```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.zip.CRC32;
import java.util.zip.Checksum;
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");

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

                // Create a scope to manage the span's lifecycle
                try (Scope scope = span.makeCurrent();
                     InputStream inputStream = clientSocket.getInputStream();
                     ObjectInputStream objectInputStream = new ObjectInputStream(
                             new GZIPInputStream(inputStream))) {

                    // Process the client's request with compression and checksum verification
                    processClientRequest(objectInputStream);
                } finally {
                    // End the span when the operation is complete
                    span.end();
                }

                // Close the client socket
                clientSocket.close();
            }
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    private static void processClientRequest(ObjectInputStream objectInputStream) throws IOException, ClassNotFoundException {
        // Read the number of files to expect
        int numFiles = objectInputStream.readInt();
        System.out.println("Expecting " + numFiles + " files from the client.");

        // Receive files from the client
        for (int i = 0; i < numFiles; i++) {
            // Read the compressed file content from the client
            byte[] compressedContent = (byte[]) objectInputStream.readObject();

            // Read the checksum from the client
            long receivedChecksum = objectInputStream.readLong();

            // Decompress the content
            String fileContent = decompress(compressedContent);

            // Verify the checksum
            long computedChecksum = computeChecksum(fileContent);
            if (receivedChecksum == computedChecksum) {
                System.out.println("Checksum verification passed for file " + (i + 1));
            } else {
                System.out.println("Checksum verification failed for file " + (i + 1));
            }
        }
    }

    private static String decompress(byte[] compressedContent) throws IOException {
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        try (GZIPInputStream gzipInputStream = new GZIPInputStream(new ByteArrayInputStream(compressedContent))) {
            byte[] buffer = new byte[1024];
            int len;
            while ((len = gzipInputStream.read(buffer)) != -1) {
                byteArrayOutputStream.write(buffer, 0, len);
            }
        }
        return byteArrayOutputStream.toString(String.valueOf(StandardCharsets.UTF_8));
    }

    private static long computeChecksum(String data) {
        Checksum checksum = new CRC32();
        checksum.update(data.getBytes(StandardCharsets.UTF_8), 0, data.length());
        return checksum.getValue();
    }
}

```