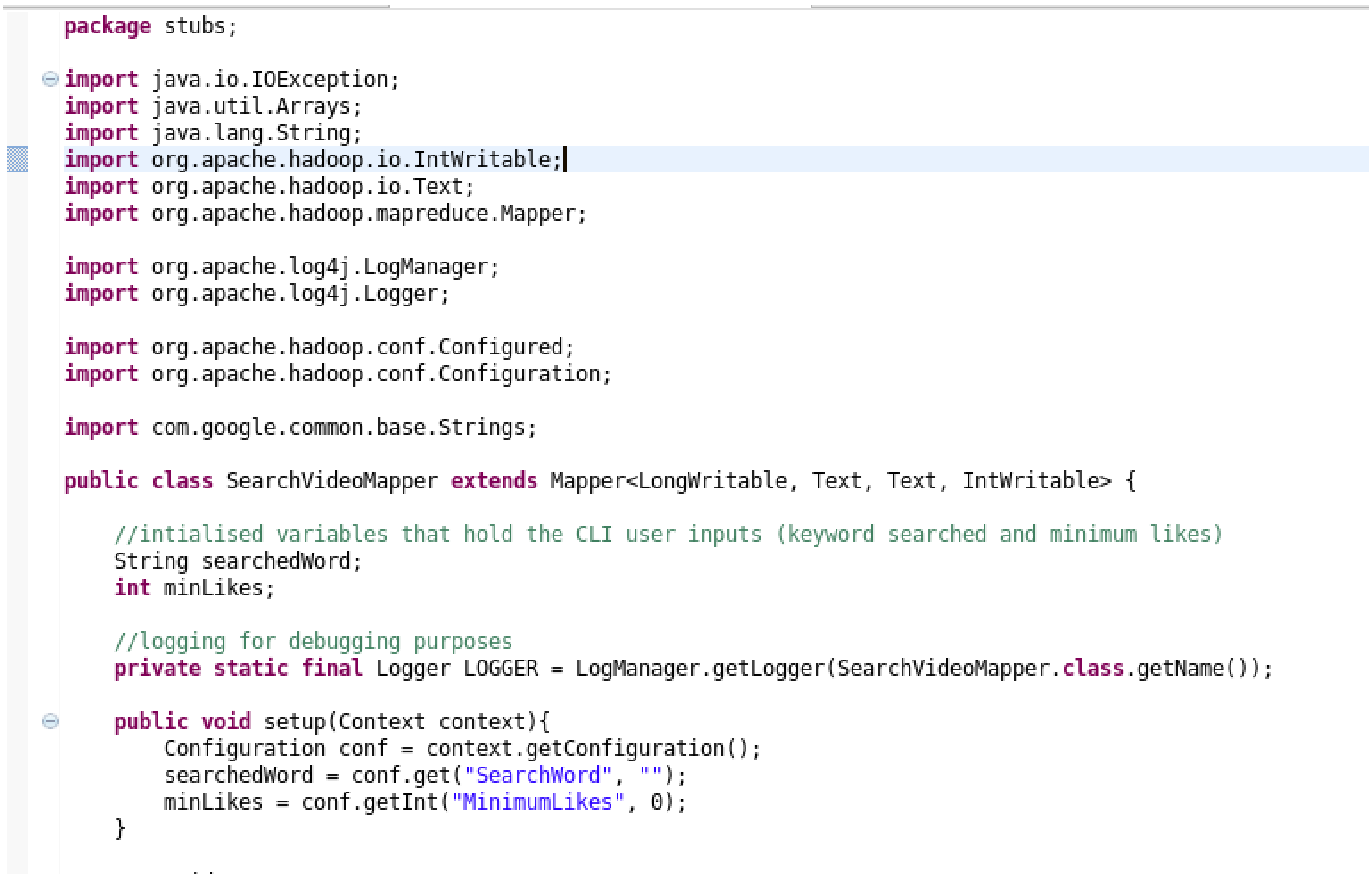
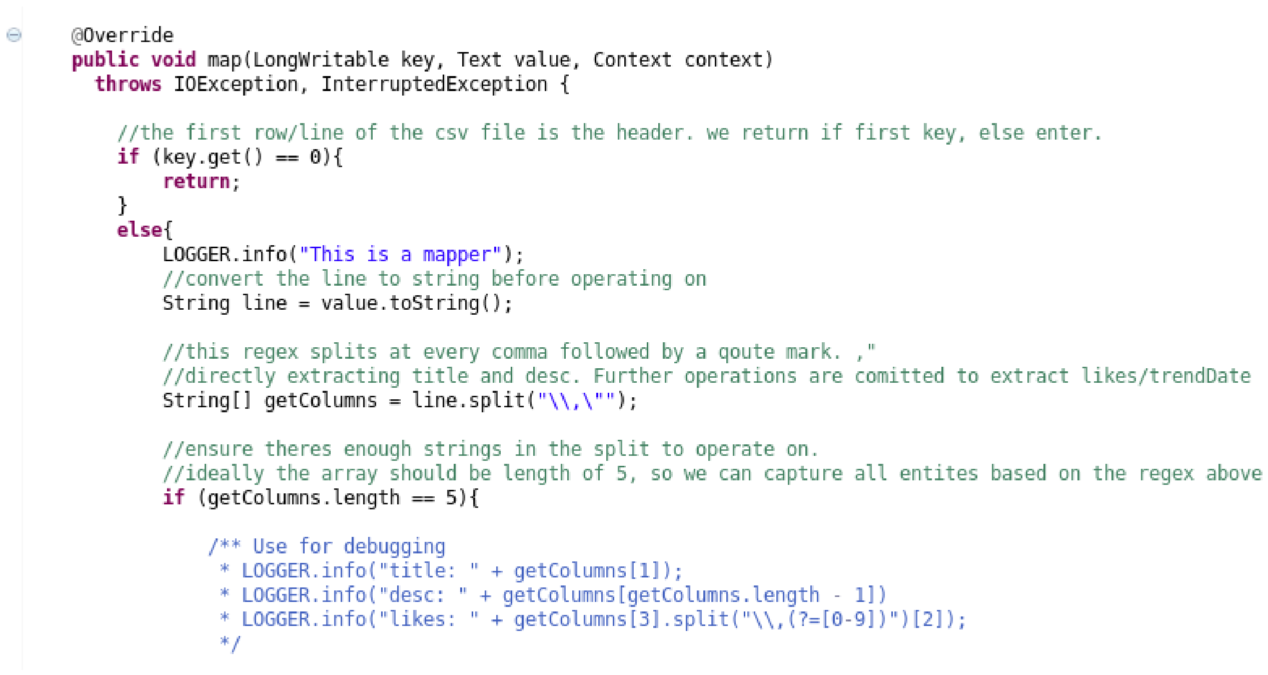


```java
package stubs;

import java.io.IOException;
import java.util.Arrays;
import java.lang.String;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.conf.Configuration;

import com.google.common.base.Strings;

public class SearchVideoMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    //intialised variables that hold the CLI user inputs (keyword searched and minimum likes)
    String searchedWord;
    int minLikes;

    //logging for debugging purposes
    private static final Logger LOGGER = LogManager.getLogger(SearchVideoMapper.class.getName());

    public void setup(Context context){
        Configuration conf = context.getConfiguration();
        searchedWord = conf.get("SearchWord", "");
        minLikes = conf.getInt("MinimumLikes", 0);
    }
```



```java
@Override
public void map(LongWritable key, Text value, Context context)
  throws IOException, InterruptedException {

    //the first row/line of the csv file is the header. we return if first key, else enter.
    if (key.get() == 0){
        return;
    }
    else{
        LOGGER.info("This is a mapper");
        //convert the line to string before operating on
        String line = value.toString();

        //this regex splits at every comma followed by a qoute mark. ,"
        //directly extracting title and desc. Further operations are comitted to extract likes/trendDate
        String[] getColumns = line.split("\\,\"");

        //ensure theres enough strings in the split to operate on.
        //ideally the array should be length of 5, so we can capture all entites based on the regex above
        if (getColumns.length == 5){

            /** Use for debugging
             * LOGGER.info("title: " + getColumns[1]);
             * LOGGER.info("desc: " + getColumns[getColumns.length - 1])
