Design Document

PubSubServerInterface
Interface class that lists each method needed for each command the user may pick. These commands are: join, leave, subscribe, unsubscribe, publish, and ping. The class also extends the Remote class for RMI communication.

SubscriberInfo Class
This class simply stores a client's IP and Port number with getters.

PubSubServer Class
This class tracks clients with two data structures. An ArrayList stores SubscriberInfo objects, where one is created for each client upon Join() and it is removed when the client calls Leave(). Our system allows at most 5 clients at a time to join the server and clients are allowed to leave the server at any point in time. We made a decision to have a maximum of 5 clients interact with the server since we didn't want to burden system resources for handling communication from more than 5 clients simultaneously. We also maintain track of which port numbers are currently in use by a particular client. In the off chance that a client is assigned an in-use port number, the client can try to join again via the "join" command defined for the PubSub Client Command Line Interface and a new random port number is assigned to them. A HashMap tracks what clients are subscribed to, with String keys in the form *type*;*originator*;*org*. Subscribe() creates a new key or appends the client's SubscriberInfo object to the existing key-value pair. When an article is published, its first three fields are used to search the map, and all sub combinations of these are used to hash for less specific subscriptions. When a client sends a "publish" command, every client that is subscribed to the article string passed receives the article, including the client that sent the publish command if it has subscribed to that article in the past. We have written functions for performing article format validation for the Publish, Subscribe and Unsubscribe functions that print relevant error messages in the server terminal if an incorrect article format is detected. The client can re-enter the publish, subscribe or unsubscribe command with an appropriate article format via the Client Command Line user interface. The Ping() function checks if the server is live.

PubSubClient Class
This class defines the Command Line user interface that allows the client to interact with the server using 5 commands: join, leave, subscribe, publish, unsubscribe and subscribe (not case-sensitive). Every new client is assigned a new port number which is a randomized number between 1024 and 65535 since every port number between 0 and 1023 are typically used for system TCP/IP communication. We defined two threads: one for making RMI calls to the server and another thread for receiving published articles from the server via UDP. If the server capacity is at 5 and a new client wants to join, then the join() operation is unsuccessful for the new client, but the client does not stop executing. They will have to wait for an existing client to leave before they can join again by entering "join" in the Client Command Line interface we have designed for this application. We also included command line format validation functions that validate the command the user enters for interacting with the server. We enforced a rule that "publish", "subscribe" and "unsubscribe" functions should always have a colon before the article string is entered. The colon helps separate the actual command from the article name and the article name can then be easily passed to the group server functions for performing article validation. We also added the PingServer class that extends the Timer class to Ping the server every 10 seconds to check if it's still live. We settled on a 10 second time interval as we didn't want to keep it too short and have the clients constantly pinging the server to check if it's live as that could potentially overburden the server's resources. We also did not want to keep the time interval too long since clients are constantly making RMI commands and should know within 10 seconds if the server is still live and working. In the event that the server is down and the Ping() function throws an exception, the client(s) gracefully terminate. In the event that the client abruptly shuts down, we decided that the server should not remove the client information from its memory. Considering the fact that we assign random port numbers to every new client, it is not possible in our design for the server to automatically recognize clients that had abruptly shut down and attempt to join the server again. It is possible that all 5 clients crash and the server isn't able to accept more clients since we would have reached our MAX_CLIENT limit. In this case the server terminal can be terminated and should be restarted again.

Test Classes
We defined multiple JUNIT test classes that tested public methods in PubSubServer such as Join, Leave, Publish, Subscribe and Unsubscribe as well as private methods which were defined for performing article format and command line format validation. In-depth analysis of our testing has been provided in the README.pdf file.