Day 24

File Handling

Java Folder Methods

1. Folder Creation (mkdir())

- **Method**: mkdir()
- **Usage**: Creates a new directory (folder) at the specified path.
- **Return Type**: boolean
  - Returns true if the directory was successfully created.
  - Returns false if the directory could not be created (e.g., if it already exists or the path is invalid).

```
File folder = new File(folderpath);
if (!folder.exists()) {
    folder.mkdir();
    System.out.println("Folder created: " + folder.getName());
}
```

2. Check Folder Existence (exists())

- **Method**: exists()
- **Usage**: Checks whether the directory (or file) at the given path exists.
- **Return Type**: boolean
  - Returns true if the directory exists.
  - Returns false if the directory does not exist.

```
File folder = new File(folderpath);
return folder.exists();
```

3. Folder Renaming (renameTo())

- **Method**: renameTo(File newFile)
- **Usage**: Renames an existing directory to a new name or moves it to a different path.
- **Return Type**: boolean
    - Returns true if the directory was successfully renamed or moved.
    - Returns false if the operation fails (e.g., the folder doesn't exist or the target path is invalid).

```
File oldFolder = new File(oldPath);
File newFolder = new File(newPath);
if (oldFolder.exists()) {
    oldFolder.renameTo(newFolder);
    System.out.println("Folder renamed to: " + newFolder.getName());
}
```

4. Folder Deletion (delete())

- **Method**: delete()
- **Usage**: Deletes the folder and its contents.
- **Return Type**: boolean
    - Returns true if the folder was successfully deleted.
    - Returns false if the folder cannot be deleted (e.g., the folder doesn't exist or it contains files).
- **Implementation Note**: In this example, the code first deletes all the files within the folder using a loop and then deletes the folder itself.

```
File folder = new File(folderPath);
if (folder.exists()) {
    for (File file : folder.listFiles()) {
        file.delete();
    }
    folder.delete();
    System.out.println("Folder deleted: " + folder.getName());
}
```

Java File Methods

1. File Creation (createNewFile())

- **Method**: createNewFile()
- **Usage**: Creates a new, empty file if it doesn't already exist.
- **Return Type**: boolean
    - Returns true if the file was created successfully.
    - Returns false if the file already exists.
- **Exception**: Throws IOException if the file cannot be created due to an I/O error (e.g., incorrect file path, permission issues).

```
File myObj = new File(filePath);
if (myObj.createNewFile()) {
    System.out.println("File created: " + myObj.getName());
} else {
    System.out.println("File already exists.");
}
```

2. File Writing (FileWriter.write())

- **Class**: FileWriter
- **Method**: write(String data)
- **Usage**: Writes data into a specified file. It will overwrite the existing content.
- **Return Type**: void
- **Close Operation**: After writing, always close the FileWriter using close() to ensure the data is properly saved and file resources are freed.
- **Exception**: Throws IOException if there's a problem with file access (e.g., file not found, no write permissions).

```
FileWriter myWriter = new FileWriter(filePath);
myWriter.write("Welcome to Java file handling...");
myWriter.close();
```

3. File Reading (Scanner.hasNextLine() and Scanner.nextLine())

- **Class**: Scanner
- **Method**:
  - hasNextLine(): Checks if the file contains another line.
  - nextLine(): Reads the next line of the file.
- **Usage**: Reads data from a file line by line using a Scanner object.
- **Return Type**: boolean (for hasNextLine()) and String (for nextLine()).
- **Exception**: Throws FileNotFoundException if the file does not exist or cannot be opened.

```
Scanner myReader = new Scanner(myObj);
while (myReader.hasNextLine()) {
    String data = myReader.nextLine();
    System.out.println(data);
}
myReader.close();
```

4. File Renaming (renameTo())

- **Method**: renameTo(File newFile)
- **Usage**: Renames an existing file to a new name.
- **Return Type**: boolean
  - Returns true if the file was renamed successfully.
  - Returns false if the file couldn't be renamed (e.g., the target file already exists or the OS restricts the operation).

```
File oldFile = new File(oldfilePath);
File newFile = new File(newfilePath);
if (oldFile.renameTo(newFile)) {
    System.out.println("File is renamed.");
} else {
    System.out.println("File cannot be renamed.");
}
```

5. File Deletion (delete())

- **Method**: delete()
- **Usage**: Deletes the specified file.
- **Return Type**: boolean
    - Returns true if the file was deleted successfully.
    - Returns false if the file cannot be deleted (e.g., file does not exist, or there are OS restrictions).

```
File myObj = new File(filePath);
if (myObj.delete()) {
    System.out.println("Deleted the file: " + myObj.getName());
} else {
    System.out.println("Failed to delete the file.");
}
```

Reading properties file

Properties Methods:

1. **load(InputStream inStream)**: Loads properties from an input stream (e.g., from a file).
2. **getProperty(String key)**: Fetches the value associated with the given key.
3. **stringPropertyNames()**: Returns a set of all property keys (as String).
4. **keySet()**: Returns a set of all property keys (as Object).
5. **values()**: Retrieves all values from the properties object.
6. **close()**: Closes the input stream or output stream to release system resources.

More Videos on File Handling:

https://shorturl.at/iAgo1