1. Write a Java program to create a new database in MySQL using JDBC.

Code Solution:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.SQLException;

public class CREATE_DB {

    public static void main(String[] args) {

        // Database URL, username and password
        String url = "jdbc:mysql://localhost:3306/";
        String username = "IshanKRajani"; // Your MySQL username
        String password = "IshanKRajani@1234"; // Your MySQL password

        // Database name to be created
        String databaseName = "SecondDatabase";

        // Connection object
        Connection connection = null;
        Statement statement = null;

        try {
                // Establish a connection to the MySQL server (without
specifying the database)
                connection = DriverManager.getConnection(url, username,
password);

            // Create a statement
            statement = connection.createStatement();

            // SQL query to create a new database
            String createDatabaseSQL = "CREATE DATABASE " + databaseName;

            // Execute the query
            statement.executeUpdate(createDatabaseSQL);
```

```
                System.out.println("Database " + databaseName + " created
successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // Close resources
            try {
                if (statement != null) statement.close();
                if (connection != null) connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Step-by-step explanation of the code:

1. Imports:

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.SQLException;

- `Connection`: This is used to create a connection to the database.
- `DriverManager`: This is a utility class used to establish a connection to the database.
- `Statement`: This is used to execute SQL queries (such as `CREATE DATABASE`).
- `SQLException`: This exception is thrown if there's any issue related to SQL operations (such as database connection errors, syntax issues, etc.).

2. Class Declaration:

public class CREATE_DB {

- This declares a public class named `CREATE_DB`. The code within this class will perform the database creation process.

3. Main Method:

public static void main(String[] args) {

- The `main` method is the entry point of any Java application. When you run the program, it starts executing from here.

4. Define Database URL, Username, and Password:

String url = "jdbc:mysql://localhost:3306/";
String username = "`IshanKRajani`"; // Your MySQL username
String password = "IshanKRajani@1234"; // Your MySQL password

- `url`: This is the JDBC URL, specifying the location of the MySQL server (here, it's running locally on the default MySQL port `3306`).
- `username`: The username to log into the MySQL database (in this case, `"IshanKRajani"`).
- `password`: The password associated with the MySQL username (`"IshanKRajani@1234"`).

5. Database Name:

String databaseName = "SecondDatabase";

- This specifies the name of the database that will be created: `"SecondDatabase"`.

6. Declare Connection and Statement Objects:

Connection connection = null;
Statement statement = null;

- `connection`: This object will hold the connection to the MySQL server.
- `statement`: This object will be used to execute the SQL query.

7. Try Block (Establishing Connection):

try {
  connection = DriverManager.getConnection(url, username, password);
  statement = connection.createStatement();

- Inside the `try` block:

- `connection = DriverManager.getConnection(url, username, password);`: This establishes a connection to the MySQL server using the provided URL, username, and password.
- `statement = connection.createStatement();`: This creates a `Statement` object that can be used to execute SQL queries against the database.

8. SQL Query to Create Database:

String createDatabaseSQL = "CREATE DATABASE " + databaseName;

- This constructs the SQL query to create a new database. The SQL query is `"CREATE DATABASE SecondDatabase"`, where `"SecondDatabase"` is the value of the `databaseName` variable.

9. Execute the SQL Query:

statement.executeUpdate(createDatabaseSQL);

- This line executes the SQL query using the `executeUpdate()` method of the `Statement` object. The `executeUpdate()` method is used for SQL statements that modify the database (such as `CREATE`, `INSERT`, `UPDATE`, `DELETE`).
- In this case, it will create the `SecondDatabase` database.

10. Success Message:

System.out.println("Database " + databaseName + " created successfully.");

- After successfully executing the SQL query, this line prints a message to the console, confirming that the database has been created successfully.

11. Catch Block (Handle Exceptions):

```
} catch (SQLException e) {
    e.printStackTrace();
}
```

- If there's any error while establishing the connection or executing the SQL query, it will be caught in this `catch` block.
- `e.printStackTrace()` prints the details of the exception (error) to the console for debugging purposes.

12. Finally Block (Close Resources):

```
finally {
    try {
        if (statement != null) statement.close();
        if (connection != null) connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

- The `finally` block ensures that resources (`statement` and `connection`) are always closed, even if an exception occurs. Closing the resources is important to avoid potential memory leaks or connection issues.
- `statement.close()` and `connection.close()` close the `Statement` and `Connection` objects, respectively.
- If there's an issue while closing these resources, the exception is caught and printed.

Summary:

- The program establishes a connection to a MySQL database server.
- It then creates a new database called `SecondDatabase` by executing a `CREATE DATABASE` SQL query.
- It handles any exceptions that might occur and ensures that database resources are properly closed afterward.