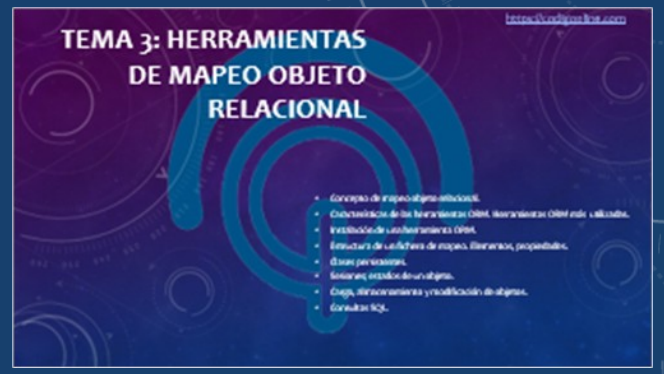
M06

ACCESO A DATOS

ÍNDICE GENERAL



TEMA 1: MANEJO DE FICHEROS (JAVA)

TRABAJO CON FICHEROS XML: ANALIZADORES SINTÁCTICOS (PARSER) Y VINCULACIÓN (BINDING)

EXCEPCIONES: DETECCIÓN Y TRATAMIENTO

FICHEROS

- Un fichero es un conjunto de bits almacenado en un dispotivio.
- Tiene una gran característica, los datos almacenados no se eliminan al apagar el dispotivo, por lo que tienen un almacenamiento persistente a diferencia de la RAM.
- Los ficheros se tienen 3 secciones muy importantes
    - RUTA → Dónde se encuentra ubicado este fichero
    - NOMBRE → Cómo se llama el fichero
    - EXTENSIÓN → Qué tipo de fichero es
- Un fichero debe tener un nombre único en su ruta, pero pueden existir dos ficheros con el mismo nombre en rutas diferentes

```
/Users/aortega/Desktop/nayade/10_0.jpeg
```

FICHEROS

RUTA

enado en un dispotivio.

almacenados no se eliminan al apagar el dispotivo, por lo que
nte a diferencia de la RAM.

- Los ficheros se tienen 3 secciones muy importantes
  - RUTA → Dónde se encuentra ubicado este fichero
  - NOMBRE → Cómo se llama el fichero
  - EXTENSIÓN → Qué tipo de fichero es
- Un fichero debe tener un nombre único en su ruta, pero pueden existir dos ficheros con el mismo nombre en rutas diferentes

FICHEROS

RUTA

enado en un dispotivio.

almacenados no se eliminan al apagar el dispotivo, por lo que
nte a diferencia de la RAM.

- Los ficheros se tienen 3 sec
  - RUTA → Dónde se
  - NOMBRE → Cómo
  - EXTENSIÓN → Qué
- Un fichero debe tener un nombre único en su ruta, pero pueden existir dos ficheros con el mismo nombre en rutas diferentes

NOMBRE

/Users/aortega/Desktop/nayade/10_0.jpeg

FICHEROS

RUTA

EXTENSIÓN

NOMBRE

- Los ficheros se tienen 3 sec
  - RUTA → Dónde se
  - NOMBRE → Cómo
  - EXTENSIÓN → Qué
- Un fichero debe tener un nombre único en su ruta, pero pueden existir dos ficheros con el mismo nombre en rutas diferentes

enado en un dispotivi

almacenados no se eli

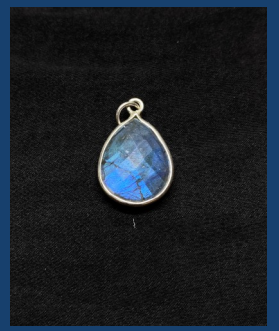
nte a diferencia de la RA

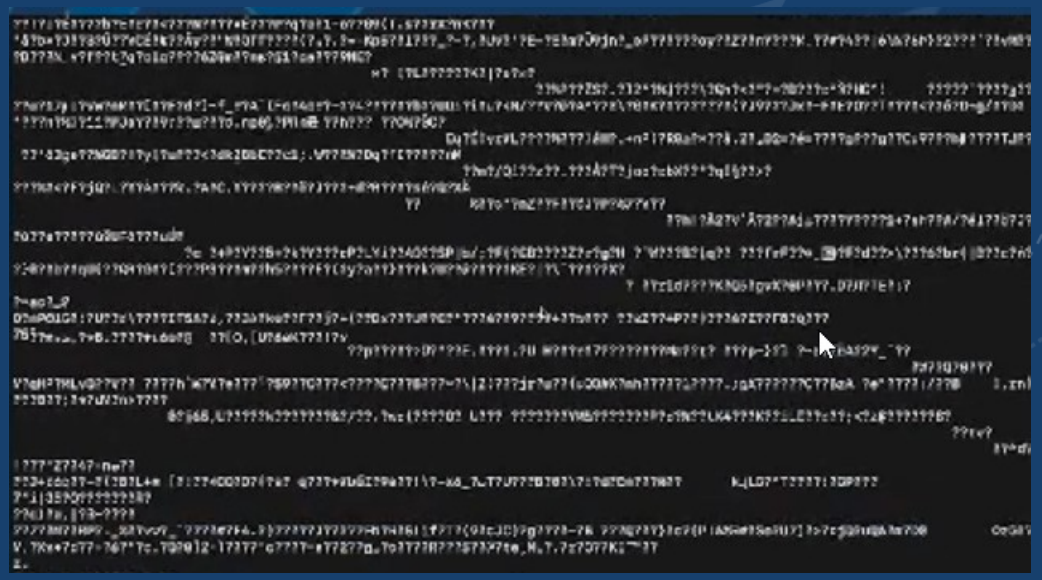
EXTENSIONES

- Las extensiones determinan que tipo de fichero es y por tanto como debe tratarlo el sistema operativo.
    - ¿Es un fichero de música?
    - ¿Es un ejecutable?
    - ¿Es un fichero web?
    - ¿Es una imagen?
- Cómo podéis observar para cada tipo de fichero debe de existir una extensión y cada software puede tener las suyas propias. Por ejemplo PowerPoint con la extensión pptx.
-
- Aunque la mayoría de las veces las extensiones tienen 3 letras suelen ir de las 2 a las 4.
    - .sh
    - .html
    - .pptx
    - .exe
    - .dmg

¿CÓMO ES UN FICHERO POR DENTRO?

