

# Documentación de prácticas 0 y 1



**Trabajo realizado por:**

Carlos Guillén Moreno

**Para el profesor:** Andrés Caro Lindo

## Contenido

Introducción .....	3
Preparación del entorno en el equipo .....	3
Oracle Database.....	3
SQL Developer .....	6
Creación de tablas.....	9
Inserción de datos en las tablas .....	11
Borrado de tablas.....	15
Eclipse.....	16
JDBC para Oracle.....	18
Importación de librerías.....	21
PreparedStatement/CreateStatement .....	22
Métodos de la clase .....	23
dbConectar().....	23
dbDesconectar().....	25
dbObtenerPersonajes1() .....	26
dbObtenerPersonajes2() .....	27
Ejercicios propuestos .....	28
4.a) dbConsultarPersonajes() .....	28
4.a) dbConsultarClases().....	29
4.b) dbInsertarClases() .....	30
4.c) dbModificarClases().....	31
4.d) dbBorrarClases() .....	32
Salida por pantalla .....	33



## Introducción

Se procede a la instalación del software necesario para implementar la práctica 1. Para ello se tendrá que seguir detalladamente los pasos que se explican a continuación.

## Preparación del entorno en el equipo

El usuario deberá instalar el siguiente software:

- Oracle DataBase 11g Express Edition y SQL Developer
- Eclipse
- JDBC para Oracle

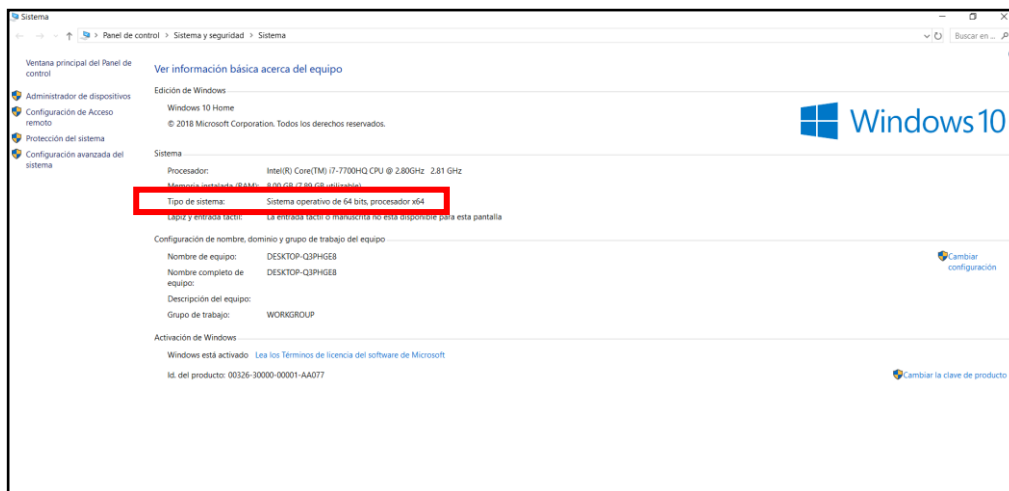
### Oracle Database

Una base de datos Oracle es una acumulación de información considerada como una unidad. El motivo de una base de datos es almacenar y recuperar datos relacionados. Oracle Database es la base de datos principal que se está utilizando para el procesamiento de grandes empresas, debido a su adaptabilidad y enfoque práctico para el manejo de datos y aplicaciones de software.

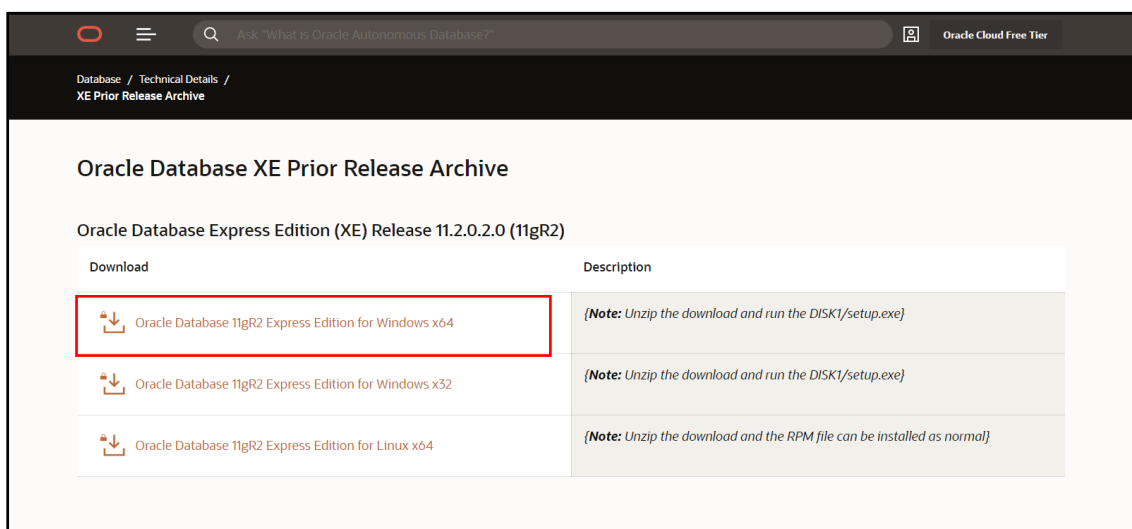
En nuestro caso hemos optado por la instalación de la versión **Oracle Database 11g** para descargar. Se descarga desde la página oficial de Oracle, en la versión de descargas. Se adjunta el link de la versión:

<https://www.oracle.com/database/technologies/xe-prior-releases.html>

Nos aparecerán varios enlaces, cada uno dependiendo de los requisitos del equipo de cada usuario, en mi caso me interesa la versión de Windows x64. Para saber la arquitectura de tu dispositivo, desde propiedades del equipo se despliega una ventana donde te muestra cuál es tu versión:

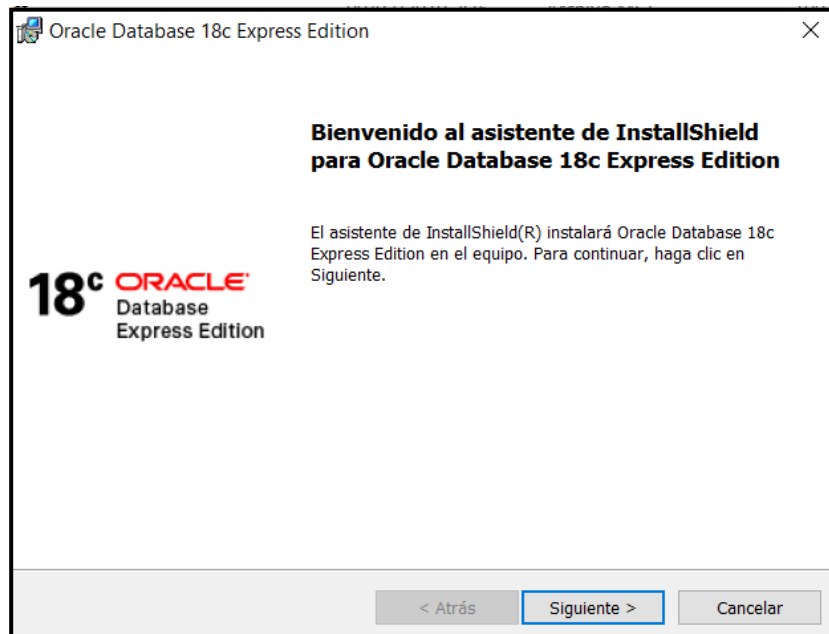


Una vez comprobado la versión que tenemos, procedemos a la descarga de la versión de Oracle que nos corresponde:

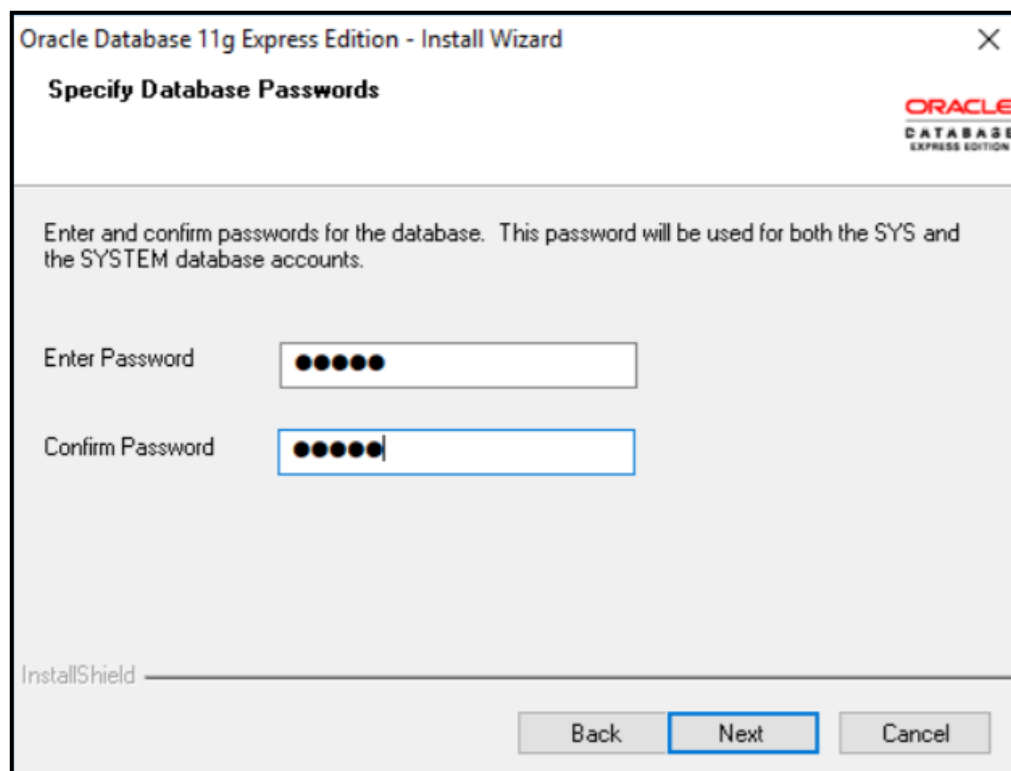


En rojo la versión que he escogido, que corresponde a la versión [Oracle Database 11gR2 Express Edition for Windows x64](#). Destacar que para que se permita realizar la descarga primero hay que realizar el proceso de registro en la página de Oracle.

