

Calidad de los Sistemas Informáticos, 2020-2021

Práctica 0 **Instalación y configuración**

- ▶ Aplicaciones requeridas
- ▶ Recursos requeridos y versiones
- ▶ Instalación
- ▶ Prueba de funcionamiento
- ▶ Buenas prácticas

Aplicaciones requeridas

Entorno Eclipse

- ▶ Lenguaje de programación Java
- ▶ Pruebas unitarias
- ▶ Cobertura de código
- ▶ IDE para interfaces de usuario



Paquete de servicios AppServer (o alternativa)

- ▶ Sistema de gestión de bases de datos MySQL
- ▶ Aplicación web phpMyAdmin



Recursos requeridos y versiones



▶ Eclipse IDE for Java Developers (2020-09-R)



▶ AppServer 9.3.0 (o similar) para servicio MySQL y phpMyAdmin



▶ Conector de MySQL 8.0.11 para Java



▶ WindowBuilder 1.9.4 para IDE de interfaces en Swing



▶ JUnit 5 para pruebas unitarias (incluido en Eclipse 2020-09)



▶ EclEmma 3.1.3 para cobertura de código (incluido en Eclipse 2020-09)

Instalación (1 de 4)

Eclipse IDE for Java Developers (2020-09-R)

<http://www.eclipse.org/downloads/eclipse-packages/>

Instalación en el equipo

1. Instalar JDK8 de la web de Oracle
2. Descargar la misma versión de Eclipse (32 ó 64) que la versión Java del equipo
3. Descomprimir y copiar en una carpeta
4. Ejecutar Eclipse.exe (o similar en otros sistemas)

The screenshot shows the Eclipse Foundation website's download page for Eclipse IDE 2020-09 R Packages. The page is titled "Eclipse IDE 2020-09 R Packages" and features two main sections for downloading the IDE. The first section, "Eclipse IDE for Java Developers", is circled in red. It lists the package size as 197 MB and shows 205,587 downloads. It provides download links for Windows 64-bit, Mac Cocoa 64-bit, and Linux 64-bit. The second section, "Eclipse IDE for Enterprise Java Developers", lists the package size as 382 MB and shows 154,394 downloads. It also provides download links for Windows 64-bit, Mac Cocoa 64-bit, and Linux 64-bit. The page includes a navigation bar at the top with links to Projects, Working Groups, Members, and More. A sidebar on the right contains a "Download" button and a "RELATED LINKS" section with links to Compare & Combine Packages, New and Noteworthy, Install Guide, and Documentation.

Instalación (2 de 4)

AppServ 9.3.0 (sólo Windows)

<https://www.appserv.org/download/>

Instalación en el equipo

- ▶ Instalar al menos Apache, MySQL y phpMyAdmin
- ▶ Servidor: myserver.com (o cualquier otro)
- ▶ Puerto Apache: 89 (o cualquier otro)
- ▶ Puerto MySQL: 3306 (o cualquier otro)

Prueba de acceso (hacer en un navegador)

- ▶ `http://localhost{:puerto}/`
- ▶ `http://localhost{:puerto}/phpmyadmin`

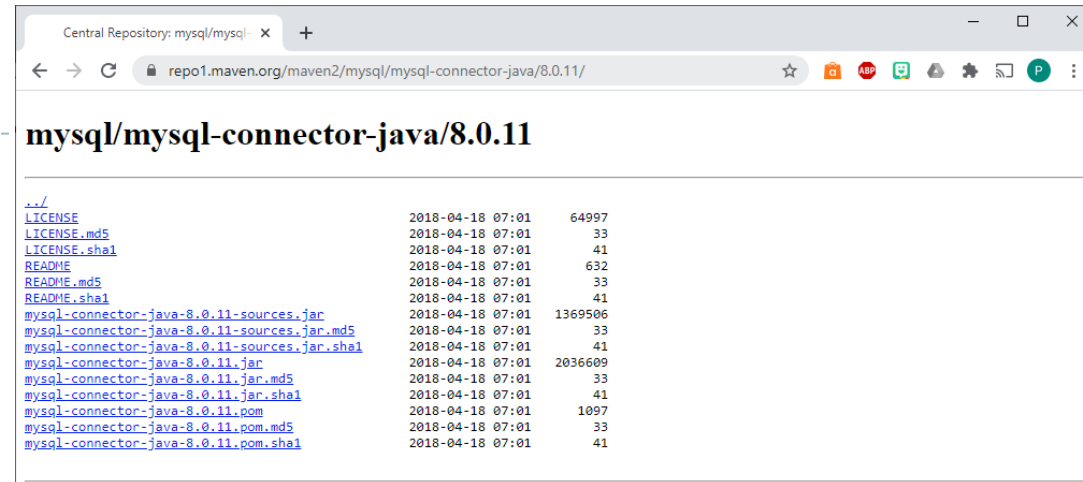
usuario: *root*, contraseña: *{:la de instalación}*



