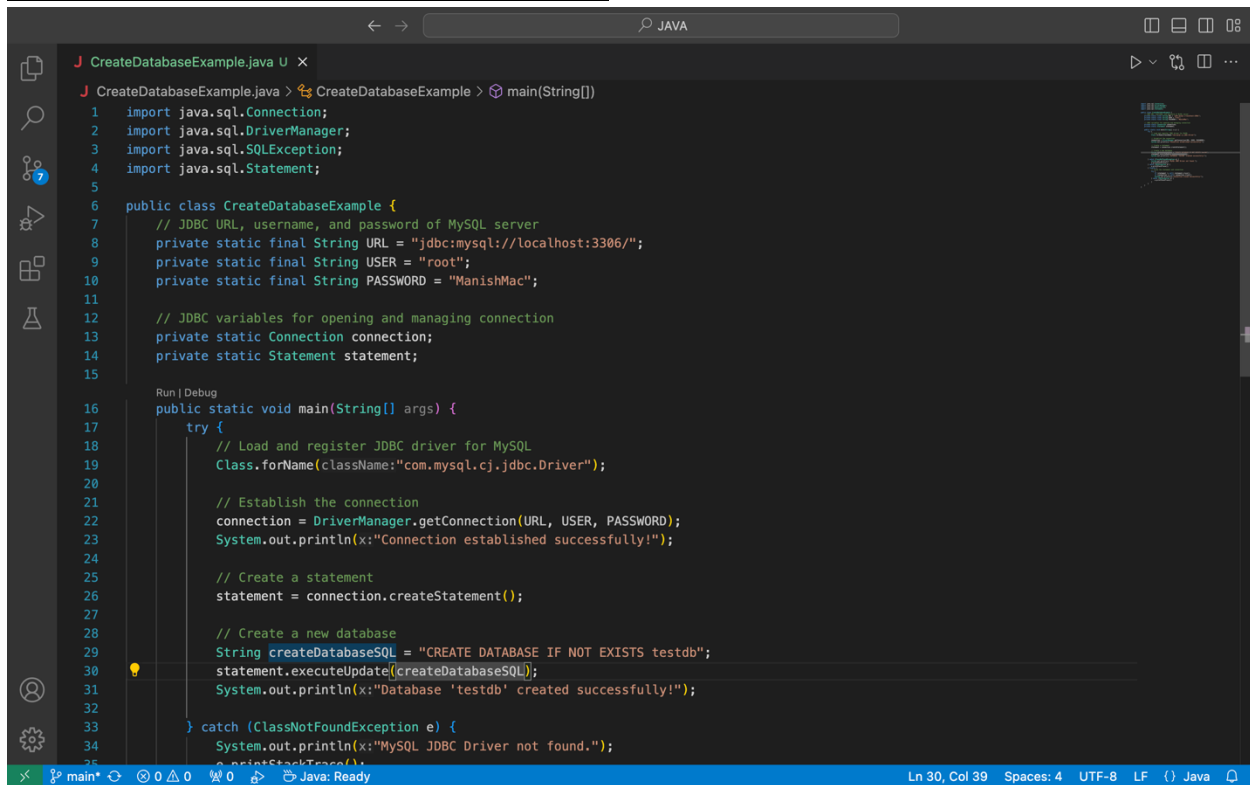
