CT414 Java RMI Assignment 1

Seán Finnegan ECE 19395091

Java RMI Working

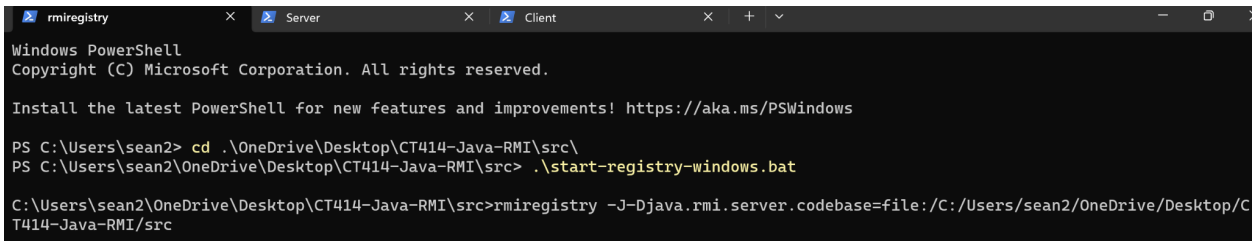

Fig. 1: rmiregistry working

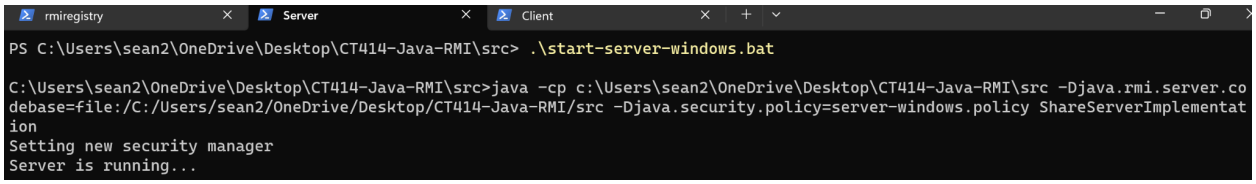

Fig.2: Server working

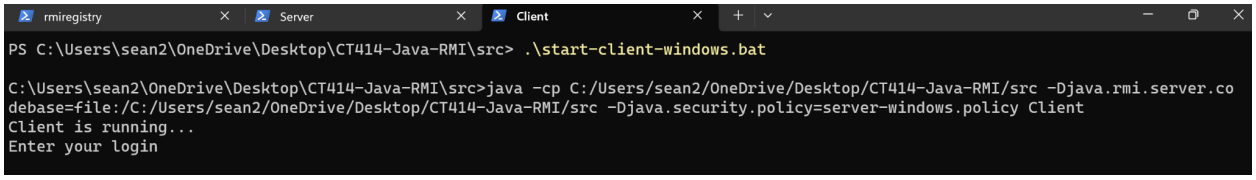

Fig. 3: Client working

Testing Methodology

In order to test if the Distributed System was working correctly I threw two exceptions: AuthenticationFailed and InsufficientFundsException.

AuthenticationFailed is triggered if (a) the login details are invalid or (b) the authentication token has expired. As per the assignment instructions the token expires after 5 minutes.

InsufficientFundsException is triggered if (a) the amount sought to be withdrawn is less than the current balance of the trading account and/or (b) the amount of shares sought to be purchased exceeds the balance of the trading account.

Other testing including using the debug feature in VSCode as I experienced initial problems with reading in my .csv file. I also used several print statements throughout to aid in understanding where there may be an issue.

Client <> Server Interaction

```
Client is running...
Enter your login
user
Enter your password
password
Finnegan's Trading Simulator
Main menu:
1 - Check balance
2 - Deposit funds
3 - View available shares
4 - Purchase shares
5 - My shares
6 - Withdraw funds
7 - Exit simulator
```

```
Enter your login
password
Enter your password
l
Error: Login details incorrect
```

```java
//Create token on successful login
@Override
public Long login(String login, String password) throws AuthenticationFailed, RemoteException {
    if(this.login.equals(login) && this.password.equals(password)) {
        return token;
    } else {
        throw new AuthenticationFailed(message: "Error: Login details incorrect");
    }
}
```

Fig. 4: Logging into the server and getting the access token. Exception is thrown if login is incorrect

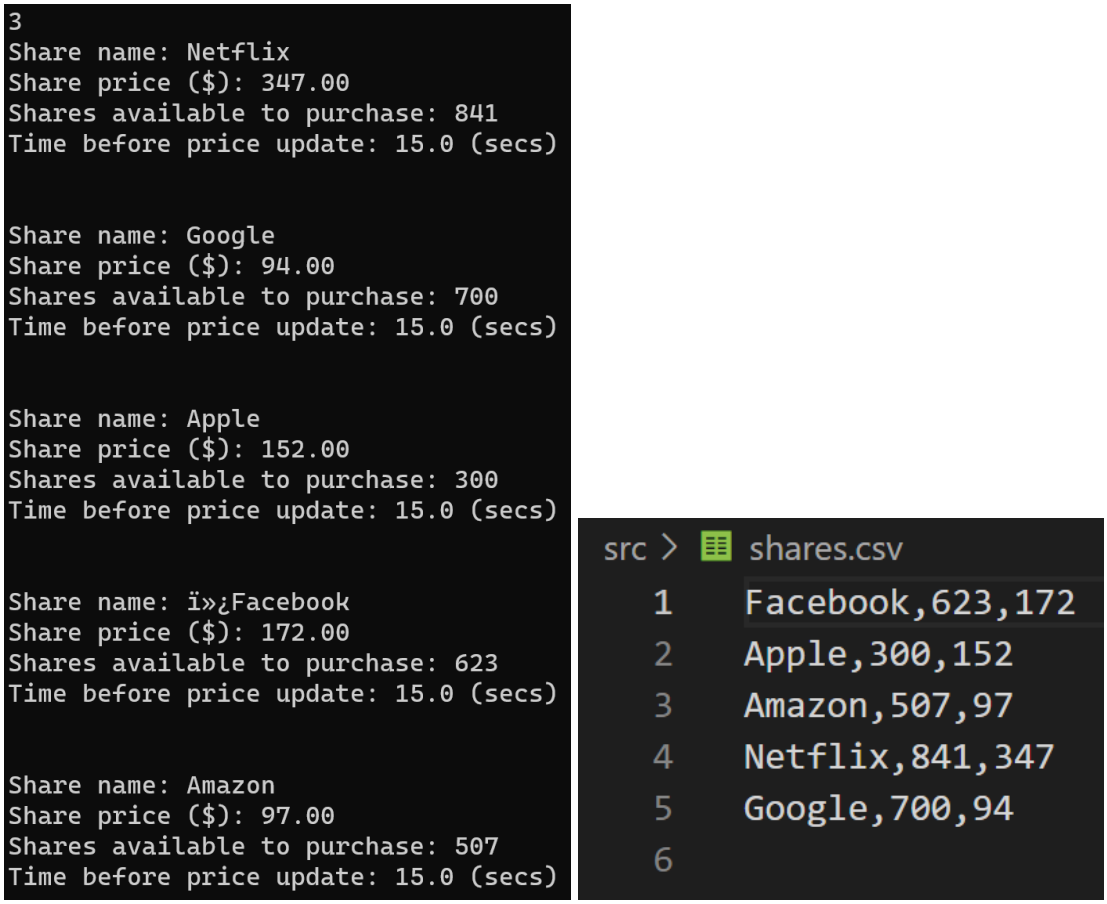

Fig. 5: Download and print out a summary of all the Shares available on the system. .csv file containing the share information

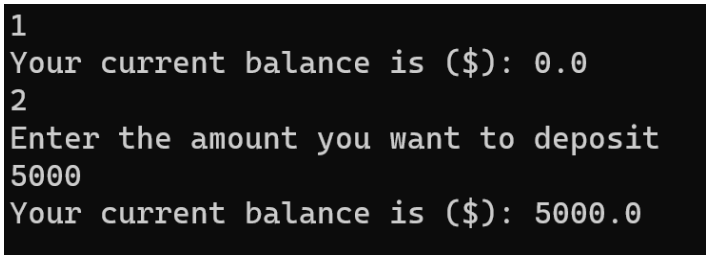

Fig. 6: Check balance and deposit funds into the trading account on the server

```
4
Enter name of share:
Google
Enter number of shares:
20
Share(s) purchased successfully
4
Enter name of share:
Amazon
Enter number of shares:
10
Share(s) purchased successfully
```

Fig. 7: Making purchase

```
5
Share name: Google
Share price ($): 70.84
Number of owned shares: 20


Share name: Amazon
Share price ($): 112.50
Number of owned shares: 10
```

Fig. 8: Download and print a summary of ShareHolding on the server

```
Your current balance is ($): 5000.0
6
Amount to withdraw ($):
2000
Your current balance is ($): 3000.0
```

```
Your current balance is ($): 3000.0
6
Amount to withdraw ($):
5000
Error: Balance cannot be lower that amount withdrawn
```

Fig. 9: Withdraw funds from trading account. InsufficientFundsException thrown if balance< withdrawn

```
Share name: Amazon
Share price: 41.59
Shares available to purchase: 507
Time before price update: 17.0 (secs)
```

```
Share name: Amazon
Share price: 41.59
Shares available to purchase: 507
Time before price update: 3.0 (secs)
```

```
Share name: Amazon
Share price: 24.28
Shares available to purchase: 507
Time before price update: 28.0 (secs)
```

```java
//5 minute token
private static void countdown() {
    final Runnable tokenTimeout = new Runnable() {
        @Override
        public void run() {
            token = new Random().nextLong();
        }
    };
    //call every 5 minutes
    countdown.scheduleAtFixedRate(tokenTimeout, initialDelay: 0, period: 5, MINUTES);
}
```

Fig. 10 Price update countdown and price change

```
Error: Session timeout
Enter your login
```

```
7
PS C:\Users\sean2\OneDrive\Desktop\CT414-Java-RMI\src>
```

Fig. 11: Exit server and server timeout. Server tracks login activity