Resource Management - Exception Handling by Monica PS - 7221964

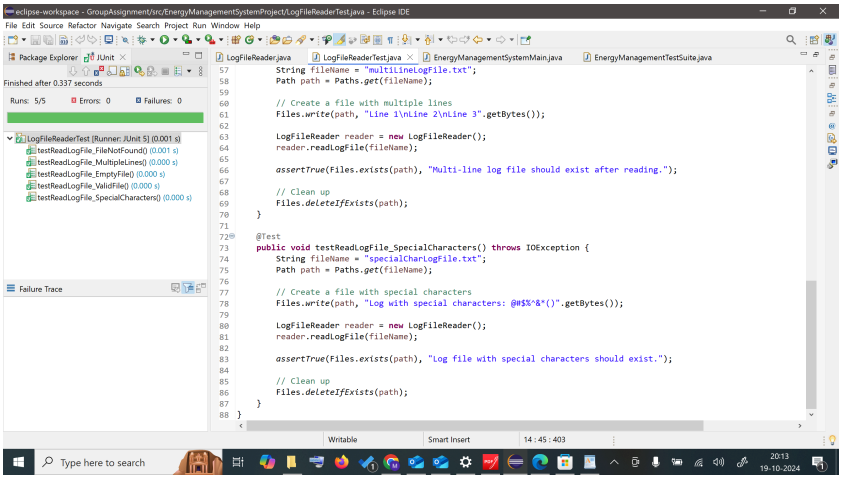
This code defines a LogFileReader class in the package EnergyManagementSystemProject, designed to read and display the contents of a log file. EnergyManagementSystemProject package, which could contain multiple classes related to energy management, like logging, file handling, etc.


```java
package EnergyManagementSystemProject;

import java.io.IOException;

public class LogFileReader {

    // Method to read the contents of a file using a more modern, compact API
    public void readLogFile(String filePath) {
        Path path = Paths.get(filePath);

        try {
            // Using Files.newBufferedReader() which is more modern and efficient
            try (var reader = Files.newBufferedReader(path)) {
                String line;
                while ((line = reader.readLine()) != null) {
                    System.out.println(line);
                }
            }
        } catch (IOException e) {
            System.out.println("[LOG READER] Error reading file: " + e.getMessage());
        }
    }
}
```

- @Test: Marks methods that are unit tests.
- assertTrue(), assertDoesNotThrow(): Assertion methods to verify test conditions.

Java I/O imports:

- IOException: For handling potential I/O exceptions.
- Files, Path, and Paths: To perform file operations (create, write, delete, and check file existence).

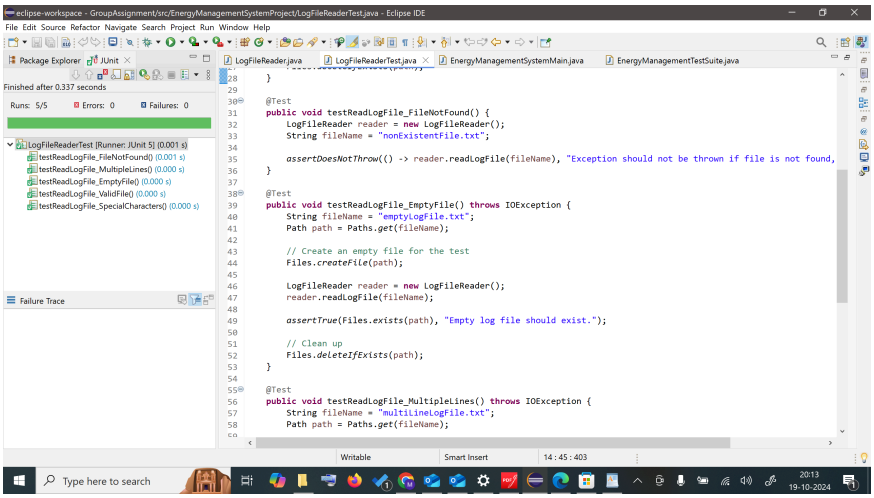


Files.newBufferedReader(path): This method returns a BufferedReader that reads the file efficiently, using the given Path object.

