Java I/O File Handling -

* 1. Write a program to create a new text file named test.txt.

Program:

* import java.io.FileWriter;
* import java.io.IOException;
* public class CreateFile {
* public static void main(String[] args) {
* String fileName = "test.txt";
* try (FileWriter writer = new FileWriter(fileName)) {
* System.out.println("File '" + fileName + "' created successfully.");
* } catch (IOException e) {
* System.err.println("An error occurred while creating the file: " + e.getMessage());
* }
* }
* }
* 2. Write a program to check whether a file exists at a given path.

Program:

* import java.io.File;
* public class CheckFileExists {
* public static void main(String[] args) {
* String filePath = "test.txt"; // Check for the file created in the previous example
* File file = new File(filePath);
* if (file.exists()) {
* System.out.println("File '" + filePath + "' exists.");
* } else {
* System.out.println("File '" + filePath + "' does not exist.");
* }
* }

}

* 3. Write a Java program to write "Hello, World!" into a file using FileWriter.

Program:

* import java.io.FileWriter;
* import java.io.IOException;
* public class WriteToFile {
* public static void main(String[] args) {
* String fileName = "output.txt";
* String content = "Hello, World!";
* try (FileWriter writer = new FileWriter(fileName)) {
* writer.write(content);
* System.out.println("Successfully wrote '" + content + "' to " + fileName);
* } catch (IOException e) {
* System.err.println("An error occurred while writing to the file: " + e.getMessage());
* }
* }

}

* 4. Write a program to read the content of a file line by line using BufferedReader.
* Program:
* import java.io.BufferedReader;
* import java.io.FileReader;
* import java.io.IOException;
* public class ReadFileLineByLine {
* public static void main(String[] args) {
* String fileName = "output.txt"; // Read the file created in the previous example
* try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
* String line;
* System.out.println("Content of '" + fileName + "':");
* while ((line = reader.readLine()) != null) {
* System.out.println(line);
* }
* } catch (IOException e) {
* System.err.println("An error occurred while reading the file: " + e.getMessage());
* }
* }
* }
* 5. Write a program to append a line of text to an existing file.
* Program:
* import java.io.FileWriter;
* import java.io.IOException;
* public class AppendToFile {
* public static void main(String[] args) {
* String fileName = "output.txt";
* String lineToAppend = "This is a new line.";
* try (FileWriter writer = new FileWriter(fileName, true)) { // true for append mode
* writer.write("\n" + lineToAppend); // Add a newline before appending
* System.out.println("Successfully appended '" + lineToAppend + "' to " + fileName);
* } catch (IOException e) {
* System.err.println("An error occurred while appending to the file: " + e.getMessage());
* }
* }

}

* 6. Write a program to count the number of lines, words, and characters in a file.
* Program:
* import java.io.BufferedReader;
* import java.io.FileReader;
* import java.io.IOException;
* public class FileMetrics {
* public static void main(String[] args) {
* String fileName = "output.txt"; // Use a file with some content for better testing
* int lineCount = 0;
* int wordCount = 0;
* int charCount = 0;
* try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
* String line;
* while ((line = reader.readLine()) != null) {
* lineCount++;
* charCount += line.length(); // Count characters in the line (excluding newline itself)
* // Split by whitespace to count words
* String[] words = line.trim().split("\\s+");
* if (!line.trim().isEmpty()) { // Avoid counting empty lines as having one word
* wordCount += words.length;
* }
* }
* System.out.println("File: " + fileName);
* System.out.println("Lines: " + lineCount);
* System.out.println("Words: " + wordCount);
* System.out.println("Characters (excluding newlines): " + charCount);
* } catch (IOException e) {
* System.err.println("An error occurred while reading the file: " + e.getMessage());
* }
* }
* }
* 7. Write a program to copy content from one file to another using FileReader and FileWriter.

Program:

* import java.io.FileReader;
* import java.io.FileWriter;
* import java.io.IOException;
* public class FileCopier {
* public static void main(String[] args) {
* String sourceFile = "source.txt"; // Make sure this file exists with some content
* String destinationFile = "destination.txt";
* try (FileReader reader = new FileReader(sourceFile);
* FileWriter writer = new FileWriter(destinationFile)) {
* int character;
* while ((character = reader.read()) != -1) {
* writer.write(character);
* }
* System.out.println("Content successfully copied from '" + sourceFile + "' to '" + destinationFile + "'.");
* } catch (IOException e) {
* System.err.println("An error occurred during file copying: " + e.getMessage());
* }
* }

}

* 8. Write a program that lists all the files in a directory.

