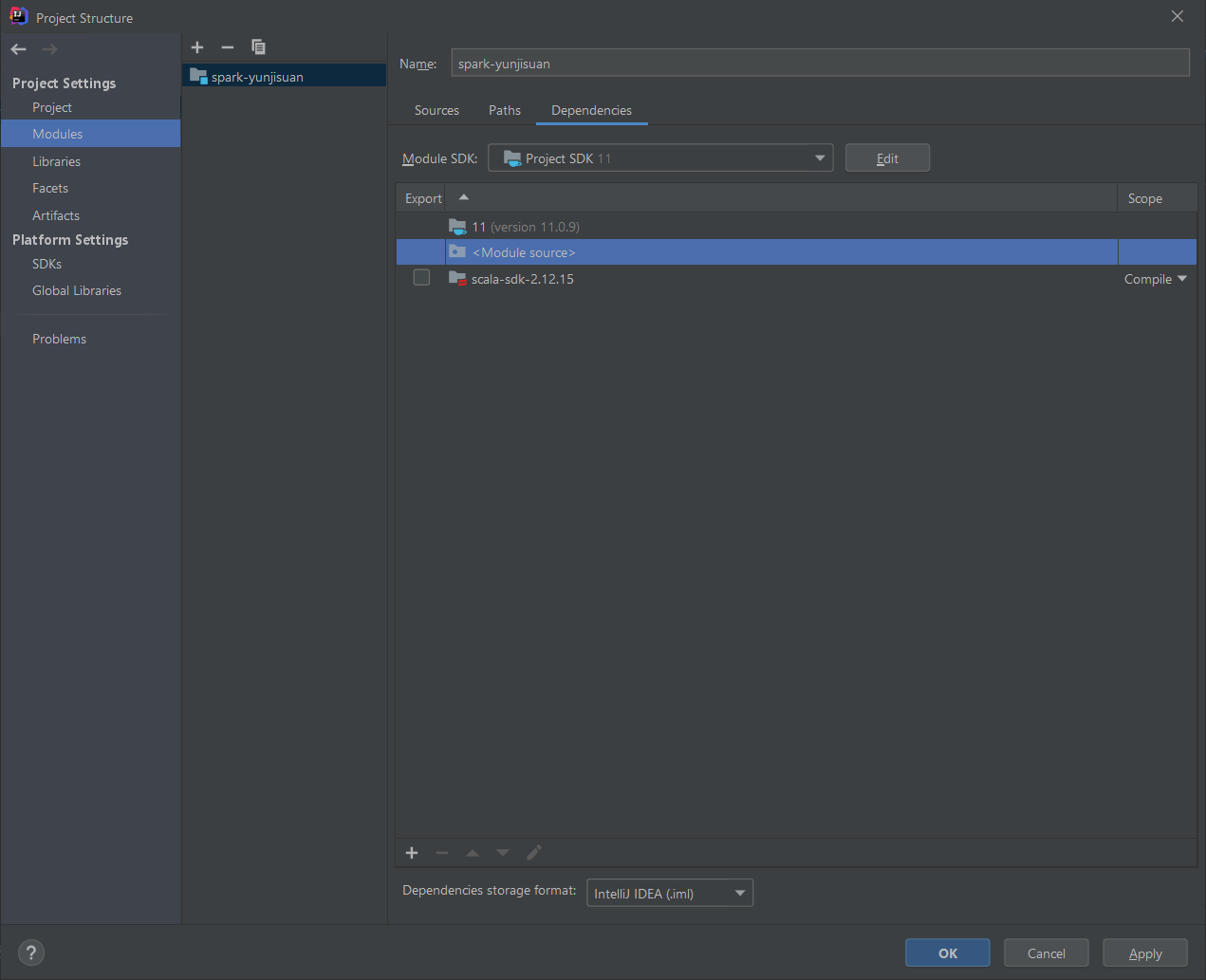
## 1.3将以下文件导入为ScalaSDK





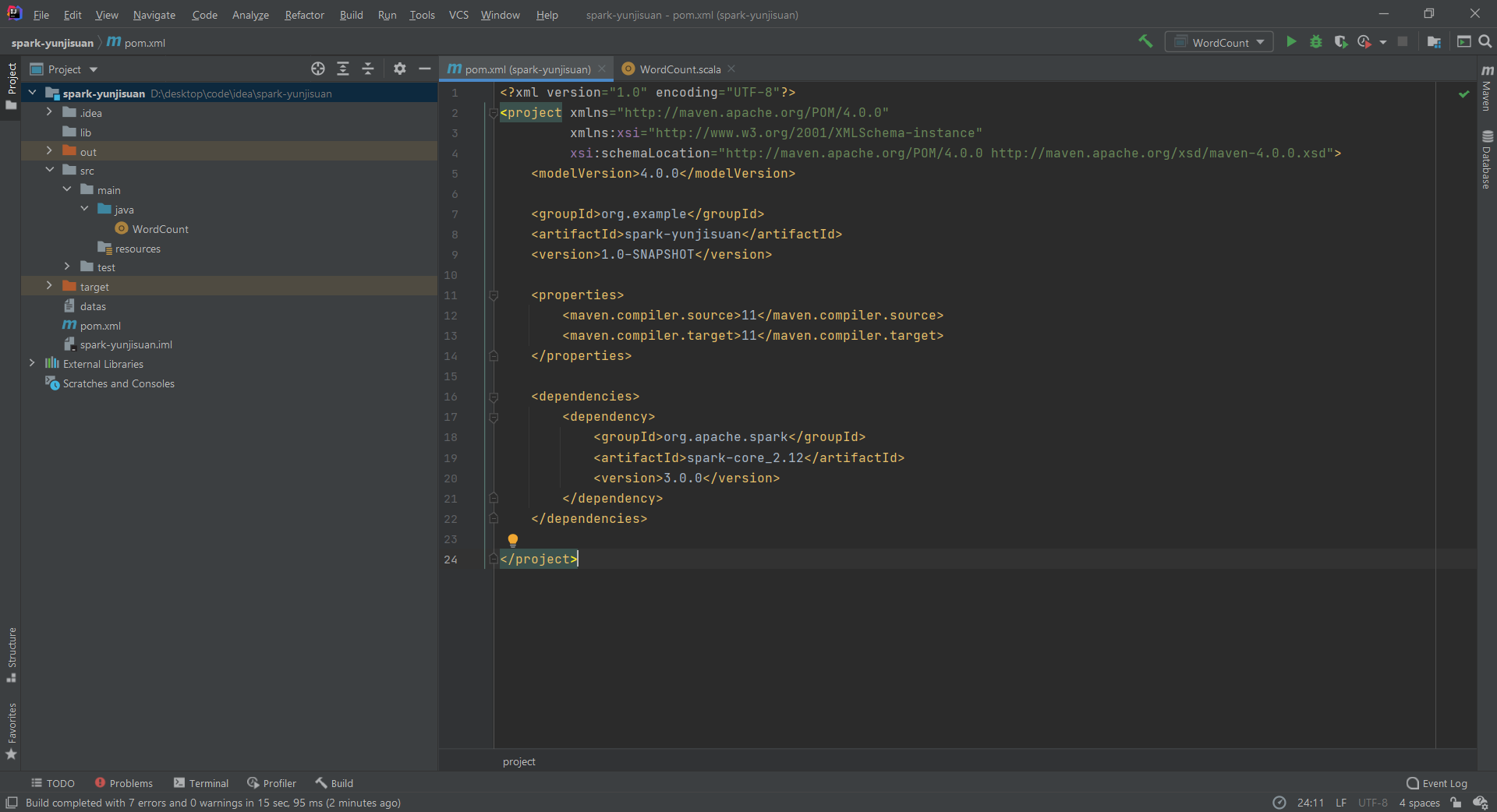


导入完成后显示如下：



## 1.4 maven引包

打开pom.xml：



导入以下代码：

--------------------------代码分割线------------------------

<!-- 基础配置 -->

<build>

<plugins>

<plugin>

<groupId>net.alchim31.maven</groupId>

<artifactId>scala-maven-plugin</artifactId>

<version>3.2.2</version>

<executions>

<execution>

<goals>

<goal>testCompile</goal>

</goals>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-assembly-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

