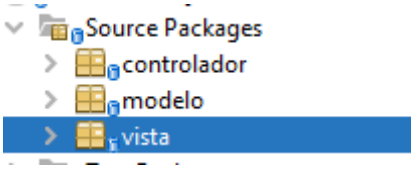
ACCESO A DATOS
UD2: MANEJO DE CONECTORES

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

1.-Todos los ejercicios tendrán como mínimo 3 paquetes:



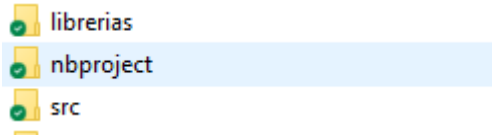
Iremos viendo a medida que avancemos que es lo que indicamos en cada paquete.

2.- Las bases de datos con las que vamos a trabajar se proporcionan en los ejercicios.

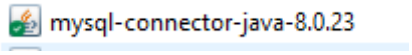
3.- Para establecer la conexión a la BD hay que utilizar el driver JDBC (Java DataBase Connectivity) correspondiente. Este driver se va a corresponder con un .jar. Nostros trabajaremos básicamente con el de mysql o el de mariadb indistintamente.

En concreto serán mysql-connector-java-8.0.23.jar para mysql, mariadb-java-cliente-2.6.2.jar para mariadb.

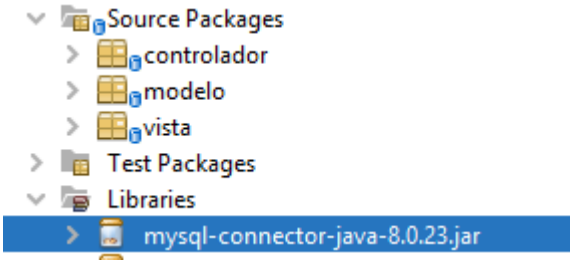
El jar tendrás que incorporarlo al proyecto cada vez que lo inicies. Como criterio crear una carpeta librería al nivel de src



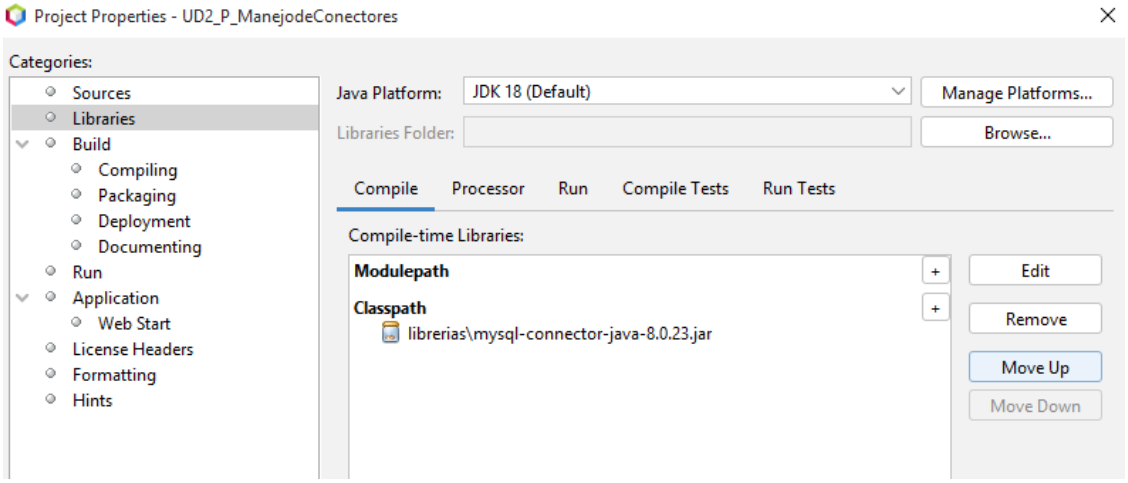
y dentro incorporarás el jar.



