Project: Command Server Project

This client-server project allows users to send commands from a client to a server, where commands are executed and results are returned to the client.

Classes:

- CommandServer
- ClientHandler
- CommandClient

First, we run the CommandServer application. A CommandServer object is created and the start() method is called. In the start method, PORT configuration is completed and the following output appears on screen.

```
Server started. PORT: 8080
Client waiting...
```

An infinite waiting loop is created for clients to connect. When we run CommandClient, the following output appears. When CommandClient connects, a ClientHandler is created on the CommandServer side and a newUserId is defined using UUID. This way, different users using the same application do not affect each other.

```
User ID: 35137a0e
Connection completed. Send command...
pwd
C:\Users\hp\AppData\Local\Temp\user_35137a0e
```

At this point, users can send commands through the CommandClient interface. We send commands through the executedCommand() method called in the run() method of ClientHandler. Here we read the command using BufferedReader and use if-else structures based on its content. For example, if the command starts with "cd" in executedCommand, we call the handleCdCommand() method. Here we can navigate to parent directories or move to desired directories based on the command content, and we save this directory as the current directory in the userDirectories map using the updateUserDirectory() method. If the command equals "pwd", we return the current directory location.

```
mkdir testdirectory
Command executed
cd testdirectory
C:\Users\hp\AppData\Local\Temp\user_35137a0e\testdirectory
```

If commands are other than these two, we proceed to the else condition and call the runSystemCommand() method. Here we can execute other system commands. We create a ProcessBuilder object to check the operating system and select cmd or Shell program. Then we set the working directory. Process starts and we read the command output. Here both output and error outputs are read. Then a timeout check is performed to prevent infinite loops and timeouts.

Here I will show examples of user commands. The application works on both UNIX and Windows systems. Since I use Windows, I will show outputs with Windows commands.

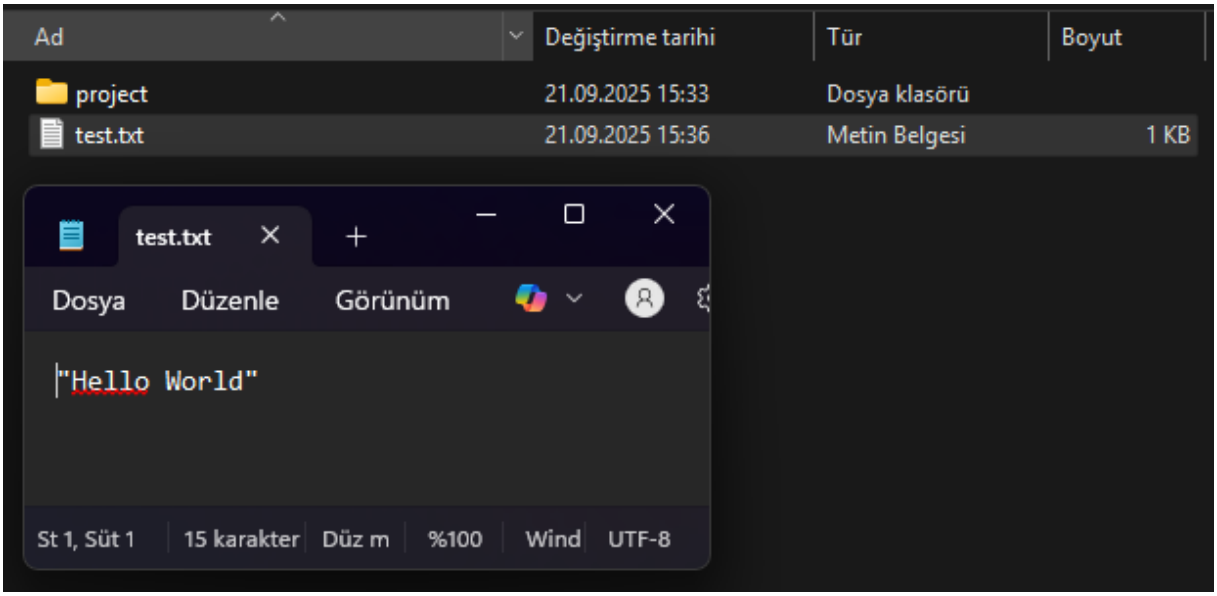
Now let's test some commands.

1. **As we can see below, we can navigate between directories.**

```
cd project
C:\Users\hp\AppData\Local\Temp\user_8697cbaf\project
pwd
C:\Users\hp\AppData\Local\Temp\user_8697cbaf\project
cd ..
C:\Users\hp\AppData\Local\Temp\user_8697cbaf
```

2. **Now let's create a file.**

```
echo "Hello World" > test.txt
Command executed
```



As we can see, we successfully executed the echo command.