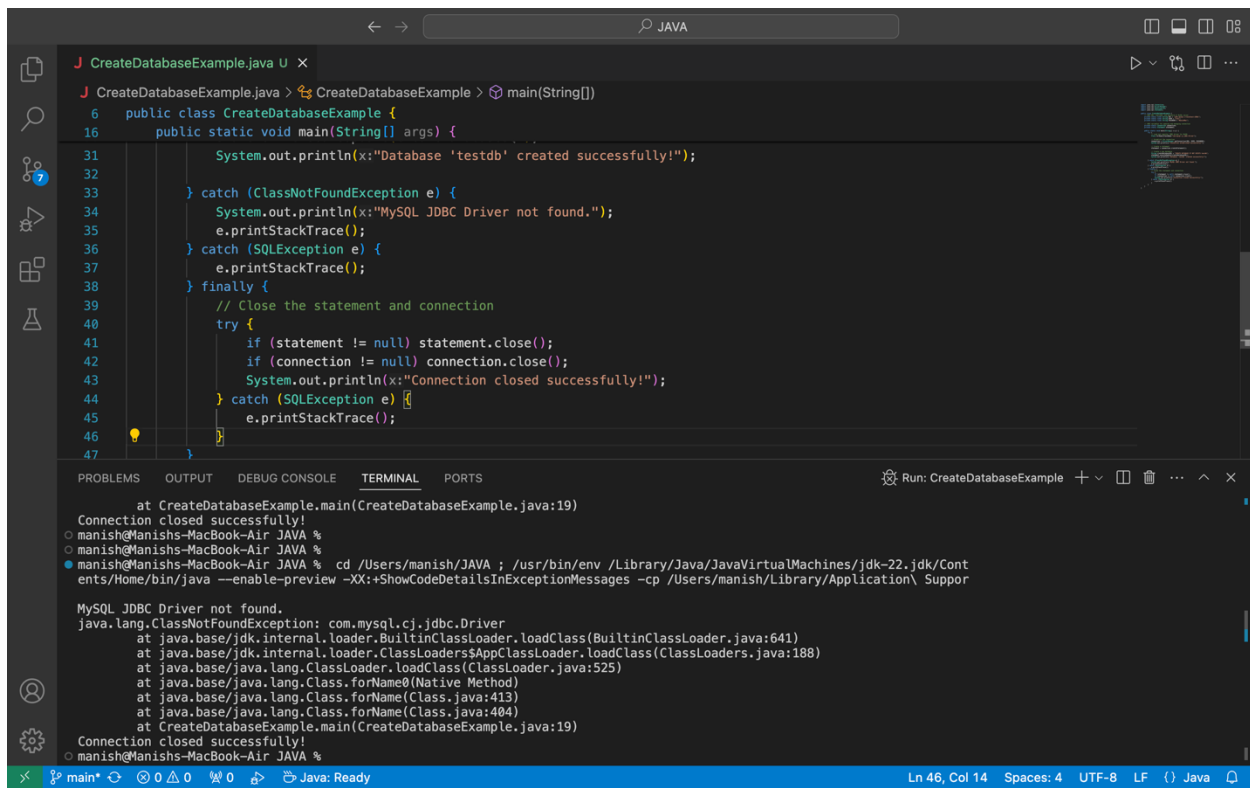


## DATABASE IS CREATED SUCESSFULLY :

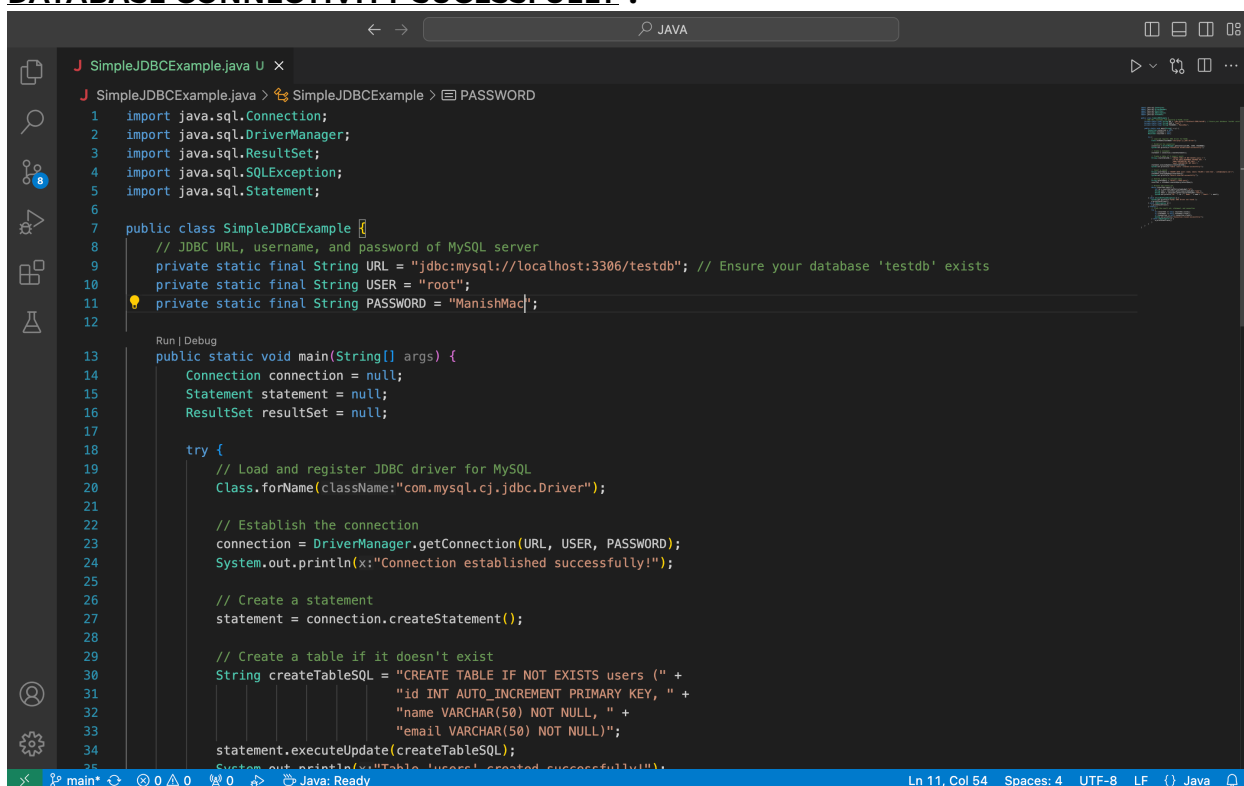


```
J CreateDatabaseExample.java U X
J CreateDatabaseExample.java > CreateDatabaseExample > main(String[])
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.SQLException;
4 import java.sql.Statement;
5
6 public class CreateDatabaseExample {
7     // JDBC URL, username, and password of MySQL server
8     private static final String URL = "jdbc:mysql://localhost:3306/";
9     private static final String USER = "root";
10    private static final String PASSWORD = "ManishMac";
11
12    // JDBC variables for opening and managing connection
13    private static Connection connection;
14    private static Statement statement;
15
16    Run | Debug
17    public static void main(String[] args) {
18        try {
19            // Load and register JDBC driver for MySQL
20            Class.forName(className:"com.mysql.cj.jdbc.Driver");
21
22            // Establish the connection
23            connection = DriverManager.getConnection(URL, USER, PASSWORD);
24            System.out.println(x:"Connection established successfully!");
25
26            // Create a statement
27            statement = connection.createStatement();
28
29            // Create a new database
30            String createDatabaseSQL = "CREATE DATABASE IF NOT EXISTS testdb";
31            statement.executeUpdate(createDatabaseSQL);
32            System.out.println(x:"Database 'testdb' created successfully!");
33        } catch (ClassNotFoundException e) {
34            System.out.println(x:"MySQL JDBC Driver not found.");
35            e.printStackTrace();
36        }
37    }
38
39    Java: Ready
Ln 30, Col 39 Spaces: 4 UTF-8 LF ( ) Java
```



