EXERCISE

JDBC HANDLING

EXERCISE OBJECTIVE

Create a program for managing people objects using JDBC. At the end we should observe the following:

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects | Files | Services | Favorites

- JdbcHandling
  - Source Packages
    - data
      - JavaConnection.java
      - JdbcPerson.java
    - domain
      - Person.java
    - test
      - JdbcHandlingTest.java
  - Test Packages
  - Other Sources
  - Dependencies
  - Java Dependencies
  - Project Files
    - pom.xml

JavaConnection.java | JdbcPerson.java | JdbcHandlingTest.java

Source | History

```java
package test;

import data.JdbcPerson;
import domain.Person;
import java.util.List;

public class JdbcHandlingTest {

    public static void main(String[] args) {
        JdbcPerson jdbcPerson = new JdbcPerson();

        //Test of the insert method
        //jdbcPerson.insert("Charly");

        //Test of the update method
        //jdbcPerson.update(1, "John");
```

Output - Run (JdbcHandlingTest)

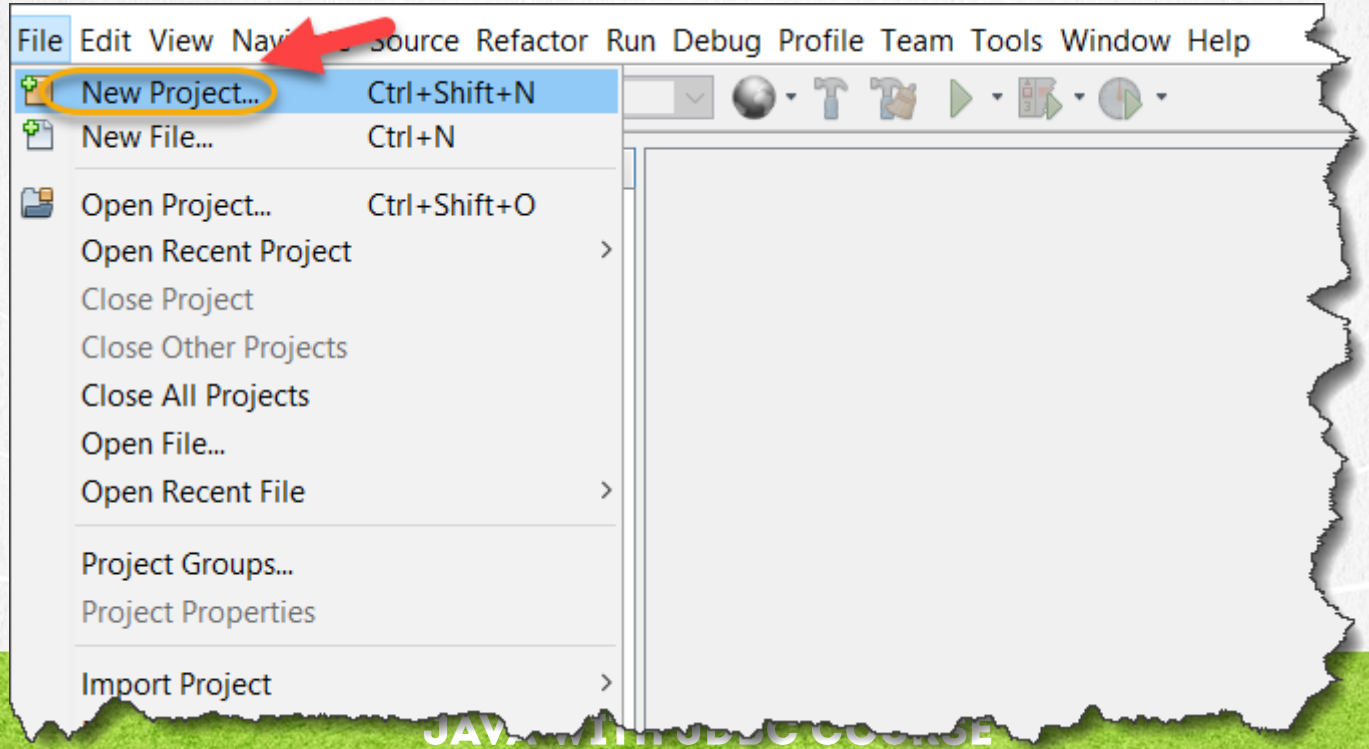
```
Scanning for projects...

------------------------------------------------------------------------
Building JdbcHandling 1
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ JdbcHandling ---
Person{idPerson=1, name=John}
Person{idPerson=2, name=Katty}
------------------------------------------------------------------------
```