Una vez descargado el archivo, se descomprime y se generará una carpeta que contendrá los archivos necesarios para la instalación. Ejecutamos el archivo ejecutable y se nos desplegará una ventana para seguir un proceso de instalación:

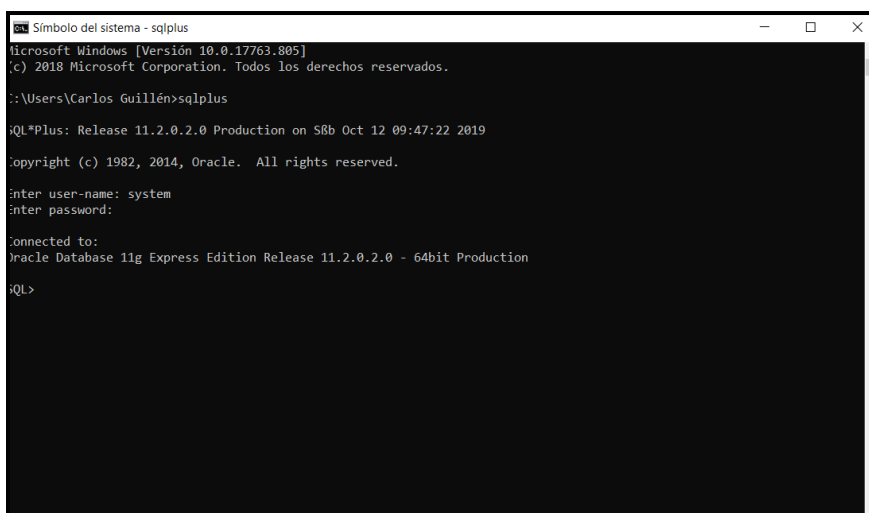


Clicamos en Siguiente y aceptamos los términos de licencia. Seguidamente se nos pedirá establecer una contraseña para la base de datos:



Esta contraseña la utilizaremos en SQL Developer para establecer conexión con la base de datos. Se decide que la contraseña que introducimos sea fácil de recordar, en este caso tendrá el valor **12345**. Una vez escrita la contraseña se acaba el proceso de instalación y tendríamos el software instalado en nuestro equipo.

Para comprobar de que no ha habido problemas de instalación desde la consola podemos ejecutar el comando sqlplus:



```
Símbolo del sistema - sqlplus
Microsoft Windows [Versión 10.0.17763.805]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Carlos Guillén>sqlplus

SQL*Plus: Release 11.2.0.2.0 Production on 08 Oct 12 09:47:22 2019

Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter user-name: system
Enter password:

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL>
```

Una vez ejecutado el comando, se introducen el usuario y la contraseña, en nuestro caso sería:

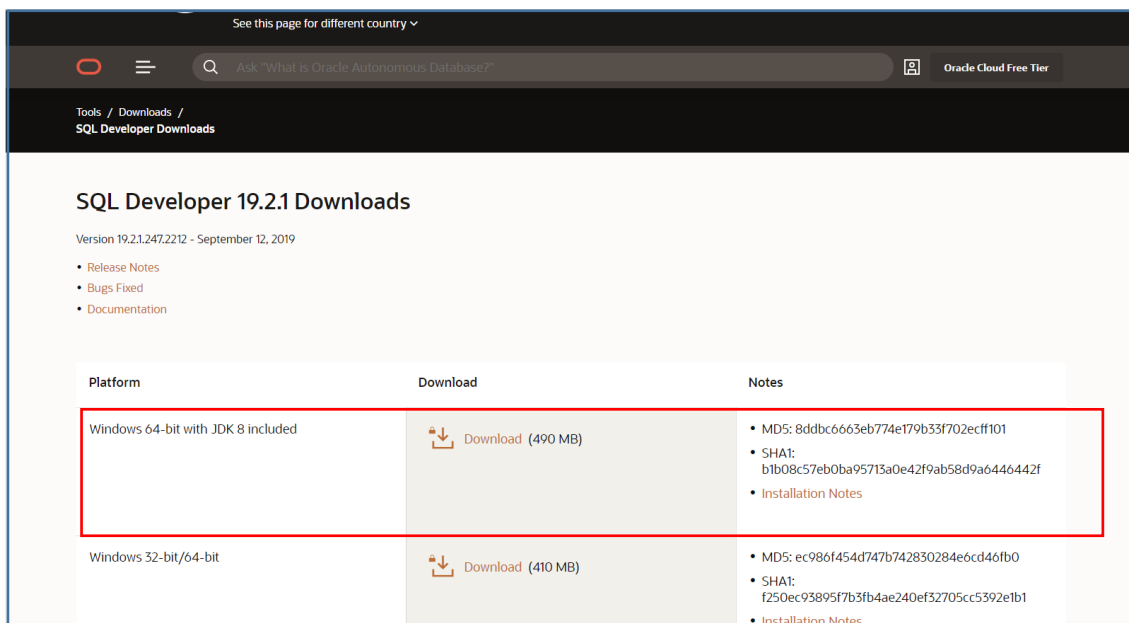
- **Usuario:** system
- **Contraseña:** 12345

Con esto nos conectaríamos a la base de datos donde se almacenará la información con la que trabajaremos más adelante en las siguientes sesiones.

## SQL Developer

Oracle SQL Developer es un entorno de desarrollo integrado y gratuito que simplifica el desarrollo y la gestión de Oracle Database en implementaciones tradicionales y en la nube. SQL Developer ofrece un completo desarrollo integral para sus aplicaciones PL/SQL, una hoja de trabajo para ejecutar consultas y scripts, una consola DBA para gestionar la base de datos, una interfaz de informe, una solución completa de modelado de datos y una plataforma de migración para mover sus bases de datos de terceros a Oracle.

Para descargar SQL Developer, accedemos a Oracle y en el apartado de herramientas (Tools) nos vamos descargas, seleccionamos SQL Developer y descargamos la versión que sea acorde con el equipo, en mi caso seguimos con la versión de Windows x64:





See this page for different country ▼

Tools / Downloads / SQL Developer Downloads

## SQL Developer 19.2.1 Downloads

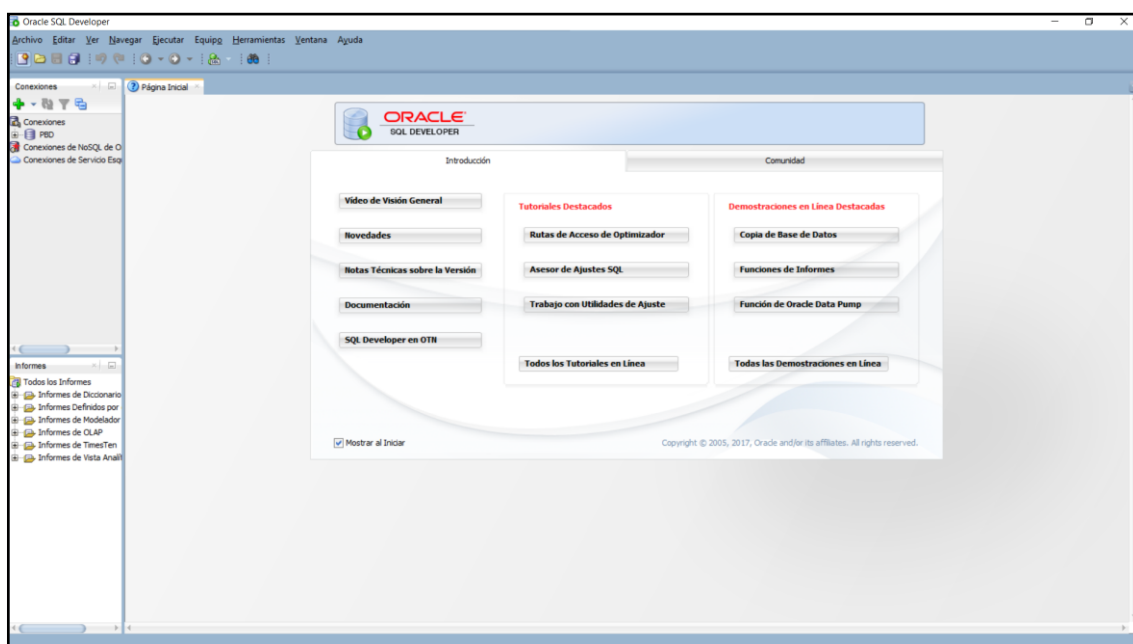
Version 19.2.1.247.2212 - September 12, 2019

- Release Notes
- Bugs Fixed
- Documentation

Platform	Download	Notes
Windows 64-bit with JDK 8 included	 Download (490 MB)	<ul style="list-style-type: none"> <li>MD5: 8ddbc663eb774e179b33f702ecff101</li> <li>SHA1: b1b08c57eb0ba95713a0e42f9ab58d9a6446442f</li> <li><a href="#">Installation Notes</a></li> </ul>
Windows 32-bit/64-bit	 Download (410 MB)	<ul style="list-style-type: none"> <li>MD5: ec986f454d747b742830284e6cd46fb0</li> <li>SHA1: f250ec93895f7b3fb4ae240ef32705cc5392e1b1</li> <li><a href="#">Installation Notes</a></li> </ul>

Se adjunta link hacia la página: <https://www.oracle.com/tools/downloads/sqldev-v192-downloads.html>

Una vez que se descarga el archivo, en la carpeta generada, ejecutamos el archivo ejecutable y se nos abrirá el programa:



Oracle SQL Developer

Archivo Editar Ver Navegar Ejecutar Equipos Herramientas Ventana Ayuda

Conexiones

Página Inicial

ORACLE SQL DEVELOPER

Introducción Comunidad

Vídeo de Visión General

Novedades

Notas Técnicas sobre la Versión

Documentación

SQL Developer en OTN

Tutoriales Destacados

Rutas de Acceso de Optimizador

Asesor de Ajustes SQL

Trabajo con Utilidades de Ajuste

Todos los Tutoriales en Línea

Demostraciones en Línea Destacadas

Copia de Base de Datos

Funciones de Informes

Función de Oracle Data Pump

Todas las Demostraciones en Línea

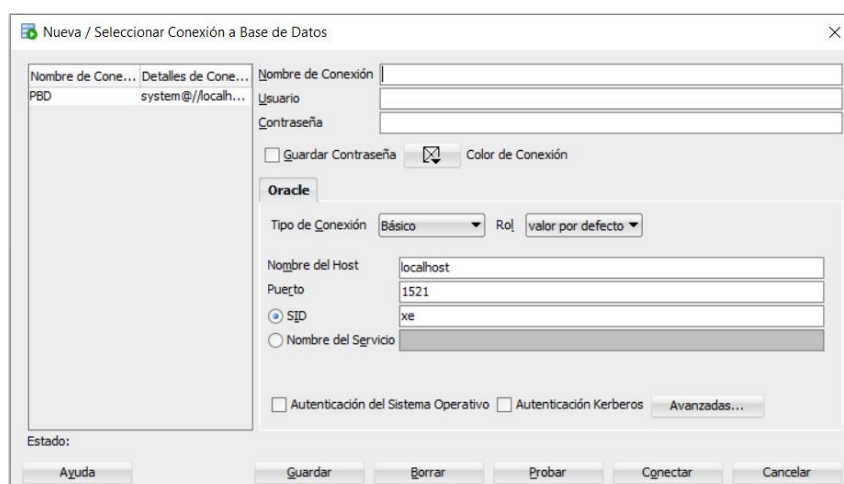
☒ Mostrar al Iniciar

Copyright © 2005, 2017, Oracle and/or its affiliates. All rights reserved.

