

The screenshot shows the IntelliJ IDEA interface with the project 'Saloni' open. The 'Client.java' file is the active editor. The code implements a client socket program using BufferedReader and BufferedWriter. It includes a constructor to initialize the socket and buffered readers/writers, and a sendMessage() method to send a message to the server. The code uses try-with-resources to handle the streams.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    private Socket socket;
    private BufferedReader bufferedReader;
    private BufferedWriter bufferedWriter;
    private String username;

    public Client(Socket socket, String username) {
        try {
            this.socket = socket;
            this.bufferedWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            this.bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            this.username = username;
        } catch (IOException e) {
            closeEverything(socket, bufferedReader, bufferedWriter);
        }
    }

    public void sendMessage() {
        try {
    
```

The screenshot shows the IntelliJ IDEA interface with the project 'Saloni' open. The 'Client.java' file is the active editor. The code implements a client socket program using BufferedReader and BufferedWriter. It includes a constructor to initialize the socket and buffered readers/writers, and a sendMessage() method to send a message to the server. The code uses try-with-resources to handle the streams.

```
try {
    bufferedWriter.write(username);
    bufferedWriter.newLine();
    bufferedWriter.flush();

    Scanner scanner = new Scanner(System.in);
    while(socket.isConnected()) {
        String messageToSend = scanner.nextLine();
        bufferedWriter.write(username + ":" + messageToSend);
        bufferedWriter.newLine();
        bufferedWriter.flush();
    }
} catch (IOException e) {
    closeEverything(socket, bufferedReader, bufferedWriter);
}

public void listenForMessage() {
    new Thread(new Runnable() {
        public void run() {
            String msgFromGroupChat;
            while(socket.isConnected()) {
                try {
                    msgFromGroupChat = bufferedReader.readLine();
                    System.out.println(msgFromGroupChat);
                } catch (IOException e) {
                    closeEverything(socket, bufferedReader, bufferedWriter);
                }
            }
        }
    }).start();
}
```

The screenshot shows the IntelliJ IDEA interface with the project 'Saloni' open. The 'Client.java' file is the active editor. The code implements a client socket program using BufferedReader and BufferedWriter. It includes a constructor to initialize the socket and buffered readers/writers, and a sendMessage() method to send a message to the server. The code uses try-with-resources to handle the streams.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    private Socket socket;
    private BufferedReader bufferedReader;
    private BufferedWriter bufferedWriter;
    private String username;

    public Client(Socket socket, String username) {
        try {
            this.socket = socket;
            this.bufferedWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            this.bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            this.username = username;
        } catch (IOException e) {
            closeEverything(socket, bufferedReader, bufferedWriter);
        }
    }

    public void sendMessage() {
        try {
    
```

The image shows two side-by-side Java code editors, likely from the IntelliJ IDEA IDE, displaying the `Client.java` and `Server.java` files respectively.

Client.java (Left Editor):

```
Saloni / src / Client - sendMessage
Saloni - Client.java
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Project Client.java ClientHandler.java
Saloni C:\Users\saloni\IdeaProjects
src Client ClientHandler Main Server
Salonuml External Libraries Scratches and Consoles
Saloni - Client.java
53     } .start();
54 }
55     3 usages
56     public void closeEverything(Socket socket, BufferedReader bufferedReader, BufferedWriter bufferedWriter) {
57         try {
58             if(bufferedReader!=null) {
59                 bufferedReader.close();
60             }
61             if(bufferedWriter!=null) {
62                 bufferedWriter.close();
63             }
64             if(socket!=null) {
65                 socket.close();
66             }
67         }catch(IOException e) {
68             e.printStackTrace();
69         }
70     } no usages
71     public static void main(String[] args) throws IOException {
72         Scanner scanner = new Scanner(System.in);
73         System.out.println("Enter your username for the group chat: ");
74         String username=scanner.nextLine();
75         Socket socket = new Socket( host "localhost", port: 1234 );
76         Client client = new Client(socket,username);
77         client.listenForMessage();
78         client.sendMessage();
79     }
80 }
81 
```

Server.java (Right Editor):

```
Saloni - Server.java
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Project Client.java ClientHandler.java
Saloni C:\Users\saloni\IdeaProjects
src Client ClientHandler Main Server
Salonuml External Libraries Scratches and Consoles
Saloni - Server.java
1 import java.io.IOException;
2 import java.net.ServerSocket;
3 import java.net.Socket;
4
5     2 usages
6     public class Server {
7         private ServerSocket serverSocket;
8
9         1 usage
10        public Server(ServerSocket serverSocket) {
11            this.serverSocket=serverSocket;
12        }
13
14        1 usage
15        public void startServer() {
16            try {
17                while(!serverSocket.isClosed()) {
18                    Socket socket = serverSocket.accept();
19                    System.out.println("A new client has connected!");
20
21                    ClientHandler clientHandler = new ClientHandler(socket);
22                    Thread thread = new Thread(clientHandler);
23                    thread.start();
24                }
25            }catch(IOException e) {
26
27            }
28        }
29    } no usages
30    public void closeServerSocket() {
31
32    }
33 
```

The code in `Client.java` handles sending messages and closing connections. It uses `BufferedReader` and `BufferedWriter` to interact with a `Socket`. The `main` method prompts the user for a username and connects to the server at port 1234.

The code in `Server.java` sets up a `ServerSocket` to listen for incoming connections. It creates a `ClientHandler` for each client and starts a new thread for each connection. The `startServer` method runs in a loop until the server socket is closed.

The screenshot shows the IntelliJ IDEA interface with the project 'Saloni' open. The 'Server.java' file is the active editor. The code implements a Java server using sockets. It includes methods for handling client connections and catching I/O exceptions. The code is annotated with usage counts for variables like 'serverSocket' and 'clientHandlers'. The bottom status bar shows system information such as battery level (22%), temperature (22°C), and date/time (08-01-2023).

```
21 } catch(IOException e) {  
22 }  
23 }  
24 }  
25 }  
26 } no usages  
27 public void closeServerSocket() {  
28     try {  
29         if(serverSocket!=null) {  
30             serverSocket.close();  
31         }  
32     } catch(IOException e) {  
33         e.printStackTrace();  
34     }  
35 }  
36 } no usages  
37 public static void main(String[] args) throws IOException{  
38     ServerSocket serverSocket = new ServerSocket( port: 1234);  
39     Server server = new Server(serverSocket);  
40     server.startServer();  
41 }  
42 }  
43 }  
44 }  
45 }
```

The screenshot shows the IntelliJ IDEA interface with the project 'Saloni' open. The 'ClientHandler.java' file is the active editor. This class implements the Runnable interface and handles client socket connections. It uses buffered readers and writers to exchange messages. The code is annotated with usage counts for various fields. The bottom status bar shows system information such as battery level (22%), temperature (22°C), and date/time (08-01-2023).

```
1 import java.io.*;  
2 import java.util.ArrayList;  
3 import java.io.BufferedReader;  
4 import java.io.BufferedWriter;  
5 import java.net.Socket;  
6 public class ClientHandler implements Runnable {  
7     3 usages  
8     public static ArrayList<ClientHandler> clientHandlers = new ArrayList<>();  
9     4 usages  
10    private Socket socket;  
11    6 usages  
12    private BufferedReader bufferedReader;  
13    7 usages  
14    private BufferedWriter bufferedWriter;  
15    5 usages  
16    private String clientUsername;  
17    1 usage  
18    public ClientHandler(Socket socket) {  
19        try {  
20            this.socket = socket;  
21            this.bufferedWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));  
22            this.bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
23            this.clientUsername = bufferedReader.readLine();  
24            clientHandlers.add(this);  
25            broadcastMessage( messageToSend: "SERVER: " + clientUsername + " has entered the chat!");  
26        } catch (IOException e) {  
27            closeEverything(socket, bufferedReader, bufferedWriter);  
28        }  
29    }  
30 }
```

The screenshot shows two instances of the IntelliJ IDEA IDE running side-by-side on a Windows desktop. Both instances have the same project structure and code editor open, displaying the `ClientHandler.java` file.