main - Navigator

Members | <empty>

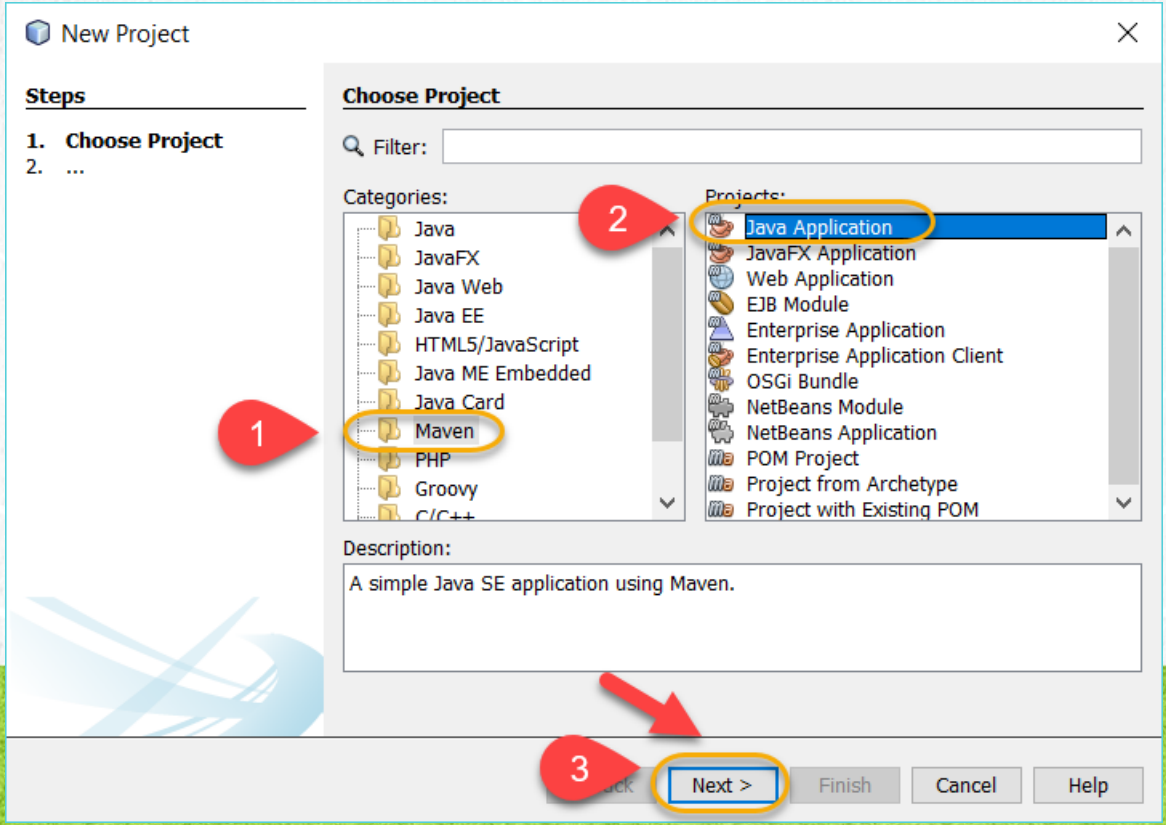
1. CREATE A NEW PROJECT

Create a new project:



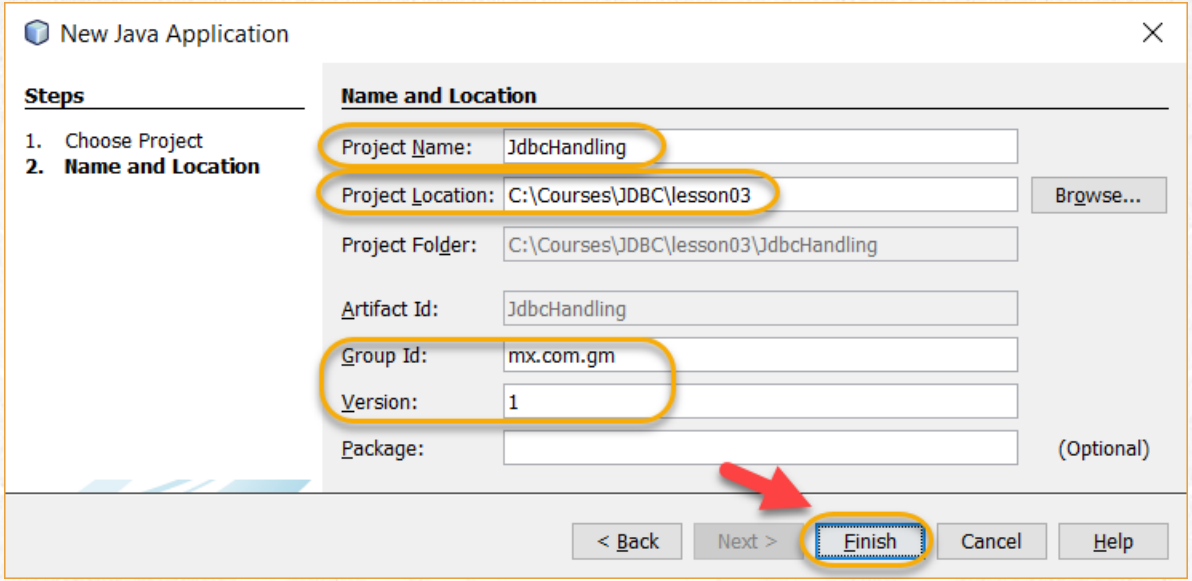
1. CREATE A NEW PROJECT

Create a new project:



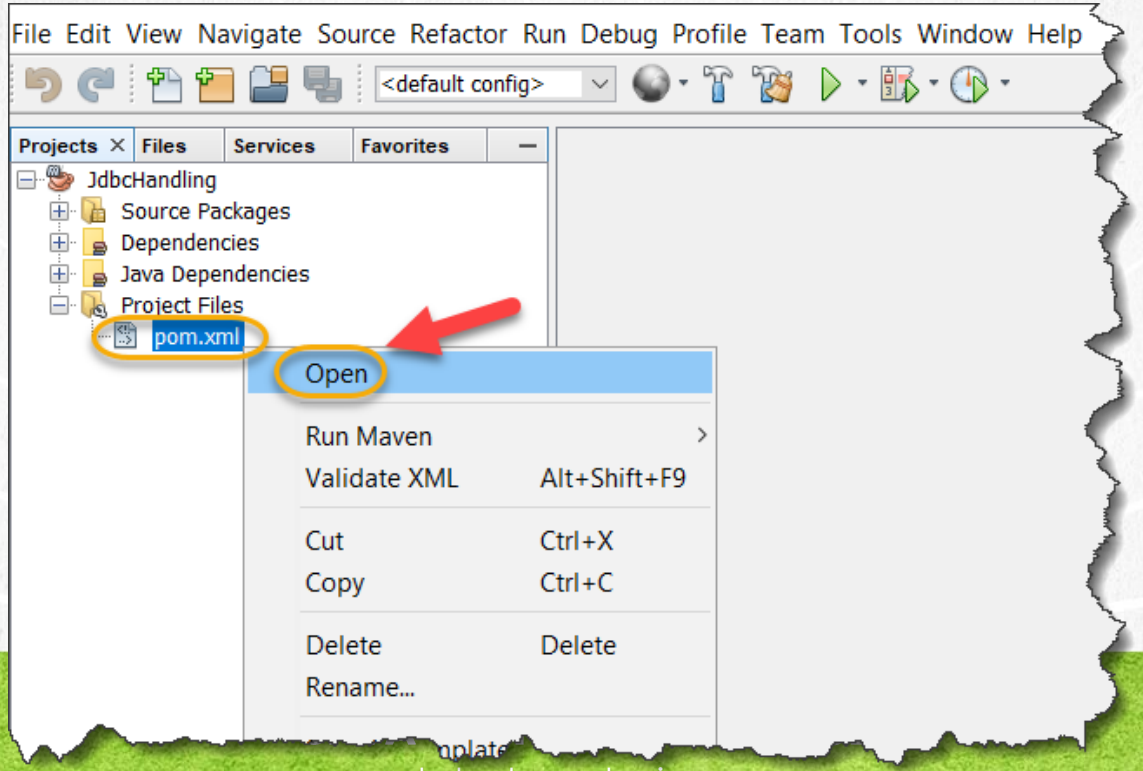
1. CREATE A NEW PROJECT

Create a new project:



2. MODIFY THE POM.XML

Modify the pom.xml:



2. MODIFY THE CODE

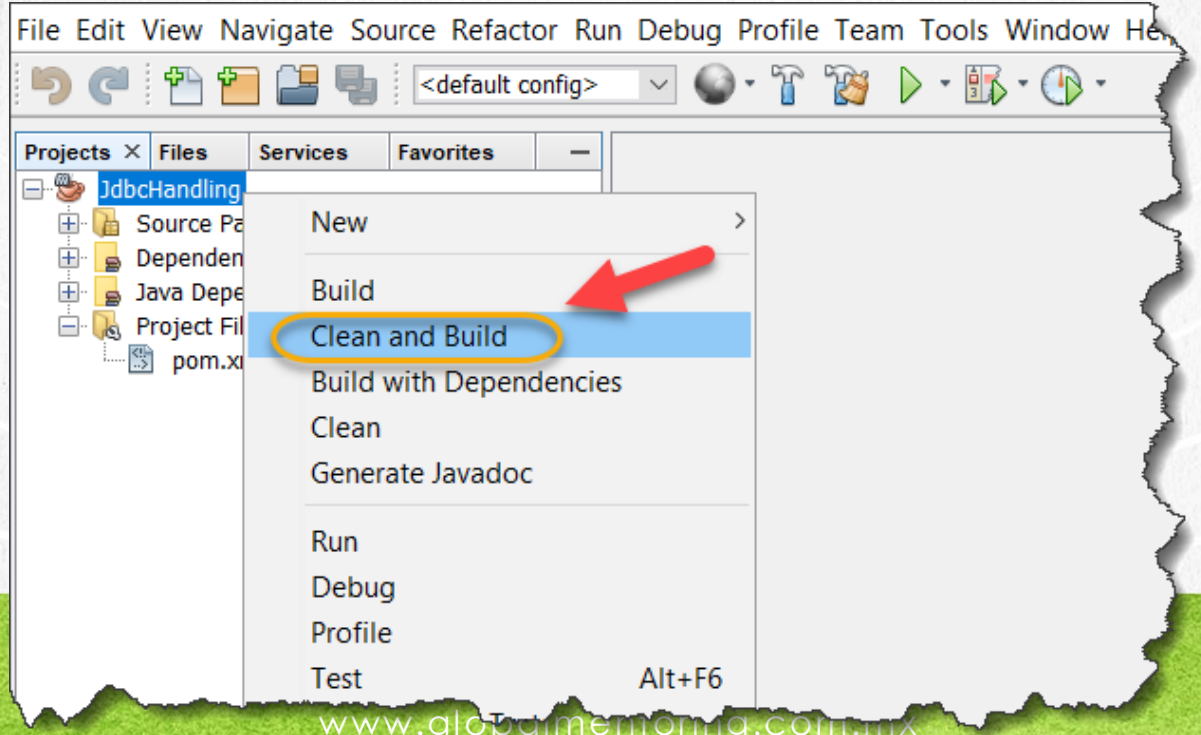
pom.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>mx.com.gm</groupId>
    <artifactId>JdbcHandling</artifactId>
    <version>1</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.46</version>
        </dependency>
    </dependencies>
</project>
```

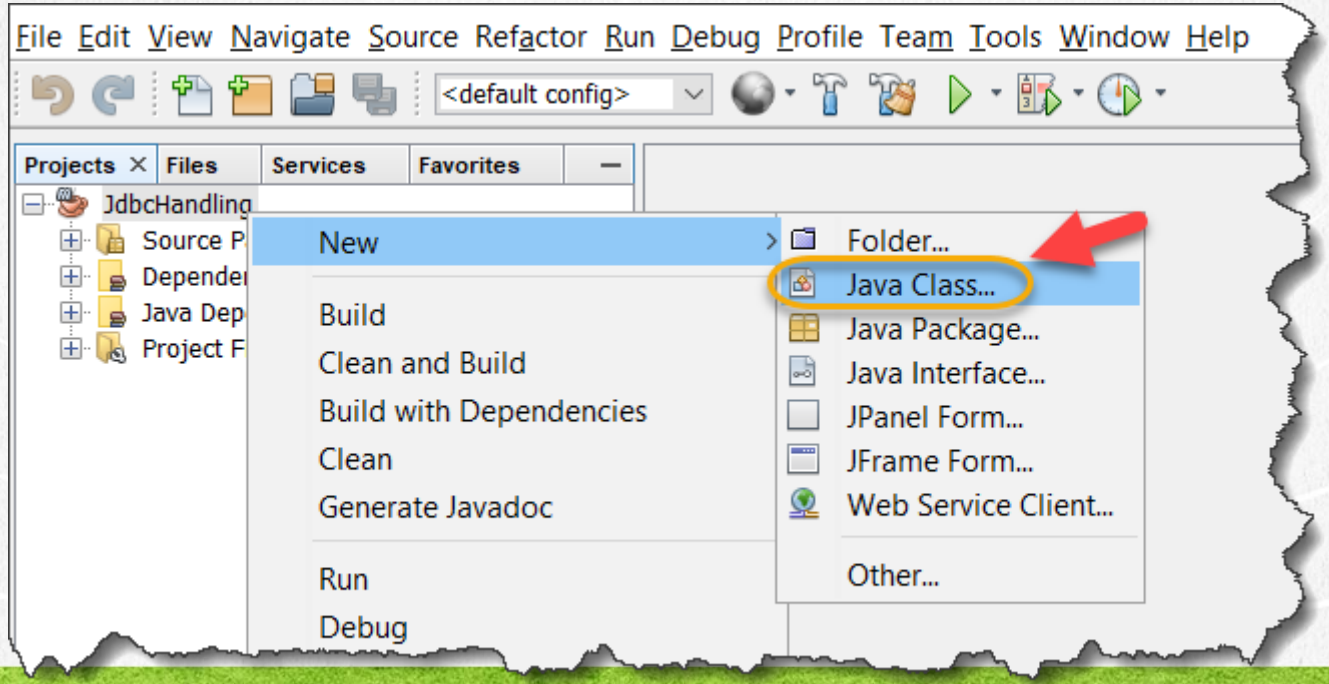
3. EXECUTE THE CLEAN & BUILD

Execute the Clean & Build option in order to download any library needed for this Project and maven:



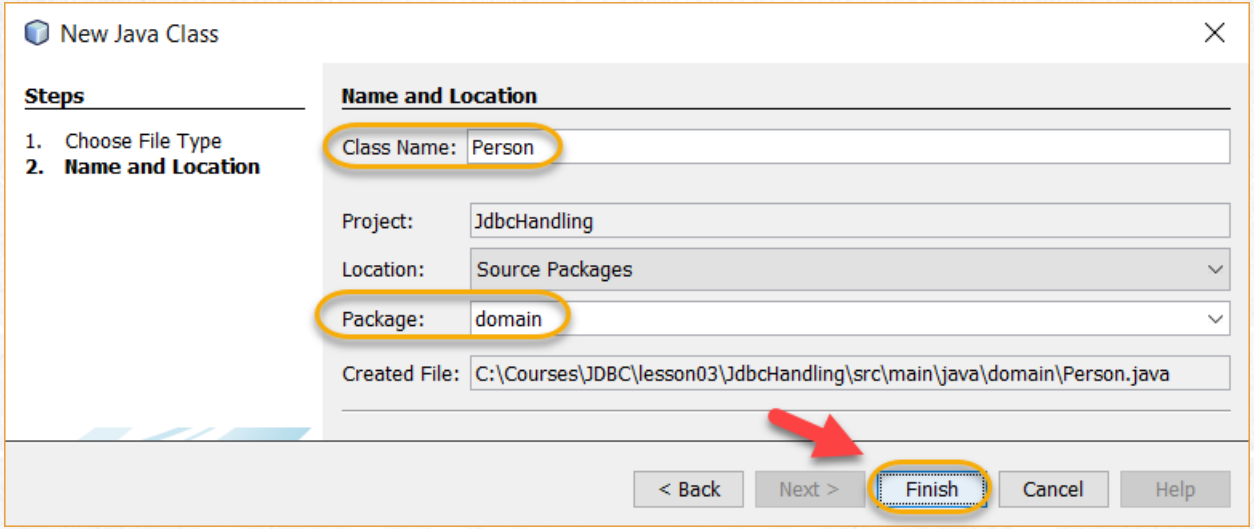
4. CREATE A NEW CLASS

Create a new class:



4. CREATE A NEW CLASS (CONT)

Create a new class:



5. MODIFY THE CODE

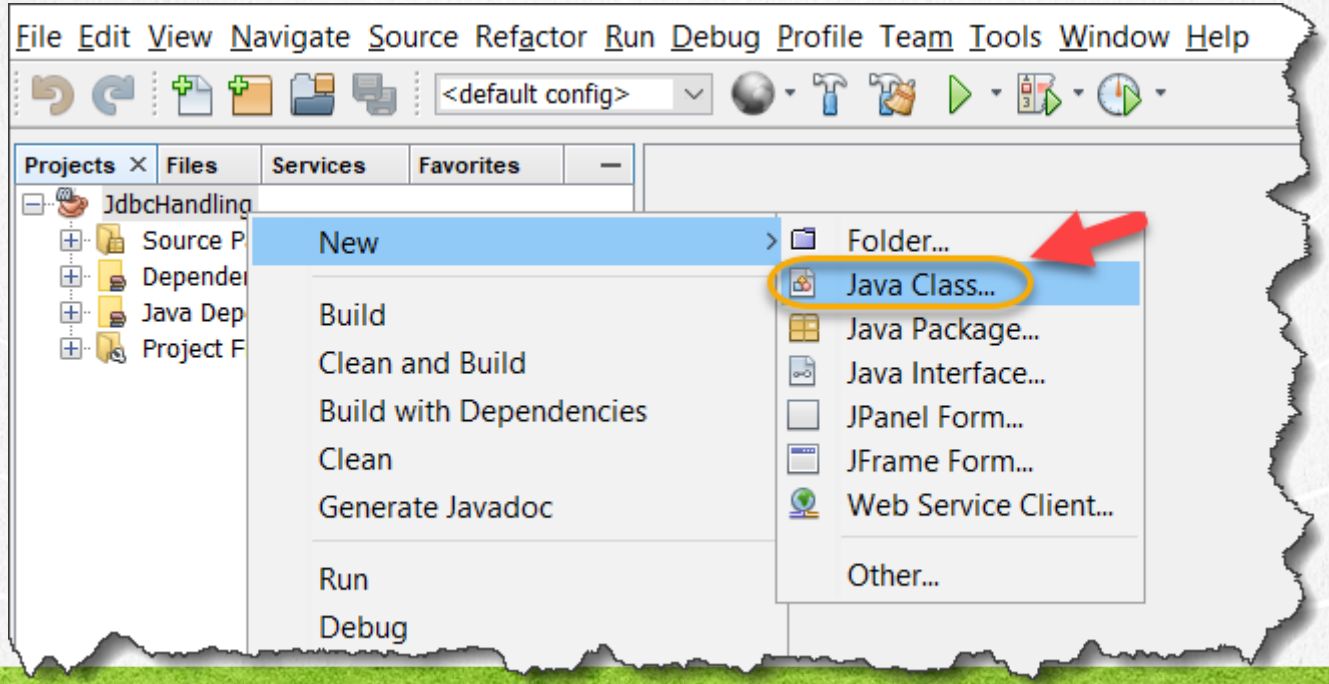
Person.java:

Click to download

