Java 8 JDBC Improvements

In Java 8, Java made two major changes in JDBC API.

1) The JDBC-ODBC Bridge has been removed.

Oracle does not support the JDBC-ODBC Bridge. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.

2) Added some new features in JDBC 4.2.

Java JDBC 4.2 introduces the following features:

- Addition of REF_CURSOR support.
- Addition of java.sql.DriverAction Interface
- Addition of security check on deregisterDriver Method in DriverManager Class
- Addition of the java.sql.SQLType Interface
- Addition of the java.sql.JDBCType Enum
- Add Support for large update counts
- Changes to the existing interfaces
- Rowset 1.2: Lists the enhancements for JDBC RowSet.

Java JDBC DriverAction

It is an interface that must be implemented when a Driver wants to be notified by DriverManager. It is added in java.sql package and contains only one abstract method.

DriverAction Method

Method	Description
void deregister()	This method called by DriverManager.deregisterDriver(Driver) to notify the JDBC driver that it was de-registered.

The deregister method is intended only to be used by JDBC Drivers and not by applications.

JDBC drivers are recommended not to implement the DriverAction in a public class.

If there are active connections to the database at the time that the deregister method is called, it is implementation specific as to whether the connections are closed or allowed to continue. Once this method is called, it is implementation specific as to whether the driver may limit the ability to create new connections to the database, invoke other Driver methods or throw a SQLException.

Java JDBC4.2 DriverAction Example

```
import java.sql.*;
// implementing DriverAction interface
class JdbcExample implements DriverAction{
   // implementing deregister method of DriverAction interface
   @Override
   public void deregister() {
      System.out.println("Driver deregistered");
   }
   public static void main(String args[]){
      try{
         // Creating driver instance
         Driver driver = new com.mysql.jdbc.Driver();
         // Creating Action Driver
         DriverAction da = new JdbcExample();
         // Registering driver by passing driver and driverAction
         DriverManager.registerDriver(driver, da);
         // Creating connection
         Connection con=DriverManager.getConnection("jdbc:mysql:///localhost:3306/student","root","mysql");
         //Here student is database name, root is username and password is mysql
         Statement stmt=con.createStatement();
         // Executing SQL query
         ResultSet rs=stmt.executeQuery("select * from user");
         while(rs.next()){
            System.out.println(rs.getInt(1)+""+rs.getString(2)+""+rs.getString(3));
         }
         // Closing connection
         con.close();
         // Calling deregisterDriver method
         DriverManager.deregisterDriver(driver);
      }catch(Exception e){ System.out.println(e);}
   }

}
```

Output:

```
1  Arun  25
2  Irfan  22
3  Neraj kumar  25
Driver deregistered
```

Java JDBC SQLType

This interface is used to identify a generic SQL type, JDBC type or a vendor specific data type.

It provides following methods.

Method	Description
String getName()	It returns the SQLType name that represents a SQL data type.
String getVendor()	It returns the name of the vendor that supports this data type. The value returned typically is the package name for this vendor.
Integer getVendorTypeNumber()	It returns the vendor specific type number for the data type.

Java JDBCType

It is an Enumeration which defines the constants that are used to identify generic SQL types, called JDBC types. It extends java.lang.Enum and implements java.sql.SQLType.

JDBCType Fields

The following table contains constants defined in the JDBCType.

Enum constant	Description
public static final JDBCType ARRAY	It identifies the generic SQL type ARRAY.
public static final JDBCType BIGINT	It identifies the generic SQL type BIGINT.
public static final JDBCType BIT	It identifies the generic SQL type BIT.
public static final JDBCType BLOB	It identifies the generic SQL type BLOB.
public static final JDBCType BOOLEAN	It identifies the generic SQL type BOOLEAN.
public static final JDBCType CHAR	It identifies the generic SQL type CHAR.
public static final JDBCType CLOB	It identifies the generic SQL type CLOB.
public static final JDBCType DATALINK	It identifies the generic SQL type DATALINK.
public static final JDBCType DATE	It identifies the generic SQL type DATE.
public static final JDBCType DECIMAL	It identifies the generic SQL type DECIMAL.
public static final JDBCType DISTINCT	It identifies the generic SQL type DISTINCT.
public static final JDBCType DOUBLE	It identifies the generic SQL type DOUBLE.
public static final JDBCType FLOAT	It identifies the generic SQL type FLOAT.
public static final JDBCType INTEGER	It identifies the generic SQL type INTEGER.
public static final JDBCType JAVA_OBJECT	It indicates that the SQL type is database-specific and gets mapped to a Java object that can be accessed via the methods getObject and setObject.
Public static final JDBCType LONGNVARCHAR	It identifies the generic SQL type LONGNVARCHAR.
public static final JDBCType NCHAR	It identifies the generic SQL type NCHAR.
public static final JDBCType NCLOB	It identifies the generic SQL type NCLOB.
public static final JDBCType NULL	It identifies the generic SQL value NULL.
public static final JDBCType NUMERIC	It identifies the generic SQL type NUMERIC.
public static final JDBCType NVARCHAR	It identifies the generic SQL type NVARCHAR.
public static final JDBCType OTHER	It indicates that the SQL type is database-specific and gets mapped to a Java object that can be accessed via the methods getObject and setObject.
public static final JDBCType REAL	It identifies the generic SQL type REAL.Identifies the generic SQL type VARCHAR.
public static final JDBCType REF	It identifies the generic SQL type REF.
public static final JDBCType REF_CURSOR	It identifies the generic SQL type REF_CURSOR.
public static final JDBCType ROWID	It identfies the SQL type ROWID.
public static final JDBCType SMALLINT	It identifies the generic SQL type SMALLINT.
public static final JDBCType SQLXML	It identifies the generic SQL type SQLXML.
public static final JDBCType STRUCT	It identifies the generic SQL type STRUCT.
public static final JDBCType TIME	It identifies the generic SQL type TIME.
public static final JDBCType TIME_WITH_TIMEZONE	It identifies the generic SQL type TIME_WITH_TIMEZONE.
public static final JDBCType TIMESTAMP	It identifies the generic SQL type TIMESTAMP.
public static final JDBCType TIMESTAMP_WITH_TIMEZONE	It identifies the generic SQL type TIMESTAMP_WITH_TIMEZONE.
public static final JDBCType TINYINT	It identifies the generic SQL type TINYINT.
public static final JDBCType VARBINARY	It identifies the generic SQL type VARBINARY.
public static final JDBCType VARCHAR	It identifies the generic SQL type VARCHAR.

JDBCType Methods

Method	Description
public String getName()	It returns the SQLType name that represents a SQL data type.
public String getVendor()	It returns the name of the vendor that supports this data type.
public Integer getVendorTypeNumber()	It returns the vendor specific type number for the data type.
public static JDBCType valueOf(int type)	It returns the JDBCType that corresponds to the specified Types value. It throws IllegalArgumentException, if this enum type has no constant with the specified Types value.
public static JDBCType valueOf(String name)	It returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. It throws IllegalArgumentException, if this enum type has no constant with the specified name. It throws NullPointerException, if the argument is null.
public static JDBCType[] values()	It returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants.