Answer’s

Task 03

1 Ans:

```
import java.io.*;
import java.net.*;
import java.util.*;

// Server class to handle multiple clients
class ChatServer {
    private static final int PORT = 12345;
    private static Set<PrintWriter> clientWriters = new HashSet<>();

    public static void main(String[] args) {
        System.out.println("Chat server started...");
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            while (true) {
                new ClientHandler(serverSocket.accept()).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static class ClientHandler extends Thread {
        private Socket socket;
        private PrintWriter out;

        public ClientHandler(Socket socket) {
            this.socket = socket;
        }

        public void run() {
            try (BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) {
                out = new PrintWriter(socket.getOutputStream(), true);
                synchronized (clientWriters) {
                    clientWriters.add(out);
                }

                String message;
                while ((message = in.readLine()) != null) {
                    System.out.println("Received: " + message);
                    synchronized (clientWriters) {
                        for (PrintWriter writer : clientWriters) {
                            writer.println(message);
                        }
                    }
                }
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                synchronized (clientWriters) {
                    clientWriters.remove(out);
```

```
            }
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

// Client class to connect to the server
class ChatClient {
    private static final String SERVER_ADDRESS = "localhost";
    private static final int PORT = 12345;

    public static void main(String[] args) {
        try (Socket socket = new Socket(SERVER_ADDRESS, PORT);
             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
             Scanner scanner = new Scanner(System.in)) {

            System.out.println("Connected to chat server.");

            Thread readerThread = new Thread(() -> {
                try {
                    String message;
                    while ((message = in.readLine()) != null) {
                        System.out.println(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            });
            readerThread.start();

            while (true) {
                String userInput = scanner.nextLine();
                if (userInput.equalsIgnoreCase("exit")) {
                    break;
                }
                out.println(userInput);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

I will create a new Java client-server chat application using sockets and multithreading. Let me know if you need any specific features! 🚀

Java Chat Application
Answer in chat instead

Here’s a Java client-server chat application using sockets and multithreading. The `ChatServer` class handles multiple clients, while the `ChatClient` connects to the server and sends messages. Let me know if you need enhancements! 🚀

2 Ans:

```
import java.io.*;
import java.net.*;
import java.util.*;

/**
 * Chat Server that handles multiple clients using Java Sockets and Multithreading.
 */
class ChatServer {
    private static final int PORT = 12345;
    private static Set<PrintWriter> clientWriters = new HashSet<>();

    public static void main(String[] args) {
        System.out.println("Chat server started...");
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            while (true) {
                new ClientHandler(serverSocket.accept()).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * Handles communication with a single client.
     */
    private static class ClientHandler extends Thread {
        private Socket socket;
        private PrintWriter out;

        public ClientHandler(Socket socket) {
            this.socket = socket;
        }

        public void run() {
            try (BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) {
                out = new PrintWriter(socket.getOutputStream(), true);
                synchronized (clientWriters) {
                    clientWriters.add(out);
                }

                String message;
                while ((message = in.readLine()) != null) {
                    System.out.println("Received: " + message);
                    synchronized (clientWriters) {
                        for (PrintWriter writer : clientWriters) {
                            writer.println(message);
                        }
                    }
                }
            } catch (IOException e) {
```

```
                e.printStackTrace();
            } finally {
                synchronized (clientWriters) {
                    clientWriters.remove(out);
                }
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

/**
 * Chat Client that connects to the server and enables real-time messaging.
 */
class ChatClient {
private static final String SERVER_ADDRESS = "localhost";
private static final int PORT = 12345;

public static void main(String[] args) {
    try (Socket socket = new Socket(SERVER_ADDRESS, PORT);
         BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
         PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
         Scanner scanner = new Scanner(System.in)) {

        System.out.println("Connected to chat server.");

        // Thread to listen for incoming messages from server
        Thread readerThread = new Thread(() -> {
            try {
                String message;
                while ((message = in.readLine()) != null) {
                    System.out.println(message);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
        readerThread.start();

        // Read user input and send messages to the server
        while (true) {
            String userInput = scanner.nextLine();
            if (userInput.equalsIgnoreCase("exit")) {
                break;
            }
            out.println(userInput);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
```

```
    }
}
```

I've refined the chat application, ensuring it supports multiple clients communicating in real-time.