Cairo University
Faculty of Computers & Artificial Intelligence
CS341 - Operating Systems 1
Assignment #1

Assignment #1 - Command Line Interpreter

Assignment Description:

Develop a basic Command Line Interpreter (CLI), similar to a Unix/Linux shell, that supports the following features:

1. **Command Execution**:
   The CLI should execute system commands like `pwd`, `cd`, `ls`, `ls -a`, `ls -r`, `mkdir`, `rmdir`, `touch`, `mv`, `rm`, `cat`, `>`, `>>`, `|`

2. **Internal Commands**:
   Implement internal commands such as:
   - `exit`: To terminate the CLI.
   - `help`: Displays available commands and their usage.

JUnit:

JUnit is a popular framework used in Java for writing and running unit tests.
Unit testing involves testing small pieces of code, typically individual methods, to ensure they work correctly. JUnit allows developers to write automated test cases that can be executed repeatedly, ensuring their code behaves as expected even after changes or updates.

Importance of JUnit in Professional Development:

1. **Code Quality**: It helps ensure that code functions as intended and catches bugs early in development.
2. **Efficiency**: Automated testing saves time by eliminating the need for manual testing, especially for large projects.
3. **Job Readiness**: Proficiency in JUnit is valuable in the industry because most companies emphasize test-driven development (TDD) and automated testing to maintain high-quality software.
4. **Debugging**: When bugs arise, JUnit makes it easier to isolate and identify which part of the code is malfunctioning.

Learning JUnit is crucial for a developer's toolkit and can enhance your employability after graduation, as many companies expect new hires to understand testing methodologies.
Here's a great JUnit tutorial for beginners on W3Schools. It covers the basics of unit testing with JUnit, including annotations like @Test, @Before, and @After, and explains the role of the Assert class in verifying test results.

Cairo University
Faculty of Computers & Artificial Intelligence
CS341 - Operating Systems 1
Assignment #1

Instructions for Writing JUnit Test Cases

As part of your assignment, you'll also need to write **JUnit test cases** to verify that your commands are functioning correctly. This helps ensure the reliability of your code and provides a foundation for automated testing in future projects.

Step-by-Step Instructions:

1. **Set up JUnit in your IDE:** Most IDEs (e.g., IntelliJ IDEA, Eclipse) support JUnit. Make sure JUnit is added to your project's dependencies.

2. **Create a Test Class:**
   - For each command you are implementing, create a corresponding test class. These classes will contain methods that test the functionality of your CLI commands.
   - Use the annotation @Test before each method to mark it as a JUnit test.

3. **Write Test Methods:**
   - Focus on testing the behavior of individual commands.
   - Use assertions like assertEquals(), assertTrue(), and assertFalse() to verify expected results.

4. **Run the Tests**

Examples:

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class CommandLineInterpreterTest {

    @Test
    void testCd() {
        CommandLineInterpreter cli = new CommandLineInterpreter();
        cli.cd("/home/user/Documents");
        assertEquals("/home/user/Documents", cli.pwd());
    }

    @Test
    void testLs() {
        CommandLineInterpreter cli = new CommandLineInterpreter();
        cli.cd("/home/user/Documents");
        String[] files = cli.ls();
        // Assuming certain files are expected in the Documents folder
        assertTrue(files.length > 0, "The ls command should return the list of files in the directory.");
    }

}


```

Cairo University
Faculty of Computers & Artificial Intelligence
CS341 - Operating Systems 1
Assignment #1

Notes:

- You must handle all the cases in each command.
  - mv command for example is used to cut and rename files **both** functionality must be developed.

- If the user enters a wrong command or bad parameters (invalid path, file instead of directory in certain commands, etc.), **the program should print some error messages without terminating**.

- You can refer to the lab document for further information on the commands

- You can add more attributes/methods.

- You can use built-in functions and predefined classes in Java.

- **Do not use** "exec" to implement any of these commands or you will lose marks.

Guidelines:

- **Language:** Java.

- **Submission Deadline:** 31st October, 2024 11:59 PM.

- **Submission Format:** You must submit only one ".zip" file containing the source code, and the submitted file name must follow this format: ID1_ID2_ID3_Group **for example** 20190000_20190001_20190002_DS1.

- **Team Members:** The assignment is submitted in groups of **4** students and **only 4** not less than or greater than 4. All team members must be from the **same group**.