```java
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.UUID;
import java.util.zip.GZIPOutputStream;

public class server {
    public static void main(String[] args) {
        try {
            // Create a server socket listening on port 8080
            ServerSocket serverSocket = new ServerSocket(8080);
            System.out.println("Server is listening on port 8080...");

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

            // Create input stream for communication with the client
            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

            // Read the number of files to expect
            int numFiles;
            try {
                numFiles = Integer.parseInt(in.readLine());
            } catch (NumberFormatException e) {
                System.err.println("Error reading the number of files from the client.");
                return;
            }
            System.out.println("Expecting " + numFiles + " files from the client.");

            // Receive files from the client
```

```
32              for (int i = 0; i < numFiles; i++) {
33                  // Generate a random file name
34                  String fileName = UUID.randomUUID().
   toString() + ".txt";
35
36                  // Read the file content from the
   client
37                  StringBuilder fileContent = new
   StringBuilder();
38                  String line;
39                  while ((line = in.readLine()) != null
    && !line.equals("END_OF_FILE")) {
40                      fileContent.append(line).append("
   \n");
41                  }
42
43                  // Introduce a bug in the compression
    algorithm
44                  byte[] compressedContent =
   corruptCompression(fileContent.toString());
45
46                  // Save the corrupted file content to
    a file
47                  try (FileOutputStream
   fileOutputStream = new FileOutputStream("received_"
    + fileName)) {
48                      fileOutputStream.write(
   compressedContent);
49                      System.out.println("Corrupted
   file content saved to received_" + fileName);
50                  } catch (IOException e) {
51                      System.err.println("Error saving
   corrupted file content to received_" + fileName);
52                      e.printStackTrace();
53                  }
54              }
55
56              // Close the streams and sockets
57              in.close();
58              clientSocket.close();
59              serverSocket.close();
```

```
60
61          } catch (IOException e) {
62              e.printStackTrace();
63          }
64      }
65
66      private static byte[] corruptCompression(String
    data) {
67          try (ByteArrayOutputStream
    byteArrayOutputStream = new ByteArrayOutputStream();
68               GZIPOutputStream gzipOutputStream = new
     GZIPOutputStream(byteArrayOutputStream)) {
69
70              // Introduce a bug by not writing the
    original data
71              gzipOutputStream.write("Corrupted data".
    getBytes());
72
73              return byteArrayOutputStream.toByteArray
    ();
74          } catch (IOException e) {
75              System.err.println("Error in corrupting
    compression.");
76              e.printStackTrace();
77              return new byte[0];
78          }
79      }
80 }
81
```