NOMBRE Y APELLIDOS:

CALIFICACIÓN

FECHA: 18/12/2024 **CURSO**: 2º DAM

MÓDULO: Acceso a datos (AAD)

UNIDADES DIDÁCTICAS: UD3

Examen 2ª Evaluación (120 min)

Instrucciones

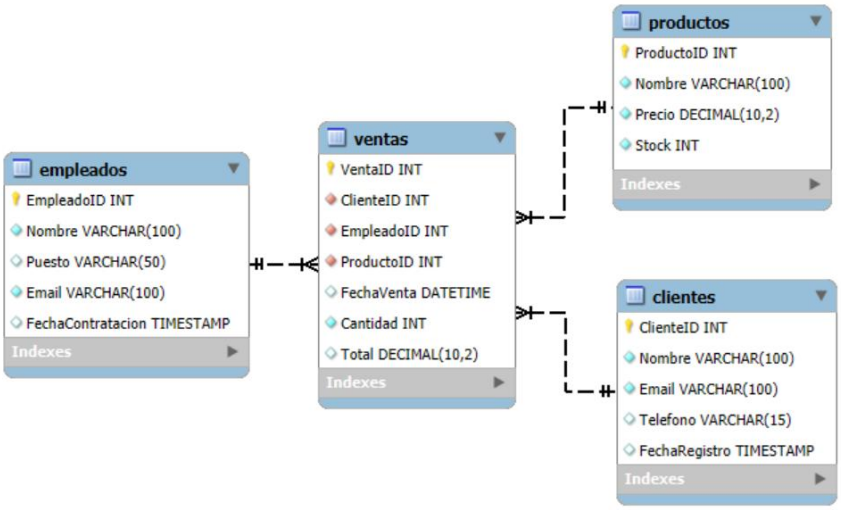
No se permite el uso de ningún material de consulta ni el acceso a Internet. Si se detecta el acceso a código de proyectos previos, acceso a internet o cualquier tipo de ayuda externa, la calificación del examen será 0.

Antes de comenzar el examen, el alumno debe:

1. Leer el enunciado del examen completo.
2. Descargar los recursos del AV necesarios para la realización del examen.
3. Desconectar el latiguillo de red de su PC.
4. Cerrar todos los proyectos abiertos del Eclipse (desde la sección "package explorer" -> botón derecho sobre cada proyecto -> close Project)
5. Una vez cerrados todos los proyectos, se deberá aplicar un filtro para ocultar los proyectos cerrados (desde la sección "package explorer" -> View menu -> Filters -> seleccionar "closed projects")

Ejercicios

Crea un proyecto Maven llamado AAD_EX01_JDBC_NomApe1 (NomApe1 es tu nombre y primer apellido) donde desarrolles un programa en JAVA que que permita realizar las siguientes interacciones con la BD de un **crm** desplegada en un SGBD relacional (MySQL).



Para ello usa una clase es.aad.**Programa** con el *main*, una clase es.aad.**DbManager** que realice la interacción con la base de datos proporcionada (crm) y que obligatoriamente debe implementar la interface com.city.ies.**RequisitosExamen** y por tanto implementar los métodos indicados en dicha interface. Además, también se debe usar la clase proporcionada es.aad.**Cliente**, sin realizar cambio alguno.

La conexión con la BD debe realizarse mediante el uso de los parámetros indicados en un fichero properties similar al siguiente:

```
driver=com.mysql.cj.jdbc.Driver
url=jdbc:mysql://localhost:3306/crm
user=root
password=root
```

La aplicación deberá permitir realizar 3 de las 4 siguientes operaciones. Obligatoriamente se deben implementar las dos primeras:

1) Dar de alta todos los productos del proveedor IBM de forma transaccional. Para ello, cada uno debe guardarse usando el método aniadirProducto **(3 puntos)**

Nombre	Precio	Stock
Teclado IBM	12.30€	10
Ratón IBM	6.75€	25
Trackpad IBM	27.95€	5

2) Permitir al usuario consultar por cada cliente existente, el total de ventas. Para ello se debe usar el método getTotalVentas que debe invocar a la función almacenada en BD TotalVentasPorCliente y la salida debe volcarse a LOG llamado igual que el proyecto y con el siguiente formato: **(3 puntos)**

Cliente [nombre=" + nombre + ", email=" + email + ", telefono=" + telefono + ", fechaRegistro=" fechaRegistro + "] con un total de [XX.XX] €"

3) ¡Nos han confirmado que los totales de cada venta almacenados son incorrectos!
   Debes actualizar el importe total de cada una de las ventas de acuerdo al precio unitario del correspondiente producto y al número de ellos incluido en cada venta (cantidad). Todas las actualizaciones se deben realizar bajo una misma transacción. Para ello se debe usar el método actualizarTotales (**4 puntos**)

4) Actualizar el stock de cada producto vendido. Para ello se debe usar el método reducirStockProductos que debe invocar al procedimiento almacenado en BD DecrementarStock. Cada producto se deberá actualiza bajo una transacción. **(4 puntos)**

Nota: Deberás comentar aquel método que no implementes de la interfaz en función de si realizas el ejercicio 3 o 4.