```java
package domain;

public class Person {

    private int idPerson;
    private String name;

    public int getIdPerson() {
        return idPerson;
    }

    public void setIdPerson(int idPerson) {
        this.idPerson = idPerson;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

     @Override
    public String toString() {
        return "Person{" + "idPerson=" + idPerson + ", name=" + name + '}';
    }
}
```

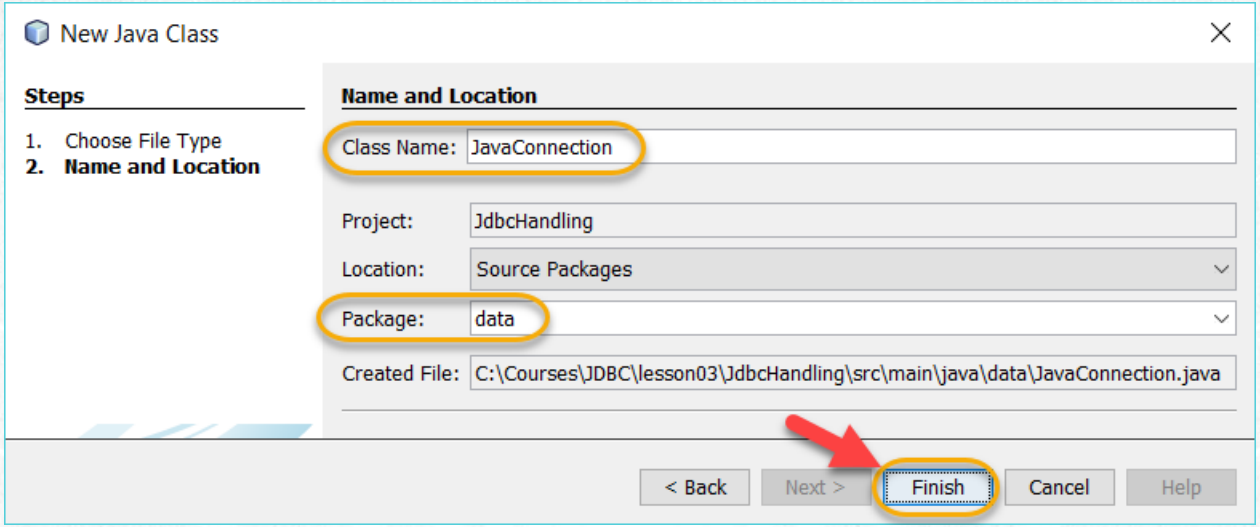
6. CREATE A NEW CLASS

Create a new class:



6. CREATE A NEW CLASS

Create a new class:



7. MODIFY THE CODE

JavaConnection.java:

Click to download

```java
package data;

import java.sql.*;

public class JavaConnection {

    private static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static final String JDBC_URL = "jdbc:mysql://localhost/test?useSSL=false";
    private static final String JDBC_USER = "root";
    private static final String JDBC_PASS = "admin";
    private static Driver driver;

    //So there are no problems when getting the connection of
    //concurrently, the word synchronized is used
    public static synchronized Connection getConnection() throws SQLException {
        if (driver == null) {
            try {
                Class jdbcDriverClass = Class.forName(JDBC_DRIVER);
                driver = (Driver) jdbcDriverClass.newInstance();
                DriverManager.registerDriver(driver);
            } catch (Exception e) {
                System.out.println("Failure to load the JDBC driver");
                e.printStackTrace(System.out);
            }
        }
        return DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASS);
    }
```

7. MODIFY THE CODE

JavaConnection.java:

Click to download

```
//Close the resultSet object
public static void close(ResultSet rs) {
    try {
        if (rs != null) {
            rs.close();
        }
    } catch (SQLException sqle) {
        sqle.printStackTrace(System.out);
    }
}

