# CSCI 5531: Advanced Operating Systems
# Project-2 [KSH]
# Distributed Cryptocurrency Trading System

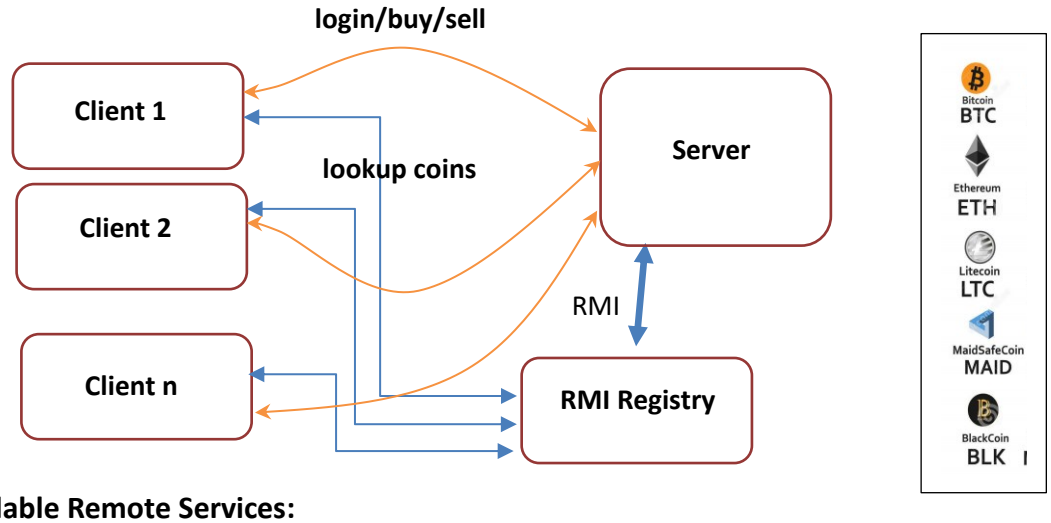
Spring 2022
April 6, 2022
100 pts

## 1. Objective

The objective of this second programming project is to learn Java RMI API to implement distributed applications. In this project, you will be utilizing distributed Java RMI registry and implementing client-server interaction using Remote Objects and Remote Interfaces. After completing the project, you will have an advanced understanding of the steps required to develop distributed applications (OO paradigm) using Java.

## 2. Project Specification

In this programming project, you are required to implement a distributed application using Java for a basic Cryptocurrency Trading system for users to buy and sell crypto coins. It is better to create a user interface for both the server and clients to ease programming aspect for you. The following diagram for the auction illustrates the system flow.




### 2.1. Available Remote Services:

- The server can *add a new Crypto coin* identified by a name, abbreviated name, description, market cap, trading volume, opening price and timestamp. Look at Wikipedia or Coinmarketcap sites for more information about cryptocoin types and others.
- Server will instantiate the CryptoCoin servant class to generate coins with unique name (use abbreviated name) and initial values. Use constructors to initialize variables.
- Server can create new coins, remove coins, and edit coins. Server needs to use bind, rebind, remove, and other operations to complete its intended work.

- When the trading system is ready with coins, a client needs to login with matching credentials to operate in the trading system. Server is responsible to provide the login service.
- After successful login, server identifies and regenerates client's state (deserialize) which includes client name, id, purchase power (amount), all crypto coins the client has purchased along with purchase price, quantity, and timestamps.
- Client is now allowed to buy or sell coins from the market. After each transaction, clients purchase power, coin details needs to be updated accordingly.
- As multiple client can buy and sell coins, you need to be aware of race condition (concurrent access) and adequately implement to handle it.
- The server needs to notify if client requests for its own state. That is, all coins with associated information.
- After the all transactions are complete and clients want to leave, the state needs to be serialized and stored for next session of trading.

## 3. Programming Notes:

I would strongly suggest that everyone *start early* by designing the *interfaces* you need for implementing the trading system. When you have the interfaces, start designing the class hierarchy and focus on encapsulation. You can create one or more remote objects (preferred) to implement all services. To make your object remote, the service/servant class need to implement remote interface(s).

You should program each process to print an informative statement whenever it takes a trading (e.g., adding product, removing product, buying a coin, selling a coin, etc), so that you can see that your remote processes are working correctly (or not!). Besides, one should be able to determine from this output if your remote processes are working correctly. You should hand in screen shots (or file content, if your process is writing to a file) of these informative messages.

Note that we will be ***using our Delta lab machines (Unix/Windows platform)*** in the lab to run and verify your codes. As the goal of the programming project is for you to learn Java RMI programming (not to build a hardened application), you may also assume that there are a set of only 5 cryptocoins in the market and a client can purchase one or all types of coins. You are required to work on this project alone. Any clarifications and revisions to the assignment will be posted on the course UHCL BB page.

### 3.1 File names:

Make sure you follow the file name guideline given below for your project:

lastNameP2XXXXInterface.java, lastNameP2Client.java, lastNameP2Server.java, lastNameP2XXXXService.java, etc.

## 4. Points Distribution:

| Bits and pieces | Points |
|---|---|
| Client Program | 20 |
| Remote Services | 35 |
| Server Program | 35 |
| Documentation and Program Style (Coding style, comments etc.) | 10 |

## 5. Submission Instructions:

This project requires the submission of a *soft copy* and a *hard copy (scan and upload in course BB).*

### 5.1. Soft Copy (Due Wednesday April 6, 2022, 11:59 pm)

The soft copy should consist of all java files and any other API file(s) along with the detailed documentation of the programs. Documentation should include your problem solving approach, UML diagram, discussion of data structures, algorithms, and screen shots of outputs. These must be submitted to course BB system.

**Hard copy (Due Wednesday April 6, 2022, beginning of the class)**

The hard copy of the project should be submitted at the beginning of the **Wednesday April 6, 2022** class. It should include the documentation of the project followed by the source code of the project. It's also require to include the screen shot of outputs in the hard copy. The source code should be very well commented and the documentation should be detail as well to get the full credit.

## 6. Late Penalty:

You have to submit your project on or before the due date to avoid any late penalty. **A late penalty of *15% per day* will be imposed after the *due date***. After one week from the due date, you will not be allowed to submit the project under any circumstances.

Please **DO NOT MODIFY** the project code in any way after the deadline of soft copy submission. The hard copy and the soft copy must be same.

**Good Luck!!**