Project Structure:

- Saloni (C:\Users\saloni\IdeaProjects)
- src
- Client
- ClientHandler
- Main
- Server
- Salonuml
- External Libraries
- Scratches and Caches

Code Editor (ClientHandler.java):

```
24 }  
25 }  
26     public void run() {  
27         String messageFromClient;  
28         while (socket.isConnected()) {  
29             try {  
30                 messageFromClient = bufferedReader.readLine();  
31                 broadcastMessage(messageFromClient);  
32             } catch (IOException e) {  
33                 closeEverything(socket, bufferedReader, bufferedWriter);  
34                 break;  
35             }  
36         }  
37     }  
38  
39     public void broadcastMessage(String messageToSend) {  
40         for (ClientHandler clientHandler : clientHandlers) {  
41             try {  
42                 if (!clientHandler.clientUsername.equals(clientUsername)) {  
43                     clientHandler.bufferedWriter.write(messageToSend);  
44                     clientHandler.bufferedWriter.newLine();  
45                     clientHandler.bufferedWriter.flush();  
46                 }  
47             } catch (IOException e) {  
48                 closeEverything(socket, bufferedReader, bufferedWriter);  
49             }  
50         }  
51     }  
52 }
```

Terminal Window:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=49901:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin" -Dfile.encoding=UTF-8 Client  
A new client has connected!  
A new client has connected!  
A new client has connected!
```

The system tray at the bottom of the screen shows the date and time as 06-01-2023, and the weather as 67°F Haze.

The image shows two side-by-side screenshots of the IntelliJ IDEA IDE interface. Both screenshots display a terminal window within the IDE's frame, showing a group chat session between three users: Bhel, Softy, and Maggi.

Terminal Content (Top Screenshot):

```
Saloni - Client.java
Saloni / src > Client
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Run: Server < Client < Client < Client < Client <
C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=49905:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin" -Dfile.encoding=UTF-8
Enter your username for the group chat:
server1
SERVER: Bhel has entered the chat!
SERVER: Softy has entered the chat!
SERVER: Maggi has entered the chat!
What's up Bhel, Softy and Maggi?
Bhel: I am good dude
Softy: I am busy
Maggi: I am always free
|
```

Terminal Content (Bottom Screenshot):

```
Saloni - Client.java
Saloni / src > Client
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Run: Server < Client < Client < Client < Client <
C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=49912:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin" -Dfile.encoding=UTF-8
Enter your username for the group chat:
server1
SERVER: Softy has entered the chat!
SERVER: Maggi has entered the chat!
Sandwich: What's up Bhel, Softy and Maggi?
i am good dude
Softy: I am busy
Maggi: I am always free
|
```

The terminals show identical initial messages from the server and user input. In the top screenshot, the user types "server1". In the bottom screenshot, the user types "server1" again. Subsequent messages from the users Bhel, Softy, and Maggi are identical in both cases.

The screenshot shows two instances of the IntelliJ IDEA IDE running side-by-side on a Windows desktop. Both instances are displaying a Java-based chat application. The top window has the title 'Saloni - Client' and the bottom window has the title 'Saloni - Client'. Both windows show the same terminal output:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=49922:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin" -Dfile.encoding=UTF-8 Client
Server
Client
Client
Client
Client
Client
Run: Server < Client < Client < Client < Client < Client <
Project < Bookmarks < Structure <
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Saloni - Client.java
Saloni > src > Client
Run: "C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=49922:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin" -Dfile.encoding=UTF-8 Client
Enter your username for the group chat:
SERVER: Maggi has entered the chat!
Sandwich: What's up Bhel, Softy and Maggi?
Bhel: I am good dude
i am busy
Maggi: I am always free
```

The terminal output shows a simple chat application where users enter their names and messages are displayed. The application starts by prompting for a username, then displays messages from other users ('SERVER: Maggi has entered the chat!', 'Sandwich: What's up Bhel, Softy and Maggi?', 'Bhel: I am good dude', 'i am busy', 'Maggi: I am always free'). The interface includes standard IntelliJ features like file navigation, code refactoring, and version control.

The screenshot shows two instances of the IntelliJ IDEA IDE running side-by-side. Both instances have the same project structure and code editor open, displaying the `Server.java` file.

Project Structure:

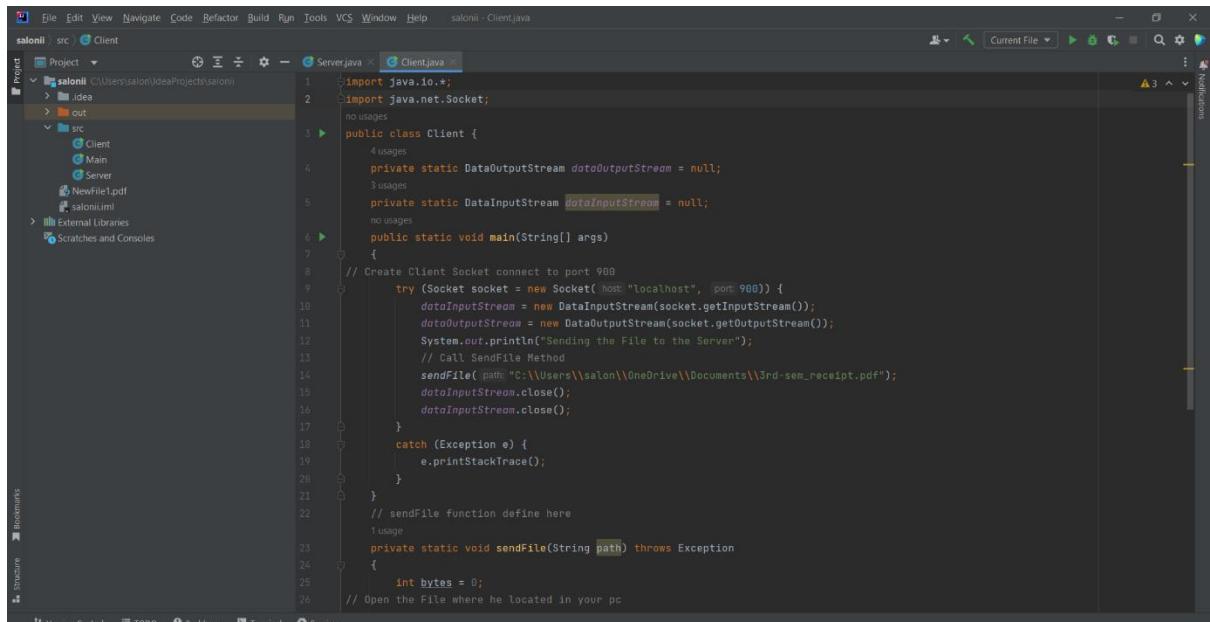
- Project: `saloni`
- src:
 - Client
 - Main
 - Server
 - Newfile.pdf
 - saloni.html
- External Libraries
- Scratches and Consoles

Code Editor (Server.java):

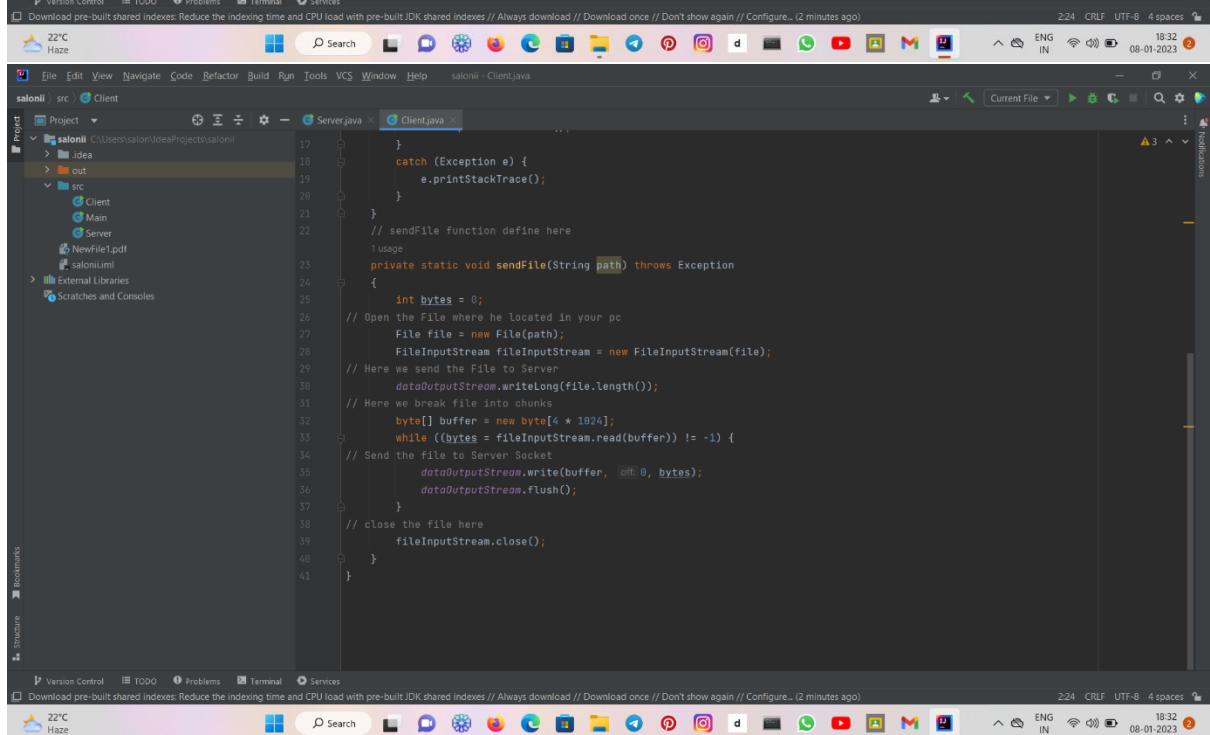
```
1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.io.FileOutputStream;
4 import java.net.ServerSocket;
5 import java.net.Socket;
6
7 public class Server {
8
9     private static DataOutputStream dataOutputStream = null;
10    private static DataInputStream dataInputStream = null;
11
12    public static void main(String[] args) {
13        try {
14            ServerSocket serverSocket = new ServerSocket( port: 900 );
15            System.out.println("Server is Starting in Port 900");
16            Socket clientSocket = serverSocket.accept();
17            System.out.println("Connected");
18            dataInputStream = new DataInputStream(clientSocket.getInputStream());
19            dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());
20
21            // Here we call receiveFile define now for that
22            receiveFile(fileName: "NewFile1.pdf");
23            dataInputStream.close();
24            dataOutputStream.close();
25            clientSocket.close();
26        } catch (Exception e) {
27
28        }
29    }
30
31    // receive file function is start here
32    private static void receiveFile(String fileName) throws Exception {
33
34        int bytes = 0;
35        FileOutputStream fileOutputStream = new FileOutputStream(fileName);
36        long size = dataInputStream.readLong(); // read file size
37        byte[] buffer = new byte[4 * 1024];
38
39        while (size > 0 && (bytes = dataInputStream.read(buffer, off: 0, (int) Math.min(buffer.length, size))) != -1) {
40            // Here we write the file using write method
41            fileOutputStream.write(buffer, off: 0, bytes);
42            size -= bytes; // read upto file size
43
44        }
45        // Here we received file
46        System.out.println("File is Received");
47        fileOutputStream.close();
48    }
49}
```

