SP22 15W CSCI5531.01 Advanced Operating Systems

Project-1 [KSH]



Instruction Manual and architecture

of

Project on distributed Shopping System

**Developed and designed by**

**Swapnil Patel (1966690)**

Hosted on: https://github.com/swap1210/java-project

Table of Contents

[Steps to run your program: 3](#_Toc97773389)

[Step 0: Navigate inside [java-project/AOS\_Proj1/src] 3](#_Toc97773390)

[Mid Server: 3](#_Toc97773391)

[Step 1: Start Mid-Server by running the command inside: 3](#_Toc97773392)

[Step 2: Enter any free port number: 3](#_Toc97773393)

[Group Server: 3](#_Toc97773394)

[Step 1: Start this program for each Group that you want to add with command: 3](#_Toc97773395)

[Step 2: Group program will now prompt you to input IP address of Mid Server along with its specific port. 3](#_Toc97773396)

[Step 3: Once connected it asks us what kind of server we are creating 3](#_Toc97773397)

[Sender: 4](#_Toc97773398)

[Step 1: Start this program for each Sender that you want to add with command: 4](#_Toc97773399)

[Step 2: Sender program will now prompt you to input IP address of Mid Server along with its specific port. 4](#_Toc97773400)

[Step 3: Input 1 to enter credentials and type CLOSE at any time to close sender socket. 4](#_Toc97773401)

[Step 4: If you enter 1 in above step this is how the menu looks like: 4](#_Toc97773402)

[Step 5: User can type close to exit connection. 4](#_Toc97773403)

[Architecture 5](#_Toc97773404)

[Mid-Server ⬄ End Groups 5](#_Toc97773405)

[Mid-Server ⬄ Sender 5](#_Toc97773406)

[Working 6](#_Toc97773407)

[PatelP1MidServer.java 6](#_Toc97773408)

[PatelP1Sender.java 6](#_Toc97773409)

[PatelP1GroupServer.java 6](#_Toc97773410)

# Steps to run your program:

### Step 0: Navigate inside [java-project/AOS\_Proj1/src]

## Mid Server:

### Step 1: Start Mid-Server by running the command inside:

javac PatelP1MidServer.java && java PatelP1MidServer

Initially it will ask you to input port on which this server application will start like this:

### Step 2: Enter any free port number:

Text

Description automatically generated

This should show the IP address to which all the groups and senders will connect.

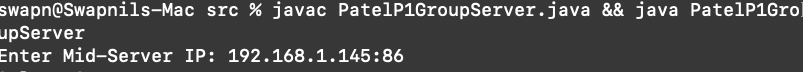
**\*Note: If this doesn’t show the correct IP but the loopback address like 127.0.0.1 then get the IP address of the server machine using system command before proceeding ahead.**

## Group Server:

### Step 1: Start this program for each Group that you want to add with command:

javac PatelP1GroupServer.java && java PatelP1GroupServer

### Step 2: Group program will now prompt you to input IP address of Mid Server along with its specific port.



### Step 3: Once connected it asks us what kind of server we are creating

Text

Description automatically generated

Group is now read to connect with all of its senders once they are available.

## Sender:

### Step 1: Start this program for each Sender that you want to add with command:

javac PatelP1Sender.java && java PatelP1Sender

### Step 2: Sender program will now prompt you to input IP address of Mid Server along with its specific port.

### Step 3: Input 1 to enter credentials and type CLOSE at any time to close sender socket.

Graphical user interface, text

Description automatically generated

### Step 4: If you enter 1 in above step this is how the menu looks like:

Text

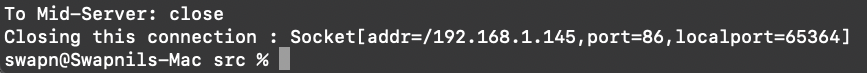
Description automatically generated

#### Step 4.1: User can choose to purchase any item by entering its serial number.

A black screen with white text

Description automatically generated with low confidence

### Step 5: User can type close to exit connection.



# Architecture

## Mid-Server ⬄ End Groups

Diagram

Description automatically generated

## Mid-Server ⬄ Sender

Diagram

Description automatically generated

# Working

## PatelP1MidServer.java

1. Take port input from user and start ServerSocket and that port.
2. Read userList.txt into the AUTH\_LIST object map.
3. We created AuthCred class to store all user records in object-oriented fashion.
4. Now it loops and accepts all the socket info, be it a Sender or a Group.

3.1 If it’s a group verify socket and add its information into GROUP\_LIST Map.

3.2 If it’s a sender verify the socket and start new thread for further communication using thread pool.

3.2.1 Ask sender for Login credentials and match it against AUTH\_LIST if match found start its communication with specific group socket.

3.2.2 Sender and specific group can freely send text to each other, and sender can close this thread by sending “close” or closing socket.

5. PatelP1MidServer and MyNode class that creates new thread for each client share some common data. That is done by extending the same class CommonMidData.

## PatelP1Sender.java

1. Get Mid-Server IP and Port and establish a socket connection.
2. Keep on reading from MidServer and send inputs to it as per Mid-Servers instruction.
3. If the MidServer is not able to map you with any of its group, the socket will end there with proper message.
4. If user input is “close” then close the socket.
5. Socket in this class runs in the main thread.

## PatelP1GroupServer.java

1. Get Mid-Server IP and Port and establish a socket connection.
2. Ask from user what type of group server they want to establish. Gold/Silver/Platinum.
3. Once group type is chosen, load the predefined dataset from items 2d array and send it to other end of socket and wait for input.
4. If a valid int value is received, then the item quantity is reduced by 1 else just the message from the other end is printed.
5. Socket in this class runs in the main thread.