Otras alternativas como WampServer son válidas. Consultar en caso de no encontrarlas para Mac y/o Linux.

Conector MySQL para Java

<https://repo1.maven.org/maven2/mysql/mysql-connector-java/8.0.11/>

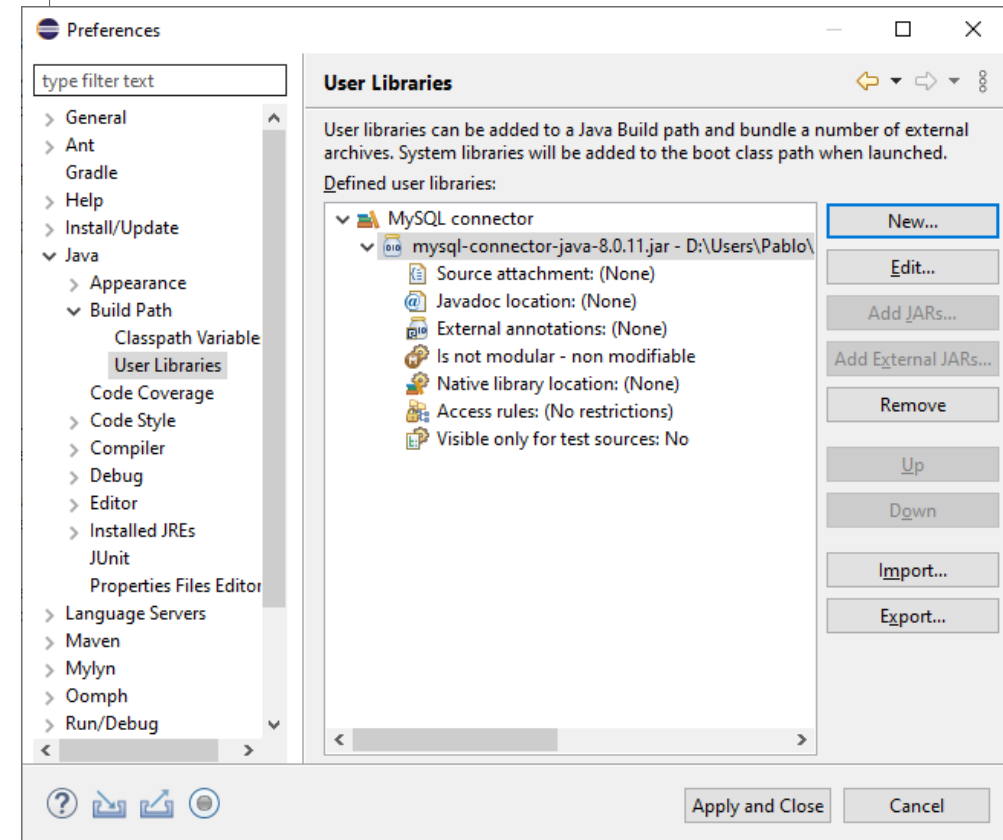


Extracción en el equipo

1. Creación —donde se desee— de carpeta `javaliib/csi` para las bibliotecas externas
2. Descompresión del ZIP del enlace
3. Copia del `.JAR` existente en el recién creado `javaliib/csi`