System Tray and Status Bar:

- Top bar: File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, saloni - Server.java
- Bottom bar: Version Control, TODO, Problems, Terminal, Services
- Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure... (2 minutes ago)
- System tray: 22°C, Haze, Search, Taskbar icons (File Explorer, Edge, Firefox, Task View, Taskbar icons), ENG IN, 18:32, 08-01-2023



```
saloni | File Edit View Navigate Code Refactor Build Run Tools VCS Window Help saloni - Client.java
Project | File | Edit | View | Navigate | Code | Refactor | Build | Run | Tools | VCS | Window | Help | saloni - Client.java
saloni | src | Client
Project | File | Edit | View | Navigate | Code | Refactor | Build | Run | Tools | VCS | Window | Help | saloni - Client.java
saloni | Client.java
1 import java.io.*;
2 import java.net.Socket;
3 public class Client {
4     private static DataOutputStream dataOutputStream = null;
5     private static DataInputStream dataInputStream = null;
6     public static void main(String[] args) {
7         try {
8             // Create Client Socket connect to port 900
9             try (Socket socket = new Socket("localhost", 900)) {
10                 dataInputStream = new DataInputStream(socket.getInputStream());
11                 dataOutputStream = new DataOutputStream(socket.getOutputStream());
12                 System.out.println("Sending the file to the Server");
13                 // Call Sendfile Method
14                .sendFile("C:\\Users\\saloni\\OneDrive\\Documents\\3rd-sem_receipt.pdf");
15                 dataInputStream.close();
16                 dataInputStream.close();
17             }
18         } catch (Exception e) {
19             e.printStackTrace();
20         }
21     }
22     // sendFile function define here
23     // usage
24     private static void sendFile(String path) throws Exception {
25         int bytes = 0;
26         // Open the File where he located in your pc
```



```
17     }
18     catch (Exception e) {
19         e.printStackTrace();
20     }
21 }
22 // sendFile function define here
23 // usage
24 private static void sendFile(String path) throws Exception {
25     int bytes = 0;
26     // Open the File where he located in your pc
27     File file = new File(path);
28     FileInputStream fileInputStream = new FileInputStream(file);
29     // Here we send the File to Server
30     dataOutputStream.writeLong(file.length());
31     // Here we break file into chunks
32     byte[] buffer = new byte[4 * 1024];
33     while ((bytes = fileInputStream.read(buffer)) != -1) {
34         // Send the file to Server Socket
35         dataOutputStream.write(buffer, 0, bytes);
36         dataOutputStream.flush();
37     }
38     // close the file here
39     fileInputStream.close();
40 }
41 }
```

The image displays two side-by-side screenshots of the IntelliJ IDEA IDE interface, showing the development of a Java application for file transfer.

