DAM - Acceso a datos

Tema 1 - Ficheros y conectores

Roberto Sanz Requena
rsanz@florida-uni.es

Florida
Grup Educatiu

Índice

1. Contexto

- Ficheros y directorios
- Múltiples orígenes de los datos (ficheros, BDD relaciones, BDD OO, etc.).
- Relación con el acceso a datos
- Operaciones sobre ficheros: Crear, Abrir, Cerrar, Leer y Escribir
- Formas de acceso: secuencial y aleatorio

2. La clase `File`

- Listar, crear, renombrar, obtener información o borrar ficheros y directorios.
- No gestiona el contenido de un fichero.
- Métodos:

```
File (String path)
File (String path, String name)
File (File dir, String name)
```

2. La clase `File`

`String[] list()` `String[] list(FileFilter filtro)`	Devuelve un array de tipo String con los nombres de los ficheros y directorios del objeto. Admite un filtro para devolver un subgrupo de nombres.
`File[] listFiles()`	Devuelve un array de tipo File con los ficheros y directorios del objeto.
`String getName()`	Devuelve un String con el nombre del objeto.
`String getPath()`	Devuelve un String con la ruta relativa.
`String getAbsolutePath()`	Devuelve un String con la ruta absoluta.
`boolean exists()`	Devuelve true si el objeto existe.
`boolean canWrite()`	Devuelve true si el objeto se puede escribir.
`boolean canRead()`	Devuelve true si el objeto se puede leer.
`boolean isFile()`	Devuelve true si el objeto es un fichero.
`boolean isDirectory()`	Devuelve true si el objeto es un directorio.
`boolean isAbsolute()`	Devuelve true si una ruta es absoluta.
`Long length()`	Devuelve el tamaño en bytes.
`boolean mkdir()`	Crea un directorio con el nombre del objeto.
`boolean renameTo(File nombrenuevo);`	Renombra el objeto.
`boolean delete()`	Borra el objeto.
`boolean createNewFile()`	Crea físicamente en disco un nuevo fichero asociado al objeto. Devuelve true si lo puede crear o false si ya existe.
`String getParent()`	Devuelve un String con el nombre del directorio padre o null si está en el directorio raíz.

