DAM - Accés a dades

Tema 1 - Fitxers i connectors

Roberto Sanz Requena

rsanz@florida-uni.es

Florida
Grup Educatiu

Índex

1. Context

- Fitxers i directoris
- Múltiples orígens de les dades (fitxers, BDD relacionals, BDD OO, etc.).
- Relació amb l'accés a dades
- Operacions sobre fitxers: Crear, Obrir, Tancar, Llegir i Escriure
- Formes d'accés: seqüencial i aleatori

2. La classe `File`

- Llistar, crear, canviar de nom, obtenir informació, borrar fitxers i directoris.
- No gestiona el contingut d'un fitxer.
- Mètodes:

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

2. La classe `File`

`String[] list()` `String[] list(FileFilter filtro)`	Torna un array de tipus String amb els noms dels fitxers i directoris de l'objecte. Admiteix un filtre per a tornar un subgrup de noms.
`File[] listFiles()`	Torna un array de tipus File amb els fitxers i directoris de l'objecte.
`String getName()`	Torna un String amb el nom de l'objecte.
`String getPath()`	Torna un String amb la ruta relativa.
`String getAbsolutePath()`	Torna un String amb la ruta absoluta.
`boolean exists()`	Torna true si l'objecte existeix.
`boolean canWrite()`	Torna true si l'objecte es pot escriure.
`boolean canRead()`	Torna true si l'objecte es pot llegir.
`boolean isFile()`	Torna true si l'objecte és un fitxer.
`boolean isDirectory()`	Torna true si l'objecte és un directori.
`boolean isAbsolute()`	Torna true si una ruta és absoluta.
`Long length()`	Torna la grandària en bytes.
`boolean mkdir()`	Crea un directori amb el nom de l'objecte.
`boolean renameTo(File nou_nom);`	Canvia de nom l'objecte.
`boolean delete()`	Borra l'objecte.
`boolean createNewFile()`	Crea físicament en disc un nou fitxer associat a l'objecte. Torna true si el pot crear i false si ja existeix.
`String getParent()`	Torna un String amb el nom del directori pare o null si està en el directori arrel.