Para crear la conexión con la base de datos hay que dar click en el icono verde con forma de + situado arriba a la izquierda de la interfaz:



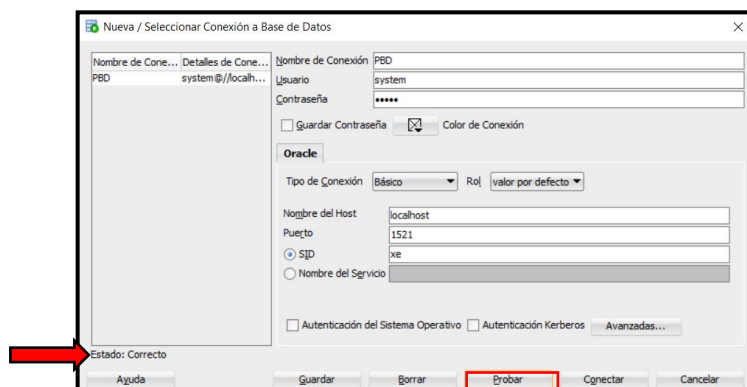
Una vez dado, se abrirá esta ventana donde hay que insertar los siguientes parámetros:



- **Nombre de Conexión:** Escogemos un nombre cualquiera para nombrar a la conexión, en este caso, lo llamaremos PBD.
- **Usuario:** El usuario que se genera por defecto en la instalación **system**.
- **Contraseña:** La contraseña que establecimos en el proceso de instalación, en nuestro caso, **12345**.
- **Nombre del host:** Como usaremos nuestra propio ordenador, escribimos **localhost** (que viene ya marcado por defecto).
- **Puerto:** Puerto desde donde el listener escucha la conexión en el servidor de la base de datos. El TNS Listener es un proceso servidor que provee conectividad de red con la base de datos Oracle. Por defecto el puerto del listener es **1521**.
- **SID:** Escribimos **xe** de la base de datos Express de Oracle.



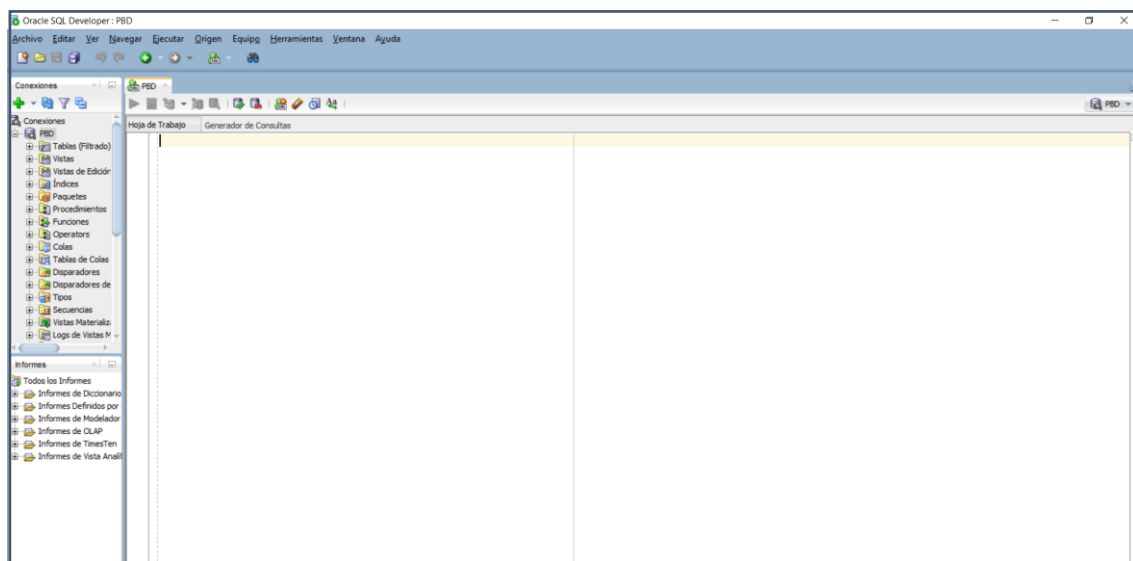
Una vez rellenos los campos, comprobamos que funciona, para ello le damos a **Probar** y si todo ha ido bien, al lado de Estado debería de aparecer **Correcto**:



Una vez que todo funciona correctamente, seleccionamos **Conectar** y ya tendremos guardada la conexión a la base de datos.

## Creación de tablas

Para la creación de las tablas que utilizaremos para insertar los datos con los que trabajaremos en las sesiones, debemos ejecutar el programa SQL Developer ya instalado. Una vez conectados a la base de datos con la conexión creada previamente (se pedirá la contraseña que la establecimos como **12345**), abriremos una hoja de trabajo (se abre sola cuando ejecutamos la conexión) y ahí copiaremos el código correspondiente a la creación de tablas:



Se adjunta el código correspondiente a la creación de las tablas en un fichero denominado

**1-Crear.sql**. Este scrip tiene el siguiente código:

```
SET SERVEROUTPUT ON;
```

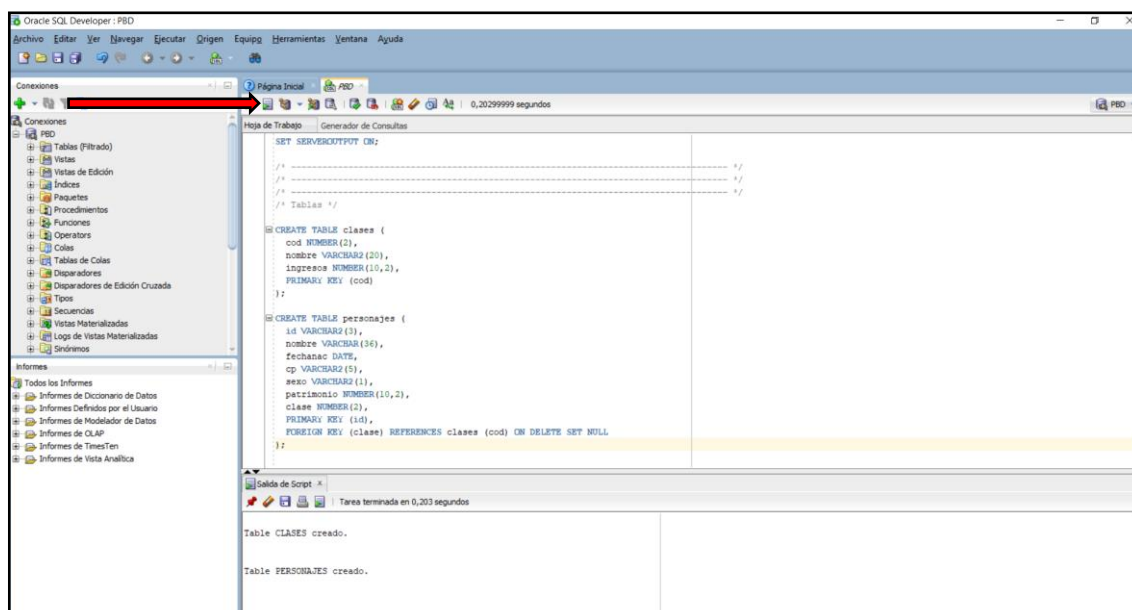
```
/* -----
----- */
/* -----
----- */
/* -----
----- */
/* Tablas */
```

```
CREATE TABLE clases (
  cod NUMBER(2),
  nombre VARCHAR2(20),
  ingresos NUMBER(10,2),

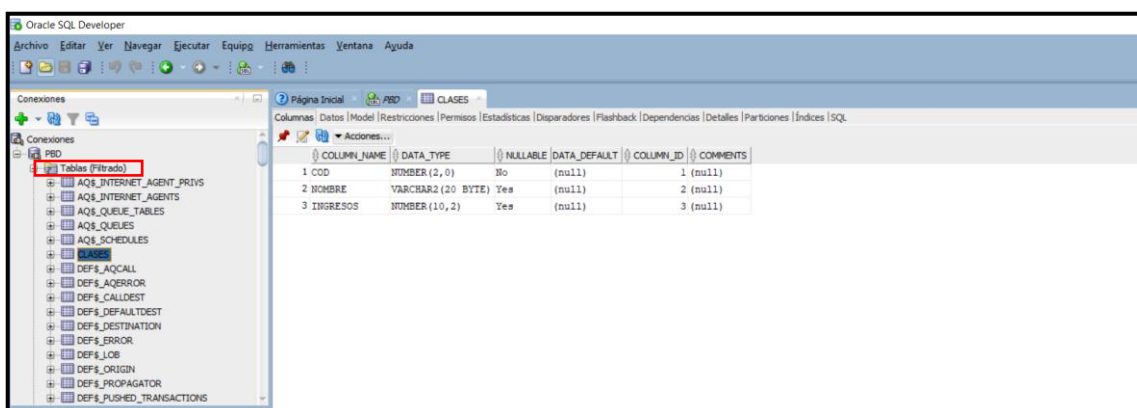
  PRIMARY KEY (cod)
);
```

```
CREATE TABLE personajes (
  id VARCHAR2(3),
  nombre VARCHAR(36),
  fechanac DATE,
  cp VARCHAR2(5),
  sexo VARCHAR2(1),
  patrimonio NUMBER(10,2),
  clase NUMBER(2),
  PRIMARY KEY (id),
  FOREIGN KEY (clase) REFERENCES clases (cod) ON DELETE SET NULL
);
```

Este código se lleva a la hoja de trabajo y se ejecuta el scrip:



Para ejecutar el scrip puedes pulsar la tecla F5 o clicar en la opción que se indica en la anterior imagen con la flecha. En la salida del scrip podemos ver que no ha habido fallos en la ejecución y el resultado de ejecutarlo que es la creación de las dos tablas. También es posible consultar las tablas creadas desplegando el icono donde pone **Tablas (Filtrado)**. Ahí aparecen los detalles de cada tabla (usaré de ejemplo la tabla Clase):



## Inserción de datos en las tablas

En el archivo que se nos adjunta llamado **2-Insertar.sql**, viene todas las inserciones de datos para las tablas previamente creadas. Este es el código del scrip:

```

/* -----
----- */
/* Inserciones */

INSERT INTO clases
VALUES (1,'Nobleza',0);

INSERT INTO clases
VALUES (2,'Clero',0);

INSERT INTO clases
VALUES (3,'Artesanos',0);

INSERT INTO clases
VALUES (4,'Campesinos',0);

/* -----
----- */

INSERT INTO personajes
VALUES ('123','Calixto Pájiz','15/11/1757','06300','H',150,4);

INSERT INTO personajes
VALUES ('777','Gervasio Romualdo','12/12/1721','06002','H',230,4);

INSERT INTO personajes
VALUES ('222','Prudencio González','01/02/1720','06300','H',220,4);

```



Universidad de Extremadura

```
INSERT INTO personajes
VALUES ('333','Macaria Gil','10/04/1731','06400','M',195,4);

INSERT INTO personajes
VALUES ('666','Honorio Mar','12/12/1742','10005','H',370,4);

INSERT INTO personajes
VALUES ('555','Venancio Fern ez','15/11/1738','10600','H',4600,3);

INSERT INTO personajes
VALUES ('696','Balbina S hez','12/04/1734','06400','M',5000,3);

INSERT INTO personajes
VALUES ('888','Faustino Martz','30/05/1731','06002','H',7000,3);

INSERT INTO personajes
VALUES ('999','Facundo Fern ez','12/03/1739','10800','H',6300,3);

INSERT INTO personajes
VALUES ('444','Pandulfa Ruiz','01/02/1750','06800','M',2500,3);

INSERT INTO personajes
VALUES ('987','Eduviges Pozo','10/05/1734','10005','M',40000,2);

INSERT INTO personajes
VALUES ('234','Abundio Hern ez','01/07/1749','06800','H',25000,2);

INSERT INTO personajes
VALUES ('345','Salustiana Moreno','07/04/1733','10300','M',40000,2);

INSERT INTO personajes
VALUES ('567','Camelia Cort s','12/05/1736','10600','M',48000,2);

INSERT INTO personajes
VALUES ('789','Sagrario M ez','22/06/1759','10800','M',18000,2);

INSERT INTO personajes
VALUES ('901','Amable Montero','07/04/1760','10300','H',140000,1);

INSERT INTO personajes
VALUES ('012','Martiniano
Zarzal','23/11/1736','10005','H',360000,1);

INSERT INTO personajes
VALUES ('876','Mederica Campos','19/03/1734','06800','M',500000,1);

INSERT INTO personajes
VALUES ('321','Filiberto Torres','08/08/1748','06002','H',250000,1);

INSERT INTO personajes
VALUES ('221','Obdulia Candil','08/08/1737','10005','M',502000,1);

SELECT * FROM clases;
SELECT * FROM personajes;
```

## Universidad de Extremadura

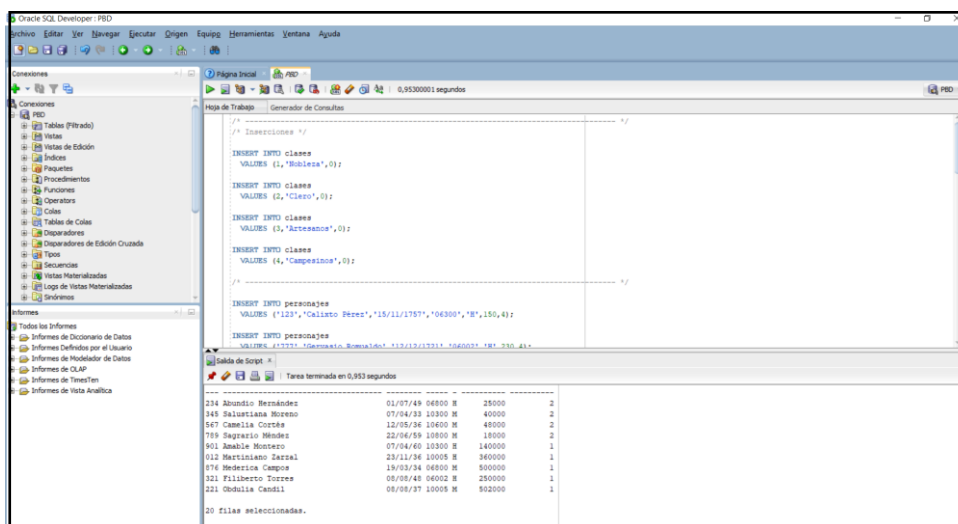
```

/*
10005 - Cáceres
10600 - Plasencia
10300 - Navalmoral de la Mata
10800 - Coria

06002 - Badajoz
06800 - Mérida
06400 - Don Benito
06300 - Zafra
*/

```

Este archivo se ejecutará en la hoja de trabajo donde hemos creado las tablas y se ejecutará de igual forma que lo hicimos previamente. Para ello, copiamos el contenido en la hoja de trabajo y ejecutamos:



Se hace una consulta al finalizar la inserción para ver los datos que hemos insertado en las tablas a partir del scrip facilitado:

COD NOMBRE		INGRESOS				
1	Nobleza	0				
2	Clero	0				
3	Artesanos	0				
4	Campesinos	0				

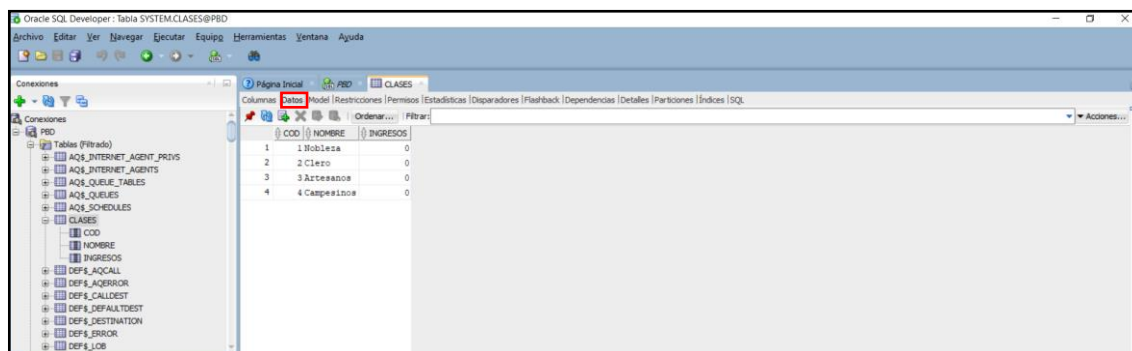
ID	NOMBRE	FECHANAC	CP	S	PATRIMONIO	CLASE
123	Calixto Pérez	15/11/57	06300	H	150	4
777	Gervasio Romualdo	12/12/21	06002	H	230	4
222	Prudencio González	01/02/20	06300	H	220	4
333	Macaria Gil	10/04/31	06400	M	195	4
666	Honorio Marín	12/12/42	10005	H	370	4
555	Venancio Fernández	15/11/38	10600	H	4600	3
696	Balbina Sánchez	12/04/34	06400	M	5000	3
888	Faustino Martínez	30/05/31	06002	H	7000	3
999	Facundo Fernández	12/03/39	10800	H	6300	3
444	Pandulfa Ruiz	01/02/50	06800	M	2500	3
987	Eduviges Pozo	10/05/34	10005	M	40000	2

ID	NOMBRE	FECHANAC	CP	S	PATRIMONIO	CLASE
234	Abundio Hernández	01/07/49	06800	H	25000	2
345	Salustiana Moreno	07/04/33	10300	M	40000	2
567	Camelia Cortés	12/05/36	10600	M	48000	2
789	Sagrario Méndez	22/06/59	10800	M	18000	2
901	Amable Montero	07/04/60	10300	H	140000	1
012	Martiniano Zarzal	23/11/36	10005	H	360000	1
876	Medérica Campos	19/03/34	06800	M	500000	1
321	Filiberto Torres	08/08/48	06002	H	250000	1
221	Obdulia Candil	08/08/37	10005	M	502000	1

También podemos ver en la sección de **Tablas (Cifrado)** los datos que hay en cada tabla, para ello seleccionamos la tabla que queremos ver los datos y se desplegará la vista con los datos.

En este caso accedemos a la tabla con nombre Clases y en la pestaña **Datos** se abre la siguiente vista:

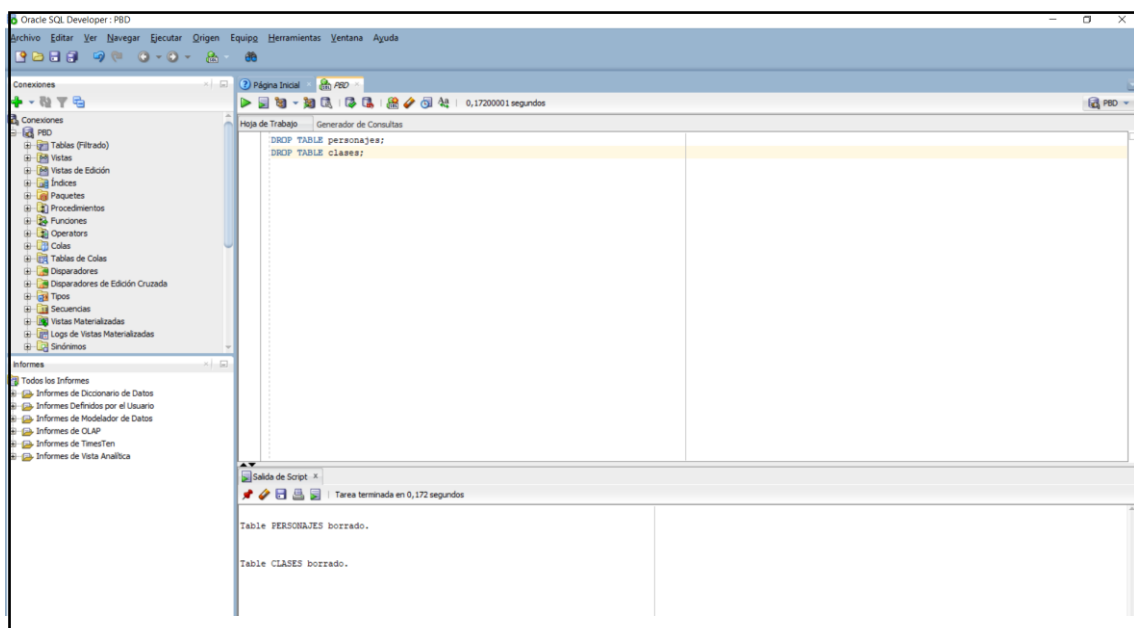


## Borrado de tablas

Una vez hayamos acabado con las consultas podemos borrar las tablas y sus datos. Para ello se adjunta un scrip denominado **3-Borrar.sql**:

```
DROP TABLE personajes;  
DROP TABLE clases;
```

De igual forma, nos vamos a la hoja de trabajo que hemos estado usando y ejecutamos el scrip. El resultado de ejecutarlo es este:



Podemos observar que se han borrado las tablas correctamente y que ya no aparecen en la sección de **Tablas (Cifrado)**.

**Nota:** Como ayuda a la hora de borrar la hoja de trabajo se puede utilizar el comando **Control + D** para borrar todo el contenido de la misma.



### *Objetos de la base de datos*

Se recomienda al usuario que una vez haya acabado de trabajar con las tablas y sus correspondientes datos, que eliminen dichas tablas para no llenar el espacio de trabajo que normalmente suele estar limitado (no es nuestro caso).

