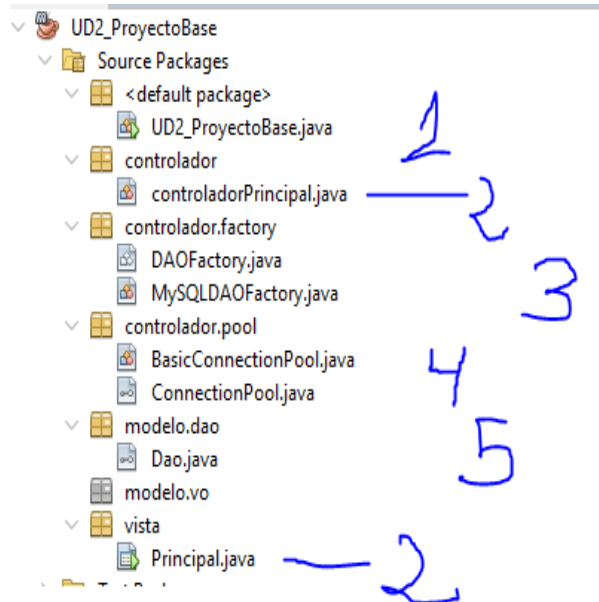


ACCESO A DATOS

UD2: MANEJO DE CONECTORES

CRITERIOS QUE UTILIZAREMOS EN LOS EJERCICIOS QUE VAMOS A PLANTEAR A LO LARGO DE LA UNIDAD.

1.- Todos los proyectos tendrán la estructura proporcionada en el proyecto base.



A continuación explicamos como vamos a trabajar con él.

1.- El proyecto se iniciará en la clase del mismo nombre del proyecto. En este caso UD2_ProyectoBase.

2.- Observa como vamos a tener un controlador por cada formulario en la vista.

3.- En el factory es donde vamos a establecer los parámetros de nuestra conexión. El DaoFactory es una clase abstracta general que concretamos para MySQL en el MySQLDAOFactory.

Los parámetros vendrán dados por

```
final static String user = "root";
final static String password = "root";
final static String BD = ""; //Indica aqui la BD
final static String IP = ""; //Indica aqui la IP
final static String url = "jdbc:mysql://" + IP + ":3306/" + BD;
```

Tal y como está el proyecto base está preparado para conectar solo a Mysql, pero si nos conectáramos a una base de datos con otro gestor como por ejemplo Mariadb, tendríamos que añadir esa opción primero en el DAO

ACCESO A DATOS

UD2: MANEJO DE CONECTORES

```
public abstract class DAOFactory {  
    // List of DAO types supported by the factory  
    public static final int MYSQL = 1;  
    public static final int MariaDB = 2;  
    public static DAOFactory getDAOFactory(  
        int whichFactory) {  
        switch (whichFactory) {  
            case MYSQL:  
                return new MySQLDAOFactory();  
            case MariaDB:  
                return new MariadbDAOFactory();  
            default:  
                return null;  
        }  
    }  
}
```

e implementar la clase MariadbDAOFactory y en esa clase cambiar la url a

```
final static String url = "jdbc:mariadb://" + IP + ":3306/" + BD;
```

4.- En el pool es donde se implementa como vamos a trabajar con la conexión a base de datos. En nuestro caso un pool de conexiones.

El pool de conexiones consiste en tener un conjunto (pool) de conexiones ya conectadas a la base de datos que puedan ser reutilizadas entre distintas peticiones. En nuestro caso base tenemos un pool máximo de 10 y al iniciar la aplicación abrimos las 10.

```
private static final int MAX_POOL_SIZE = 10;  
private static final int MAX_TIMEOUT = 1000;  
private String url;  
private String user;  
private String password;  
private List<Connection> connectionPool;  
private List<Connection> usedConnections = new ArrayList<>();  
private static int INITIAL_POOL_SIZE = 10;  
private BasicConnectionPool bcp = null;
```

5.- En el modelo es donde vamos a implementar la operativa a realizar en la base de datos.

Iremos viendo su funcionalidad a medida que avancemos en nuestros proyectos.

ACCESO A DATOS

UD2: MANEJO DE CONECTORES

Pasos previos a realizar siempre que inicies el proyecto.

