```java
import io.opentelemetry.api.GlobalOpenTelemetry;
import io.opentelemetry.api.trace.Span;
import io.opentelemetry.api.trace.Tracer;
import io.opentelemetry.api.trace.TracerProvider;
import io.opentelemetry.context.Scope;

import java.io.*;
import java.net.Socket;
import java.util.UUID;

public class client {
    public static void main(String[] args) {
        System.setProperty("otel.tracer.provider", "io.opentelemetry.api.trace.propagation.B3Propagator$Factory");

        try {
            System.out.println("Connecting to the server...");

            // Create a socket and connect to the server on localhost, port 8080
            Socket socket = new Socket("localhost", 8080);

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

            // Start a new span for the client operation
            TracerProvider tracerProvider = GlobalOpenTelemetry.getTracerProvider();
            Tracer tracer = tracerProvider.get("client-tracer");

            // Manually create a span for the client operation
            Span span = tracer.spanBuilder("client-operation").startSpan();

            // Create a scope to manage the span's
```

```
30 lifecycle
31             try (Scope scope = span.makeCurrent()) {
32                 // Create input and output streams
   for communication with the server
33                 BufferedReader in = new
   BufferedReader(new InputStreamReader(socket.
   getInputStream()));
34                 PrintWriter out = new PrintWriter(
   socket.getOutputStream(), true);
35
36                 // Specify the folder path on the
   client side
37                 String desktopPath = System.
   getProperty("user.home") + "/Desktop" + "/Files";
38                 File folder = new File(desktopPath);
39
40                 // List files in the folder
41                 File[] files = folder.listFiles();
42                 if (files == null) {
43                     System.err.println("No files
   found in the folder: " + desktopPath);
44                     return;
45                 }
46
47                 // Send the number of files to the
   server
48                 out.println(files.length);
49
50                 // Send up to 20 files at a time
51                 int filesToSend = Math.min(20, files.
   length);
52                 for (int i = 0; i < filesToSend; i
   ++) {
53                     File file = files[i];
54
55                     // Send the file name to the
   server
56                     out.println(file.getName());
57
58                     // Read the file content and send
    it to the server
```

```
59                      try (BufferedReader fileReader =
   new BufferedReader(new FileReader(file))) {
60                          String line;
61                          while ((line = fileReader.
   readLine()) != null) {
62                              out.println(line);
63                          }
64                      }
65
66                      System.out.println("File sent: "
    + file.getName());
67                  }
68
69                  // Close the streams and socket
70                  in.close();
71                  out.close();
72                  socket.close();
73              } finally {
74                  // End the span when the operation is
    complete
75                  span.end();
76              }
77          } catch (IOException e) {
78              System.err.println("Connection failed.
   Make sure the server is running and check your
   firewall settings.");
79              e.printStackTrace();
80          }
81      }
82 }
83
```