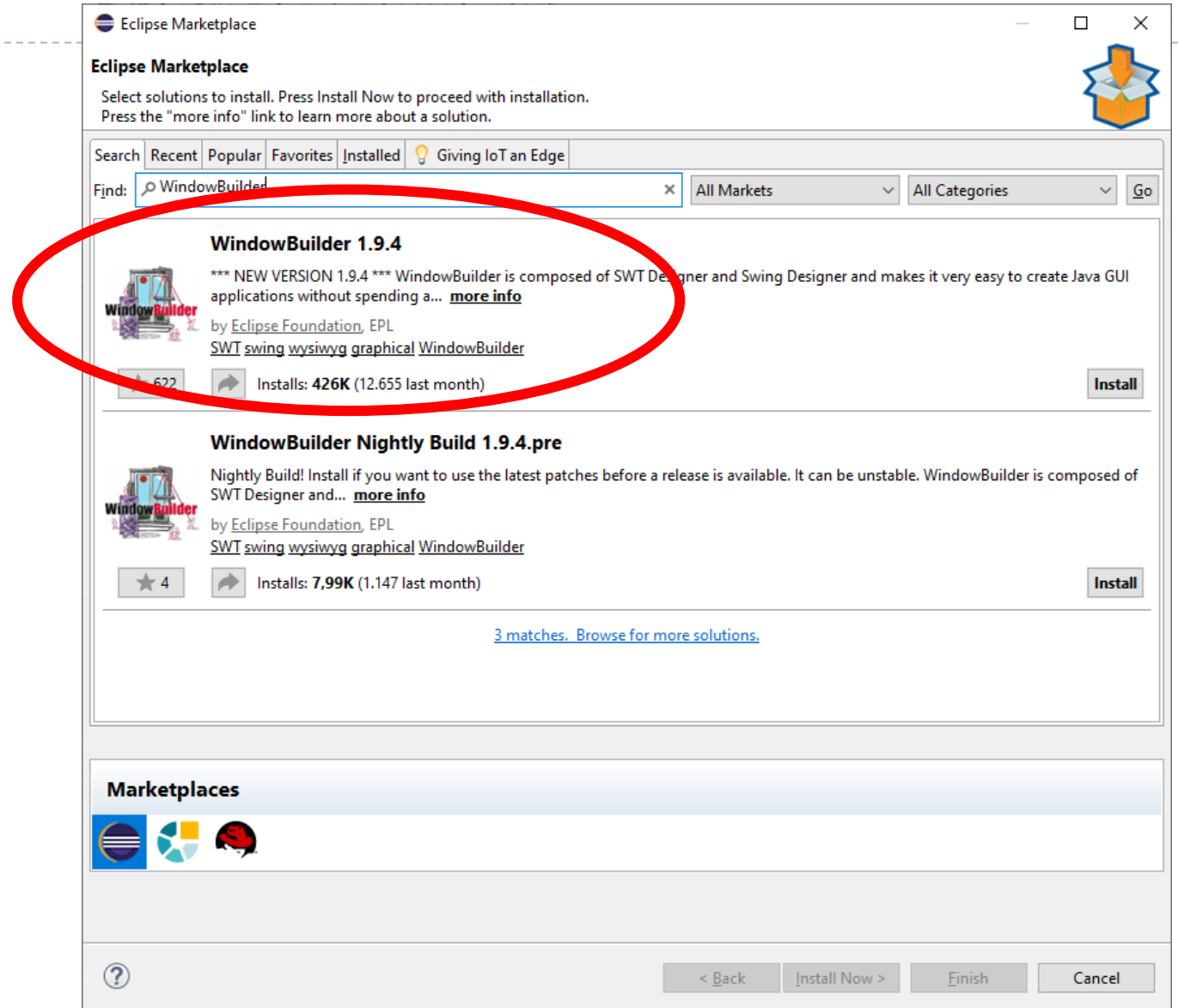
Configuración en Eclipse

1. Ir a *Window > Preferences*
2. Ir a *Java > Build Path > User Libraries*
3. *New...*
4. Escribir *MySQL connector*
5. *Add External JARs...*
6. Seleccionar el `.JAR` de `javaliib/csi`
7. *Apply and Close*



WindowBuilder 1.9.4

1. Ir a *Help > Eclipse Marketplace...*
2. Buscar *WindowBuilder*
3. Instalar *WindowBuilder 1.9.4*
4. Reiniciar Eclipse
5. Comprobar que en
File > New... > Other...
aparece la carpeta
WindowBuilder