Se enumeran aquí los pasos que debes realizar para conectar a la base de datos.

1.- Debes **añadir la dependencia** necesaria en el fichero pom.xml para poder acceder a mysql.

```
<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.33</version>
  </dependency>
</dependencies>
```

2.- En el **factory** vamos a **establecer a que base de datos nos vamos a conectar, la ip** donde está y que tipo de operaciones vamos a realizar con la conexión a la misma. En nuestro proyecto base lo tenemos preparado para conectar a MySQL. En la clase MySQLDAOFactory indicamos la base de datos y la ip (este valor puede variar en función de donde tengas instalada la base de datos).

```
final static String user = "root";
final static String password = "root";
final static String BD = "ejemplo"; //Indica aqui la BD
final static String IP = "192.168.56.117"; //Indica aqui la IP
final static String url = "jdbc:mysql://" + IP + ":3306/" + BD;
```

3.- Crearemos por cada tabla afectada de la base de datos con la que vayamos a trabajar 2 clases. Supongamos que trabajamos con la tabla departamentos. Crearemos 2 clases DepartamentoDAO y departamento. Posteriormente en la clase DAOFactory debes añadir los métodos abstractos por cada DAO que puedan crearse. Estos métodos se implementarán en la factory que utilizemos. En concreto en la factory de mysql.

```
//En el DAOFactory debes crear los métodos por cada dao que se vaya a utilizar
//Se implementarán en el factory a utilizar. En nuestro caso Mysql.
public abstract DepartamentoDAO getDepartamentoDAO();
// public abstract EmpleadoDAO getEmpleadoDAO();
```

4.- En el MysqlFactory realizamos la implementación del método.

```
@Override
public DepartamentoDAO getDepartamentoDAO() {
    return new DepartamentoDAO();
}
```

5.- Diseñas la pantalla y crear los métodos set de los elementos de la pantalla con los que vayas a interactuar en el controlador. (Serán los cuadros de texto, los textArea, los combos,...)

6.- En el evento correspondiente se hace la llamada al controlador.

ACCESO A DATOS

UD2: MANEJO DE CONECTORES

MÉTODOS MÍNIMOS A IMPLEMENTAR EN EL CONTROLADOR

public static void iniciar() : Es el método para iniciar el formulario.

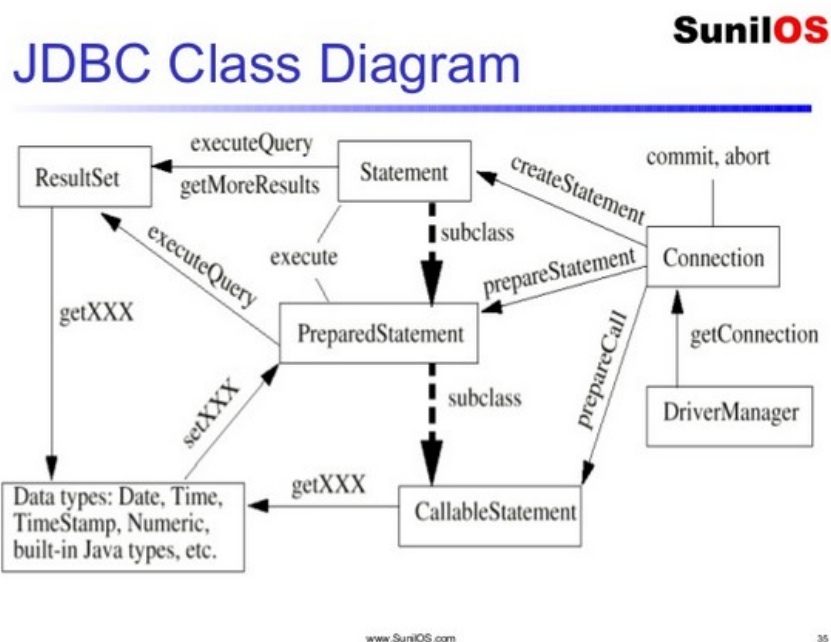
public static void iniciaFactory() : Es el método para iniciar los parámetros básicos para conectar a la base de datos. Aquí iniciaremos el objeto factory y lo DAO implicados.

public static void cerrarFactory(): Es el método para cerrar el factory.

