NOMBRE Y APELLIDOS:

FECHA: 20/11/2024 **CURSO**: 2º DAM

MÓDULO: Acceso a datos (AAD)

CALIFICACIÓN

Examen 1ª Evaluación (150 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")

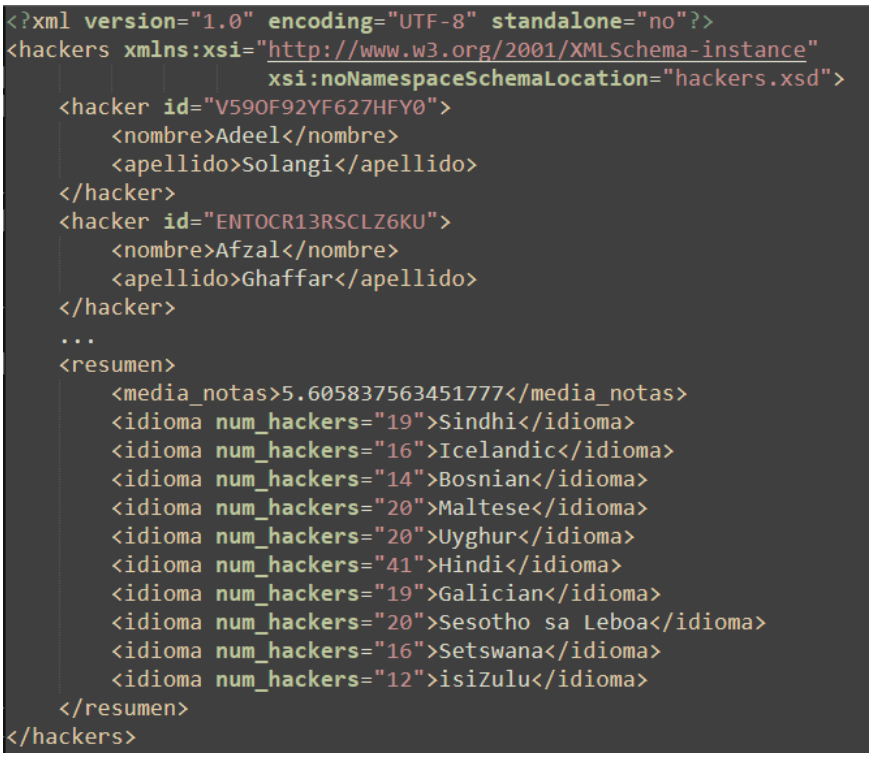
Ejercicios

1. Crea un proyecto Maven llamado AAD_EX01_NomApe1 (NomApe1 es tu nombre y primer apellido) donde desarrolles un programa en JAVA que parsee el fichero "**hackers.json**" y que, a partir de él, cree un nuevo fichero xml llamado **hackers_NomApe1.xml** (NomApe1 es tu nombre y primer apellido) que deberá validarse contra el XSD proporcionado (**hackers.xsd**).

El fichero Json original contiene información de numerosos hackers (id, nombre y apellido, idioma, biografía y una nota que determina sus habilidades como hacker. Pero ¡ojo, nos dicen que hay muchos repetidos! La siguiente captura es una muestra del fichero:

```
[
  {
    "nombre y apellido": "Adeel Solangi",
    "idioma": "Sindhi",
    "id": "V59OF92YF627HFY0",
    "biografia": "Donec lobortis eleifend condimentum."
    "nota": 6.1
  },
  {
    "nombre y apellido": "Afzal Ghaffar",
    "idioma": "Sindhi",
    "id": "ENTOCR13RSCLZ6KU",
    "biografia": "Aliquam sollicitudin ante ligula."
    "nota": 1.88
  }
  ...
]
```

El fichero resultante formatea dicha información eliminando duplicados y realizando un resumen al final con la nota media de todos los hackers y el total de hackers distintos por cada idioma. La siguiente captura es una muestra del fichero resultante:



