

Project 4 Report

Yashdeep Deswal - Rebecca Mantione

Both	
StoreServer.java, StoreThread.java	Multithreading
Account.java AdminAccount.java CustomerAccount.java	Server design: StoreThread and Account classes
CustomerScene.java	GUI design: Scenes
Individual 1: Yashdeep	
ServerChat.java LoginScene.java (add button)	Chat Apps
AccountsReader.java + data Test_AccountsReader.java	XML data file design JUnit testing
login method changePassword method	Server design: StoreThread and Account classes Server design: StoreThread and Account classes
viewOrders method Test_viewOrders.java	Server design: StoreThread and Account classes JUnit testing
ViewOrdersScene.java	GUI design: Scenes
Individual 2: Rebecca	
CustomerChat.java SettingsScene.java (add button)	Chat Apps
InventoryReader.java + data Test_InventoryReader.java	XML data file design JUnit testing
sendInventory method Test_sendInventory.java	Server design: StoreThread and Account classes JUnit testing
getOrder method Test_getOrder.java	Server design: StoreThread and Account classes JUnit testing
PlaceOrderScene.jav	GUI design: Scenes

Accounts Reader Test:

The screenshot shows the Eclipse IDE interface with the JUnit view selected. The status bar at the top indicates "Finished after 0.243 seconds". Below it, the results summary shows "Runs: 1/1" with "Errors: 0" and "Failures: 0". A green progress bar is shown. The main area displays the Java code for the `Test_AccountsReader` class, which contains a single test method that reads an XML file and asserts its contents against expected values.

```
10 import static org.junit.jupiter.api.Assertions.*;
11
12 class Test_AccountsReader {
13     @Test
14     void test() {
15         String accounts = AccountsReader.readFile("Accounts.xml");
16         List<Account> accountsList = new ArrayList<Account>();
17         accountsList.add(Account.fromXML(accounts));
18         assertEquals(accountsList.size(), 2);
19         assertEquals(accountsList.get(0).getUsername(), "Admin1");
20         assertEquals(accountsList.get(0).getPassword(), "Admin99");
21         assertEquals(accountsList.get(0).getAccountType(), "AdminAccount");
22         assertEquals(accountsList.get(1).getUsername(), "Customer1");
23         assertEquals(accountsList.get(1).getPassword(), "Customer99");
24         assertEquals(accountsList.get(1).getAccountType(), "CustomerAccount");
25     }
26 }
```

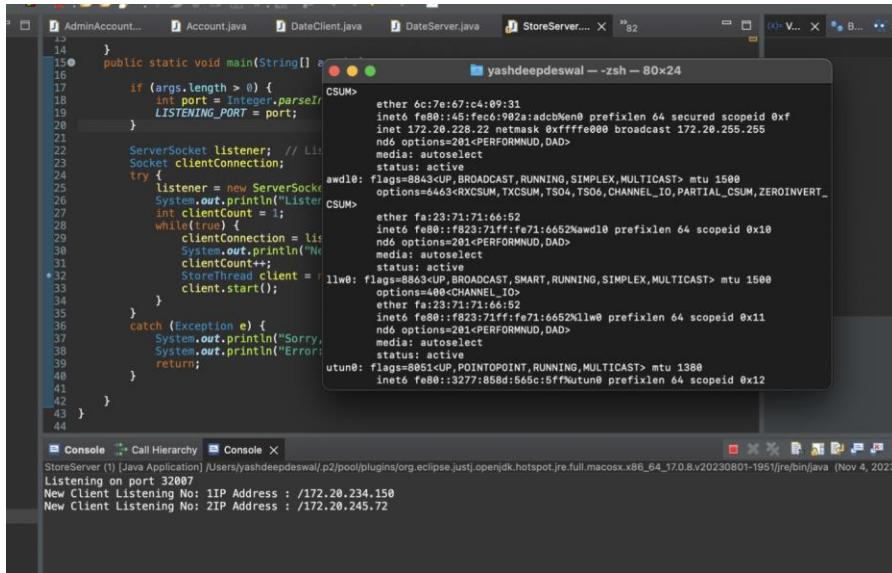
Inventory Reader Screenshot:

The screenshot shows the Eclipse IDE interface with the JUnit view selected. The status bar at the top indicates "Finished after 0.709 seconds". Below it, the results summary shows "Runs: 1/1" with "Errors: 0" and "Failures: 0". A green progress bar is shown. The main area displays the Java code for the `Test_InventoryReader` class, which contains a single test method that reads an XML file and asserts its contents against expected values.

```
10 import static org.junit.jupiter.api.Assertions.*;
11
12 class Test_InventoryReader {
13     @Test
14     void test() {
15         String inventory = InventoryReader.readFile("Inventory.xml");
16         List<InventoryItem> inventoryList = new ArrayList<InventoryItem>();
17         inventoryList.add(InventoryItem.fromXML(inventory));
18         assertEquals(inventoryList.size(), 1);
19         assertEquals(inventoryList.get(0).getName(), "Laptop");
20         assertEquals(inventoryList.get(0).getPrice(), 1000);
21         assertEquals(inventoryList.get(0).getStockLevel(), 50);
22     }
23 }
```

3 Terminal Screenshots

Server:



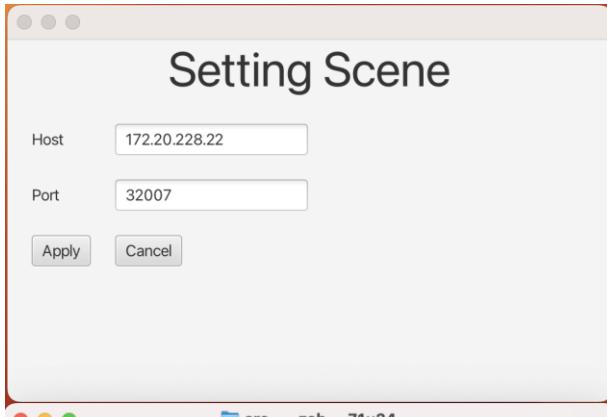
```
14 }
15     public static void main(String[] args) {
16         if (args.length > 0) {
17             int port = Integer.parseInt(args[0]);
18             LISTENING_PORT = port;
19         }
20     }
21     ServerSocket listener; // Listener
22     Socket clientConnection;
23     try {
24         listener = new ServerSocket(LISTENING_PORT);
25         System.out.println("Listener Started");
26         int clientCount = 1;
27         while(true) {
28             clientConnection = listener.accept();
29             System.out.println("New Client Listening No: " + clientCount);
30             clientConnection.setSoTimeout(5000);
31             clientCount++;
32             StoreThread client = new StoreThread(clientConnection);
33             client.start();
34         }
35     } catch (Exception e) {
36         System.out.println("Sorry, " + e.getMessage());
37         System.out.println("Error: " + e);
38         return;
39     }
40 }
41
42 }
43 }
44 }
```

CSUM> ether 6c:7e:67:c4:09:31
inetd fe80::45:fec6:902a:adcb%en0 prefixlen 64 secured scopeid 0xf
inet 172.20.228.22 netmask 0xffffffff broadcast 172.20.255.255
ndo options=201<PERFORMNUD,DAD>
media: autoselect
status: active
awd10: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=643<RXCSUM,TXCSUM,TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
ether fa:23:71:71:66:52
inetd fe80::f023:71ff:fe71:6652%awd10 prefixlen 64 scopeid 0x10
ndo options=201<PERFORMNUD,DAD>
media: autoselect
status: active
llw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=643<CHANNEL_IO>
ether fa:23:71:71:66:52
inetd fe80::f023:71ff:fe71:6652%llw0 prefixlen 64 scopeid 0x11
ndo options=201<PERFORMNUD,DAD>
media: autoselect
status: active
utun0: flags=8851<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1388
inetd fe80::3277:858d%utun0 prefixlen 64 scopeid 0x12

