Name:-Bhatt Kinjal Hasmukhbhai (12008109)

Name:-Gadhavi Devi Vinodbhai (12008117)

**Assignment Topic:-javax.sql package**

What is javax.sql package:-

* The javax.sql package contains what was originally the JDBC 2.0 optional packages; these are now a part of the J2SE 1.4.
* The classes and interfaces in this package provide new functionality, such as connection pooling, that does not fall under the scope of the original JDBC API and can therefore be safely packaged separately.
* The DataSource interface serves as a factory for Connection objects;
* DataSource objects can be registered with a JNDI server, making it possible to get the name of a database from a name service.
* PooledConnection supports connection pooling, which allows an application to handle multiple database connections in a fairly transparent manner.
* RowSet extends the ResultSet interface into a JavaBeans component that can be manipulated at design-time and used with non-SQL data sources.
* Interfaces of javax.sql package

|  |  |
| --- | --- |
| CommonDataSource | **Interface that defines the methods which are common between DataSource, XADataSource and ConnectionPoolDataSource.** |
| ConnectionEventListener | **An object that registers to be notified of events generated by a PooledConnection object.** |
| ConnectionPoolDataSource | |  | | --- | | **A factory for PooledConnection objects.** | |  | |
| DataSource | |  | | --- | | **A factory for connections to the physical data source that this DataSource object represents.** | |  | |
| PooledConnection | **An object that provides hooks for connection pool management.** |
| RowSet | **The interface that adds support to the JDBC API for the JavaBeans component model.** |
| RowSetInternal | **The interface that a RowSet object implements in order to present itself to a RowSetReader or RowSetWriter object**. |
| RowSetListener | |  | | --- | | **An interface that must be implemented by a component that wants to be notified when a significant event happens in the life of a RowSet object.** | |  | |
| RowSetMetaData | **An object that contains information about the columns in a RowSet object.** |
| RowSetReader | |  | | --- | | **The facility that a disconnected RowSet object calls on to populate itself with rows of data.** | |  | |
| RowSetWriter | |  | | --- | | **An object that implements the RowSetWriter interface, called a *writer*.** | | [**St**](https://docs.oracle.com/javase/7/docs/api/javax/sql/StatementEventListener.html) | |
| StatementEventListener | **An object that registers to be notified of events that occur on PreparedStatements that are in the Statement pool.** |
| XAConnection | **An object that provides support for distributed transactions.** |
| XADataSource | **A factory for XAConnection objects that is used internally.** |

* Classes

|  |  |
| --- | --- |
| [**ConnectionEvent**](https://docs.oracle.com/javase/7/docs/api/javax/sql/ConnectionEvent.html) | **An  Event  object that provides information about the source of a connection-related event.** |
| [**RowSetEvent**](https://docs.oracle.com/javase/7/docs/api/javax/sql/RowSetEvent.html) | **An  Event object generated when an event occurs to a RowSet object.** |
| [**StatementEvent**](https://docs.oracle.com/javase/7/docs/api/javax/sql/StatementEvent.html) | **A StatementEvent is sent to all StatementEventListeners which were registered with a PooledConnection.** |

Difference between java.sql and javax.sql package::-

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Java.sql** | **Javax.sql** |
| 1. | If you want DataSource related information we can use javax.sql package. | If we want normal connection with DB we can use java.sql package. |
| 2. | java.sql.\* package is used for the basic jdbc connections. | javax.sql.\* packageis used for the advance jdbc. |
| 3. | Provides the API for accessing and processing data stored in a data source (usually a relational database) using the JavaTM programming language. | Provides the API for server side data source access and processing from the JavaTM programming language. |

**Interfaces::--**

**CommonDataSource**

public interface **CommonDataSource**

Interface that defines the methods which are common between DataSource, XADataSource and ConnectionPoolDataSource.

**Method and Description:-**

Int [**getLoginTimeout**](https://docs.oracle.com/javase/8/docs/api/javax/sql/CommonDataSource.html#getLoginTimeout--)():-Gets the maximum time in seconds that this data source can wait while attempting to connect to a database.

