

Java-Reading a File

Project Abstract

The purpose of this project is to demonstrate how to read the content of a file in Java using the `Scanner` class. This task helps you understand how to check the existence of a file, read data from it, and handle file reading errors.

This project focuses on:

1. Understanding how to read files in Java using the `Scanner` class.
2. Checking if the file exists before attempting to read it.
3. Handling errors that may occur during the file reading process.
4. Demonstrating how to read the content of the file line by line and print it to the console.

Tasks Overview

Task 1: Read Content from a File Using the `Scanner` Class

Objective: Read the content of a file named "readFile.txt" using the `Scanner` class and print it to the console.

Detailed Description: In this task, you will use the `File` class to represent the file "readFile.txt". Before reading the file, the program will check if the file exists. If the file exists, the program will use the `Scanner` class to read its content line by line and print each line to the console. If the file does not exist, the program will print a message indicating that the file does not exist.

Steps:

1. **Create a File Instance:**
 - Declare a `File` object named `file` using the constructor `new File("readFile.txt")`.

This will represent the file "readFile.txt", which you will attempt to read from.

2. **Check if the File Exists:**
 - Use the `exists()` method to check if the file already exists. If it does not exist, print the message "File does not exist." and exit the method.

3. **Read the Content of the File:**
 - Use the `Scanner` class to read the content of the file. Declare a `Scanner` object named `reader` and pass the `file` object to it.
 - Use the `hasNextLine()` method in a `while` loop to check if there are more lines in the

file. Inside the loop, use the `nextLine()` method to read each line from the file and print it to the console.



4. ****Close the Scanner:****

- After reading all lines from the file, close the `Scanner` object using `reader.close()` to release system resources.

5. ****Error Handling:****

- If the file is not found, the program will catch the `FileNotFoundException`. In this case, print the message "An error occurred. File not found." and display the exception stack trace.

Execution Steps to Follow:

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top)  Terminal  New Terminal.
3. This editor Auto Saves the code.
4. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
6. To run your project use command:
`mvn compile exec:java -Dexec.mainClass="com.yaksha.assignment.FileReading"`
7. To test your project test cases, use the command
`mvn test`
8. You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.