- No existe una forma predeterminada de como es la estructura de un fichero ya que cada desarrollador puede diseñarlo como el crea conveniente, pero lo más importante es que está formado por bloques de bytes que guardan la información deseada.
- Estos bloques se denominan registros y gracias a la extensión y el software correspondiente el SO puede interpretar el fichero y mostrarlo de forma correcta.
- ¿Has abierto alguna vez una imagen en formato texto?
-
-





FICHEROS EN JAVA

- Para gestionar todas las operaciones con los ficheros, se utilizan dos librerías incluídas en el jdk de Java.
    - Java.io → Java Input Output
    - Java.nio → Java Non-blocking Input Output
        - Esta segunda librería supone una mejora en la forma en la que se realizan las operaciones. Ambas pueden ser utilizadas, pero java.nio corrigió muchas de las deficiencias de java.io

OPERACIONES SOBRE FICHEROS

- Las operaciones básicas que un fichero admite son:
  - **Creación del fichero**: El fichero se almacena en el disco duro, este debe tener un nombre y extensión única.
  - **Apertura del fichero**: para poder realizar operaciones sobre un fichero debe estar abierto dentro de nuestro programa y así de esta manera tener un apuntador hacia sus dirección de memoria.
    - **Lectura:** Consiste en leer datos del fichero y así poder recuperarlos cuando sea necesario. Se debe disponer del permiso de lectura sobre el fichero.
    - **Escritura de datos**: Consiste en escribir datos en el fichero y que así la información sea persistida. Se deberán tener permisos de escritura sobre el fichero.
      - Altas
      - Modificaciones
      - Bajas → Dependiendo del tipo de acceso se realizará de una forma u de otra:
        - Secuenciales: Se crea un nuevo archivo sin los datos que se desean
        - Aleatorios: Se desactiva el registro para posteriormente cuando se haga una modificación sobreescribirlo.
          - Valor 1: el registro está activo y no se puede sobreescribir
          - Valor 0: el registro NO está activo y se puede sobreescribir
  - **Cierre del fichero**: El fichero se debe cerrar para que quede disponible para otros programas y así no producir errores en su estructura de datos. Si se queda abierto de forma indefinida puede que se den situaciones de bloqueo en el acceso a la escritura del mismo.

FORMAS DE ACCESO A UN FICHERO

- Acceso secuencial: El acceso se produce desde el primer registro y se va avanzando registro a registro para leer la información. Si deseamos acceder al registro 39 debemos leer los 38 anteriores.
    - Cómo podemos observar son los menos óptimos ya que es obligatorio recorrer de forma constante el fichero desde el inicio.
    - Un ejemple puede ser un VHS, (película antigua) donde para ir a un fragmento de la misma debíamos avanzar o rebobinar sobre la cinta electromagnética para leer su información
    -
- Acceso aleatorio: Se puede acceder directamente a un registro sin haber recorrido los anteriores. Casi todos los sistemas actuales utilizan este formato ya que es mucho más rápido que el secuencial.
    - Un ejemplo es un DVD donde podemos avanzar y retroceder a nuestro antojo desde cualquier punto.
        - Pasaría lo mismo con un vídeo online, es un fichero al que podremos acceder a cualquiera de sus segundos en cualquier momento sin tener que ver todo el vídeo entero.

FORMAS DE ACCESO A UN FICHERO

e produce desde el primer registro y se va avanzando registro a registro para amos acceder al registro 39 debemos leer los 38 anteriores.

ar son los menos óptimos ya que es obligatorio recorrer de forma constante el fichero

n VHS ntigua) donde para ir a un fragmento de la misma debíamos avanzar cinta nética para leer su información

Acceso secuencial

mente a un registro sin haber recorrido los anteriores. Casi ste formato ya que es mucho más rápido que el secuencial.

emos avanzar y retroceder a nuestro antojo desde cualquier punto.

vídeo online, es un fichero al que podremos acceder a cualquiera de sus segundos en n tener que ver todo el vídeo entero.

FORMAS DE ACCESO A UN FICHERO

- Acceso secuencial: El acceso se produce desde ndo registro a registro para leer la información. Si deseamos acceder al r nteriores.
  - Cómo podemos observar son los menos ó de forma constante el fichero desde el inicio.
  - Un ejemple puede ser un VHS, (película an le la misma deb ar o rebobinar sobre la cinta electromagnética para leer su información
  -
- Acceso aleatorio: Se puede acceder directame do los a todos los sistemas actuales utilizan este for que el se .
  - Un ejemplo es un DVD donde podemos av esde cual o.
    - Pasaría lo mismo con un vídeo onlin cualquie e su gundos en cualquier momento sin tener que ver todo el vídeo entero.

Acceso aleatorio

JAVA.IO

- Es la librería inicial de entrada y salida.
- Se encarga de gestionar las operaciones de entrada y salida de nuestro programa
    - Es muy usada, ya que las operaciones System.out y System.err la utilizan para poder mostrar los mensajes
- Trabaja con streams, un flujo de datos (en formato de bytes).
- La clase principal se llama File, ya que es la que nos permite abrir o crear ficheros donde luego leeremos/escribiremos

CLASE FILE

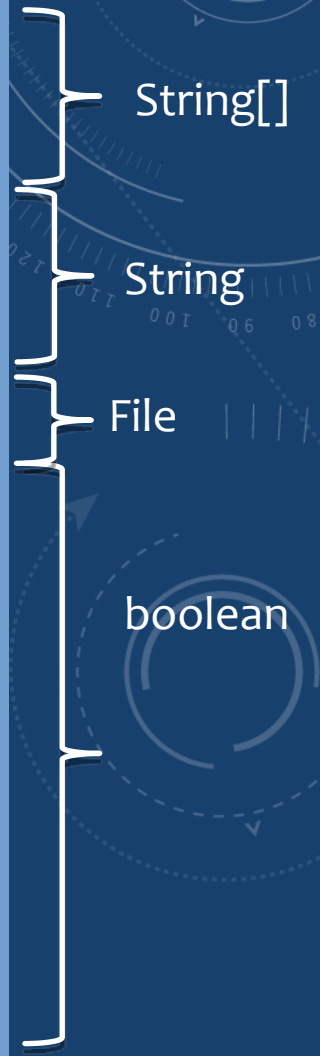
- La clase File, proporciona las herramientas necesarias para poder trabajar con ficheros y conocer sus propiedades.
- Tiene 4 constructores diferentes, dependiendo de como le pasemos la información
-
- Dependiendo de nuestras necesidades podemos utilizar uno u otro.
    - Aunque la clase se llama File sirve para gestionar directorios y ficheros
    - Un detalle muy importante, el constructor no crea el fichero en el sistema, solo en RAM por lo que debemos asegurarnos de crear el fichero o directorio de forma programática (siguiente diapositiva)

File file1 = new File(“path + nombre_fichero”);
File file2 = new File (“path”,”nombre_fichero”);
File file3 = new File(new File(”path”),”nombre_fichero”);
File file4 = new File(Uri uri);

```
File file1 = new File( pathname: "ficheros/file1.txt");
File file2 = new File( parent: "ficheros", child: "file2.txt");
File path  = new File( pathname: "ficheros");
File file3 = new File(path, child: "file3.txt");
```