Console Call Hierarchy Console

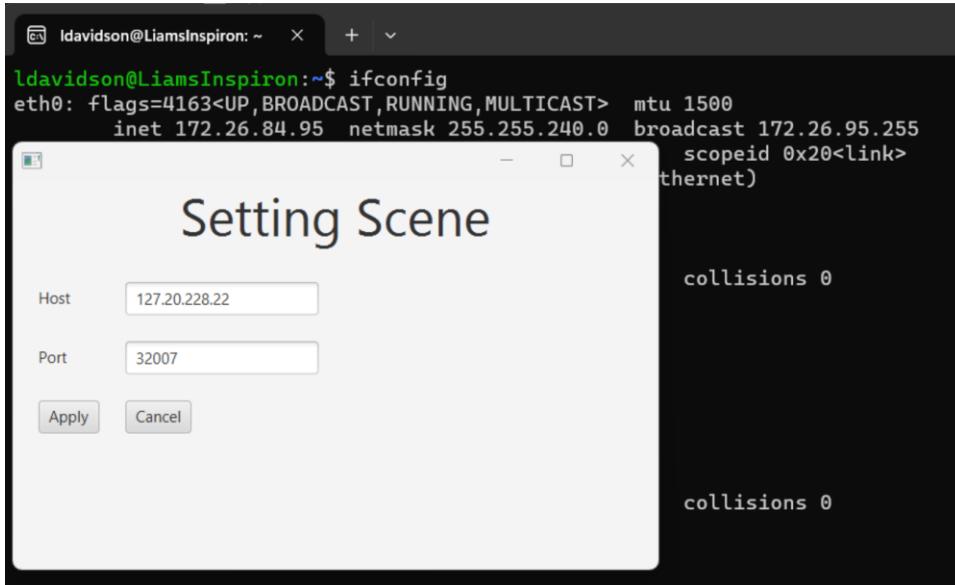
StoreServer (1) [Java Application] /Users/yashdeepdeswal/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.8.v20230801-195/jre/bin/java (Nov 4, 2023)
Listening on port 32007
New Client Listening No: IP Address : /172.20.234.150
New Client Listening No: IP Address : /172.20.245.72

Client 1:

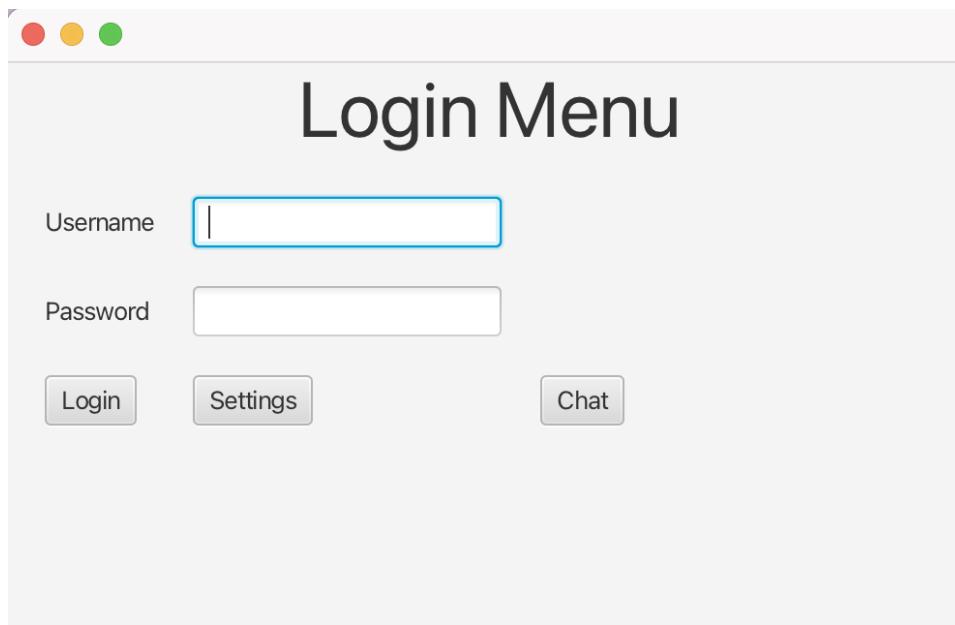


```
src -- zsh -- 71x24
inet6 fe80::1036:ba2b:b315:4694%en0 prefixlen 64 secured scopeid 0xc
inet 172.20.234.150 netmask 0xfffffe000 broadcast 172.20.255.255
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: active
awdl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=6463<RXCSUM,TXCSUM,TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZERO_INVERT_CSUM>
ether 16:5f:9f:4e:46:64
inet6 fe80::145f:9fff:fe4e:4664%awdl0 prefixlen 64 scopeid 0xd
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: active
llw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=400<CHANNEL_IO>
ether 16:5f:9f:4e:46:64
inet6 fe80::145f:9fff:fe4e:4664%llw0 prefixlen 64 scopeid 0xe
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: inactive
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
inet6 fe80::16c9:f7eb:297b:2951%utun0 prefixlen 64 scopeid 0xf
nd6 options=201<PERFORMNUD,DAD>
```

Client 2:



Login Scene:



Chat:

A screenshot of a "Two-user Networked Chat" application. It shows two windows: one for connecting to localhost on port 1501 and another for listening on port 1501. Both windows show a transcript of a conversation where two users exchange "Hi" and "how are you doing".

Two-user Networked Chat

CONNECTING TO localhost ON PORT 1501

CONNECTION ESTABLISHED

You: Hi
RECEIVE: how are you doing

Your Message: Hi

Two-user Networked Chat

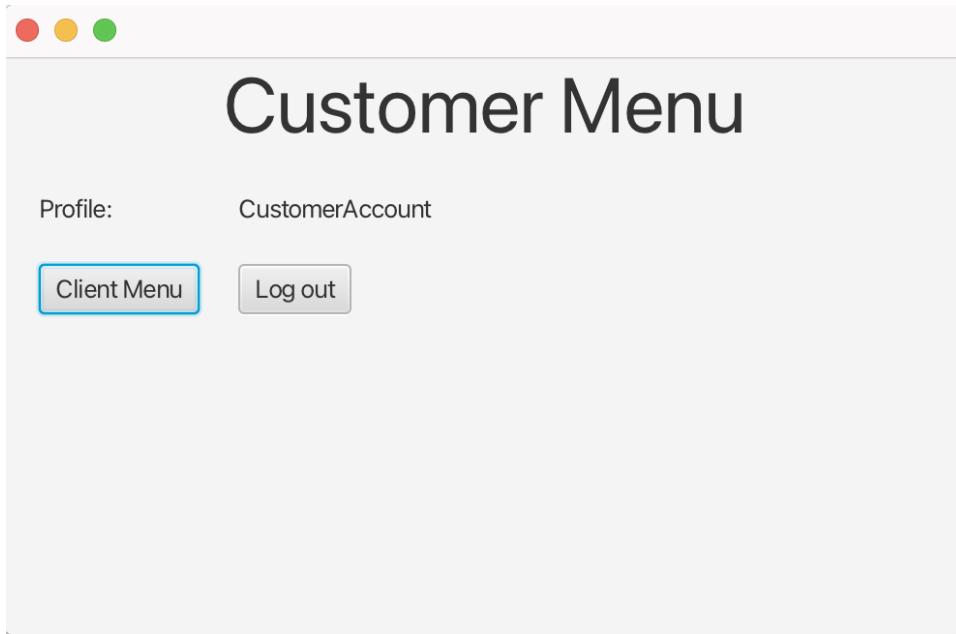
LISTENING ON PORT 1501

CONNECTION ESTABLISHED

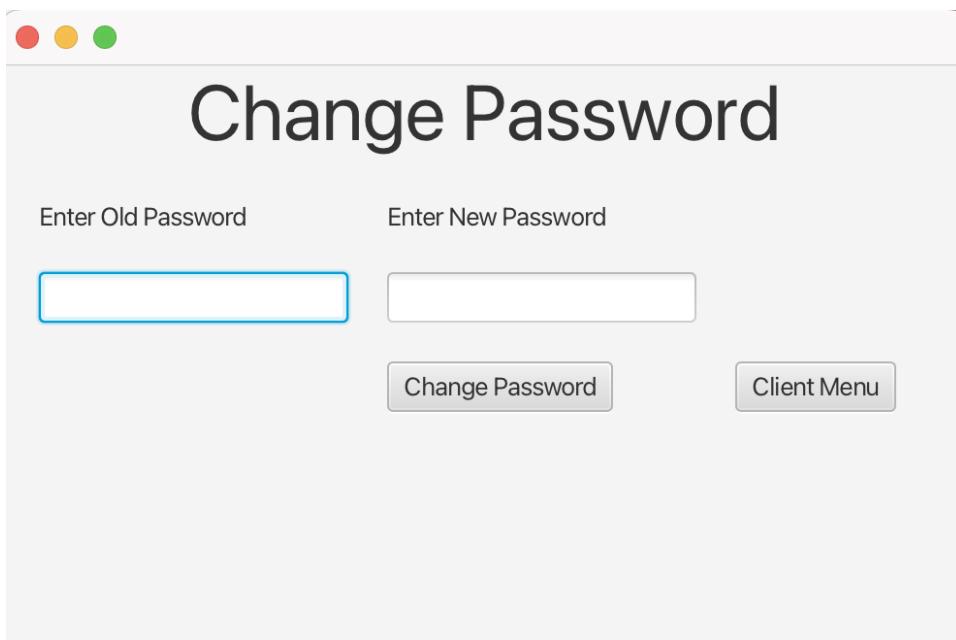
RECEIVE: Hi
SEND: how are you doing

Your Message: how are you doing

Show Profile:



Change Password:



Place Orders:

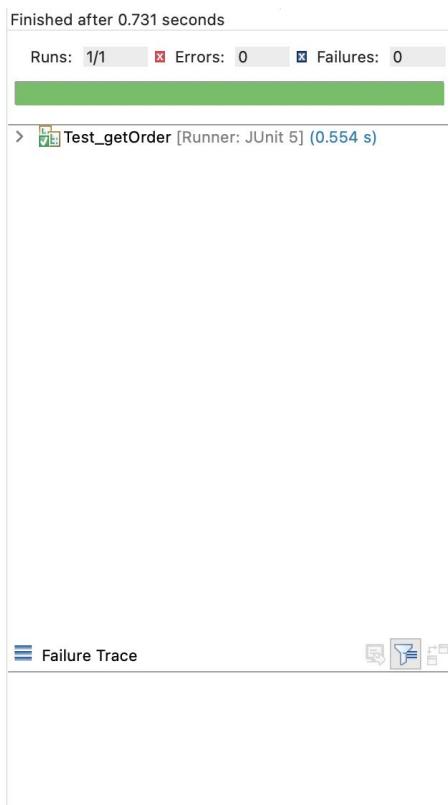
The screenshot shows a window titled "Ordering". Inside, there's a table header "Stock # Description" followed by two rows of data: "45 Thingamajig" and "46 Thingamajig2". Below the table are input fields for "Stock #" and "Quantity", each with a corresponding text input box. At the bottom are "Submit" and "Cancel" buttons.

Stock #	Description
45	Thingamajig
46	Thingamajig2

Stock #

Quantity

JUnit Test getOrders:



JUnit Test sendInventory:

