**JDBC Interfaces**

* The four key JDBC API Interfaces are, **Driver**, **Connection**, **Statement**, and **ResultSet**.
* The **Driver** interface establishes a database connection.
* The **Connection** interface communicates with the database.
* The **Statement** interface provides methods for executing SQL statements.
* The **ResultSet** interface holds data returned from database tables.

**The Statement Interface / Object**

* A **Statement** object is created using the **createStatement()** method.
* The **createStatement()** method must be called by a **Connection** reference.
* Once you have a **Statement** reference, you can call methods for executing SQL statements such as:
  + **execute()**
  + **executeUpdate()**
  + **executeQuery()**

**The execute() Method**

* The **execute()** methods runs a SQL statement (accepted as a **String** argument) and returns a **Boolean**.
* If the **execute()** method returns **true**, a SQL **SELECT** statement was executed
* If the **execute()** method returns **false**, a SQL **INSERT**, **UPDATE**, or **DELETE** statement was executed.

**The getUpdateCount() Method**

* The **getUpdateCount()** method returns the number of rows affected by an **INSERT**, **UPDATE**, or **DELETE** command.
* The **getUpdateCount()** method should be called by a Statement reference if the **execute()** method returns **false**.

**The ResultSet Object**

* The **ResultSet** object contains a collection of objects that map records from a database table or multiple tables.
* **The ResultSet** object provides members for scrolling through the it such as **next(),** **previous(),** **first(),** **last(),** **beforeFirst(),** **afterLast()**, etc.
* The **next()** method moves the ResultSet Cursor forward or down as it returns **true** if the cursor points to a valid row of data, and **false** if it doesn’t.

**The getResultSet() Method**

* The **getResultSet()** method returns a **ResultSet** Object.
* **getResultSet()** should be called by a **Statement** reference if the **execute()** method returns **true**.

**ResultSet get Methods**

* The **ResultSet** has methods for retrieving data it holds
* The following table lists **ResultSet** **get()** Methods, their Java Return type, and the SQL type equivalent:

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Java Return Type** | **SQL Type** |
| getString() | String | CHAR, VARCHAR |
| getDouble() | Double | DOUBLE |
| getInt() | Int | INTEGER |
| getBoolean() | Boolean | BOOLEAN |
|  |  |  |
| getLong() | Long | BIGINT |
| getObject() | Object | Any type |
|  |  |  |
| getDate() | java.sql.Date | DATE |
| getTime() | java.sql.Time | TIME |
| getTimeStamp() | java.sql.TimeStamp | TIMESTAMP |

**What is an Exception?**

* An Exception is an Error that crashes your program.
* An Exception can be thrown using a **throws** clause, or caught using a **try / catch block**.
* Some programming statements require an Exception to be thrown or caught.
* Throwing an Exception does not prevent your program from crashing but catching an Exception does.

**SQL Exceptions**

* An **SQLException** occurs if a SQL statement is written incorrectly.

**SQL Injection Attacks**

* A SQL Injection Attack occurs when SWL statements entered through an application to expose or harm a database.
* Companies that have been affected by SQL injection attacks are Sony, TJX, MasterCard, PBS, etc.
* SQL Injection Attacks can occur by incorrectly handling escape characters such as quotes, assigning conditions as input, etc.

**Cons of the Statement Object**

* The **Statement** Object does not escape quotes.
* Writing **INSERT** statements requires **concatenation** making it tedious to put one together.

**The Prepared Statement Object**

* A **PreparedStatement** is a subclass of the **Statement** class with processes for handling vulnerabilities or inconveniences when working with a SQL database.
* **PreparedStatement** can accept raw SQL statements in Java without the need for concatenation, escape characters, etc.

**Creating A PreparedStatement**

* A **PreparedStatement** object is created using the **prepareStatement()** method.
* The **prepareStatement()** method accepts a **String** argument representing the SQL statement to be executed.
* Like the **createStatement()** method, the **prepareStatement()** method must be called by a **Connection** reference.
* Once you have a **PreparedStatement** reference, you can call methods for mapping data to **?’s Place Holder**s in your SQL statement before executing it using the **execute()** method.