[**PrintWrite r**](https://docs.oracle.com/javase/8/docs/api/java/io/PrintWriter.html)[**getLogWriter**](https://docs.oracle.com/javase/8/docs/api/javax/sql/CommonDataSource.html#getLogWriter--)():-Retrieves the log writer for this DataSource object.

[**Logger**](https://docs.oracle.com/javase/8/docs/api/java/util/logging/Logger.html) [**getParentLogger**](https://docs.oracle.com/javase/8/docs/api/javax/sql/CommonDataSource.html#getParentLogger--)():-Return the parent Logger of all the Loggers used by this data source.

Void [**setLoginTimeout**](https://docs.oracle.com/javase/8/docs/api/javax/sql/CommonDataSource.html#setLoginTimeout-int-)(int seconds):-Sets the maximum time in seconds that this data source will wait while attempting to connect to a database.

Void [**setLogWriter**](https://docs.oracle.com/javase/8/docs/api/javax/sql/CommonDataSource.html#setLogWriter-java.io.PrintWriter-)(**[PrintWriter](https://docs.oracle.com/javase/8/docs/api/java/io/PrintWriter.html" \o "class in java.io)** out):-Sets the log writer for this DataSource object to the given java.io.PrintWriter object.

## ConnectionEventListener

public interface **ConnectionEventListener** extends [EventListener](https://docs.oracle.com/javase/8/docs/api/java/util/EventListener.html)

An object that registers to be notified of events generated by a PooledConnection object.

The ConnectionEventListener interface is implemented by a connection pooling component.

A connection pooling component will usually be provided by a JDBC driver vendor or another system software vendor.

A JDBC driver notifies a ConnectionEventListener object when an application is finished using a pooled connection with which the listener has registered.

The notification occurs after the application calls the method close on its representation of a PooledConnection object.

A ConnectionEventListener is also notified when a connection error occurs due to the fact that the PooledConnection is unfit for future use---the server has crashed, for example.

The listener is notified by the JDBC driver just before the driver throws an SQLException to the application using the PooledConnection object.

**Method and Description:-**

|  |  |
| --- | --- |
| Void [**connectionClosed**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionEventListener.html#connectionClosed-javax.sql.ConnectionEvent-)([**ConnectionEvent**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionEvent.html) event) | Notifies this ConnectionEventListener that the application has called the method close on its representation of a pooled connection. |
| Void [**connectionErrorOccurred**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionEventListener.html#connectionErrorOccurred-javax.sql.ConnectionEvent-)([**ConnectionEvent**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionEvent.html) event) | Notifies this ConnectionEventListener that a fatal error has occurred and the pooled connection can no longer be used. |

## ConnectionPoolDataSource

public interface **ConnectionPoolDataSource**

extends [CommonDataSource](https://docs.oracle.com/javase/8/docs/api/javax/sql/CommonDataSource.html)

A factory for PooledConnection objects.

An object that implements this interface will typically be registered with a naming service that is based on the Java™ Naming and Directory Interface (JNDI).

**Method and Description:-**

|  |  |
| --- | --- |
| [**PooledConnection**](https://docs.oracle.com/javase/8/docs/api/javax/sql/PooledConnection.html) | [**getPooledConnection**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionPoolDataSource.html#getPooledConnection--)()  Attempts to establish a physical database connection that can be used as a pooled connection. |
| [**PooledConnection**](https://docs.oracle.com/javase/8/docs/api/javax/sql/PooledConnection.html) | [**getPooledConnection**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionPoolDataSource.html#getPooledConnection-java.lang.String-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) user, [**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) password)  Attempts to establish a physical database connection that can be used as a pooled connection. |

## RowSet

public interface **RowSet** extends [ResultSet](https://docs.oracle.com/javase/8/docs/api/java/sql/ResultSet.html)

The interface that adds support to the JDBC API for the JavaBeans™ component model. A rowset, which can be used as a JavaBeans component in a visual Bean development environment, can be created and configured at design time and executed at run time.

