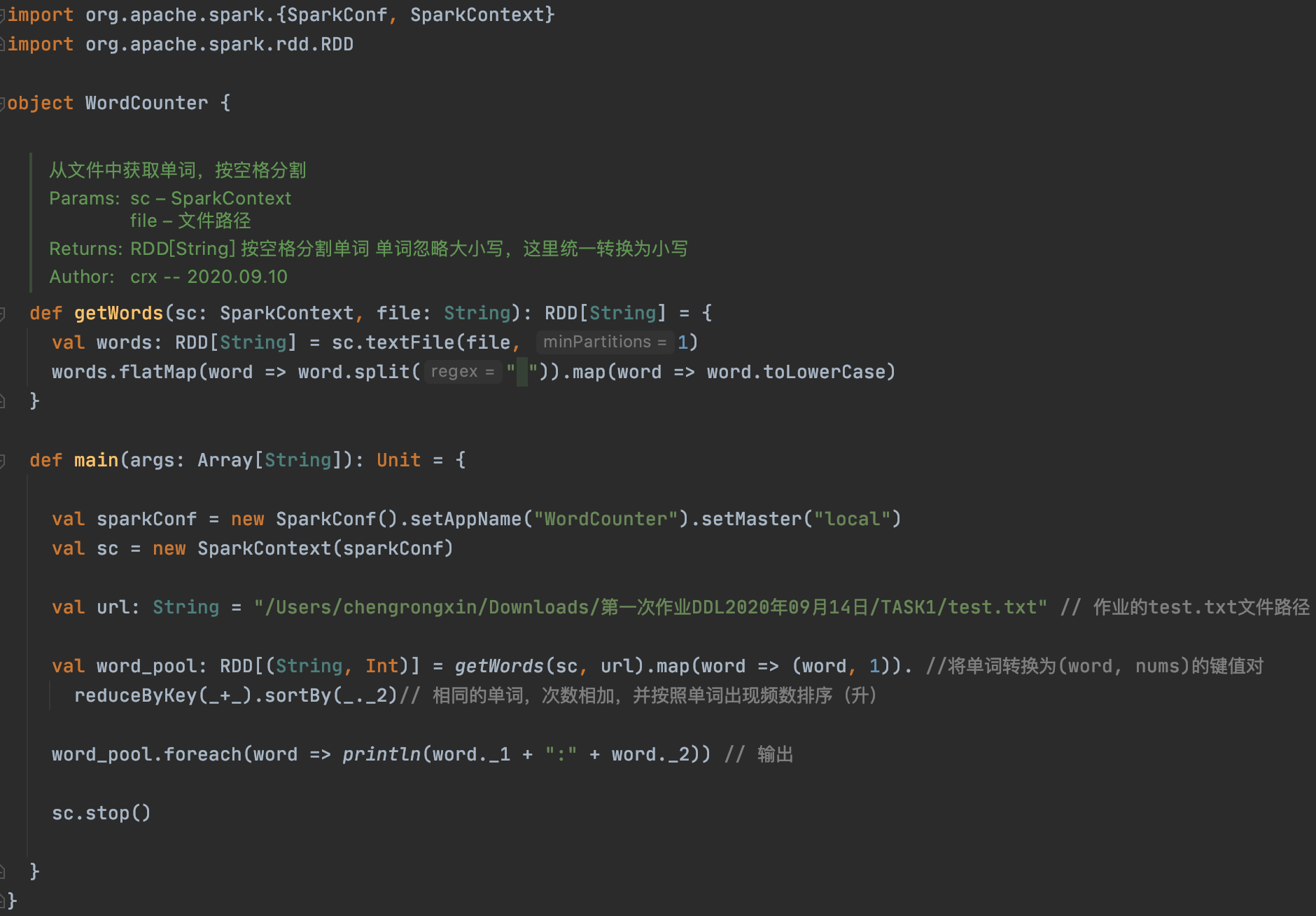
# BDA第一次作业

## 源代码&运行结果

**源代码**

代码截图如下所示：



Scala-SDK版本为2.12.12

JDK版本1.8

代码文本如下：

import org.apache.spark.{SparkConf, SparkContext}

import org.apache.spark.rdd.RDD

object WordCounter {

/\*\*从文件中获取单词，按空格分割

\* @param sc SparkContext

\* @param file 文件路径

\* @return RDD[String]