Criterios de evaluación

El examen tiene una calificación máxima de 10 puntos y para obtener la máxima puntuación en cada apartado, se tendrá en cuenta:

- No hay errores de compilación.
- Cumple con los requisitos funcionales del examen:
  - Se implementa correctamente la interface proporcionada.
  - Se gestionan correctamente las transacciones solicitadas.
  - Los valores calculados son exactos.
  - Se actualizan correctamente los registros de la base de datos.
  - Se utiliza correctamente los recursos de la base de datos (conexión, statements, setencias sql, resultsets, etc)
- Se entrega en el Aula virtual toda la información requerida, en el plazo establecido y con la nomenclatura y formato exigido.

- Se hace uso de las **buenas prácticas de programación** (evitando hardcodes, parametrizando la aplicación, optimizando el uso de recursos, modularizando funcionalidades en métodos, reutilizando los recursos siempre que sea posible, organizando en clases la información y funcionalidades…)
- Se gestionan correctamente las posibles **excepciones** que se puedan disparar en el programa evitando su propagación desde el método main.
- Se vuelcan sobre **LOG** los hitos fundamentales de cada ejercicio, así como los datos obligatorios solicitados en cada enunciado haciendo uso de los distintos niveles de LOG (**debug**, **info, error, etc**) para facilitar futuras correcciones o cambios de alcance en la funcionalidad de los mismos.

La calificación de dicho examen se tendrá en consideración para la evaluación de los siguientes **Resultados de Aprendizaje** y **Criterios de Evaluación**:

RA2. Desarrolla aplicaciones que gestionan información almacenada en bases de datos relacionales identificando y utilizando mecanismos de conexión.
a) Se han valorado las ventajas e inconvenientes de utilizar conectores.
b) Se han utilizado gestores de bases de datos embebidos e independientes.
c) Se ha utilizado el conector idóneo en la aplicación.
d) Se ha establecido la conexión.
e) Se ha definido la estructura de la base de datos.
f) Se han desarrollado aplicaciones que modifican el contenido de la base de datos.
g) Se han definido los objetos destinados a almacenar el resultado de las consultas.
h) Se han desarrollado aplicaciones que efectúan consultas.
i) Se han eliminado los objetos una vez finalizada su función.
j) Se han gestionado las transacciones.
k) Se han ejecutado procedimientos almacenados en la base de datos.

Entrega

El/la alumno/a entregará en la tarea habilitada del Aula Virtual de la asignatura tres ficheros:

1. Un fichero comprimido **en formato jar** donde se incluya el código fuente.

2. Otro fichero comprimido en **formato jar ejecutable** con los binarios, dependencias y ficheros de configuración (logback.xml, librerías, etc).
3. El fichero **LOG** generado.

Para generar los ficheros JAR debes adaptar tu pom.xml de Maven de acuerdo al fichero pom.xml que tienes disponible en el AV.

Actualiza el proyecto.

Desde **Maven Build**, indica como **Goals** *clean package* para generar los ficheros jar. Dichos ficheros los encontrarás en el directorio target del proyecto.

Tips & Tricks

Puedes apoyarte en la siguiente sección de código para recuperar información de un fichero Properties:

```
Properties fichProperties = new Properties();
fichProperties.load(new FileInputStream(<variable File>));
String valor = fichProperties.getProperty(<clave>);
```

Puedes apoyarte en el siguiente código para la creación del LOGGER:

```
private static final Logger LOGGER = LoggerFactory.getLogger(Principal.class);
```

Puedes apoyarte en la siguiente sección de código para gestionar la invocación de funciones y procedimientos almacenados en la BD relacional:

```
public static final String invocacion_nombre_de_la_funcion  = "{ ? = call nombre_de_la_funcion(?,?,?)}";
public static final String invocacion_nombre_del_procedimiento= "{call nombre_del_procedimiento(?,?,?)}";
```

Puedes apoyarte en el siguiente código para establecer conexión con la BD relacional:

```
Class.forName(<driver del SGBD>);
Connection conexion=DriverManager.getConnection(<url BD>,<usuario BD>,<contraseña BD>);
```