También podemos hacer varios tipos de consultas para ver los objetos que hemos creado como usuarios del sistema:

```
SELECT *  
FROM dba_objects  
WHERE owner='SYSTEM' and created>=to_date('03/10/2019') AND  
generated='N';
```

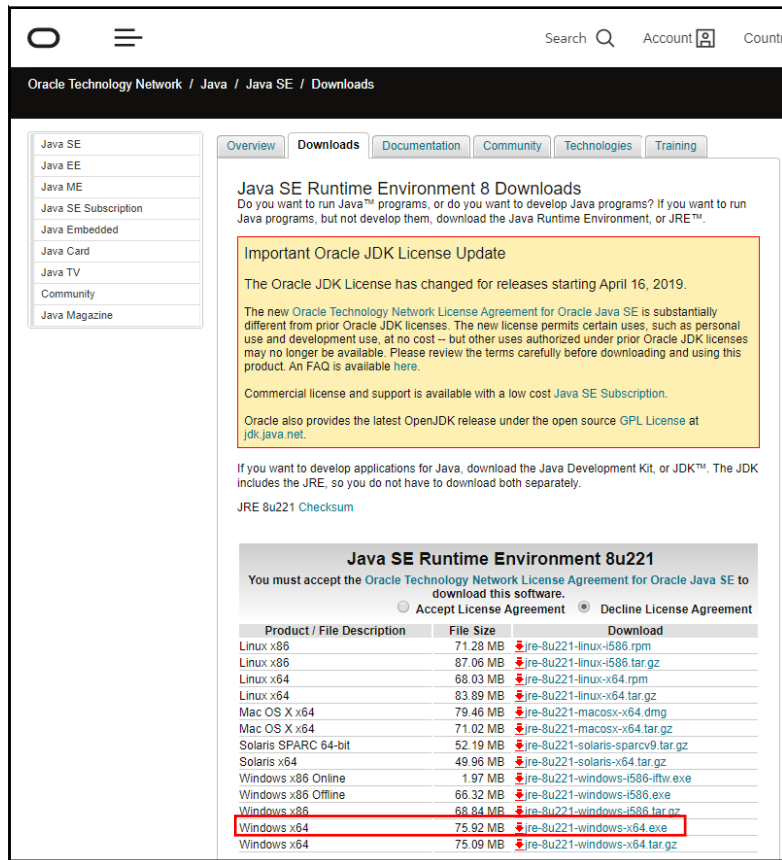
Con esta consulta vemos los objetos que hemos creado a partir del día 3 de octubre de 2019 y que no hayan sido generados por el sistema.

### Eclipse

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Para el transcurso de la práctica es necesario instalar la versión de **Eclipse Oxygen 4.7**. Para ello, accedemos a la página de Oracle y en la sección descargas buscamos la versión acorde a nuestras características del ordenador, en mi caso **Windows x64**. Se puede dar el caso de que no esté instalado en el equipo **Java SE Runtime Environment**, y es necesario para que Eclipse pueda funcionar. Se accede a la página de Oracle y en descargas seleccionamos la versión que necesite cada equipo. Antes de descargar debemos aceptar el acuerdo de licencia:





Oracle Technology Network / Java / Java SE / Downloads

Java SE Runtime Environment 8 Downloads

Do you want to run Java™ programs, or do you want to develop Java programs? If you want to run Java programs, but not develop them, download the Java Runtime Environment, or JRE™.

**Important Oracle JDK License Update**

The Oracle JDK License has changed for releases starting April 16, 2019.

The new Oracle Technology Network License Agreement for Oracle Java SE is substantially different from prior Oracle JDK licenses. The new license permits certain uses, such as personal use and development use, at no cost -- but other uses authorized under prior Oracle JDK licenses may no longer be available. Please review the terms carefully before downloading and using this product. An FAQ is available [here](#).

Commercial license and support is available with a low cost Java SE Subscription.

Oracle also provides the latest OpenJDK release under the open source GPL License at [jdk.java.net](#).

If you want to develop applications for Java, download the Java Development Kit, or JDK™. The JDK includes the JRE, so you do not have to download both separately.

JRE 8u221 Checksum

**Java SE Runtime Environment 8u221**

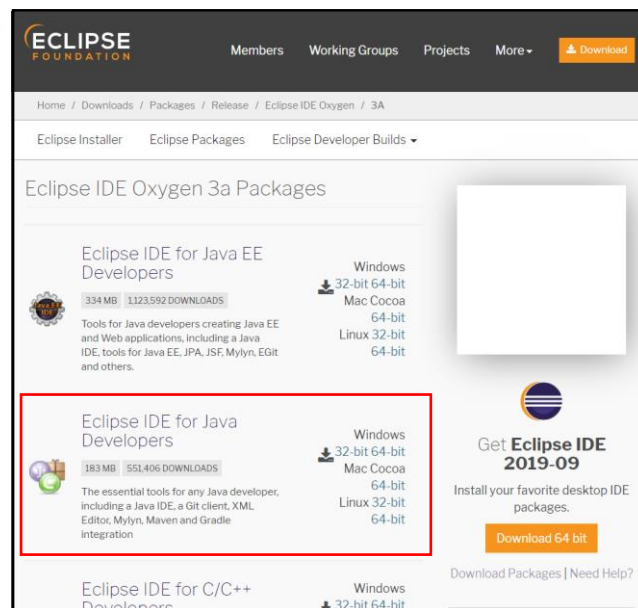
You must accept the [Oracle Technology Network License Agreement for Oracle Java SE](#) to download this software.

☐ Accept License Agreement ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	71.28 MB	<a href="#">jre-8u221-linux-i586.rpm</a>
Linux x86	87.06 MB	<a href="#">jre-8u221-linux-i586.tar.gz</a>
Linux x64	68.03 MB	<a href="#">jre-8u221-linux-x64.rpm</a>
Linux x64	83.89 MB	<a href="#">jre-8u221-linux-x64.tar.gz</a>
Mac OS X x64	79.46 MB	<a href="#">jre-8u221-macosx-x64.dmg</a>
Mac OS X x64	71.02 MB	<a href="#">jre-8u221-macosx-x64.tar.gz</a>
Solaris SPARC 64-bit	52.19 MB	<a href="#">jre-8u221-solaris-sparcv9.tar.gz</a>
Solaris x64	49.96 MB	<a href="#">jre-8u221-solaris-x64.tar.gz</a>
Windows x86 Online	1.97 MB	<a href="#">jre-8u221-windows-i586-iftw.exe</a>
Windows x86 Offline	66.32 MB	<a href="#">jre-8u221-windows-i586.exe</a>
Windows x86	68.84 MB	<a href="#">jre-8u221-windows-i586.tar.gz</a>
Windows x64	75.92 MB	<a href="#">jre-8u221-windows-x64.exe</a>
Windows x64	75.09 MB	<a href="#">jre-8u221-windows-x64.tar.gz</a>

Una vez descargamos el archivo ejecutable, hacemos doble click y se iniciaría el proceso de instalación en nuestro equipo. No podemos seguir en Eclipse sin tenerlo instalado.

Cuando se termine de instalar descargamos la versión de Eclipse antes descrita, la versión **Eclipse IDE for Java Developers**:



ECLIPSE FOUNDATION

Members Working Groups Projects More Download

Home / Downloads / Packages / Release / Eclipse IDE Oxygen / 3A

Eclipse Installer Eclipse Packages Eclipse Developer Builds

**Eclipse IDE Oxygen 3a Packages**

**Eclipse IDE for Java EE Developers**

334 MB 1123592 DOWNLOADS

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

Windows 32-bit 64-bit  
Mac Cocoa 64-bit  
Linux 32-bit 64-bit

**Eclipse IDE for Java Developers**

183 MB 551406 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

Windows 32-bit 64-bit  
Mac Cocoa 64-bit  
Linux 32-bit 64-bit

**Get Eclipse IDE 2019-09**

Install your favorite desktop IDE packages.

Download 64 bit

Download Packages | Need Help?

En este link se puede acceder a la página de descarga:

<https://www.eclipse.org/downloads/packages/release/oxygen/3a>


Una vez descargado, descomprimos el archivo y lo almacenamos en la ruta que deseemos. Ejecutamos Eclipse y seleccionamos nuestro espacio de trabajo donde se almacenará los proyectos que vayamos realizando.

## JDBC para Oracle

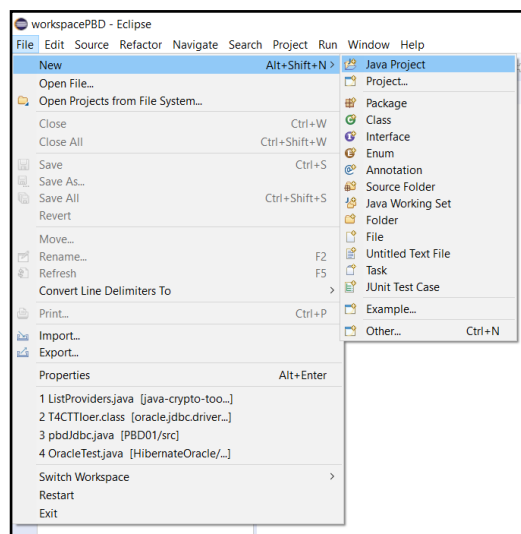
Para que cuando trabajemos con el proyecto de la sesión pueda conectarse a la base de datos, es necesario instalar un driver. Accedemos a Oracle a la sección de drivers y seleccionamos la versión [Oracle Database 11g Release 2 \(11.2.0.4\) drivers](#). Aquí llegados, descargamos el archivo [ojdbc6.jar](#):

### Unzipped JDBC Driver and Companion JARs

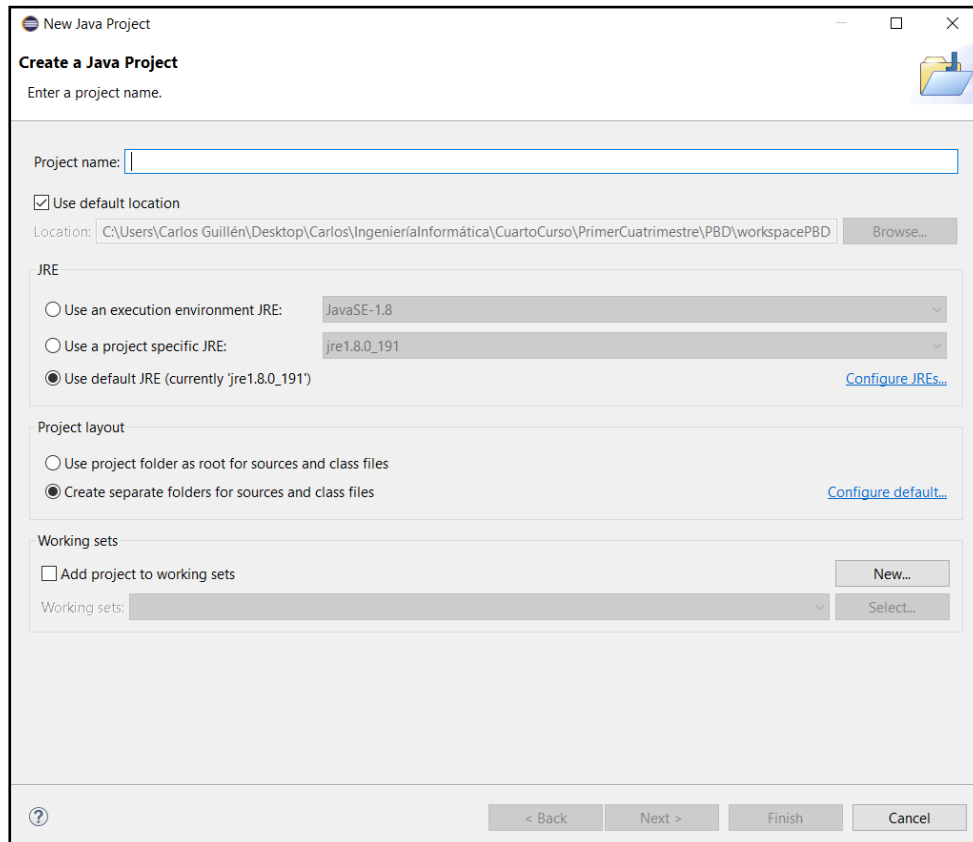
The JARs included in the [ojdbc-full.tar.gz](#) are also available as individual downloads in this section.