</configuration>

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

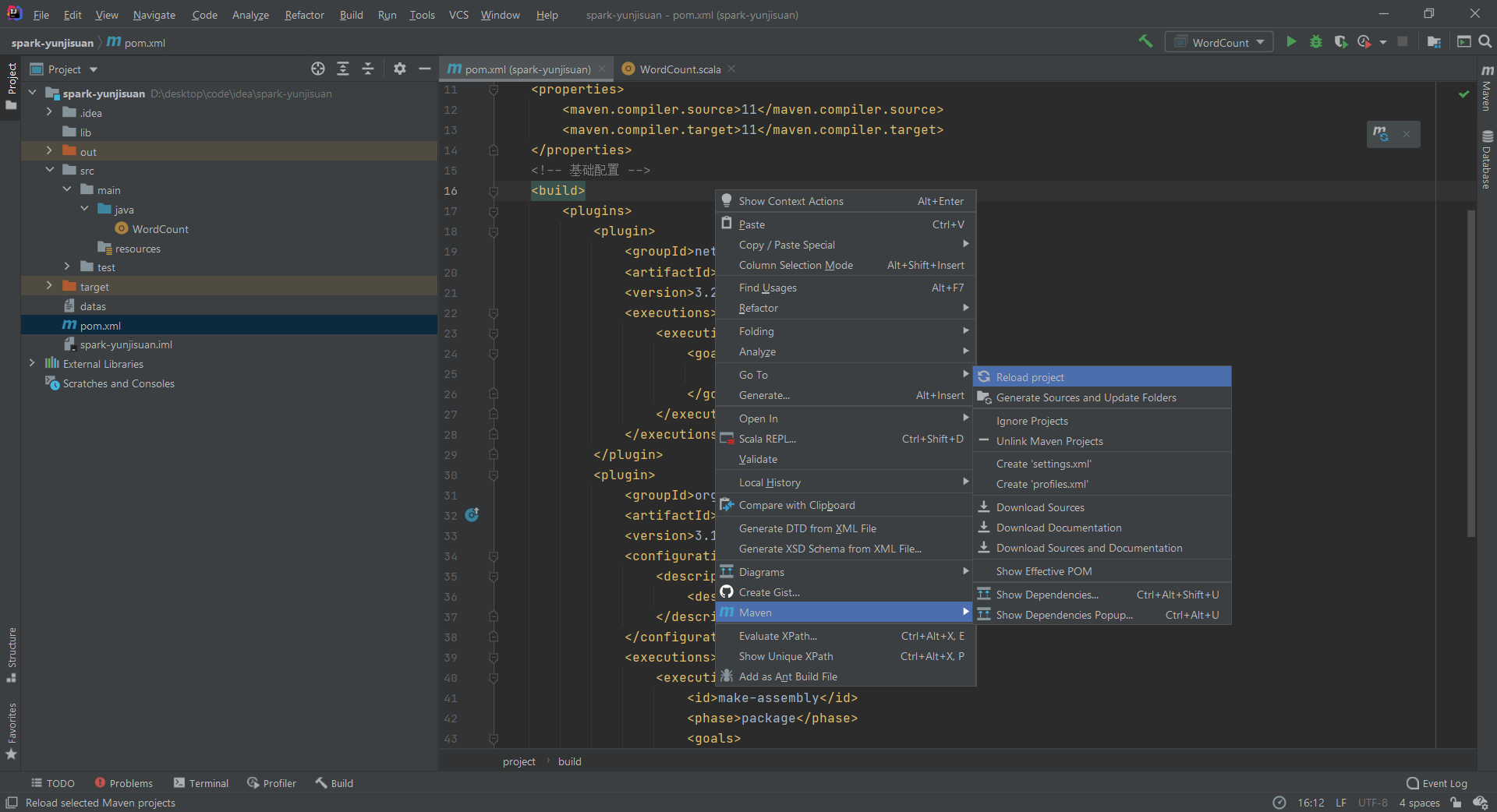
</plugin>

</plugins>

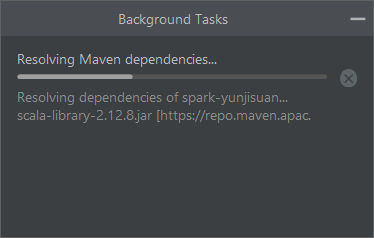
</build>

--------------------------代码分割线------------------------

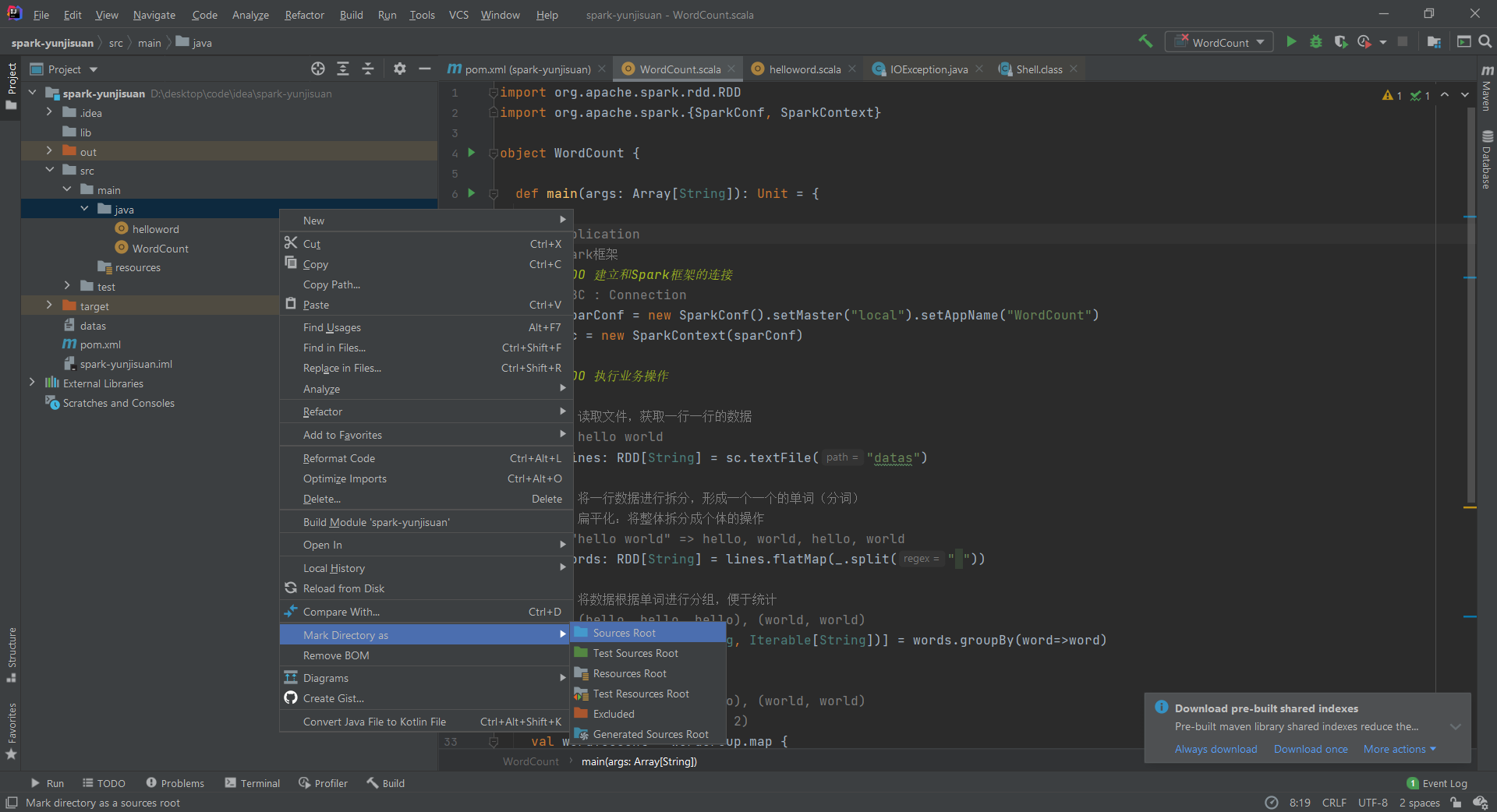
重新加载maven：



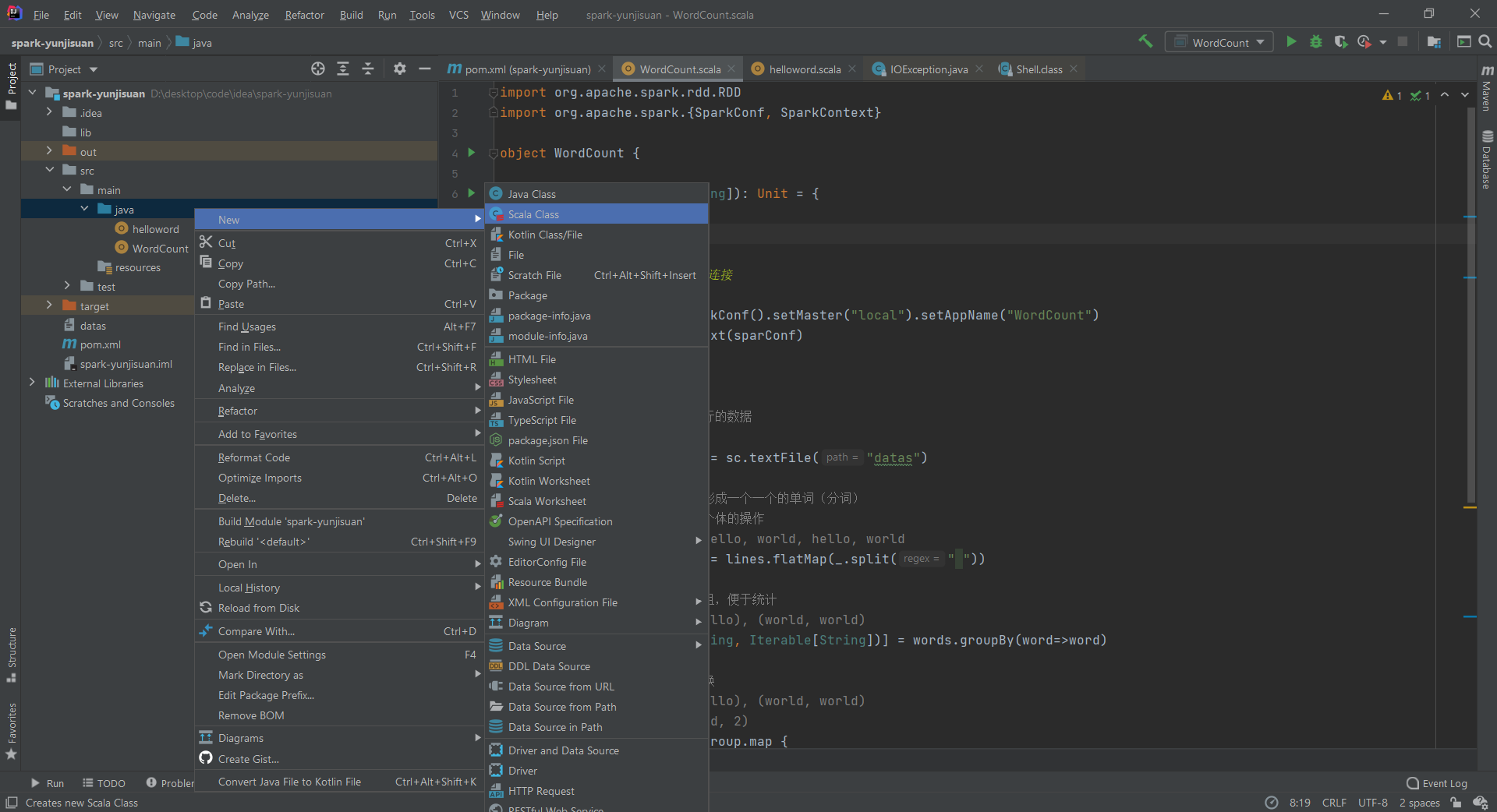
等待加载完成：



## 1.5将src.main.java文件标记为Sources Boot



## 1.6环境搭建完成，右键创建Scala类即可



## 1.7参考资料

[spark环境搭建(idea版本)\_idea搭建spark开发环境\_红目香薰的博客-CSDN博客](https://blog.csdn.net/feng8403000/article/details/122693015)

# 2程序编写

## 2.1题目要求

问题描述:

K-mer是一个长度为K ( K > 0 )的子串，K-mer计数是指整个序列中K-mer出现的频度，本题需要返回出现次数排在前N的子串和出现频率。

假如输入的字符串为ABCDA，K=2，那么子串包含 AB, BC, CD, DA

参数：

K = 2（子串长度）

N = 10（返回排在前N的子串和频率）

输入格式：KM-input.txt

S（长度任意的字符串）

输出结果：

1. 问题1：共N组输出，每组输出为字符串和出现频率，按照出现次数降序排序，若出现次数相同，按照字符串的字典序降序排序。
2. 问题2：输出一个浮点数，为所有字符串出现次数的平均值

