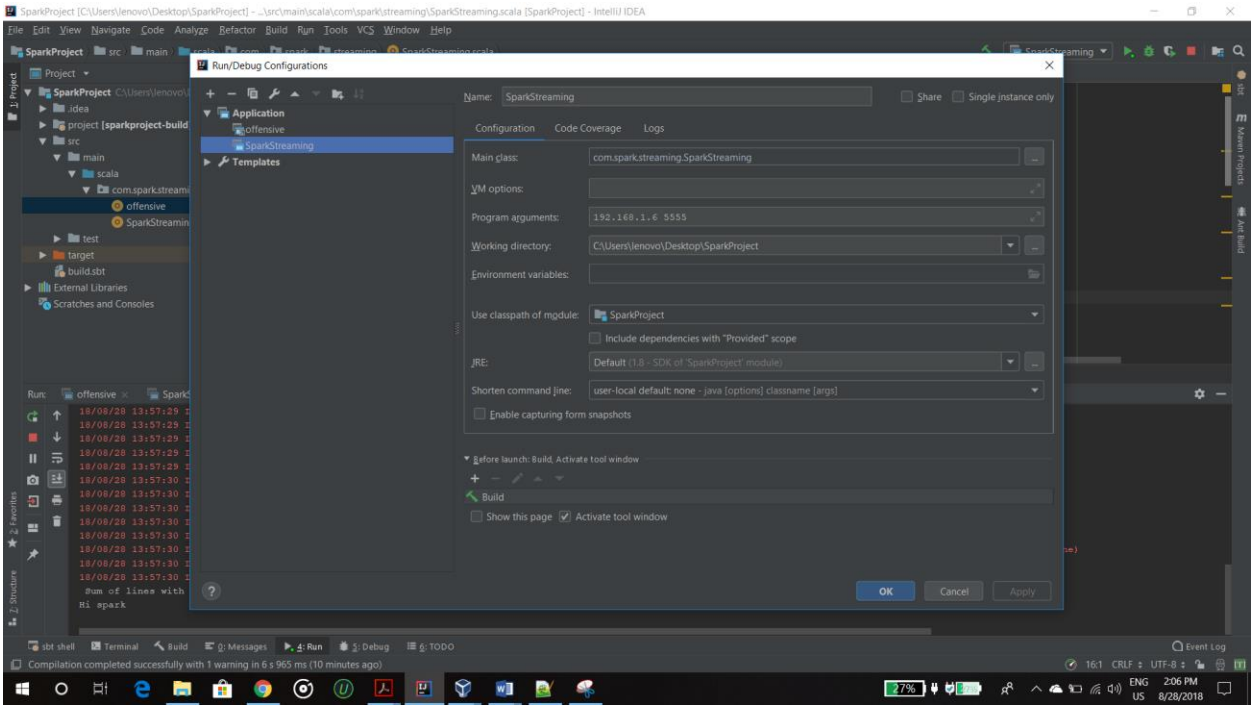
Session 26:
SPARK STREAMING

Task 1

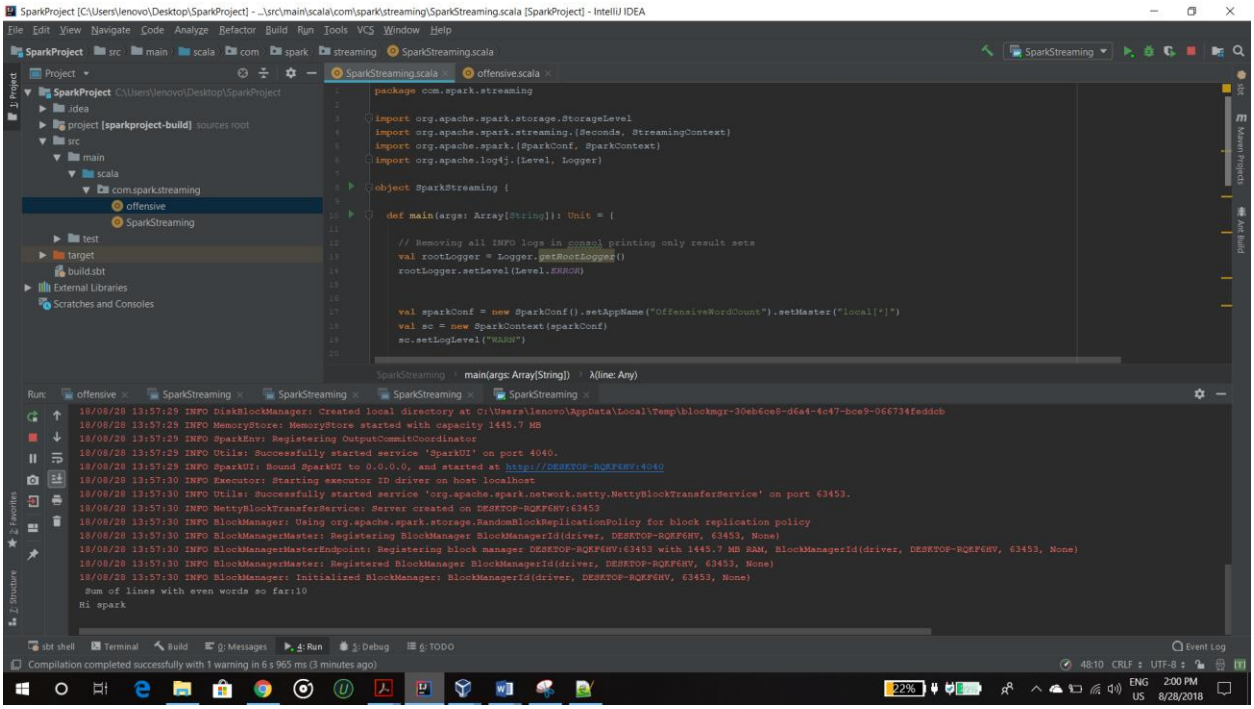
Read a stream of Strings, fetch the words which can be converted to numbers. Filter out the rows, where the sum of numbers in that line is odd.
Provide the sum of all the remaining numbers in that batch

I have executed jobs in Intellij & connected to VM from port:



- Here, we have a list of words and associate integer to it in a variable
- "wordtoNumbers"
- Broadcast this variable to a variable called "wordtoBroadcast"
- To match the words with numbers, we write a function "linewordNumbers", which
- takes input as argument and then we split the input with space and broadcast the words to numbers.
- We create a streaming context, which takes the input every 60 seconds.
- We take the input from the netcat with host as localhost and port number 9999
- We find the count of evenlines by taking the sum of the numbers associated with
- each word and then find if the number is even or not. If it is even, we print the
- number of evenlines, if not we just print the inputted data.

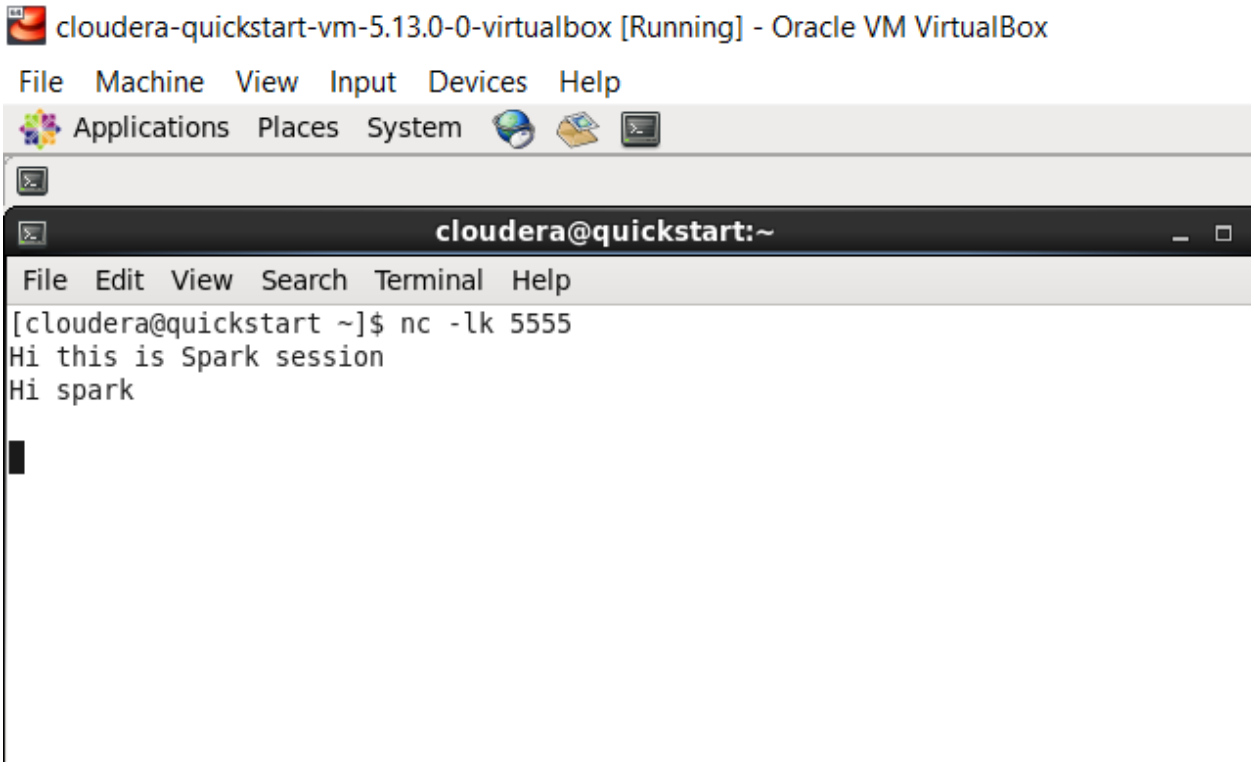
Session 26:
SPARK STREAMING




```scala
package com.spark.streaming

import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.log4j.{Level, Logger}

object SparkStreaming {

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

    // Removing all INFO logs in consol printing only result sets
    val rootLogger = Logger.getRootLogger()
    rootLogger.setLevel(Level.ERROR)


    val sparkConf = new SparkConf().setAppName("OffensiveWordCount").setMaster("local[*]")
    val sc = new SparkContext(sparkConf)
    sc.setLogLevel("WARN")



    val evenlines = sc.accumulator(0)
    val wordtoNumbers = Map("Hi" -> 1, "this" -> 2, "is" -> 3, "Spark" -> 4, "Session" -> 5, "which" -> 6, "Is"
-> 7, "Good" -> 8, "for" -> 9, "Learning" -> 10)
    val wordtoBroadcast = sc.broadcast(wordtoNumbers)