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<hackers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation="hackers.xsd">
    <hacker id="V59OF92YF627HFY0">
        <nombre>Adeel</nombre>
        <apellido>Solangi</apellido>
    </hacker>
    <hacker id="ENTOCR13RSCLZ6KU">
        <nombre>Afzal</nombre>
        <apellido>Ghaffar</apellido>
    </hacker>
    ...
    <resumen>
        <media_notas>5.605837563451777</media_notas>
        <idioma num_hackers="19">Sindhi</idioma>
        <idioma num_hackers="16">Icelandic</idioma>
        <idioma num_hackers="14">Bosnian</idioma>
        <idioma num_hackers="20">Maltese</idioma>
        <idioma num_hackers="20">Uyghur</idioma>
        <idioma num_hackers="41">Hindi</idioma>
        <idioma num_hackers="19">Galician</idioma>
        <idioma num_hackers="20">Sesotho sa Leboa</idioma>
        <idioma num_hackers="16">Setswana</idioma>
        <idioma num_hackers="12">isiZulu</idioma>
    </resumen>
</hackers>
```

Para ello usa una clase **Programa** con el main, **FileManager** que te permita *parsear* el JSON y crear el XML y una clase adicional llamada **Hacker** cuyos atributos sean *id* de tipo String, *nombre* de tipo String y *apellido* de tipo String, nota de tipo double.

Debes volcar sobre un **LOG** (llamado igual que el proyecto) los hitos principales del ejercicio que tú consideres, así como obligatoriamente los valores calculados (media de notas, número de hackers totales y número de hackers por cada idioma)

También debes documentar con **Javadoc** únicamente la clase FileManager.

2. Crea un proyecto Maven llamado AAD_EX02_NomApe1 (NomApe1 es tu nombre y primer apellido) donde desarrolles un programa en JAVA que, a partir de una nota pasada como parámetro, recupere los hackers almacenados en la BDOO proporcionada en el Aula Virtual cuya nota supere la nota parametrizada y que los vuelque sobre un fichero de texto llamado **hackers_superior_X_NomApe1.txt** siendo **X**, el valor de la nota solicitada por el usuario y **NomApe1**, tu nombre y primer apellido).

   Cada registro debe tener el siguiente formato:

   Id | Nombre | Apellido | Nota

   Además de volcar los hackers identificados al fichero:

   A) Se deben borrar de la BD los hackers identificados de manera **transaccional.**
   B) Se debe volcar sobre un **LOG** (llamado igual que el proyecto) los hitos principales del ejercicio que tú consideres y obligatoriamente la siguiente información:
      - Cada hacker identificado que cumpla la condición de nota.
      - Número de hackers que hay en la BD inicialmente.
      - Número de ellos que se borran transaccionalmente.
      - Número de hackers que quedan en la BD tras el borrado.
   C) En caso de que ningún hacker cumpla la condición de nota, el fichero de texto, NO DEBE CREARSE y se debe alertar de tal circunstancia en las trazas mediante un warning donde se indique qué nota no alcanza nadie.
   D) No olvides añadir cabecera en el fichero de texto cuando se cree.

   Para ello usa una clase **Programa** con el main, otra clase **DbManager** que te permita interactuar con la Base de Datos (consultar y borrar) y la clase **Hacker** del ejercicio anterior.

Criterios de evaluación

Cada ejercicio puntúa sobre 10:

- El ejercicio 1 contribuye a la evaluación del RA1
- El ejercicio 2 contribuye a la evaluación del RA4.

Para obtener la máxima puntuación en cada ejercicio, se tendrá en cuenta:

- No hay errores de compilación.
- Cumple con los requisitos funcionales de la práctica:
    - El XML resultante se valida contra el XSD proporcionado.
    - Los valores calculados del resumen son exactos.
    - La búsqueda y borrado en BD se realiza de manera precisa.
    - El volcado de información en fichero de texto se realiza de manera precisa y con el formato solicitado.
    - Las trazas incluyen la información obligatoria.
- Se adelanta y gestiona posibles errores provocados por el usuario controlando número de parámetros, tipos, 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 documenta el código haciendo uso de comentarios **JavaDoc** que describen la funcionalidad de las clases y métodos utilizados para los que no sea inmediato conocer su lógica.
- 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**:

RA1. Desarrolla aplicaciones que gestionan información almacenada en ficheros identificando el campo de aplicación de los mismos y utilizando clases específicas.

c) Se han utilizado clases para recuperar información almacenada en ficheros.
d) Se han utilizado clases para almacenar información en ficheros.
e) Se han utilizado clases para realizar conversiones entre diferentes formatos de ficheros.
f) Se han previsto y gestionado las excepciones.
g) Se han probado y documentado las aplicaciones desarrolladas.

RA4. Desarrolla aplicaciones que gestionan la información almacenada en bases de datos objeto relacionales y orientadas a objetos valorando sus características y utilizando los mecanismos de acceso incorporados.

a) Se han identificado las ventajas e inconvenientes de las bases de datos que almacenan objetos.
b) Se han establecido y cerrado conexiones.
c) Se ha gestionado la persistencia de objetos simples.
e) Se han desarrollado aplicaciones que realizan consultas.
g) Se han gestionado las transacciones.
h) Se han probado y documentado las aplicaciones desarrolladas.

Entrega

Ejercicio 1:

El/la alumno/a entregará en la tarea habilitada del Aula Virtual de la asignatura cuatro 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 en **formato XML** generado.

4. 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.

Ejercicio 2:

El/la alumno/a entregará en la tarea habilitada del Aula Virtual de la asignatura cinco 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 en **formato TXT** generado.
4. La **BD db4o**.
5. 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 comenzar la creación del XML mediante API DOM:

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db =dbf.newDocumentBuilder();
Document documento = db.newDocument();

Element raiz = documento.createElement(<etiqueta elemento raiz>);
raiz.setAttribute("xmlns:xsi", "http://www.w3.org/2001/XMLSchema-instance");
raiz.setAttribute("xsi:noNamespaceSchemaLocation", "hackers.xsd");
```

Puedes apoyarte en la siguiente sección de código para volcar el XML a fichero:

```
TransformerFactory tf = TransformerFactory.newInstance();
Transformer t =tf.newTransformer();
DOMSource ds = new DOMSource(documento);
StreamResult sr = new StreamResult(new FileWriter(<nombre fichero salida>));
t.setOutputProperty(OutputKeys.METHOD, "xml");
t.setOutputProperty(OutputKeys.VERSION, "1.0");
t.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
t.setOutputProperty(OutputKeys.INDENT, "yes");
t.transform(ds, sr);
```

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 escribir el fichero de texto:

```
FileWriter ficheroEscribible = new FileWriter(<variable File>);
PrintWriter pt = new PrintWriter(ficheroEscribible);
pt.println(<texto a escribir>);
```

Puedes apoyarte en el siguiente código para la apertura de la BDOO:

```
ObjectContainer db = Db4oEmbedded.openFile(Db4oEmbedded.newConfiguration(),<variable nombre de la Base De Datos>);
```