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

**CODE:**

import java.io.File;

import java.io.FilenameFilter;

public class FileExtensionFilter {

    public static void main(String[] args) {

        String folderPath = "your\_folder\_path\_here";

        String extension = ".txt"; // Change this to your desired extension

        File folder = new File(folderPath);

        File[] filteredFiles = folder.listFiles(new FilenameFilter() {

            @Override

            public boolean accept(File dir, String name) {

                return name.toLowerCase().endsWith(extension);

            }

        });

        if (filteredFiles != null) {

            System.out.println("Files with extension " + extension + " in folder " + folderPath + ":");

            for (File file : filteredFiles) {

                System.out.println(file.getName());

            }

        } else {

            System.out.println("No files found with extension " + extension + " in folder " + folderPath);

        }

    }

}

**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**

**CODE:**

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class PositiveNumberChecker {

    public static void main(String[] args) {

        try {

            File file = new File("test.txt");

            Scanner scanner = new Scanner(file);

            System.out.print("Content of test.txt: ");

            while (scanner.hasNext()) {

                int num = scanner.nextInt();

                System.out.print(num + " ");

                if (num > 0) {

                    throw new Exception("Error: Positive number found: " + num);

                }

            }

            System.out.println();

            System.out.println("No positive numbers found in the file.");

            scanner.close();

        } catch (FileNotFoundException e) {

            System.out.println("File not found: " + e.getMessage());

        } catch (Exception e) {

            System.out.println(e.getMessage());

        }

    }

}

**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.**

**Input:**

**Enter directory name : TextFolder**

**Output:**

**Word: world, Frequency: 3  
Word: java, Frequency: 2  
Word: hello, Frequency: 2  
Word: is, Frequency: 1  
Word: a, Frequency: 1  
Word: programming, Frequency: 1  
Word: language, Frequency: 1**

**CODE:**

import java.io.\*;

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();

        findMostCommonWord(directoryPath);

    }

    private static void findMostCommonWord(String directoryPath) {

        Map<String, Integer> wordFrequency = new HashMap<>();

        File directory = new File(directoryPath);

        File[] files = directory.listFiles();

        if (files != null) {

            for (File file : files) {

                try (Scanner fileScanner = new Scanner(file)) {

                    while (fileScanner.hasNext()) {

                        String[] words = fileScanner.nextLine().toLowerCase().split("\\s+");

                        for (String word : words) {

                            word = word.replaceAll("[^a-zA-Z]", "");

                            if (!word.isEmpty()) {

                                wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);

                            }

                        }

                    }

                } catch (FileNotFoundException e) {

                    e.printStackTrace();

                }

            }

        }

        int maxFrequency = Collections.max(wordFrequency.values());

        System.out.println("Most common word(s) with frequency " + maxFrequency + ":");

        for (Map.Entry<String, Integer> entry : wordFrequency.entrySet()) {

            if (entry.getValue() == maxFrequency) {

                System.out.println("Word: " + entry.getKey() + ", Frequency: " + entry.getValue());

            }

        }

    }

}