    def linewordNumbers(line: String): Int = {
      var sum:Int = 0
      val words = line.split(" ")
      for (word <- words)
        sum += wordtoBroadcast.value.get(word).getOrElse(0)
      sum
    }


    val ssc = new StreamingContext(sc, Seconds(60))
    val stream = ssc.socketTextStream("192.168.1.6", 5555, StorageLevel.MEMORY_AND_DISK_SER)
    stream.foreachRDD(line => {
      val linestr = line.collect().toList.mkString(" ")
      if (linestr != "") {
```

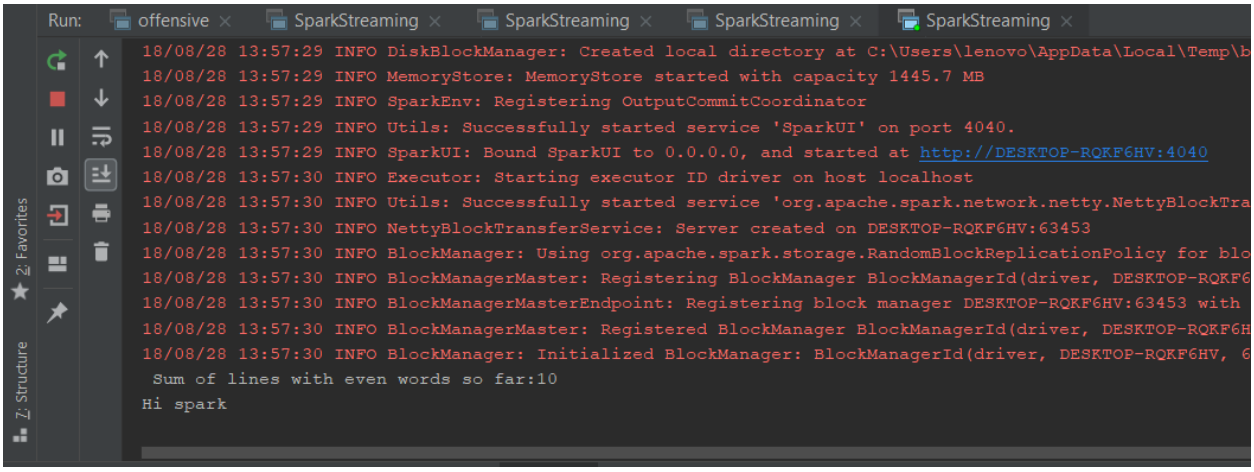
Session 26:
SPARK STREAMING

```
        var numTotlal = linewordNumbers(linestr);
        if (numTotlal %2 == 1){
          println(linestr)
        }
        else {
          evenlines += numTotlal
          println(" Sum of lines with even words so far:" +evenlines.value.toInt)
        }
      }
    })
    ssc.start()
    ssc.awaitTermination()
  }
}
```