Download	Release Notes
 <a href="#">ojdbc6.jar</a>	(2,739,670 bytes) - (SHA1 Checksum: a483a046eee2f404d864a6ff5b09dc0e1be3fe6c) <b>Certified with JDK8, JDK7, and JDK6;</b>

Una vez tenemos esto descargado, podemos crear un proyecto en Eclipse. Para crear el proyecto en Eclipse, abrimos Eclipse y seleccionamos **File->New->Java Project**:



Se abrirá la siguiente ventana que se rellenará con los siguientes valores:



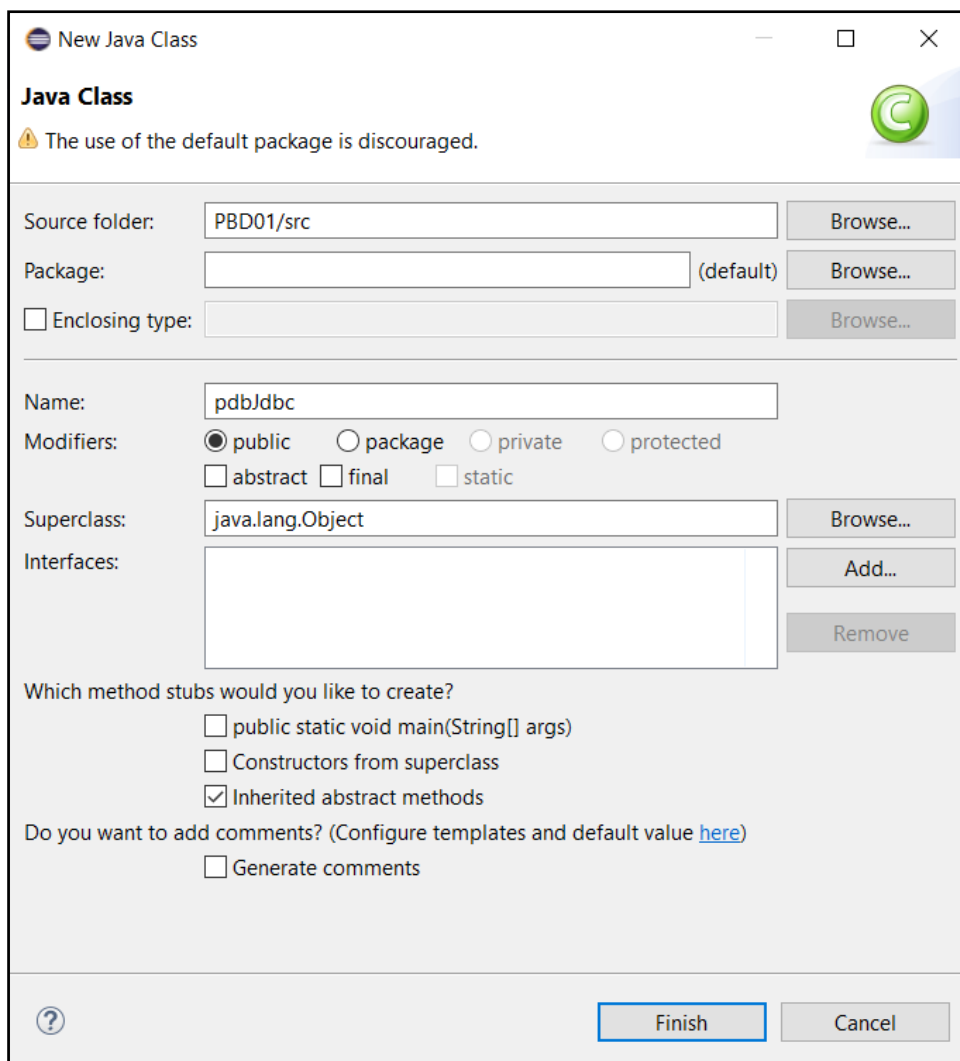
-**Project name:** En este caso llamaremos al proyecto **PBD01**.

-**JRE:** Seleccionamos la opción **Use default JRE (currently `jre1.8.0\_191`)** que en mi caso es la versión actual.

-**Project layout:** Seleccionamos la opción **Create separate folders for sources and classs files**.

Una vez hecho esto, le damos a **Finish** y tendríamos el proyecto creado.

Ahora, haciendo click derecho con el ratón sobre el proyecto creado, **PBD01**, seleccionamos **New->Class** y creamos una clase con nombre **pbdJdbc**.



**New Java Class**

**Java Class**

⚠ The use of the default package is discouraged.

Source folder: PBD01/src Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: pdbJdbc

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...  
Remove

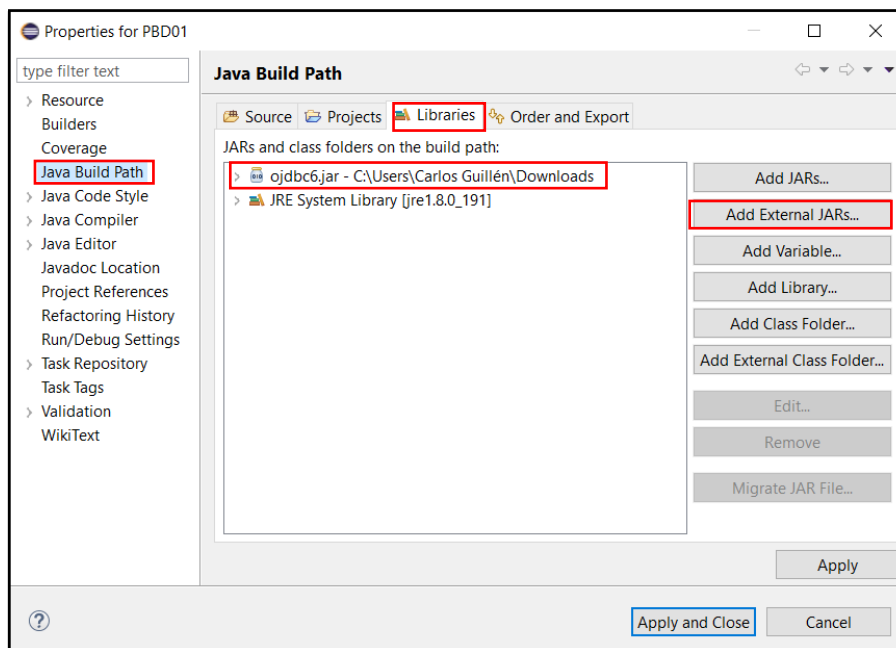
Which method stubs would you like to create?

☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

? Finish Cancel

Ahora hay que configurar el **Build Path** del proyecto para añadirle el driver **ojdbc6.jar** que hemos descargado antes. Con este driver ya seremos capaces de conectarnos a la base de datos vía JDBC. Para hacer esto, hacemos click derecho del ratón sobre el proyecto **PBD01** y seleccionamos la opción **Properties**. Ahí se nos despliega una ventana donde seleccionaremos en el panel lateral izquierdo la opción que dice **Java Build Path**. Aquí nos vamos a la pestaña **Libraries** y después seleccionamos la opción que dice **Add External JARs**. Buscamos el archivo **ojdbc6.jar** en nuestro equipo y lo añadimos al proyecto. Quedaría así:



Se marca en rojo las opciones descritas previamente y el resultado de las mismas que es la inserción del **ojdbc6.jar** en la lista.

Una vez hecho esto ya estaría creado el proyecto con el que podemos trabajar, quedaría implementar las funciones dentro del mismo.

## Importación de librerías

Es necesario importar una serie de librerías al proyecto creado para manejar algunas funcionalidades. Son las siguientes:

```
import java.io.IOException;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

Librería	Función
<code>import java.io.IOException;</code>	Manejar las excepciones de Java
<code>import java.sql.DriverManager;</code>	Servicio básico para gestionar drivers JDBC
<code>import java.sql.Connection;</code>	Iniciar sesión en una base de datos. Los comandos SQL necesitan esta conexión abierta para ejecutarse
<code>import java.sql.ResultSet;</code>	Tabla de datos con los resultados de una consulta. Mantiene un cursor en la fila actual apuntando al elemento anterior al primero
<code>import java.sql.SQLException;</code>	Informar sobre errores de acceso a la base de datos
<code>import java.sql.Statement;</code>	Objeto que almacena una sentencia SQL precompilada. Necesita los parámetros de la consulta
<code>import java.sql.PreparedStatement;</code>	Subclase de Statement. Permite preparar la sentencia antes de concretar los parámetros. Recomendable si una consulta se ejecuta varias veces

Después de insertar todos los imports que se describen en la imagen anterior en el proyecto, se crea un atributo estático dentro de la clase para establecer la conexión:

```
public class pbdJdbc {

    private static Connection conexion = null ;

}
```

Este atributo se usa para la interacción del usuario con la base de datos, ya que almacena la conexión local con la base de datos.

## PreparedStatement/CreateStatement

En JDBC hay una clase llamada Statement, que se divide en PreparedStatement y CallableStatement. Existen 3 tipos de Statement:

- **Statement:** para implementar sentencias simples de SQL sin parámetros.
- **PreparedStatement:** se usa para precompilar sentencias SQL que puede tener o no parámetros de entrada.
- **CallableStatement:** se usa para ejecutar procedimientos almacenados que puede tener parámetros de entrada o salida.

## Métodos de la clase

### dbConectar()

Método que se usa para conectarse a la base de datos. El código es el siguiente:

```
public boolean dbConectar() {

    System.out.println("---dbConectar---");

    String driver = "oracle.jdbc.OracleDriver";
    // Crear la conexion a la base de datos
    String servidor = "localhost"; // Direccion IP
    String puerto = "1521";
    String sid = "xe"; // Identificador del servicio o instancia
    String url = "jdbc:oracle:thin:@//" + servidor + ":" + puerto + "/" + sid;
    String usuario = "system"; // usuario
    String contrasena = "12345"; // contrasena

    try {
        System.out.println("---Conectando a Oracle---");

