

## CSCI 5105 Project 1 PubSub

Youfu Yan ([yan00111@umn.edu](mailto:yan00111@umn.edu)), Bin Hu ([hu000525@umn.edu](mailto:hu000525@umn.edu))

### Description:

**PubSub System by JavaRMI** This project involves implementing a simple publish-subscribe system called PubSub, using two forms of communication: UDP messages and Java RMI. The system will consist of clients and a server. The clients will communicate with the server to publish and subscribe to articles. The server will match the set of clients for a published article and then propagate the article to each client via UDP. To handle communication efficiently, Multithreading: The server can handle multiple clients and incoming articles simultaneously, the server will allow a maximum of MAXCLIENT clients at any one time, and clients can leave the server at any time. The article can be propagated as soon as it is published.

### Design and Implementation:

The implementation consists of two packages: Client and Server. In the Client package, we have the following method implemented: (1) Join: The client will call the "Join" method on the server via RPC/RMI to register with the server and provide its IP, and Port for subsequent UDP communication; (2) Leave: The client can call the "Leave" method to unregister from the server. (3) Subscribe: The client can call the "Subscribe" method to request a subscription to articles of a certain type, originator, or organization; (4) Unsubscribe: The client can call the "Unsubscribe" method to cancel a subscription; (5) Publish: The client can call the "Publish" method to send a new article to the server; (6) Ping: The client can call the "Ping" method to check if the server is up; (7) ReceiveUDP: The client will have a separate thread that blocks and listens for incoming articles from the server via UDP.

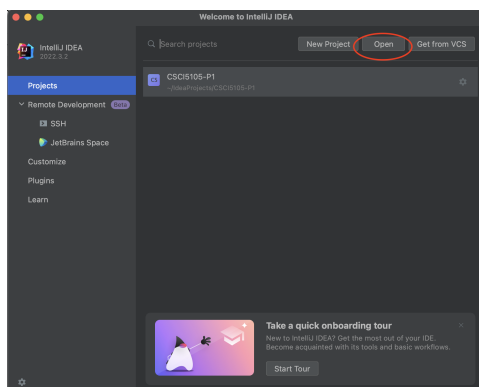
In the Server package, the following method has been implemented: (1) Join: The server will add the client to its list of registered clients and return a confirmation of successful registration; (2) Leave: The server will remove the client from its list of registered clients and return a confirmation of successful unregistration. (3) Subscribe: The server will store the client's subscription and return a confirmation of a successful subscription. (4) Unsubscribe: The server will remove the client's subscription and return a confirmation of successful unsubscription; (5) Publish: The server will receive the article and match it against the list of subscriptions to determine the set of clients to propagate the article to. It will then send the article to each matching client via UDP. (6) Ping: The server will return a confirmation of being up.

### How to Comlie, Run, and Test Code:

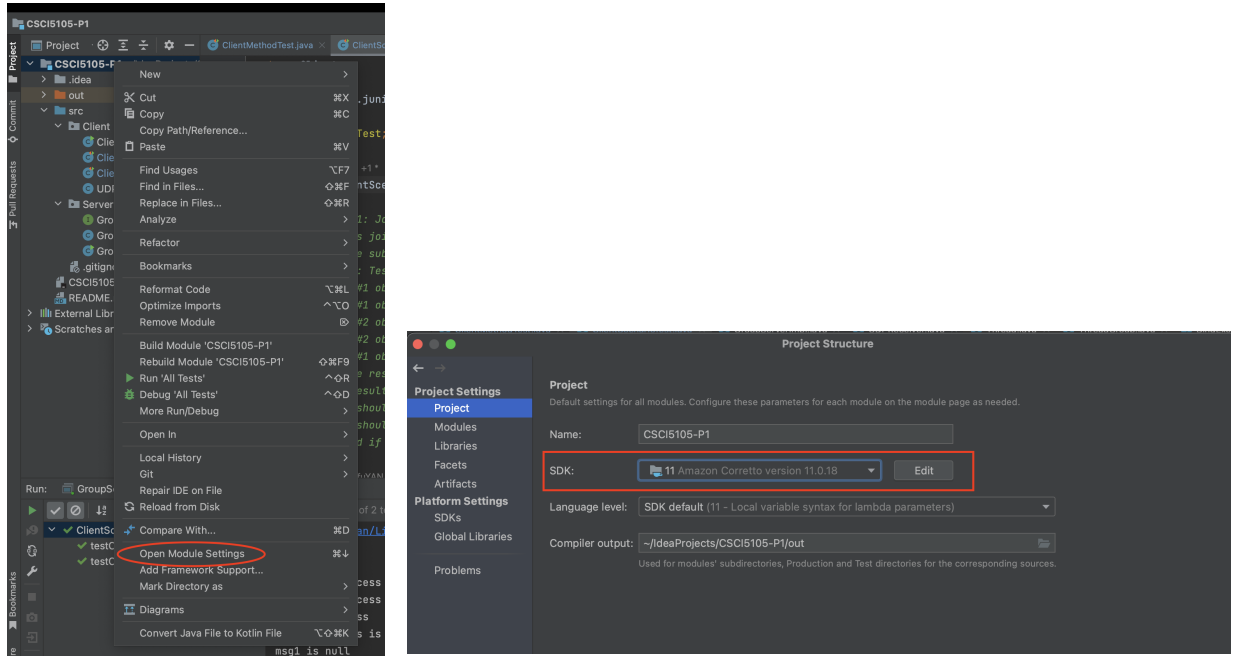
The project is built using the IntelliJ IDEAProject format, and it is recommended to run the code through IntelliJ without making any modifications. All the CSE labs have installed IntelliJ as default.