Prueba de funcionamiento (1 de 9)

1. Crear un nuevo proyecto de Java (*File > New > Java Project*)
 - ▶ Project name: `es.uca.gii.csi20.check`
2. Incluir biblioteca de acceso a MySQL creada antes, marcando (*Next > Libraries > Add Library... > User Library > MySQL connector*)
3. Crear un archivo de acceso a la base de datos
 - ▶ Botón derecho sobre el nombre del proyecto, *New > File*
 - ▶ File name: `db.properties`
 - ▶ Incluir contenido de la diapositiva siguiente

db.properties

```
jdbc.driverClassName=com.mysql.cj.jdbc.Driver
```

```
jdbc.url=jdbc:mysql://localhost/mysql?useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC
```

```
jdbc.username={:usuario MySQL}
```

```
jdbc.password={:contraseña}
```

Para versiones previas de MySQL, el driver es `com.mysql.jdbc.Driver`.

Nombre la base de datos –para la prueba se usará *mysql*, creada por defecto por MySQL–.

Si el puerto no es 3306, debe indicarse tras *localhost*.

Debe modificarse el usuario y la contraseña por las correctas, sin incluirse las llaves ni comillas.

4. Crear paquete donde irán las clases generales de utilidades
 - ▶ Botón derecho sobre `src`, *New > Package*
 - ▶ Name: `es.uca.gii.csi20.check.util`

5. Crear clase de acceso a la configuración
 - ▶ Botón derecho sobre `es.uca.gii.csi20.check.util`, *New > Class*
 - ▶ Name: `Config`
 - ▶ Incluir código fuente de la imagen de la diapositiva siguiente

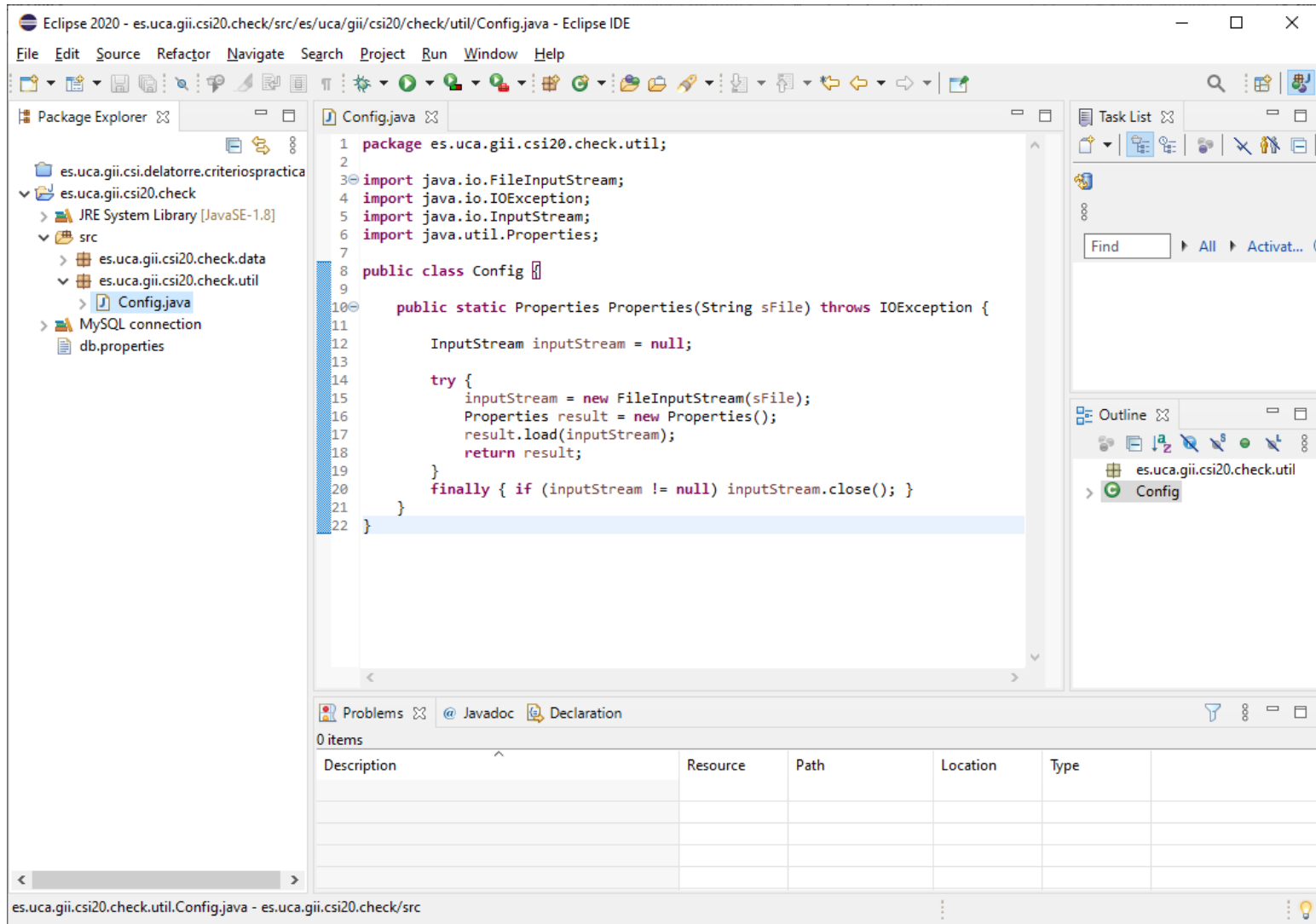
Prueba de funcionamiento (4 de 9)

util/Config.java

La clase `Config` proporciona un método estático `Properties` que, a partir de un archivo de propiedades —como `db.properties`—, devuelve una instancia de la clase `Properties` de Java.

A su vez, la clase `Properties` de Java. proporciona el método `getProperty` para acceder al valor de la propiedad correspondiente. Ejemplo de uso:

```
Properties properties =  
    Config.Properties("db.properties");  
String sUrl =  
    properties.getProperty("jdbc.url");
```



6. Crear paquete donde irán las clases de acceso a datos
 - ▶ Botón derecho sobre `src`, *New > Package*
 - ▶ Name: `es.uca.gii.csi20.check.data`

7. Crear clase para acceso a datos
 - ▶ Botón derecho sobre `es.uca.gii.csi20.check.data`, *New > Class*
 - ▶ Name: `Data`
 - ▶ Incluir código fuente de la imagen de la diapositiva siguiente

data/Data.java

Data aporta los métodos estáticos:

- a) **LoadDrive**, que debe invocarse una sola vez y que carga el drive de conexión correspondiente, a partir del archivo de propiedades.
- b) **Connection()**, que devuelve una conexión a la base de datos – instancia de la clase **Connection** de Java–, a partir del archivo de propiedades.
- c) **getPropertiesUrl()**, que devuelve la URL del archivo de propiedades.

```
Data.LoadDriver();
```

```
Connection con = Data.Connection();
```

```
Eclipse 2020 - es.uca.gii.csi20.check/src/es/uca/gii/csi20/check/data/Data.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
es.uca.gii.csi20.check
  es.uca.gii.csi20.check
    JRE System Library [JavaSE-1.8]
    src
      es.uca.gii.csi20.check.data
        Data.java
      es.uca.gii.csi20.check.util
        Config.java
      MySQL connection
        db.properties

Config.java
Data.java

1 package es.uca.gii.csi20.check.data;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.sql.DriverManager;
6 import java.util.Properties;
7
8 import es.uca.gii.csi20.check.util.Config;
9
10 public class Data {
11
12     public static String getPropertiesUrl() { return "./db.properties"; }
13
14     public static Connection Connection() throws Exception {
15
16         try {
17
18             Properties properties = Config.Properties(getPropertiesUrl());
19
20             return DriverManager.getConnection(
21                 properties.getProperty("jdbc.url"),
22                 properties.getProperty("jdbc.username"),
23                 properties.getProperty("jdbc.password"));
24         }
25         catch (Exception ee) { throw ee; }
26     }
27
28     public static void LoadDriver() throws InstantiationException, IllegalAccessException,
29         ClassNotFoundException, IOException {
30
31         Class.forName(Config.Properties(Data.getPropertiesUrl()).getProperty(
32             "jdbc.driverClassName")).newInstance();
33     }
34 }
```

Task List

Find All Acti...