Program:

* import java.io.File;
* import java.util.Scanner;
* public class ListDirectoryContents {
* public static void main(String[] args) {
* Scanner scanner = new Scanner(System.in);
* System.out.print("Enter the directory path to list: ");
* String directoryPath = scanner.nextLine();
* File directory = new File(directoryPath);
* if (!directory.exists()) {
* System.out.println("Error: Directory does not exist.");
* } else if (!directory.isDirectory()) {
* System.out.println("Error: The provided path is not a directory.");
* } else {
* File[] filesAndDirs = directory.listFiles(); // Get all files and directories
* if (filesAndDirs != null) {
* System.out.println("\nContents of directory '" + directoryPath + "':");
* for (File item : filesAndDirs) {
* if (item.isFile()) {
* System.out.println("FILE: " + item.getName());
* } else if (item.isDirectory()) {
* System.out.println("DIR: " + item.getName());
* }
* }
* } else {
* System.out.println("Could not list contents of the directory.");
* }
* }
* scanner.close();
* }

}

* 9. Write a program to filter and display only .txt files from a folder using FilenameFilter.
* Program:
* import java.io.File;
* import java.io.FilenameFilter;
* import java.util.Scanner;
* public class FilterTxtFiles {
* public static void main(String[] args) {
* Scanner scanner = new Scanner(System.in);
* System.out.print("Enter the directory path to filter .txt files: ");
* String directoryPath = scanner.nextLine();
* File directory = new File(directoryPath);
* if (!directory.exists() || !directory.isDirectory()) {
* System.out.println("Error: Invalid directory path.");
* scanner.close();
* return;
* }
* // Create a FilenameFilter to accept only .txt files
* FilenameFilter textFileFilter = new FilenameFilter() {
* @Override
* public boolean accept(File dir, String name) {
* return name.toLowerCase().endsWith(".txt");
* }
* };
* File[] txtFiles = directory.listFiles(textFileFilter);
* if (txtFiles != null && txtFiles.length > 0) {
* System.out.println("\n.txt files in '" + directoryPath + "':");
* for (File file : txtFiles) {
* System.out.println(file.getName());
* }
* } else {
* System.out.println("No .txt files found in '" + directoryPath + "'.");
* }
* scanner.close();
* }
* }
* 10. Write a program to serialize and deserialize a Student object to and from a file.
* Program:
* import java.io.\*;
* // The class must implement Serializable to be serialized
* class Student implements Serializable {
* private static final long serialVersionUID = 2L; // Recommended for version control
* private int id;
* private String name;
* private double marks;
* public Student(int id, String name, double marks) {
* this.id = id;
* this.name = name;
* this.marks = marks;
* }
* // Getters for accessing data (optional for serialization but good practice)
* public int getId() { return id; }
* public String getName() { return name; }
* public double getMarks() { return marks; }
* @Override
* public String toString() {
* return "Student [id=" + id + ", name=" + name + ", marks=" + marks + "]";
* }
* }
* public class StudentSerializationDemo {
* public static void main(String[] args) {
* String fileName = "student.ser";
* // --- Serialization ---
* Student studentOut = new Student(101, "Jane Doe", 92.5);
* try (FileOutputStream fileOut = new FileOutputStream(fileName);
* ObjectOutputStream out = new ObjectOutputStream(fileOut)) {
* out.writeObject(studentOut);
* System.out.println("Student object serialized to " + fileName);
* } catch (IOException e) {
* System.err.println("Error during serialization: " + e.getMessage());
* }
* // --- Deserialization ---
* Student studentIn = null;
* try (FileInputStream fileIn = new FileInputStream(fileName);
* ObjectInputStream in = new ObjectInputStream(fileIn)) {
* studentIn = (Student) in.readObject();
* System.out.println("Student object deserialized from " + fileName);
* System.out.println("Deserialized Student: " + studentIn);
* } catch (IOException i) {
* System.err.println("Error during deserialization: " + i.getMessage());
* } catch (ClassNotFoundException c) {
* System.err.println("Student class not found during deserialization.");
* c.printStackTrace();
* }
* }
* }
* 11. Write a program to read a file using Scanner and display the tokens.
* Program:
* import java.io.File;
* import java.io.FileNotFoundException;
* import java.util.Scanner;
* public class ReadFileWithScanner {
* public static void main(String[] args) {
* String fileName = "tokens.txt"; // Make sure this file exists with some text
* try (Scanner scanner = new Scanner(new File(fileName))) {
* System.out.println("Tokens from '" + fileName + "':");
* while (scanner.hasNext()) {
* System.out.println(scanner.next()); // Reads the next token (word)
* }
* } catch (FileNotFoundException e) {
* System.err.println("File not found: " + e.getMessage());
* }
* }
* }
* 12. Write a program to search for a specific word in a file and count its occurrences.
* Program:
* import java.io.BufferedReader;
* import java.io.FileReader;
* import java.io.IOException;
* import java.util.Scanner;
* import java.util.regex.Matcher;
* import java.util.regex.Pattern;
* public class WordSearchAndCount {
* public static void main(String[] args) {
* String fileName = "search\_data.txt"; // Create this file with some text
* Scanner scanner = new Scanner(System.in);
* System.out.print("Enter the word to search and count: ");
* String searchWord = scanner.nextLine();
* int occurrenceCount = 0;
* try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
* String line;
* // Compile a regex pattern for whole word matching (case-insensitive)
* Pattern pattern = Pattern.compile("\\b" + Pattern.quote(searchWord) + "\\b", Pattern.CASE\_INSENSITIVE);
* while ((line = reader.readLine()) != null) {
* Matcher matcher = pattern.matcher(line);
* while (matcher.find()) {
* occurrenceCount++;
* }
* }