//Close the PrepareStatement object
public static void close(PreparedStatement stmt) {
    try {
        if (stmt != null) {
            stmt.close();
        }
    } catch (SQLException sqle) {
        sqle.printStackTrace(System.out);
    }
}
```

7. MODIFY THE CODE

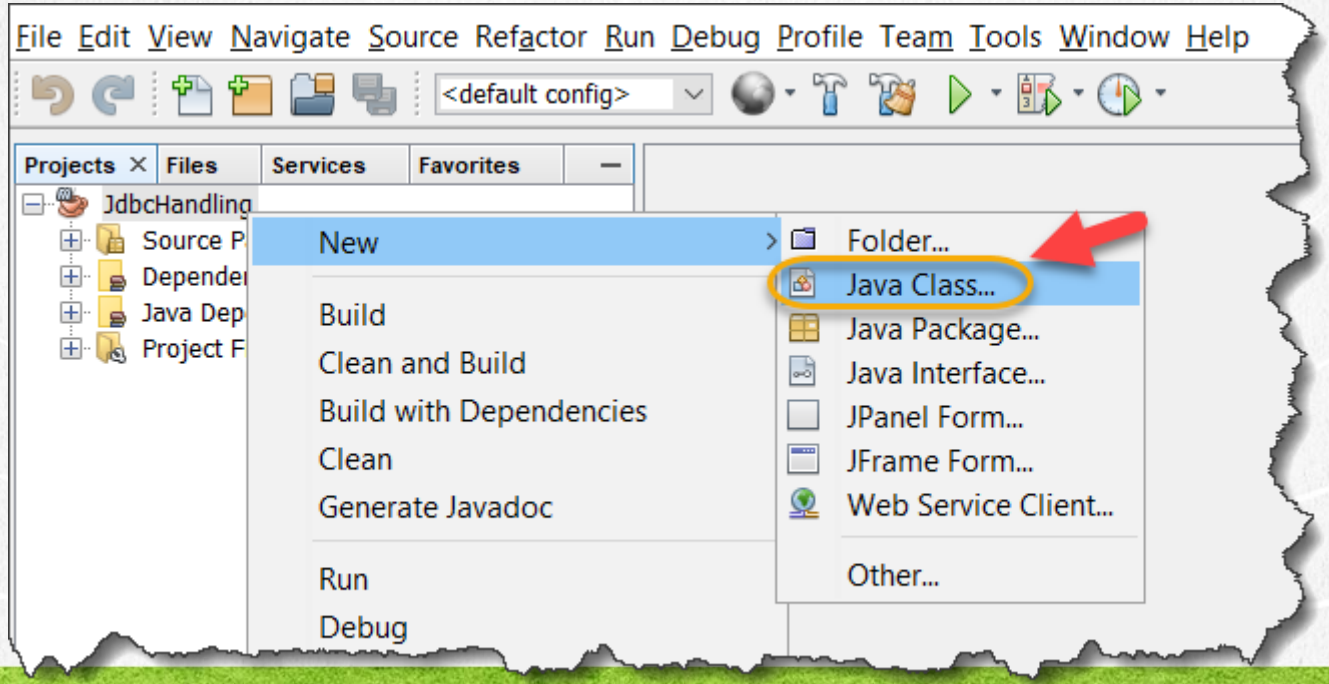
JavaConnection.java:

Click to download

```
    //Close the connection object
    public static void close(Connection conn) {
        try {
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException sqle) {
            sqle.printStackTrace(System.out);
        }
    }
}
```

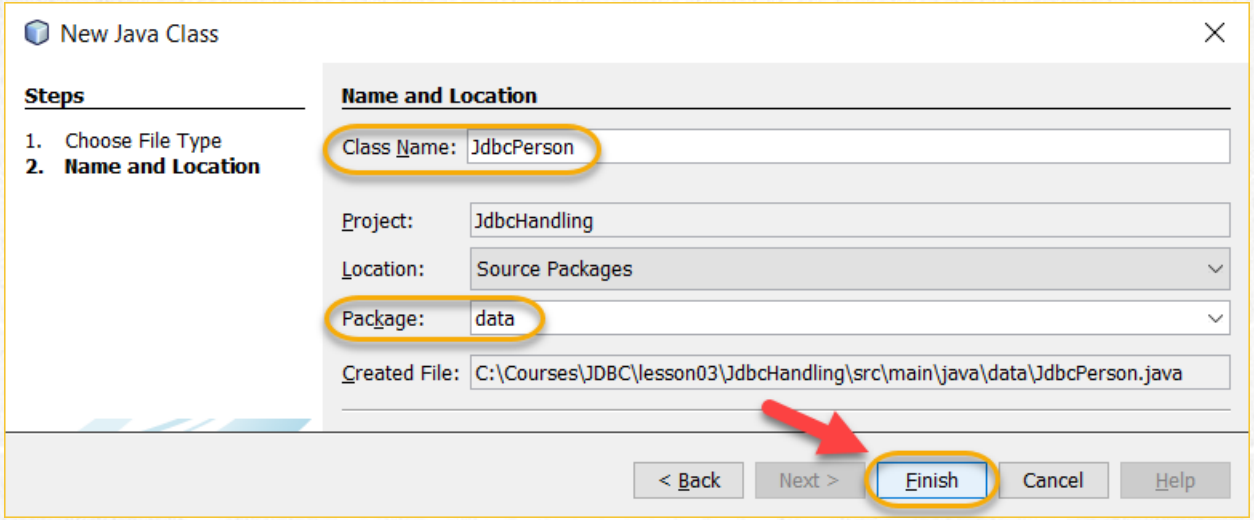
8. CREATE A NEW CLASS

Create a new class:



8. CREATE A NEW CLASS

Create a new class:



9. MODIFY THE CODE

JdbcPerson.java:

Click to download

```java
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 {

    // We rely on MySql's autoincrementable primary key
    // so the id_person field is omitted
    // A prepareStatement is used, so we can
    // use parameters (signs of ?)
    // which will later be replaced by the respective parameter
    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 = ?";
```

9. MODIFY THE CODE

JdbcPerson.java:

Click to download

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

/**
 * Method that inserts a record in the Person table
 *
 * @param name
 */
public int insert(String name) {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0; //affected rows
    try {
        conn = 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);
    } catch (SQLException e) {
        e.printStackTrace(System.out);
    } finally {
        JavaConnection.close(stmt);
        JavaConnection.close(conn);
    }
    return rows;
}
```

