Building Map Reduce Application for get null exception count in a log file

I used IntelliJ IDEA to generate the application.

Step 01 :
Create a new project

Step 02:

Add following to pom.file

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.uom.big.data</groupId>
    <artifactId>EMRNullPointerException</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
    </properties>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>3.2.4</version>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <finalName>uber-${artifactId}-${version}</finalName>
                </configuration>
```

```
        </plugin>
      </plugins>
   </build>
   <dependencies>
      <dependency>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <version>4.11</version>
         <scope>test</scope>
      </dependency>
      <dependency>
         <groupId>org.apache.hadoop</groupId>
         <artifactId>hadoop-client</artifactId>
         <version>2.2.0</version>
      </dependency>
   </dependencies>



</project>
```

Step 03:

Create java class "EMRNullPointerExcepCount" as follows

```
package com.uom.big.data;

/**
 * created by shanika prasangika
 *
 */

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class EMRNullPointerExcepCount
{
   public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
      private final static IntWritable one = new IntWritable(1);
```

```
    private Text word = new Text();

     public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
          if(tokenizer.nextToken().contains("NullPointerException")){
            word.set(tokenizer.nextToken());
            context.write(word, one);
          }

        }
    }
  }

  public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
      int nullPointerExceptionCount = 0;
      for (IntWritable val : values) {
        nullPointerExceptionCount += val.get();
      }
      context.write(key, new IntWritable(nullPointerExceptionCount));
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    Job job = new Job(conf, "nullpointerexceptioncount");

    job.setJarByClass(EMRNullPointerExcepCount.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
```

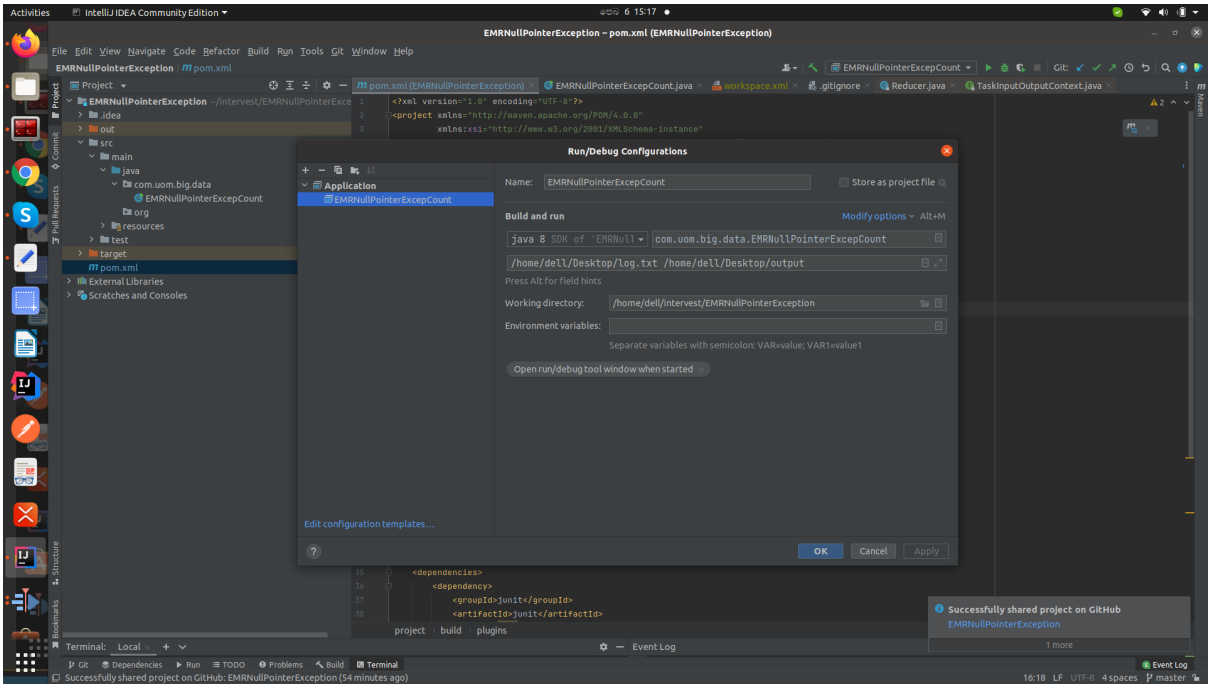
```
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```

Step 04:

Add arguments to the application

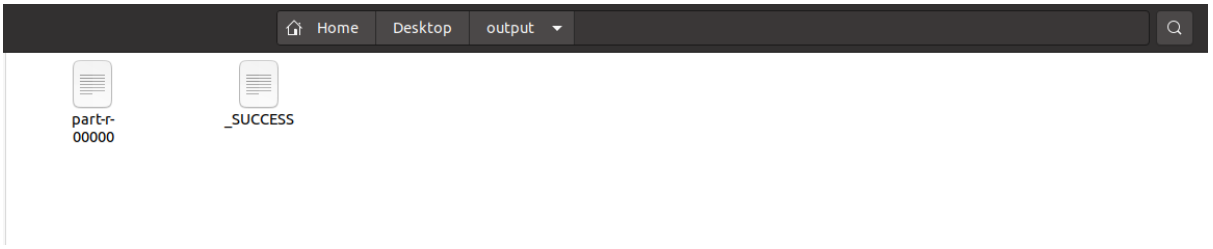
Run -> Edit Configurations and set arguments as follows.

In this example there are two arguments . One is the input file which is log.txt and other is output destination which is output directory.



Step 05 :

Run the project and you can see the output folder in Desktop location. You can see the files inside it.