https://docs.oracle.com/javase/7/docs/api/java/io/File.html

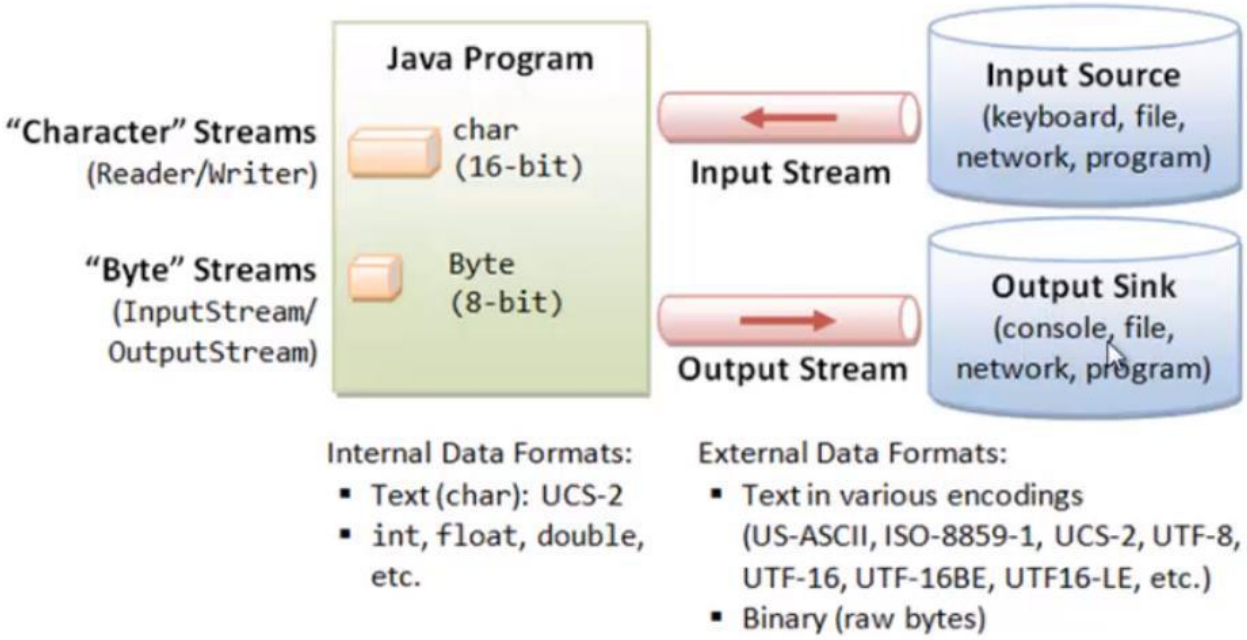
2. La clase `File`

- Ejemplos
    - Información de un fichero
    - Listar contenido directorio
    - Listar contenido directorio filtrando por extensión

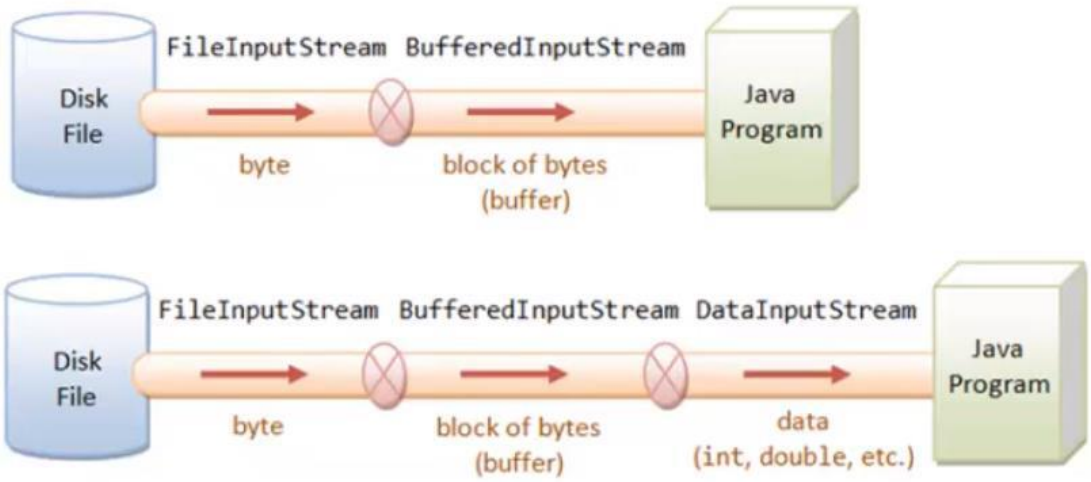
Actividad Entregable 1 - Ficheros

Presentación de la Actividad Entregable 1 (AE01_T1_1_Ficheros)

3. Streams (flujos de datos)



3. Streams (flujos de datos)



3. Streams (flujos de datos)

- Procesos de lectura/escritura

LECTURA	ESCRITURA
Abrir canal de comunicación	Procesar datos (opcional)
Leer datos (mientras existan)	Abrir canal de comunicación
Procesar datos (opcional)	Procesar datos (opcional)
Cerrar canal de comunicación	Escribir datos
Procesar datos (opcional)	Cerrar canal de comunicación

