**In-Memory File System Documentation**

**Implementation Overview:**

The in-memory file system is implemented in Java and consists of three main classes:

1. **FileSystem:** Represents the main file system and contains methods for creating directories, changing the current directory, listing contents, creating files, writing to files, and performing various file system operations.

2. **Directory:** Represents a directory in the file system and contains information about its name, parent directory, subdirectories, and files. It also provides methods for adding, removing, and accessing subdirectories and files.

3. **File:** Represents a file in the file system and contains information about its name and content. It provides methods for setting and getting file content, as well as creating a copy of the file.

**Data Structures Used:**

**FileSystem Class:**

- Uses a single instance of the `Directory` class as the root directory.

- Maintains a reference to the current working directory, allowing operations within the context of this directory.

**Directory Class:**

- Uses a list to store subdirectories (`List<Directory>`).

- Uses a list to store files (`List<File>`).

- Contains references to the parent directory.

**File Class:**

- Stores the file name and content as strings.

**Design Decisions:**

**Serialization:** The `FileSystem` class implements the `Serializable` interface to allow the serialization and deserialization of the entire file system state. This enables saving and loading the file system state from a file.

**Recursive Operations:** The file system supports recursive operations, such as removing a directory and its contents (`rm` command) and searching for a string in files across directories (`grep` command).

**User Interface:** The system uses a simple command-line interface, accepting user commands for file system operations.

**Setup Script (Comments):**  
// Setup Script:

// 1. Compile the Java code using the Java compiler (javac).

// 2. Run the compiled Java program.

// Step 1: Compile Java Code

// Open terminal and navigate to the directory containing the code.

// Execute the following command to compile:

// javac InMemoryFileSystem.java

// Step 2: Run the Compiled Program

// After successful compilation, run the program using the following command:

// java InMemoryFileSystem

// You can provide additional command-line arguments, such as "load\_state", to load a saved state.

// Note: Ensure Java Development Kit (JDK) is installed on your system.

**Docker File (Example):**

# Dockerfile

# This Dockerfile assumes that you have Java installed on your system.

# Use an official OpenJDK runtime as a base image

FROM openjdk:latest

# Set the working directory inside the container

WORKDIR /usr/src/app

# Copy the compiled Java program to the container

COPY . .

# Compile the Java code

RUN javac InMemoryFileSystem.java

# Command to run the Java program

CMD ["java", "InMemoryFileSystem"]

**Conclusion:**

This documentation provides an overview of the in-memory file system, its implementation, data structures, design decisions, and includes basic instructions for setting up and running the system. Please adapt the setup instructions based on your specific environment and requirements.