/**

* **Programmer:** Razia Sultana Patan

* **Project:** Distributed Cryptocurrency Trading System using Java RMI

* **Environment:** any machine with java setup.

* **Files Included** PatanP2Server.java, PatanP2CryptoCoinServant.java,
*PatanP2Coin.java, PatanP2CryptoCoinServicesInterface.java,
*PatanP2CoinInteface.java, PatanP2ClientModel.java, userList.txt,
*PatanP2Client.java

* **Purpose:** The authorized user can use 'Distributed Cryptocurrency Trading
*Application' to sell and buy coins which are created by the server.

* **Input:** Messages from user in PatanP2Client.java, list of authorized users in
* userList.txt

* **Preconditions:** User credentials must present in userlist.txt for a user to use the
*application.

* **Output:** The trading interactions between Client and the Server.

* **Postconditions:** If user enters invalid inputs, he/she will not be allowed to trade his/her coins.

* **Document:** This Document talks about algorithm, steps for running the application, Output screen shots, Class diagram and the flow chart for the application.

**/

Algorithm:

PatanP2CoinInteface:

This interface **extends Remote** Interface and has following members, all related to a crypto coin:

```
     public String getName() throws RemoteException;
     public void setName(String name) throws RemoteException;
     public String getAbbreviatedName()  throws
RemoteException;
     public void setAbbreviatedName(String abbreviatedName)
throws RemoteException;
     public String getDescription()  throws RemoteException;
     public void setDescription(String description) throws
RemoteException;
     public long getMarketcap() throws RemoteException;
     public void setMarketcap(long marketcap) throws
RemoteException;
     public long getTradingVolume() throws RemoteException;
     public void setTradingVolume(long tradingVolume) throws
RemoteException;
     public float getOpeningPrice()  throws RemoteException;
     public void setOpeningPrice(float openingPrice) throws
RemoteException;
     public String getTimestamp() throws RemoteException;
     public void setTimestamp(String timestamp) throws
RemoteException;
```

PatanP2Coin:

This class implements **PatanP2CoinInteface** and **extends UnicastRemoteObject** , It implements all the methods of PatanP2CoinInterface and has Constructor with parameters.

PatanP2ClientModel:

This Class is **Seralizable** class, whose members are used to hold client state, it has constrctor with parameters and getter setter methods:

```
private String clientID;
private String clientName;
private float purchasePower;
private ArrayList<PatanP2CoinInterface> coins;
```

PatanP2CryptoCoinServicesInterface:

This interface **extends Remote** Interface and has following members, all related to a crypto coin services

```
boolean   buy (String coinAbbreviation, float price, String username) throws RemoteException, FileNotFoundException, IOException;

boolean  sell (String coinAbbreviation, float price, String username) throws RemoteException , FileNotFoundException, IOException;

boolean autheticationCheck (String username, String password) throws IOException, RemoteException;

PatanP2ClientModel getClientStateFromFiles (String clientID) throws RemoteException;
```

PatanP2CryptoCoinServant:

- This class implements **PatanP2CryptoCoinServant** and **extends UnicastRemoteObject** , It implements all the methods of PatanP2CryptoCoinServicesInterface.
- **synchronized boolean buy**(String coinAbbreviation, float price, String username)
    1. The buy operation is enabled only if the price is same as the price of the coin and the coin exist at server side coins.

2. If price matches, it retrieves the client state using getClientStateFromFiles(username) and if the client is the new user, then it creates PatanP2ClientModel object for the client where its adds the bought coin and updating the purchase power of the client with trading amount.
3. It stores the client state using storeClientState(String clientID, PatanP2ClientModel clientState) method which serializes client object and stores it in a file ClientId.ser at server side for each client.
4. Once the trading operation is done, it updates the coin information using editcoin, in this application it subtracts trading amount from the volume of the coin.
5. If the coin mentioned is no present or price entered is not the price of coin, it returns false.

