-- phpMyAdmin SQL Dump
-- version 4.9.1
-- https://www.phpmyadmin.net/
--
-- Servidor: localhost
-- Tiempo de generaciÃ³n: 19-10-2020 a las 12:25:03
-- VersiÃ³n del servidor: 8.0.17
-- VersiÃ³n de PHP: 7.3.10

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `csi20[redacted]`
--

Falta la instrucción que crea la base de datos -0.5

-- --

--
-- Estructura de tabla para la tabla `[redacted]`
--

-0.5

No se indica que es clave primaria

CREATE TABLE `[redacted]` (
  `[redacted]` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `[redacted]` int(200) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

Déjalo en 11.

--
-- Volcado de datos para la tabla `[redacted]`
--

INSERT INTO [redacted] VALUES
[redacted]

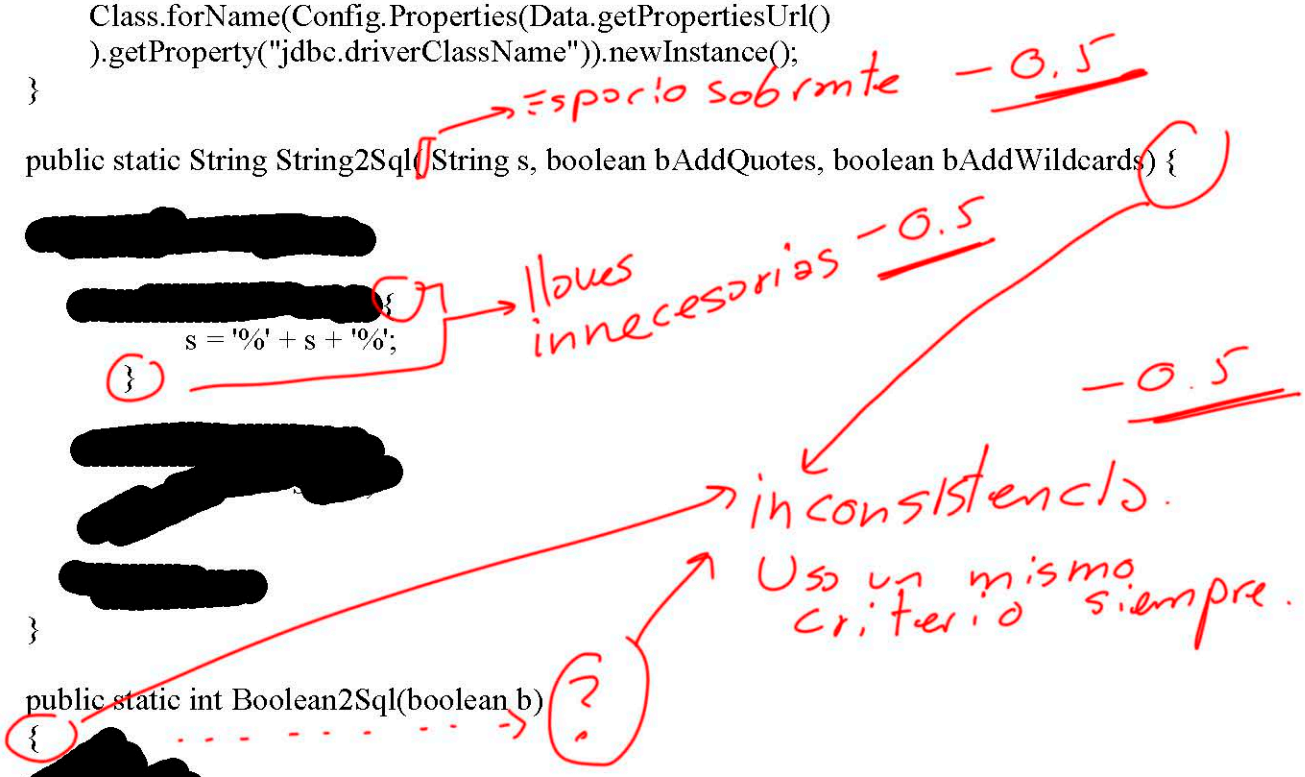
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