Outline

es.uca.gii.csi20.check.d

Data

- getPropertiesUrl(): S
- Connection(): Conn
- LoadDriver(): void

Problems @ Javadoc Declaration

0 items

Description	Resource	Path	Location	Type
	Writable	Smart Insert	34 : 2 : 1061	

8. Crear paquete donde irán las clases para pruebas unitarias
 - ▶ Botón derecho sobre `src`, *New > Package*
 - ▶ Name: `es.uca.gii.csi20.check.test`

9. Crear clase para pruebas de la clase *Data*
 - ▶ Botón derecho sobre `es.uca.gii.csi20.check.test`, *New > JUnit Test Case*
 - ▶ Name: `DataTest` (es decir, test para la clase `Data`)
 - ▶ Seleccionar la inclusión del método `setUpBeforeClass`
 - ▶ Confirmar la inclusión de JUnit 5 en las bibliotecas
 - ▶ Incluir código fuente de la imagen de la diapositiva siguiente

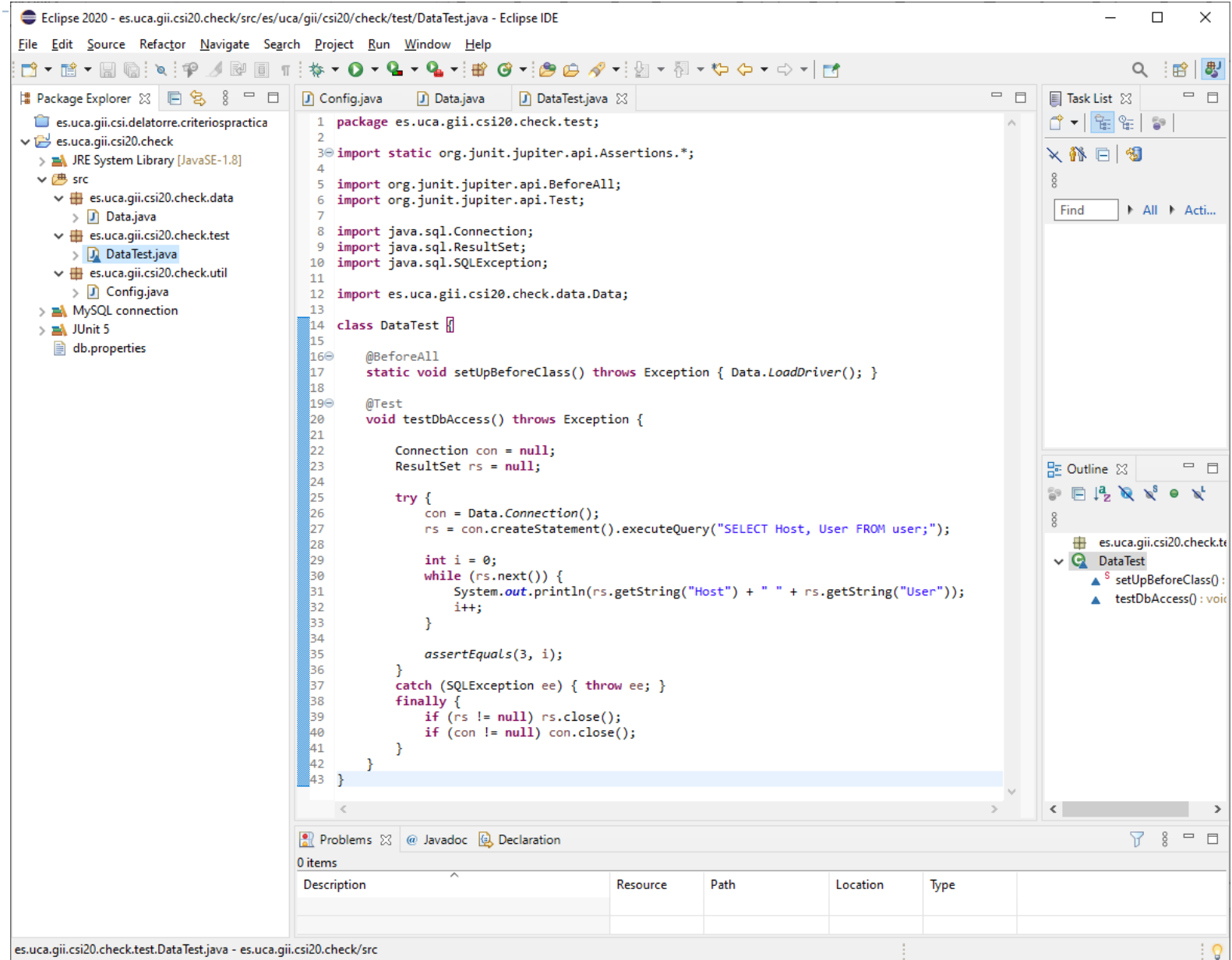
Prueba de funcionamiento (8 de 9)