A continuación añade ese jar a la librería del proyecto



Debe quedar algo tal que así



4.- Trabajaremos con la clase java.sql.*;

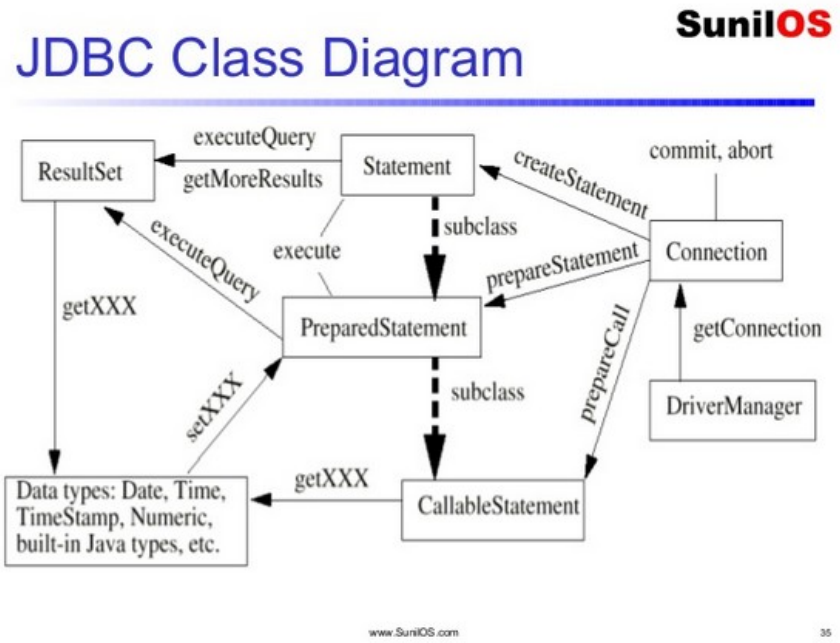
5.- Para trabajar con BD es necesario tres pasos básicos:

1.- Conectar a la BD.

2.- Trabajamos con la BD. (consultamos, insertamos, actualizamos, borramos)

3.- Desconectamos de la BD.

6.- Este diagrama muestra como vamos a trabajar con la BD. Iremos viendo como funciona a lo largo de la unidad.



ACCESO A DATOS
UD2: MANEJO DE CONECTORES

ENUNCIADO 1: Conexion a base de datos : mariadb. CONSOLA

Realiza un programa que permita conectarse a una BD mariadb y realizar una consulta cualquiera.

Observa que en el punto 1. a través del DriverManager establecemos la conexión a la base de datos indicándole en la url que va a ser a mariadb. Como parámetros tiene

IP:192.168.56.101
BD: ejemplo
Usuario: root
Contraseña:root
puerto: normalmente es el 3306 si estuviera en otro puerto tendrías que poner el valor correspondiente.

```
package vista;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;

public class ConectarMariaDB {

    public static void main(String[] args) {

        try {
            Class.forName(("org.mariadb.jdbc.Driver"));
            try (Connection conexion = DriverManager.getConnection("jdbc:mariadb://192.168.56.101:3306/ejemplo", "root", "root");
                 Statement sentencia = conexion.createStatement()) {

                String consulta = "select * from empleados";
                try (ResultSet resul = sentencia.executeQuery(consulta)) {
                    while (resul.next()) {
                        System.out.println(resul.getInt(1) + " " + resul.getString(2) + "----" + resul.getString("oficio"));
                    }
                }
            }
        } catch (ClassNotFoundException | SQLException ex) {
            JOptionPane.showMessageDialog(null, "Error en la conexión a la base de datos");
        }
    }
}
```

Nota: En este caso la conexión la hace en el punto 1, y la desconexión por como está programado al salir del try ya la cierra.

En el punto 2 recogemos la consulta en un **resultset** y mostramos los datos recorriendo ese resulset. Observa que tienes que sacar cada campo de la tabla atendiendo al tipo. Puedes indicarlo bien por numero de campo o por nombre del campo (fijate en oficio).

ACCESO A DATOS
UD2: MANEJO DE CONECTORES

ENUNCIADO 2: Conexion a base de datos : mySQL. CONSOLA.

Realiza un programa que permita conectarse a una BD mysql y realizar una consulta cualquiera.