3. **Let's test the type command.**

```
type test.txt
"Hello World"
```

4. **We can copy files and list both file information and file names separately.**

```
copy test.txt testcopy.txt
1 file(s) copied.
dir
Volume in drive C has no label.
 Volume Serial Number is 58F7-EC09

 Directory of C:\Users\hp\AppData\Local\Temp\user_8697cbaf

21.09.2025  15:41    <DIR>          .
21.09.2025  15:38    <DIR>          ..
21.09.2025  15:33    <DIR>          project
21.09.2025  15:36                16 test.txt
21.09.2025  15:36                16 testcopy.txt
               2 File(s)             32 bytes
               3 Dir(s)  117.752.741.888 bytes free
dir /b
project
test.txt
testcopy.txt
```

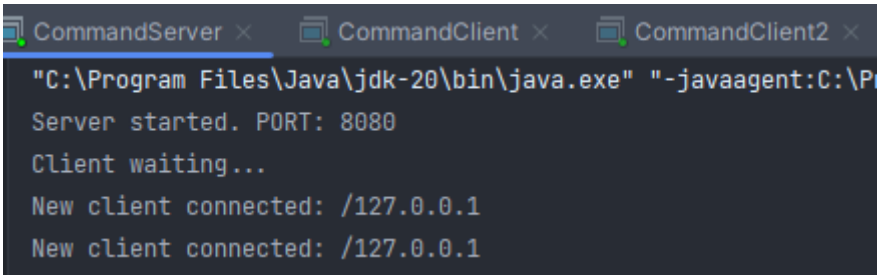
5. **Here we can perform file moving operations.**

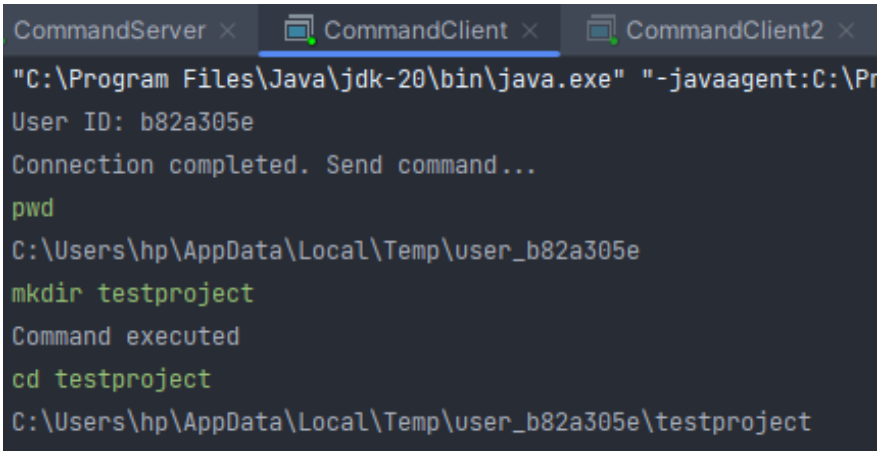
```
move testcopy.txt project\testcopy.txt
1 file(s) moved.
```

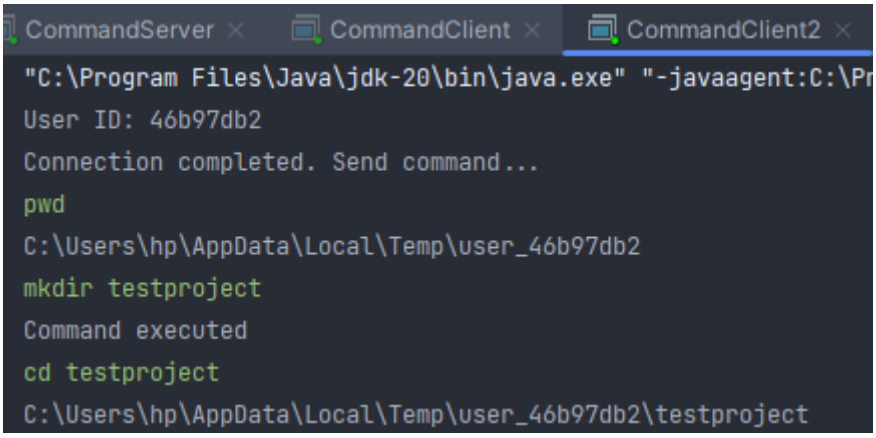
6. **Now let's perform file deletion.**

```
del testcopy.txt
Command executed
dir
Volume in drive C has no label.
 Volume Serial Number is 58F7-EC09

 Directory of C:\Users\hp\AppData\Local\Temp\user_8697cbaf\project

21.09.2025  15:46    <DIR>          .
21.09.2025  15:44    <DIR>          ..
               0 File(s)              0 bytes
```

7. **Now let's test two users connecting to the server at the same time.**







As we can see, both users are already working in isolation from each other as soon as they connect. This way, different people do not affect each other.

8. **When we write exit here, the finally block in the run() method executes and the connection closes.**