Operations to run the project in IntelliJ:

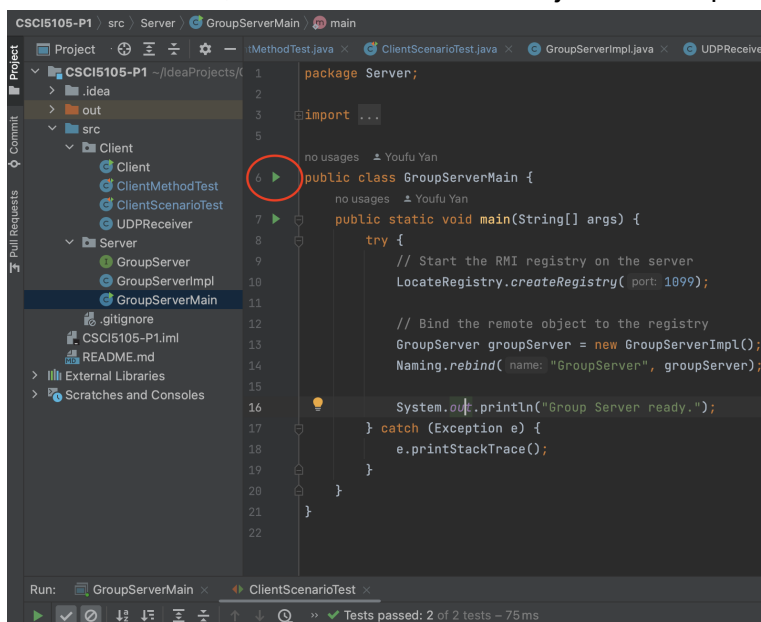
1. Click and open the IntelliJ application. On the welcome page, click `Open` and direct to the project folder, then open the project.



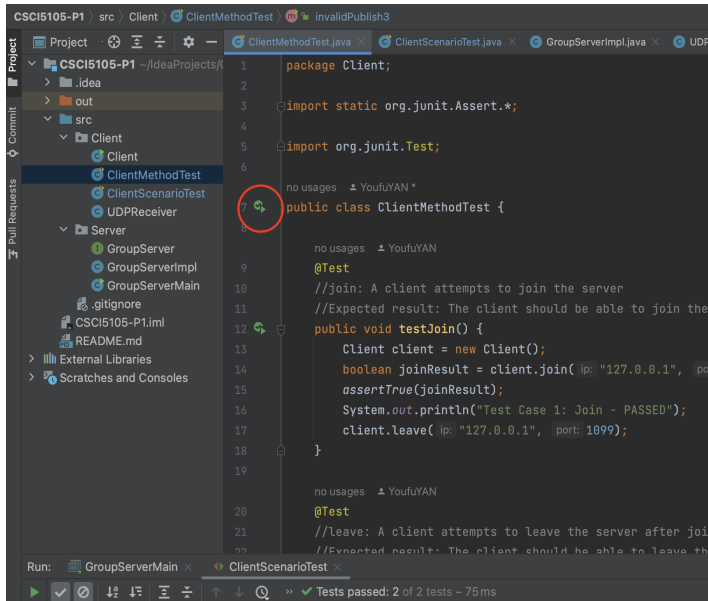
2. After opening the project in IntelliJ, it will automatically load the JUnit library. The SDK used in this project is Java 11, which is the same as the Java version installed on the CSE lab computer.
3. (Optional if the SDK is not Java 11) To change the SDK setting, right-click on the name of the Project folder on the left panel, and click 'Open Module Settings'. Then, on the left panel, select 'Project' under 'Project Setting'.



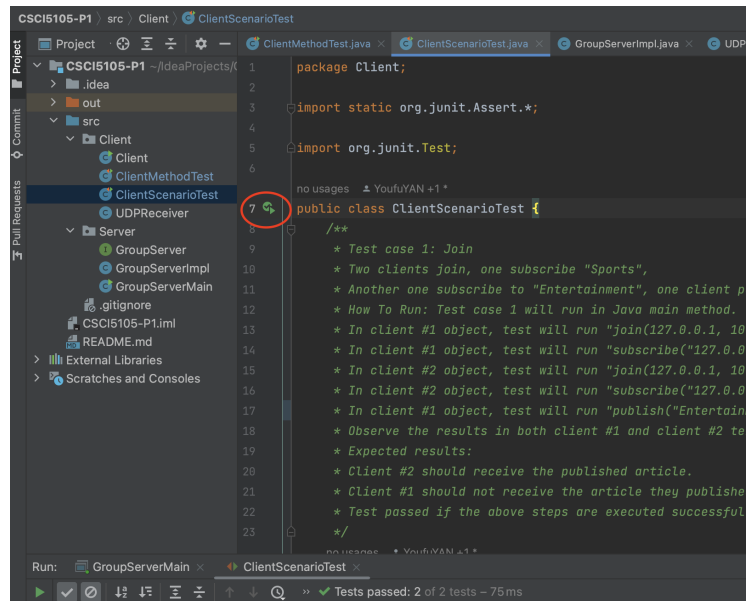
4. To run the project, the first step is to ensure that the RMI registry is up and running. Direct to the 'src/Server' folder, click the 'GroupServerMain.java' class and execute it. Once that's done, it will bind the remote object to its implementation.



5. With the GroupServer now running, you can run the various ClientTest classes to simulate different scenarios with different numbers of clients performing different operations. Direct to the `src/Client` folder, click on different test cases(**ClientMethodTest.java**, **ClientScenarioTest.java**), and execute them. The console will display all of the operations in a meaningful manner.



```
1 package Client;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class ClientMethodTest {
8
9     //no usages YoufuYAN *
10
11     //no usages YoufuYAN
12     @Test
13     //Join: A client attempts to join the server
14     //Expected result: The client should be able to join the
15     public void testJoin() {
16         Client client = new Client();
17         boolean joinResult = client.join("127.0.0.1", 1099);
18         assertTrue(joinResult);
19         System.out.println("Test Case 1: Join - PASSED");
20         client.leave("127.0.0.1", 1099);
21     }
22
23     //no usages YoufuYAN
24     @Test
25     //Leave: A client attempts to leave the server after join
26     //Expected result: The client should be able to leave the
27 }
```



```
1 package Client;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class ClientScenarioTest {
8
9     //no usages YoufuYAN +1 *
10
11     /**
12      * Test case 1: Join
13      * Two clients join, one subscribe "Sports",
14      * Another one subscribe to "Entertainment", one client p
15      * How To Run: Test case 1 will run in Java main method.
16      * In client #1 object, test will run "join(127.0.0.1, 10
17      * In client #1 object, test will run "subscribe("127.0.0
18      * In client #2 object, test will run "join(127.0.0.1, 10
19      * In client #2 object, test will run "subscribe("127.0.0
20      * In client #1 object, test will run "publish("Entertain
21      * Observe the results in both client #1 and client #2 te
22      * Expected results:
23      * Client #2 should receive the published article.
24      * Client #1 should not receive the article they publishe
25      * Test passed if the above steps are executed successful
26      */
27 }
```

## Test Design:

We are using the JUnit testing framework, and each test case will check the expected result and print out the result of the test case. The output will contain the information of each test case indicating whether it passed or failed.

In the **ClientMethodTest.java** file, we will test the joining and leaving of clients to the group server, the subscription, and unsubscription of clients to articles, the publishing of articles by clients and their delivery to the subscribers, and the ping functionality of the group server.

### Test Case Descriptions:

The list of test cases attempted with their expected results is as follows:

1. Join - Positive Test Case:  
Description: A client attempts to join the server.  
Expected Result: The client should be able to join the server and return true.  
Result: PASSED
2. Leave - Positive Test Case:  
Description: A client attempts to leave the server after joining the server.  
Expected Result: The client should be able to leave the server and return true.  
Result: PASSED
3. Subscribe - Positive Test Case:  
Description: A client attempts to subscribe to a topic.  
Expected Result: The client should be able to subscribe to a topic and return true.  
Result: PASSED
4. Unsubscribe - Positive Test Case:

Description: A client attempts to unsubscribe to a topic after subscribing to a topic.  
Expected Result: The client should be able to unsubscribe to a topic and return true.  
Result: PASSED

5. Unsubscribe All - Positive Test Case:

Description: A client attempts to unsubscribe to all topics after subscribing to all topics.  
Expected Result: The client should be able to unsubscribe to all topics and return true.  
Result: PASSED

