Lec 44 File IO P3:

This session explains how to handle files in Java using FileReader, FileWriter, FileInputStream, and FileOutputStream. It shows how to efficiently read and write text files, manage binary files, and build a simple file management system that can create, show, search, delete, backup, and restore files.

Understanding FileReader and FileWriter

FileReader and FileWriter are used for reading and writing text files in Java. They are more efficient for text data compared to FileInputStream and FileOutputStream, which are better suited for binary data like images and videos.

Efficiency of Character Streams vs Byte Streams

FileReader and FileWriter use character streams, making them faster and more suitable for text, while FileInputStream and FileOutputStream use byte streams, which require converting characters to bytes and are less efficient for text.

Choosing the Right Stream for Data Type

Use FileReader/FileWriter for text files, and FileInputStream/FileOutputStream for binary files. Using the wrong stream type can lead to inefficiency or data corruption.

Writing Text to a File with FileWriter

To write text to a file, create a FileWriter with a file path, use the write() method, and close the writer. There's no need to convert text to bytes.

Reading Text from a File with FileReader

To read text, use FileReader to read characters one by one in a loop until the end of the file, then close the reader. This method is efficient for text files.

How FileReader/FileWriter Work Internally

FileReader/FileWriter work with character streams, avoiding the overhead of converting between bytes and characters, which increases the speed of reading and writing text.

Mini Project: File Management System Overview

The project involves creating a text file database where users can create new files, list all files, search by name, delete files, backup deleted files, and restore them from backup, all through a simple menu.

Building the Menu-Driven Application

The application uses a single class with different methods for each operation (create, show, search, delete, restore) to keep the focus on file handling logic rather than object-oriented design.

Creating Files in a Directory

Files are created by specifying a directory and file name, using Java's File class and createNewFile() method, with checks to handle duplicates and exceptions.

Listing All Files in a Folder

Java's File.listFiles() method retrieves all files in a directory. Their names can be stored in an ArrayList for easy display and searching.

Searching for a File by Name

The file name is checked in the ArrayList using indexOf(). If found, its name is returned; otherwise, a “file not found” message is given.

Opening and Displaying File Contents

To open a file, check if it exists, then use FileReader to read and display its contents. Proper error handling prevents crashes if the file is missing.

Deleting Files with Backup (Soft Delete)

Before deleting, the file is copied to a backup folder using FileReader and FileWriter. After backup, the file is deleted from the main folder, allowing for future restoration.

Restoring Deleted Files from Backup

To restore, the file is copied from the backup folder back to the main directory, and then deleted from the backup. Existence checks and error handling ensure smooth restoration.

Deleting All Files in a Folder

All files are deleted by iterating over each file object in the directory and calling the delete method, with confirmation prompts to prevent accidental loss.

Importance of Practice and Project Work

Building projects helps solidify concepts and reveals practical challenges like exception handling and logic building, which are not obvious from theory alone.--- This summary provides a structured learning path through the key concepts and practical implementations of file handling in Java, along with reflective questions to encourage deeper learning.