\* 按空格分割单词

\* 单词忽略大小写，这里统一转换为小写

\* @author crx -- 2020.09.10

\*/

def getWords(sc: SparkContext, file: String): RDD[String] = {

val words: RDD[String] = sc.textFile(file, 1)

words.flatMap(word => word.split(" ")).map(word => word.toLowerCase)

}

def main(args: Array[String]): Unit = {

val sparkConf = new SparkConf().setAppName("WordCounter").setMaster("local")

val sc = new SparkContext(sparkConf)

val url: String = "/Users/chengrongxin/Downloads/第一次作业DDL2020年09月14日/TASK1/test.txt" // 作业的test.txt文件路径

val word\_pool: RDD[(String, Int)] = getWords(sc, url).map(word => (word, 1)). //将单词转换为(word, nums)的键值对

reduceByKey(\_+\_).sortBy(\_.\_2)// 相同的单词，次数相加，并按照单词出现频数排序（升）

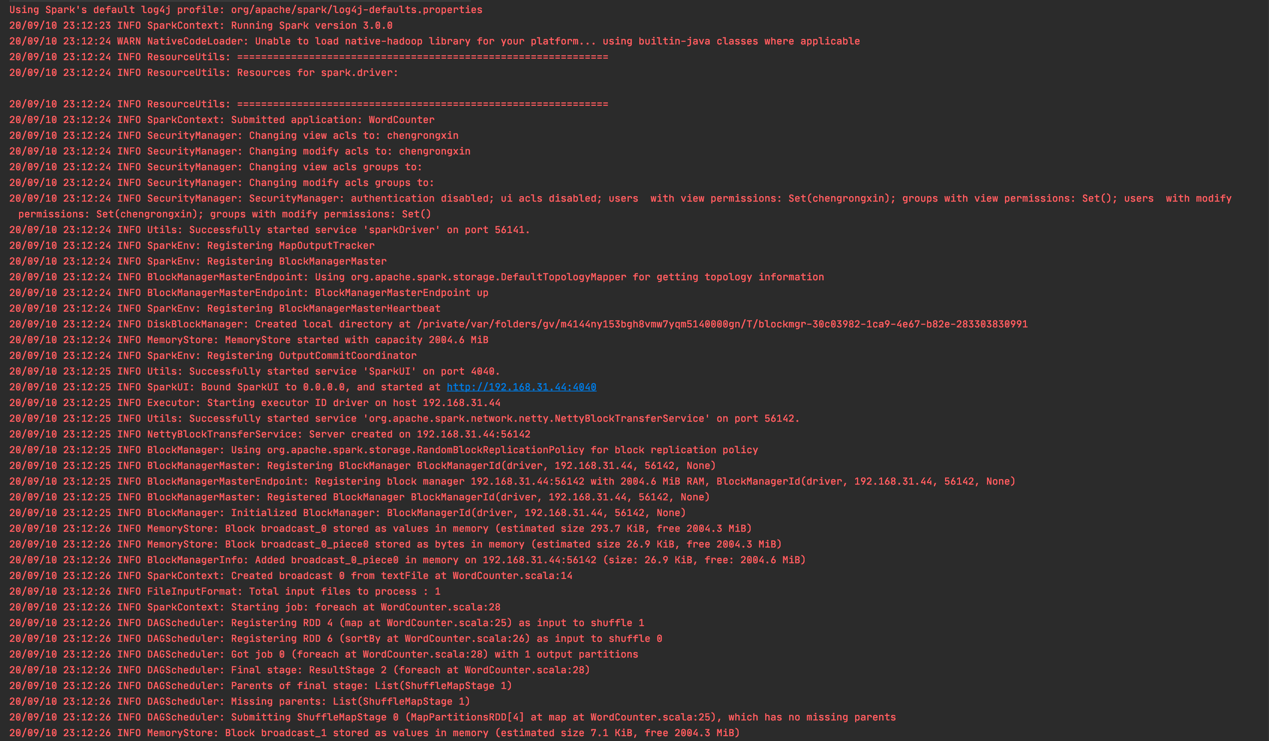
word\_pool.foreach(word => println(word.\_1 + ":" + word.\_2)) // 输出