FILE → FUNCIONES PRINCIPALES

Función	Explicación	Ejemplo
list()	Devuelve un listado con todos los ficheros y directorios del directorio utilizado	File ficheros = new File("ficheros"); ficheros.list();
listFiles()	Devuelve un listado con los ficheros del directorio	File ficheros = new File("ficheros"); ficheros.listFiles();
getPath()	Obtiene la ruta del fichero, el inicio es el punto de ejecución del programa	File ficheros = new File("ficheros"); System.out.println(ficheros.getPath());
getAbsolutePath()	Obtiene la ruta completa desde la raíz del sistema al fichero	File ficheros = new File("ficheros"); System.out.println(ficheros. getAbsolutePath());
getParent()	Obtiene el nombre del directorio padre	File ficheros = new File("ficheros"); System.out.println(ficheros. getParent());
canRead()	Devuelve true o false, dependiendo si se puede o no leer	File ficheros = new File("ficheros"); System.out.println(ficheros. canRead());
canWrite()	Devuelve true o false, dependiendo si se puede o no escribir	File ficheros = new File("ficheros"); System.out.println(ficheros. canWrite());
mkdir()	Crea un nuevo directorio	File ficheros = new File("ficheros"); System.out.println(ficheros. mkdir());
createNewFile()	Crea un nuevo fichero	File ficheros = new File("ficheros"); System.out.println(ficheros. createNewFile());
delete()	Elimina un fichero o directorio, si es un directorio debe estar vacío.	File ficheros = new File("ficheros"); System.out.println(ficheros. delete());
renameTo()	Renombra un fichero o directorio	File ficheros = new File("ficheros"); System.out.println(ficheros.renameTo(new File("eliminados")));



FILE → FUNCIONES PRINCIPALES

Función	Explicación	Ejemplo	
list()	Devuelve un listado con todos los ficheros y directorios del directorio utilizado	File ficheros = new File("ficheros"); ficheros.list();	String[]
listFiles()	Devuelve un listado con los ficheros del directorio	File ficheros = new File("ficheros"); ficheros.listFiles();	
getPath()	Obtiene la ruta del fichero, el inicio es el punto de ejecución del programa	File ficheros = new File("ficheros"); System.out.println(ficheros.getPath());	String
getAbsolutePath()	Obtiene la ruta completa desde la raíz del sistema al fichero	File ficheros = new File("ficheros"); System.out.println(ficheros. getAbsolutePath());	
getParent()	Obtiene el nombre del directorio padre	File ficheros = new File("ficheros"); System.out.println(ficheros. getParent());	File
mkdir()	Crea un nuevo directorio	File ficheros = new File("ficheros"); System.out.println(ficheros. mkdir());	
createNewFile()	Crea un nuevo fichero	File ficheros = new File("ficheros"); System.out.println(ficheros. createNewFile());	
delete()	Elimina un fichero o directorio, si es un directorio debe estar vacío.	File ficheros = new File("ficheros"); System.out.println(ficheros. delete());	
renameTo()	Renombra un fichero o directorio	File ficheros = new File("ficheros"); System.out.println(ficheros.renameTo(new File("eliminados")));	boolean

canRead y canWrite nos devolverán false hasta que creemos el fichero/directorio

FILE → FUNCIONES PRINCIPALES

Función	Explicación	Ejemplo	
list()	Devuelve un listado con todos los ficheros y directorios del directorio utilizado	File ficheros = new File("ficheros"); ficheros.list();	String[]
listFiles()	Devuelve un listado con los ficheros del directorio	File ficheros = new File("ficheros"); ficheros.listFiles();	String[]
getPath()	Obtiene la ruta del fichero, el inicio es el punto de ejecución del programa	File ficheros = new File("ficheros"); System.out.println(ficheros.getPath());	String
getAbsolutePath()	Obtiene la ruta completa desde la raíz del sistema al fichero	File ficheros = new File("ficheros"); System.out.println(ficheros. getAbsolutePath());	String
getParent()	Obtiene el nombre del directorio padre	File ficheros = new File("ficheros"); System.out.println(ficheros. getParent());	File

canRead y canWrite nos devolverán false hasta que creemos el fichero/directorio

Con mkdir/createFile creamos el fichero/directorio de forma física en el disco, hasta entonces sólo existen de manera virtual, si se cierra el programa se eliminan.

delete()	Elimina un fichero o directorio, si es un directorio debe estar vacío.	File ficheros = new File("ficheros"); System.out.println(ficheros. delete());	boolean
renameTo()	Renombra un fichero o directorio	File ficheros = new File("ficheros"); System.out.println(ficheros.renameTo(new File("eliminados")));	boolean

FILE → FUNCIONES PRINCIPALES

Función	Explicación	Ejemplo
list()	Devuelve un listado con todos los ficheros y directorios del directorio utilizado	File ficheros = p ("ficheros"); ficheros.list()
listFiles()	Devuelve un listado con los ficheros del directorio	File fi ("ficheros");
getPath()	Obtiene la ruta del fich l punto de del programa	
getAbsolutePath()	Obtiene la ruta completa d	
getParent()	Obti	
	Con mkdir/createFil	en el d sta entonces sólo el p na se eliminan.
delete()	Elimina un fichero o dir si es un directorio debe estar vacío.	File ficheros = new File("ficheros"); System.out.println(ficheros. delete());
renameTo()	Renombra un fichero o directorio	File ficheros = new File("ficheros"); System.out.println(ficheros.renameTo(new File("eliminados")));

La función createNewFile() puede lanzar la excepción IOException por lo que deberemos tratar la excepción en un bloque try/catch o lanzarla a la función superior

String[]

String

File

boolean

OPERACIONES BÁSICAS DE FICHEROS

- Existen 4 operaciones sobre los ficheros esenciales para la gestión de los mismos:
  - Creación
  - Eliminación
  - Copia
  - Movimiento
  -
- Por desgracia no todas ellas se pueden realizar con el API básico de java.io, en esta están disponibles las operaciones de
  - Creación
  - Eliminación
  - Renombre
  -

```
File file = new File(“teoria/operaciones_básicas”);
file.createNewFile();
```

```
File file = new File("teoria/operaciones_basicas");
file.delete();
```

Existe otra operación que no está incluida en este bloque pero me gusta hacer hincapié en ella, el renombramiento de ficheros

```
File file = new File("teoria/operaciones_basicas");
File rename = new File("teoria/operaciones_basicas1");
file.renameTo(rename);
```

