```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.ServerSocket;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
import java.util.UUID;
import java.util.zip.GZIPInputStream;

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

        try {
            // Create a server socket listening on port 8080
            ServerSocket serverSocket = new ServerSocket(8080);
            System.out.println("Server is listening on port 8080...");

            while (true) {
                // Wait for a client to connect
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected.");

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

```

```
32                  // Manually create a span for the
   server operation
33                  Span span = tracer.spanBuilder("
   server-operation").startSpan();
34
35                  // Create a scope to manage the span'
   s lifecycle
36                  try (Scope scope = span.makeCurrent
   ();
37                       InputStream inputStream =
   clientSocket.getInputStream();
38                       ObjectInputStream
   objectInputStream = new ObjectInputStream(new
   GZIPInputStream(inputStream))) {
39
40                      // Process the client's request
   with compression
41                      processClientRequest(
   objectInputStream);
42                  } finally {
43                      // End the span when the
   operation is complete
44                      span.end();
45                  }
46
47                  // Close the client socket
48                  clientSocket.close();
49              }
50          } catch (IOException | ClassNotFoundException
    e) {
51              e.printStackTrace();
52          }
53      }
54
55      private static void processClientRequest(
   ObjectInputStream objectInputStream) throws
   IOException, ClassNotFoundException {
56          // Read the number of files to expect
57          int numFiles = objectInputStream.readInt();
58          System.out.println("Expecting " + numFiles +
   " files from the client.");
```

```
59
60          // Receive files from the client
61          for (int i = 0; i < numFiles; i++) {
62              // Generate a random file name
63              String fileName = UUID.randomUUID().
   toString() + ".txt";
64
65              // Read the compressed file content from
   the client and decompress it
66              byte[] compressedContent = (byte[])
   objectInputStream.readObject();
67              String fileContent = decompress(
   compressedContent);
68
69              // Save the decompressed file content to
   a file
70              try (FileOutputStream fileOutputStream =
   new FileOutputStream("received_" + fileName)) {
71                  fileOutputStream.write(fileContent.
   getBytes(StandardCharsets.UTF_8));
72                  System.out.println("File content
   saved to received_" + fileName);
73              } catch (IOException e) {
74                  System.err.println("Error saving file
    content to received_" + fileName);
75                  e.printStackTrace();
76              }
77          }
78      }
79
80      private static String decompress(byte[]
   compressedData) throws IOException {
81          try (ByteArrayInputStream
   byteArrayInputStream = new ByteArrayInputStream(
   compressedData);
82               GZIPInputStream gzipInputStream = new
   GZIPInputStream(byteArrayInputStream);
83               InputStreamReader inputStreamReader =
   new InputStreamReader(gzipInputStream,
   StandardCharsets.UTF_8);
84               BufferedReader bufferedReader = new
```

```
84 BufferedReader(inputStreamReader)) {
85
86             StringBuilder decompressedContent = new
   StringBuilder();
87             String line;
88             while ((line = bufferedReader.readLine
   ()) != null) {
89                 decompressedContent.append(line).
   append("\n");
90             }
91
92             return decompressedContent.toString();
93         }
94     }
95 }
96
```