9. 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
 */
public int update(int idPerson, String name) {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = 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);
    } catch (SQLException e) {
        e.printStackTrace(System.out);
    } finally {
        JavaConnection.close(stmt);
        JavaConnection.close(conn);
    }
    return rows;
}
```

9. MODIFY THE CODE

JdbcPerson.java:

Click to download

```
/**
 * Method that deletes an existing record
 *
 * @param idPerson Primary key
 * @return int rows affected
 */
public int delete(int idPerson) {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = 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);
    } catch (SQLException e) {
        e.printStackTrace(System.out);
    } finally {
        JavaConnection.close(stmt);
        JavaConnection.close(conn);
    }
    return rows;
}
```

9. MODIFY THE CODE

JdbcPerson.java:

Click to download

```
/**
 * Method that returns the contents of the Person table
 *
 * @return list of person objects
 */
public List<Person> select() {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    Person persona = null;
    List<Person> personas = new ArrayList<>();
    try {
        conn = 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);
        }
```

9. MODIFY THE CODE

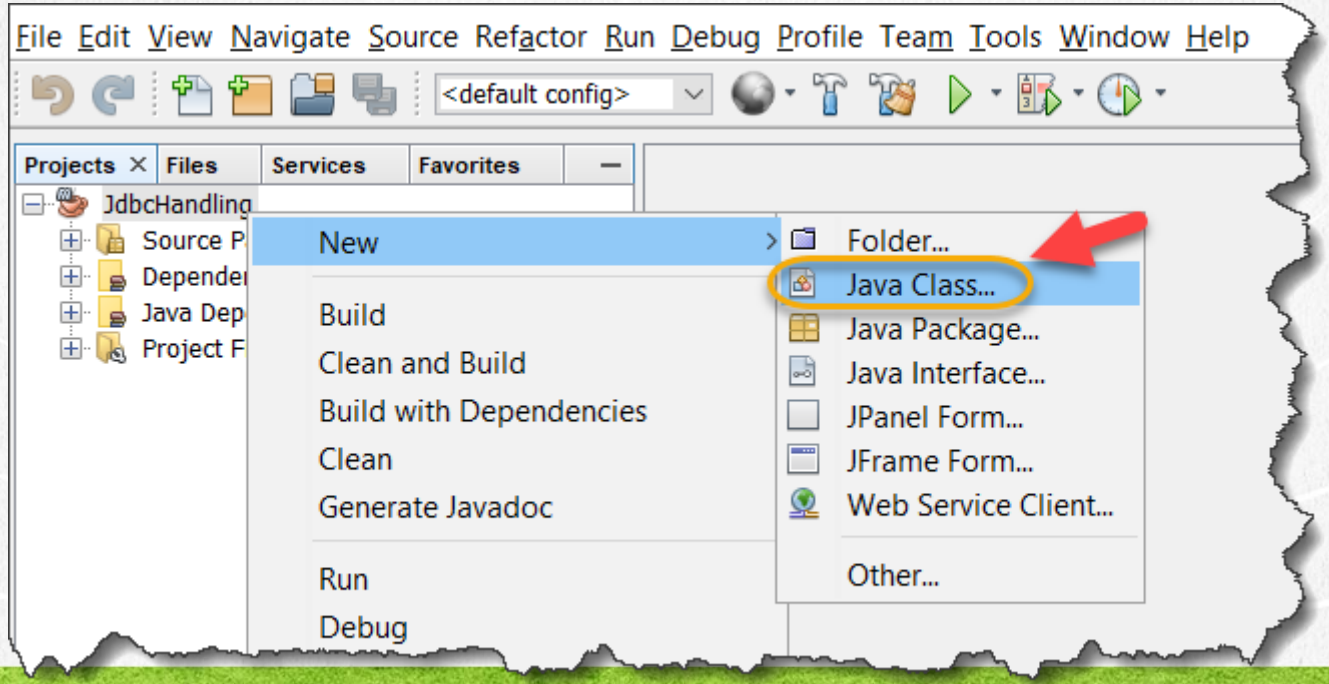
JdbcPerson.java:

Click to download

```
        } catch (SQLException e) {
            e.printStackTrace(System.out);
        } finally {
            JavaConnection.close(rs);
            JavaConnection.close(stmt);
            JavaConnection.close(conn);
        }
        return personas;
    }
}
```

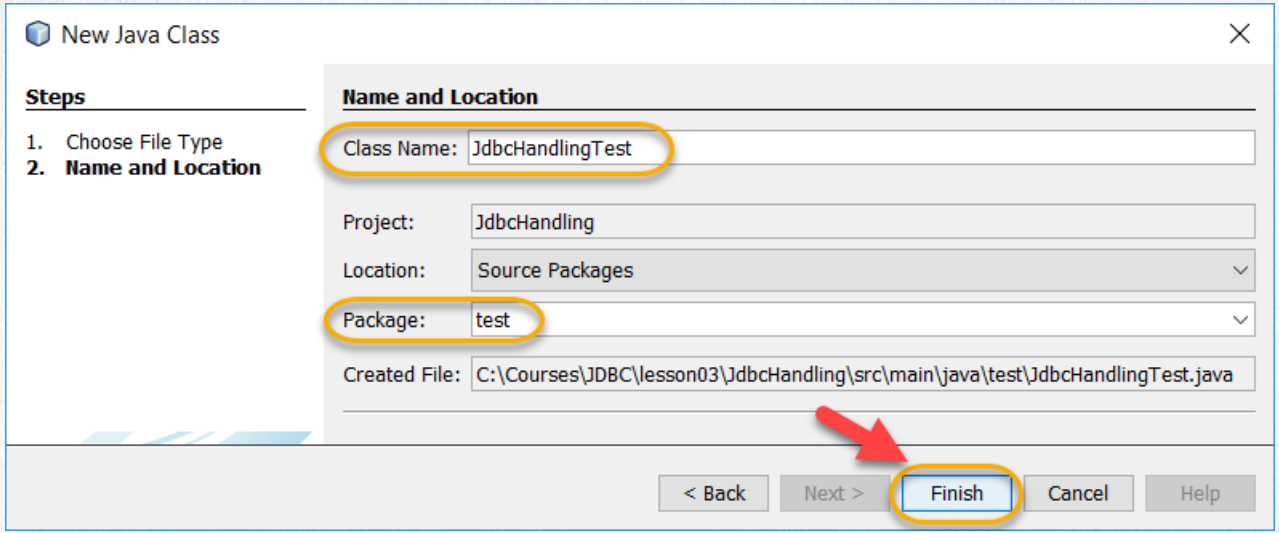
10. CREATE A NEW CLASS

Create a new class:



10. CREATE A NEW CLASS

Create a new class:



11. MODIFY THE CODE

JdbcHandlingTest.java:

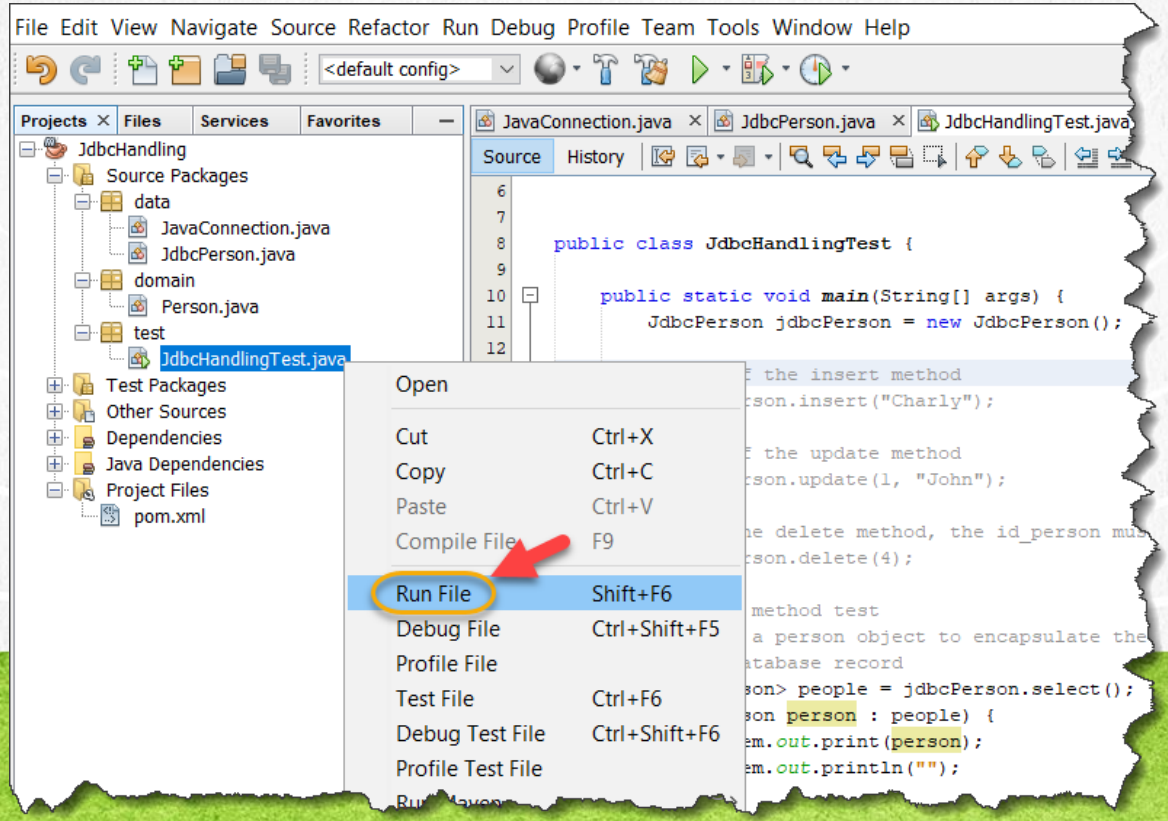
Click to download

```java
package test;

import data.JdbcPerson;
import domain.Person;
import java.util.List;

public class JdbcHandlingTest {

    public static void main(String[] args) {
        JdbcPerson jdbcPerson = new JdbcPerson();

        //Test of the insert method
        //jdbcPerson.insert("Charly");

        //Test of the update method
        //jdbcPerson.update(1, "John");

        //Test the delete method, the id_person must exist in the database
        //jdbcPerson.delete(4);

        //Select method test
        //Use of a person object to encapsulate the information
        //of a database record
        List<Person> people = jdbcPerson.select();
        for (Person person : people) {
            System.out.print(person);
            System.out.println("");
        }
    }
}
```

Go running each case and check that it works correctly in the database or making a selection

12. EXECUTE THE PROJECT

We execute our project. We give right click -> Run:



12. 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)

```
Running NetBeans Compile On Save execution. Phase execution is skipped and output
Scanning for projects...

------------------------------------------------------------------------
Building JdbcHandling 1
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ JdbcHandling ---
Person{idPerson=1, name=John}
Person{idPerson=2, name=Katty}
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 0.860s
```

EXERCISE CONCLUSION

- With this exercise we have put into practice several concepts, such as the basic separation of responsibilities in a connection class, another kind of Person that represents a record in the database, and another JdbcPerson class that contains the basic operations to manipulate this information in the table people of the database, such as insert, update, delete and select.

ONLINE COURSE

JAVA WITH JDBC

By: Eng. Ubaldo Acosta