FILEUTILS

- Para poder realizar más operaciones sobre los ficheros, disponemos de la clase FileUtils, de la librería de Apache Commons.
    - No viene con el jdk por defecto por lo que si deseamos utilizarla debemos añadirla de forma manual a nuestro proyecto.
- Está nos ofrece un conjunto de mecanismos para poder controlar los ficheros de forma muy sencilla, por ejemplo las funciones:
    - Copia
    - Movimiento
-

FILEUTILS

- Para poder realizar más operaciones sobre los ficheros, disponemos de la clase FileUtils, de la librería de Apache Commons.
    - No viene con el jdk por defecto por lo que si deseamos utilizarla debemos añadirla de forma manual a nuestro proyecto.
- Está nos ofrece un conjunto de mecanismos para poder controlar los ficheros de forma muy sencilla, por ejemplo las funciones:
    - Copia
    - Movimiento
-

```
try {
   FileUtils.copyFileToDirectory(file, destino);
   System.out.println("Fichero copiado");

} catch (IOException ex) {
   System.err.println("Error al copiar el
archivo");
   ex.printStackTrace();
}
```

FILEUTILS

- Para poder realizar más operaciones sobre los ficheros, disponemos de la clase FileUtils, de la librería de Apache Commons.
    - No viene con el jdk por defecto por lo que si deseamos utilizarla debemos añadirla de forma manual a nuestro proyecto.
- Está nos ofrece un conjunto de mecanismos para poder controlar los ficheros de forma muy sencilla, por ejemplo las funciones:
    - Copia
    - Movimiento
-

```
try {
  FileUtils.moveFileToDirectory(file, destino,
true);
  System.out.println("Fichero movido");
} catch (IOException ex) {
  System.err.println("Error al mover el
archivo");
  ex.printStackTrace();
}
```

FILEUTILS

- Para poder realizar más operaciones s… …ils, de la librería de Apache Commons.
  - No viene con el jdk por defecto po… …de forma manual a nuestro proyecto.
- Está nos ofrece un conjunto de m… …el controlar los ficheros de forma muy sencilla, por ejemplo las funciones:
  - Copia
  - Movimiento
-

EJERCICIOS

1. Crea un directorio llamado ”ejercicios”
2. Crea un fichero llamado ejercicio1, dentro del directorio ejercicios
3. Muestra por pantalla la longitud del fichero con nombre “ejercicio1”
4. Crea un fichero llamado ejercicio2 , dentro del directorio ejercicios
5. Muestra todos los ficheros del directorio ejercicios
6. Elimina el fichero llamado ejercicio1
7. Muestra todos los ficheros del directorio ejercicios
8. Elimina nuevamente el fichero llamado fichero1.
    - ¿Has podido?

STREAMS – ACCESO SECUENCIAL

- Al momento de trabajar en Java con los streams deberemos diferenciar dos tipos de ellos:
    - Streams de bytes: Se basan en enviar la información en bloques de 8 bits, 1 byte, también son cocidos como ficheros binarios
    - Streams de caracteres: Se basan en enviar la información en bloques de 16 bits, 2 bytes, esto es debido a la codificación Unicode que usa 16bits para representar cada carácter(UTF-16), también conocidos como archivos de caracteres

- Es muy importante determinar que tipo de stream necesitamos y hacer la programación adecuada. La elección nos hará utilizar unas clases u otras.
    -

STREAMS – ACCESO SECUENCIAL

- Al momento de trabajar en Java con los strea
  - Streams de bytes: Se basan en enviar la
    como ficheros binarios
  - Streams de caracteres: Se basan en enviar la
    a la codificación Unicode que usa 16bits para r
    como archivos de caracteres

¿Un fichero Word es binario o de caracteres?

- Es muy importante determinar que t am necesitamos y hacer la programación adecuada.
  La elección nos hará utilizar unas s.
  -

STREAMS – ACCESO SECUENCIAL

- Al momento de trabajar en Java con los strea
  - Streams de bytes: Se basan en enviar la
    como ficheros binarios
  - Streams de caracteres: Se basan en enviar la
    a la codificación Unicode que usa 16bits para
    como archivos de caracteres

¿Un fichero Word es binario o de caracteres?

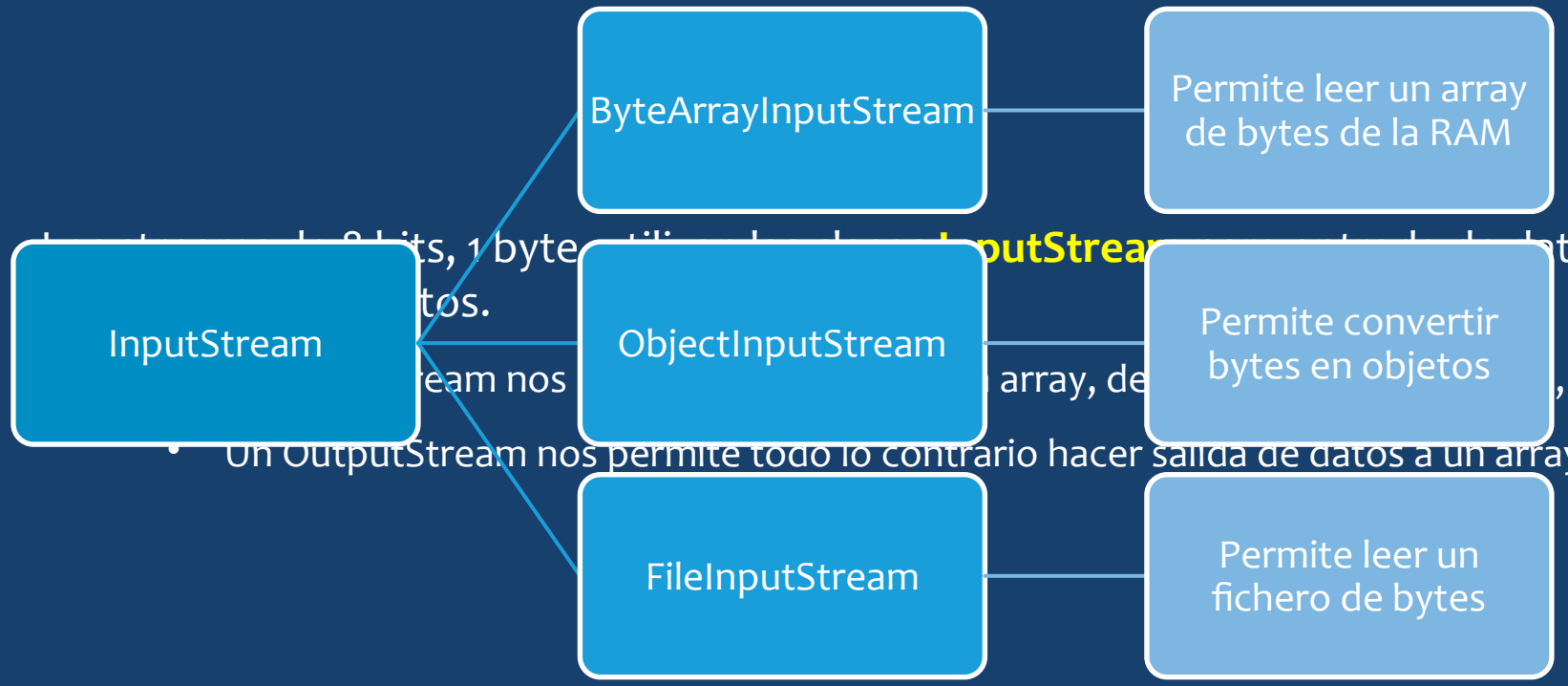
Aunque su uso principal es para texto, este es un archivo binario ya que puede incluir imágenes y formas por lo que necesita de un software especial para ser leído.