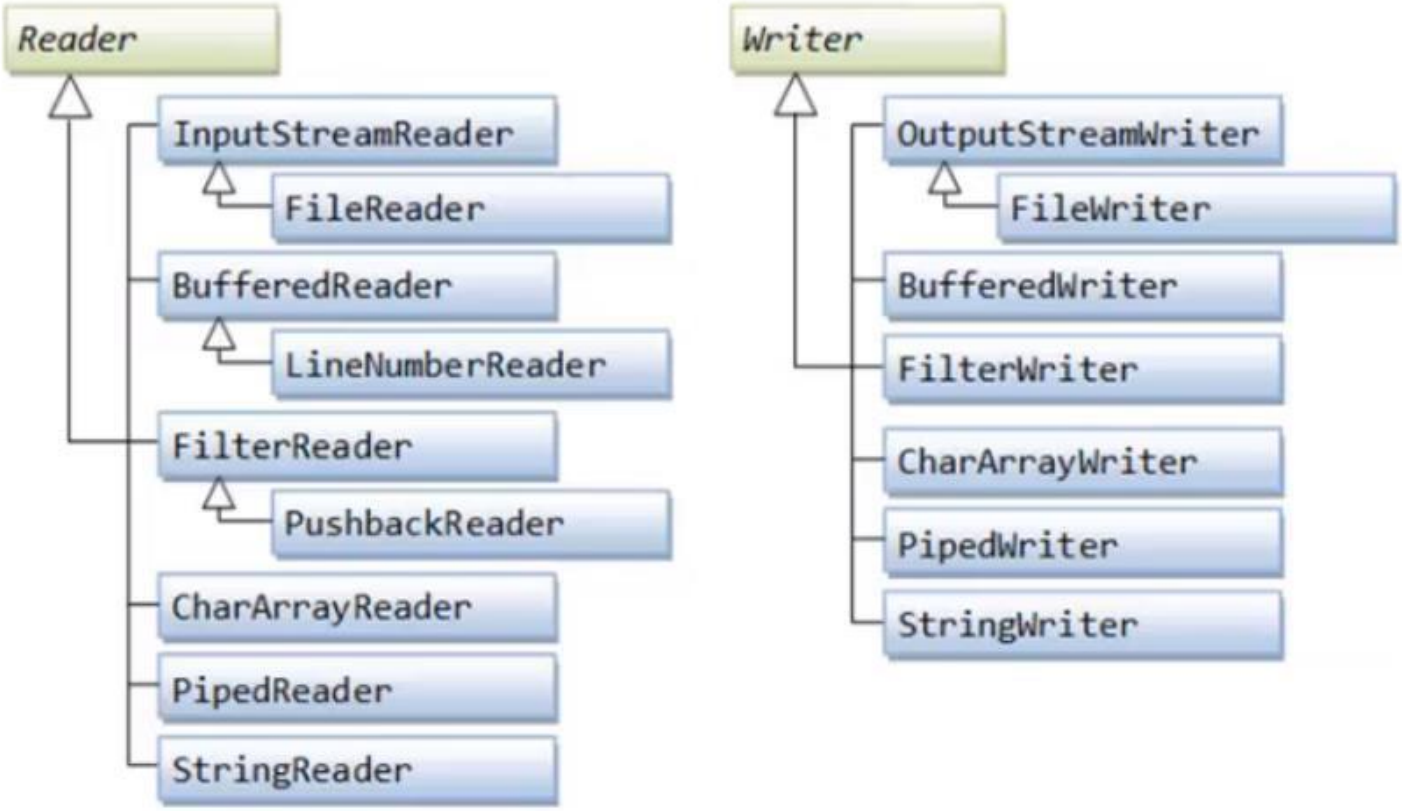
3. Streams (flujos de datos)

- Flujo de caracteres (16 bits):
  - Lectura/escritura de caracteres Unicode.
  - Las clases principales son `Reader` y `Writer`.

- Flujo de bytes (8 bits):
  - Lectura/escritura de datos binarios (bytes).
  - Las clases principales son `InputStream` y `OutputStream`.

- Flujo de tipos de datos primitivos:
  - Lectura/escritura de tipos específicos (boolean, byte, int, short, char, long, float, double, etc.).
  - Las clases principales son `DataInputStream` y `DataOutputStream`.
  - Se utilizan métodos específicos para cada tipo de dato.

3. Streams (flujos de datos)



3. Streams (flujos de datos)

Flujo de caracteres - Lectura

- `FileReader`

`FileReader(File fichero)`	Constructor
`int read()`	Lee un carácter y lo devuelve como entero
`int read(char[] conjunto)`	Lee conjunto.length caracteres y los va almacenando en el array conjunto como enteros

- `BufferedReader`

`BufferedReader(FileReader fr)`	Constructor
`String readLine()`	Lee una línea completa del fichero

https://docs.oracle.com/javase/7/docs/api/java/io/FileReader.html
https://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html

