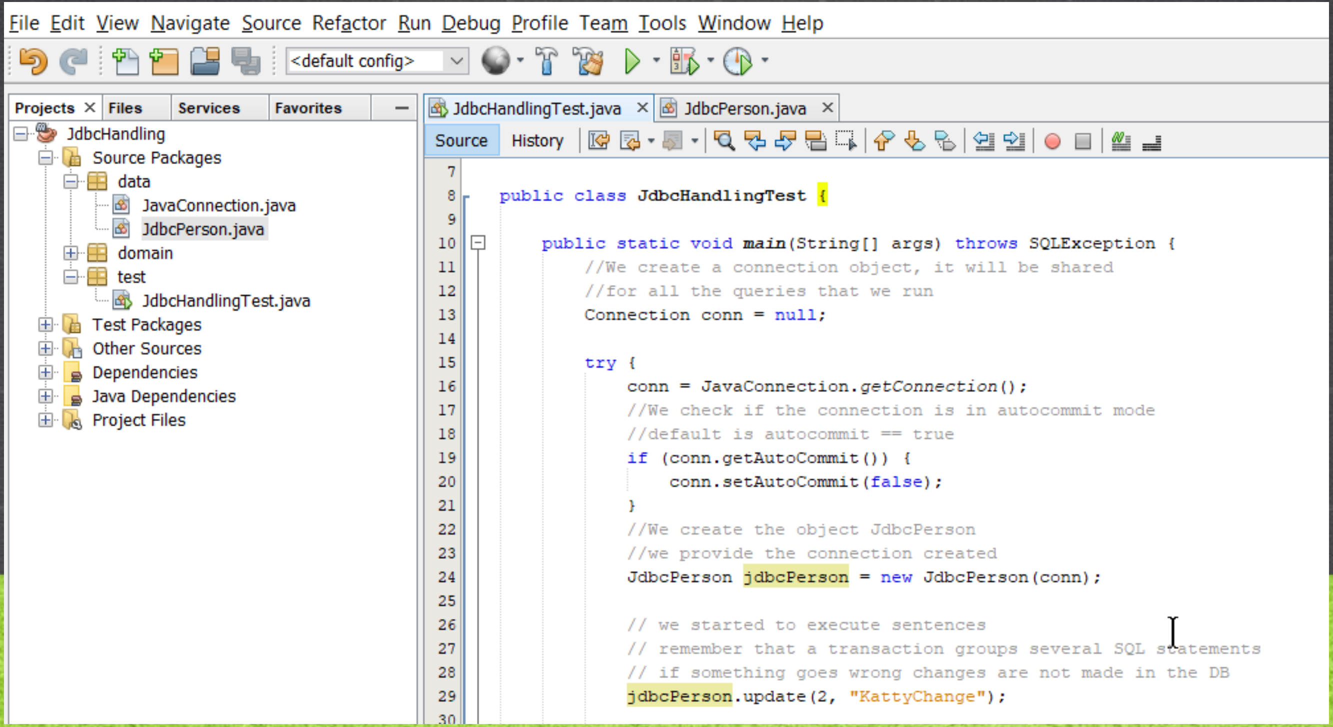
EXERCISE

JDBC TRANSACTIONS

Experiencia y Conocimiento para tu vida

EXERCISE OBJECTIVE

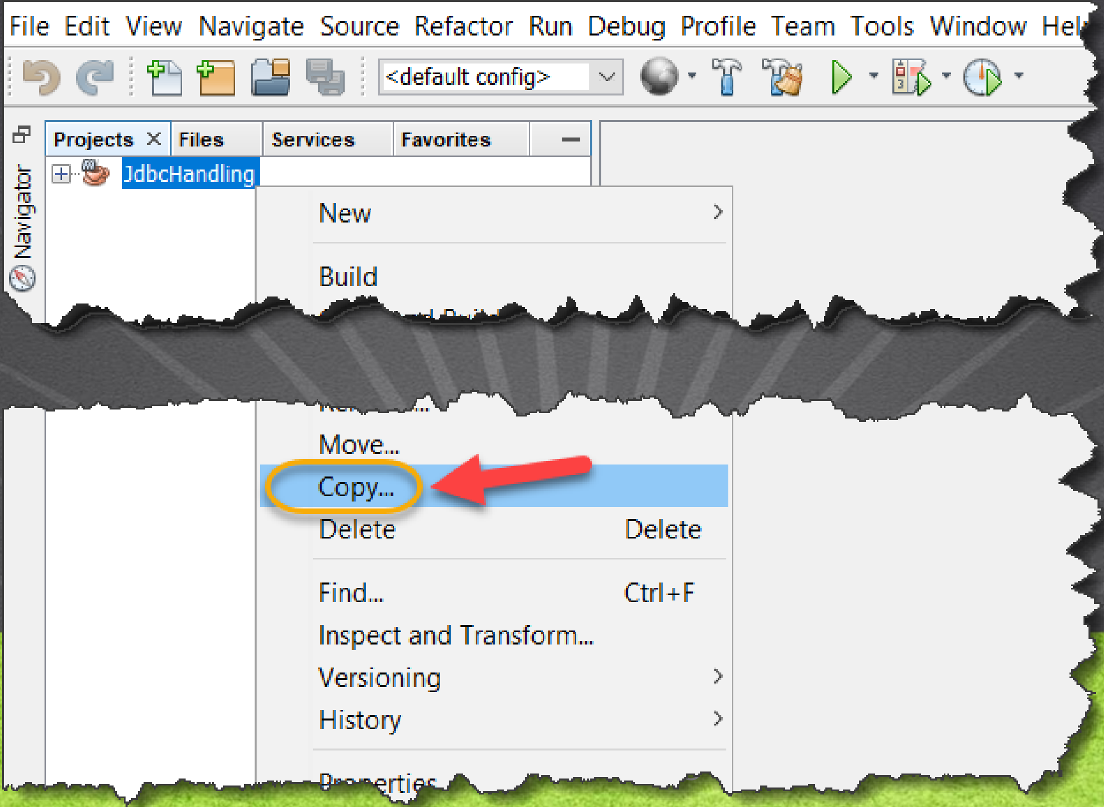
Create a program to practice the concept of transactions with JDBC. At the end we should observe the following:



```java

public class JdbcHandlingTest {

    public static void main(String[] args) throws SQLException {
        //We create a connection object, it will be shared
        //for all the queries that we run
        Connection conn = null;

        try {
            conn = JavaConnection.getConnection();
            //We check if the connection is in autocommit mode