- First example is for even Values
- Second example is for ODD value
- Accordingly we got the o/p in Intellij

Session 26:
SPARK STREAMING



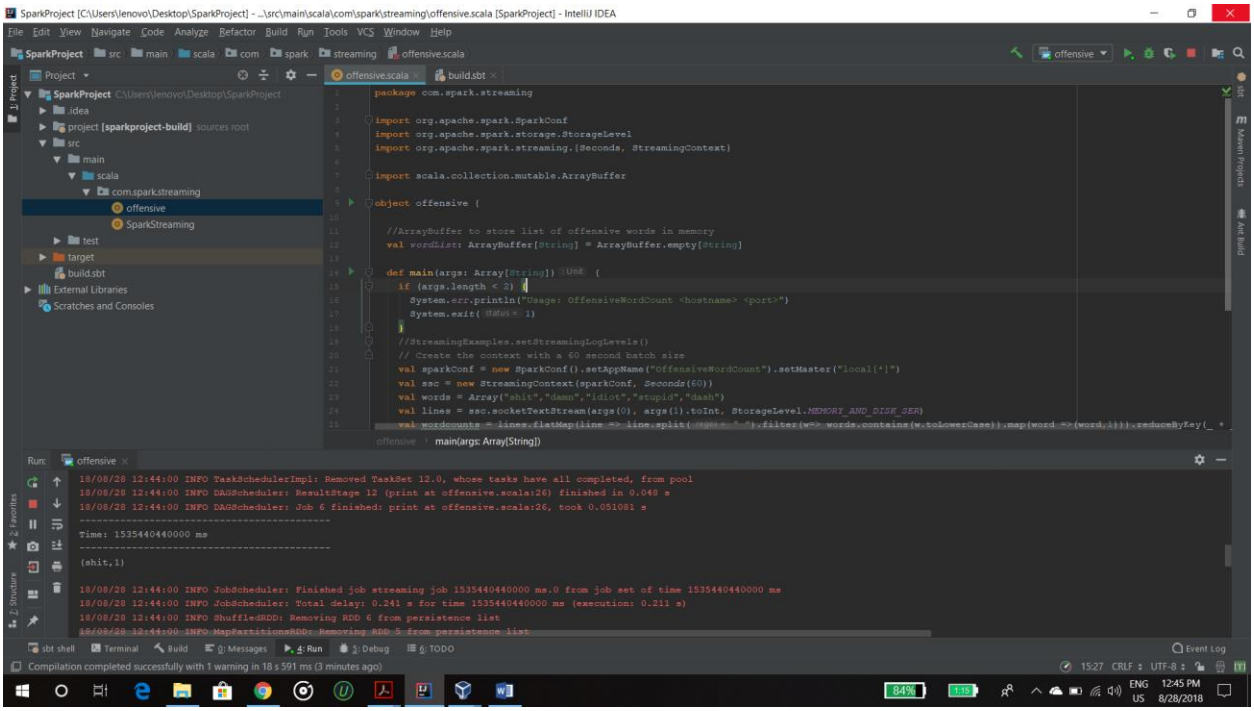
Task 2

Read two streams

1. List of strings input by user
2. Real-time set of offensive words

Find the word count of the offensive words inputted by the user as per the real-time set of offensive Words

