**Map:**

1. save as App.java
2. mkdir -m 755 maponly\_classes

* Creates a directory named maponly\_classes with permissions 755 to store the compiled class files

1. javac -classpath $(hadoop classpath) -d maponly\_classes App.java

* Compiles the App.java file using Hadoop's classpath and places the compiled .class files into the maponly\_classes directory.

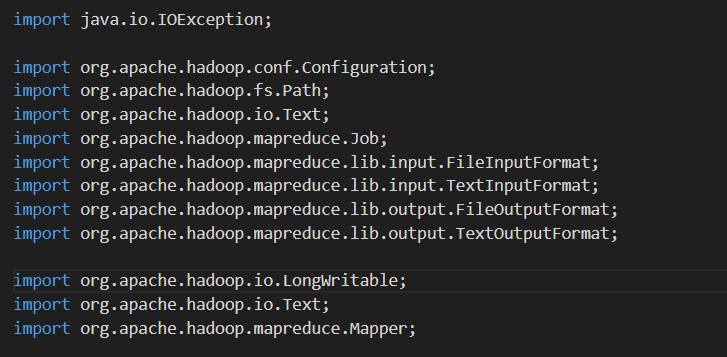
1. jar -cvf ${HOME}/scripts/maponly.jar -C maponly\_classes/ .

* Creates a JAR file (maponly.jar) containing the compiled classes from the maponly\_classes directory.

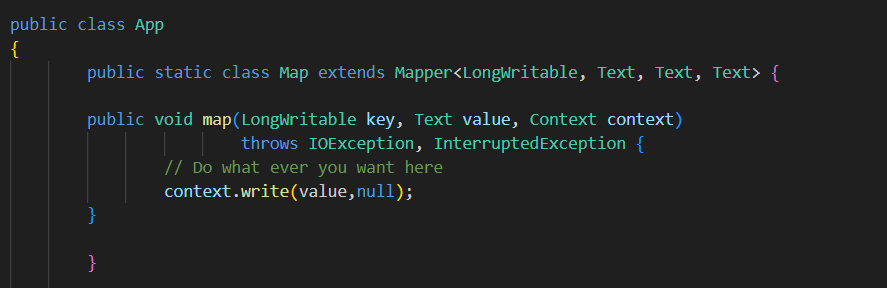
1. hadoop jar ${HOME}/scripts/maponly.jar App /volcheck /mapres.1

* Executes the MapReduce job using the JAR file. App is the class containing the main method, /volcheck is the input path on HDFS, and /mapres.1 is the output path on HDFS.

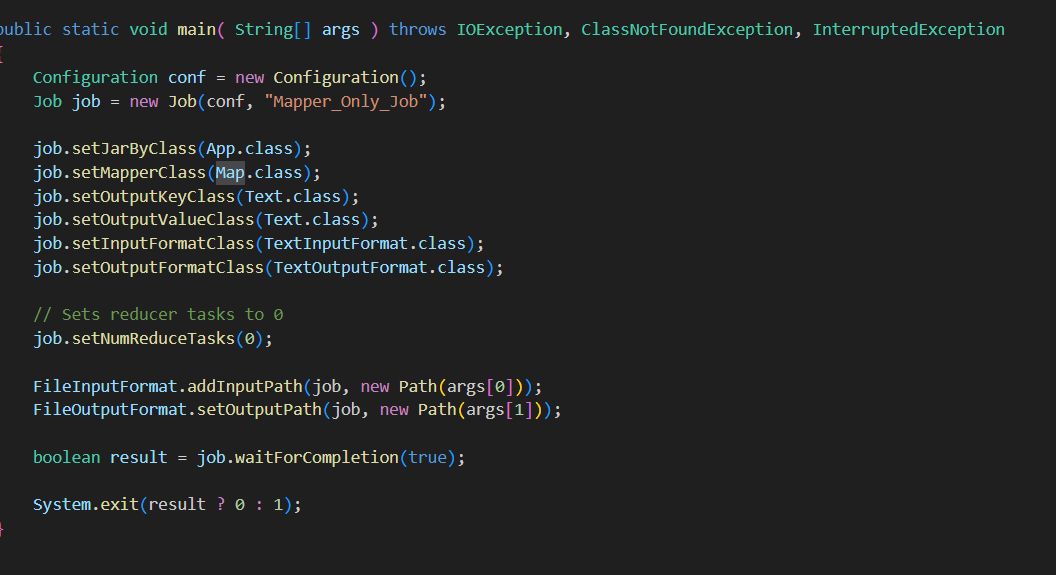
1.Save as APP.java



* These imports include necessary Hadoop classes for I/O operations, MapReduce framework, and configuration.



* Defines a Mapper that processes each input record.
* map Method: Simply writes each input record as the key with a null value. This effectively copies the input to the output without modification.



* setJarByClass: Sets the JAR file containing the job classes.
* setMapperClass: Specifies the mapper class (Map).
* setOutputKeyClass and setOutputValueClass: Define the types for the output of the mapper.
* setInputFormatClass and setOutputFormatClass: Set input and output formats.
* setNumReduceTasks(0): Specifies that no reducer tasks should be used, meaning the job only runs the mapper.
* FileInputFormat.addInputPath and FileOutputFormat.setOutputPath: Set the input and output paths for the job.
* Implements a Hadoop MapReduce job that performs mapping but skips the reduction phase. Each line of the input is output as-is with a null value.
* Set up the environment, compile the code, package it into a JAR file, and run the job.