Documentation for ChatApplication\_UsingSocket program

**Introduction:**

* The program is developed in Java language using Jdk 11.0.7.
* The program allows two users to connect and chat with each other.
* Users can send n number of messages to each other.
* The program can be run as Server-Client on two different machines but on same port.
* The user which will behave as Client should know the IP and Port of Server on which it is running.
* To exit from the program, any user can type “Exit” as message and send to the other user. The program will then stop executing at both user’s side.

**Source Code Files:**

The Program has two source code files and two help files.

The source code files are “Server.java” and “Client.java”. The other two files are “Readme.txt” and “Code Documentation.docx”.

1. Server.java

This file is to start the Server. The execution of this file will act as Server from Server-Client Application which accepts connection from Client.

This file has only one class “Server” which has three methods named as below:

* read\_message : This method takes input as DataInputStream object to read the message received. It returns the Thread object which is created to continuously read messages. It overrides run() method of interface Runnable to read and print the message received from Client using input object. If message received is “Exit”, then it will stop executing the thread.
* send-message : This method takes two input as DataOutputStream object and Scanner object to send and print the message to the connected Client. It returns the Thread object which is created to continuously send messages. It also overrides run() method of Runnable interface which has the code to take input from the user and send it to client. It also checks the message length and gives message to user to type something if the user tries to send an empty message. If the user types “Exit” as message, it will stop the thread execution.
* main : Jdk will execute this method first before any other method. This method takes Port number as input from user on which server will be running. It starts the Server using The object of ServerSocket class. It also accepts the connection from Client using accept method of ServerSocket. After the connection has been established, the objects of two threads are created to read and send messages. When the start() method of Thread is executed, the code written in read-message and send\_message is executed. At last, it checks if the threads are alive or not, and it closes the connection and exit the program if the threads are not alive.

1. Client.java

This file is to start the Client. The execution of this file will act as Client from Server-Client Application which connects to a Server using its ip and port.

This file has only one class “Client” which has three methods named as below:

* read\_message : This method takes input as DataInputStream object to read the message received. It returns the Thread object which is created to continuously read messages. It overrides run() method of interface Runnable to read and print the message received from Server using input object. If message received is “Exit”, then it will stop executing the thread.
* send-message : This method takes two input as DataOutputStream object and Scanner object to send and print the message to the connected Server. It returns the Thread object which is created to continuously send messages. It also overrides run() method of Runnable interface which has the code to take input from the user and send it to server. It also checks the message length and gives message to user to type something if the user tries to send an empty message. If the user types “Exit” as message, it will stop the thread execution.
* main : Jdk will execute this method first before any other method. This method takes IP Number and Port number as input from user on which server will be running. It starts the Client using the object of Socket class. The isConnected() method is used to check if the client has been connected to the Server or not. After the connection has been established, the objects of two threads are created to read and send messages. When the start() method of Thread is executed, the code written in read-message and send\_message is executed. At last, it checks if the threads are alive or not, and it closes the connection and exit the program if the threads are not alive.

1. Readme.txt

This file explains how to compile and run the program step-by-step. It also has details of installation of jdk.

1. Code Documentation.docx

This file itself explains the source code of the program for each files.