- Es muy importante determinar que t am necesitamos y hacer la programación adecuada.
  La elección nos hará utilizar unas s.
  -

STREAM 8 BITS

- Los streams de 8 bits, 1 byte, utilizan las clases **InputStream** para entrada de datos y **OutputStream** para salida de datos.
    - Un InputStream nos permite leer bytes de un array, de un String, de un fichero, etc.
    - Un OutputStream nos permite todo lo contrario hacer salida de datos a un array o un fichero

STREAM 8 BITS

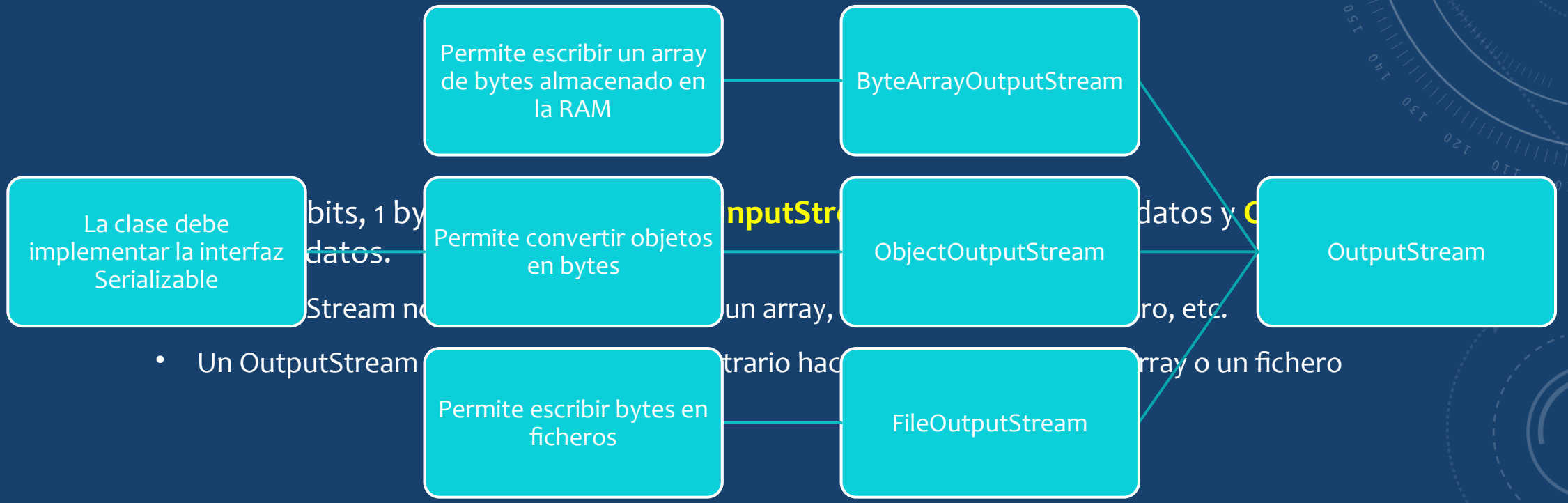


- ts, 1 byte putStrea tos y **OutputStream**
  tos.
  eam nos array, de , etc.
  - Un OutputStream nos permite todo lo contrario hacer salida de datos a un array o un fichero

STREAM 8 BITS

- Los streams de 8 bits, 1 byte, utilizan las clases **InputStream** para entrada de datos y **OutputStream** para salida de datos.
    - Un InputStream nos permite leer bytes de un array, de un String, de un fichero, etc.
    - Un OutputStream nos permite todo lo contrario hacer salida de datos a un array o un fichero

STREAM 8 BITS



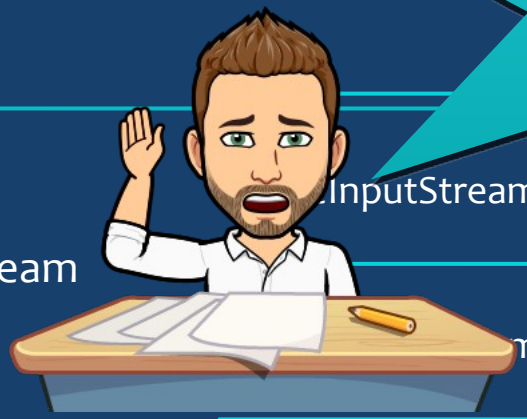
CONSTRUCTORES: *INPUTSTREAM

ByteArrayInputStream	ByteArrayInputStream(byte[] buf)	buf: buffer de entrada
	ByteArrayInputStream(byte[] buf, int offset,int length)	buf: Buffer de entrada offset: Desplazamiento inicial dentro del buffer length: Número máximo de bytes leídos en el buffer
ObjectInputStream	ObjectInputStream(InputStream in)	in: InputStream con los datos para leer
FileInputStream	FileInputStream(String name)	name: Nombre del fichero
	FileInputStream(File file)	file: Fichero de entrada

CONSTRUCTORES: *INPUTSTREAM

ByteArrayInputStream	ByteArrayInputStream(byte[] buf)	buf: buffer de entrada
	ByteArray… length)	…ial dentro del buffer …ytes leídos en el buffer
ObjectInputStream	ObjectInp…	…tos para leer
FileInputStream	…InputStream(String name)	name: Nombre del fichero
	…m(File file)	file: Fichero de entrada

Pero… ¿Qué es un buffer?
Un Buffer es un espacio de memoria en el que se almacenan los datos de forma temporal, una vez que los datos se leen estos se eliminan



