**A black and white logo

Description automatically generated**

**Distributed Systems**

**Assignment 1**

|  |
| --- |
| Name: Mostapha Abdulaziz Abdullah. |
| Specialization: AI A3. |
| ID: 227824 |

**Peer to peer Chat Ring topology**

* **Diagram:**

**A diagram of a communication system

Description automatically generated**

* **Sending Private Messages:**
* Our ring chat system is mainly depending in a Coordinator and other four nodes.
* All nodes can send private messages to each other directly without involving the coordinator (By initializing a private port for each node ‘priv\_msg\_port’ and establishing a private socket ‘privateManagesocket’.
* Then there’s an identifier prefix that fixies the message if it starts with ‘Private.’ It takes this message and send it as a parameter in the method ‘encryptMessage’.
* This method encrypts the message by applying simple encryption (shifting all characters by 1).
* Then the sender node sends the message by ‘sendprivateMessage’ method.
* Then the recipient node listings for incoming Private messages on its private socket(privateMessageSocket).
* Then the recipient node decrypts the received message by the ‘decryptMessage’ method and then prints it in its console).
* **Sending Public Messages:**
* Once the program is running and the coordinator starts to listen for any incoming connections on its port (2002).
* Nodes Establishes Connections with the Coordinator using sockets.
* Each node sends its name when connecting to the coordinator.
* Coordinator maintains the connected nodes in a node connections List <nodeConnectios> of type array list.
* Any node wants to send a public message it sends it by the output stream of the coordinator and the coordinator then receives the message and displays it to all the connected nodes in the connections list and send to them the message on their output stream.
* **Main methods:**

- node.run():

It represents the main method of the Node class. It establishes the connection with the coordinator, sends the node's name, and starts a separate thread to listen for incoming messages from the coordinator.

-Node.sendMessage():

It sends a message to the coordinator by writing it to the output stream of the connection. The message will be displayed to all connected nodes.

-Node.receiveMessage():

It listens for incoming messages from the coordinator by reading from the input stream. Then it runs in a separate thread and continuously listens for messages until the Tconnection is closed.

- Node.receiveMessage():

It listens for incoming messages from the coordinator by reading from the input stream.And it runs in a separate thread and continuously listens for messages until theconnection is closed.

-Node.sendPrivateMessage():

It sends a private message to a specific node. It establishes a socket for communication and encrypts the message using a simple encryption algorithm before sending it to the recipient node.

-Node.receivePrivateMessage():

It listens for incoming private messages on the socket. It decrypts the received message using the reverse operation of the encryption algorithm and prints the decrypted message.

-Coordinator.run():

It represents the main method of the Coordinator class. It listens for incoming connections from nodes, maintains a list of connected nodes, and displays the received messages to all connected nodes.

-Coordinator.listenForConnections():

It listens for incoming connections from nodes. When a new node connects, it adds the node's name and output stream to the list of connected nodes.

-Coordinator.receiveMessages():

It listens for incoming messages from nodes. It reads the messages from the input streams of connected nodes and displays them to all other nodes.

* **GUI:**

1. **Nodes GUI**

**A screenshot of a computer

Description automatically generated**

1. **Coordinator’s GUI**

A screenshot of a computer

Description automatically generated