3. Streams (flujos de datos)

Flujo de caracteres – Escritura

- `FileWriter`

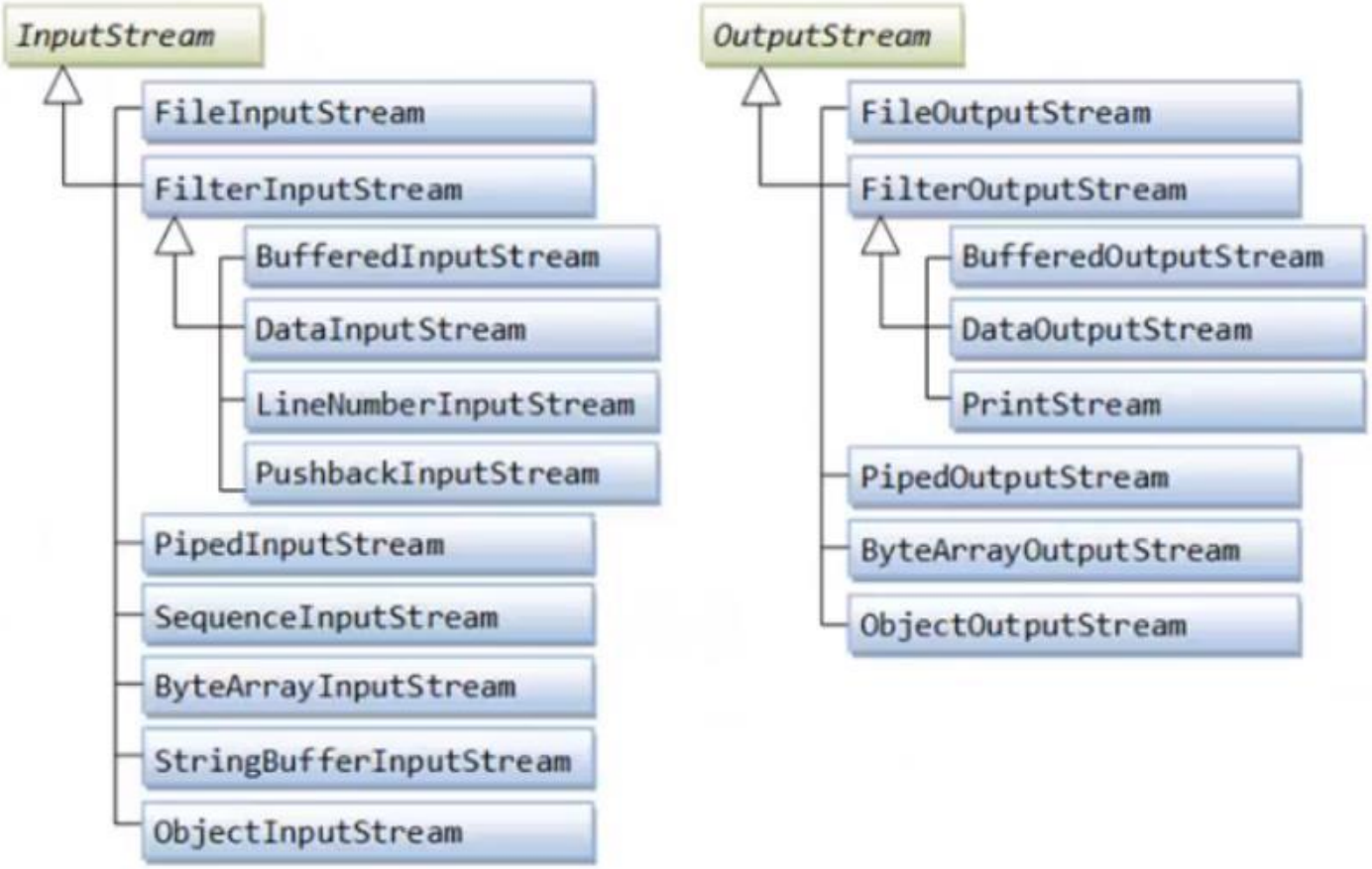
`FileWriter(File fichero)` `FileWriter(File fichero, true)`	Constructor: borrará lo que haya en el fichero Constructor: añadirá al contenido existente
`void write(int c)`	Escribe un carácter
`void write(char[] conjunto)`	Escribe un array de caracteres
`void write(String str)`	Escribe un string
`void append(char c)`	Añade un carácter al final del fichero

- `BufferedWriter`

`BufferedWriter(FileWriter fw)`	Constructor
`write(String str)`	Escribe un string en fichero
`void newLine()`	Añade un salto de línea
`append(String str)`	Añade un string al final del fichero

https://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html
https://docs.oracle.com/javase/7/docs/api/java/io/BufferedWriter.html

3. Streams (flujos de datos)



3. Streams (flujos de datos)

Flujo de bytes: funcionamiento similar

`FileInputStream(File fichero)`	Constructor
`int read()`	Lee un byte y lo devuelve como entero.
`int read(byte[] conjunto)`	Lee conjunto.length bytes y los almacena en array como enteros.
`BufferedInputStream (FileInputStream fis)`	Constructor con buffer.

`FileOutputStream(File fichero)`	Constructor
`void write(int b)`	Escribe un byte.
`BufferedOutputStream (FileOutputStream fos)`	Constructor con buffer.

https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/BufferedOutputStream.html

3. Streams (flujos de datos)

Ejemplos:

- Línea de comandos
  - Leer TXT carácter a carácter y mostrarlo
  - Leer TXT línea a línea y mostrarlo
  - Escribir en fichero
- Operaciones a través de GUI (PDF Complemento - Repaso MVC)

Actividad Entregable 2 - Streams

Presentación de la Actividad Entregable 2 (AE02_T1_2_Streams)

4. Acceso secuencial y acceso aleatorio

- Las anteriores clases son para acceso secuencial.
- Para acceso aleatorio se utiliza la clase RandomAccessFile.
- Modos de acceso:
  - r: modo solo lectura (el fichero debe existir)
  - rw: modo lectura/escritura (si no existe, se crea)

`RandomAccessFile(String fichero, String modo)` `RandomAccessFile(File fichero, String modo)`	Constructores
`long getFilePointer()`	Devuelve la posición del puntero del fichero.
`void seek(long pos)`	Desplaza el puntero a la posición indicada.
`long length()`	Devuelve el tamaño del fichero en bytes.
`int skipBytes(int posiciones)`	Desplaza el puntero un número de posiciones desde la posición actual
`char readChar()`	Lee dato char (16 bits Unicode) Similar para otros tipos primitivos
`void writeChar(int i)`	Escribe dato char (16 bits) Similar para otros tipos primitivos

https://docs.oracle.com/javase/7/docs/api/java/io/RandomAccessFile.html

5. Gestión de ficheros XML

XML (eXtensible Markup Language) es un **metalenguaje** que permite estructurar y jerarquizar la información en base a etiquetas.

Usos:

- Estructurar información (base de datos)
- Ficheros de configuración de programas
- Ejecución de instrucciones en servidores remotos (protocolo SOAP, Simple Object Access Protocol)

Acceso:

- Mediante ***parser***: analizador de etiquetas

5. Gestión de ficheros XML

Tipos de ***parsers***:

Secuenciales (analizadores sintácticos):

- Permiten extraer el contenido a medida que se leen las etiquetas
- Muy rápidos
- Problema: cada vez que se quiere acceder a parte del contenido se debe leer el documento entero
- Parser secuencial más popular: SAX (Simple API for XML)

Jerárquicos:

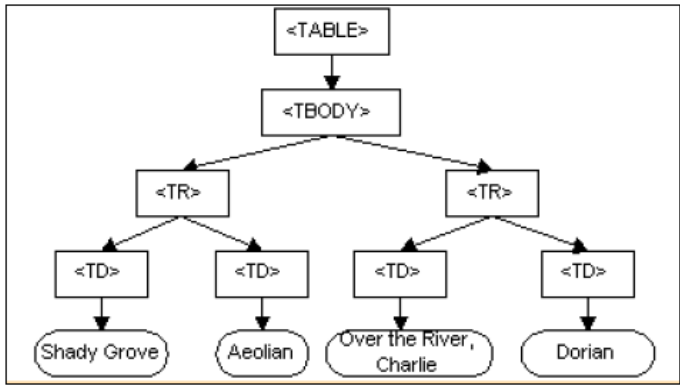
- Guardan todos los datos del XML en memoria dentro de una estructura jerárquica, llamada DOM (Document Object Model) (también utilizado en HTML)
- En Java se implementa mediante interfaces. La principal es `Document` y representa todo el documento XML.
- Ideales para aplicaciones que requieren una consulta continua de los datos.

5. Gestión de ficheros XML

Parser jerárquico DOM:

- La estructura DOM es un árbol donde cada parte del XML está representada en forma de nodo.
- En función de la posición habrá distintos números de nodos.

```
<TABLE>
      <ROWS>
         <TR>
            <TD>Shady Grove</TD>
            <TD>Aeolian</TD>
         </TR>
         <TR>
            <TD>Over the river, Charlie</TD>
            <TD>Dorian</TD>
         </TR>
      </ROWS>
</TABLE>
```



5. Gestión de ficheros XML

Parser jerárquico DOM:

- Para generar la estructura DOM a partir de un XML se utiliza la clase abstracta `DocumentBuilder` (no se pueden hacer objetos
- Para poder crear un objeto se debe especificar `DocumentBuilderFactory` y la clase `Document`
- Pasos para crear la estructura DOM a partir de un archivo XML:

```
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(new File(String ficheroXML));
```

Importante: Al resolver las dependencias fijarse que la librería importada para trabajar con la clase `Document` es: `import org.w3c.dom.Document;`

https://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/DocumentBuilder.html
https://docs.oracle.com/javase/7/docs/api/org/w3c/dom/Document.html

5. Gestión de ficheros XML

Serialización y persistencia

- El DOM es un objeto en memoria de ejecución del programa Java.
- Serializar: transformar el objeto en un flujo de bytes para poder transmitirlo o guardarlo en memoria.
- Persistencia: asegurar que los cambios realizados en el DOM se guarden correctamente.
- Primero se carga el DOM en memoria, se modifica y luego hay que asegurar su persistencia, para lo que se puede utilizar las clases `TransformerFactory` y `Transformer`. También hay librerías (`XMLSerializer`, `XStream`).

https://docs.oracle.com/javase/7/docs/api/javax/xml/transform/TransformerFactory.html
https://docs.oracle.com/javase/7/docs/api/javax/xml/transform/Transformer.html
https://x-stream.github.io/
https://xerces.apache.org/xerces-j/apiDocs/org/apache/xml/serialize/XMLSerializer.html

5. Gestión de ficheros XML

Parser jerárquico DOM – Ejemplos:

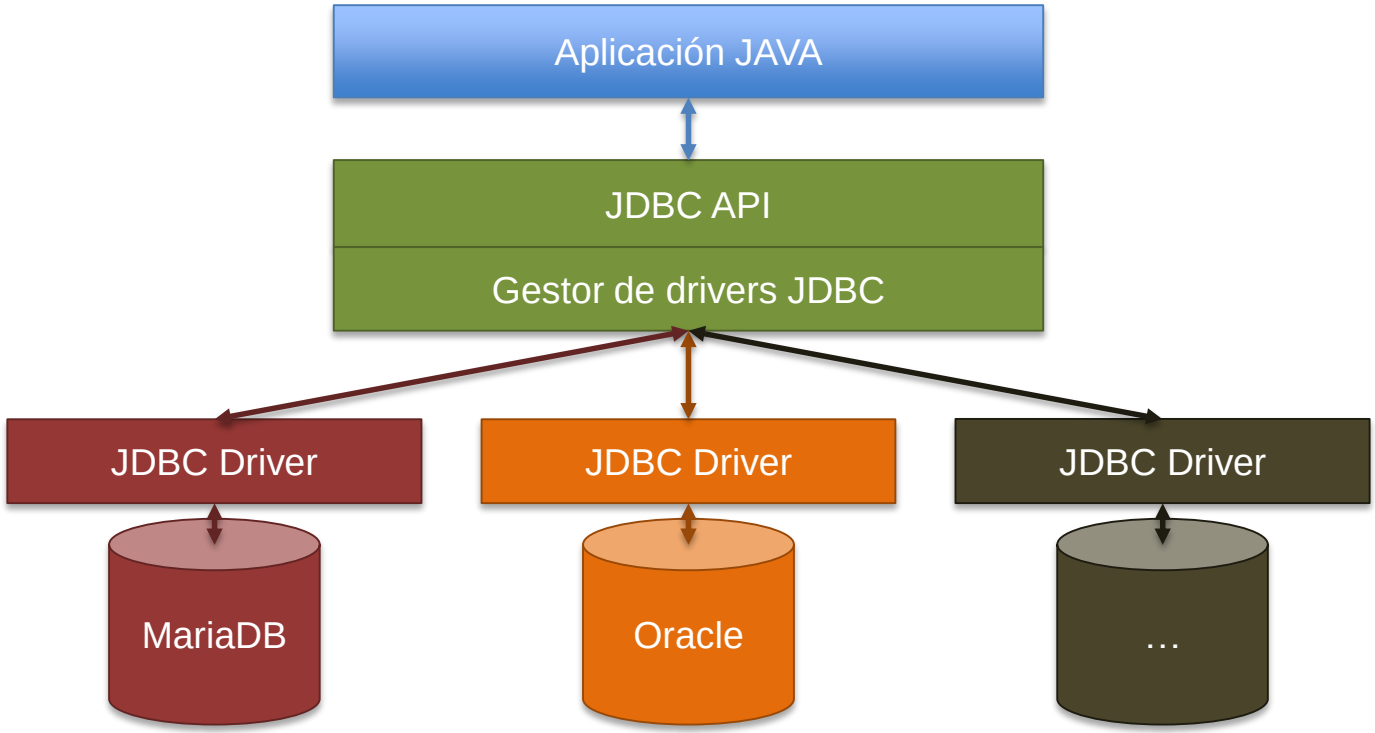
- Leer un documento XML
- Añadir elementos a un documento XML y serializarlo

Actividad Entregable 3 - XML

Presentación de la Actividad Entregable 3 (AE03_T1_3_XML)

6. Conectores: JDBC

JDBC es una interfaz orientada a objetos de **Java** para **SQL** que permite conectarse y ejecutar sentencias SQL contra un sistema gestor de bases de datos (**DBMS**, Database Management System) utilizando para ello un **driver**.



6. Conectores: JDBC

Clases principales

`Driver`	Driver específico del DBMS que permite la conexión a la BDD
`DriverManager`	Gestiona todos los drivers instalados
`DriverPropertyInfo`	Datos informativos sobre el driver
`Connection`	Objeto conexión a la BDD
`DatabaseMetadata`	Datos informativos sobre la BDD
`Statement`	Ejecución de sentencia SQL sin parámetros
`PreparedStatement`	Ejecución de sentencia SQL con parámetros
`CallableStatement`	Ejecución de sentencia SQL con parámetros de entrada y salida
`ResultSet`	Objeto con las referencias a los resultados de una consulta SELECT (los datos no se guardan en el objeto, sólo referencias)
`ResultSetMetadata`	Datos informativos sobre la estructura de los resultados

https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

6. Conectores: JDBC

Algunos métodos útiles

`Class.forName`	Cargar el driver (previamente instalado en el IDE)
`DriverManager.getConnection(String url, String user, String pass)`	Crear la conexión a la BDD que está en la URL correspondiente.
`Connection.createStatement()`	Crear la sentencia
`Statement.executeQuery(String query)`	Ejecutar la sentencia
`ResultSet.next()`	Mueve el puntero al siguiente registro
`ResultSet.previous()`	Mueve el puntero al registro anterior
`ResultSet.first()`	Mueve el puntero al primer registro
`ResultSet.last()`	Mueve el puntero al último registro
`ResultSet.getRow()`	Devuelve como entero el número de registro actual
`ResultSet.getInt(int pos)` `ResultSet.getInt(String columna)`	Devuelve como entero el valor que haya en la columna determinada por la posición pos o por su nombre
`ResultSet.getString(int pos)` `ResultSet.getString(String columna)`	Devuelve como String el valor que haya en la columna determinada por la posición pos o por su nombre

https://docs.oracle.com/javase/tutorial/jdbc/basics/index.htm

6. Conectores: JDBC

Flujo de trabajo habitual:

1. Importar clases necesarias
2. Cargar el driver JDBC correspondiente
3. Identificar el origen de datos
4. Crear una conexión
5. Crear una sentencia
6. Ejecutar la sentencia
7. Gestionar el resultado
8. Cerrar el resultado
9. Cerrar la sentencia
10. Cerrar la conexión

Necesario importar java.sql.* y también el JAR correspondiente al DBMS.

6. Conectores: JDBC

Ejemplos: JDBC y MySQL

- Importar BDD *world-db* a servidor MySQL local
- Ejecutar algunas consultas de ejemplo
- Manipular los datos obtenidos para presentarlos por consola

Actividad Entregable 4 - JDBC

Presentación de la Actividad Entregable 4 (AE04_T1_4_JDBC)