1. import java.io.File;

```
import java.io.FilenameFilter;

public class FileFilterExample {

    public static void main(String[] args) {
        // Specify the folder path and extension
        String folderPath = "/path/to/your/folder";
        String fileExtension = "txt"; // Change this to the desired file extension

        // Get the list of files with the specified extension
        File folder = new File(folderPath);
        File[] filteredFiles = getFilesWithExtension(folder, fileExtension);

        // Display the list of filtered files
        if (filteredFiles != null && filteredFiles.length > 0) {
            System.out.println("Files with extension ." + fileExtension + " in folder " + folderPath + ":");
            for (File file : filteredFiles) {
                System.out.println(file.getName());
            }
        } else {
            System.out.println("No files with extension ." + fileExtension + " found in folder " +
folderPath);
        }
    }

    private static File[] getFilesWithExtension(File folder, final String extension) {
        // Create a FilenameFilter to filter files by extension
        FilenameFilter filter = new FilenameFilter() {
            @Override
            public boolean accept(File dir, String name) {
                return name.endsWith("." + extension);
            }
        };

        // Use the FilenameFilter to get the list of filtered files
        return folder.listFiles(filter);
    }
}
```

2. import java.io.BufferedReader;

```
import java.io.FileReader;
import java.io.IOException;

public class PositiveNumberChecker {

    public static void main(String[] args) {
        String filePath = "test.txt"; // Replace with the path to your file

        try {
            checkForPositiveNumbers(filePath);
        } catch (PositiveNumberException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("Error reading the file: " + e.getMessage());
        }
    }

    private static void checkForPositiveNumbers(String filePath) throws IOException,
PositiveNumberException {
        try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
            String line = br.readLine();
            System.out.println("Content of " + filePath + ": " + 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);
            }
          }
        }
      }

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

3. import java.io.IOException;

```
import java.nio.file.*;
import java.util.*;

public class MostCommonWordFinder {

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter directory name: ");
    String directoryPath = scanner.nextLine();

    try {
      Map<String, Integer> wordFrequencyMap = getWordFrequencies(directoryPath);
      printMostCommonWords(wordFrequencyMap);
```

```
        } catch (IOException e) {
            System.out.println("Error reading files: " + e.getMessage());
        }
    }

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

        Path directory = Paths.get(directoryPath);
        try (DirectoryStream<Path> directoryStream = Files.newDirectoryStream(directory, "*.txt")) {
            for (Path filePath : directoryStream) {
                List<String> lines = Files.readAllLines(filePath);
                for (String line : lines) {
                    String[] words = line.split("\\s+");
                    for (String word : words) {
                        // Ignore case sensitivity
                        word = word.toLowerCase();

                        // Update word frequency in the map
                        wordFrequencyMap.put(word, wordFrequencyMap.getOrDefault(word, 0) + 1);
                    }
                }
            }
        }

        return wordFrequencyMap;
    }

    private static void printMostCommonWords(Map<String, Integer> wordFrequencyMap) {
        int maxFrequency = 0;
        for (int frequency : wordFrequencyMap.values()) {
            if (frequency > maxFrequency) {
                maxFrequency = frequency;
            }
        }

        System.out.println("Most common words:");

        for (Map.Entry<String, Integer> entry : wordFrequencyMap.entrySet()) {
            if (entry.getValue() == maxFrequency) {
                System.out.println("Word: " + entry.getKey() + ", Frequency: " + entry.getValue());
            }
        }
    }
}
```