            //default is autocommit == true
            if (conn.getAutoCommit()) {
                conn.setAutoCommit(false);
            }
            //We create the object JdbcPerson
            //we provide the connection created
            JdbcPerson jdbcPerson = new JdbcPerson(conn);

            // we started to execute sentences
            // remember that a transaction groups several SQL statements
            // if something goes wrong changes are not made in the DB
            jdbcPerson.update(2, "KattyChange");
```

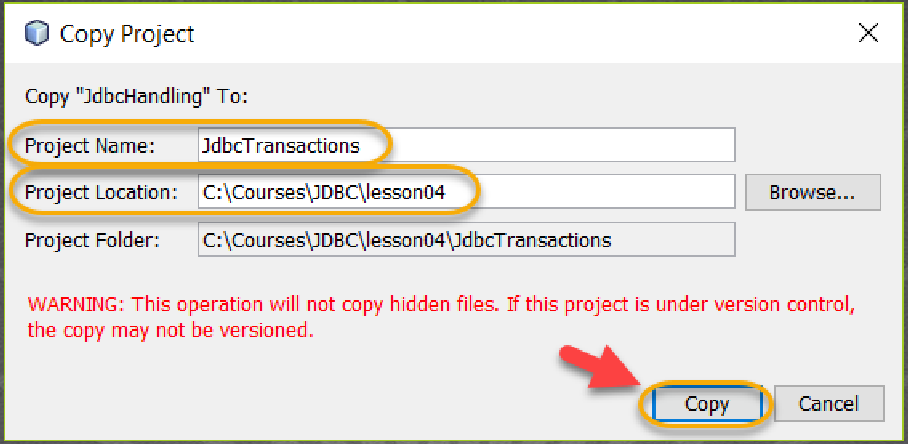
1. COPY THE PREVIOUS PROJECT

Copy the JdbcHandling project:



1. COPY THE PREVIOUS PROJECT

Copy the JdbcHandling project:



Copy "JdbcHandling" To:

Project Name: JdbcTransactions

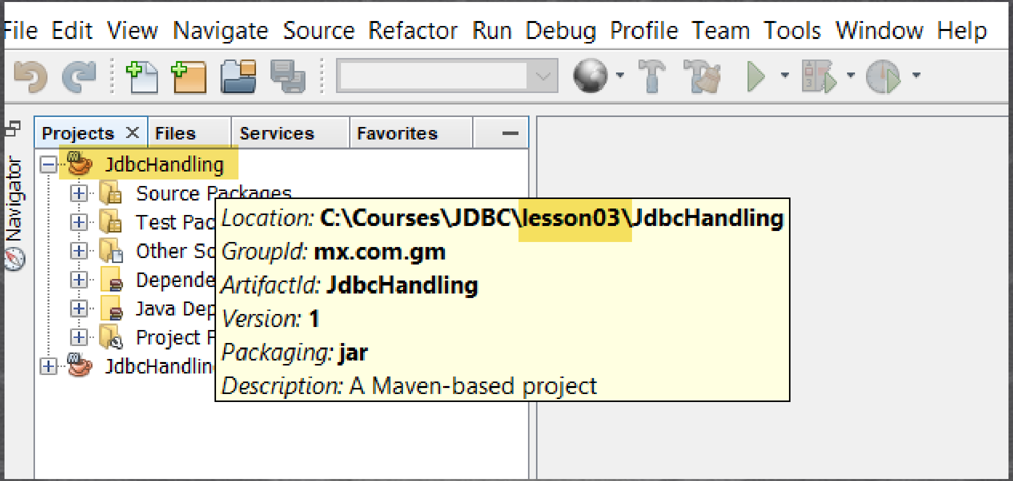
Project Location: C:\Courses\JDBC\lesson04

Project Folder: C:\Courses\JDBC\lesson04\JdbcTransactions

WARNING: This operation will not copy hidden files. If this project is under version control, the copy may not be versioned.

