M.Kowsthabi

20A21A1240

Swarnandhra College of Engineering and Technology.

# CRUD Application using JDBC Connectivity:

Java Database Connectivity(JDBC) is a java based API(Application Programming Interface) that provides a standard interface for connecting java applications with relational database.

STEPS IN JDBC CONNECTIVITY:

1)Load JDBC Driver

2)create a Database URL

3)Provide Database credentials

4)Establish Connection

5)Create a Statement or PreparedStatement

6)Execute SQL Queries

7)Process Results

8)close Resource .

1)Load JDBC Driver: Load the JDBC driver class using Class.forName(). Class.forName("com.mysql.cj.jdbc.Driver");

2)Create Database URL: Define the JDBC URL with information about the database location, port name. Url="jdbc:mysql://localhost:3306/your\_database";

3)Provide Database Credentials: Specify the username and password for database access. ¬String user = "your\_username"; ¬String password = "your\_password";

4)Establish Connection: Use DriverManager.getConnection() to connect to the database ¬Connection connection = DriverManager.getConnection(url, user, password);

5)Create Statement or PreparedStatement: Create a Statement or PreparedStatement for executing SQL queries.

Statement statement = connection.createStatement();

(or)

PreparedStatement preparedStatement = connection.

prepareStatement

("SELECT \* FROM your\_table WHERE column = ?")

6)Execute SQL Queries: Use executeQuery() for SELECT queries and executeUpdate() for INSERT, UPDATE, DELETE queries.

ResultSet resultSet = statement.executeQuery

("SELECT \* FROM your\_table");

7)Process Results: If executing a SELECT query, process the results using the ResultSet object.

¬while (resultSet.next())

{ // Process each row

}

8)Close Resources: Close the ResultSet, Statement, and Connection objects to free up resources. resultSet.close();

statement.close();

connection.close();

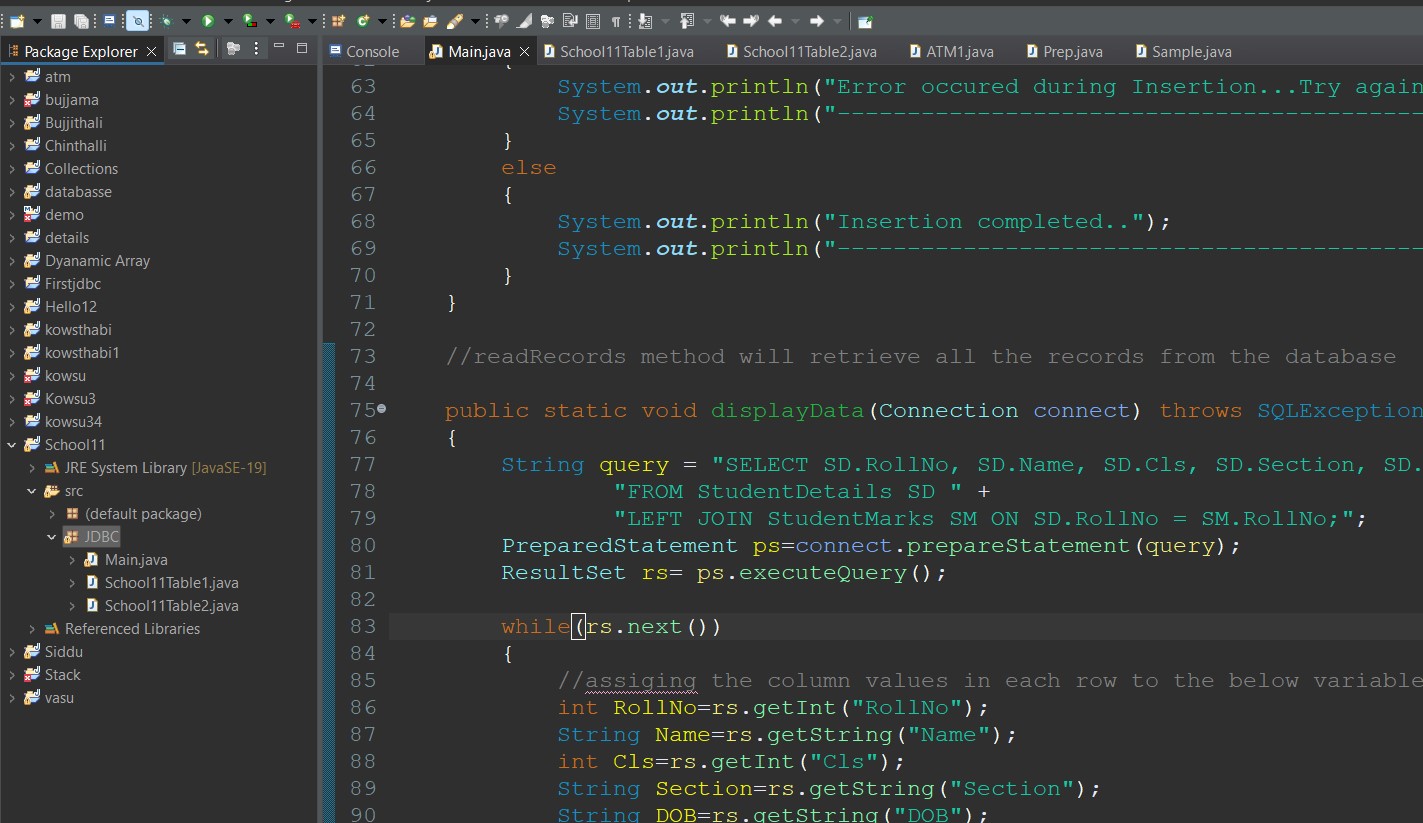
# OPEN ECLIPSE IDE :