Top Window (Server Side):

- Project:** The project structure shows files `Server.java` and `Client.java`.
- Run Tab:** The run configuration is set to "Server". The output window shows:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=50658:C:\Program Files\JetBrains\Intel
```

Server is Starting in Port 900
Connected
File is Received

Process finished with exit code 0
- Bottom Bar:** Shows system status including temperature (64°F), battery level (23%), and date/time (06-01-2023).

Bottom Window (Client Side):

- Project:** The project structure shows files `Client.java` and `main`.
- Run Tab:** The run configuration is set to "Client". The output window shows:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=50669:C:\Program Files\JetBrains\Intel
```

Sending the File to the Server

Process finished with exit code 0
- Bottom Bar:** Shows system status including temperature (64°F), battery level (23%), and date/time (06-01-2023).

The image displays two side-by-side Java code editors, likely from an IDE like IntelliJ IDEA, showing the `Server.java` and `Client.java` files for a Java application named `saloni0`.

Server.java:

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.net.ServerSocket;
import java.net.Socket;

no usages

public class Server {
    public static void main(String[] args) throws IOException {
        JFrame jFrame = new JFrame("Server");
        jFrame.setSize(400, 400);
        jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel jLabelText = new JLabel("Waiting for image from client...");
        jFrame.add(jLabelText, BorderLayout.SOUTH);
        jFrame.setVisible(true);
        ServerSocket serverSocket = new ServerSocket(port: 1234);
        Socket socket = serverSocket.accept();
        InputStream inputStream = socket.getInputStream();
        BufferedImage bufferedImage = ImageIO.read(inputStream);
        inputStream.close();
        socket.close();
        JLabel jLabelPic = new JLabel(new ImageIcon(bufferedImage));
        jLabelText.setText("Image Received.");
        jFrame.add(jLabelPic, BorderLayout.CENTER);
    }
}
```

Client.java:

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.BufferedOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;

no usages

public class Client {
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket(host: "localhost", port: 1234);
        System.out.println("Connected to server.");
        JFrame jFrame = new JFrame("Client");
        jFrame.setSize(400, 400);
        jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ImageIcon imageIcon = new ImageIcon(filename: "C:\\Users\\saloni0\\OneDrive\\Pictures\\Screenshots\\Wreck-it-retro.png");
        JLabel jLabelPic = new JLabel(imageIcon);
        JButton jButton = new JButton(text: "Send image to server.");
        jButton.addActionListener(new ActionListener() {
            @Override public void actionPerformed(ActionEvent event) {
                try{
                    OutputStream outputStream = socket.getOutputStream();
                    BufferedOutputStream bufferedOutputStream = new BufferedOutputStream(outputStream);
                    Image image = imageIcon.getImage();
                    bufferedOutputStream.write(image.getRGB());
                    bufferedOutputStream.flush();
                }
                catch (IOException e) {
                    e.printStackTrace();
                }
            }
        });
        jFrame.add(jLabelPic);
        jFrame.add(jButton);
        jFrame.setVisible(true);
    }
}
```

The code implements a simple client-server application where the client connects to a server listening on port 1234. The server receives a buffered image from the client and displays it in a `JLabel`. The client sends a buffered image to the server.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Client.java

Project Client

```

saloni0 C:\Users\saloni0
  > idea
  > out
    > Client
      > Main
      > Server
        > demoimg
        > saloni0.mml
  > src
    > Client
      > Main
      > Server
        > demoimg
        > saloni0.mml
  > External Libraries
  > Scratches and Caches

