CRC:1

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class TCPCRCClient1 {
   private static final String CRC_POLYNOMIAL = "100000111";
   public static void main(String[] args) {
      String serverAddress = "127.0.0.1";
      int serverPort = 54321;
      Scanner userInputScanner = new Scanner(System.in);
      try (Socket clientSocket = new Socket(serverAddress, serverPort);
          BufferedReader serverReader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
          PrintWriter serverWriter = new PrintWriter(clientSocket.getOutputStream(), true)
      ) {
         System.out.println("Server: " + serverReader.readLine());

         System.out.print("Enter binary message: ");
         String userMessage = userInputScanner.nextLine();
         String messagePadded = userMessage + "00000000";
         String computedCRC = computeCRC(messagePadded);
         String fullMessage = userMessage + computedCRC;

         System.out.println("Sending encoded message: " + fullMessage);
         serverWriter.println(fullMessage);

         System.out.println("Server: " + serverReader.readLine());
      } catch (IOException ex) {
         System.out.println("Connection error: " + ex.getMessage());
      }
```

```
        userInputScanner.close();
    }
    private static String computeCRC(String inputBits) {
        char[] bitArray = inputBits.toCharArray();
        int totalLength = bitArray.length;
        int polynomialLength = CRC_POLYNOMIAL.length();
        for (int i = 0; i <= totalLength - polynomialLength; i++) {
            if (bitArray[i] == '1') {
                for (int j = 0; j < polynomialLength; j++) {
                    bitArray[i + j] = (bitArray[i + j] == CRC_POLYNOMIAL.charAt(j)) ? '0' : '1';
                }
            }
        }
        int crcIndexStart = totalLength - polynomialLength + 1;
        StringBuilder crcResult = new StringBuilder();
        for (int i = crcIndexStart; i < totalLength; i++) {
            crcResult.append(bitArray[i]);
        }

        return crcResult.toString();
    }
}
```

CRC:2

```
import java.io.*;
import java.net.*;
public class TCPCRCServer1 {
    private static final String CRC_GENERATOR = "100000111";
    public static void main(String[] args) {
        int serverPort = 54321;
        try (ServerSocket serverSocket = new ServerSocket(serverPort)) {
            System.out.println("Server started on port " + serverPort);
```

```
        Socket clientSocket = serverSocket.accept();
        System.out.println("Client connected.");
        BufferedReader clientInput = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        PrintWriter clientOutput = new PrintWriter(clientSocket.getOutputStream(), true);
        clientOutput.println("Connected to CRC Server!");
        String receivedCodeword = clientInput.readLine();
        System.out.println("Received codeword: " + receivedCodeword);
        boolean isValid = verifyCRC(receivedCodeword);
        if (isValid) {
            clientOutput.println("No errors found in transmission.");
        } else {
            clientOutput.println("Error detected in transmission.");
        }

        clientSocket.close();
    } catch (IOException e) {
        System.out.println("Server error: " + e.getMessage());
    }
}
private static boolean verifyCRC(String codeword) {
    char[] bits = codeword.toCharArray();
    int generatorLength = CRC_GENERATOR.length();
    int totalBits = bits.length;

    for (int i = 0; i <= totalBits - generatorLength; i++) {
        if (bits[i] == '1') {
            for (int j = 0; j < generatorLength; j++) {
                bits[i + j] = (bits[i + j] == CRC_GENERATOR.charAt(j)) ? '0' : '1';
            }
        }
```

```
        }
        for (int i = totalBits - generatorLength + 1; i < totalBits; i++) {
            if (bits[i] == '1') {
                return false;
            }
        }
        return true; // No errors
    }
}
```

Bit:1

```
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class TCPBitStuffingClient {
    public static void main(String[] args) {
        String serverAddress = "127.0.0.1";
        int serverPort = 54321;

        try (
            Socket clientSocket = new Socket(serverAddress, serverPort);
            PrintWriter toServer = new PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader fromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            Scanner userInput = new Scanner(System.in)
        ) {
            System.out.print("Enter bit sequence: ");
            String inputBits = userInput.nextLine();

            String stuffedData = applyBitStuffing(inputBits);
            System.out.println("Stuffed Data Sent: " + stuffedData);
```

```
        toServer.println(stuffedData); // Send stuffed data to server

        String serverResponse = fromServer.readLine(); // Get server reply
        System.out.println("Server Response (Destuffed): " + serverResponse);

    } catch (IOException e) {
        System.out.println("Connection error: " + e.getMessage());
    }
}

// Method to apply bit stuffing
private static String applyBitStuffing(String data) {
    StringBuilder stuffed = new StringBuilder("01111110"); // Start flag
    int oneCount = 0;

    for (char bit : data.toCharArray()) {
        stuffed.append(bit);
        if (bit == '1') {
            oneCount++;
            if (oneCount == 5) {
                stuffed.append('0'); // Stuff '0' after five '1's
                oneCount = 0;
            }
        } else {
            oneCount = 0;
        }
    }

    stuffed.append("01111110"); // End flag
    return stuffed.toString();
```

```
    }
}
```

Bit 2:

```
import java.io.*;
import java.net.*;


public class TCPBitStuffingServer {
    public static void main(String[] args) {
        int serverPort = 54321;


        try (ServerSocket serverSocket = new ServerSocket(serverPort)) {
            System.out.println("Server is running on port " + serverPort);


            while (true) {
                try (
                    Socket clientSocket = serverSocket.accept();
                    BufferedReader fromClient = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    PrintWriter toClient = new PrintWriter(clientSocket.getOutputStream(), true)
                ) {
                    String receivedData = fromClient.readLine();
                    System.out.println("Received Stuffed Data: " + receivedData);


                    String originalData = removeBitStuffing(receivedData);
                    toClient.println(originalData); // Send destuffed data back
                    System.out.println("Destuffed Data Sent: " + originalData);
                }
            }


        } catch (IOException e) {
            System.out.println("Server error: " + e.getMessage());
```

```
        }
    }

    // Method to remove bit stuffing from the received data
    private static String removeBitStuffing(String stuffedData) {
        StringBuilder destuffed = new StringBuilder();
        int oneCount = 0;

        // Skip the starting and ending flag (8 bits each)
        for (int i = 8; i < stuffedData.length() - 8; i++) {
            char bit = stuffedData.charAt(i);
            destuffed.append(bit);

            if (bit == '1') {
                oneCount++;
                if (oneCount == 5) {
                    i++; // Skip the stuffed '0'
                    oneCount = 0;
                }
            } else {
                oneCount = 0;
            }
        }

        return destuffed.toString();
    }
}
```

Byte:1

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
```

```
public class TCPByteStuffingSender {
    private static final String FLAG_BYTE = "01111110";
    private static final String ESCAPE_BYTE = "01111101";

    public static void main(String[] args) {
        String serverAddress = "127.0.0.1";
        int serverPort = 6000;

        try (
            Socket clientSocket = new Socket(serverAddress, serverPort);
            PrintWriter toServer = new PrintWriter(clientSocket.getOutputStream(), true);
            Scanner userInput = new Scanner(System.in)
        ) {
            System.out.print("Enter binary data (8-bit bytes): ");
            String inputBits = userInput.nextLine();

            String stuffedMessage = applyByteStuffing(inputBits);
            System.out.println("Stuffed Data Sent: " + stuffedMessage);

            toServer.println(stuffedMessage); // Send to server

        } catch (IOException e) {
            System.out.println("Connection error: " + e.getMessage());
        }
    }

    // Method to apply byte stuffing
    public static String applyByteStuffing(String data) {
        StringBuilder stuffed = new StringBuilder(FLAG_BYTE); // Start with flag
```

```
        for (int i = 0; i < data.length(); i += 8) {
            String byteChunk = data.substring(i, Math.min(i + 8, data.length()));

            // If it's a flag or escape byte, prepend an escape byte
            if (byteChunk.equals(FLAG_BYTE) || byteChunk.equals(ESCAPE_BYTE)) {
                stuffed.append(ESCAPE_BYTE);
            }

            stuffed.append(byteChunk);
        }

        stuffed.append(FLAG_BYTE); // End with flag
        return stuffed.toString();
    }
}
```

Byte:2

```
import java.io.*;
import java.net.*;

public class TCPByteStuffingReceiver {
    private static final String FLAG_BYTE = "01111110";
    private static final String ESCAPE_BYTE = "01111101";

    public static void main(String[] args) {
        int serverPort = 6000;

        try (ServerSocket serverSocket = new ServerSocket(serverPort)) {
            System.out.println("Server is running on port " + serverPort);
            System.out.println("Waiting for client connection...");

            Socket clientSocket = serverSocket.accept();
```

```
            System.out.println("Client connected!");

            BufferedReader fromClient = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            String receivedStuffedData = fromClient.readLine();

            System.out.println("Received Stuffed Data: " + receivedStuffedData);

            String originalData = removeByteStuffing(receivedStuffedData);
            System.out.println("Unstuffed Data: " + originalData);

            clientSocket.close();

        } catch (IOException e) {
            System.out.println("Server error: " + e.getMessage());
        }
    }

    // Method to remove byte stuffing
    public static String removeByteStuffing(String stuffedData) {
        // Remove starting and ending FLAG_BYTE
        String cleanData = stuffedData.substring(FLAG_BYTE.length(), stuffedData.length() -
FLAG_BYTE.length());
        StringBuilder unstuffedData = new StringBuilder();

        for (int i = 0; i < cleanData.length(); i += 8) {
            String currentByte = cleanData.substring(i, Math.min(i + 8, cleanData.length()));

            if (currentByte.equals(ESCAPE_BYTE)) {
                i += 8; // Skip escape byte
                currentByte = cleanData.substring(i, Math.min(i + 8, cleanData.length()));
            }
```

```
            unstuffedData.append(currentByte);
        }

        return unstuffedData.toString();
    }
}
```

TCP:1

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;

public class Client {
    public static void main(String[] args) {
        try {
            // Connect to server at localhost on port 5555
            Socket s = new Socket("localhost", 5555);

            // Create input and output streams
            DataInputStream din = new DataInputStream(s.getInputStream());
            DataOutputStream dout = new DataOutputStream(s.getOutputStream());

            // Message to send
            String str = "hello world";

            // Send message to server
            dout.writeUTF(str);
            dout.flush();

            // Close connection
```

```
            s.close();

            System.out.println("Message sent to server: " + str);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TCP 2:

```
import java.io.DataInputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) {
        try {
            // Create a server socket on port 5555
            ServerSocket ss = new ServerSocket(5555);
            System.out.println("Server is running and waiting for a client...");

            // Accept a client connection
            Socket s = ss.accept();
            System.out.println("Client connected.");

            // Read data from client
            DataInputStream din = new DataInputStream(s.getInputStream());
            String message = din.readUTF();

            // Display received message
            System.out.println("Received from client: " + message);
```

```
            // Close connection
            s.close();
            ss.close();

            System.out.println("Connection closed.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```