```
//TCP Client
import java.io.*;
import java.net.Socket;
import java.net.UnknownHostException;
import java.nio.charset.StandardCharsets;

public class FileTransferClient {
    // Corresponds to #define SERVER_PORT
1238
    private static final int SERVER_PORT =
1238;
    // Corresponds to the buffer size char
buf[2000] and the read size.
    private static final int BUFFER_SIZE =
2000;

    public static void main(String[] args) {
        // Corresponds to if(argc!=3)
validation
        if (args.length != 2) {
            System.out.println("usage: java
FileTransferClient <server_name>
<file_name>");
            // Corresponds to an implicit
exit after printing usage
```

```
            return;
        }

        String serverName = args[0]; //
Corresponds to argv[1]
        String fileName = args[1];   //
Corresponds to argv[2]

        // Socket handles the connection,
similar to gethostbyname(), socket(), and
connect()
        // The try-with-resources statement
ensures the Socket is automatically closed
(like close(s))
        try (
            Socket socket = new
Socket(serverName, SERVER_PORT);

            // PrintWriter to send the
filename as a line of text to the server
            PrintWriter out = new
PrintWriter(socket.getOutputStream(), true,
StandardCharsets.UTF_8);
```

```
            // InputStream to read the raw
file data (bytes) from the server
            InputStream in =
socket.getInputStream();

            // DataOutputStream is used here
to write to standard output (write(1, buf,
bytes))
            DataOutputStream stdOut = new
DataOutputStream(new
BufferedOutputStream(System.out));
        ) {
            // Corresponds to write(s,
argv[2], strlen(argv[2])+1)
            // println() automatically sends
the string followed by a newline and flushes
the buffer (due to auto-flush=true in
PrintWriter)
            out.println(fileName);

            byte[] buffer = new
byte[BUFFER_SIZE];
            int bytesRead;
            // Corresponds to the while(1)
loop with bytes=read(s, buf, 2000)
            while ((bytesRead =
in.read(buffer)) != -1) {
                // Corresponds to write(1,
buf, bytes)
                stdOut.write(buffer, 0,
bytesRead);
            }

            // Ensure all buffered output is
written to the console before exiting
            stdOut.flush();

        } catch (UnknownHostException e) {
            // Catches errors related to
gethostbyname failed
            System.err.println("Error:
Unknown host " + serverName);
        } catch (IOException e) {
            // Catches errors related to
connect Failed or socket issues
            System.err.println("Error
connecting to server or reading/writing
data: " + e.getMessage());
```

```
        }

        // Corresponds to exit(0) when
communication is complete
    }
}
```