- **synchronized boolean sell**(String coinAbbreviation, float price, String username)
    1. The sell operation is enabled only if the price is same as the price of the coin and the coin exist at server side coins.
    2. If price matches, it retrieves the client state using getClientStateFromFiles(username) and if the requested coin to sell is already present for the client, it removes the coin from coins of the client and updating the purchase power of the client with trading amount.
    3. It stores the modified client state using storeClientState(String clientID, PatanP2ClientModel clientState) method which serializes client object and stores it in a file ClientId.ser at server side for each client.
    4. Once the trading operation is done, it updates the coin information using editcoin, in this application it adds the trading amount to the volume of the coin.
    5. If the coin mentioned is no present or price entered is not the price of coin or the client is new user, it returns false.
- **boolean autheticationCheck**(String username, String password) retrieves the users data from "userList.txt" file and checks if the username and password are valid.

- **PatanP2ClientModel getClientStateFromFiles**(String clientID) reads the requested client sate into PatanP2ClientModel from clientID+".ser" file and if file doesn't exist, it returns null.
- Apart from methods of PatanP2CryptoCoinServicesInterface, it also has other methods like

  getCoins(), editCoin(), removeCoin(), addCoin(), unBinder(), storeClientState()

- **storeClientState**(String clientID, PatanP2ClientModel clientState) method serializes the client data into clientID+"ser" file stored at the server side.
- It has two HashMaps, one for storing the users from a file for login service and another for storing coins created by server which is static to provide coin data to all the authenticated clients.

  Private Map<String, Object> coins

  Private Map<String,String> userList

PatanP2Server:

- PatanP2Server creates coin by using PatanP2Coin constructor and uses PatanP2CryptoCoinServant services like addCoin(), getCoins().
- It binds servant class with url "rmi://localhost/"+"servant" in order to enable client for buy , sell, login, viewclientstate service.
- It binds all the crated coins with url : "rmi://localhost/"+"coin"+String.valueOf(j) where j is the index for a coin. In this way each coin has their own url.

PatanP2Client:

- PatanP2Client looks for "rmi://localhost/"+"servant" in rmiregistry and uses autheticationCheck() method to validate credentials entered by the user.
- Once user is authenticated, client provides following services:

System.out.println("**********Services**********
**********"); System.out.println("Press 1 to display coins information");
System.out.println("Press 2 to buy a coin");
System.out.println("Press 3 to sell a coin");
System.out.println("Press 4 to view your account details");
System.out.println("Press 5 to exit");

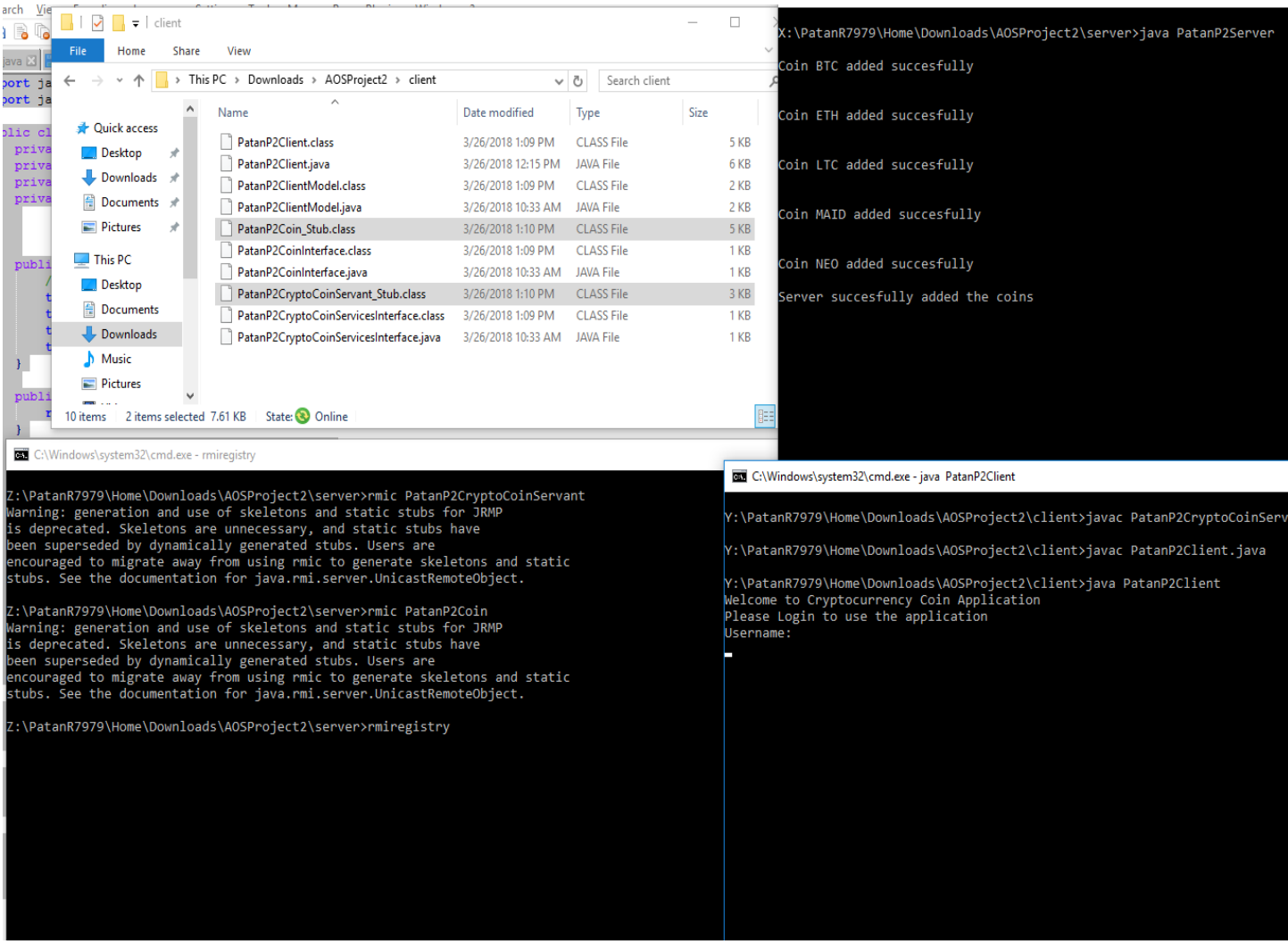
- If client enters 1: It looksfor url = "rmi://localhost/"+"coin"+String.valueOf(i) iteratively with index value i into **PatanP2CoinInterface** and displays the coins information to the user.
- If client enters 2: It looksfor url = "rmi://localhost/"+"servant" into **PatanP2CryptoCoinServicesInterface** and passes the coin abbreviated name and amount to buy() method. Upon successful buying, it displays success message to the user.
- If client enters 3: It looksfor url = "rmi://localhost/"+"servant" into **PatanP2CryptoCoinServicesInterface** and passes the coin abbreviated name and amount to sell() method. Upon successful selling, it displays success message to the user.
- If client enters 4: It looksfor url = "rmi://localhost/"+"servant" into **PatanP2CryptoCoinServicesInterface** and uses **`getClientStateFromFiles`** () method to display the users details like purchase power and coins the user bought.
- If client enters 5: this exists the application. As all the trading activities of a client are stored as clientID.ser file at server side, Next time when he/she logs in the client state is retrieved from .ser files using **`getClientStateFromFiles`** of **PatanP2CryptoCoinServicesInterface.**

Steps for Implementing the Application:

1. Place **PatanP2Client.class, PatanP2CryptoCoinServicesInterface.class, PatanP2CoinInteface.class, PatanP2ClientModel.class** files in **Client** folder.
2. Place **PatanP2Server.class, PatanP2CryptoCoinServant.class, PatanP2Coin .class PatanP2CryptoCoinServicesInterface.class, PatanP2CoinInteface.class, PatanP2ClientModel.class** files in **Server** folder.

3. run command **rmic PatanP2CryptoCoinServant** and **rmic PatanP2Coin** at server folder using command prompt.
4. Copy **PatanP2CryptoCoinServant_Stub.class, PatanP2Coin_Stub.class** into the client folder
5. Run **rmiregistry** command at server folder folder using command prompt.
6. Open a new command prompt window at server folder and run the server using **java PatanP2Server**
7. Open command prompt window at client folder and run client using **java PatanP2Client.**

Output Screenshots:



Client 1:

```
Y:\PatanR7979\Home\Downloads\AOSProject2\client>java PatanP2Client
Welcome to Cryptocurrency Coin Application
Please Login to use the application
Username:
anna
Password:
a86H6T0c
!!!! Invalid credentials !!!!
Please Login to use the application
Username:
anna
Password:
a86H6T0c
Succesfully Logged In!!
**********Services**********
Press 1 to display coins information
Press 2 to buy a coin
Press 3 to sell a coin
Press 4 to view your account details
Press 5 to exit
1
BTC Price: $8525.77 Volume: $5446460000 MarketCap: $144281605710
ETH Price: $612.59 Volume: $1429600000 MarketCap: $60174824420
NEO Price: $69.96 Volume: $115780000 MarketCap: $4547231000
LTC Price: $169.78 Volume: $448359000 MarketCap: $9449752582
MAID Price: $0.299873 Volume: $912932 MarketCap: $135708249
**********Services**********
Press 1 to display coins information
Press 2 to buy a coin
Press 3 to sell a coin
Press 4 to view your account details
Press 5 to exit
2
Enter the coin(abbreviation) you wish to buy
BTC
Enter the amount
8525.77
8525.77
true
Succesfully bought the coin!!
BUY Servers
**********Services**********
Press 1 to display coins information
Press 2 to buy a coin
Press 3 to sell a coin
Press 4 to view your account details
Press 5 to exit
4
____Username: anna    Purchase: 8525.76953125
```

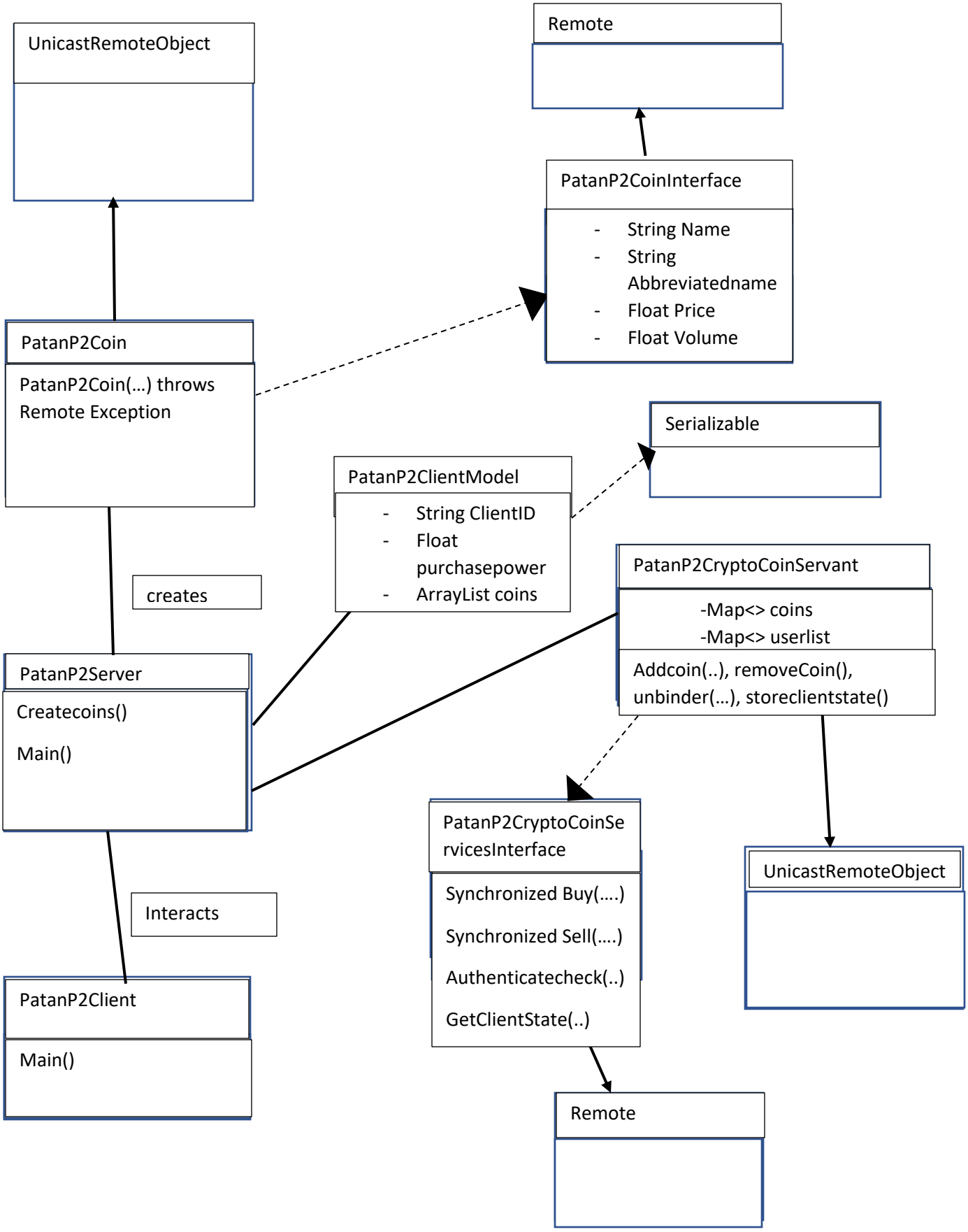
Client 2:

```
Welcome to Cryptocurrency Coin Application
Please Login to use the application
Username:
barbara
Password:
G6M7p8az
Succesfully Logged In!!
**********Services**********
Press 1 to display coins information
Press 2 to buy a coin
Press 3 to sell a coin
Press 4 to view your account details
Press 5 to exit
1
BTC Price: $8525.77 Volume: $5446451475 MarketCap: $14428160
ETH Price: $612.59 Volume: $1429600000 MarketCap: $601748244
NEO Price: $69.96 Volume: $115780000 MarketCap: $4547231000
LTC Price: $169.78 Volume: $448359000 MarketCap: $9449752582
MAID Price: $0.299873 Volume: $912932 MarketCap: $135708249
**********Services**********
Press 1 to display coins information
Press 2 to buy a coin
Press 3 to sell a coin
Press 4 to view your account details
Press 5 to exit
1
BTC Price: $8525.77 Volume: $5446451475 MarketCap: $14428160
ETH Price: $612.59 Volume: $1429600000 MarketCap: $601748244
NEO Price: $69.96 Volume: $115780000 MarketCap: $4547231000
LTC Price: $169.78 Volume: $448359000 MarketCap: $9449752582
MAID Price: $0.299873 Volume: $912932 MarketCap: $135708249
**********Services**********
Press 1 to display coins information
Press 2 to buy a coin
Press 3 to sell a coin
Press 4 to view your account details
Press 5 to exit
2
Enter the coin(abbreviation) you wish to buy
NEO
Enter the amount
69.96
69.96
true
Succesfully bought the coin!!
BUY Servers
**********Services**********
Press 1 to display coins information
Press 2 to buy a coin
Press 3 to sell a coin
Press 4 to view your account details
Press 5 to exit
4
____Username: barbara    Purchase: 69.95999908447266
NEO Price: $69.96Purchased date:2018.03.26.13.12.09
**********Services**********
Press 1 to display coins information
Press 2 to buy a coin
Press 3 to sell a coin
Press 4 to view your account details
Press 5 to exit
5
```

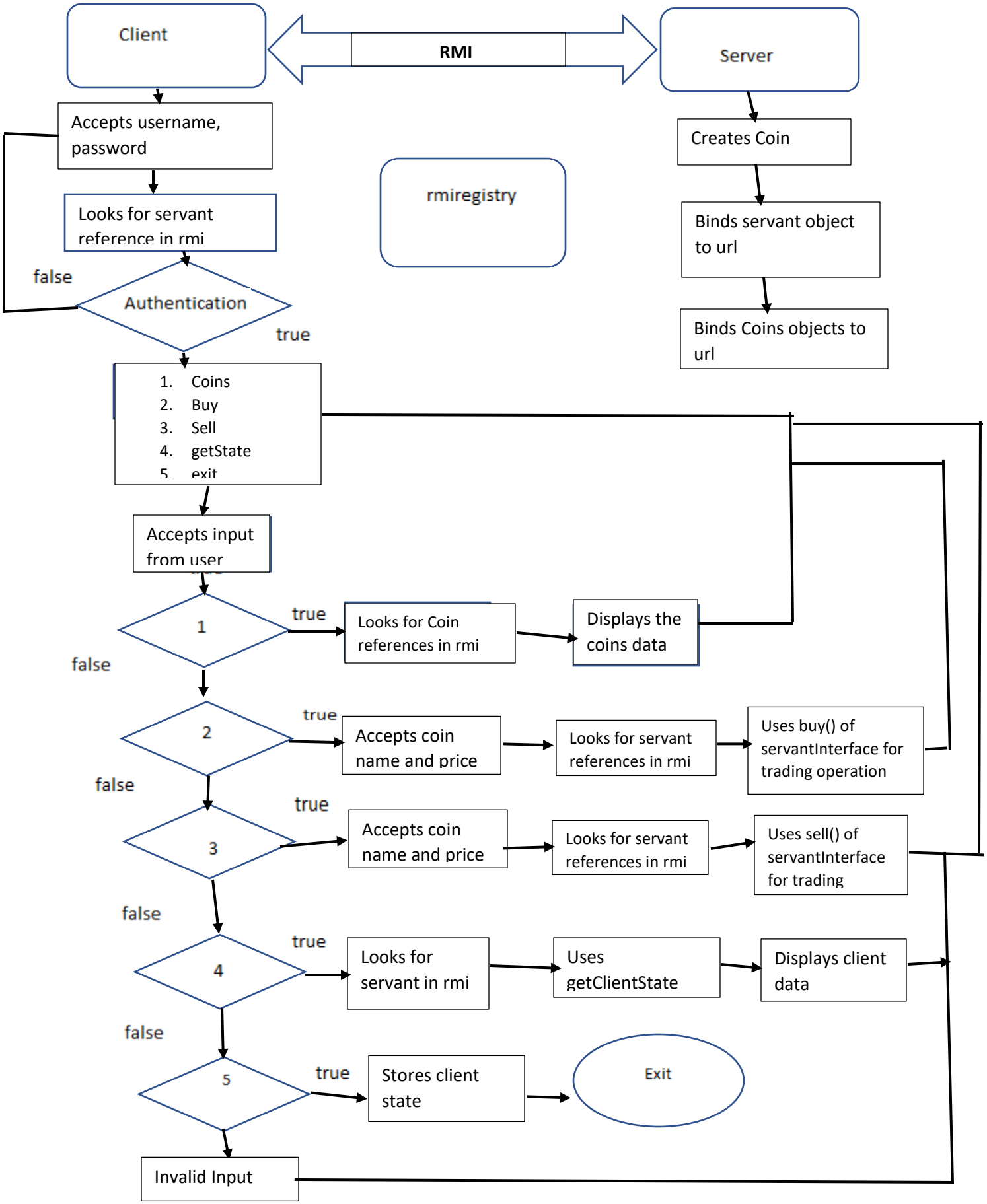
Server:

```
X:\PatanR7979\Home\Downloads\AOSProject2\server>java PatanP2Server

Coin BTC added succesfully


Coin ETH added succesfully


Coin LTC added succesfully


Coin MAID added succesfully


Coin NEO added succesfully

Server succesfully added the coins
Username anna Entered prices are equal
Username anna Client file not found
Username anna ClientStatenull
Username anna Created Client state
Username anna added coin to client

Coin BTC edited succesfully

Username anna Edited the coin volume
Username anna updated the purchase power
Username anna client state saved in server folder
Username barbara Entered prices are equal
Username barbara Client file not found
Username barbara ClientStatenull
Username barbara Created Client state
Username barbara added coin to client

Coin NEO edited succesfully

Username barbara Edited the coin volume
Username barbara updated the purchase power
Username barbara client state saved in server folder
```

Class Diagram:



Flow Chart:


Client
RMI
Server
Accepts username, password
Creates Coin
rmiregistry
Looks for servant reference in rmi
Binds servant object to url
false
Authentication
true
Binds Coins objects to url
1. Coins
2. Buy
3. Sell
4. getState
5. exit
Accepts input from user
1
true
Looks for Coin references in rmi
Displays the coins data
false
2
true
Accepts coin name and price
Looks for servant references in rmi
Uses buy() of servantInterface for trading operation
false
3
true
Accepts coin name and price
Looks for servant references in rmi
Uses sell() of servantInterface for trading
false
4
true
Looks for servant in rmi
Uses getClientState
Displays client data
false
5
true
Stores client state
Exit
Invalid Input