Readme

Individual contributions:

1. Jashwin Acharya (X500: achar061):
    a. Implemented initial versions of the Subscribe, Publish and Unsubscribe functions
    b. Implemented article validation functions for Publish, Subscribe and Unsubscribe commands and also wrote multiple test cases to test those validation methods
    c. Implemented the complete PubSub Client command line interface along with the corresponding command line format validation functions and complete multi-thread implementations for making RMI calls and receiving published articles via UDP
    d. Implemented functionality for pinging the server every 10 seconds from the client.
2. Al Yaqdhan Al Maawali (X500: almaa009):
    a. Implemented threading and command line code base
    b. Implemented the ping function
    c. Created some J-unit tests to test the validity of the clients input in the command line.
3. William Stahl (X500: stahl186)
    a. Implemented data structures and logic for publishing to clients according to their subscriptions.
    b. Implemented initial versions of Join, Leave.
    c. Wrote tests for public methods of PubSubServer class, which includes ClientTestThread.

Running RMI server and clients on the same computer:

The following steps have to executed in the order specified below for successfully running the Client(s) and server:

1. Navigate to the "src" folder in the attached zip file
2. Open 3 separate terminals in the "src" folder
3. In one terminal, type "rmiregistry 1099 &" and hit "ENTER" on the keyboard. Keep this terminal running while manually running the PubSub server and clients
4. In the second terminal, do the following steps:
    a. Type "javac PubSubServer.java" and hit ENTER
    b. Type "java PubSubServer" and hit ENTER. Upon pressing ENTER, the following message will be displayed in the terminal: "Publish-Subscribe server is ready as: PubSubServer"
5. In the third terminal, do the following steps:
    a. Type "javac PubSubClient.java" and hit ENTER

b. Type “java PubSubClient localhost” and hit ENTER. Upon pressing ENTER, a welcome message will be shown which highlight all possible options for making RMI calls to the server

Multiclient support: If you want to test **multiple clients**, then you can open multiple more terminals in the “src” folder and follow **step 5 above** to have multiple clients communicate with the server. The maximum clients our application can support is 5, but new clients can join the server once an existing client leaves.

Running RMI Server and Client(s) on different Computers:

Our implementation allows the server and clients to be on different computers and still communicate with each other. To test this feature, please perform the following steps (assuming you have a 2 computer setup):

1. Navigate to the “src” folder on both computers
2. Open a terminal window in the “src” folder on both computers
3. Type “rmiregistry 1099 &” in both terminals and hit “ENTER” on the keyboard. Keep these terminals running while manually running the PubSub server and clients
4. Open one more terminal in the 1st computer and perform the following steps:
    a. Type “javac PubSubServer.java” and hit ENTER
    b. Type “java PubSubServer” and hit ENTER. Upon pressing ENTER, the following message will be displayed in the terminal: “Publish-Subscribe server is ready as: PubSubServer”
5. Open one more terminal in the 2nd computer and perform the following steps:
    a. Type “javac PubSubClient.java” and hit ENTER
    b. Type “java PubSubClient localhost” and hit ENTER. Upon pressing ENTER, a welcome message will be shown which highlight all possible options for making RMI calls to the server

Publish-Subscribe Client Interface Usage:

When the client file is run, you will see the following “Welcome” screen:

```
Welcome to the PubSub Client!
Before entering your request, please see the following rules for the 6 operation
s that can be performed (none of these are case sensitive):

1. Enter "Join" to join the group server.
2. Enter "Leave" to leave the group server.
3. Enter "Publish" to send a new article.
4. Enter "Subscribe" to request a subscription to the group server.
5. Enter "Unsubscribe" to request an unsubscribe to the group server.
6: Enter "Display" to display published articles.

[CLIENT]: Enter command:
```

Note: All the commands listed below are not case sensitive i.e., “join” and “Join” have the same utility, “Publish:” and “publish:” have the same utility and so on.

Join Command:

Type “join” in the client terminal(s) to join the server:

```
[CLIENT]: Enter command:
join
```

Once the client has successfully joined, the server terminal will show a message such as below:

```
(base) jashwinacharya@Jashwins-MacBook-Pro src % java PubSubServer

Publish-Subscribe Server is ready as: PubSubServer
[SERVER]: Client pinged server. Server is online.
[SERVER]: Added new client with IP: 127.0.0.1, Port: 49370
```

Note: We have added functionality for the Client to Ping the server every 10 seconds to check if the server is still live. For that reason, you will regularly see the message “Client pinged server. Server is online.” in the server terminal.