Nota: Nuestras aplicaciones serán en un formulario básico. Con lo que estos métodos los haremos para cada formulario. Si fuera una aplicación más compleja con distintos formularios se iniciarían el factory una vez por aplicación.

MÉTODO DE TRABAJO. Importante

- 1.- Iniciar el factory incluyendo los DAO afectados.
- 2.- Trabajamos con la BD. (consultamos, insertamos, actualizamos, borramos).
- 3.- Cerrar el factory.



ENUNCIADO 1: Primera Conexión a BD. Listar los datos del departamento en un textArea.

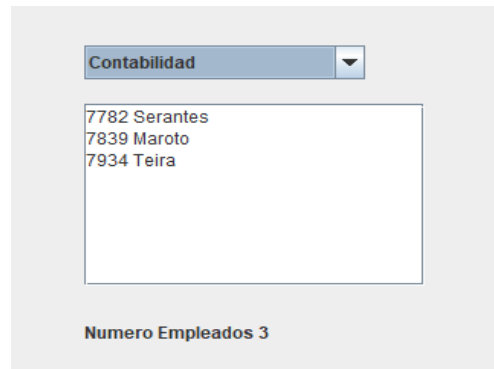
Crea un formulario que establezca una conexión a la base de datos para listar los departamentos en un textArea.

ACCESO A DATOS

UD2: MANEJO DE CONECTORES

ENUNCIADO 2: Mostrar empleados por departamento.

Crea un combo de objetos departamento donde solo se visualizaran los departamentos. Una vez eliga en el combo el departamento mostrará los empleados del departamento correspondiente además del número de empleados que tiene.



Observación: Con los datos iniciales hay un departamento (Producción) que no tiene empleados. Puedes dejar el textArea sin empleados o bien indicar en el textarea que no tiene empleados (esta segunda versión es un poco más compleja).

Recuerda:

Es importante que inicies el factory y lo cierres. Lo podemos iniciar en el constructor del formulario y cerrarlo en el evento closed.

El combo es de objetos, para ello debes indicar en el tipo de parámetros la clase que vas a cargar.

Type Parameters <Departamento>

Y vaciar el modelo

model		...
selectedIndex	-1	...
selectedItem	null	...

En el controlador crearás el modelo y lo asignarás al combo.

```
static DefaultComboBoxModel modelocombo = new DefaultComboBoxModel();

public static void iniciar() {
    ventana.setVisible(b: true);
    ventana.setLocationRelativeTo(c: null);
    //Como hay un combo en el iniciar el formulario asignamos el modelo al combo
    ventana.getCmbDepartamento().setModel(aModel: modelocombo);
}
```

Recuerda que en combo se muestra el toString del modelo.

ACCESO A DATOS

UD2: MANEJO DE CONECTORES

En los eventos del formulario llamas al controlador.

En el controlador siempre conectamos, operamos y desconectamos.

Todas las operaciones se hacen en los DAO correspondientes.

Si las operaciones no tienen parámetros utilizaremos `createStatement()` , si tiene parámetros `PreparedStatement`.

ENUNCIADO 3: Manejo de Departamentos

Crea un formulario que permita introducir datos de los departamentos.

Nº Departamento	<input type="text"/>	<input type="button" value="Insertar"/>
Nombre	<input type="text"/>	<input type="button" value="Borrar"/>
Localidad	<input type="text"/>	<input type="button" value="Modificar"/>

Aunque la idea es que en ejercicios posteriores podamos insertar, borrar y modificar departamentos. En este primer ejercicio lo que harás es buscar los datos del departamento a partir del número de departamento en su `lostfocus`. De tal manera que:

- 1.- Si se introduce un departamento existente carga los datos cuando el cuadro de texto pierde el foco.
- 2.- Si se introduce un departamento no existente deja en blanco los valores.
- 3.- Si no se introduce nada lo indica y deja en blanco los valores.
- 4.- Si se introduce un valor incorrecto lo indica. (Ejemplo abc cuando debe introducir un valor numérico) y deja en blanco los valores

Prueba su funcionamiento con estos valores

numero:10 (existe el departamento y carga los datos)

numero =60 (departamento no existente y deja en blanco los campos)

numero vacio: (indica que faltan datos)

numero = abc (valor incorrecto)