## 2.2解题代码

（粘贴到word有些乱，可以看代码后的截图，比较整齐）

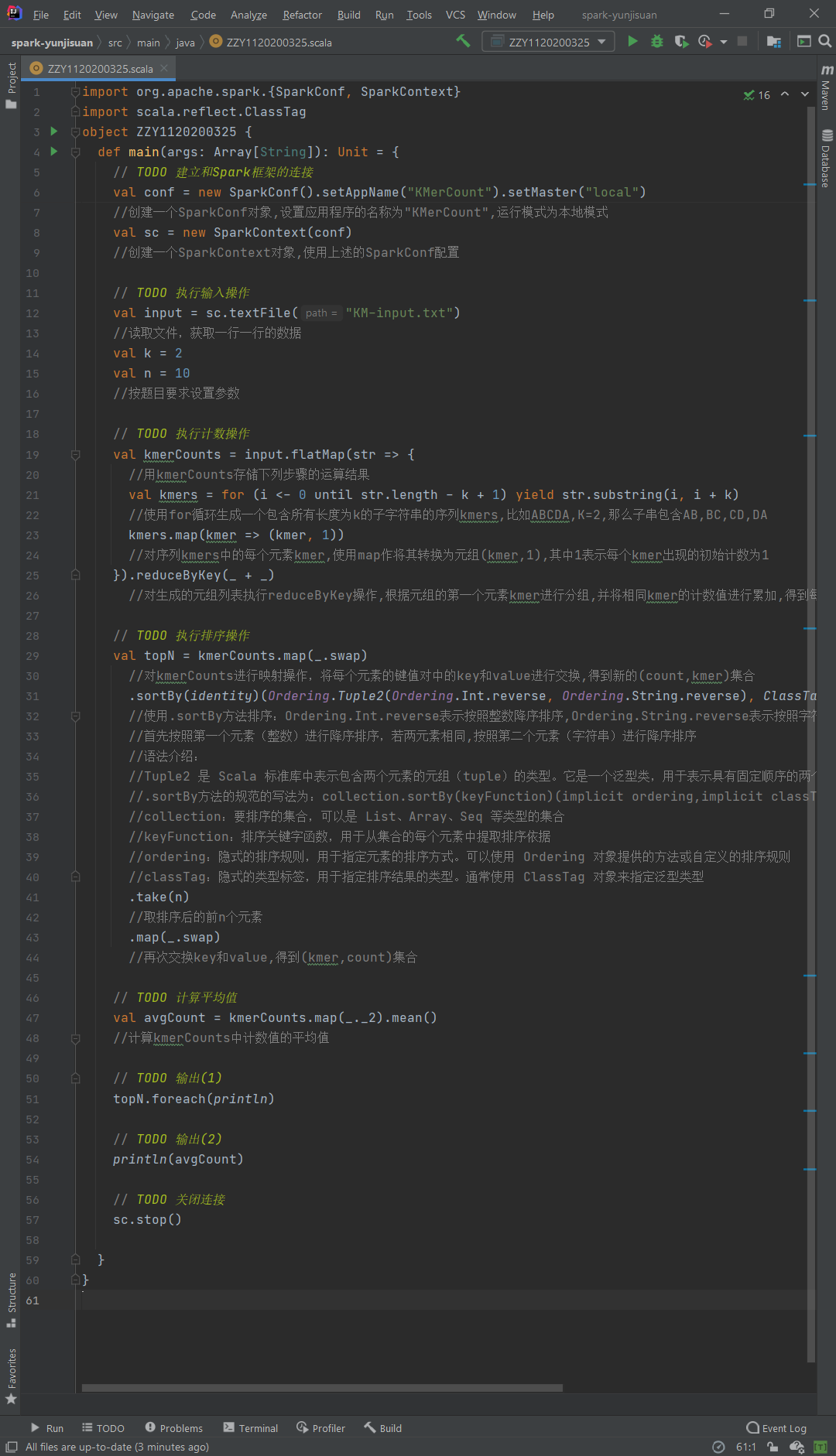
--------------------------代码分割线------------------------

**import** org.apache.spark.{SparkConf, SparkContext}  
**import** scala.reflect.ClassTag  
**object** ZZY1120200325 {  
 **def** main(args: Array[String]): Unit = {  
 *//* ***TODO 建立和Spark框架的连接* val** conf = **new** SparkConf().setAppName(**"KMerCount"**).setMaster(**"local"**)  
 *//创建一个SparkConf对象,设置应用程序的名称为"KMerCount",运行模式为本地模式* **val** sc = **new** SparkContext(conf)  
 *//创建一个SparkContext对象,使用上述的SparkConf配置  
  
 //* ***TODO 执行输入操作* val** input = sc.textFile(**"KM-input.txt"**)  
 *//读取文件，获取一行一行的数据* **val** k = 2  
 **val** n = 10  
 *//按题目要求设置参数  
  
 //* ***TODO 执行计数操作* val** kmerCounts = input.flatMap(str => {  
 *//用kmerCounts存储下列步骤的运算结果* **val** kmers = **for** (i <- 0 until str.length - k + 1) **yield** str.substring(i, i + k)  
 *//使用for循环生成一个包含所有长度为k的子字符串的序列kmers,比如ABCDA,K=2,那么子串包含AB,BC,CD,DA* kmers.map(kmer => (kmer, 1))  
 *//对序列kmers中的每个元素kmer,使用map作将其转换为元组(kmer,1),其中1表示每个kmer出现的初始计数为1* }).reduceByKey(\_ + \_)  
 *//对生成的元组列表执行reduceByKey操作,根据元组的第一个元素kmer进行分组,并将相同kmer的计数值进行累加,得到每个K-mer出现的次数  
  
 //* ***TODO 执行排序操作* val** topN = kmerCounts.map(\_.swap)  
 *//对kmerCounts进行映射操作，将每个元素的键值对中的key和value进行交换,得到新的(count,kmer)集合* .sortBy(*identity*)(*Ordering*.*Tuple2*(*Ordering*.Int.reverse, *Ordering*.String.reverse), *ClassTag*[(Int, String)](*classOf*[(Int, String)]))  
 *//使用.sortBy方法排序：Ordering.Int.reverse表示按照整数降序排序,Ordering.String.reverse表示按照字符串降序排序  
 //首先按照第一个元素（整数）进行降序排序，若两元素相同,按照第二个元素（字符串）进行降序排序  
 //语法介绍：  
 //Tuple2 是 Scala 标准库中表示包含两个元素的元组（tuple）的类型。它是一个泛型类，用于表示具有固定顺序的两个值的组合  
 //.sortBy方法的规范的写法为：collection.sortBy(keyFunction)(implicit ordering,implicit classTag)  
 //collection：要排序的集合，可以是 List、Array、Seq 等类型的集合  
 //keyFunction：排序关键字函数，用于从集合的每个元素中提取排序依据  
 //ordering：隐式的排序规则，用于指定元素的排序方式。可以使用 Ordering 对象提供的方法或自定义的排序规则  
 //classTag：隐式的类型标签，用于指定排序结果的类型。通常使用 ClassTag 对象来指定泛型类型* .take(n)  
 *//取排序后的前n个元素* .map(\_.swap)  
 *//再次交换key和value,得到(kmer,count)集合  
  
