📄 Java File Documentation: file01.java

**1. Project Package**

* **package myjavaproject;**

This statement declares that the class belongs to the myjavaproject package. Organizing classes into packages helps manage large projects and avoid name conflicts.

**2. Imports**

* **import java.io.\*;**
* **import java.util.\*;**

java.io.\*: Imports all Input/Output classes like BufferedReader, BufferedWriter, FileReader, FileWriter, IOException, etc.

java.util.\*: Imports utility classes, including Scanner for user input.

**3. Class Declaration**

* **public class file01**

Defines a public class named file01. It contains methods to perform file operations like writing, reading, and modifying text files.

**4. Methods**

* **writeToFile(String filename, String content)**

public static void writeToFile(String filename, String content)

Purpose: Writes the given content to the specified filename.

Operation:

Uses BufferedWriter with FileWriter to write content.

If the file exists, it overwrites it.

If the file doesn’t exist, it creates a new one.

Error Handling: Catches IOException and prints the stack trace.

* **readFromFile(String filename)**

public static void readFromFile(String filename)

Purpose: Reads and prints the contents of the specified file line by line.

Operation:

Uses BufferedReader with FileReader.

Prints each line of the file to the console.

Error Handling: Catches and logs any read errors.

* **modifyFile(String filename, String target, String replacement)**

public static void modifyFile(String filename, String target, String replacement)

Purpose: Searches the file for a specific target word and replaces all its occurrences with the replacement word.

Operation:

Reads the file content using BufferedReader.

Replaces words using String.replaceAll() and stores modified content in a StringBuilder.

Writes the updated content back to the same file using BufferedWriter.

Note: The entire file is read into memory before writing back.

Error Handling: Separate try-catch blocks handle read and write exceptions.

**5. Main Method**

public static void main(String[] args)

Purpose: Provides a basic console interface to:

Accept file name and content

Write to the file

Read the written content

Accept a word to replace and its replacement

Modify and re-read the updated file

✅ Steps:

Prompt for file name and user input for content.

Call writeToFile() to save the content.

Display the file content via readFromFile().

Ask the user for a target word and a replacement word.

Call modifyFile() to perform search and replace.

Display the modified file again.

**6. Program Output Sample**

Enter filename: sample.txt

Enter content to write to the file: Java is fun. Java is powerful.

File written successfully.

Reading the file:

Java is fun. Java is powerful.

Enter word to replace: Java

Enter replacement word: Python

File modified successfully.

Reading the modified file:

Python is fun. Python is powerful.

**7. Error Handling**

Each file operation includes proper exception handling using try-catch.

If any I/O error occurs (e.g., file not found, permission denied), a message is shown and the error is printed using e.printStackTrace().

**8. Enhancements You Can Add**

Append mode in writing

Line-by-line modification option

Word count and search

GUI for file input/output

Use regex-based modifications

**9. Conclusion**

This class demonstrates basic file handling using Java I/O in a clean, modular way. It’s an excellent example for:

Beginners learning Java file handling

Simple applications needing file input/output

Demonstrating modification of text files