**Problem 1: Write and Read a Text File**

**Problem Statement:**

Write a Java program that writes "Hello, World!" to a file named **hello.txt** and then reads the content from the same file and displays it on the console.

**Solution:**

* Use FileWriter to write data to the file.
* Use FileReader to read the content.

import java.io.\*;

public class WriteReadFile {

public static void main(String[] args) {

String fileName = "hello.txt";

String content = "Hello, World!";

// Writing to a file

try (FileWriter writer = new FileWriter(fileName)) {

writer.write(content);

System.out.println("Data written to file.");

} catch (IOException e) {

e.printStackTrace();

}

// Reading from the file

try (FileReader reader = new FileReader(fileName)) {

int i;

System.out.print("File Content: ");

while ((i = reader.read()) != -1) {

System.out.print((char) i);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Expected Output:**

Data written to file.

File Content: Hello, World!

**Problem 2: Copy Contents of One File to Another**

**Problem Statement:**

Write a Java program that **copies the content** of source.txt to destination.txt.

**Solution:**

* Use FileInputStream to read bytes from source.txt.
* Use FileOutputStream to write bytes to destination.txt.

import java.io.\*;

public class FileCopy {

public static void main(String[] args) {

String sourceFile = "source.txt";

String destinationFile = "destination.txt";

try (FileInputStream fis = new FileInputStream(sourceFile);

FileOutputStream fos = new FileOutputStream(destinationFile)) {

int byteContent;

while ((byteContent = fis.read()) != -1) {

fos.write(byteContent);

}

System.out.println("File copied successfully.");

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Expected Output:**

File copied successfully.

**Problem 3:** Write a java program that reads a file name from the user, and then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes.

### ****Check File Information****

import java.io.File;

import java.util.Scanner;

public class FileInfo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Get file name from user

System.out.print("Enter file name (with path if needed): ");

String fileName = scanner.nextLine();

// Create a File object

File file = new File(fileName);

// Check if file exists

if (file.exists()) {

System.out.println("File exists: Yes");

System.out.println("Readable: " + (file.canRead() ? "Yes" : "No"));

System.out.println("Writable: " + (file.canWrite() ? "Yes" : "No"));

// Get file extension

String fileType = getFileExtension(file);

System.out.println("File Type: " + (fileType.isEmpty() ? "Unknown" : fileType));

// Get file size

System.out.println("File Size: " + file.length() + " bytes");

} else {

System.out.println("File does not exist.");

}

scanner.close();

}

// Method to extract file extension

private static String getFileExtension(File file) {

String name = file.getName();

int lastIndex = name.lastIndexOf(".");

return (lastIndex == -1) ? "" : name.substring(lastIndex + 1);

}

}

### ****Example Run & Output****

#### ****Case 1: File exists (****document.txt ****exists)****

Enter file name (with path if needed): document.txt

File exists: Yes

Readable: Yes

Writable: Yes

File Type: txt

File Size: 1024 bytes

(if *document.txt* has 1024 bytes of content)

#### ****Case 2: File does not exist****

Enter file name (with path if needed): unknownfile.txt

File does not exist.

### ****Explanation:****

1. The user enters a **file name**.
2. The program **checks** if the file exists.
3. If it exists:
   * It checks if the file is **readable** and **writable**.
   * It extracts and prints the **file type (extension)**.
   * It prints the **file size in bytes**.
4. If the file does not exist, it informs the user.

**Problem 4: Count Number of Words in a File**

**Problem Statement:**

Write a Java program that counts the **number of words** in a given text file.

**Solution:**

* Use BufferedReader to read the file line by line.
* Split the content into words using split("\\s+").

import java.io.\*;

public class WordCount {

public static void main(String[] args) {

String fileName = "sample.txt";

int wordCount = 0;

try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

String line;

while ((line = reader.readLine()) != null) {

String[] words = line.split("\\s+");

wordCount += words.length;

}

System.out.println("Total Words: " + wordCount);

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Expected Output:**

(If sample.txt contains "Java is fun and powerful")

Total Words: 5

**Problem 5: Read User Input and Write to File**

**Problem Statement:**

Write a Java program that takes user input from the console and writes it to a file.

**Solution:**

* Use BufferedReader to read user input.
* Use BufferedWriter to write the input to a file.

import java.io.\*;

public class UserInputToFile {

public static void main(String[] args) {

String fileName = "userInput.txt";

try (BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {

System.out.print("Enter text to save in file: ");

String input = reader.readLine();

writer.write(input);

System.out.println("Data saved successfully.");

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Expected Output (User enters "Hello Java!")**

Enter text to save in file: Hello Java!

Data saved successfully.

*(Content of userInput.txt will be: Hello Java!)*

**Problem 6: Reverse Content of a File**

**Problem Statement:**

Write a Java program that reads a file and prints its content in **reverse order**.

**Solution:**

* Use BufferedReader to read the file.
* Store lines in a Stack and print them in reverse order.

import java.io.\*;

import java.util.Stack;

public class ReverseFileContent {

public static void main(String[] args) {

String fileName = "textfile.txt";

Stack<String> stack = new Stack<>();

try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

String line;

while ((line = reader.readLine()) != null) {

stack.push(line);

}

System.out.println("Reversed Content:");

while (!stack.isEmpty()) {

System.out.println(stack.pop());

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Expected Output:**

(If textfile.txt contains)

Line 1

Line 2

Line 3

**Output:**

Reversed Content:

Line 3

Line 2

Line 1

**Problem 7:** Write a java program that displays the number of characters, lines and words in a text file.

Java program that displays the number of characters, lines, and words in a text file, along with a step-by-step explanation and an example of the output.

**Java Program: Count Characters, Words, and Lines in a File**

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class FileAnalyzer {

public static void main(String[] args) {

// Replace with your actual file path or file name

String filePath = "sample.txt";

int lineCount = 0;

int wordCount = 0;

int charCount = 0;

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

String line;

while ((line = reader.readLine()) != null) {

lineCount++; // Count the current line

// Count characters (includes spaces but not newline characters)

charCount += line.length();

// Split line by whitespace to get words

String[] words = line.trim().split("\\s+");

// Avoid counting empty lines

if (!line.trim().isEmpty()) {

wordCount += words.length;

}

}

// Output the results

System.out.println("File: " + filePath);

System.out.println("Total Lines: " + lineCount);

System.out.println("Total Words: " + wordCount);

System.out.println("Total Characters: " + charCount);

} catch (IOException e) {

System.out.println("An error occurred while reading the file.");

e.printStackTrace();

}

}

}

**Explanation**

* BufferedReader and FileReader are used to read the file line by line.
* lineCount++ increments the line count for each line read.
* charCount += line.length() adds the number of characters in the current line (excluding newline characters).
* line.trim().split("\\s+") splits the line into words by whitespace.
* wordCount += words.length adds the number of words in each line.

**Sample Input File (sample.txt)**

Hello world!

This is a sample text file.

It contains multiple lines.

**Expected Output**

File: sample.txt

Total Lines: 3

Total Words: 11

Total Characters: 68