COMMAND SERVER PROJECT REPORT

Based on the communication between the client and the server, this project is intended for the server to respond to the windows commands sent by the client.

The following steps have been followed for this purpose:

1. Creating basic client – server communication.

This is main method of Server class and with this code, created a listener server for listening request from client.

```
public static void main(String args[])throws Exception{
    ss=new ServerSocket( port: 3333);
     s=ss.accept();
    din=new DataInputStream(s.getInputStream());
    dout=new DataOutputStream(s.getOutputStream());
```

This is main method of Client class.

```
public static void main(String args[])throws Exception{
    s=new Socket( host: "localhost", port: 3333);
    DataOutputStream dout=new DataOutputStream(s.getOutputStream());
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
```

2. If we want to create a command line server we need to reach the current path first.

- For that manner, client uses “cmd” command.

```
cmd
C:\Users\acer\IdeaProjects\Kodluyoruz>
```

For doing this,

```
private static  File currentPath;

static {
    try {
        currentPath = new File(System.getProperty("user.dir")).getCanonicalFile();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

This code blog will be helpful. Here we are getting the current path where we are standing on.

- For going a little forward, client can use “dir” command for seeing the file content.

```
C:\Users\acer\IdeaProjects\Kodluyoruz\src\Arrays>dir
C:\Users\acer\IdeaProjects\Kodluyoruz\src\Arrays
Arrays
ArraysPratik.java
Frequency.java
Transpose.java
C:\Users\acer\IdeaProjects\Kodluyoruz\src\Arrays>
```

As you can see here, we can see the content of the Arrays file.

```
} else if (str.equals("dir")) {
    name.add(String.valueOf(currentPath));

    Path dir = Paths.get(currentPath.toURI());
    Files.walk(dir).forEach(path -> {
        try {
            showFile(path.toFile());
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    });
    ObjectOutputStream objectOutput = new ObjectOutputStream(s.getOutputStream());
    objectOutput.writeObject(name);
```

```
public static void showFile(File file) throws IOException {

    if (file.isDirectory()) {
        name.add(file.getName());
    } else {
        name.add(file.getName());
    }


}
```

Thanks to these two code blogs, we are able to see the content of the current directory.

- And lets go back to previous directory and next directory.

For previous directory, we need to consider parent directory.

```
C:\Users\acer\IdeaProjects\Kodluyoruz\src\Arrays>cd ..
C:\Users\acer\IdeaProjects\Kodluyoruz\src>
```

For doing this;

```
if (strArr[1].equals("..")){
    System.out.println(currentPath);
    currentPath = new File(currentPath.getParent()).getCanonicalFile();
    ObjectOutputStream objectOutput = new ObjectOutputStream(s.getOutputStream());
    name = new ArrayList<>();
    name.add(String.valueOf(currentPath));
    objectOutput.writeObject(name);
```

In here, we are setting the current directory to parent directory via getParent() static method.

For next directory, we need to read the name of the directory that are wanted to pass. In Windows cd <nextFile> command is used. And first we need to check that written file exist or not.

If exist;

```
C:\Users\acer\IdeaProjects\Kodluyoruz\src>cd Arrays
C:\Users\acer\IdeaProjects\Kodluyoruz\src\Arrays>
```

Otherwise; dont move any where.

```
    File newPath = new File( pathname: currentPath +"\\" + strArr[1]);
    System.out.println(newPath);
    if (newPath.exists()){
        currentPath = newPath;
        ObjectOutputStream objectOutput = new ObjectOutputStream(s.getOutputStream());
        name = new ArrayList<>();
        name.add(String.valueOf(currentPath));
        objectOutput.writeObject(name);
    }else{
        ObjectOutputStream objectOutput = new ObjectOutputStream(s.getOutputStream());
        name = new ArrayList<>();
        name.add(String.valueOf(currentPath));
        objectOutput.writeObject(name);
    }
}
```

Firstly, we are creating new path and checking if it is exists. If it is, set the current directory to the new one. Otherwise, print the current directory.

- Lastly, we focus on “mkdir” command for creating new file in our cuurent directory.

```
} else if (strArr[0].equals("mkdir")) {
    File directory = new File( pathname: currentPath+"\\"+strArr[1]);
    if (!directory.exists()) {
        directory.mkdir();
        ObjectOutputStream objectOutput = new ObjectOutputStream(s.getOutputStream());
        name = new ArrayList<>();
        name.add(String.valueOf(currentPath));
        objectOutput.writeObject(name);
    }
} else {
```

Here, we are using mkdir method and if no exists such directory system will be create that written named directory.