**Spring and Hibernate**

Orders Management System - A user can place multiple orders in the system (BUY/SELL) for securities.

1. Order service should be able to place buy/sell order for a user. A buy/sell order will have Order Id, Client Id, Security, Price, Quantity, Side, Order Type etc.
2. Order Service should be able to get all the placed orders for a user.
3. Order Service should be able to get all the executed orders for a user.
4. Logging Manager: - This is responsible to log all the order requests (new or cancel) and all the execution messages asynchronously to files. Please use AOP to enable logging in the different methods.
5. Validation Manager: - This is responsible to validate the order.
6. Order Pool: - Order pool is responsible to hold all the orders not executed so far but can be executed with the new orders coming to the system. Use any data structure for now.
7. Execution Manager: - This component is responsible to execute the order based on order pojo and order pool. Based on the order type either it will execute order or do the necessary action associated with order type.
8. Use JUnit to write test cases for Order Service and Validation Manager.
9. Please use AOP to find out timing for each function.
10. Do proper exception handling whenever required and use custom exceptions.
11. We need to make sure that not more than 5 clients should send the orders at same time. If 6th client is trying to connect then that client should wait until one of the existing client disconnect from the system.

**With Transaction**

* Portfolio manager is responsible for post execution settlement. In Post Execution Settlement, Portfolio manager needs to do the following things:-
  + Add/remove the number of stocks purchase/sold by the user.
  + Credit/debit the amount (number of stocks \* stock price) from the users account
* For each execution, portfolio manager will get List of Execution Objects. An Execution Object will have *Execution Id, Order id, Side, Price, Security, Client id*.
* Write a method of Order Service using Spring Transaction which is taking care of above unit of work in transaction.
* Please demonstrate Programmatic and Declarative transaction technique with proper propagation level.