Fadel Alshammasi & Lillian Krohn
COM S 454 Term Project
11/14/2020

Distributed RMI File Storage System - File Keeper

Introduction

For our term project, we have created a distributed file storage system implementation using Java RMI. Our project features an RMI interface containing our project functions, a Client class which allows the user to interact with the application on their terminal/command line, and Server class which keeps track of clients, maintains a hashmap of user credentials and another to keep track of notes for each user, and executes the functions that power our project. We will break down each element of our project in more detail below.

Features

User Sign Up

- In order to sign up for File Keeper, a user is first prompted to answer if they have an account already (y/n). If they respond ‘n’, they will be prompted to enter a username and password. If the username is already in use by another user, they will be prompted to try again. Once the username and password are accepted, the user will be prompted to log in by entering their newly created username and password. Our hashmap userInfo is used to keep track of usernames and passwords as key / value pairs.

User Log In

- To log into File Keeper, a user is first prompted if they have an account already (y/n). If they respond ‘y’, they will be prompted to enter their already-existing username and password. If the username or password entered does not match what is already stored by the server, they will be prompted to try again. Our hashmap userInfo is used to keep track of usernames and passwords as key / value pairs.

Drive Folder Creation

- Our server contains a Drive folder which holds a separate folder for each user that signs up. All of a user’s files will be stored in their own personal folder within the Drive, and users can only access their own folder. The user does not see any of the file creation actions, however, as this is all done behind the scenes on the server side. Once a user signs up, a new folder is created by the server named “username” (whatever the user chose to be their username). This created folder is very important, as all of the menu options use this folder to store their actions.

Menu Options

1. Upload Files to the Server

   If a user selects Menu Option “1”, the user will be prompted to enter the full file path of the file they wish to upload. If the file path is invalid, an appropriate error message is displayed and the user will have to try again. If a valid file path is uploaded, the server will invoke the remote method *upload* to upload the file into the user’s personal folder within the Drive. The server then saves the uploaded file to the user’s folder. Then, the user will be taken back to the main options menu.

2. Download Files from the Server

   If a user selects Menu Option “2”, the server will display all the files that the user has in their specific folder/directory (which is maintained by the server). Then, the user will be prompted to choose which file they want to download by entering the filename in the terminal/command line. After the file is specified, the user will be prompted to enter the location (pathname) that they want that file to be downloaded to (if the pathname is invalid, the user will be prompted to try again by being taken back to the main menu). The server will then save the file into the user’s specified location. The user can then open the location they specified and see the downloaded file. Then, the user will be taken back to the main options menu.

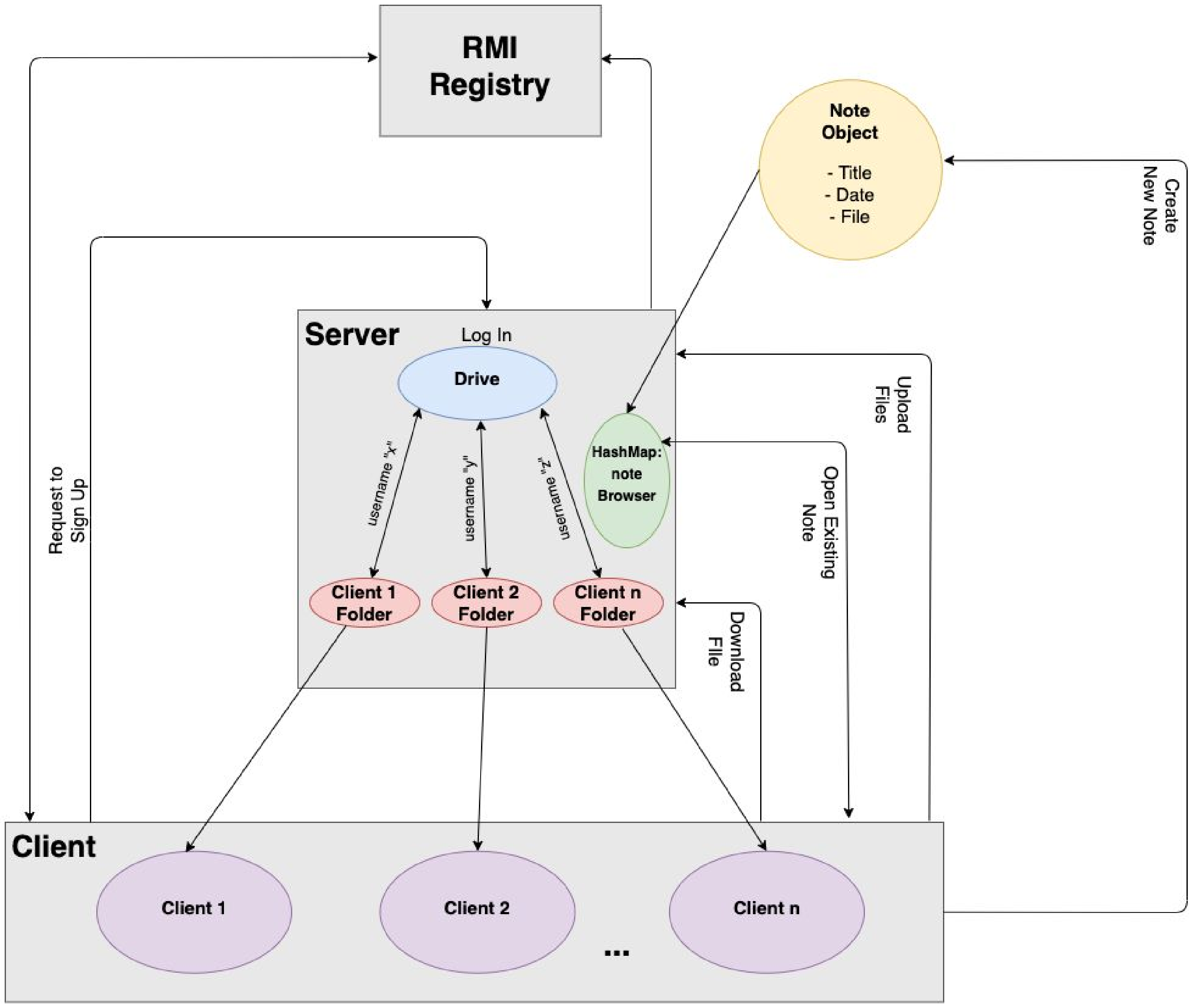
3. Create a New Note

   If a user selects Menu Option “3”, the server will prompt the user to enter the title they want to have for their new note (**Note: please do not include spaces in the title names - however you may include underscores, dashes, slashes, etc**.). Notes are customized objects that take in the title of the note, the date the note was created, and a file that will contain the body of the note - these are recorded by the server. Once a title is entered, the user will be prompted to enter the “body” (text) that will make up their note. The user can then enter as much text as they want, and can signify that they are done writing (this can be done by hitting the **crtl+Z** keys on windows or **ctrl+D** on mac/linux). We create a file object that contains the text entered by the user. The note object is now ready to be created and sent to the server by invoking the *uploadNoteToServer* method. On the server side, the notes are managed in a hashmap(key = username, value = a list of notes for that username). If it is the user’s first time uploading a note, a new array list is created to hold the note. If the user has already uploaded notes before, any new notes they upload will be added to the pre existing array list. Then, the user will be taken back to the main options menu.

4. Open an Existing Note

If a user selects Menu Option “4”, the server will display all the notes that the user has previously created or uploaded, along with the creation dates of the notes. Then, the user will be prompted to enter the name of the note that they wish to open/view. Once the name of the file is entered, the server will then open the note that the user specified and display it in the terminal/command line. Then, the user will be taken back to the main options menu.

System Diagram



Running our Project

To run our project, users should first download the files from canvas. Unzip the files, open the package, and open the “Server.java” and “Client.java” files. **You need to add the file path where you want the Drive items to be saved to on line 36 in “Server.java” and line 23 in “Client.java” - replace the string that is currently there** . Then save your changes. Now open a terminal window. Navigate to the location of the downloaded package and complete the following steps below:

1. Compile the Note class first using **“javac Note.java”**.
2. Compile all of the other files using “**javac *.java**”.
3. Type the command “**rmic Server**” to generate the java RMI class stubs (*please disregard any RMI warnings displayed*).
4. Then run the command “**rmiregistry**” to start up the RMI Registry..
5. Next, open up at least two more terminal windows - one for the server and one for the client (*Note: you can have more than one client window open if you wish*).
6. In one of the opened windows, execute the command “**java Server**” to start up the server file.
7. In the other opened window, execute the command “**java Client**” to start up the client file.
8. Now all of the necessary windows are open and running. You can start interacting with the client window by answering **(y/n)** to the prompt that asks if you already have an account or not.
9. You can then sign up with a new username and password, or log in with existing ones (*Note that if it is your first time running the application you must sign up first.*)
10. After signing up, you will be prompted to log in and will be taken to the main menu where you can select which option you want to choose by typing a number.
11. Select an option by typing the number - you will be taken back to the main menu after each chosen option is completed, allowing you to continue and select another option.