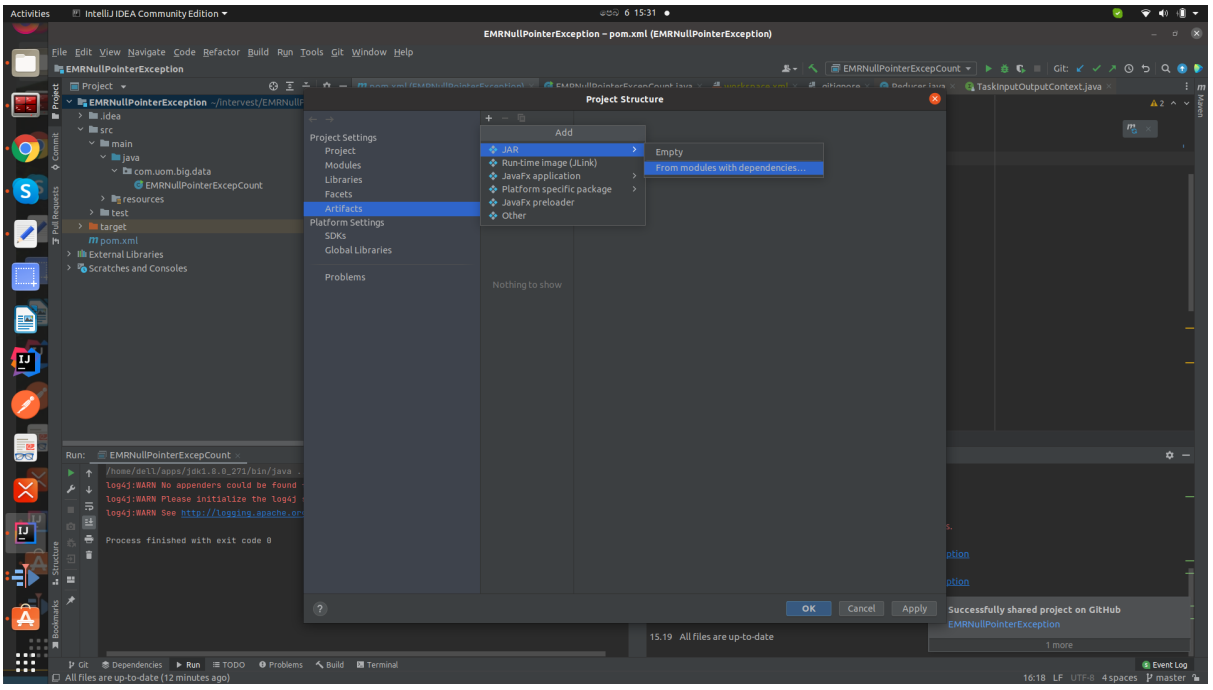


When open the part-f-00000 file you can see the output as there are NullPointerException count as 4.

Export as Executable jar file

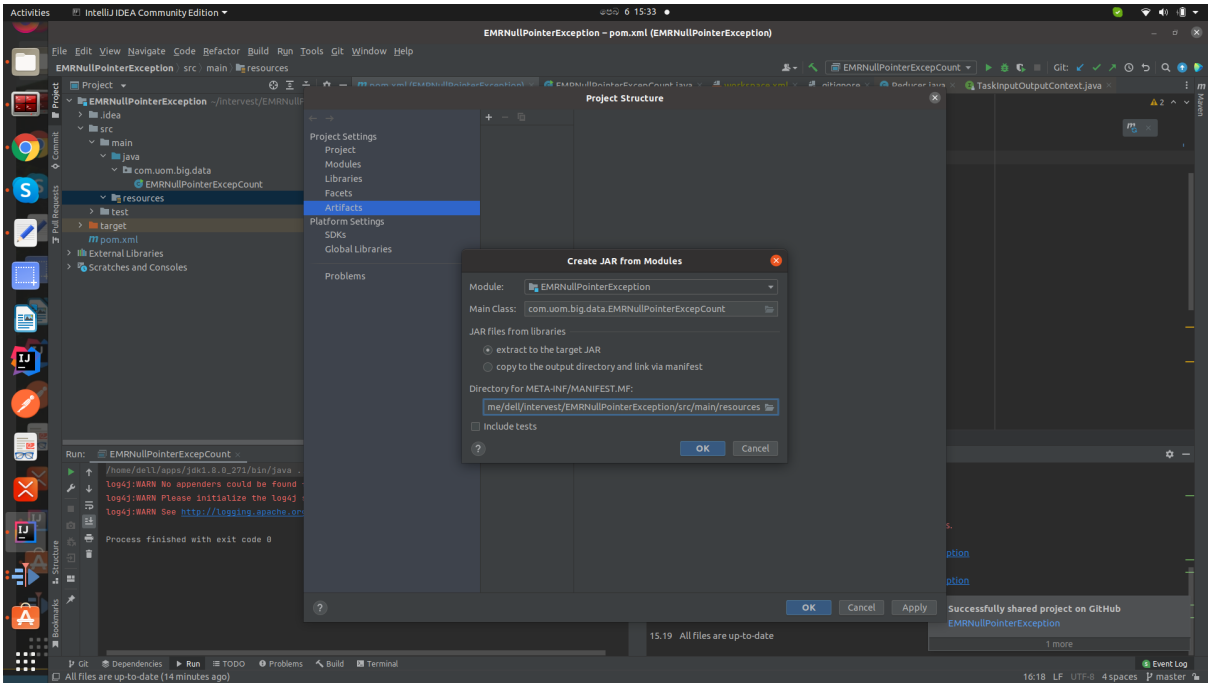
Step 01:

File -> Project Structure->Artifacts and goto following location



Step 02: create jar from modules

Note : please select resource location for MANIFEST.MF file



Step 03: Build -> Build Artifacts -> Build

You can see generated executable jar as follows