6. Publish - Positive Test Case:

Description: A client attempts to publish a message to a topic.  
Expected Result: The client should be able to publish a message to a topic and return true.  
Result: PASSED

7. Greeting -Positive Test Case:

Description: A client attempts to get the greeting message from the server  
Expected result: The client should be able to get the greeting message from the server, and return true  
Result: PASSED

8. Ping - Positive Test Case:

Description: A client attempts to ping the server  
Expected result: The server responds and the client receives return true from the ping method  
Result: PASSED

9. invalidSubscribe - Negative Test Case:

Description: A client attempts to subscribe to a topic with invalid topic name.  
Expected Result: The client should not be able to subscribe to the topic and return false.  
Result: PASSED

10. invalidSubscribe2 - Negative Test Case:

Description: A client attempts to subscribe to a topic but with wrong order for the article.  
Expected Result: The client should not be able to subscribe to the topic and return false.  
Result: PASSED

11. invalidUnsubscribe - Negative Test Case:

Description: A client attempts to unsubscribe from a topic that it is not subscribed to  
Expected Result: The client should not be able to unsubscribe from a topic, and return false  
Result: PASSED

12. validPublish2 - Positive Test Case:

Description: A client attempts to publish to a topic without originator field  
Expected Result: The client should be able to publish to a topic without originator field, and return true  
Result: PASSED

13. invalidPublish2 - Negative Test Case:

Description: A client attempts to publish to a topic that does not exist  
Expected Result: The client should not be able to publish to a topic, and return false  
Result: PASSED

14. invalidPublish3 - Negative Test Case:

Description: A client attempts to publish with wrong order of fields for the article

Expected Result: The client should not be able to publish to a topic, and return false

Result: PASSED

15. invalidPublish3 - Negative Test Case:

Description: A client attempts to publish without the content field

Expected Result: The client should not be able to publish to a topic, and return false

Result: PASSED

16. publishandSend - Positive Test Case:

Description: A client publish a article and server send it to the subscriber right after it is published.

Expected Result: The client who subscribe this article should receive it.

Result: PASSED

17. subscribeandSend - Positive Test Case:

Description: A client subscribe a article type and then server send a article meeting the requirements.

Expected Result: The client receive the related articles.

Result: PASSED

18. unsubscribeandnotSend - NegativeTest Case:

Description: A client subscribe a article type and then server send a article meeting the requirements. The client unsubscribe the article type and then the client do not receive such kinds of article.

Expected Result: Expected result: The client do not receive the related articles after unsubscribing.

Result: PASSED

In **ClientScenarioTest.java**, we are testing two different scenarios. The testing approach includes creating two client objects and performing a series of actions on them to test different functionalities.

Testing Case Description:

1. Two clients subscribe to different articles

Description: This test case checks whether a client can join a group and subscribe to a particular category. It also tests whether the clients can publish an article in the correct category and whether other clients can receive it or not.

Expected results are as follows:

Client #2 should receive the published article.

Client #1 should not receive the article they published.

Result: PASSED

2. Two clients subscribed to the same article, and one was left between two publishing actions.

Description: This test case checks whether a client can leave a group and whether they can still receive published articles or not.

Expected results are as follows:

Client #2 should receive the first published article.

Client #1 should not receive the second published article.

Result: PASSED

3. One client subscribe to the multiple topics

Description: one client joins, subscribes "Sports" article, then subscribes "Sports and UMN" article, observing the number of UDP that client receives.

Expected results are as follows:

The client should receive the UDP only 1 time.

Result: PASSED

4. client get articles in accord with his subscribe

Description: Client1 join, client2 join, client1 subscribe "Politics", then client2 publish "Politics;;UMN;UMN wins", Observe the number of UDP of client #1 and the received msg. Client1 unsubscribe "Politics", then client1 subscribe "Politics;;UMN;", Observe the number of UDP of client #1 and the received msg. Client2 publish "Politics;;;UMN wins wins 2", Observe the number of UDP of client #1 and the received msg.

Expected results are as follows:

Client #1 first receive msg "Politics;;UMN;UMN wins", and UDP count of Client #1 is 1.

Then after second subscribe, Client #1 first receive msg "Politics;;UMN;UMN wins", and UDP count of Client #1 is 2. Then after second publish, Client #1 first does not receive msg ""Politics;;;UMN wins wins 2"", and UDP count of Client #1 keeps 2.

Result: PASSED

The list of test cases includes both positive and negative test cases to ensure that the code meets the required specifications and that it handles exceptions as expected. The positive test cases aim to test the code's functionality with valid inputs, while the negative test cases aim to test the code's robustness with invalid inputs. All the positive test cases have been successfully PASSED. The code can handle invalid input and exceptions correctly.

**Source code, makefiles and/or a script to start the system, and executables**

All Java codes are all tar into the submitted file, including this documentation file.