test/DataTest.java

DataTest prueba el funcionamiento de la clase **Data**:

a) **setUpBeforeClass** se ejecuta una vez al principio y carga el driver **MySQL**.

b) **testDbAccess()**, que realiza una consulta a **MySQL**, imprime los registros —ilustrativamente, ya que no es necesario en una prueba unitaria— y comprueba vía **assertEquals** que el número de elementos devuelve es el que se espera —esto sí se requiere—.



```
1 package es.uca.gii.csi20.check.test;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 import org.junit.jupiter.api.BeforeAll;
6 import org.junit.jupiter.api.Test;
7
8 import java.sql.Connection;
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 import es.uca.gii.csi20.check.data.Data;
13
14 class DataTest {
15
16     @BeforeAll
17     static void setUpBeforeClass() throws Exception { Data.LoadDriver(); }
18
19     @Test
20     void testDbAccess() throws Exception {
21
22         Connection con = null;
23         ResultSet rs = null;
24
25         try {
26             con = Data.Connection();
27             rs = con.createStatement().executeQuery("SELECT Host, User FROM user;");
28
29             int i = 0;
30             while (rs.next()) {
31                 System.out.println(rs.getString("Host") + " " + rs.getString("User"));
32                 i++;
33             }
34
35             assertEquals(3, i);
36         }
37         catch (SQLException ee) { throw ee; }
38         finally {
39             if (rs != null) rs.close();
40             if (con != null) con.close();
41         }
42     }
43 }
```