        Class.forName (driver); // Cargar el driver JDBC para Oracle
        conexion = DriverManager.getConnection (url, usuario, contrasena);
        System.out.println ("Conexion realizada a la base de datos " + conexion);
        return true;
    } catch (ClassNotFoundException e) {
        // Error. No se ha encontrado el driver de la base de datos
        e.printStackTrace();
        return false;
    } catch (SQLException e) {
        // Error. No se ha podido conectar a la BD
        e.printStackTrace();
        return false;
    }
}
```

#### Class.forName (driver):

-Se utiliza para poder activar el plugin de la base de datos Oracle.

#### DriverManager.getConnection(url, usuario, contraseña):

-Método que nos permite establecer una conexión con la base de datos.

-Parámetros:

- **Url:** URL de la base de datos. Suele ser de la forma jdbc:subprotocol:subname.
- **Usuario:** usuario con el que se quiere establecer la conexión a la base de datos. Su valor es el de por defecto de la instalación.

- **Contraseña:** contraseña con el que se quiere establecer la conexión a la base de datos. En nuestro caso, establecimos como contraseña el valor 12345.
- **Info:** lista de pares clave/valor con los parámetros de la conexión. Suelen llevar incluidos usuario y contraseña. Este parámetro no lo usamos en el método pero lo explico por si se quisiera usar en algún momento.

Guardamos en la variable privada de la clase (**conexion**) la conexión establecida con el método anterior. Este método se inicia al comenzar el método principal (main):

```
public static void main(String[] args) {
    pbdJdbc cliente = new pbdJdbc();
    System.out.println("---Programa principal---");

    if (!cliente.dbConectar())
        System.out.println("Error: Conexion no realizada.");

    cliente.dbObtenerPersonajes1();
    cliente.dbObtenerPersonajes2();

    cliente.dbConsultarPersonajes();
    cliente.dbConsultarClases();

    cliente.dbInsertarClases();
    cliente.dbConsultarClases();

    cliente.dbModificarClases();
    cliente.dbConsultarClases();

    cliente.dbBorrarClases();
    cliente.dbConsultarClases();

    if (!cliente.dbDesconectar())
        System.out.println("Desconexión no realizada");

    System.out.println("---Fin de programa---");
}
```



## dbDesconectar()

Esta función se llama para cerrar la conexión que tenemos como atributo de la clase y que usamos para conectarnos con la base de datos. El código de la función es el siguiente:

```
public boolean dbDesconectar() {
    System.out.println("---dbDesconectar---");

    try {
        conexion.commit();
        conexion.close();
        System.out.println("Desconexión realizada correctamente");
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

Aquí la función a destacar es la de `conexion.close()` que cierra la conexión. A este método se llama en la función principal al final, después de ejecutar todos los métodos para hacer las consultas:

```
public static void main(String[] args){
    pbdJdbc cliente = new pbdJdbc();
    System.out.println("---Programa principal---");

    if (!cliente.dbConectar())
        System.out.println("Error: Conexion no realizada.");

    cliente.dbObtenerPersonajes1();
    cliente.dbObtenerPersonajes2();

    cliente.dbConsultarPersonajes();
    cliente.dbConsultarClases();

    cliente.dbInsertarClases();
    cliente.dbConsultarClases();

    cliente.dbModificarClases();
    cliente.dbConsultarClases();

    cliente.dbBorrarClases();
    cliente.dbConsultarClases();

    if (!cliente.dbDesconectar())
        System.out.println("Desconexión no realizada");

    System.out.println("---Fin de programa---");
}
```

## dbObtenerPersonajes1()

A continuación se muestra una forma de mostrar los atributos de un personaje específico. El código de la función es el siguiente:

```
public void dbObtenerPersonajes1() {
    PreparedStatement ps;
    String IDobjetivo;

    System.out.println("---dbObtenerPersonajes1---");

    try {
        ps = conexion.prepareStatement("SELECT NOMBRE, CP, SEXO,
PATRIMONIO, CLASE FROM PERSONAJES WHERE ID = ");

        // Por ejemplo, buscar Personajes con ID 987
        IDobjetivo = readEntry("Introduce ID de Personajes: ");
        ps.clearParameters();
        ps.setString(1, IDobjetivo);
        ResultSet rset = ps.executeQuery();

        if (rset.next()) {
            System.out.println("Nombre: "+rset.getString(1));
            //...

            //////////////////////////////////////
            System.out.println("Código postal:
"+rset.getString(2));
            System.out.println("Sexo: "+rset.getString(3));
            System.out.println("Patrimonio: "+rset.getFloat(4));

            //////////////////////////////////////
            System.out.println("Clase: "+rset.getInt(5));
        }
        rset.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Se muestra en azul el uso de la sentencia PreparedStatement. Optamos por esta sentencia en este caso porque es una consulta de selección determinada previamente y al utilizar el atributo ID que es clave primaria solo devolverá un resultado o ninguno si no existe ninguna instancia con ese valor. Se realiza la sentencia SQL y usamos como ID el valor que entramos en la variable IDobjetivo que leemos por teclado. Realizamos un if por si hemos obtenido un valor y mostramos la información relativa a la instancia, en este caso, se muestran los atributos nombre, código postal, sexo, patrimonio y clase. Usamos la sentencia rset.getTipoDato(posición) para obtener de la consulta los valores de cada atributo según la posición que establezcamos dentro del paréntesis. Por ejemplo, rset.getString(3) me devolvería un tipo String del tercer atributo que establecemos en la consulta. Cerramos el ResultSet con la sentencia rset.close().

## dbObtenerPersonajes2()

Se muestra una forma de mostrar los atributos de un personaje específico de forma más adecuada, ya que este tipo de consultas solo se ejecutan una vez. El código de la función es el siguiente:

```
public void dbObtenerPersonajes2() {
    Statement ps;
    String consulta;

    System.out.println("---dbObtenerPersonajes2---");

    try {
        ps = conexion.createStatement();

        // Por ejemplo, buscar Personajes con ID 987
        consulta = readEntry("Introduce ID de Personajes: ");
        String result = "SELECT nombre, id, sexo, patrimonio,
        clase FROM Personajes WHERE id = '" + consulta + "'";
        ResultSet rset = ps.executeQuery(result);

        System.out.println(result);

        if (rset.next()) {
            System.out.println("Nombre: "+rset.getString(1));

            //////////////////////////////////////
            System.out.println("Código postal:
            "+rset.getString(2));
            System.out.println("Sexo: "+rset.getString(3));
            System.out.println("Patrimonio: "+rset.getFloat(4));

            //////////////////////////////////////
            System.out.println("Clase: "+rset.getInt(5));
            System.out.println("-----");
        }
        rset.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Se muestra en **amarillo** el uso de la sentencia CreateStatement. Optamos por esta sentencia en este caso porque es una consulta que solo se va a ejecutar una vez. Se realiza la sentencia **SQL** y usamos como ID el valor que entramos en la variable **consulta** que leemos por teclado. Realizamos un if por si hemos obtenido un valor y mostramos la información relativa a la instancia, en este caso, se muestran los atributos **nombre**, **código postal**, **sexo**, **patrimonio** y **clase**. Usamos la sentencia **rset.getTipoDato(posición)** para obtener de la consulta los valores

de cada atributo según la posición que establezcamos dentro del paréntesis. Por ejemplo, `rset.getString(3)` me devolvería un tipo `String` del tercer atributo que establecemos en la consulta. Cerramos el `ResultSet` con la sentencia `rset.close()`.

## Ejercicios propuestos

### 4.a) dbConsultarPersonajes()

En este método se consulta la información de la población de la base de datos. El código de la función es el siguiente:

```
public void dbConsultarPersonajes() {
    Statement ps;
    String consulta;

    System.out.println("---dbConsultarPersonajes---");

    try {
        ps = conexion.createStatement();
        consulta = "SELECT ID, NOMBRE, FECHANAC, CP, SEXO,
        PATRIMONIO, CLASE FROM PERSONAJES";
        ResultSet rset = ps.executeQuery(consulta);

        System.out.println(consulta);

        while (rset.next()) {
            System.out.println("ID: "+rset.getString(1));
            System.out.println("Nombre: "+rset.getString(2));
            //...

            //////////////////////////////////////
            System.out.println("FechaNac: "+rset.getString(3));
            System.out.println("CP: "+rset.getString(4));
            System.out.println("Sexo: "+rset.getString(5));
            System.out.println("Ingresos: "+rset.getFloat(6));

            //////////////////////////////////////
            System.out.println("Clase: "+rset.getInt(7));
            System.out.println("-----");
        }

        rset.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Se decide por crear una variable ps de tipo `Statement`, ya que se quiere ejecutar una única vez la sentencia y es la que más se recomienda en estos casos. Se guarda la `consulta` en una variable de tipo String y el resultado de la consulta se guarda en la variable `rset` utilizando el método `executeQuery()` para ejecutar la consulta. Se muestran los datos como en los apartados anteriores haciendo uso del `rset.getAttribute(posición)` y cuando se muestran los datos se cierra la variable `rset` con la función `close()`.

#### 4.a) dbConsultarClases()

En este método se consulta la información de las clases de la base de datos. El código de la función es el siguiente:

```
public void dbConsultarClases() {
    Statement ps;
    String consulta;

    System.out.println("---dbConsultarClases---");

    try {
        ps = conexion.createStatement();

        consulta = "SELECT cod, Nombre, Ingresos FROM CLASES";
        ResultSet rset = ps.executeQuery(consulta);

        System.out.println(consulta);

        while (rset.next()) {
            System.out.println("Codigo Clase: "+rset.getInt(1));

            System.out.println("Nombre Clase: "+rset.getString(2));

            System.out.println("Ingresos Clase: "+rset.getFloat(3));
            System.out.println("-----");
        }
        rset.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Se decide por crear una variable `ps` de tipo `Statement`, ya que se quiere ejecutar una única vez la sentencia y es la que más se recomienda en estos casos. Se guarda la `consulta` en una variable de tipo `String` y el resultado de la consulta se guarda en la variable `rset` utilizando el método `executeQuery()` para ejecutar la consulta. Se muestran los datos como en los apartados anteriores haciendo uso del `rset.getAttribute(posición)` y cuando se muestran los datos se cierra la variable `rset` con la función `close()`.