 //* ***TODO 计算平均值* val** avgCount = kmerCounts.map(\_.\_2).mean()  
 *//计算kmerCounts中计数值的平均值  
  
 //* ***TODO 输出(1)*** topN.foreach(*println*)  
  
 *//* ***TODO 输出(2)*** *println*(avgCount)  
  
 *//* ***TODO 关闭连接*** sc.stop()  
  
 }  
}

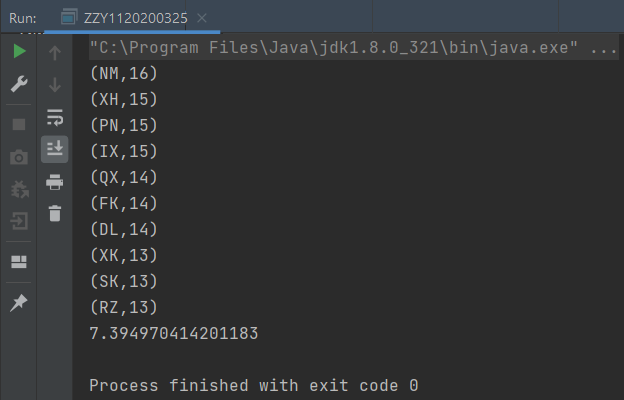
--------------------------代码分割线------------------------

## 2.3运行截图

程序：



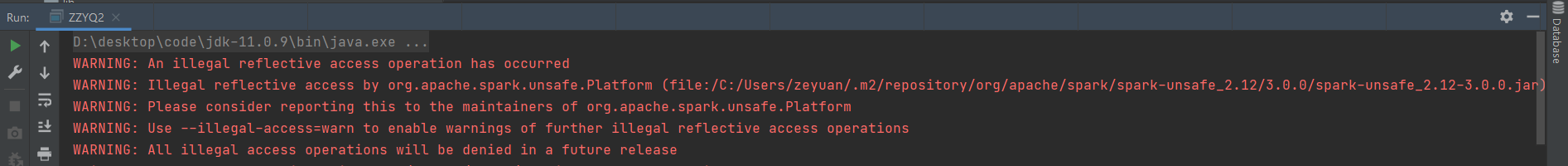
结果：



# 3出现的问题及解决方案

## 3.1出现大量警告

代码截图：



出现原因：

使用的Java版本过高

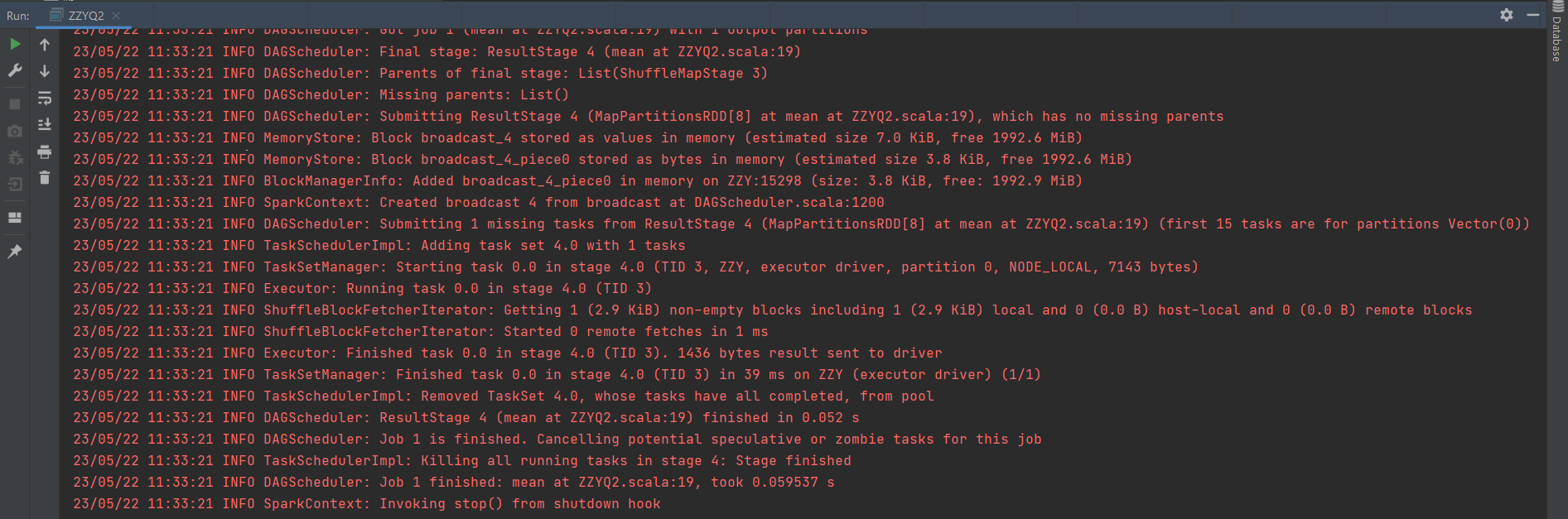
解决方法：

将java版本切换到1.8后警告消失。

参考资料：[测试时出现警告：WARNING: An illegal reflective access operation has occurred WARNING: Illegal reflective acc-CSDN博客](https://blog.csdn.net/Alomed/article/details/123821123)

## 3.2控制台出现大量红色信息

代码截图：



出现原因：

spark输出大量日志

解决方法：

在resource文件夹下，新建log4j.properties文件，输入以下代码：

--------------------------代码分割线------------------------

log4j.rootCategory=ERROR, console

log4j.appender.console=org.apache.log4j.ConsoleAppender

log4j.appender.console.target=System.err

log4j.appender.console.layout=org.apache.log4j.PatternLayout

log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Set the default spark-shell log level to ERROR. When running the spark-shell, the

# log level for this class is used to overwrite the root logger's log level, so that

# the user can have different defaults for the shell and regular Spark apps.

log4j.logger.org.apache.spark.repl.Main=ERROR

# Settings to quiet third party logs that are too verbose

log4j.logger.org.spark\_project.jetty=ERROR

log4j.logger.org.spark\_project.jetty.util.component.AbstractLifeCycle=ERROR

log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=ERROR

log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=ERROR

log4j.logger.org.apache.parquet=ERROR

log4j.logger.parquet=ERROR

# SPARK-9183: Settings to avoid annoying messages when looking up nonexistent UDFs in SparkSQL with Hive support

log4j.logger.org.apache.hadoop.hive.metastore.RetryingHMSHandler=FATAL

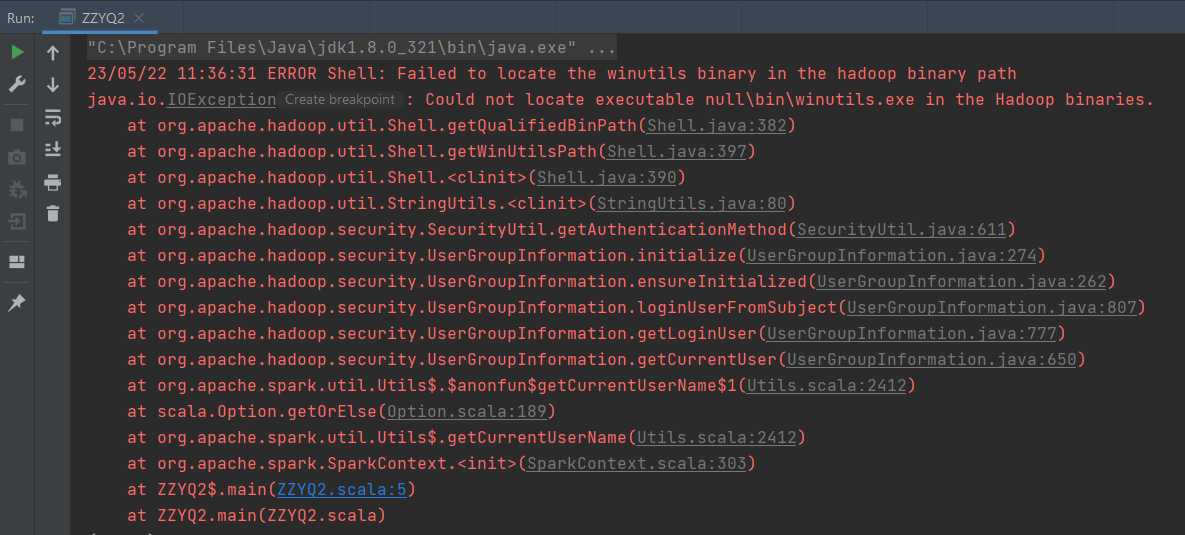
log4j.logger.org.apache.hadoop.hive.ql.exec.FunctionRegistry=ERROR

--------------------------代码分割线------------------------

参考资料：[修改spark中显示日志等级\_using spark's default log4j profile: org/apache/sp\_偶白的博客-CSDN博客](https://blog.csdn.net/OWBY_Phantomhive/article/details/123086181)

## 3.3出现大量报错

代码截图：

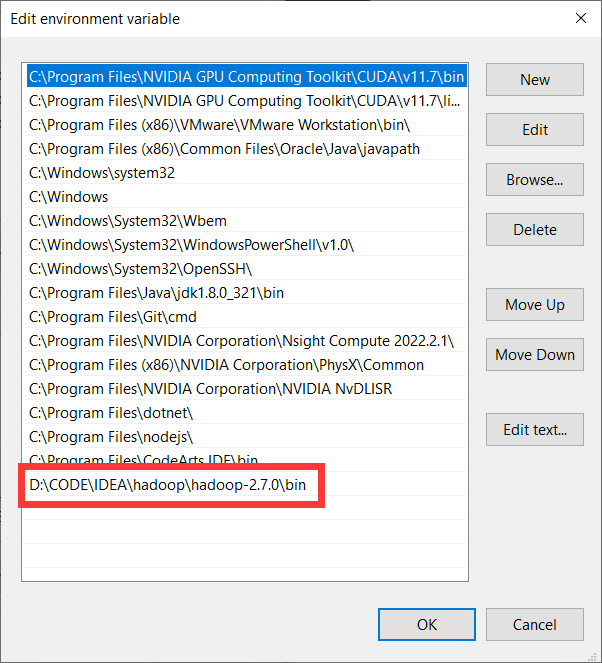
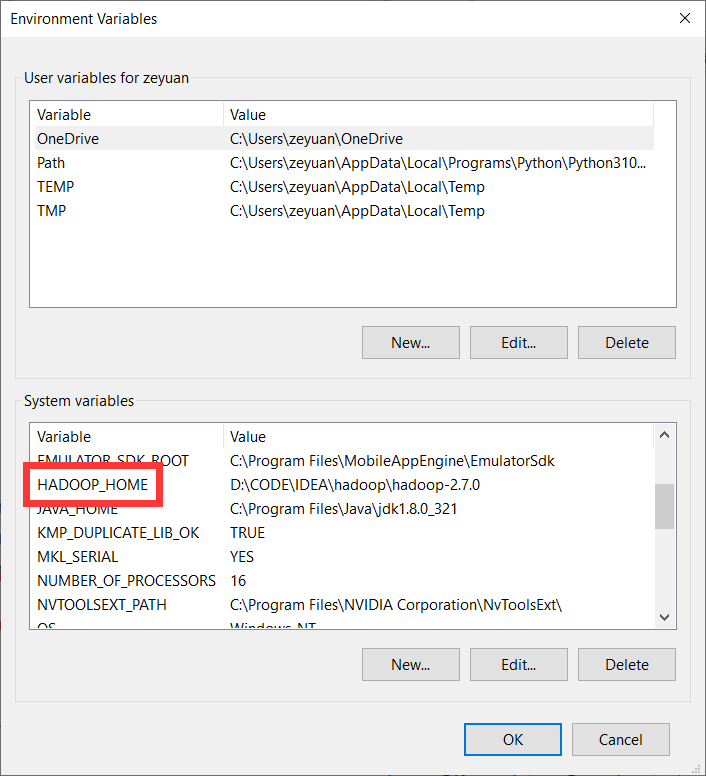