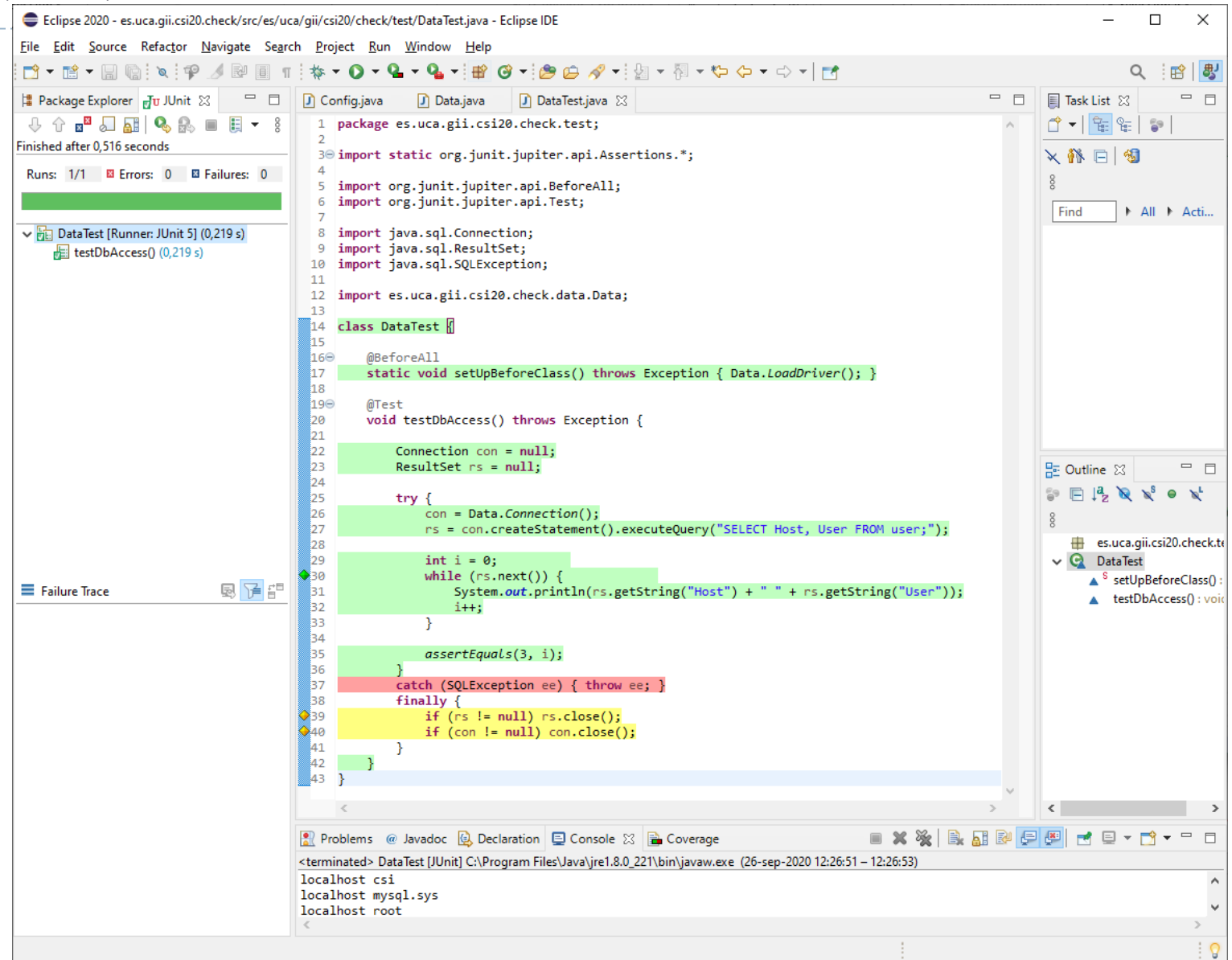
Problems 0 items

Description	Resource	Path	Location	Type
-------------	----------	------	----------	------

Ejecución del test

Ejecutar como JUnit Test con cobertura de código (botón derecho sobre el nombre del proyecto, Coverage As > JUnit Test). Una vez ejecutado:

- A la izquierda, **JUnit mostrará el resultado de la prueba**, verde si ha pasado el `assertEquals` y rojo si no.
- En la pestaña inferior Coverage aparecerá el **código total que se ha ejecutado**. El árbol permite expansión y muestra cuánto se ha ejecutado de cada archivo. Pinchando dos veces en un archivo se accede a éste.
- En la pestaña inferior Console se mostrarán los **resultados impresos con `System.out.println`** dentro del bucle, que son todos los registros de la tabla `User`.



En caso de no pasar el test de JUnit, debe actuarse poniendo en el primer parámetro de `assertEquals` lo que realmente se espera.

MUY
IMPORTANTE

Buenas prácticas para todas las prácticas de la asignatura

- ▶ Estudiar el código del método `testDbAccess()` de *DataTest.java*. Se usa el **mismo esquema en todas las consultas de selección**.
- ▶ Todas las conexiones `Connection` y cursores `ResultSet` creados en un método `no` recibidos como parámetros **deben ser cerrados al final** del método, haya habido error en éste o no. El modo de asegurarse es hacerlo en el `finally` de un `try`. Primero se cierran los cursores y, luego, la conexión.
- ▶ El objetivo de los métodos de prueba no es imprimir. Puede imprimirse para comprobar visualmente algunos datos en consola, pero no es necesario. El objetivo de los métodos de prueba es tener **descritos adecuadamente sus correspondientes asserts** para que JUnit pueda validarlos, comparando lo que se espera `primer parámetro` con lo que recibe `segundo parámetro`.
- ▶ La cobertura indica qué porcentaje de código se ejecuta: **un porcentaje de código bajo (< 75%) no justificado puede significar código innecesario** en el conjunto de métodos que se está probando.