I created a new project with name – School11 now import the mysql-connector jar file to the project. this file is essential in establishing the connection between java application and MySQL.

The MySQL Connector/J is JDBC-compliant, which means it adheres to the standards defined by the JDBC API.

JDBC is a Java-based API that enables Java applications to interact with relational databases.

MySQL Connector/J provides the implementation of the JDBC API for MySQL databases.



# OPEN MYSQL :

I created a database called school9. And verified whether it is created or not using the show databases command. And it is created successfully.

Now ,we need to perform CRUD(create ,read ,update and delete) operationsIn Eclipse, I created a package in the School9 Project inside that package create 2 different classes to create two tables one table to store student details and the other is for student marks .

class name for table1 is School11Table1 ¬

class name for table2 is School11Table2.

School11Table1:

the columns like Reg number , Name ,Class , Section , Date-of-birth.

# The School11Table1 class consists of five methods:

1.Creates the "student\_details" table in the "school9" database.

2)addStudent(): Inserts a new student record into the "student\_details" table.

3)displayStudents(): ¬Retrieves and displays all student records from the "student\_details" table.

4)updateStudent(): Updates the details of an existing student record based on the provided redno (ID).

5)deleteStudent(): Deletes a student record from the "student\_details" table based on the provided redno (ID)

These methods collectively provide functionality for creating, retrieving, updating, and deleting student records in the MySQL database.

School11Table2:

¬ contains the columns like Reg number , English ,Telugu, Tamil , Marks. ¬

The School11Table2 class actually consists of five methods.

Here is the breakdown:

1)createMarksTable(): Creates the "student\_marks" table in the "school9" database

2)addMarks() : ¬Inserts a new row into the "student\_marks" table with the provided marks for English, Telugu, Tamil, and overall marks.

3)displayMarks(): Retrieves and displays all rows from the "student\_marks" table, printing the redno, English marks, Telugu marks, Tamil marks, and overall marks for each student

4)updateMarks(): Updates an existing row in the "student\_marks" table based on the provided redno (ID), modifying the marks for English, Telugu, Tamil, and overall marks for the specified student.

5)deleteMarks(): Deletes a row from the "student\_marks" table based on the provided redno (ID), effectively removing the corresponding student's marks from the database.

School11Main.java :

The School11Main class is a Java program that serves as the main entry point for your application. It interacts with two other classes, School9Table1 and School9Table2, which are responsible for handling database operations related to student details and marks.

School11Main class consists of following methods:

1)main(String[] args): The main entry point of the program that establishes a connection to the MySQL database and calls the handleTableChoice method.

2)handleTableChoice() : Handles the user's choice between student details, student marks, or exiting the program. Calls methods from School9Table1 or School9Table2 based on the user's selection.

3)handleStudentDetails(): Provides a submenu for performing operations related to student details. Calls methods from the School9Table1 class based on user input.

4)handleStudentMarks(): Provides a submenu for performing operations related to student marks. Calls methods from the School9Table2 class based on user input

OUTPUT:

Whenever we run the application the following output will come ¬Performing CRUD operations for School9Table1(Student Details)

CHOICE 1:INSERT:

CHOICE 2:READ:

CHOICE 3:UPDATE:

CHOICE 4:DELETE :

CHOICE 5:EXIT:

CRUD operations for School11Table1(Student Details)

CHOICE 1: CREATE

CHOICE 2 : READ

CHOICE 3:UPDATE

CHOICE 4: DELETE