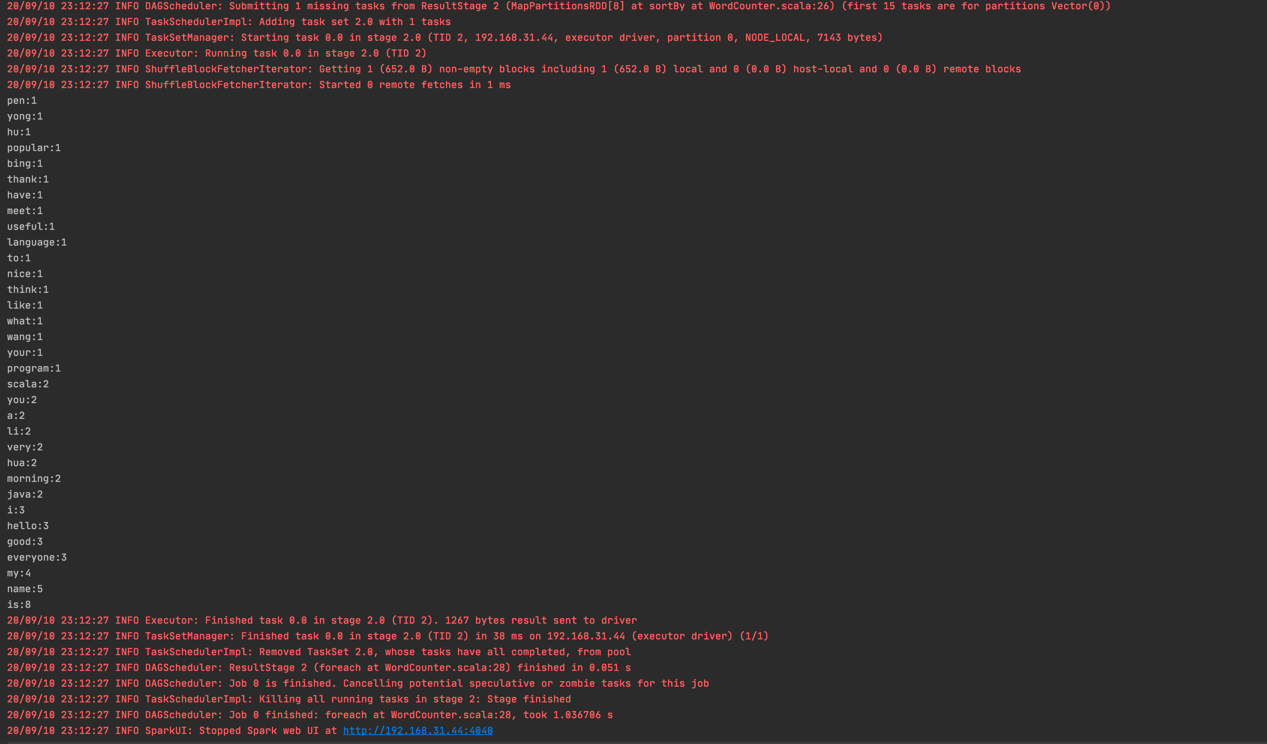
sc.stop()

}

}

**运行结果**

****



具体结果为：

pen:1

yong:1

hu:1

popular:1

bing:1

thank:1

have:1

meet:1

useful:1

language:1

to:1

nice:1

think:1

like:1

what:1

wang:1

your:1

program:1

scala:2

you:2

a:2

li:2

very:2

hua:2

morning:2

java:2

i:3

hello:3

good:3

everyone:3

my:4

name:5

is:8

## 实现思路

1. 按照作业要求，使用Scala语言实现，同时应用Spark RDD编程
2. 单词计数问题很简单，建立（单词，次数）的键值对即可
   1. 先使用map操作建立(word, 1)键值对
   2. 随后使用reduce操作，将相同键的键值对次数合并
3. 最后，将得到的结果排序输出即可

具体代码中，我首先使用了flatMap(word=>word.split(“ ”)函数，将test.txt中的内容按照空格分割并统一转化为小写，随后用map()函数建立映射表，使用更适用于(key, value)模式的reduceByKey(\_+\_)函数实现reduce过程，最后再调用sortBy排序输出结果。

## 实验心得

这是我第一次接触Scala语言，简单熟悉Scala语法大概花了2个多小时，认识到这门语言有不少语法糖，使用起来极为方便，我本来好奇大数据分析为什么不用Python，现在发现Scala方便程度与 Python不相上下。

第一次使用还是踩了坑的，我一开始下载的是最新的2.13.3版本，但是本地会报一个“循环继承”的错，查百度，换版本（2.12+），解决。

同时，也是第一次使用Spark RDD，我个人认为RDD就类似Python中的numpy或pandas，本质还是数组，只不过做了一些优化和扩充。

总体来说第一次作业不难，达到了熟悉Scala、Spark RDD编程的目标。