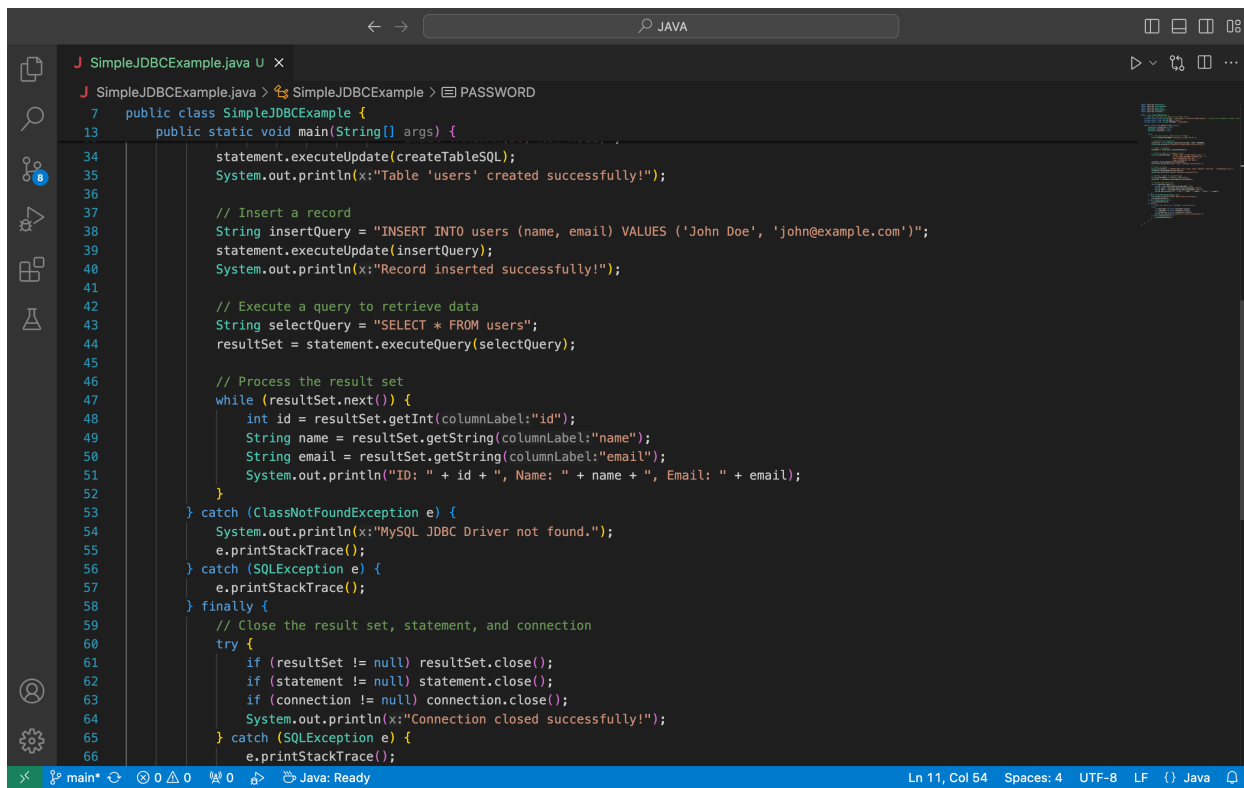
```
J CreateDatabaseExample.java U X
J CreateDatabaseExample.java > CreateDatabaseExample > main(String[])
6 public class CreateDatabaseExample {
16 public static void main(String[] args) {
31     System.out.println(x:"Database 'testdb' created successfully!");
32
33     } catch (ClassNotFoundException e) {
34         System.out.println(x:"MySQL JDBC Driver not found.");
35         e.printStackTrace();
36     } catch (SQLException e) {
37         e.printStackTrace();
38     } finally {
39         // Close the statement and connection
40         try {
41             if (statement != null) statement.close();
42             if (connection != null) connection.close();
43             System.out.println(x:"Connection closed successfully!");
44         } catch (SQLException e) {
45             e.printStackTrace();
46         }
47     }
48 }
49
50 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run: CreateDatabaseExample + - - - - -
at CreateDatabaseExample.main(CreateDatabaseExample.java:19)
Connection closed successfully!
manish@Manishs-MacBook-Air JAVA %
manish@Manishs-MacBook-Air JAVA %
manish@Manishs-MacBook-Air JAVA % cd /Users/manish/JAVA ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-22.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/manish/Library/Application\ Support
MySQL JDBC Driver not found.
java.lang.ClassNotFoundException: com.mysql.cj.jdbc.Driver
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:641)
    at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:188)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:525)
    at java.base/java.lang.Class.forName0(Native Method)
    at java.base/java.lang.Class.forName(Class.java:413)
    at java.base/java.lang.Class.forName(Class.java:404)
    at CreateDatabaseExample.main(CreateDatabaseExample.java:19)
Connection closed successfully!
manish@Manishs-MacBook-Air JAVA %
Ln 46, Col 14 Spaces: 4 UTF-8 LF ( ) Java
```