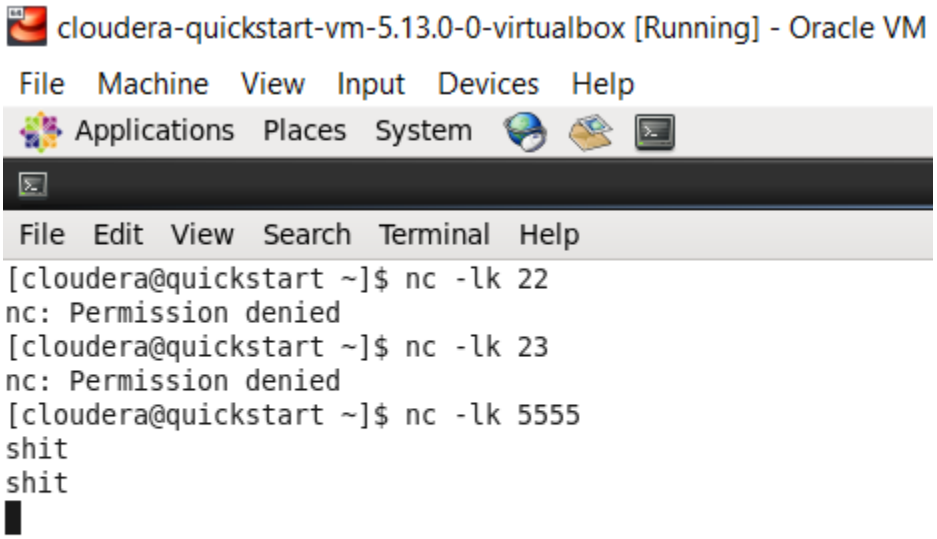
- Set of real time offensive words are defined in Array
- Same data is passed from netcat terminal from VM
- Spark program gave two times output as we entered data two times



Session 26:
SPARK STREAMING

Code:

```
package com.spark.streaming

import org.apache.spark.SparkConf
import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.{Seconds, StreamingContext}

import scala.collection.mutable.ArrayBuffer

object offensive {

  //ArrayBuffer to store list of offensive words in memory
  val wordList: ArrayBuffer[String] = ArrayBuffer.empty[String]

  def main(args: Array[String]) {
    if (args.length < 2) {
      System.err.println("Usage: OffensiveWordCount <hostname> <port>")
      System.exit(1)
    }
    //StreamingExamples.setStreamingLogLevels()
    // Create the context with a 60 second batch size
    val sparkConf = new SparkConf().setAppName("OffensiveWordCount").setMaster("local[*]")
    val ssc = new StreamingContext(sparkConf, Seconds(60))
    val words = Array("shit","damn","idiot","stupid","dash")
    val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY_AND_DISK_SER)
    val wordcounts = lines.flatMap(line => line.split(" ").filter(w=> words.contains(w.toLowerCase)).map(word
=>(word,1))).reduceByKey(_ + _)
    wordcounts.print()

    ssc.start()
    ssc.awaitTermination()
  }

}
```



Session 26:
SPARK STREAMING

```
18/08/28 12:42:00 INFO TaskSetManager: Finished task 1.0
18/08/28 12:42:00 INFO DAGScheduler: ResultStage 4 (prin
18/08/28 12:42:00 INFO TaskSchedulerImpl: Removed TaskSe
18/08/28 12:42:00 INFO DAGScheduler: Job 2 finished: pri
-------------------------------------------
Time: 1535440320000 ms
-------------------------------------------
(shit,1)

18/08/28 12:42:00 INFO JobScheduler: Finished job stream
18/08/28 12:42:00 INFO JobScheduler: Total delay: 0.731
```

```
18/08/28 12:44:00 INFO TaskSchedulerImpl: Removed TaskSet 12.0, whose tasks have all completed, from p
18/08/28 12:44:00 INFO DAGScheduler: ResultStage 12 (print at offensive.scala:26) finished in 0.048 s
18/08/28 12:44:00 INFO DAGScheduler: Job 6 finished: print at offensive.scala:26, took 0.051081 s
-------------------------------------------
Time: 1535440440000 ms
-------------------------------------------
(shit,1)

18/08/28 12:44:00 INFO JobScheduler: Finished job streaming job 1535440440000 ms.0 from job set of tim
18/08/28 12:44:00 INFO JobScheduler: Total delay: 0.241 s for time 1535440440000 ms (execution: 0.211
18/08/28 12:44:00 INFO ShuffledRDD: Removing RDD 6 from persistence list
```