What is JDBC?

- **JDBC** (Java database connectivity) is an **API** for the Java programming language that defines how a client may access a database.

- It provides methods for *querying* and *updating* data in a database.

- JDBC is oriented towards *relational* databases.

- The technology is simple, robust and well-tested (introduced in 1977).
JDBC Architecture Components

FIGURE 5-1  Relationships between major classes and interface in the java.sql package
What is JDBC? - Components

- **JDBC connections** support creating and executing statements.

- **Statements** may be *action statements* such as: SQL CREATE, INSERT, UPDATE and DELETE or they may be *query statements* using the SELECT statement.

- **Stored procedures** may also be invoked through a statement.

- **Statements** are one of the following types:
  - **Statement** – the statement is sent to the database server each and every time.
  - **PreparedStatement** – the statement is cached and then the execution path is predetermined on the database server allowing it to be executed multiple times in an efficient manner.
  - **CallableStatement** – used for executing stored procedures on the database.
What is JDBC? - Components

- **Action statements** such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

- **Query statements** return a JDBC row ResultSet.
  - Rows are used to traverse the result set.
  - Individual columns in a row are retrieved either by name or by column number.
  - There may be any number of rows in the result set.
  - The row result set has metadata that describes the names of the columns and their types.
What is JDBC? - Components

- A ResultSet is a Java object that contains the rows resulting from executing a SQL query.
- The data stored in a ResultSet object is retrieved with various `getXXX` methods (such as `getInt(...)`, `getString(...)` and so on) that allows access to the columns of the current row.
- The `.next` method is used to move to the next row of the ResultSet.
- The Statement methods `executeQuery` and `getResultSet` both return a ResultSet object.

Example

```java
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from employee");
```
A Brief Example – How do JDBC apps look like?

A typical JDBC based application makes possible to do three things:

1. Establish a connection with a data source
2. Send queries and update statements to the data source
3. Process the results
A Brief Example – How JDBC apps look like

```java
private void TypicalCode() throws ClassNotFoundException, SQLException {
    Class.forName("com.mysql.jdbc.Driver");
    String URL = "jdbc:mysql://localhost:3306/world";
    Connection con = DriverManager.getConnection(
        URL, "myUserName", "myPassword");

    Statement stmt = con.createStatement();
    String mySQL = "select fname, salary from employee";
    ResultSet rs = stmt.executeQuery(mySQL);

    while (rs.next()) {
        String fName = rs.getString("fname");
        float salary = rs.getFloat("salary");
        //... Do some work with the data
    }
    rs.close();
    con.close();
}
```
Download JDBC Driver

- MySQL
  - http://dev.mysql.com/downloads/connector/j/5.0.html
- Oracle
- MS SQL Server
**Example: Using the Company Database**

**Example.** Connecting to the Company Database

We assume you have installed Eclipse, Connector-j, and MySQL on which you have defined and populated the Company database shown below
1. Write a Java application, call it MySQLDemo1MakeConnection.
2. Right-click on JRE System Library icon.
3. Select options: Build Path > Configure Built Path > Add External Jars
4. Locate the folder where your Connector J driver was installed (in our example c:\Program Files (x86)\MySQL\Connector J 5.1.20.0
5. Click on the jar to be added to the Java app (see image below)
import java.sql.Connection;

public class MySQLDemo1MakeConnection {

    public static void main(String[] args) throws ClassNotFoundException, SQLException {

        Class.forName("com.mysql.jdbc.Driver");

        Connection cnn = DriverManager.getConnection("jdbc:mysql://localhost:3306/company", "csuperson", "euclid");

        if (cnn == null) {
            System.out.println("Problems: No connection...");
            System.exit(0);
        }

        System.out.println("Success: connection is available...");

        useStatement(cnn);
        executeActionQueryInsert(cnn);
        executeActionQueryDelete(cnn);
        usePreparedStatement(cnn);

        cnn.close();
    }
}
private static void useStatement(Connection cnn) throws SQLException {

    Statement stm = cnn.createStatement();

    String mySQLQuery = "select fname, lname, salary from employee";

    ResultSet rs = stm.executeQuery(mySQLQuery);

    System.out.println("Simple Statement - Retrieving Employees using\n");

    while (rs.next()) {
        System.out.println(rs.getFloat("salary") + " " + rs.getString("fname") + " " + rs.getString(2) );
    }

    rs.close();
}
//useStatement

Console
Simple Statement - Retrieving Employees using

30000.0 John Smith
40000.0 Franklin Wong
25000.0 Joyce English
38000.0 Ramesh Narayan
55000.0 James Borg
43000.0 Jennifer Wallace
25000.0 Ahmad Jabbar
25000.0 Alicia Zelaya
private static void executeActionQueryInsert(Connection cnn) {
    String mySqlString = "insert into works_on values (123456789, 99, 40)";
    try {
        Statement stmt = cnn.createStatement();
        stmt.executeUpdate(mySqlString);
        System.out.println("\nTotal rec. inserted " + stmt.getUpdateCount());
    } catch (Exception e) {
        System.out.println(" BIG MISTAKE>>> " + e.getMessage());
    }
}

private static void executeActionQueryDelete(Connection cnn) {
    String mySqlString = "delete from works_on where essn=123456789 and pno=99";
    try {
        Statement stmt = cnn.createStatement();
        stmt.executeUpdate(mySqlString);
        System.out.println("\nTotal rec. deleted " + stmt.getUpdateCount());
    } catch (Exception e) {
        System.out.println(" BIG MISTAKE>>> " + e.getMessage());
    }
}
private static void usePreparedStatement(Connection cnn) throws SQLException {
    Statement stmt = cnn.createStatement();
    String mySQL = "select * from employee where dno = ? ";
    PreparedStatement stm = cnn.prepareStatement(mySQL, 
                                            ResultSet.FETCH_FORWARD);
    stm.setString(1, "4");
    ResultSet rs = stm.executeQuery();
    System.out.println("\nPreparedStatement - Employees working for Dno 4\n");
    while (rs.next()) {
        System.out
          .println(rs.getString("ssn") + " "
          + rs.getString("dno") + " "
          + rs.getString("Fname") + " "
          + rs.getString("Lname");
    }
    rs.close();
}//usePreparedStatement

PreparedStatement - Employees working for Dno 4
987654321 4 Jennifer Wallace
987987987 4 Ahmad Jabbar
999887777 4 Alicia Zelaya