* System.out.println("The word '" + searchWord + "' appeared " + occurrenceCount + " time(s) in " + fileName);
* } catch (IOException e) {
* System.err.println("An error occurred while reading the file: " + e.getMessage());
* } finally {
* scanner.close();
* }
* }
* }
* 13. Write a program to create, move, and delete a file using Files and Paths.
* Program:
* import java.io.IOException;
* import java.nio.file.Files;
* import java.nio.file.Path;
* import java.nio.file.Paths;
* import java.nio.file.StandardCopyOption;
* public class FileOperationsNIO {
* public static void main(String[] args) {
* Path originalPath = Paths.get("temp\_file.txt");
* Path movedPath = Paths.get("moved\_file.txt");
* try {
* // 1. Create a file
* Files.createFile(originalPath);
* System.out.println("File created: " + originalPath.toAbsolutePath());
* // 2. Write some content to the created file (optional, for visibility)
* Files.write(originalPath, "This is a temporary file.".getBytes());
* // 3. Move the file
* // REPLACE\_EXISTING option will overwrite the target if it already exists
* Files.move(originalPath, movedPath, StandardCopyOption.REPLACE\_EXISTING);
* System.out.println("File moved from " + originalPath.toAbsolutePath() + " to " + movedPath.toAbsolutePath());
* // 4. Delete the file
* Files.delete(movedPath);
* System.out.println("File deleted: " + movedPath.toAbsolutePath());
* } catch (IOException e) {
* System.err.println("An error occurred during file operations: " + e.getMessage());
* }
* }
* }
* 14. Write a program to read all lines of a file using Files.readAllLines() and print them.
* Program:
* import java.io.IOException;
* import java.nio.file.Files;
* import java.nio.file.Path;
* import java.nio.file.Paths;
* import java.util.List;
* public class ReadAllLinesNIO {
* public static void main(String[] args) {
* Path filePath = Paths.get("lines\_to\_read.txt"); // Create this file with some content
* try {
* List<String> lines = Files.readAllLines(filePath);
* System.out.println("Content of '" + filePath.getFileName() + "':");
* for (String line : lines) {
* System.out.println(line);
* }
* } catch (IOException e) {
* System.err.println("An error occurred while reading the file: " + e.getMessage());
* }
* }
* }
* 15. Write a program to write data into a file using Files.write() and append using StandardOpenOption.APPEND.
* Program:
* import java.io.IOException;
* import java.nio.file.Files;
* import java.nio.file.Path;
* import java.nio.file.Paths;
* import java.nio.file.StandardOpenOption;
* import java.util.Arrays;
* import java.util.List;
* public class WriteAndAppendNIO {
* public static void main(String[] args) {
* Path filePath = Paths.get("data\_nio.txt");
* List<String> initialLines = Arrays.asList("Line 1: Initial content.", "Line 2: More initial content.");
* List<String> linesToAppend = Arrays.asList("Appended line 1.", "Appended line 2.");
* try {
* // Write initial content (this will overwrite if file exists)
* Files.write(filePath, initialLines);
* System.out.println("Initial content written to " + filePath.getFileName());
* // Append more content to the file
* Files.write(filePath, linesToAppend, StandardOpenOption.APPEND);
* System.out.println("Content appended to " + filePath.getFileName());
* // Verify content by reading it back
* System.out.println("\nFinal content of " + filePath.getFileName() + ":");
* Files.readAllLines(filePath).forEach(System.out::println);
* } catch (IOException e) {
* System.err.println("An error occurred: " + e.getMessage());
* }
* }
* }
* 16. Write a program to walk through a directory tree and display file names using Files.walk().
* Program:
* import java.io.IOException;
* import java.nio.file.Files;
* import java.nio.file.Path;
* import java.nio.file.Paths;
* import java.nio.file.StandardOpenOption;
* import java.util.Arrays;
* import java.util.List;
* public class WriteAndAppendNIO {
* public static void main(String[] args) {
* Path filePath = Paths.get("data\_nio.txt");
* List<String> initialLines = Arrays.asList("Line 1: Initial content.", "Line 2: More initial content.");
* List<String> linesToAppend = Arrays.asList("Appended line 1.", "Appended line 2.");
* try {
* // Write initial content (this will overwrite if file exists)
* Files.write(filePath, initialLines);
* System.out.println("Initial content written to " + filePath.getFileName());
* // Append more content to the file
* Files.write(filePath, linesToAppend, StandardOpenOption.APPEND);
* System.out.println("Content appended to " + filePath.getFileName());
* // Verify content by reading it back
* System.out.println("\nFinal content of " + filePath.getFileName() + ":");
* Files.readAllLines(filePath).forEach(System.out::println);
* } catch (IOException e) {
* System.err.println("An error occurred: " + e.getMessage());
* }
* }
* }
* 17. Write a program to copy a file using Files.copy() with REPLACE\_EXISTING option.
* Program:
* import java.io.IOException;
* import java.nio.file.Files;
* import java.nio.file.Path;
* import java.nio.file.Paths;
* import java.nio.file.StandardCopyOption;
* public class CopyFileNIO {
* public static void main(String[] args) {
* Path sourcePath = Paths.get("source\_copy.txt"); // Create this file with some content
* Path destinationPath = Paths.get("destination\_copy.txt");
* try {
* // Ensure source file exists for demonstration
* if (!Files.exists(sourcePath)) {
* Files.write(sourcePath, "Content to be copied.".getBytes());
* System.out.println("Created source\_copy.txt for demonstration.");
* }
* Files.copy(sourcePath, destinationPath, StandardCopyOption.REPLACE\_EXISTING);
* System.out.println("File copied from " + sourcePath.getFileName() + " to " + destinationPath.getFileName() + ".");
* System.out.println("If 'destination\_copy.txt' existed, it was replaced.");
* } catch (IOException e) {
* System.err.println("An error occurred during file copying: " + e.getMessage());
* }
* }