Leave Command:

Type “leave” in the client terminal(s) to leave the server:

```
[CLIENT]: Enter command:
leave
```

Once the client has successfully left the server, the server terminal will show a message such as below:

```
[SERVER]: Removed client at address 127.0.0.1 and Port 19628.
```

Subscribe Command:

Type “subscribe: <article>” in the client terminal(s) to subscribe to a particular article:

```
[CLIENT]: Enter command:
subscribe: Sports;;;
```

The server responds with the following message:

```
[SERVER]: Client at Port Number 56097 has subscribed to Article: "Sports;;;".
```

Note: The format of the “subscribe” command matters. “subscribe: Sports;;;” is a valid command, but “subscribe Sports;;;” is **not** a valid command since the colon is missing. Also, the article being subscribed to cannot be more than 60 characters long. If the article is more than 60 characters long (120 bytes), then the client needs to type a new subscribe command with a shorter Article.

Publish Command:

Type “publish: <article>” in the client terminal(s) to publish an article to a group of clients:

```
[CLIENT]: Enter command:
publish: Sports;;;contents
```

The server responds with the following message:

```
[SERVER]: Sent article Sports;;;contents to client with IP Address: 127.0.0.1 and Port Number: 56097
```

Note: When the “publish” command is entered, the client terminal receives a response shown below:

```
[CLIENT]: Enter command:
[CLIENT]: Response from server: Sports;;;contents
```

You can enter another command instantly once the response from the server is received. For example:

```
[CLIENT]: Enter command:
[CLIENT]: Response from server: Sports;;;contents
leave
```

Once you press “Enter”, the command will be executed and you can continue entering commands as below:

```
[CLIENT]: Enter command:
[CLIENT]: Response from server: Sports;;;contents
leave

[CLIENT]: Enter command:
```

Similar to the “subscribe” command, the format of the “publish” command matters. For example: “publish: Sports;;;contents” is valid but “publish Sports;;;contents” is invalid since the colon is missing.

Unsubscribe Command:

Type “unsubscribe: Sports;;;” in the client terminal(s) to unsubscribe from an article:

```
[CLIENT]: Enter command:
unsubscribe: Sports;;;
```

The server responds with the following message:

```
[SERVER]: Client with IP Address 127.0.0.1 and Port Number 42372 has unsubscribed from Article "Sports;;;".
```

Similar to the “subscribe” command, the format of the “unsubscribe” command matters. For example: “unsubscribe: Sports;;;” is valid but “unsubscribe Sports;;;” is invalid since the colon is missing.

Display Command:

The “Display” command displays all the published articles a client has been and is currently subscribed to. Type “display” in the client terminal(s) to display all the articles as shown below:

```
[CLIENT]: The following articles have been published to this Client:

[CLIENT]: Article #1: Entertainment;UMN;;contents
[CLIENT]: Article #2: Sports;;;contents
```

Unit tests (Execution instructions in README.md):

Testing 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. We designed 27 test cases (14 positive and 13 negative) that were designed for catching both valid and invalid article formats for the Publish, Subscribe and Unsubscribe functions. Invalid article formats for the Publish function included articles where the first three fields are all empty or only the contents field is missing or all 4 fields are empty. Invalid article formats for the Subscribe functions included articles where either an empty string was passed as the

Article name or the contents field was populated or when none of the fields are populated. We also included 9 test cases for checking the command format of the input received from the PubSubClient User interface which included 3 positive test cases and 6 negative test cases that tested incorrect formats such as “publish Sports;;;contents”.

The public facing method tests are in another class, and we have included 9 tests:

1. CheckJoin: 5 negative (invalid Port/IP, already joined, etc.) cases and 1 positive case
2. CheckServerCapacity: Negative case where client is rejected from full server
3. CheckLeave: Negative cases before and after client has left (which is a positive case)
4. CheckInvalidSubscribe: Negative cases where the topic is invalid, contents are present, empty fields, using IP/Port info that hasn’t joined
5. CheckValidSubscription: Positive cases of all 6 combinations ranging from all three fields to just one of the three
6. CheckInvalidUnsubscirbe: Negative cases of unsubscribing from articles the client wasn’t subscribed to EXACTLY, invalid topics, strings, non-joined IP/Port, not confusing with other client’s subscriptions
7. CheckValidUnsubscribe: Positive cases where clients unsubscribing do not clobber each other’s subscriptions
8. CheckPing: Positive case where server is successfully pinged
9. CheckReception: Positive case where client receives content it is subscribed to and not other content.
    a. NOTE: This test uses UDP, so UDP failures may cause this test to fail.