```

Client.java

```

19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

```

```

JLabel jLabelPic = new JLabel(imageIcon);
JButton jButton = new JButton("Send image to server.");
jFrame.add(jLabelPic);
jFrame.add(jButton);
jFrame.setVisible(true);
jButton.addActionListener(new ActionListener() {
    @Override public void actionPerformed(ActionEvent event) {
        try{
            OutputStream outputStream = socket.getOutputStream();
            BufferedOutputStream bufferedOutputStream = new BufferedOutputStream(outputStream);
            Image image = imageIcon.getImage();
            BufferedImage bufferedImage = new BufferedImage(image.getWidth(null), image.getHeight(null), BufferedImage.TYPE_INT_RGB);
            Graphics graphics = bufferedImage.createGraphics();
            graphics.drawImage(image, 0, 0, null);
            graphics.dispose();
            ImageIO.write(bufferedImage, "png", bufferedOutputStream);
            bufferedOutputStream.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
});
}
}

```

Version Control TODO Problems Terminal Services

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure... (2 minutes ago)

22°C Haze

Server Client

"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=50669:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin" -Dfile.encoding=UTF-8

Sending the File to the Server

Process finished with exit code 0

Server

Image Received.

The image shows two IntelliJ IDEA windows side-by-side, illustrating a Java application for transferring images between a client and a server.

Server Window:

- File: `salon00 - Server.java`
- Code snippet:

```
socket = serverSocket.accept();
InputStream inputStream = socket.getInputStream();
BufferedInputStream bufferedInputStream = new BufferedInputStream(inputStream);
Image bufferedImage = ImageIO.read(bufferedInputStream);
inputStream.close();
labelPic = new JLabel(new ImageIcon(bufferedImage));
labelPic.setText("Image Received.");
labelPic.setLayout(BorderLayout.CENTER);
```
- Status bar: "Waiting for image from client..."
- Run tab: "C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=63771:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin"

Client Window:

- File: `salon00 - Client.java`
- Code snippet:

```
ImageIO;
*
event.ActionEvent;
event.ActionListener;
image	BufferedImage;
OutputStream;
IOException;
OutputStream;
Socket;
```
- Method:

```
public void main(String[] args) throws IOException {
    Socket socket = new Socket("localhost", port 1234);
    System.out.println("Connected to server.");
}
```
- Status bar: "Connected to server."
- Run tab: "C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=63774:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin"

eclipse-workspace - Saloni/src/Saloni/Whois.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer X

> Saloni

> IRE System Library [JavaSE-17]

> src

> Saloni

> InetSocketAddress.java

> MyClient.java

> MyServer.java

> Whois.java

MyClient.java MyServer.java InetAddressTest.java Whois.java

Saloni > src > Saloni > Whois > main(String[]):void

T package Saloni;

Problems Javadoc Declaration Console

<terminated> Whois [Java Application] C:\Users\saloni\p2\pool\plugins\org.eclipse.jst\openjdkhotspot\jre\full\win32\x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (06-Jan-2023)

For more information on Whois status codes, please visit <https://icann.org/epp>

NOTICE: The expiration date displayed in this record is the date the registrar's sponsorship of the domain name registration in the registry is currently set to expire. This date does not necessarily reflect the expiration date of the domain name registrant's agreement with the sponsoring registrar. Users may consult the sponsoring registrar's Whois database to view the registrar's reported date of expiration for this registration.

TERMS OF USE: You are not authorized to access or query our Whois database through the use of electronic processes that are high-volume and automated, except as reasonably necessary to register domain names or modify existing registrations. The Data in VeriSign Global Registry Services' ("VeriSign") Whois database is provided by VeriSign for information purposes only, and to assist persons in obtaining information about or related to a domain name registration record. VeriSign does not guarantee its accuracy. By submitting a Whois query, you agree to abide by the following terms of use: You agree that you may use this Data only for lawful purposes and that under no circumstances will you use this Data to: (1) allow, enable, or otherwise support the transmission of mass unsolicited, commercial advertising or solicitations via e-mail, telephone, or fax; and (2) attempt to探查、遍历、抓取、爬取、镜像或以其他未经授权的电子或机械方式访问、复制、修改、使用或翻译该数据。The compilation, repackaging, dissemination or other use of this Data is expressly prohibited without the prior written consent of VeriSign. You agree not to use electronic processes that are automated and high-volume to access or query the Whois database except as reasonably necessary to register domain names or modify existing registrations. VeriSign reserves the right to restrict your access to the Whois database in its sole discretion to ensure operational stability. VeriSign may restrict or terminate your access to the Whois database for failure to abide by these terms of use. VeriSign reserves the right to modify these terms at any time.

The Registry database contains ONLY .COM, .NET, .EDU domains and Registrars.