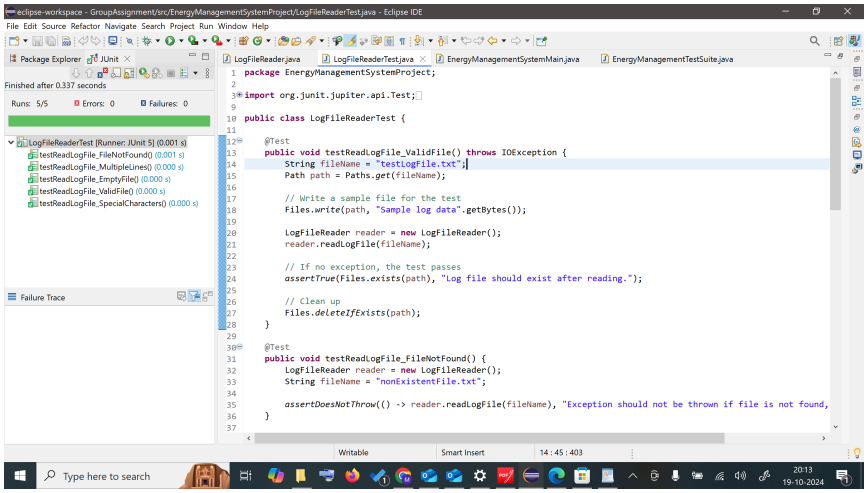
try (var reader = ...): This is a *try-with-resources* block, which automatically closes the BufferedReader after reading the file, ensuring no resources (like file handles) are left open.

reader.readLine(): Reads each line from the file. If a line exists, it is printed to the console using System.out.println(line);.

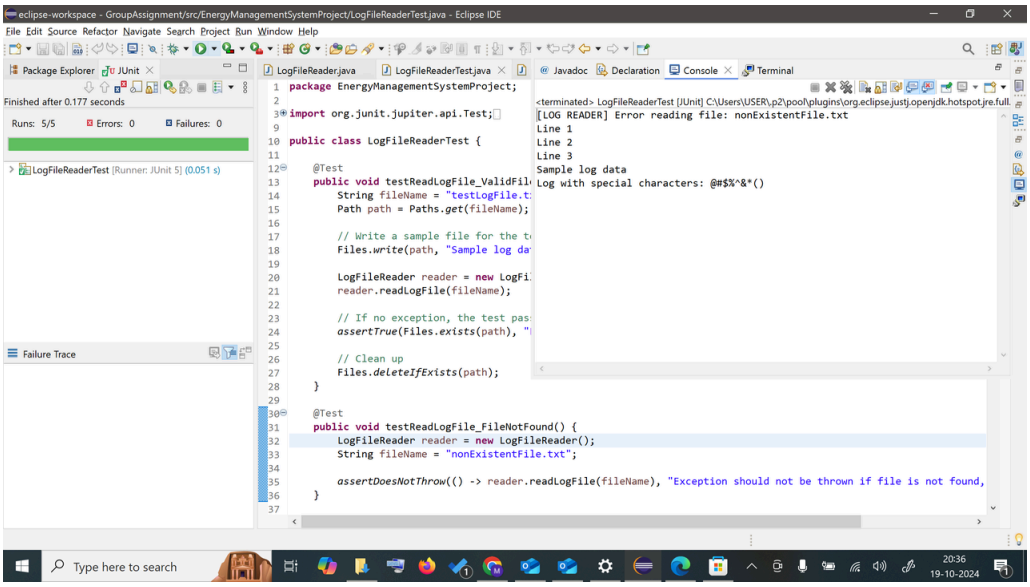
Error Handling: If an IOException occurs (e.g., if the file doesn't exist or can't be read), it is caught, and an error message is printed



A file (specialCharLogFile.txt) is created with special characters in its content.The readLogFile() method reads and prints the file.The test asserts that the file still exists after being read.The file is deleted after the test.This public class contains unit tests to validate the behavior of LogFileReader.



The code is robust enough to handle common edge cases, like empty files or missing files, without throwing exceptions unnecessarily.


eclipse-workspace - GroupAssignment/src/EnergyManagementSystemProject/LogFileReaderTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
Finished after 0.177 seconds
Runs: 5/5
Errors: 0
Failures: 0
LogFileReaderTest [Runner: JUnit 5] (0.051 s)
Failure Trace
LogFileReader.java
LogFileReaderTest.java
package EnergyManagementSystemProject;
import org.junit.jupiter.api.Test;
public class LogFileReaderTest {
@Test
public void testReadLogFile_ValidFil
String fileName = "testLogFile.t
Path path = Paths.get(fileName);
// Write a sample file for the t
Files.write(path, "Sample log da
LogFileReader reader = new LogFi
reader.readLogFile(fileName);
// If no exception, the test pas
assertTrue(Files.exists(path), "
// Clean up
Files.deleteIfExists(path);
}
@Test
public void testReadLogFile_FileNotFound() {
LogFileReader reader = new LogFileReader();
String fileName = "nonExistentFile.txt";
assertDoesNotThrow(() -> reader.readLogFile(fileName), "Exception should not be thrown if file is not found,
}
Javadoc
Declaration
Console
Terminal
<terminated> LogFileReaderTest [JUnit] C:\Users\USER\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full
[LOG READER] Error reading file: nonExistentFile.txt
Line 1
Line 2
Line 3
Sample log data
Log with special characters: @#$%^&*()
Type here to search
20:36
19-10-2024