CONSTRUCTORES: *OUTPUTSTREAM

ByteArrayOutputStream	**ByteArrayOutputStream(int size)**	size: Capacidad inicial
ObjectOutputStream	**ObjectOutputStream(OutputStream output)**	output: outputStream donde escribir los datos
FileOutputStream	FileOutputStream(String name)	name: Nombre del fichero de salida
	FileOutputStream(String name, boolean append)	name: Nombre del fichero de salida append: opción para escribir al inicio o final de fichero
	FileOutputStream(File file)	file: Fichero de salida
	FileOutputStream(File file, boolean append)	file: Fichero de salida append: opción para escribir al inicio o final de fichero

EJEMPLO: CONVERTIR UN OBJETO EN UN FICHERO

```
File file = new File("ficheros/8bits");
FileOutputStream fileOutputStream = new FileOutputStream(file);
Ejemplo ejemplo = new Ejemplo(1, "Texto de prueba para el ejemplo");
System.out.println("Ejemplo antes de fichero:");
System.out.println(ejemplo);
byte[] bytes = objetToBytes(ejemplo);
fileOutputStream.write(bytes);
fileOutputStream.close();
```

```
private static byte[] objetToBytes(Object object) {
  byte[] bytes = new byte[] {};
  try {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    ObjectOutputStream oos = new ObjectOutputStream(baos);
    oos.writeObject(object);
    bytes = baos.toByteArray();
    baos.close();
    oos.close();
  } catch (IOException ex) {
    System.err.println("Error en la descomposición del fichero");
    ex.printStackTrace();
  }
  return bytes;
}
```

EJEMPLO: CONVERTIR UN OBJETO EN UN FICHERO

```
File file = new File("ficheros/8bits");
FileOutputStream fileOutputStream = new FileOutputStream(file);
Ejemplo ejemplo = new Ejemplo(1, "Texto de prueba para el ejemplo");
System.out.println("Ejemplo antes de fichero:");
System.out.println(ejemplo);
byte[] bytes = objetToBytes(ejemplo);
fileOutputStream.write(bytes);
fileOutputStream.close();
```

```
private static byte[] objetToBytes(Object object) {
  byte[] bytes = new byte[] {};
  try {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    ObjectOutputStream oos = new ObjectOutputStream(baos);
    oos.writeObject(object);
    bytes = baos.toByteArray();
    baos.close();
    oos.close();
  } catch (IOException ex) {
    System.err.println("Error en la descomposición del fichero");
    ex.printStackTrace();
  }
  return bytes;
}
```

Es importante cerrar los streams para asegurarnos que todos los datos han sido procesados (y así limpiar los buffers)

EJEMPLO: CONVERTIR UN FICHERO EN UN OBJETO

```
File file = new File("ficheros/8bits");
FileInputStream fileInputStream = new FileInputStream(file);
Ejemplo ejemplo1 = (Ejemplo)
byteToObject(fileInputStream.readAllBytes());
fileInputStream.close();
System.out.println("Ejemplo despues del fichero");
System.out.println(ejemplo1);
```

```
private static Object byteToObject(byte[] bytes) {
  Object object = new Object();
  try {
    ByteArrayInputStream bais = new ByteArrayInputStream(bytes);
    ObjectInputStream ois = new ObjectInputStream(bais);
    object = ois.readObject();
    ois.close();
    bais.close();
  } catch (IOException | ClassNotFoundException ex) {
    System.err.println("Error en la creación del objecto desde un fichero");
    ex.printStackTrace();
  }
  return object;
}
```

PUNTOS IMPORTANTES

- Todo objeto que se desee guardar en un fichero deberá implementar la interfaz Serializable
- public class Ejemplo implements Serializable
- Existen diferentes tipos de Exceptions que deben ser controladas:
    - IOException: Excepción producida al leer o escribir datos
    - FileNotFoundException: Excepción producida cuando no encuentra el fichero solicitado
    - ClassNotFoundException: Excepción producida cuando la JVM no es capaz de recuperar la clase que se está indicando.

STREAM DE 16 BITS

- Los streams de 16 bits, 2 bytes, orientados a caracteres utilizan unas clases que ya están preparadas para esta lectura de datos. En este caso nos encontramos con las clases **Reader** y **Writer**.
- De estás clases utilizaremos las implementaciones de FileReader y FileWriter para nuestras operaciones de lectura y escritura respectivamente para tratar los ficheros.
    - Al crear un objeto de la clase FileWriter se puede pasar un segundo parámetro de tipo boolean, este especificará si escribirá en el fichero desde el inicio (false) o continuará desde el último punto (true).
-
-

CONSTRUCTORES

FileReader	FileReader(String name)
	FileReader(String File)
	FileReader(String name, Charset charset)
	FileReader(File file, Charset charset)
FileWriter	FileWriter(String name)
	FileWriter(String name, boolean append)
	FileWriter(File file)
	FileWriter(File file, boolean append)
	FileWriter(String name, Charset charset)
	FileWriter(String name, Charset charset, boolean append)
	FileWriter(File file, Charset charset)
	FileWriter(File file, Charset charset, boolean append)

CONSTRUCTORES

FileReader

- FileReader(String name)
- FileReader(String File)
- FileReader(String name, Charset charset)
- FileReader(File file, Charset charset)

FileWriter

- FileWriter(String name)
- FileWriter(String name, boolean append)
- FileWriter(File file)
- FileWriter(File file, boolean append)
- FileWriter(String name, Charset charset)
- FileWriter(String name, Charset charset, boolean append)
- FileWriter(File file, Charset charset)
- FileWriter(File file, Charset charset, boolean append)

Un charset es la codificación de los caracteres. Nos permite leer caracteres que de forma inicial no están previstos como símbolos de otros lenguajes

EJEMPLO: ESCRIBIR EN UN FICHERO

```
File file = new File("ficheros/caracteres.txt");
try {
  FileWriter fileWriter = new FileWriter(file);
  fileWriter.write("Esto es un texto de prueba");
  fileWriter.close();
} catch (IOException ex) {
  System.err.println("Error de apertura/escritura en el fichero: " +
file.getName());
}
```

EJEMPLO: LEER DE UN FICHERO

```
try {
  File file = new File("ficheros/caracteres.txt");
  FileReader fileReader = new FileReader(file);
  int read;
  while ((read = fileReader.read()) != -1) {
    System.out.print((char) read);
  }
  System.out.println();
} catch (IOException ex) {
  System.err.println("Error de apertura/esc[illegible] en el fichero: " +
file.getName());
}
```

La función read, lee en cada vuelta del bucle 2 bytes, correspondientes a 1 carácter.
Esta función realiza la lectura en número enteros por lo que debemos convertirlo en un carácter. Esta última acción se puede hacer con un simple casteo ya que cada número entero tiene una representación en forma de carácter.
Podemos ver estás equivalencias gracias a una tabla de código ASCII
https://elcodigoascii.com.ar