             * LOGGER.info("likes: " + getColumns[3].split("\\,(?=[0-9])")[2]);
             */
```

```
                 */

                //description position tends to be at the end.
                //convert to lowercase (easier to compare with searchedWord)
                //both title and desc have an unnecessary closing quote mark due to regex split. this is replaced
                String desc = getColumns[getColumns.length - 1].replace("\"", "").toLowerCase();
                String title = getColumns[1].replace("\"", "");
                //likes are in the 3rd index of the getColumns regex. other info is present
                //so, another regex split is committed within the 3rd index to extract likes amount
                //it is then parsed as integer to perform quantity based operation in the next if cond.
                int likes = Integer.parseInt(getColumns[3].split("\\,(?=[0-9])")[2]);
                //trend data is the 0 index in getColumns. other unnecessary info is present.
                //is further split at comma and trend date is the 2nd element in that array.
                String trendDate = getColumns[0].split("\\,")[1];
                //search in CLI is included here and convert to lowercase (comparable to desc)
                boolean isSearch = desc.contains(searchedWord.toLowerCase());

                //TASK 2. check if theres video with no desc.
                if (Strings.isNullOrEmpty(desc)){
                    LOGGER.error("missing video description: " + trendDate + " - " + title );
                }
                //finally check if criteria is met and map key, value, ready to use by reducer.
                if (isSearch && likes >= minLikes){
                    context.write(new Text(title), new IntWritable(1));
                }

            }


        }

    }
```