2. CLOSE THE PREVIOUS PROJECT

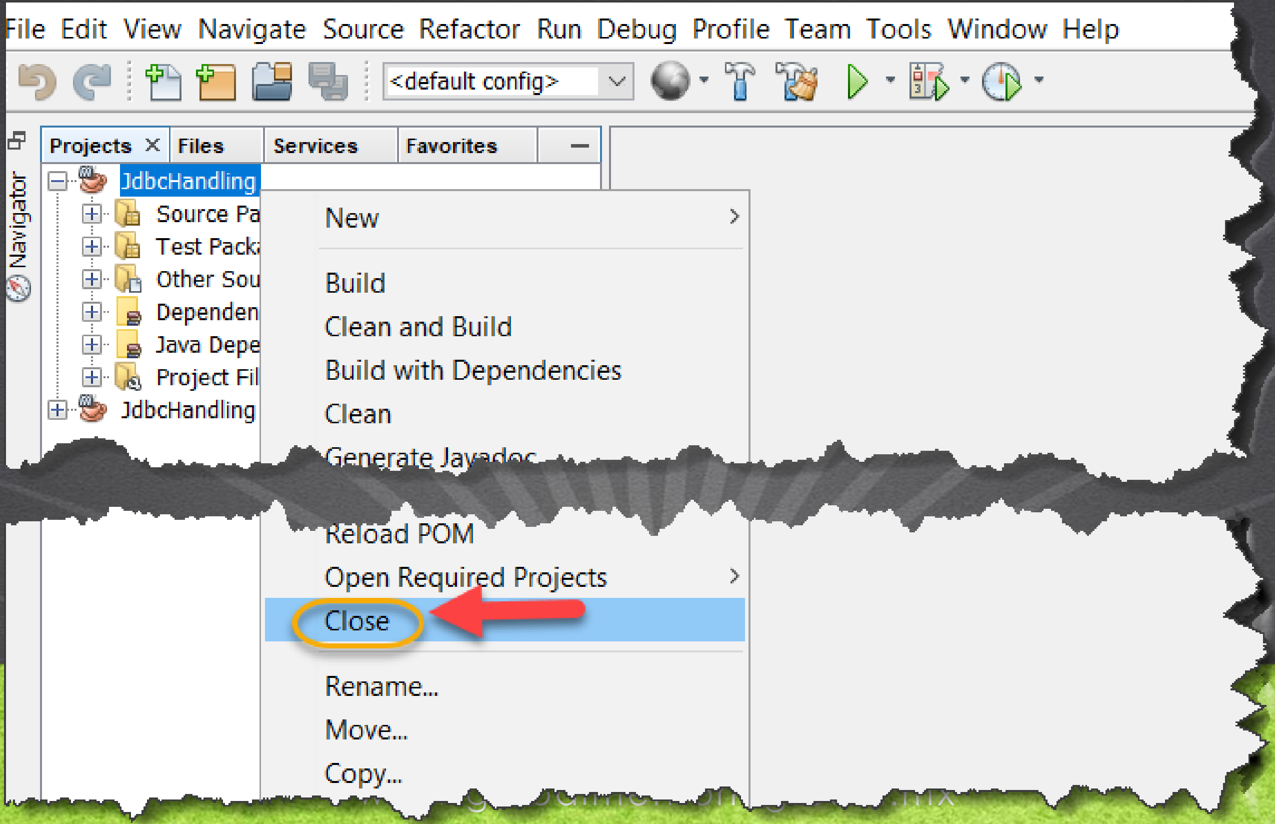
Close the previous Project. To identify the Project leave the cursor over the Project and it will show the information to detect which Project to close:



Location: C:\Courses\JDBC\lesson03\JdbcHandling
GroupId: mx.com.gm
ArtifactId: JdbcHandling
Version: 1
Packaging: jar
Description: A Maven-based project

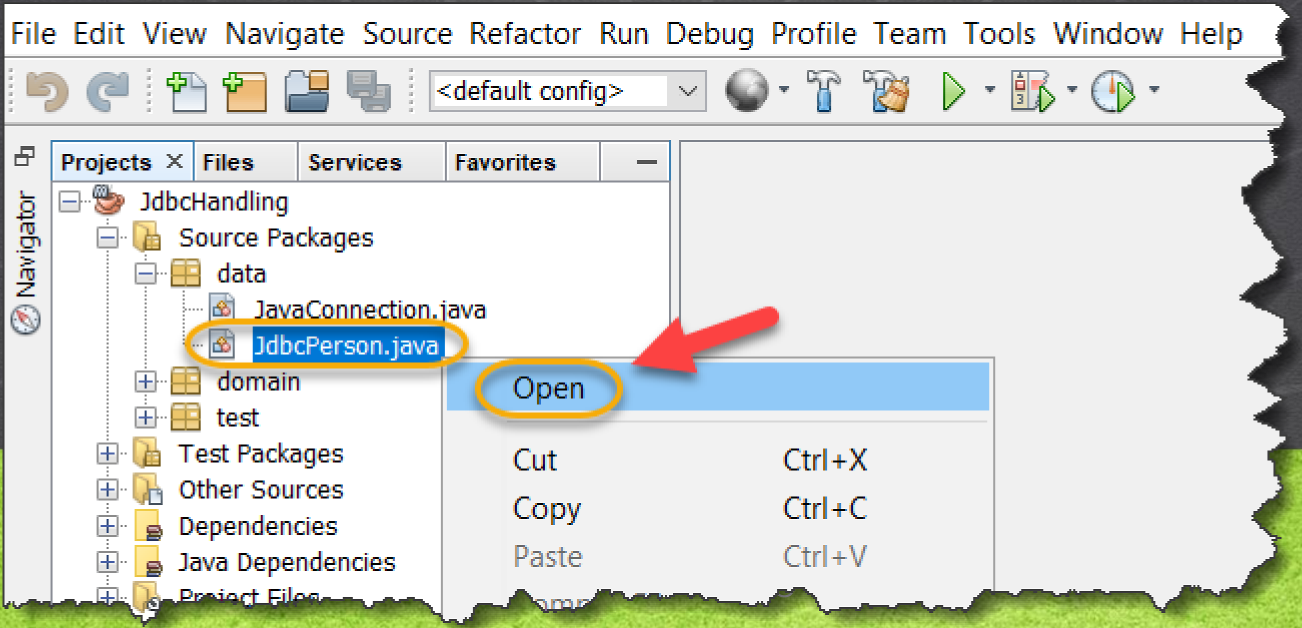
2. CLOSE THE PREVIOUS PROJECT

Close the previous Project:



3. MODIFY THE CODE

Open the JdbcPerson.java class and modify the code. We eliminate the cath block of each method, so that any error is propagated towards the class that sends to call the methods of this class:



3. MODIFY THE CODE

JdbcPerson.java:

Click to download

```
package data;

import domain.Person;
import java.sql.*;
import java.util.*;

/**
 * Class that contains the methods of SELECT, INSERT, UPDATE and DELETE for the
 * Person table in MYSQL
 *
 * @author Ing. Ubaldo Acosta
 *
 */
public class JdbcPerson {

    //Variable that stores a connection as a reference
    //this option is received in the constructor of this class
    //and allows to reuse the same connection to execute
    //several queries of this class, optionally you can
    //use for the use of a transaction in SQL
    private java.sql.Connection userConn;

    private final String SQL_INSERT = "INSERT INTO person(name) VALUES(?)";

    private final String SQL_UPDATE = "UPDATE person SET name=? WHERE id_person=?";

    private final String SQL_DELETE = "DELETE FROM person WHERE id_person = ?";

    private final String SQL_SELECT = "SELECT id_person, name FROM person ORDER BY id_person";
```

3. MODIFY THE CODE

JdbcPerson.java:

Click to download

```
    /*
     * Add the empty constructor
     */
    public JdbcPerson() {
    }

    /**
     * Constructor that assigns an existing connection to be used in the queries
     * of this class
     *
     * @param conn Connection to the DB previously created
     */
    public JdbcPerson(Connection conn) {
        this.userConn = conn;
    }
```

3. MODIFY THE CODE

JdbcPerson.java:

Click to download

```java
/**
 * Method that inserts a record in the Person table
 *
 * @param name
 * @return int the number of modified rows
 * @throws java.sql.SQLException
 */
public int insert(String name) throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0; //affected rows
    try {
        //If the connection to reuse is different from null, it is used, if not
        //create a new connection
        conn = (this.userConn != null) ? this.userConn : JavaConnection.getConnection();
        stmt = conn.prepareStatement(SQL_INSERT);
        stmt.setString(1, name);//param 1 => ? name
        System.out.println("Executing query:" + SQL_INSERT);
        rows = stmt.executeUpdate();
        System.out.println("Affected records:" + rows);
    } finally {
        if (this.userConn == null) {
            JavaConnection.close(conn);
        }
    }
    return rows;
}
```

3. MODIFY THE CODE

JdbcPerson.java:

Click to download

```
/**
 * Method that updates an existing record
 *
 * @param idPerson Primary key
 * @param name Name value
 * @return int modified rows
 * @throws java.sql.SQLException
 */
public int update(int idPerson, String name) throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = (this.userConn != null) ? this.userConn : JavaConnection.getConnection();
        System.out.println("Executing query:" + SQL_UPDATE);
        stmt = conn.prepareStatement(SQL_UPDATE);
        stmt.setString(1, name);//param 1 => ?  name
        stmt.setInt(2, idPerson);//param 2 => ? id_person
        rows = stmt.executeUpdate();
        System.out.println("Updated records:" + rows);
    } finally {
        if (this.userConn == null) {
            JavaConnection.close(conn);
        }
    }
    return rows;
}
```

3. MODIFY THE CODE

JdbcPerson.java:

Click to download

```
/**
 * Method that deletes an existing record
 *
 * @param idPerson Primary key
 * @return int rows affected
 * @throws java.sql.SQLException
 */
public int delete(int idPerson) throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = (this.userConn != null) ? this.userConn : JavaConnection.getConnection();
        System.out.println("Executing query:" + SQL_DELETE);
        stmt = conn.prepareStatement(SQL_DELETE);
        stmt.setInt(1, idPerson);//param 1 => ? id_person
        rows = stmt.executeUpdate();
        System.out.println("Deleted records:" + rows);
    } finally {
        if (this.userConn == null) {
            JavaConnection.close(conn);
        }
    }
    return rows;
}
```

3. MODIFY THE CODE

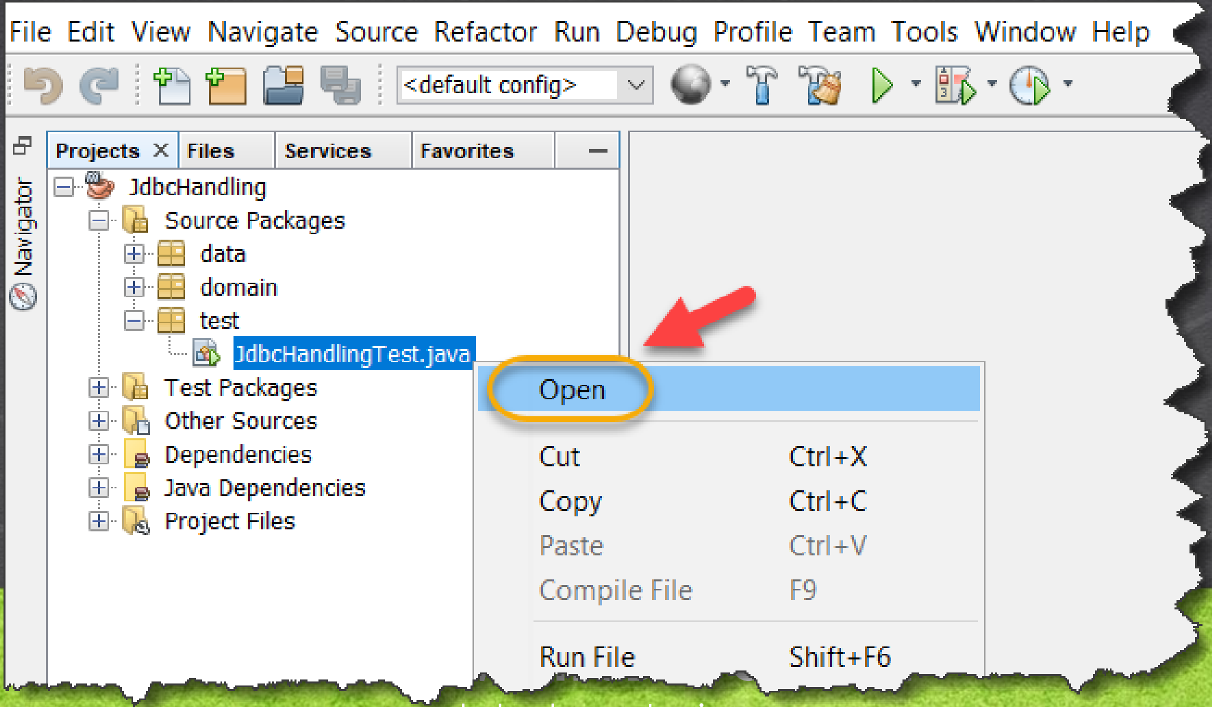
JdbcPerson.java:

Click to download

```
    /**
     * Method that returns the contents of the Person table
     *
     * @return list of person objects
     * @throws java.sql.SQLException
     */
    public List<Person> select() throws SQLException {
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        Person persona = null;
        List<Person> personas = new ArrayList<>();
        try {
            conn = (this.userConn != null) ? this.userConn : JavaConnection.getConnection();
            stmt = conn.prepareStatement(SQL_SELECT);
            rs = stmt.executeQuery();
            while (rs.next()) {
                int id_persona = rs.getInt(1);
                String nombre = rs.getString(2);
                persona = new Person();
                persona.setIdPerson(id_persona);
                persona.setName(nombre);
                personas.add(persona);
            }
        } finally {
            if (this.userConn == null) {
                JavaConnection.close(conn);
            }
        }
        return personas;
    }
}
```

4. MODIFY THE CODE

Open the JdbcHandlingTest.java class and modify the code to test the concept of transactions in JDBC:



4. MODIFY THE CODE

JdbcHandlingTest.java:

Click to download

```
package test;

import data.*;
import domain.Person;
import java.sql.*;
import java.util.List;

public class JdbcHandlingTest {

    public static void main(String[] args) throws SQLException {
        //We create a connection object, it will be shared
        //for all the queries that we run
        Connection conn = null;

        try {
            conn = JavaConnection.getConnection();
               //We check if the connection is in autocommit mode
            //default is autocommit == true
            if (conn.getAutoCommit()) {
                conn.setAutoCommit(false);
            }
               //We create the object JdbcPerson
            //we provide the connection created
            JdbcPerson jdbcPerson = new JdbcPerson(conn);
```

4. MODIFY THE CODE

JdbcHandlingTest.java:

Click to download

```
            // we started to execute sentences
            // remember that a transaction groups several SQL statements
            // if something goes wrong changes are not made in the DB
            jdbcPerson.update(2, "KattyChange");

            //We caused an error exceeding 45 characters
            //of the name field
            jdbcPerson.insert("Miguel21234123123123123123123123123123123123123");

            //save the changes if no error is found
            conn.commit();
        } catch (SQLException e) {
            //if any error is found, we execute the rollback
            try {
                System.out.println("Execute the rollback");
                //We print the exception to the console
                e.printStackTrace(System.out);
                //executhe the rollback
                conn.rollback();
            } catch (SQLException e1) {
                e1.printStackTrace(System.out);
            }
        }
        //List people to see any change
        listPeople();
    }
```

4. MODIFY THE CODE

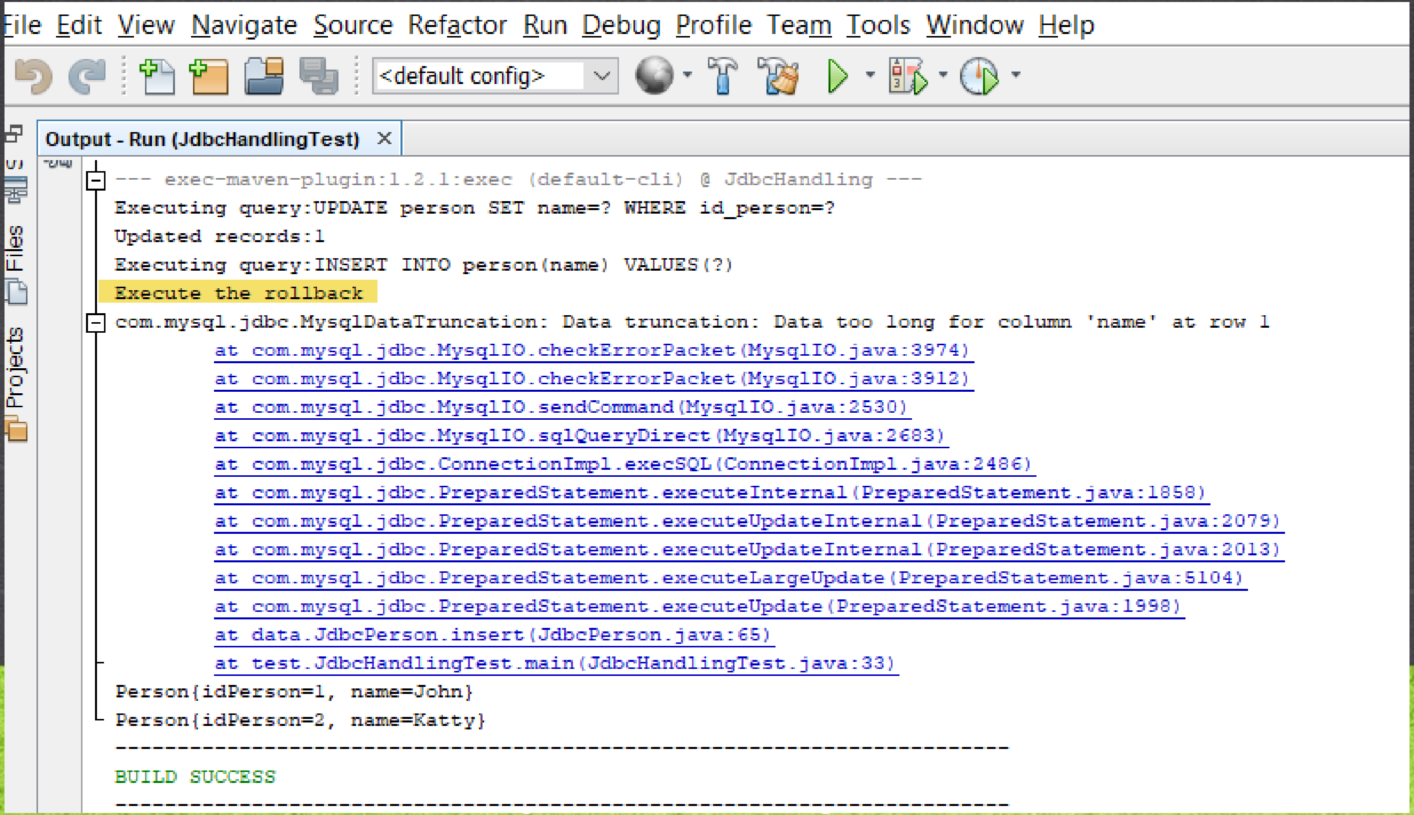
JdbcHandlingTest.java:

Click to download

```
    private static void listPeople() throws SQLException{
        JdbcPerson jdbcPerson = new JdbcPerson();
        List<Person> people = jdbcPerson.select();
        for (Person person : people) {
            System.out.print(person);
            System.out.println("");
        }
    }

}
```

5. EXECUTE THE PROJECT

The result is as follows:


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - Run (JdbcHandlingTest)
Files
Projects

```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ JdbcHandling ---
Executing query:UPDATE person SET name=? WHERE id_person=?
Updated records:1
Executing query:INSERT INTO person(name) VALUES(?)
Execute the rollback
com.mysql.jdbc.MysqlDataTruncation: Data truncation: Data too long for column 'name' at row 1
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3974)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3912)
        at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2530)
        at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2683)
        at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2486)
        at com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:1858)
        at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2079)
        at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2013)
        at com.mysql.jdbc.PreparedStatement.executeLargeUpdate(PreparedStatement.java:5104)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1998)
        at data.JdbcPerson.insert(JdbcPerson.java:65)
        at test.JdbcHandlingTest.main(JdbcHandlingTest.java:33)
Person{idPerson=1, name=John}
Person{idPerson=2, name=Katty}
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

EXERCISE CONCLUSION

- With this exercise we have put into practice the concept of transactions in JDBC.
- We have seen that when executing a statement that causes an error, we can rollback the entire transaction and therefore we will not affect the state of the database.
- For more information on this topic:
- https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html

ONLINE COURSE

JAVA WITH JDBC

By: Eng. Ubaldo Acosta

Global Mentoring

JAVA WITH JDBC COURSE

www.globalmentoring.com.mx