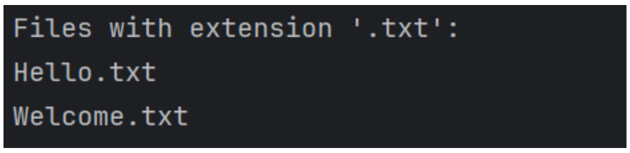
1 Write a Java program to get files with specific extension from a specified folder.

```
import java.io.File;
import java.io.FilenameFilter;

public class Main {

    public static void main(String[] args) {

        String folderPath = "C:\\Users\\shobi\\HelloWorld\\src\\Sample";
        String fileExtension = ".txt";

        File[] filteredFiles = getFilesWithExtension(folderPath, fileExtension);

        System.out.println("Files with extension '" + fileExtension + "':");
        for (File file : filteredFiles) {
            System.out.println(file.getName());
        }
    }

    private static File[] getFilesWithExtension(String folderPath, final String extension) {
        File folder = new File(folderPath);


        FilenameFilter filter = new FilenameFilter() {
            @Override
            public boolean accept(File dir, String name) {
                return name.endsWith(extension);
            }
        };

        return folder.listFiles(filter);
    }
}
```



```
Files with extension '.txt':
Hello.txt
Welcome.txt
```

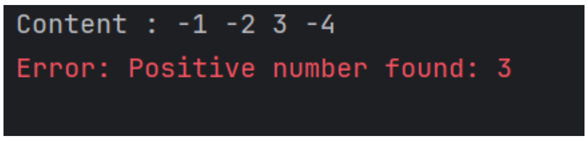
2 Write a Java program that reads a list of numbers from a file and throws an exception if any of the numbers are positive.

Output:
Content of test.txt: -1 -2 -3 4
Error: Positive number found: 4

```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;
```

```
public class Main{
      public static void main(String[] args) {
       try
       {
            String filePath = "C:\\Users\\shobi\\HelloWorld\\src\\Sample\\Hello.txt";
            checkForPositiveNumbers(filePath);
       }
      catch (IOException e)
      {
           System.err.println("Error reading the file: " + e.getMessage());
      }
     catch (PositiveNumberException e)
     {
           System.err.println("Error: " + e.getMessage());
       }
    }
private static void checkForPositiveNumber(String filePath) throws IOException,PositiveNumberException
{
       try (BufferedReader reader = new BufferedReader(new FileReader(filePath)))
       {
           String line = reader.readLine();
           if (line != null)
           {
               System.out.println("Content " + ": " + line);
               String[] numbers = line.split("\\s+");

               for (String numStr : numbers) {
                   int num = Integer.parseInt(numStr);
                   if (num > 0) {
                       throw new PositiveNumberException("Positive number found: " + num);
                   }
               }
           System.out.println("No positive numbers found.");
           }
```

```
        else {
            System.out.println("File is empty.");
          }
        }
    }
}

class PositiveNumberException extends Exception {
    public PositiveNumberException(String message) {
        super(message);
    }
}
```



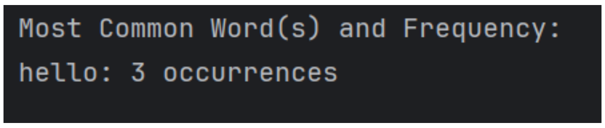
```
Content : -1 -2 3 -4
Error: Positive number found: 3
```

3 You are given a directory path that contains a number of text files. Each text file contains words separated by spaces.

Your task is to write a Java program that finds the most common word across all the files. Consider a word as a sequence of characters separated by spaces. Ignore case sensitivity, meaning "hello" and "Hello" should be considered the same word.

Write a Java program that takes the directory path as input and outputs the most common word along with its frequency. If there are multiple words with the same highest frequency, output all of them.

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class Main {

    public static void main(String[] args) {
        try {

            String directoryPath = "C:\\Users\\shobi\\HelloWorld\\src\\Sample";
            Map<String, Integer> wordFrequencyMap = findMostCommonWord(directoryPath);
            System.out.println("Most Common Word(s) and Frequency:");
            for (Map.Entry<String, Integer> entry : wordFrequencyMap.entrySet()) {
                System.out.println(entry.getKey() + ": " + entry.getValue() + " occurrences");
            }
        } catch (IOException e) {
            System.err.println("Error reading files: " + e.getMessage());
        }
    }

    private static Map<String, Integer> findMostCommonWord(String directoryPath) throws
IOException {
        Map<String, Integer> wordFrequencyMap = new HashMap<>();


        File directory = new File(directoryPath);
        File[] files = directory.listFiles((dir, name) -> name.toLowerCase().endsWith(".txt"));

        if (files != null) {

            for (File file : files) {
                Path filePath = file.toPath();

                Files.lines(filePath)
                        .flatMap(line -> Arrays.stream(line.split("\\s+")))
                        .forEach(word -> {
                            // Convert to lowercase when updating the map, but keep the original case for display
                            wordFrequencyMap.merge(word.toLowerCase(), 1, Integer::sum);
                        });
            }
        }

        return findWordsWithMaxFrequency(wordFrequencyMap);
    }

    private static Map<String, Integer> findWordsWithMaxFrequency(Map<String, Integer>
wordFrequencyMap) {
        Map<String, Integer> maxFrequencyMap = new HashMap<>();
        int maxFrequency = 0;

        for (Map.Entry<String, Integer> entry : wordFrequencyMap.entrySet()) {
            int frequency = entry.getValue();

            if (frequency > maxFrequency) {
                maxFrequency = frequency;
                maxFrequencyMap.clear();
                maxFrequencyMap.put(entry.getKey(), frequency);
            } else if (frequency == maxFrequency) {
                maxFrequencyMap.put(entry.getKey(), frequency);
            }
        }

        return maxFrequencyMap;
    }
}
```



```
Most Common Word(s) and Frequency:
hello: 3 occurrences
```