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

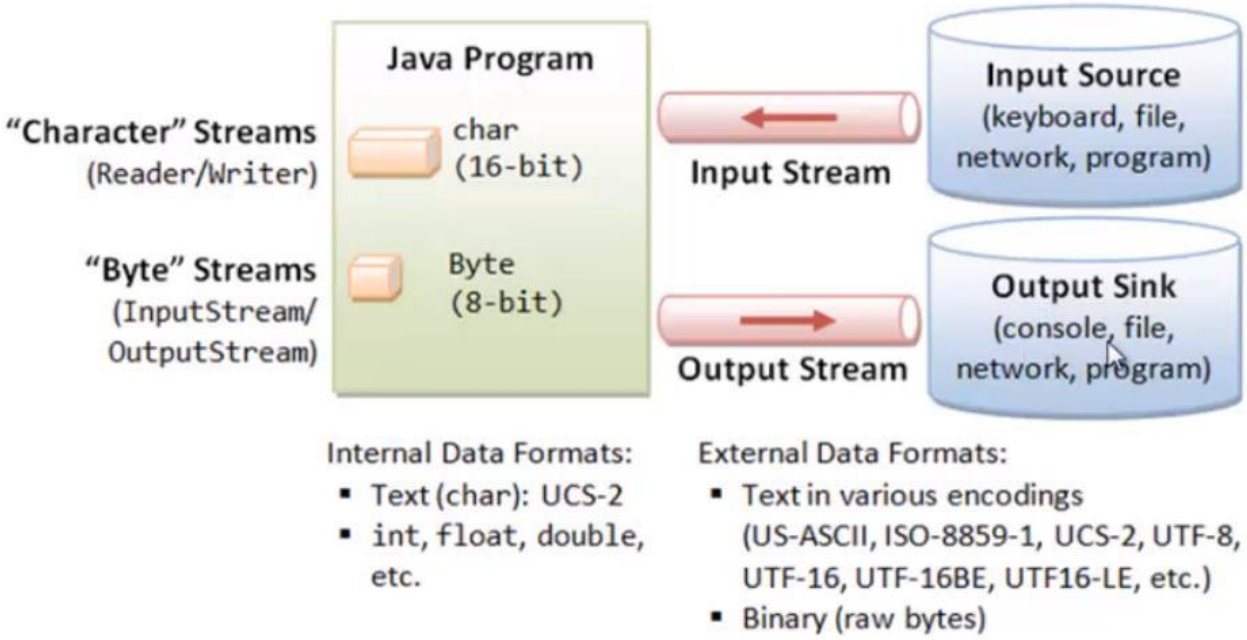
2. La clase `File`

- Exemples
    - Informació d'un fitxer
    - Llistar contingut de directori
    - Llistar contingut de directori filtrant per extensió

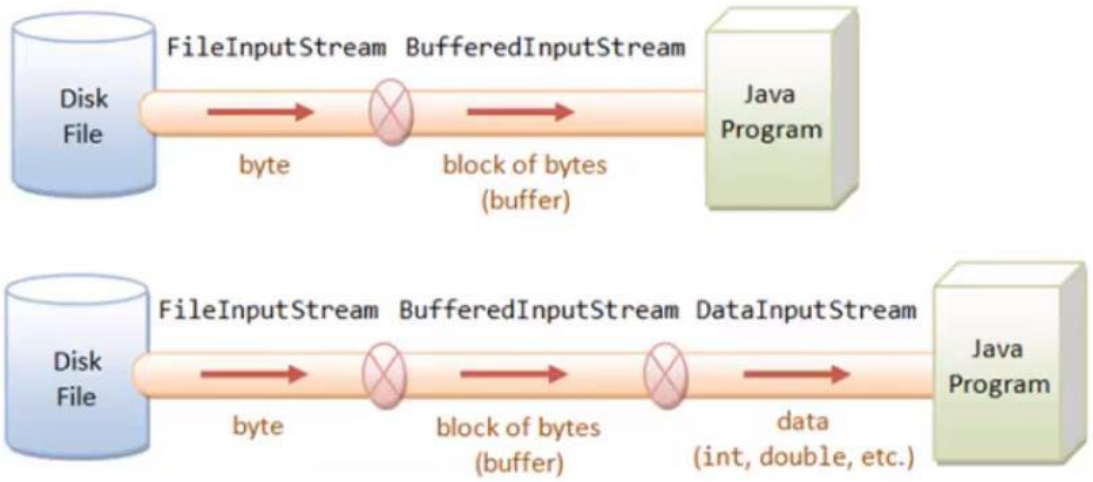
Activitat Entregable 1 - Fitxers

Presentació de l'Activitat Entregable 1 (AE01_T1_1_Fitxers)

3. Streams (fluxes de dades)



3. Streams (fluxes de dades)



3. Streams (fluxes de dades)

- Processos de lectura/escriptura

LECTURA	ESCRIPTURA
Obrir canal de comunicació	Processar dades (opcional)
Llegir dades (mentre existeixen)	Obrir canal de comunicació
Processar dades (opcional)	Processar dades (opcional)
Tancar canal de comunicació	Escriure dades
Processar dades (opcional)	Tancar canal de comunicació

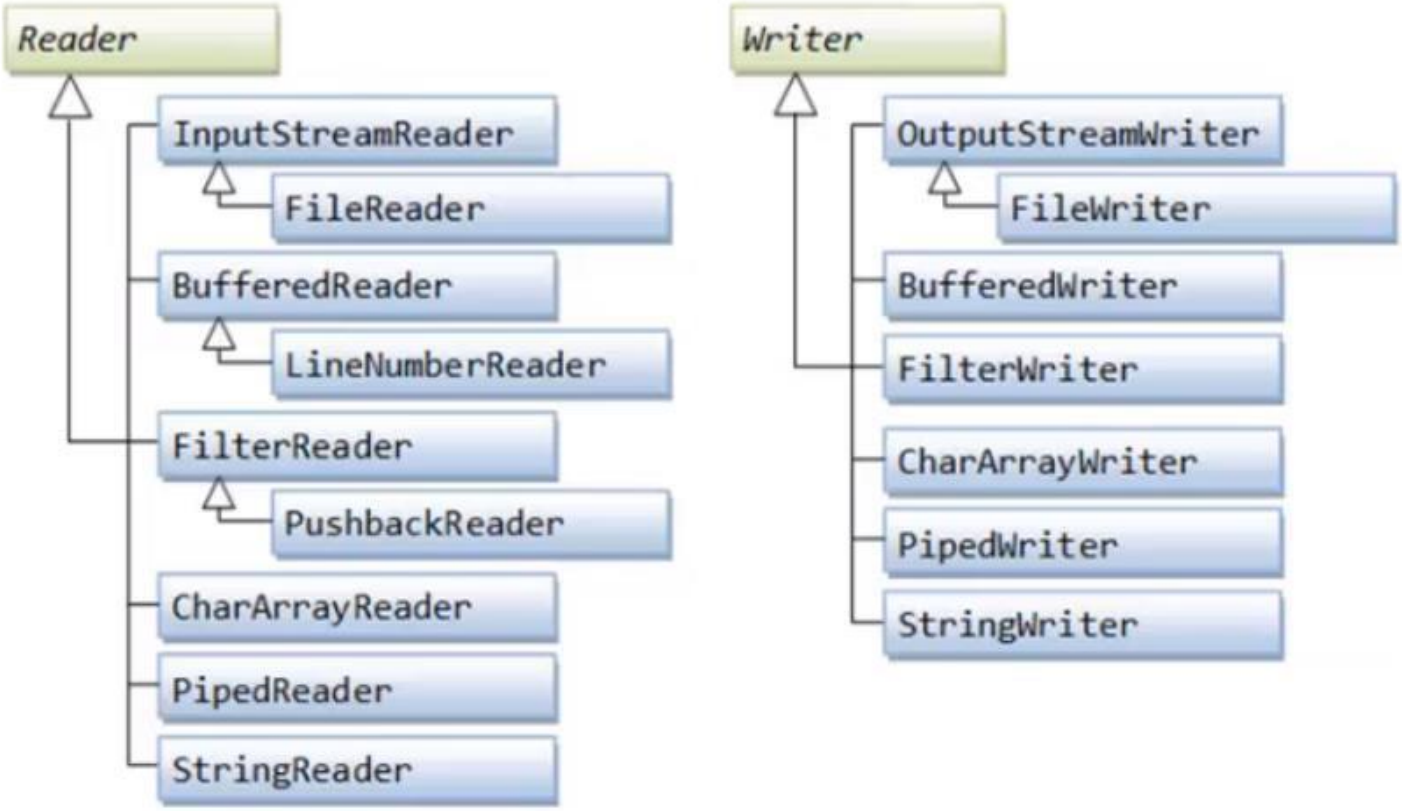
3. Streams (fluxes de dades)

- Flux de caràcters (16 bits):
  - Lectura/escriptura de caràcters Unicode.
  - Les classes principals son `Reader` i `Writer`.

- Flux de bytes (8 bits):
  - Lectura/escriptura de dades binàries (bytes).
  - Les classes principales són `InputStream` i `OutputStream`.

- Flux de tipus de dades primitives:
  - Lectura/escriptura de tipus específics (boolean, byte, int, short, char, long, float, double, etc.).
  - Les classes principals són `DataInputStream` i `DataOutputStream`.
  - S'utilitzen mètodes específics per a cada tipus de dada.

3. Streams (fluxes de dades)



3. Streams (fluxes de dades)

Flux de caràcters - Lectura

- `FileReader`

`FileReader(File fitxer)`	Constructor
`int read()`	Llig un caràcter i el torna com a enter
`int read(char[] conjunt)`	Llig `conjunt.length` caràcters i els va guardant en l'array `conjunt` com a enters

- `BufferedReader`

`BufferedReader(FileReader fr)`	Constructor
`String readLine()`	Llig una línia completa del fitxer

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)

Flux de caràcters – Escriptura

- `FileWriter`

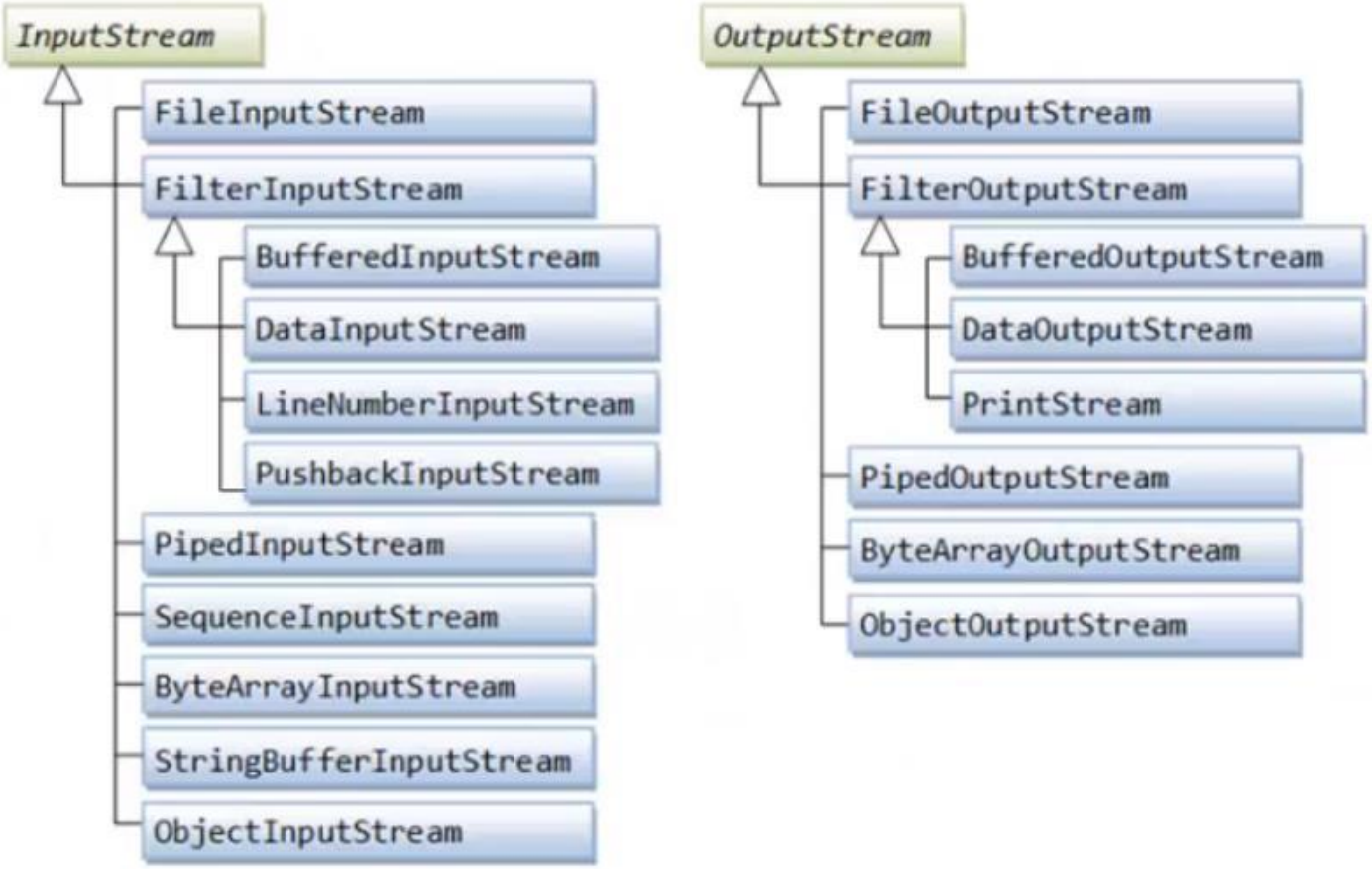
`FileWriter(File fitxer)` `FileWriter(File fitxer, true)`	Constructor: borrarà el que hi haja al fixer Constructor: afegirà al contingut existent
`void write(int c)`	Escriu un caràcter
`void write(char[] conjunt)`	Escriu un array de caràcters
`void write(String str)`	Escriu un string
`void append(char c)`	Afegeix un caràcter al final del fitxer

- `BufferedWriter`

`BufferedWriter(FileWriter fw)`	Constructor
`write(String str)`	Escriu un string en fitxer
`void newLine()`	Afegeix un salt de línia
`append(String str)`	Afegeix un string al final del fitxer

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 (fluxes de dades)



3. Streams (fluxes de dades)

Flux de bytes: funcionament similar

`FileInputStream(File fitxer)`	Constructor
`int read()`	Llig un byte i el torna como a enter.
`int read(byte[] conjunt)`	Llig `conjunt.length` bytes i els guarda en array com enters.
`BufferedInputStream (FileInputStream fis)`	Constructor amb buffer.

`FileOutputStream(File fitxer)`	Constructor
`void write(int b)`	Escriu un byte.
`BufferedOutputStream (FileOutputStream fos)`	Constructor amb 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 (fluxes de dades)

Exemples:

- Línia de comandaments
  - Llegir TXT caràcter a caràcter i mostrar-lo
  - Llegir TXT línia a línia i mostrar-lo
  - Escriure en fitxer
- Operacions a través de GUI (PDF Complement - Repàs MVC)

Activitat Entregable 2 - Streams

Presentació de l'Activitat Entregable 2 (AE02_T1_2_Streams)

4. Accés seqüencial i accés aleatori

- Les anteriors classes són per a accés seqüencial.
- Per a accés aleatori s'utilitza la classe RandomAccessFile.
- Formes d'accés:
  - r: sols lectura (el fitxer ha d'existir)
  - rw: lectura/escriptura (si no existeix, es crea)

`RandomAccessFile(String fitxer, String forma)` `RandomAccessFile(File fitxer, String forma)`	Constructors
`long getFilePointer()`	Torna la posició del punter del fitxer.
`void seek(long pos)`	Desplaça el punter a la posició indicada.
`long length()`	Torna la grandària del fitxer en bytes.
`int skipBytes(int posicions)`	Desplaça el punter un nombre de posicions des de la posició actual
`char readChar()`	Llig una dada de tipus char (16 bits Unicode) Similar per a altres tipus primitius
`void writeChar(int i)`	Escriu una dada char (16 bits) Similar per a altres tipus primitius

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

5. Gestió de fitxers XML

XML (eXtensible Markup Language) és un **metallenguatge** que permet estructurar i jerarquitzar la informació sobre la base d'etiquetes.

Usos:

- Estructurar informació (bases de dades)
- Fitxers de configuració de programes
- Execució d'instruccions en servidors remots (protocol SOAP, Simple Object Access Protocol)

Accés:

- Mitjançant ***parser***: analitzador d'etiquetes

5. Gestió de fitxers XML

Tipus de ***parsers***:

Seqüencials (analitzadors sintàctics):

- Permeten extraure el contingut a mesura que es lligen les etiquetes.
- Molt ràpids.
- Problema: cada vegada que es vol accedir a part del contingut s'ha de llegir el document sencer.
- Parser seqüencial més popular: SAX (Simple API for XML).

Jeràrquics:

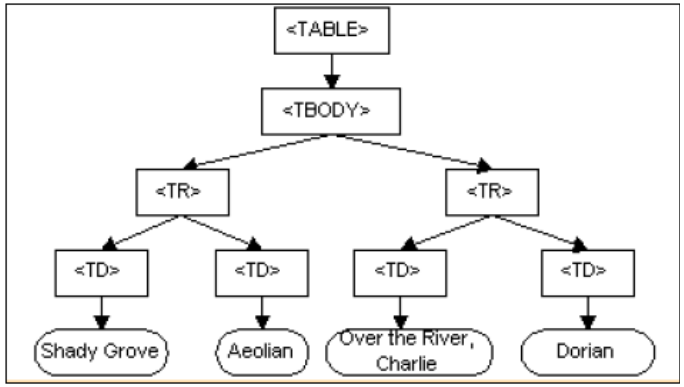
- Guarden totes les dades de l'XML en memòria dins d'una estructura jeràrquica, anomenada DOM (Document Object Model) (també utilitzat en HTML).
- En Java s'implementa mitjançant interfícies. La principal és `Document` i representa tot el document XML.
- Ideals per a aplicacions que requereixen una consulta contínua de les dades.

5. Gestió de fitxers XML

Parser jeràrquic DOM:

- L'estructura DOM és un arbre on cada part de l'XML està representada en forma de node.
- En funció de la posició hi haurà distints nombres de nodes.

```
<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. Gestió de fitxers XML

Parser jeràrquic DOM:

- Per a generar l'estructura DOM a partir d'un XML s'utilitza la classe abstracta `DocumentBuilder` (no es poden fer objectes)
- Per a poder crear un objecte s'ha d'especificar `DocumentBuilderFactory` i la classe `Document`
- Passos per a crear l'estructura DOM a partir d'un fitxer XML:

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

Important: per resoldre les dependències fixar-se que la llibreria importada per treballar amb la classe `Document` és: `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. Gestió de fitxers XML

Serialització i persistència

- El DOM és un objecte en memòria d'execució del programa Java.
- **Serialitzar**: transformar l'objecte en un flux de bytes per poder transmetre'l o guardar-lo en memòria.
- **Persistència**: assegurar que els canvis realitzats en el DOM es guarden correctament.
- Primer es carrega el DOM en memòria, es modifica i després cal assegurar la seua persistència, per a la qual cosa es poden utilitzar les classes `TransformerFactory` i `Transformer`. També hi ha llibreries (`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. Gestió de fitxers XML

Parser jeràrquic DOM – Exemples:

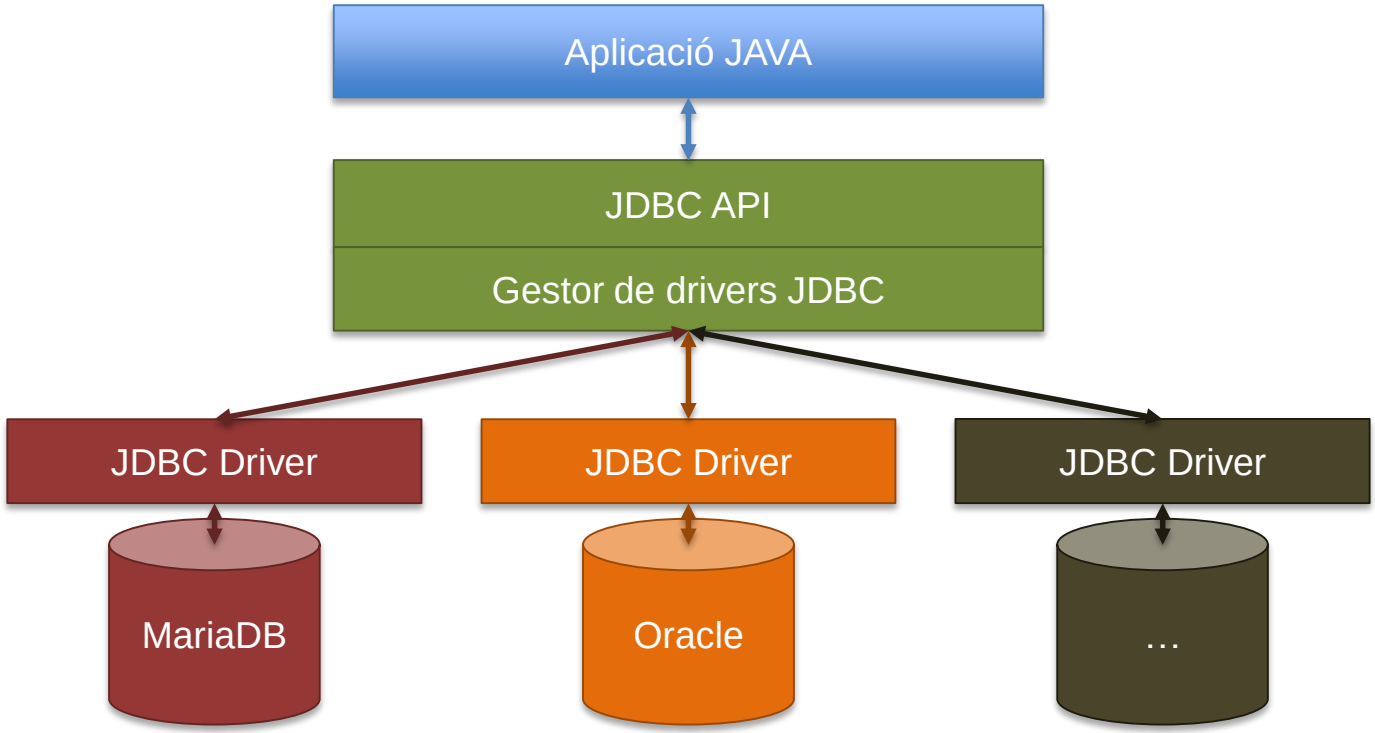
- Llegir un document XML
- Afegir elements a un document XML i serialitzar-lo

Activitat Entregable 3 - XML

Presentació de l'Activitat Entregable 3 (AE03_T1_3_XML)

6. Connectors: JDBC

JDBC és una interfície orientada a objectes de **Java** per a **SQL** que permet connectar-se i executar sentències SQL contra un sistema gestor de bases de dades (**DBMS**, Database Management System) mitjançant un **driver**.



6. Connectores: JDBC

Classes principals

`Driver`	Driver específic del DBMS que permet la connexió a la BDD
`DriverManager`	Gestiona tots els drivers instal·lats
`DriverPropertyInfo`	Dades informatives sobre el driver
`Connection`	Objecte connexió a la BDD