jdbc.driverClassName=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://localhost/csi20[redacted]?useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC
jdbc.username=[redacted]
jdbc.password=[redacted]

```
package es.uca.gii.csi20[redacted].util;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class Config {

	public static Properties Properties(String sFile) throws IOException {

		InputStream inputStream = null;

		try {
		   inputStream = new FileInputStream(sFile);
		   Properties result = new Properties();
		   result.load(inputStream);
		   return result;
		}
		finally { if (inputStream != null) inputStream.close(); }
	}

}
```

```
package es.uca.gii.csi20[redacted].data;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.Properties;

import es.uca.gii.csi20[redacted].util.Config;

public class Data {

    public static String getPropertiesUrl() { return "./db.properties"; }

    public static Connection Connection() throws Exception {

        try {
            Properties properties = Config.Properties(getPropertiesUrl());
            return DriverManager.getConnection(
                properties.getProperty("jdbc.url"),
                properties.getProperty("jdbc.username"),
                properties.getProperty("jdbc.password"));
        }
        catch (Exception ee) { throw ee; }
        }

    public static void LoadDriver()
        throws InstantiationException, IllegalAccessException,
        ClassNotFoundException, IOException {

            Class.forName(Config.Properties(Data.getPropertiesUrl()
            ).getProperty("jdbc.driverClassName")).newInstance();
    }

    public static String String2Sql(String s, boolean bAddQuotes, boolean bAddWildcards) {
[redacted]
                {
                    s = '%' + s + '%';
                }
[redacted]
    }

    public static int Boolean2Sql(boolean b)
    {
[redacted]
    }
}
```



```
package es.uca.gii.csi20.[redacted].test;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

import es.uca.gii.csi20[redacted].data.Data;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;

class DataTest {

	@BeforeAll
	static void setUpBeforeClass() throws Exception {
		Data.LoadDriver();
	}

	@Test
	void String2SqlTest() {
		assertEquals("hola", Data.String2Sql("hola", false, false));
		assertEquals("\'hola\'", Data.String2Sql("hola", true, false));
		assertEquals("%hola%", Data.String2Sql("hola", false, true));
		assertEquals("\'%hola%\'", Data.String2Sql("hola", true, true));
		assertEquals("O\'\'Connell", Data.String2Sql("O'Connell", false, false));
		assertEquals("\'O\"Connell\'", Data.String2Sql("O'Connell", true, false));
		assertEquals("%\"Smith \"%", Data.String2Sql("\'Smith '", false, true));
		assertEquals("\'\"Smith \"\'", Data.String2Sql("\'Smith '", true, false));
		assertEquals("\'%\"Smith \"%\'", Data.String2Sql("\'Smith '", true, true));
	}

	@Test
	void Boolean2SqlTest() {
		assertEquals(1, Data.Boolean2Sql(true));
		assertEquals(0, Data.Boolean2Sql(false));
	}

	@Test
	void testTableAccess() throws Exception {

		Connection con = null;
		ResultSet rs = null;
		int columnCount;
		int elementsCount;
		try {
			con = Data.Connection();
			rs = con.createStatement().executeQuery("SELECT COUNT(*) FROM[redacted]");

			rs.next();
			elementsCount = rs.getInt(1);

			assertEquals(2, elementsCount);
		}
```

0 → salto de líneas innecesario −0.5

A partir de la práctica 2, se penalizará no usar notación húngara.

Uso salto de líneas −0.5

0 → salto de líneas innecesario −0.5

```
        catch(SQLException ee) { throw ee; }

    try {
        con = Data.Connection();
        rs = con.createStatement().executeQuery("SELECT * FROM [redacted];");

        int i = 0;
        while(rs.next()) {
            System.out.println(rs.getString("[redacted]") + " " + rs.getString("[redacted]"));
            i++;
        }

        columnCount = rs.getMetaData().getColumnCount();

        assertEquals(2, i);
      assertEquals(3, columnCount);
    }
    catch(SQLException ee) { throw ee; }
    finally {
        if(rs != null) rs.close();
        if( con != null) con.close();
    }
  }
}
```

innecesario -0,5

conexión abierta sin cerrar lo anterior -0,5

no se ha cerrado el ResultSet -0,5

Dale un sentido cronológico código -0,5