#### 4.b) `dbInsertarClases()`

En este método se inserta una nueva tupla en la tabla `Clases`. El código de la función es el siguiente:

```
public void dbInsertarClases() {
    Statement ps;
    String consulta;
    String codigo, nombre;
    int resultado;
    int ingresos;

    System.out.println("---dbInsertarClases---");

    try {
        ps = conexion.createStatement();

        codigo = readEntry("Codigo clase: "); // por ejemplo 0
        nombre = readEntry("Nombre Clase: ");
        ingresos = 0;

        consulta = "INSERT INTO CLASES VALUES ('" + codigo + "', '"
+ nombre + "', '" + ingresos + "'" );
        // OJO: Las cadenas en el insert deben ir entre comillas
        simples ''
        System.out.println(consulta);
        resultado = ps.executeUpdate(consulta);

        System.out.println("Numero de filas afectadas:
"+resultado);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Para realizar la inserción utilizamos una variable `ps` de tipo `Statement`. Pedimos al usuario que introduzca el valor por teclado de los atributos de la tabla código y nombre ya que los ingresos los establecemos a 0 al crear la instancia. Utilizamos la sentencia `insert` para insertar los datos que hemos obtenido anteriormente y ejecutamos la sentencia con la función `executeUpdate` que devuelve el valor de las instancias afectadas en la variable `resultado`, que se muestra después en pantalla.

#### 4.c) dbModificarClases()

La función de este método es modificar la clase que hemos insertado anteriormente. El código de la función es el siguiente:

```
public void dbModificarClases() {
    Statement ps;
    String consulta;
    String nombre;
    int resultado;

    System.out.println("---dbModificarClases---");

    try {
        ps = conexion.createStatement();

        nombre = "Desconocido"; // Cambiar el nombre por
        'Desconocido'

        consulta = "UPDATE CLASES SET NOMBRE = '" + nombre + "' WHERE COD = 0";
        // OJO: Las cadenas en el insert deben ir entre comillas
        simples ''
        resultado = ps.executeUpdate(consulta);

        System.out.println(consulta);
        System.out.println("Numero de filas afectadas: " + resultado);

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Para realizar la actualización utilizamos una variable ps de tipo **Statement**. Creamos una variable local llamada nombre donde escribiremos su valor, en este caso **Desconocido**. Utilizamos la sentencia **Update** para modificar los datos de la instancia insertada anteriormente con valor 0 en el atributo cod (código de la clase) y ejecutamos la sentencia con la función **executeUpdate** que devuelve el valor de las instancias afectadas en la variable **resultado**, que se muestra después en pantalla.

#### 4.d) dbBorrarClases()

La función de este método es eliminar la tupla de la tabla clase que hemos insertado anteriormente. El código de la función es el siguiente:

```
public void dbBorrarClases() {
    Statement ps;
    String consulta;
    int resultado, numero;

    System.out.println("---dbBorrarClases---");

    try {
        ps = conexion.createStatement();

        numero = 0; // Borrar el sector 0

        consulta = "DELETE FROM CLASES WHERE cod="+numero+"";
        resultado = ps.executeUpdate(consulta);

        System.out.println(consulta);
        System.out.println("Numero de filas afectadas: "+resultado);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Para realizar la actualización utilizamos una variable ps de tipo **Statement** ya que solo lo ejecutaremos una vez. Creamos una variable local numero donde almacenamos el valor del atributo cod de la tabla Clases que se asocia a la inserción que hicimos previamente. Construimos la **consulta** y la almacenamos en la variable **consulta**. Con la función **executeUpdate** le pasamos la variable **consulta**, donde se encuentra la sentencia, para realizar el borrado y guardamos el número de filas afectadas en la variable local **resultado** para mostrarlo seguidamente por pantalla.



## Salida por pantalla

```
---Programa principal---
---dbConectar---
---Conectando a Oracle---
Conexion realizada a la base de datos
oracle.jdbc.driver.T4CConnection@5b80350b
---dbObtenerPersonajes1---
Introduce ID de Personajes: 987
Nombre: Eduviges Pozo
Código postal: 10005
Sexo: M
Patrimonio: 40000.0
Clase: 2
---dbObtenerPersonajes2---
Introduce ID de Personajes: 987
SELECT NOMBRE, CP, SEXO, PATRIMONIO, CLASE FROM Personajes WHERE ID =
'987'
Nombre: Eduviges Pozo
Código postal: 10005
Sexo: M
Patrimonio: 40000.0
Clase: 2
-----
---dbConsultarPersonajes---
SELECT ID, NOMBRE, FECHANAC, CP, SEXO, PATRIMONIO, CLASE FROM
PERSONAJES
ID: 123
Nombre: Calixto Pérez
FechaNac: 1757-11-15 00:00:00.0
CP: 06300
Sexo: H
Ingresos: 150.0
Clase: 4
-----
ID: 777
Nombre: Gervasio Romualdo
FechaNac: 1721-12-12 00:00:00.0
CP: 06002
Sexo: H
Ingresos: 230.0
Clase: 4
-----
ID: 222
Nombre: Prudencio González
FechaNac: 1720-02-01 00:00:00.0
CP: 06300
Sexo: H
Ingresos: 220.0
Clase: 4
-----
ID: 333
Nombre: Macaria Gil
FechaNac: 1731-04-10 00:00:00.0
CP: 06400
```



## Universidad de Extremadura

Sexo: M

Ingresos: 195.0

Clase: 4

---

ID: 666

Nombre: Honorio Marín

FechaNac: 1742-12-12 00:00:00.0

CP: 10005

Sexo: H

Ingresos: 370.0

Clase: 4

---

ID: 555

Nombre: Venancio Fernández

FechaNac: 1738-11-15 00:00:00.0

CP: 10600

Sexo: H

Ingresos: 4600.0

Clase: 3

---

ID: 696

Nombre: Balbina Sánchez

FechaNac: 1734-04-12 00:00:00.0

CP: 06400

Sexo: M

Ingresos: 5000.0

Clase: 3

---

ID: 888

Nombre: Faustino Martínez

FechaNac: 1731-05-30 00:00:00.0

CP: 06002

Sexo: H

Ingresos: 7000.0

Clase: 3

---

ID: 999

Nombre: Facundo Fernández

FechaNac: 1739-03-12 00:00:00.0

CP: 10800

Sexo: H

Ingresos: 6300.0

Clase: 3

---

ID: 444

Nombre: Pandulfa Ruiz

FechaNac: 1750-02-01 00:00:00.0

CP: 06800

Sexo: M

Ingresos: 2500.0

Clase: 3

---

ID: 987

Nombre: Eduviges Pozo

FechaNac: 1734-05-10 00:00:00.0

CP: 10005

Sexo: M

Ingresos: 40000.0

Clase: 2



## Universidad de Extremadura

-----  
ID: 234  
Nombre: Abundio Hernández  
FechaNac: 1749-07-01 00:00:00.0  
CP: 06800  
Sexo: H  
Ingresos: 25000.0  
Clase: 2  
-----

ID: 345  
Nombre: Salustiana Moreno  
FechaNac: 1733-04-07 00:00:00.0  
CP: 10300  
Sexo: M  
Ingresos: 40000.0  
Clase: 2  
-----

ID: 567  
Nombre: Camelia Cortés  
FechaNac: 1736-05-12 00:00:00.0  
CP: 10600  
Sexo: M  
Ingresos: 48000.0  
Clase: 2  
-----

ID: 789  
Nombre: Sagrario Méndez  
FechaNac: 1759-06-22 00:00:00.0  
CP: 10800  
Sexo: M  
Ingresos: 18000.0  
Clase: 2  
-----

ID: 901  
Nombre: Amable Montero  
FechaNac: 1760-04-07 00:00:00.0  
CP: 10300  
Sexo: H  
Ingresos: 140000.0  
Clase: 1  
-----

ID: 012  
Nombre: Martiniano Zarzal  
FechaNac: 1736-11-23 00:00:00.0  
CP: 10005  
Sexo: H  
Ingresos: 360000.0  
Clase: 1  
-----

ID: 876  
Nombre: Mederica Campos  
FechaNac: 1734-03-19 00:00:00.0  
CP: 06800  
Sexo: M  
Ingresos: 500000.0  
Clase: 1  
-----

ID: 321  
Nombre: Filiberto Torres



## Universidad de Extremadura

FechaNac: 1748-08-08 00:00:00.0

CP: 06002

Sexo: H

Ingresos: 250000.0

Clase: 1

-----  
ID: 221

Nombre: Obdulia Candil

FechaNac: 1737-08-08 00:00:00.0

CP: 10005

Sexo: M

Ingresos: 502000.0

Clase: 1

-----  
---dbConsultarClases---

SELECT COD, NOMBRE, INGRESOS FROM CLASES

Codigo Clase: 1

Nombre Clase: Nobleza

Ingresos Clase: 0.0

-----  
Codigo Clase: 2

Nombre Clase: Clero

Ingresos Clase: 0.0

-----  
Codigo Clase: 3

Nombre Clase: Artesanos

Ingresos Clase: 0.0

-----  
Codigo Clase: 4

Nombre Clase: Campesinos

Ingresos Clase: 0.0

-----  
---dbInsertarClases---

Codigo clase: 0

Nombre Clase: Realeza

INSERT INTO CLASES VALUES ('0', 'Realeza', '0' )

Numero de filas afectadas: 1

---dbConsultarClases---

SELECT COD, NOMBRE, INGRESOS FROM CLASES

Codigo Clase: 1

Nombre Clase: Nobleza

Ingresos Clase: 0.0

-----  
Codigo Clase: 2

Nombre Clase: Clero

Ingresos Clase: 0.0

-----  
Codigo Clase: 3

Nombre Clase: Artesanos

Ingresos Clase: 0.0

-----  
Codigo Clase: 4

Nombre Clase: Campesinos

Ingresos Clase: 0.0

-----  
Codigo Clase: 0

Nombre Clase: Realeza

Ingresos Clase: 0.0



## Universidad de Extremadura

---dbModificarClases---

UPDATE CLASES SET NOMBRE = 'Desconocido' WHERE COD=0

Numero de filas afectadas: 1

---dbConsultarClases---

SELECT COD, NOMBRE, INGRESOS FROM CLASES

Codigo Clase: 1

Nombre Clase: Nobleza

Ingresos Clase: 0.0

-----  
Codigo Clase: 2

Nombre Clase: Clero

Ingresos Clase: 0.0

-----  
Codigo Clase: 3

Nombre Clase: Artesanos

Ingresos Clase: 0.0

-----  
Codigo Clase: 4

Nombre Clase: Campesinos

Ingresos Clase: 0.0

-----  
Codigo Clase: 0

Nombre Clase: Desconocido

Ingresos Clase: 0.0

-----  
---dbBorrarClases---

DELETE FROM CLASES WHERE COD=0

Numero de filas afectadas: 1

---dbConsultarClases---

SELECT COD, NOMBRE, INGRESOS FROM CLASES

Codigo Clase: 1

Nombre Clase: Nobleza

Ingresos Clase: 0.0

-----  
Codigo Clase: 2

Nombre Clase: Clero

Ingresos Clase: 0.0

-----  
Codigo Clase: 3

Nombre Clase: Artesanos

Ingresos Clase: 0.0

-----  
Codigo Clase: 4

Nombre Clase: Campesinos

Ingresos Clase: 0.0

-----  
---dbDesconectar---

Desconexión realizada correctamente

---Fin de programa---