}

* 18. Write a program to check and print the size of a file in bytes using Files.size().
* Program:
* import java.io.IOException;
* import java.nio.file.Files;
* import java.nio.file.Path;
* import java.nio.file.Paths;
* public class FileSizeNIO {
* public static void main(String[] args) {
* Path filePath = Paths.get("output.txt"); // Use a file that exists
* try {
* if (Files.exists(filePath)) {
* long size = Files.size(filePath);
* System.out.println("The size of '" + filePath.getFileName() + "' is " + size + " bytes.");
* } else {
* System.out.println("File '" + filePath.getFileName() + "' does not exist.");
* }
* } catch (IOException e) {
* System.err.println("An error occurred while getting file size: " + e.getMessage());
* }
* }
* }
* 19. Write a program to serialize a class Employee and store it in employee.ser.
* Program:
* import java.io.FileOutputStream;
* import java.io.IOException;
* import java.io.ObjectOutputStream;
* import java.io.Serializable;
* class Employee implements Serializable {
* private static final long serialVersionUID = 3L; // Recommended
* private String name;
* private int id;
* private double salary;
* public Employee(String name, int id, double salary) {
* this.name = name;
* this.id = id;
* this.salary = salary;
* }
* // Getters
* public String getName() { return name; }
* public int getId() { return id; }
* public double getSalary() { return salary; }
* @Override
* public String toString() {
* return "Employee [name='" + name + "', id=" + id + ", salary=" + salary + "]";
* }
* }
* public class SerializeEmployee {
* public static void main(String[] args) {
* Employee emp = new Employee("Michael Scott", 789, 85000.00);
* String fileName = "employee.ser";
* try (FileOutputStream fileOut = new FileOutputStream(fileName);
* ObjectOutputStream out = new ObjectOutputStream(fileOut)) {
* out.writeObject(emp);
* System.out.println("Employee object serialized and saved to " + fileName);
* } catch (IOException i) {
* i.printStackTrace();
* }
* }
* }
* 20. Write a program to deserialize the employee.ser file and display the object data.
* Program:
* import java.io.FileInputStream;
* import java.io.IOException;
* import java.io.ObjectInputStream;
* public class DeserializeEmployee {
* public static void main(String[] args) {
* Employee emp = null;
* String fileName = "employee.ser";
* try (FileInputStream fileIn = new FileInputStream(fileName);
* ObjectInputStream in = new ObjectInputStream(fileIn)) {
* emp = (Employee) in.readObject();
* System.out.println("Employee object deserialized from " + fileName);
* System.out.println("Deserialized Employee Data: " + emp);
* } catch (IOException i) {
* i.printStackTrace();
* return;
* } catch (ClassNotFoundException c) {
* System.out.println("Employee class not found.");
* c.printStackTrace();
* return;
* }
* }
* }