EJERCICIOS

9. Crea una nueva clase llamada Persona con los atributos (id,nombre,edad, dni).
    1. Crea una función para guardar un objeto Persona en un fichero con el nombre persona1
    2. Crea una función para recuperar un objeto Persona del fichero persona1
    3. Modifica sus propiedades y vuelve a guardarlo en el fihcero persona 1

10. Crea un fichero de texto utilizando la clase FileWriter
    - El fichero debe contener la información:

    “Esto es un texto de prueba,

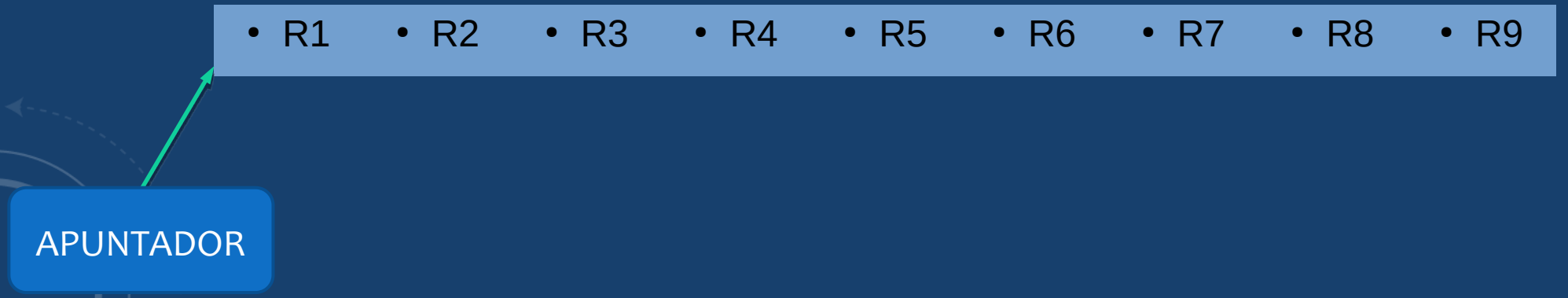
    Estamos creando nuestro primer fichero de texto

    https://codigonline.com”

11. Lee un fichero de tipo imagen y muestra su contenido por pantalla.
    - ¿Se puede leer de forma correcta?

ACCESO ALEATORIO

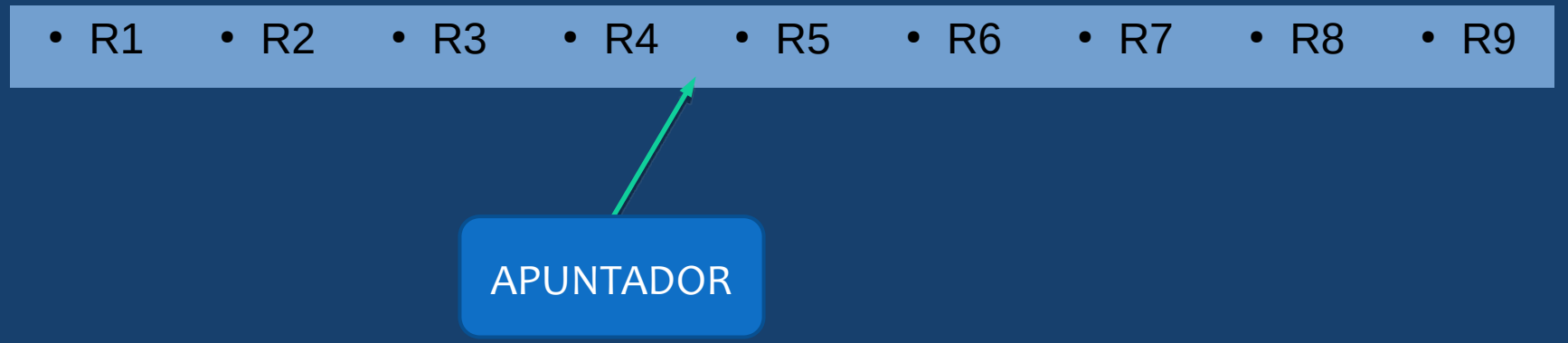
- Para el acceso secuencial hemos visto diferentes clases ya sean para trabajar con ficheros binarios (1byte) o ficheros de caracteres (2 bytes).
- Para el acceso aleatorio la cosa se simplifica y únicamente disponemos de una sola clase. RandomAccessFile, esta nos proporciona todos los mecanismos para movernos por un fichero de forma aleatorio utilizando un apuntador que se va moviendo por los registros.



ACCESO ALEATORIO

- Para el acceso secuencial hemos visto diferentes clases ya sean para t
  (1byte) o ficheros de caracteres (2 bytes).
- Para el acceso aleatorio la cosa se simplifica y únicamente disponemos de un
  RandomAccessFile, esta nos proporciona todos los mecanismos para movernos por un fichero de forma aleatorio utilizando un apuntador que se va moviendo por los registros.

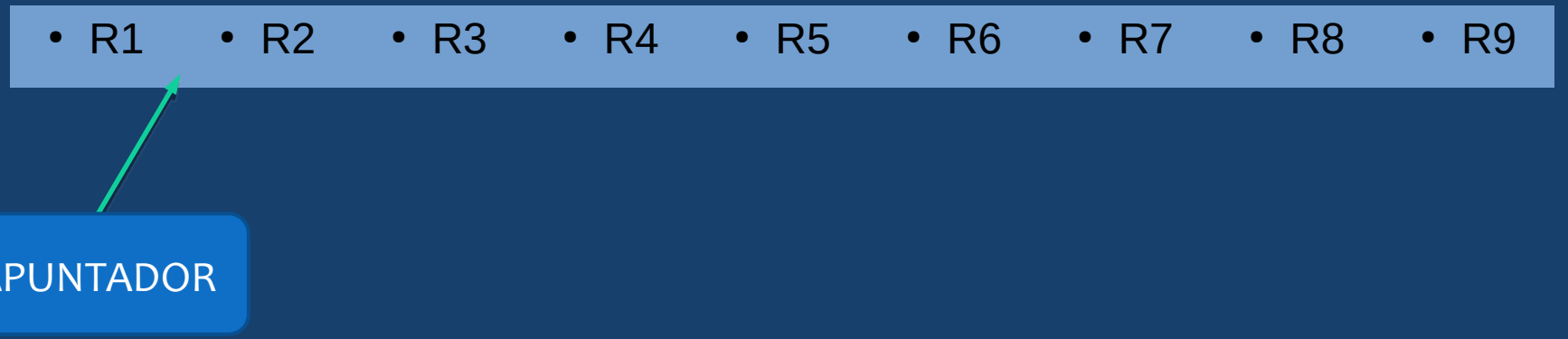
VAMOS AL REGISTRO 5 (R5)



ACCESO ALEATORIO

- Para el acceso secuencial hemos visto diferentes clases ya sean para t
  (1byte) o ficheros de caracteres (2 bytes).
- Para el acceso aleatorio la cosa se simplifica y únicamente disponemos de un
  RandomAccessFile, esta nos proporciona todos los mecanismos para movernos por un fichero de forma aleatorio utilizando un apuntador que se va moviendo por los registros.

