1. **Create a database**
2. **Execute the transactions for rollback**
   * Import the database
   * Load and register drivers if necessary
   * Create a new connection
   * Create a statement for commit/rollback
   * Execute the query for commit/rollback
   * Process the results
   * Close the connection else previous processing may lose if any.

**Step 1:**We can also rollback the modifications in the database up to a particular flag or save point by just passing the needed Save points name as a parameter into this below method −

**// Set the Flag or Save point**

**con.rollback("MysavePoint");**

**Step 2. To roll back a transaction:**Load the JDBC driver, by using the API method ***[forName(String className)](https://www.geeksforgeeks.org/class-forname-method-in-java-with-examples/)*** of the Class. In this example, we are using the Oracle

* Register the required driver using the [registerDriver( )](https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/) method

**// To register the needed Driver**

**DriverManager.registerDriver(new com.mysql.jdbc.Driver());**

* Get the connection information using the getConnection() API method of the DriverManager:

**// For getting the connection**

**String url = "jdbc:mysql://localhost/mydatabase/icpc";**

**Connection conn = DriverManager.getConnection(string url, String user, String password);**

* Disable the auto-commit using the API method of off connection setAutoCommit(boolean auto-commit) method as:

// Set the auto commit false. This will execute all

// SQL statements as individual transactions

con.setAutoCommit(false);

* Now, set the save point using the setSavepoint() or, commit the transaction using the API method commit( ) of connection as shown below−

**Savepoint savePoint = con.setSavepoint("MysavePoint");**

**Con.commit();**

* If any SQL exception is found then, in that case, invoke rollback( ) API method for the whole transaction to till the previously set savepoint:

**con.rollback() Or,**

**con. rollback(my\_Savepoint);**

**Implementation:**Java program to demonstrate both rollback() and commit() program is as follows

import java.io.\*;

import java.sql.\*;

import java.sql.SQLException;

import java.sql.DriverManager;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.PreparedStatement;

import java.sql.Date;

class demo {

static String DB\_URL= **"jdbc:mysql:///world"**;

static String DB\_USER = "local";

static String DB\_PASSWORD = "test";

public static void main(String args[])

{

try {

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

Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

// set auto commit of the connection to false

conn.setAutoCommit(false);

String sql\_= "INSERT INTO Employee (empid, empname) VALUES (?, ?)";

PreparedStatement ps = conn.prepareStatement(sql\_);

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

while (true) {

System.out.print("Enter emp\_Id: ");

String s\_1 = br.readLine();

int empid = Integer.parseInt(s\_1);

System.out.print("Enter emp\_name: ");

String name = br.readLine();

ps.setInt(1, empid);

ps.setString(2, name);

ps.executeUpdate();

System.out.println("commit or rollback");

String answer = br.readLine();

// If user wants to commit

if (answer.equals("commit")) {

conn.commit();

}

// If user wants to rollback

if (answer.equals("rollback")) {

// Rollback the update in case if some flaw in your record

conn.rollback();

}

System.out.println("Do you want to include more records");

System.out.println("\n yes/no");

String answ = br.readLine();

if (answ.equals("no")) {

break;

}

}

conn.commit();

System.out.println("record is successfully saved");

conn.close();

}

catch (Exception exc) {

exc.printStackTrace();

}

}

}