The RowSet interface provides a set of JavaBeans properties that allow a RowSet instance to be nfigured to connect to a JDBC data source and read some data from the data source. A group of setter methods (setInt, setBytes, setString, and so on) provide a way to pass input parameters to a rowset's command property. This command is the SQL query the rowset uses when it gets its data from a relational database, which is generally the case.

The RowSet interface supports JavaBeans events, allowing other components in an application to be notified when an event occurs on a rowset, such as a change in its value.

The RowSet interface is unique in that it is intended to be implemented using the rest of the JDBC API. In other words, a RowSet implementation is a layer of software that executes "on top" of a JDBC driver. Implementations of the RowSet interface can be provided by anyone, including JDBC driver vendors who want to provide a RowSet implementation as part of their JDBC products.

A RowSet object may make a connection with a data source and maintain that connection throughout its life cycle, in which case it is called a *connected* rowset. A rowset may also make a connection with a data source, get data from it, and then close the connection. Such a rowset is called a *disconnected* rowset. A disconnected rowset may make changes to its data while it is disconnected and then send the changes back to the original source of the data, but it must reestablish a connection to do so.

A disconnected rowset may have a reader (a RowSetReader object) and a writer (a RowSetWriter object) associated with it.

The reader may be implemented in many different ways to populate a rowset with data, including getting data from a non-relational data source.

The writer can also be implemented in many different ways to propagate changes made to the rowset's data back to the underlying data source.

Advantages of Rowset:-

Rowsets are easy to use.

The RowSet interface extends the standard java.sql.ResultSet interface.

The RowSetMetaData interface extends the java.sql.ResultSetMetaData interface.

Thus, developers familiar with the JDBC API will have to learn a minimal number of new APIs to use rowsets.

In addition, third-party software tools that work with JDBC ResultSet objects will also easily be made to work with rowsets.

## RowSetWriter

public interface **RowSetWriter**

An object that implements the RowSetWriter interface, called a *writer*. A writer may be registered with a RowSet object that supports the reader/writer paradigm.

If a disconnected RowSet object modifies some of its data, and it has a writer associated with it, it my be implemented so that it calls on the writer's writeData method internally to write theupdates back to the data source. In order to do this, the writer must first establish a connection with the rowset's data source.

If the data to be updated has already been changed in the data source, there is a conflict, in which case the writer will not write the changes to the data source. The algorithm the writer uses for preventing or limiting conflicts depends entirely on its implementation.

Method and Description:

|  |  |
| --- | --- |
| boolean | [**writeData**](https://docs.oracle.com/javase/8/docs/api/javax/sql/RowSetWriter.html#writeData-javax.sql.RowSetInternal-)([**RowSetInternal**](https://docs.oracle.com/javase/8/docs/api/javax/sql/RowSetInternal.html) caller)  Writes the changes in this RowSetWriter object's rowset back to the data source from which it got its data. |

## RowSetReader

public interface **RowSetReader**

The facility that a disconnected RowSet object calls on to populate itself with rows of data.

A reader (an object implementing the RowSetReader interface) may be registered with a RowSet object that supports the reader/writer paradigm.

When the RowSet object's execute method is called, it in turn calls the reader's readData method.

**Method and Description:-**

|  |  |
| --- | --- |
| void | [**readData**](https://docs.oracle.com/javase/8/docs/api/javax/sql/RowSetReader.html#readData-javax.sql.RowSetInternal-)([**RowSetInternal**](https://docs.oracle.com/javase/8/docs/api/javax/sql/RowSetInternal.html) caller)  Reads the new contents of the calling RowSet object. |

**Classes::-**

## Class ConnectionEvent

public class **ConnectionEvent** extends [EventObject](https://docs.oracle.com/javase/8/docs/api/java/util/EventObject.html)

An Event object that provides information about the source of a connection-related event. ConnectionEvent objects are generated when an application closes a pooled connection and when an error occurs. The ConnectionEvent object contains two kinds of information:

* The pooled connection closed by the application
* In the case of an error event, the SQLException about to be thrown to the application