## DATABASE CONNECTIVITY SUCESSFULLY :



The screenshot shows an IDE window titled 'SimpleJDBCExample.java'. The code defines a class 'SimpleJDBCExample' with a 'main' method. It imports necessary JDBC classes and sets up connection parameters for a MySQL database. The 'main' method initializes connection variables, loads the JDBC driver, establishes a connection, creates a statement, and attempts to create a table named 'users' if it does not exist. The status bar at the bottom indicates 'Ln 11, Col 54', 'Spaces: 4', 'UTF-8', 'LF', and 'Java'.

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.ResultSet;
4 import java.sql.SQLException;
5 import java.sql.Statement;
6
7 public class SimpleJDBCExample {
8     // JDBC URL, username, and password of MySQL server
9     private static final String URL = "jdbc:mysql://localhost:3306/testdb"; // Ensure your database 'testdb' exists
10    private static final String USER = "root";
11    private static final String PASSWORD = "ManishMac";
12
13    public static void main(String[] args) {
14        Connection connection = null;
15        Statement statement = null;
16        ResultSet resultSet = null;
17
18        try {
19            // Load and register JDBC driver for MySQL
20            Class.forName(className="com.mysql.cj.jdbc.Driver");
21
22            // Establish the connection
23            connection = DriverManager.getConnection(URL, USER, PASSWORD);
24            System.out.println(x:"Connection established successfully!");
25
26            // Create a statement
27            statement = connection.createStatement();
28
29            // Create a table if it doesn't exist
30            String createTableSQL = "CREATE TABLE IF NOT EXISTS users (" +
31                                   "id INT AUTO_INCREMENT PRIMARY KEY, " +
32                                   "name VARCHAR(50) NOT NULL, " +
33                                   "email VARCHAR(50) NOT NULL)";
34            statement.executeUpdate(createTableSQL);
35            System.out.println(x:"Table 'users' created successfully!");
```



The screenshot shows the continuation of the 'SimpleJDBCExample' class. It includes code for inserting a record into the 'users' table, querying the data, and processing the result set. It also includes exception handling for 'ClassNotFoundException' and 'SQLException'. The 'main' method concludes by closing the result set, statement, and connection. The status bar at the bottom indicates 'Ln 11, Col 54', 'Spaces: 4', 'UTF-8', 'LF', and 'Java'.

```
36
37    // Insert a record
38    String insertQuery = "INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com')";
39    statement.executeUpdate(insertQuery);
40    System.out.println(x:"Record inserted successfully!");
41
42    // Execute a query to retrieve data
43    String selectQuery = "SELECT * FROM users";
44    resultSet = statement.executeQuery(selectQuery);
45
46    // Process the result set
47    while (resultSet.next()) {
48        int id = resultSet.getInt(columnLabel:"id");
49        String name = resultSet.getString(columnLabel:"name");
50        String email = resultSet.getString(columnLabel:"email");
51        System.out.println("ID: " + id + ", Name: " + name + ", Email: " + email);
52    }
53 } catch (ClassNotFoundException e) {
54     System.out.println(x:"MySQL JDBC Driver not found.");
55     e.printStackTrace();
56 } catch (SQLException e) {
57     e.printStackTrace();
58 } finally {
59     // Close the result set, statement, and connection
60     try {
61         if (resultSet != null) resultSet.close();
62         if (statement != null) statement.close();
63         if (connection != null) connection.close();
64         System.out.println(x:"Connection closed successfully!");
65     } catch (SQLException e) {
66         e.printStackTrace();
```

The screenshot shows an IDE with two panes. The top pane displays the source code of `SimpleJDBCExample.java`. The code is as follows:

```

1 public class SimpleJDBCExample {
2     public static void main(String[] args) {
3         try {
4             System.out.println("MySQL JDBC Driver not found.");
5             e.printStackTrace();
6         } catch (SQLException e) {
7             e.printStackTrace();
8         } finally {
9             // Close the result set, statement, and connection
10            try {
11                if (resultSet != null) resultSet.close();
12                if (statement != null) statement.close();
13                if (connection != null) connection.close();
14                System.out.println("Connection closed successfully!");
15            } catch (SQLException e) {
16                e.printStackTrace();
17            }
18        }
19    }
20 }

```

The bottom pane shows the terminal output. The command to run the program is `java SimpleJDBCExample`. The output is:

```

manish@Manishs-MacBook-Air JAVA % java SimpleJDBCExample
MySQL JDBC Driver not found.
java.lang.ClassNotFoundException: com.mysql.cj.jdbc.Driver
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:641)
    at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:188)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:525)
    at java.base/java.lang.Class.forName0(Native Method)
    at java.base/java.lang.Class.forName(Class.java:413)
    at java.base/java.lang.Class.forName(Class.java:404)
    at SimpleJDBCExample.main(SimpleJDBCExample.java:20)
Connection closed successfully!
manish@Manishs-MacBook-Air JAVA %

```

The error message `java.lang.ClassNotFoundException: com.mysql.cj.jdbc.Driver` is highlighted in red in the original image.

## ECHOSERVER :

The screenshot displays an IDE window titled "J EchoServer.java X". The editor shows the source code for a Java class named `EchoServer`. The code is as follows:

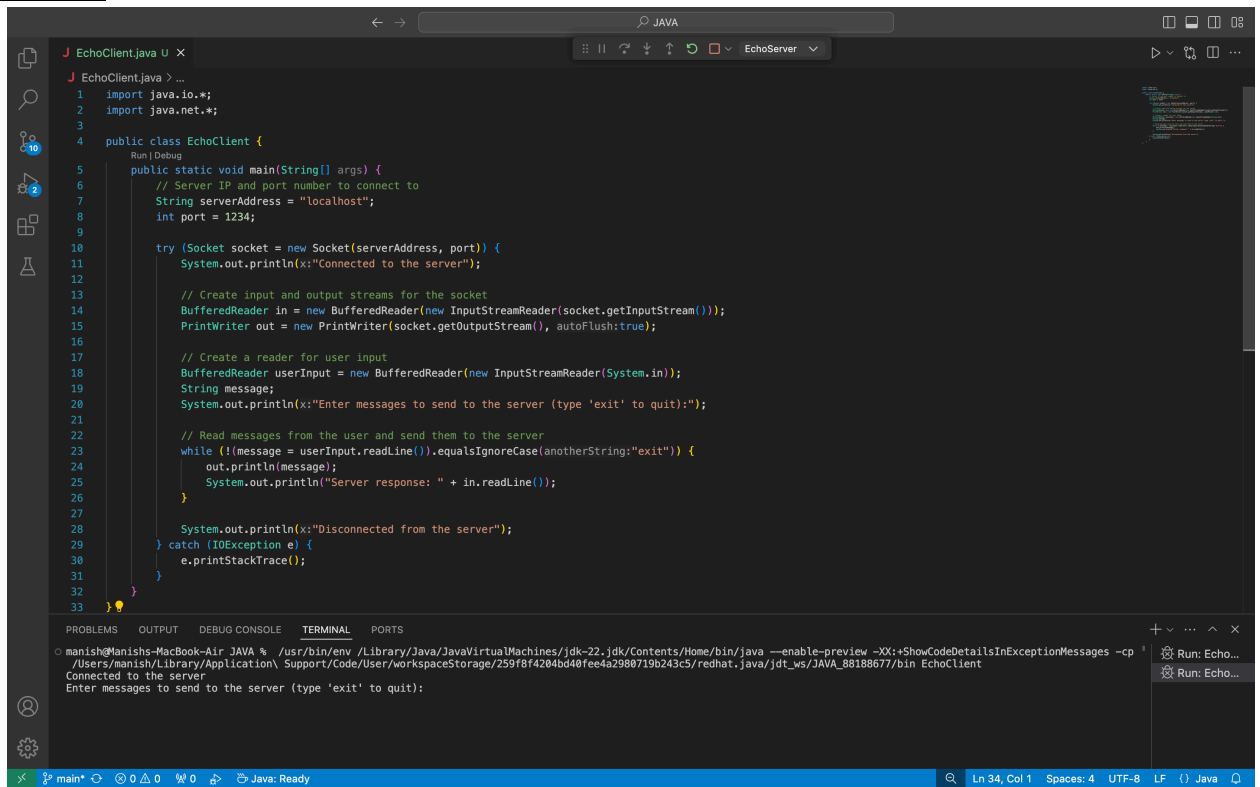
```
1
2 import java.io.*;
3 import java.net.*;
4
5 public class EchoServer {
6     @SuppressWarnings("unused")
7     public static void main(String[] args) {
8         // Port number to listen on
9         int port = 1234;
10
11         try {
12             (ServerSocket serverSocket = new ServerSocket(port)) {
13                 System.out.println("Server started. Listening on port " + port);
14
15                 // Wait for a connection
16                 Socket clientSocket = serverSocket.accept();
17                 System.out.println("Client connected");
18
19                 // Create input and output streams for the client
20                 BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
21                 PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), autoFlush:true);
22
23                 String message;
24                 // Read messages from the client and echo them back
25                 while ((message = in.readLine()) != null) {
26                     System.out.println("Received: " + message);
27                     out.println("Echo: " + message);
28                 }
29
30                 System.out.println("Client disconnected");
31             } catch (IOException e) {
32                 e.printStackTrace();
33             }
34         }
35     }
36 }
```

Below the editor, the "TERMINAL" tab is active, showing the command used to run the application and its output:

```
manish@Manishs-MacBook-Air JAVA % ./usr/bin/env /Library/Java/JavaVirtualMachines/jdk-22_jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/manish/Library/Application Support/Code/User/workspaceStorage/259f8f4284bd40f4e42988719b27435/redhat.java/jdt_ws/JAVA_88188677 /bin EchoServer
Server started. Listening on port 1234
```

The status bar at the bottom indicates the current file is `main.java`, the cursor is at line 34, column 1, and the IDE is ready.

## ECHOCLIENT :



The screenshot displays an IDE window with the file `EchoClient.java` open. The code is a Java program that connects to a server at `localhost:1234` and echoes back any messages received. The terminal output shows the program's execution, including the connection status and the prompt for user input.

```
J EchoClient.java U X
J EchoClient.java > ...
1 import java.io.*;
2 import java.net.*;
3
4 public class EchoClient {
5     public static void main(String[] args) {
6         // Server IP and port number to connect to
7         String serverAddress = "localhost";
8         int port = 1234;
9
10        try (Socket socket = new Socket(serverAddress, port)) {
11            System.out.println(x:"Connected to the server");
12
13            // Create input and output streams for the socket
14            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
15            PrintWriter out = new PrintWriter(socket.getOutputStream(), autoFlush:true);
16
17            // Create a reader for user input
18            BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
19            String message;
20            System.out.println(x:"Enter messages to send to the server (type 'exit' to quit):");
21
22            // Read messages from the user and send them to the server
23            while (!(message = userInput.readLine()).equalsIgnoreCase("exit")) {
24                out.println(message);
25                System.out.println("Server response: " + in.readLine());
26            }
27
28            System.out.println(x:"Disconnected from the server");
29        } catch (IOException e) {
30            e.printStackTrace();
31        }
32    }
33 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

manish@Manishs-MacBook-Air JAVA % /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-22.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/manish/Library/Application\ Support/Code/User/workspaceStorage/259f8f4204bd40fee4a2980719b243c5/redhat.java/jdt\_ws/JAVA\_88188677/bin EchoClient

Connected to the server

Enter messages to send to the server (type 'exit' to quit):

Run: Echo...  
Run: Echo...

main\* 0 0 0 Java: Ready Ln 34, Col 1 Spaces: 4 UTF-8 LF () Java