Módulo: ACCESO A DATOS (AD)
I.E.S.: Aguadulce **C.F.G.S. : Desarrollo de Aplicaciones Multiplataforma (DAM)**
Fecha: 9 de Febrero de 2017 **Hora de Comienzo: 15:30**
Examen solución : Práctica **Duración:**
Nombre: **D.N.I.:**
Centro en el que se realiza el examen:

EJERCICIO 1: (2 puntos) Utilizando JAXB debes completar el siguiente código de Java con las líneas que faltan para que se pueda leer el fichero datos_centro.xml, se pueda modificar los datos de la dirección del centro como se muestra en el programa. Por último se debe actualizar el fichero xml y obtener por consola del IDE el resultado en formato XML. Ya se han compilado y generado las clases a partir del fichero esquema y están dentro del paquete jaxb.centro listas para ser utilizadas. También se ha validado el fichero XML. Sólo tienes que añadir las líneas de código necesarias y en el orden correcto para que el programa funcione correctamente. No hace falta que añadas comentarios en las líneas de código.

try {

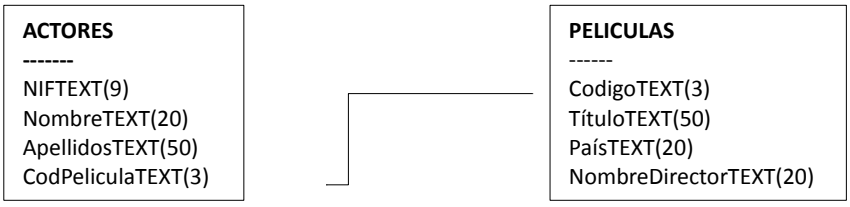
```
//Crear una instancia para poder manipulas las clases generadas, que están en el paquete jaxb.centro
//Inserta aquí el código adecuado.

JAXBContext jaxbContext= JAXBContext.newInstance("jaxb.centro")



// Crear un objeto de tipo Unmarshaller para convertir datos XML en un árbol de objetos Java .
//Inserta aquí el código adecuado.
Unmarshaller u=jaxbContext.createUnmarshaller();



// La clase JAXBElement representa a un elemento de un documento XML en  este caso a un elemento del documento datos_centro.xml
//Inserta aquí el código adecuado.

JAXBElement  jaxbElement= (JAXBElement) u.unmarshal(newFileInputStream("datos_centro.xml")
```

```
// El método getValue() retorna el modelo de contenido (contentmodel) y el valor de los atributos del elemento
        CentroTypecentroType = (CentroType) jaxbElement.getValue();

// Obtenemos una instancia de tipo CentroType para obtener un Objeto de tipo Direccion
        Direcciondireccion = centroType.getDirigidoA();
// Establecemos los nuevos datos del centro
        direccion.setNombre("IES Aguadulce");
        direccion.setCalle("Alhambra, 11");
        direccion.setCiudad("Aguadulce");
        direccion.setProvincia("Almería");
```

```
// Crear un objeto de tipo Marshaller para posteriormente convertir el  árbol de objetos Java a datos XML
//Inserta aquí el código adecuado.
Marshaller m=jaxbContext.createMashaller();



/ /Crear el resultado XML no formateado para lectura de las personas,  con saltos de línea, etc.
//Inserta aquí el código adecuado.
m.setproperty(Marshaller.JAXB_FORMATTED_OUTPUT,boolean.TRUE)
// Escribir el elemento obtenido como primer parámetro por la salida estándar.
//Inserta aquí el código adecuado.
  m.marshal(jaxbElement, System.out)


//actualizar el fichero XML datos_centro.xml
//Inserta aquí el código adecuado.
m.marshal(jaxbElement,new  FileOutStream("datos_centro.xml")
```

```
        } catch (JAXBException je) {
                System.out.println(je.getCause()) ;
        } catch (IOExceptionioe) {
                System.out.println(ioe.getMessage());
}
```

EJERCICIO 2: (4 puntos). Se ha creado una base de datos en Access llamada films.accdb colocado en la carpeta DATOS de nuestro proyecto, que contiene las tablas ACTORES y PELÍCULAS con la especificación de campos que se detallan en la siguiente figura.



Se quiere trabajar con JDK8 con un driver JDBC puro y se han instalado las librerías de UCanAccess. Ya está todo preparado para ser usado desde la aplicación.

En cuanto a la aplicación Java, el equipo de proyecto en el que trabajas ya ha realizado parte de la aplicación y te han pedido que te encargues de realizar los cuatro métodos siguientes utilizando las plantillas que encontrarás al final del enunciado.

A. **Cargar el driver y obtener la conexión de la base de datos.** ***El programa debe funcionar tanto en Windows como en Linux.***
B. **Insertar un actor utilizando tus datos personales y en el campo CodPelícula el valor 2B**
C. **Obtener un listado con el nombre y los apellidos de los actores que han participado en películas cuyo país sea España.**
D. **Actualizar el campo NombreDirector (poniendo como nuevo valor = "Lola Del Amor Gómez") para la película cuyo código es 2B. Utiliza para ello consultas preparadas.**

ACLARACIONES:

- **No hay que hacer un programa completo: las librerías necesarias ya están importadas, el menú de opciones ya está codificado y las variables static(*) listas para ser usadas.**
- **Sólo es necesario completar con las líneas de código necesarias las plantillas de los métodos que**

encontrarás al final del enunciado, para su correcto funcionamiento.

