1) What is JDBC?

Answer:
JDBC stands for "Java Database Connectivity." It is a Java API that allows Java applications to interact with databases. JDBC enables Java programs to execute SQL statements, retrieve results from the database, and manipulate data.

Here are some key features and components of JDBC:

API: JDBC provides a set of classes and interfaces in the **java.sql** and **javax.sql** packages for database interaction.

Driver Manager: The **java.sql.DriverManager** class manages a list of database drivers. It helps in establishing a connection with a database by selecting an appropriate driver from the list.

Connection: The **java.sql.Connection** interface represents a connection to a specific database. It allows you to create and execute statements, manage transactions, and handle database metadata.

Statement: The **java.sql.Statement** interface is used to execute SQL statements in the database. There are different types of statements like **Statement**, **PreparedStatement**, and **CallableStatement** for different purposes.

ResultSet: The **java.sql.ResultSet** interface represents the result set of a SQL query. It allows you to traverse and access the data retrieved from the database.

SQL Exceptions: JDBC methods may throw **java.sql.SQLException** which must be handled appropriately in Java applications.

JDBC provides database independence, meaning that you can write database code once and run it on different databases with minimal changes, as long as you use standard SQL queries and JDBC API methods.

Overall, JDBC is a fundamental technology for Java developers working with databases, enabling them to build robust and scalable database-driven applications.

2) How to connect your Java program to a database?

Answer:
To connect a Java program to a database using JDBC, you typically follow these steps:

Load the JDBC Driver: Load the JDBC driver class corresponding to the database you want to connect to. This is done using the **Class.forName()** method.

Establish a Connection: Use the **DriverManager.getConnection()** method to establish a connection to the database by providing the database URL, username, and password.

Create a Statement: Create a **Statement**, **PreparedStatement**, or **CallableStatement** object to execute SQL queries or commands.

Execute SQL Queries: Use the created statement object to execute SQL queries or commands against the database.

Process Results: If your query returns a result set, use the **ResultSet** object to process the retrieved data.

Close Resources: Close the **Connection**, **Statement**, and **ResultSet** objects when you're done with them to release database resources and avoid memory leaks.

Here's a basic example demonstrating how to connect a Java program to a MySQL database using