**Constructors::-**

|  |
| --- |
|  |
| [**ConnectionEvent**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionEvent.html#ConnectionEvent-javax.sql.PooledConnection-)([**PooledConnection**](https://docs.oracle.com/javase/8/docs/api/javax/sql/PooledConnection.html) con)  Constructs a ConnectionEvent object initialized with the given PooledConnection object. |
| [**ConnectionEvent**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionEvent.html#ConnectionEvent-javax.sql.PooledConnection-java.sql.SQLException-)([**PooledConnection**](https://docs.oracle.com/javase/8/docs/api/javax/sql/PooledConnection.html) con, [**SQLException**](https://docs.oracle.com/javase/8/docs/api/java/sql/SQLException.html) ex)  Constructs a ConnectionEvent object initialized with the given PooledConnection object and SQLException object. |

**Method and Description:-**

|  |  |
| --- | --- |
| [**SQLException**](https://docs.oracle.com/javase/8/docs/api/java/sql/SQLException.html) | [**getSQLException**](https://docs.oracle.com/javase/8/docs/api/javax/sql/ConnectionEvent.html#getSQLException--)()  Retrieves the SQLException for this ConnectionEvent object. |

## RowSetEvent

public class **RowSetEvent** extends [EventObject](https://docs.oracle.com/javase/8/docs/api/java/util/EventObject.html)

An Event object generated when an event occurs to a RowSet object. A RowSetEvent object is generated when a single row in a rowset is changed, the whole rowset is changed, or the rowset cursor moves.

When an event occurs on a RowSet object, one of the RowSetListener methods will be sent to all registered listeners to notify them of the event. An Event object is supplied to the RowSetListener method so that the listener can use it to find out which RowSet object is the source of the event.

**Constructors::-**

[**RowSetEvent**](https://docs.oracle.com/javase/8/docs/api/javax/sql/RowSetEvent.html#RowSetEvent-javax.sql.RowSet-)([**RowSet**](https://docs.oracle.com/javase/8/docs/api/javax/sql/RowSet.html) source)

Constructs a RowSetEvent object initialized with the given RowSet object.

## StatementEvent

public class **StatementEvent** extends [EventObject](https://docs.oracle.com/javase/8/docs/api/java/util/EventObject.html)

A StatementEvent is sent to all StatementEventListeners which were registered with a PooledConnection.

This occurs when the driver determines that a PreparedStatement that is associated with the PooledConnection has been closed or the driver determines is invalid.

**Constructors::-**

|  |
| --- |
| [**StatementEvent**](https://docs.oracle.com/javase/8/docs/api/javax/sql/StatementEvent.html#StatementEvent-javax.sql.PooledConnection-java.sql.PreparedStatement-)([**PooledConnection**](https://docs.oracle.com/javase/8/docs/api/javax/sql/PooledConnection.html) con, [**PreparedStatement**](https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html) statement)  Constructs a StatementEvent with the specified PooledConnection and PreparedStatement. |
| [**StatementEvent**](https://docs.oracle.com/javase/8/docs/api/javax/sql/StatementEvent.html#StatementEvent-javax.sql.PooledConnection-java.sql.PreparedStatement-java.sql.SQLException-)([**PooledConnection**](https://docs.oracle.com/javase/8/docs/api/javax/sql/PooledConnection.html) con, [**PreparedStatement**](https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html) statement, [**SQLException**](https://docs.oracle.com/javase/8/docs/api/java/sql/SQLException.html) exception)  Constructs a StatementEvent with the specified PooledConnection, PreparedStatement and SQLException |

**Method and Description:-**

|  |  |
| --- | --- |
| [**SQLException**](https://docs.oracle.com/javase/8/docs/api/java/sql/SQLException.html) | [**getSQLException**](https://docs.oracle.com/javase/8/docs/api/javax/sql/StatementEvent.html#getSQLException--)()  Returns the SQLException the driver is about to throw |
| [**PreparedStatement**](https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html) | [**getStatement**](https://docs.oracle.com/javase/8/docs/api/javax/sql/StatementEvent.html#getStatement--)()  Returns the PreparedStatement that is being closed or is invalid |