- **Es importante controlar el manejo de excepciones correctamente.**
- **Los resultados de las consultas y/o mensajes de actualizaciones serán mostrados por la consola del IDE, no es necesario utilizar ningún componente para mostrarlos.**

PLANTILLAS PARA LA CODIFICACIÓN DE CADA UNO DE LOS MÉTODOS DE LOS APARTADOS A,B,C y D RESPECTIVAMENTE.

APARTADO A:

```
public static Connection obtenerConexion()
{
  try {
  con=DriverManager.getConnection("jdbc:ucanaccess://DATOS/films.accdb");
  } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
  }
return con;
}
```

APARTADO B:

```
public static void insertarActor()
{
   con = obtenerConexion();
   String insertString;
  insertString = "insert into Actores values('00000000X', 'Nombre', 'Apellido', '2B')";
  try {
            stmt = con.createStatement();
            stmt.executeUpdate(insertString);
            System.out.println("Actor insertado correctamente");

            stmt.close();
            con.close();

 } catch(SQLException ex) {
    System.err.println("SQLException: " + ex.getMessage());
  }

}
```

APARTADO C:

```
public static void obtenerActores()
{
   con = obtenerConexion ();
String result = null;  String selectString;
selectString = "select A.Nombre, A.Apellido from Actores A, Peliculas  P where A.CodPelicula = P.Codigo and P.pais='España'";
result ="Nombre\t \t Apellidos\n";
try {
            stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(selectString);
while (rs.next()) {
            String nombre = rs.getString("Nombre");
            String apellidos = rs.getString("Apellidos");
            result += nombre + "\t \t" + apellidos+"\n";
} System.out.println(result);
 stmt.close();
 con.close();
} catch(SQLException ex) {
System.err.println("SQLException: " + ex.getMessage());
}
```

}

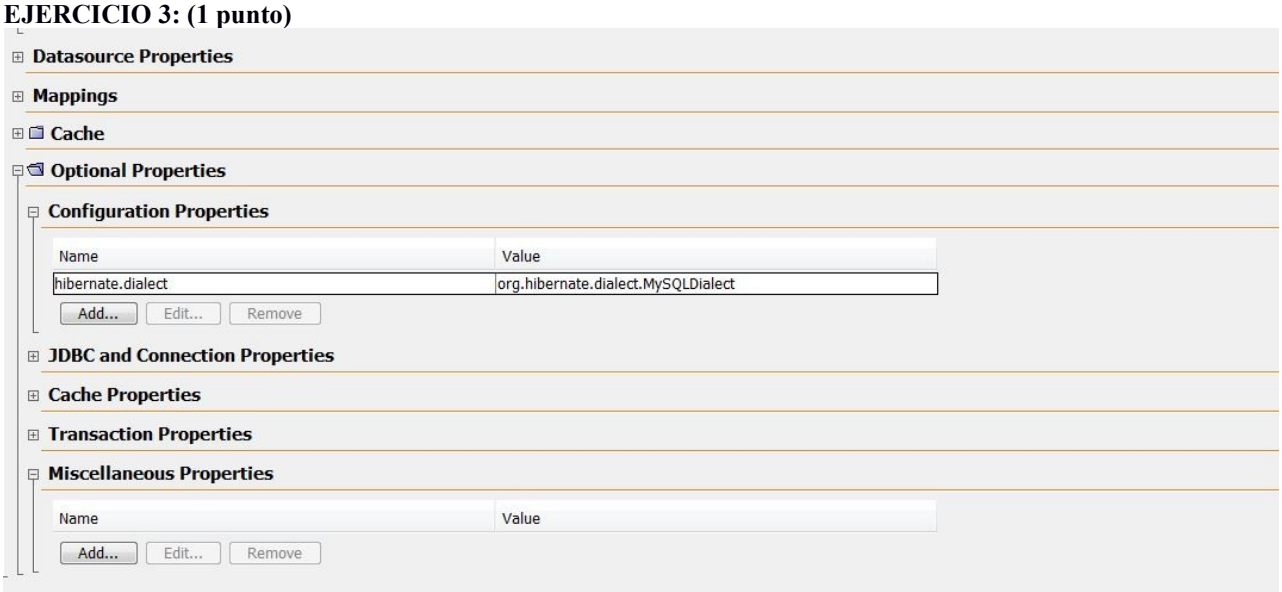
APARTADO D:

```
public static void actualizarNombreDirector()
{
  con = obtenerConexion ();
  try {
            pstmt = con.prepareStatement ("update Peliculas set NombreDirector = ? where Codigo = ?");
            pstmt.setString(1, “Lola Del Amor Gomez”);
            pstmt.setString(2, "2B");
            pstmt.executeUpdate();
            System.out.println("El registro se ha actualizado con éxito");
            pstmt.close();
            con.close();
} catch(SQLException ex) {
                        System.err.println("SQLException: " + ex.getMessage());
 } finally {
            pstmt.close();
           con.close();    }
}
```

(*) Las siguientes variables están declaradas para poder ser usadas en cualquier método.

```
static Statement stmt;
static PreparedStatement pstmt;
static Connection con;
```

EJERCICIO 3: (1 punto)



Se ha instalado la base de datos MySQL Films en NetBeans. Se ha creado el archivo de configuración *Hibernate.cfg.xml*, se trata de que indiques qué propiedades habría que añadir o configurar para que todo fuese correcto. Las propiedades JDBC y las referidas a la conexión ya se encuentran correctamente configuradas. (Puedes realizarlo sobre la imagen en el lugar correspondiente).

Habría que añadir en :

Configuration properties:

hibernate.show.sql ***TRUE***

Miscellaneous Properties

hibernate.query.factory.class ***org.hibernate.hql.internal.classic.ClassicQueryTranslatorFactory***