`DatabaseMetadata`	Dades informatives sobre la BDD
`Statement`	Execució de sentència SQL sense paràmetres
`PreparedStatement`	Execució de sentència SQL amb paràmetres
`CallableStatement`	Execució de sentència SQL amb paràmetres d'entrada i eixida
`ResultSet`	Objecte amb les referències als resultats d'una consulta SELECT (les dades no es guarden en l'objecte, sols referències)
`ResultSetMetadata`	Dades informatives sobre l'estructura dels resultats

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

6. Connectors: JDBC

Alguns mètodes útils

`Class.forName`	Carregar el driver (prèviament instal·lat en l'IDE)
`DriverManager.getConnection(String url, String user, String pass)`	Crear la connexió a la BDD que està a la URL corresponent.
`Connection.createStatement()`	Crear la sentència
`Statement.executeQuery(String query)`	Executar la sentència
`ResultSet.next()`	Mou el punter al següent registre
`ResultSet.previous()`	Mou el punter al registre anterior
`ResultSet.first()`	Mou el punter al primer registre
`ResultSet.last()`	Mou el punter a l'últim registre
`ResultSet.getRow()`	Torna com a enter el nombre de registre actual
`ResultSet.getInt(int pos)` `ResultSet.getInt(String columna)`	Torna com a enter el valor que hi haja a la columna determinada per la posició pos o el seu nom
`ResultSet.getString(int pos)` `ResultSet.getString(String columna)`	Torna como String el valor que hi haja a la columna determinada por la posició pos o pel seu nom

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

6. Connectors: JDBC

Flux de treball habitual:

1. Importar classes necessàries
2. Cargar el driver JDBC corresponent
3. Identificar l'origen de dades
4. Crear una connexió
5. Crear una sentència
6. Executar la sentència
7. Gestionar el resultat
8. Tancar el resultat
9. Tancar la sentència
10. Tancar la connexió

Necessari importar java.sql.* i també el JAR corresponent al DBMS.

6. Connectors: JDBC

Exemples: JDBC i MySQL

- Importar BDD *world-db* a servidor MySQL local
- Executar algunes consultes d'exemple
- Manipular les dades obtingudes per a presentar-les per consola

Activitat Entregable 4 - JDBC

Presentació de l'Activitat Entregable 4 (AE04_T1_4_JDBC)