|  |  |  |
| --- | --- | --- |
|  |  | REMOTE FILE SYSTEM BROWSER  DISTRIBUTED SYSTEMS |

8/11/2018

# Report

A screenshot of a cell phone

Description automatically generated

## IMPLEMENTATION

To begin with the project named Remote File System Browser I had the reference form the Java.io.file for implementing the methods. I created a server which accepts unicast and when started it bounds and waits for the client to connect .With the help of interface and its implementation class I created a client which have access to the file system with command line interface. Where it accepts basic linux command line , ls,cd,mkdir,touch,and rm. Through which ls lists the files ,cd goes into the file,mkdir makes a directory, touch makes a file and rm removes.Beside ls every other command should have the two parameters one is command and another is the name of the file or directory.

The methods I implemented are:

List<String> ls() **throws** RemoteException;

**boolean** mkdir(String name) **throws** RemoteException;

String cd(String path) **throws** RemoteException;

**boolean** touch(String name) **throws** RemoteException;

**boolean** rm(String name) **throws** RemoteException;

**boolean** exists() **throws** RemoteException;

**boolean** isDirectory() **throws** RemoteException;

**boolean** isFile() **throws** RemoteException;

here The title says the REMOTE FILE SYSTEM and we can see the methods implemented on the interface but it works in a client and a server with two mains with the help of RMI(Remote Method Invocation).

**RMI:**

**RMI** (Remote Method Invocation) is a way that a programmer, using the Java programming language and development environment, can write object-oriented programming in which objects on different computers can interact in a distributed network. It is an API that provides a mechanism to create distributed application in java. The **RMI** allows an object to invoke methods on an object running in another JVM. The **RMI** provides remote communication between the applications using two objects stub and skeleton.

The **Stub**/Skeleton hides the communication details away from the developer. The**Stub** is the class that implements the remote interface. It serves as a client-side placeholder for the remote object. The **stub** communicates with the server-side skeleton. The skeleton is the **stub's** counterpart on server-side.

-Rmi uses are remote object **registry** is a bootstrap naming service that is used by **RMI** servers on the same host to bind remote objects to names. Clients on local and remote hosts can then look up remote objects and make remote method invocations

*The advantages of RMI are:*

Object Oriented, Mobile Behavior, Design Patterns, Safe and Secure, Easy to Write/Easy to Use, Connects to Existing/Legacy Systems, Write Once, Run Anywhere, Distributed Garbage Collection, Parallel Computing: