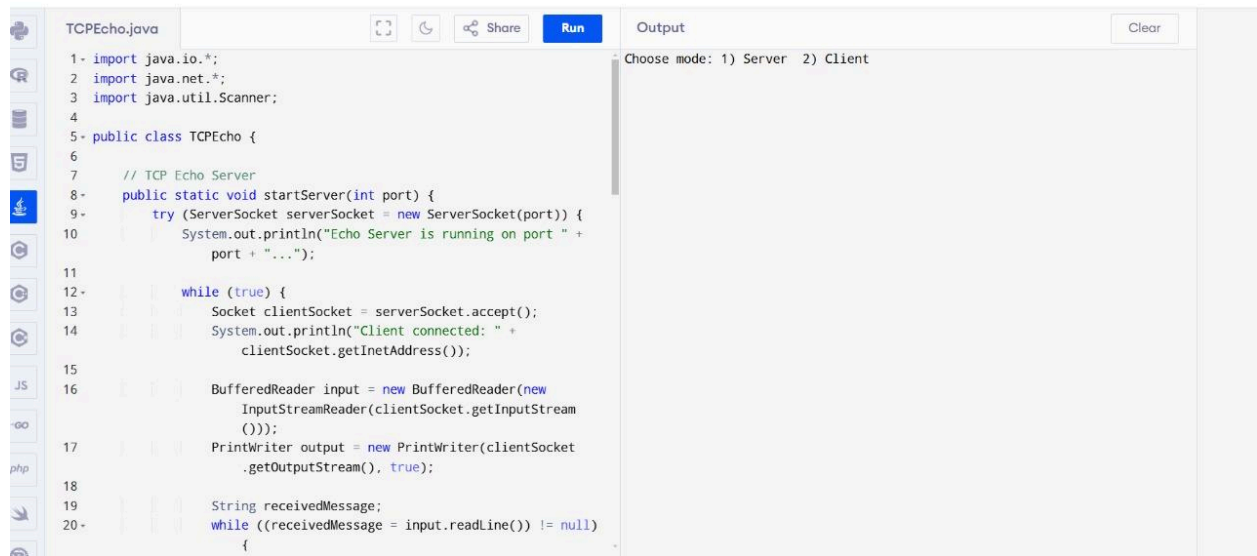
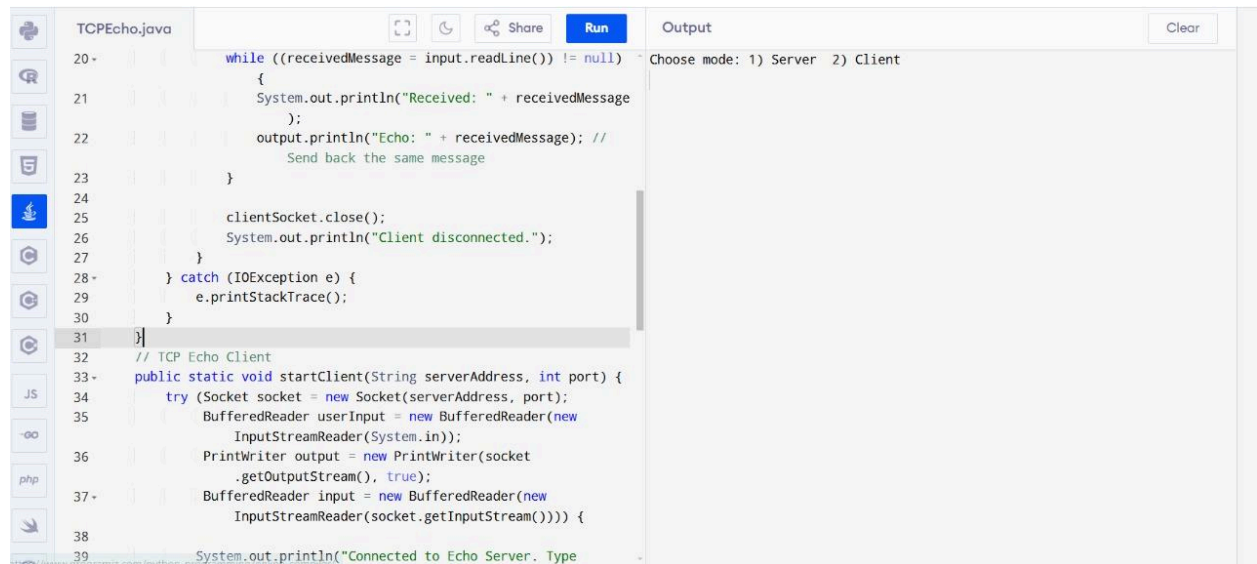


29)



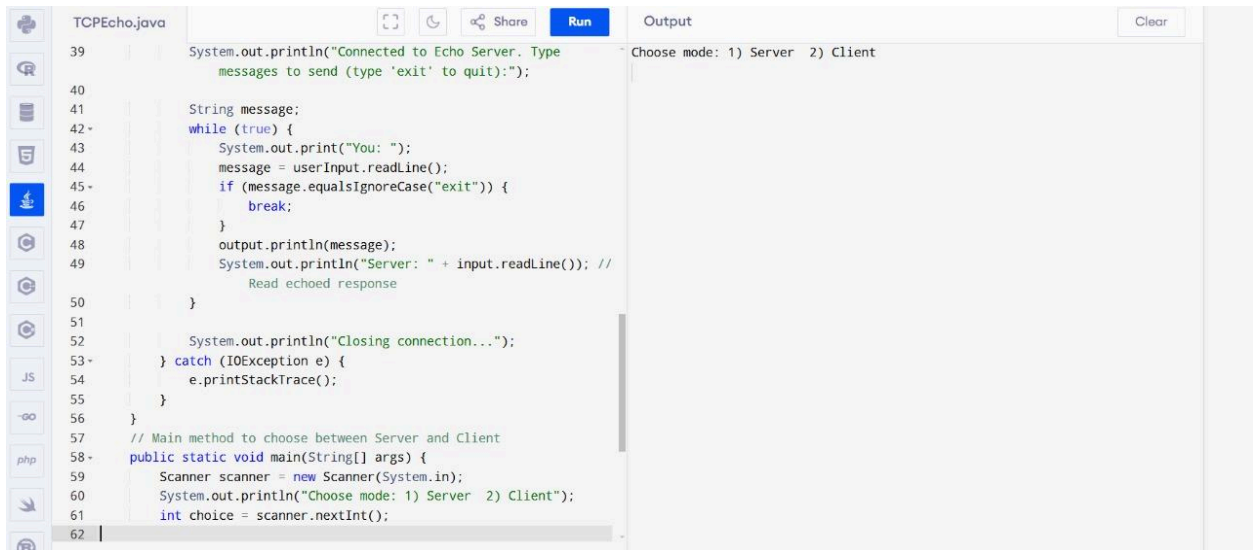
```
1- import java.io.*;
2- import java.net.*;
3- import java.util.Scanner;
4
5- public class TCPEcho {
6
7-     // TCP Echo Server
8-     public static void startServer(int port) {
9-         try (ServerSocket serverSocket = new ServerSocket(port)) {
10-             System.out.println("Echo Server is running on port " +
11-                 port + "...");
12-
13-             while (true) {
14-                 Socket clientSocket = serverSocket.accept();
15-                 System.out.println("Client connected: " +
16-                     clientSocket.getInetAddress());
17-
18-                 BufferedReader input = new BufferedReader(new
19-                     InputStreamReader(clientSocket.getInputStream()
20-                     ()));
21-                 PrintWriter output = new PrintWriter(clientSocket
22-                     .getOutputStream(), true);
23-
24-                 String receivedMessage;
25-                 while ((receivedMessage = input.readLine()) != null)
26-                     {
```

Output: Choose mode: 1) Server 2) Client



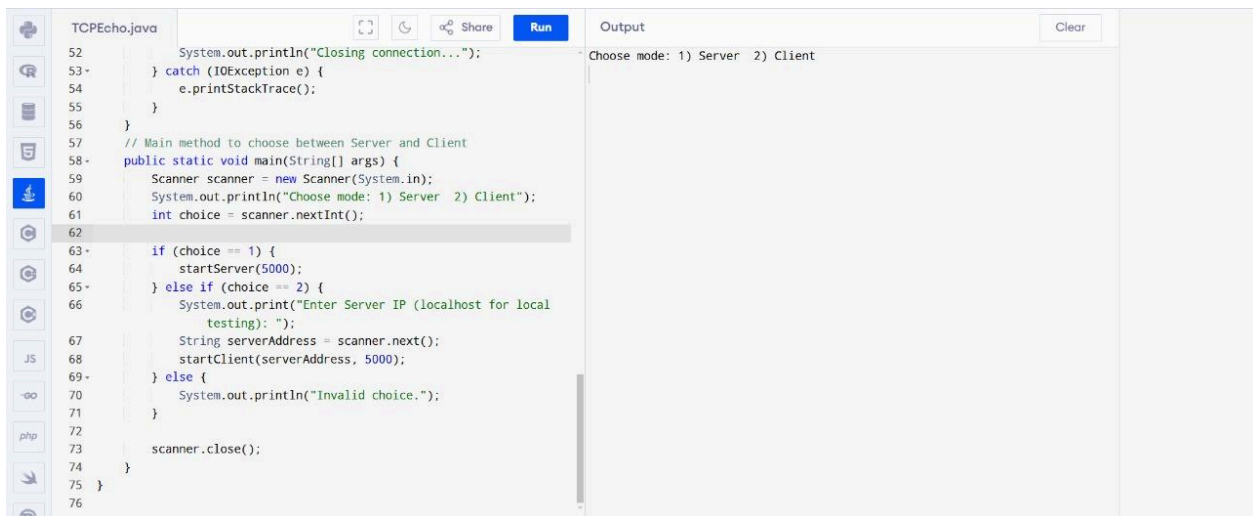
```
20-         while ((receivedMessage = input.readLine()) != null)
21-             {
22-                 System.out.println("Received: " + receivedMessage
23-                     );
24-                 output.println("Echo: " + receivedMessage); //
25-                     Send back the same message
26-             }
27-         clientSocket.close();
28-         System.out.println("Client disconnected.");
29-     } catch (IOException e) {
30-         e.printStackTrace();
31-     }
32- }
33- // TCP Echo Client
34- public static void startClient(String serverAddress, int port) {
35-     try (Socket socket = new Socket(serverAddress, port);
36-         BufferedReader userInput = new BufferedReader(new
37-             InputStreamReader(System.in));
38-         PrintWriter output = new PrintWriter(socket
39-             .getOutputStream(), true);
40-         BufferedReader input = new BufferedReader(new
41-             InputStreamReader(socket.getInputStream()))) {
42-         System.out.println("Connected to Echo Server. Type
```

Output: Choose mode: 1) Server 2) Client



```
TCPEcho.java
39      System.out.println("Connected to Echo Server. Type
40      messages to send (type 'exit' to quit):");
41
42      String message;
43      while (true) {
44          System.out.print("You: ");
45          message = userInput.readLine();
46          if (message.equalsIgnoreCase("exit")) {
47              break;
48          }
49          output.println(message);
50          System.out.println("Server: " + input.readLine()); //
51          Read echoed response
52      }
53      System.out.println("Closing connection...");
54      } catch (IOException e) {
55          e.printStackTrace();
56      }
57      // Main method to choose between Server and Client
58      public static void main(String[] args) {
59          Scanner scanner = new Scanner(System.in);
60          System.out.println("Choose mode: 1) Server 2) Client");
61          int choice = scanner.nextInt();
62      }
```

Output: Choose mode: 1) Server 2) Client



```
TCPEcho.java
52      System.out.println("Closing connection...");
53      } catch (IOException e) {
54          e.printStackTrace();
55      }
56      // Main method to choose between Server and Client
57      public static void main(String[] args) {
58          Scanner scanner = new Scanner(System.in);
59          System.out.println("Choose mode: 1) Server 2) Client");
60          int choice = scanner.nextInt();
61
62          if (choice == 1) {
63              startServer(5000);
64          } else if (choice == 2) {
65              System.out.print("Enter Server IP (localhost for local
66              testing): ");
67              String serverAddress = scanner.next();
68              startClient(serverAddress, 5000);
69          } else {
70              System.out.println("Invalid choice.");
71          }
72
73          scanner.close();
74      }
75      }
76      }
```