Observa que en el punto 1. a través del DriverManager establecemos la conexión a la base de datos indicándole en la url que va a ser a mysq1l. Como parámetros tiene

IP:192.168.56.101
BD: ejemplo
Usuario: root
Contraseña:root

```
public class ConectarMySQL {

    public static void main(String[] args) {
        Connection conexion = null;
        ResultSet resul=null;
        Statement sentencia=null;

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");
            conexion = DriverManager.getConnection("jdbc:mysql://192.168.56.101:3306/ejemplo", "root", "root");

            sentencia = conexion.createStatement();

            String consulta = "select * from empleados";
            resul = sentencia.executeQuery(consulta);

            while (resul.next()) {
                System.out.println(resul.getInt(1) + " " + resul.getString(2) + "----" + resul.getString("oficio"));
            }
        } catch (ClassNotFoundException | SQLException ex) {
            JOptionPane.showMessageDialog(null, "Error en la conexión a la base de datos");
        }finally{
            try {
                sentencia.close();
                resul.close();
                conexion.close();
            } catch (SQLException ex) {
                Logger.getLogger(ConectarMySQL.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

    }
```

Nota: En este caso la conexión la hace en el punto 1

En el punto 2 recogemos la consulta en un **resultset** y mostramos los datos recorriendo ese resulset. Observa que tienes que sacar cada campo de la tabla atendiendo al tipo. Puedes indicarlo bien por numero de campo o por nombre del campo (fijate en oficio).

En el punto 3 finalmente cierra la conexión y libera los recursos de memoria. Es importante que lo hagas en el finally para que lo haga siempre seguro.

ACCESO A DATOS
UD2: MANEJO DE CONECTORES

ENUNCIADO 3: Conexion a base de datos : Oracle. CONSOLA

Realiza un programa que permita conectarse a una BD oracle y realizar una consulta cualquiera.

En este caso la conexión sería:

```
Class.forName("oracle.jdbc.driver.OracleDriver");
conexion=DriverManager.getConnection("jdbc:oracle:thin:@192.168.56.11:1521:xe","SYSTEM","admin");
```

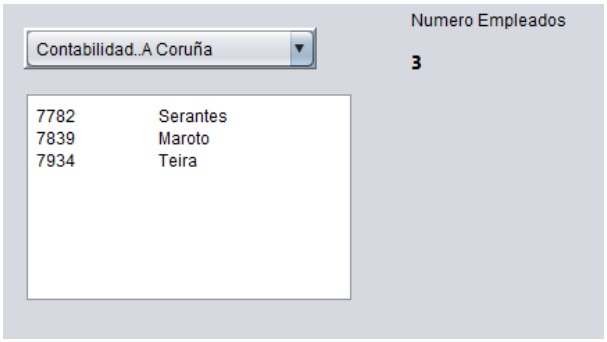
y el código idéntico a los anteriores.

ENUNCIADO 4: Conexion a base de datos (Pool) : mySQL. GRAFICO

Este va a ser el primer ejemplo de como vamos a trabajar realmente. Utilizando el pool de conexiones.

La clase Pool.java se te proporcionará siempre para los exámenes junto con la BD.

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



Aclaraciones a la solución del enunciado 4:

MUY IMPORTANTE

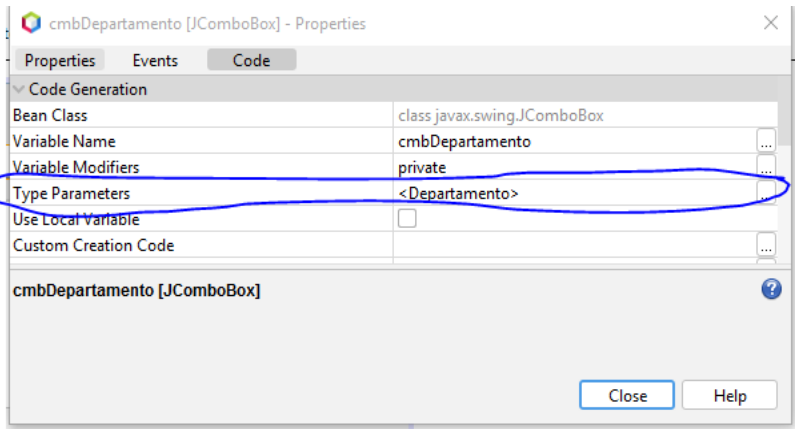
Al iniciar el formulario que inicia la aplicación tenemos que iniciar el pool de conexiones.

```
try {
    Pool.IniciaPool();
    initComponents();
```

Y cargamos el combo

```
DefaultComboBoxModel<Departamento> cmbBoxModel = new DefaultComboBoxModel<>();

/**
 * Creates new form FrmPrimero
 */
public Enunciado4() {
    try {
        Pool.IniciaPool();
        initComponents();
        cmbDepartamento.setModel(cmbBoxModel);
        GestionDepartamento.cargarCombo(cmbDepartamento);
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error al iniciar");
    } finally {
        Pool.Cerrar(); //Cerramos por iniciar la conexion en el combo
    }
}
```

A comentar: Para cargar en el combo los departamentos. Tienes que cargar objetos, y este mostrará lo que tenga en el método toString el modelo departamento.



```
@Override
 public String toString() {
     return this.getNombredep()+".."+this.getLoc(); //Enunciado4
 }
```

Para el combo le hemos puesto en la vista que visualice los empleados cuando haya un item seleccionado. Para evitar que rompa el programa incorporar en el cargarCombo al principio un null y al final le quitais el item 0.

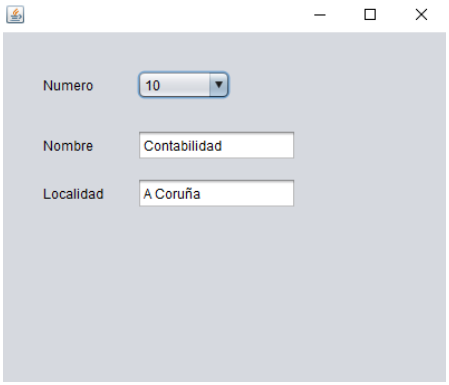
```
public static void cargarCombo(JComboBox<Departamento> cmbDepartamento) {
    Departamento d;
    try {
        String consulta = "Select * from departamentos";
        Statement sentencia = Pool.getCurrentConexion().createStatement();

        ResultSet rs = sentencia.executeQuery(consulta);
        cmbDepartamento.addItem(null);
        while (rs.next()) {
            d = new Departamento(rs.getInt(1), rs.getString(2), rs.getString(3));
            cmbDepartamento.addItem(d);
        }
        //Si quisieramos quitarle el nulo inicial del combo
        //indicamos
       // cmbDepartamento.removeItemAt(0);
        sentencia.close();
        rs.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error al cargar el combo", "Error", JOptionPane.ERROR_MESSAGE);
    } finally {

    }
}
```

Además incorporamos en la vista que muestre los datos cuando seleccione un item y que este sea distinto de null. Asi evitamos que rompa el programa.

```
private void cmbDepartamentoActionPerformed(java.awt.event.ActionEvent evt) {
    //Como criterio para la gestion de errores lo vamos a  hacer en la vista.
    try {
        if (cmbDepartamento.getSelectedItem() != null) {
            GestionDepartamento.listarEmpleadosycuenta((Departamento) cmbDepartamento.getSelectedItem(), txtArea, lblnumero);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error en la conexión a la base de datos");
    }finally{
        Pool.Cerrar();
    }
}
```

ENUNCIADO 5: Manejo de formulario 1

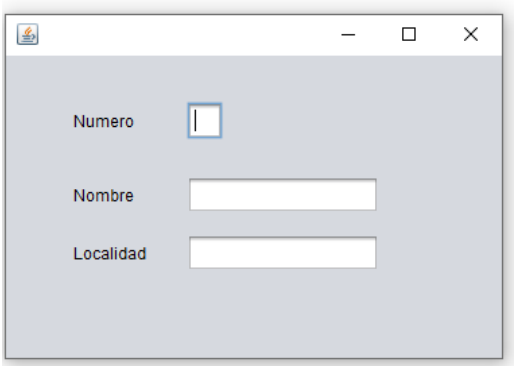
Crea un combo de objetos departamento donde solo se visualizará el número de departamento. Una vez eliga en el combo el número de departamento mostrará los datos del departamento correspondiente.



Aclaraciones a la solución del enunciado 5: Es un ejercicio muy pequeño ,pero podeis ver en el inicio de la aplicación como iniciaria el pool con el fichero. Tambien recordad que para cambiar lo que muestra el combo, teneis que cambiar el toString en el modelo .

ENUNCIADO 6: Manejo de formulario 2

Crea un formulario



Que debe funcionar tal y como se indica:

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 se introduce un valor incorrecto lo indica. (Ejemplo abc cuando debe introducir un valor númerico) y deja en blanco los valores
4.- Si no se introduce nada lo indica y deja en blanco los valores.

Con el jar que se te proporciona prueba su funcionamiento con estos valores

numero:10 (existe el departamento y carga los datos)
numero vacio: (indica que faltan datos)
numero = abc (valor incorrecto)
numero =60 (departamento no existente y deja en blanco los cambpos)