Name: - OM MITTAL

Introduction

This document serves to explain the responsibilities and interactions between the classes in a simple chat application. The application is designed to simulate a conversation between two players, each running on a separate thread, taking turns to send and receive messages. The document will outline the purpose of each class and its role within the application, providing insight into how the different components work together.

Class Documentation

➢ MULTI – PROCESS CLASSES

Class: Player

- **Responsibilities:**
    - As the Player class, I act as the communication handler for both Player1 and Player2. My main task is to manage the socket connection and facilitate the sending and receiving of messages.
    - I maintain a count of how many messages have been exchanged. For each message I receive, I append the message count before passing it back.
    - I also handle the clean-up process by closing the socket connection when the chat ends, ensuring no further communication occurs.

Class: Player1

- **Responsibilities:**
    - As Player1, I am responsible for initiating the chat by setting up a server socket and waiting for Player2 to connect to me.
    - I handle the process of sending and receiving messages between Player2 and me. I ensure that I send a message first, then wait for a response, repeating this process until I've sent and received 10 messages.
    - Once the conversation is complete, I close the connection to signal that our chat has ended

Class: Player2

- **Responsibilities:**
    - As Player2, I connect to Player1's server to participate in the chat. My job is to respond to each message that Player1 sends to me.
    - I receive messages from Player1, display them, and then send my own response. This back-and-forth continues until we've exchanged 10 messages in total.
    - Like Player1, I am responsible for closing the connection when our chat is complete, signalling the end of our conversation.

➢ SINGLE-PROCESS CLASSES

Class: Main

- **Responsibilities:**
  - As the Main class, I am responsible for orchestrating the start of the chat interaction between the two players. I create two Player instances, player1 and player2, and set them as opponents of each other.
  - I ensure that the chat starts by invoking the start () method on both players, kicking off their conversation in separate threads. This sets the stage for the message exchange process to begin.

Class: Player

- **Responsibilities:**
  - As the Player class, my primary responsibility is to manage the sending and receiving of messages between myself and my opponent. I handle this through a threaded execution, ensuring smooth communication.
  - I am in charge of determining whether it's my turn to send a message or wait for my opponent's message. If it's my turn, I prompt the user to enter a message and then send it; if not, I wait until it's my turn.
  - I keep track of the total number of messages exchanged using a shared messageCounter variable to ensure that we stop after 10 messages each.
  - I store the last message received or sent in the temp variable to display it during communication.
  - When it's my turn to send a message, I call allowSending () on my opponent, notifying them that they can now send their message. This synchronization helps in maintaining the order and flow of communication between both players.

CONTACT ME: -

- **E-Mail: -Ommittal.careers@gmail.com**
- **Phone no: - +91 9548615488**
- **Linkedin:- Linkedin**
- **Github:- GitHub**
- **Porfolio:-Portfolio**