Output: Choose mode: 1) Server 2) Client

30) server

```
Server is listening on port 8080...
```

main.c

Share

Run

```
39     perror("Failed to create socket");
40     exit(EXIT_FAILURE);
41 }
42
43 // Define server address
44 server_addr.sin_family = AF_INET;
45 server_addr.sin_addr.s_addr = INADDR_ANY;
46 server_addr.sin_port = htons(PORT);
47
48 // Bind the socket to the specified IP and port
49 if (bind(server_socket, (struct sockaddr*)&server_addr,
50         sizeof(server_addr)) < 0) {
51     perror("Bind failed");
52     close(server_socket);
53     exit(EXIT_FAILURE);
54 }
55
56 // Listen for incoming connections
57 if (listen(server_socket, 3) < 0) {
58     perror("Listen failed");
59 }
```

Output

Server is listening on port 8080...

main.c

Share

Run

```
57     perror("Listen failed");
58     close(server_socket);
59     exit(EXIT_FAILURE);
60 }
61
62 printf("Server is listening on port %d...\n", PORT);
63
64 while (1) {
65     // Accept a new connection
66     client_socket = accept(server_socket, (struct sockaddr
67                             *)&client_addr, &client_len);
68     if (client_socket < 0) {
69         perror("Accept failed");
70         close(server_socket);
71         exit(EXIT_FAILURE);
72     }
73
74     printf("New client connected.\n");
75     handle_client(client_socket);
76 }
```

Output

Server is listening on port 8080...

```
63
64 while (1) {
65     // Accept a new connection
66     client_socket = accept(server_socket, (struct sockaddr
        *)&client_addr, &client_len);
67
68     if (client_socket < 0) {
69         perror("Accept failed");
70         close(server_socket);
71         exit(EXIT_FAILURE);
72     }
73
74     printf("New client connected.\n");
75     handle_client(client_socket);
76 }
77
78 close(server_socket);
79 return 0;
80 }
```

Client response

```
main.c      Run      Output
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  #include <arpa/inet.h>
6
7  #define PORT 8080
8  #define BUFFER_SIZE 1024
9
10 int main() {
11     int client_socket;
12     struct sockaddr_in server_addr; // Fixed this line
13     char buffer[BUFFER_SIZE];
14
15     // Create socket
16     client_socket = socket(AF_INET, SOCK_STREAM, 0);
17     if (client_socket == -1) {
18         perror("Socket creation failed");
19         exit(EXIT_FAILURE);
20     }
```

```
=== Code Exited With Errors ===
```

```
main.c
20 }
21
22 // Define server address
23 server_addr.sin_family = AF_INET;
24 server_addr.sin_port = htons(PORT);
25 server_addr.sin_addr.s_addr = inet_addr("127.0.0.1"); //
    Connect to localhost
26
27 // Connect to the server
28 if (connect(client_socket, (struct sockaddr*)&server_addr,
    sizeof(server_addr)) < 0) {
29     perror("Connection failed");
30     close(client_socket);
31     exit(EXIT_FAILURE);
32 }
33
34 printf("Connected to the server.\n");
35
36 while (1) {
37     // Send message to server
```

Output

Connection failed: Connection refused

=== Code Exited With Errors ===

```
main.c
33
34 printf("Connected to the server.\n");
35
36 while (1) {
37     // Send message to server
38     printf("You: ");
39     fgets(buffer, BUFFER_SIZE, stdin);
40     send(client_socket, buffer, strlen(buffer), 0);
41
42     // Receive response from server
43     memset(buffer, 0, BUFFER_SIZE);
44     recv(client_socket, buffer, BUFFER_SIZE, 0);
45     printf("Server: %s\n", buffer);
46 }
47
48 close(client_socket);
49 return 0;
50 }
51
```

Output

Connection failed: Connect

=== Code Exited With Error