62°F Clear 22:39 06-01-2023 ENG IN

eclipse-workspace - Saloni/src/Saloni/Whois.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer X

> Saloni

> IRE System Library [JavaSE-17]

> src

> Saloni

> InetSocketAddress.java

> MyClient.java

> MyServer.java

> Whois.java

MyClient.java MyServer.java InetAddressTest.java Whois.java

Saloni > src > Saloni > Whois > main(String[]):void

T package Saloni;

Problems Javadoc Declaration Console

<terminated> Whois [Java Application] C:\Users\saloni\p2\pool\plugins\org.eclipse.jst\openjdkhotspot\jre\full\win32\x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (06-Jan-2023)

Domain Name: RKNEC.EDU
Registrar: Educause, Inc.
Registrar WHOIS Server: whois.educause.net
Registrar URL: <http://www.educause.edu/edudomain>
Updated Date: 2022-06-03T09:12:02Z
Creation Date: 1999-02-16T05:00:00Z
Registry Expiry Date: 2023-07-31T11:59:59Z
Registrar: Educause
Registrar IANA ID: 365
Registrar Abuse Contact Email:
Registrar Abuse Contact Phone:
Domain Status: clientTransferProhibited <https://icann.org/epp#clientTransferProhibited>
Name Server: NS1.FOXGLOVE.ARVIXE.COM
Name Server: NS2.FOXGLOVE.ARVIXE.COM
DNSSEC: unsigned
URL of the ICANN Whois Inaccuracy Complaint Form: <https://www.icann.org/wicf/>

>>> Last update of whois database: 2023-01-06T16:54:57Z <<

For more information on Whois status codes, please visit <https://icann.org/epp>

NOTICE: The expiration date displayed in this record is the date the registrar's sponsorship of the domain name registration in the registry is currently set to expire. This date does not necessarily reflect the expiration date of the domain name registrant's agreement with the sponsoring registrar. Users may consult the sponsoring registrar's Whois database to view the registrar's reported date of expiration for this registration.

TERMS OF USE: You are not authorized to access or query our Whois database through the use of electronic processes that are high-volume and automated, except as reasonably necessary to register domain names or modify existing registrations. The Data in VeriSign Global Registry Services' ("VeriSign") Whois database is provided by VeriSign for information purposes only, and to assist persons in obtaining information about or related to a domain name registration record. VeriSign does not guarantee its accuracy. By submitting a Whois query, you agree to abide by the following terms of use: You agree that you may use this Data only for lawful purposes and that under no circumstances will you use this Data to:

62°F Clear 22:39 06-01-2023 ENG IN

eclipse-workspace - Saloni/src/Saloni/MyClient.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X MyClient.java X MyServer.java X
Saloni
src
MyClient.java
MyServer.java
```

```
1 package Saloni;
2 import java.io.*;
3 import java.net.*;
4 public class MyClient{
5     public static void main(String[] args){
6         try{
7             Socket s=new Socket("localhost",6666);
8             DataOutputStream dout=new DataOutputStream(s.getOutputStream());
9             dout.writeUTF("HelloSaloni");
10            dout.flush();
11            dout.close();
12            s.close();
13        }catch(Exception e){System.out.println(e);}
14    }
15 }
```

Problems Javadoc Declaration Console X
<terminated> MyServer [Java Application] C:\Users\saloni\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (09-Dec-2022, 7:18:37 pm – 7:18:51 pm) [pid: HELLOSALONI]

eclipse-workspace - Saloni/src/Saloni/MyServer.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X MyClient.java X MyServer.java X
Saloni
src
MyClient.java
MyServer.java
```

```
1 package Saloni;
2 import java.io.*;
3 import java.net.*;
4 public class MyServer {
5     public static void main(String[] args){
6         try{
7             ServerSocket ss=new ServerSocket(6666);
8             Socket s=s.accept(); //establishes connection
9             DataInputStream dis=new DataInputStream(s.getInputStream());
10            String str=dis.readUTF();
11            for(int i=0;i<str.length();i++){
12                char ch=str.charAt(i);
13                ch=(char)(ch-32);
14                System.out.print(ch);
15            }
16        }catch(Exception e){System.out.println(e);}
17    }
18 }
```

Problems Javadoc Declaration Console X
<terminated> MyServer [Java Application] C:\Users\saloni\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (09-Dec-2022, 7:18:37 pm – 7:18:51 pm) [pid: HELLOSALONI]