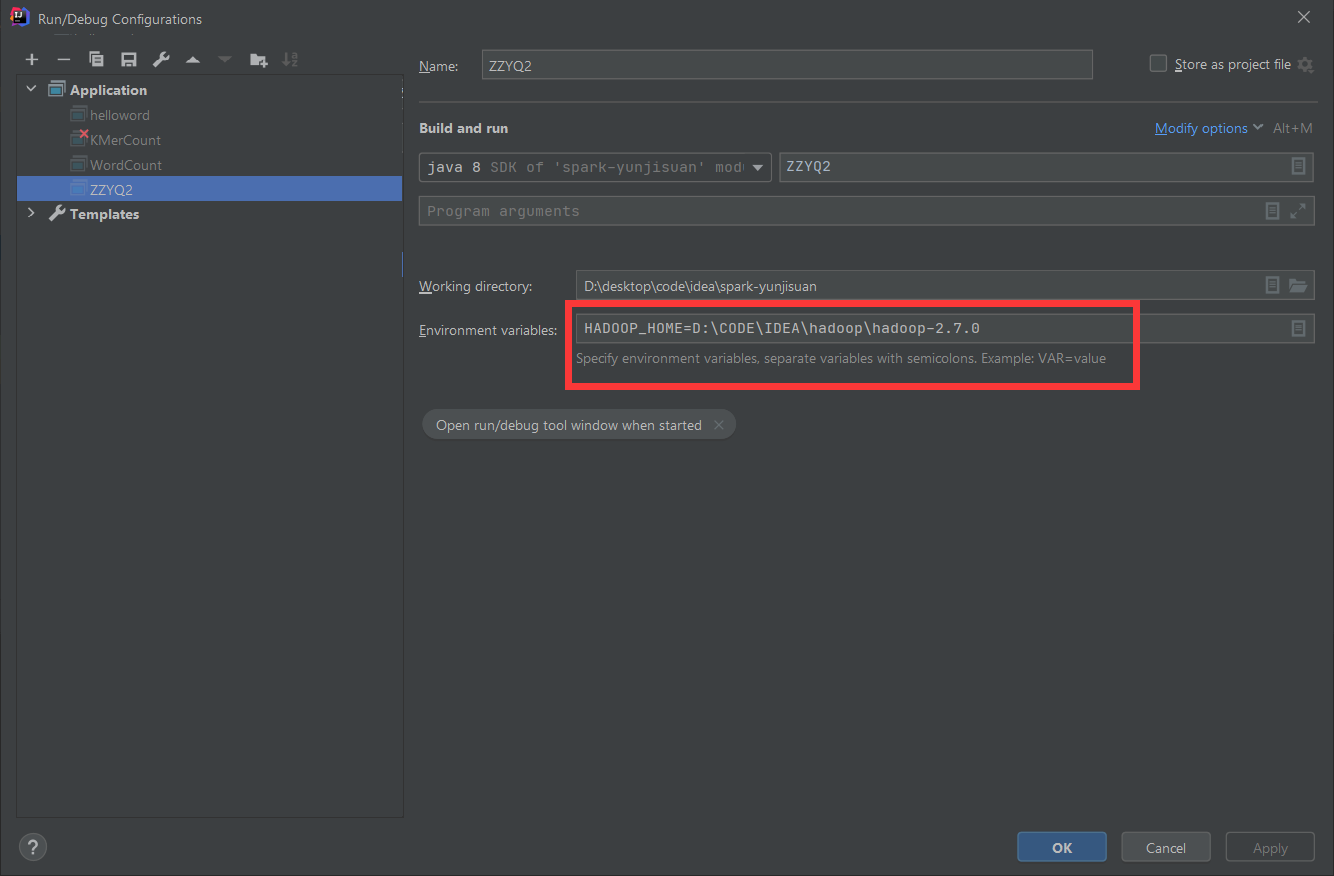
出现原因：

程序用到了hadoop相关的服务，但windows本地无法获取到hadoop的相关配置

解决方法：

下载HADOOP并配置环境变量：





参考资料：[ERROR Shell: Failed to locate the winutils binary in the hadoop binary path java.io.IOException:\_皮哥四月红的博客-CSDN博客](https://blog.csdn.net/weixin_43230682/article/details/107717609)

# 备注

完整的项目文件可访问：

[vhthree/YunJiSuan2023: 云计算与应用作业提交备份 (github.com)](https://github.com/vhthree/YunJiSuan2023)