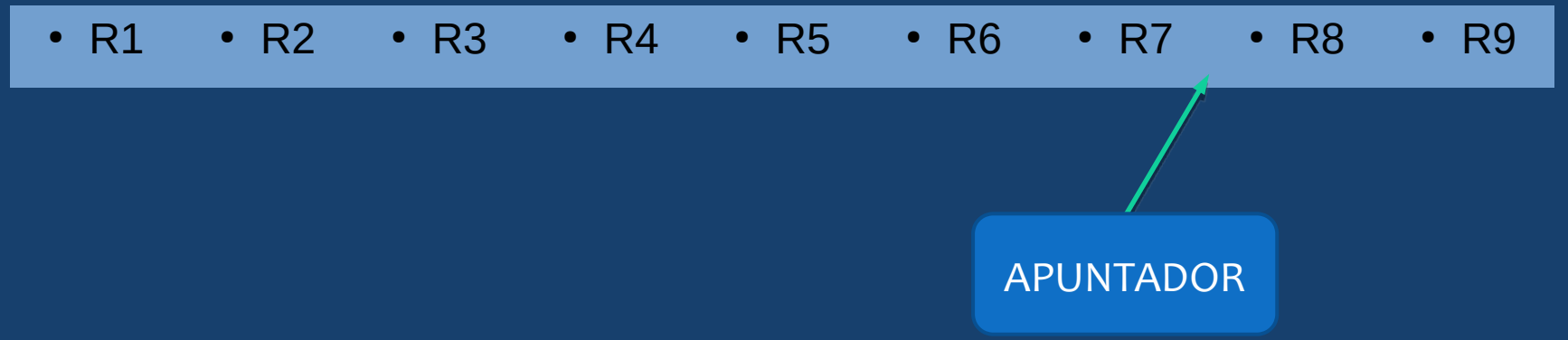
VAMOS AL REGISTRO 2 (R2)



ACCESO ALEATORIO

- Para el acceso secuencial hemos visto diferentes clases ya sean para t
  (1byte) o ficheros de caracteres (2 bytes).
- Para el acceso aleatorio la cosa se simplifica y únicamente disponemos de un
  RandomAccessFile, esta nos proporciona todos los mecanismos para movernos por un fichero de forma aleatorio utilizando un apuntador que se va moviendo por los registros.

VAMOS AL REGISTRO 8 (R8)



CONSTRUCTORES

RandomAccesFile

RandomAccesFile(String name, String accesMode)

name: Nombre del ficero a abrir

accesMode: Modo de acceso al fichero

r: lectura

rw: lectura y escritura

RandomAccesFile(File file, String accesMode)

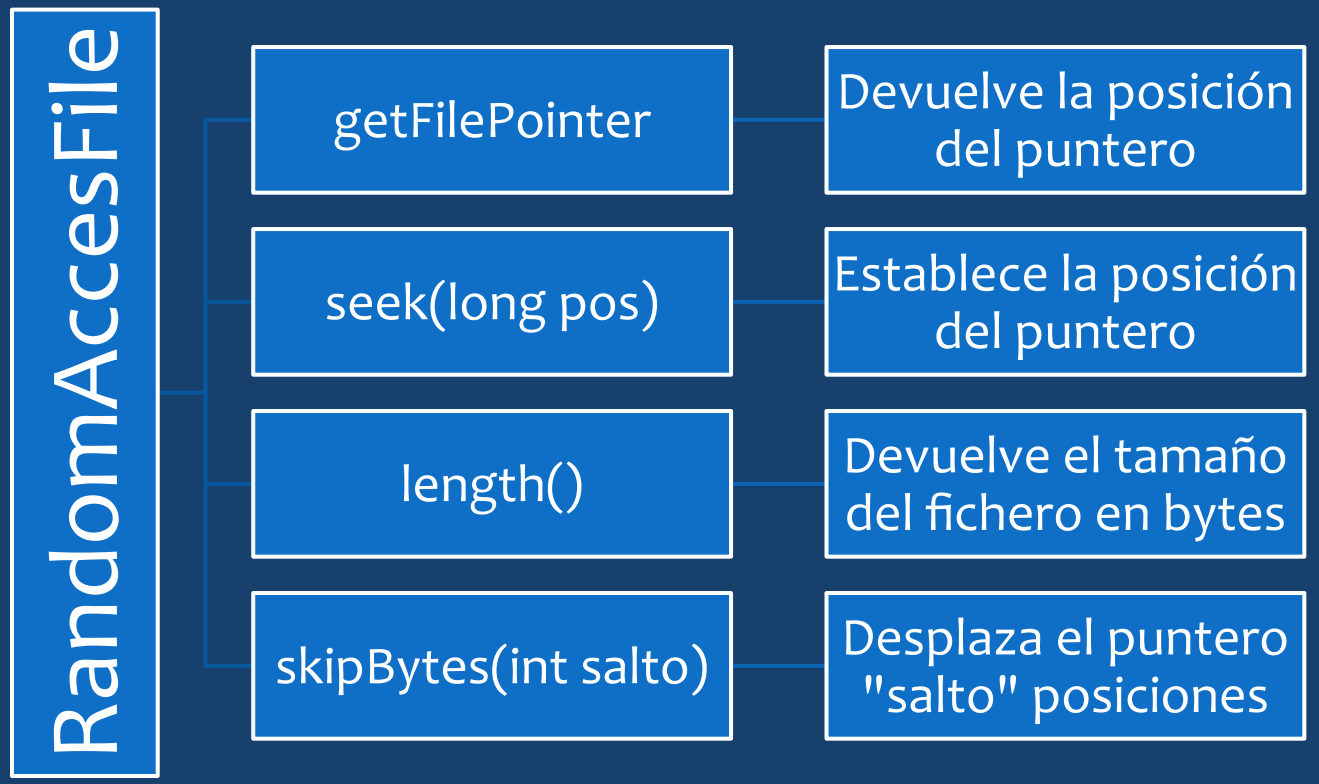
file: fichero a abrir

accesMode: Modo de acceso al fichero

r: lectura

rw: lectura y escritura

FUNCIONES MÁS IMPORTANTES



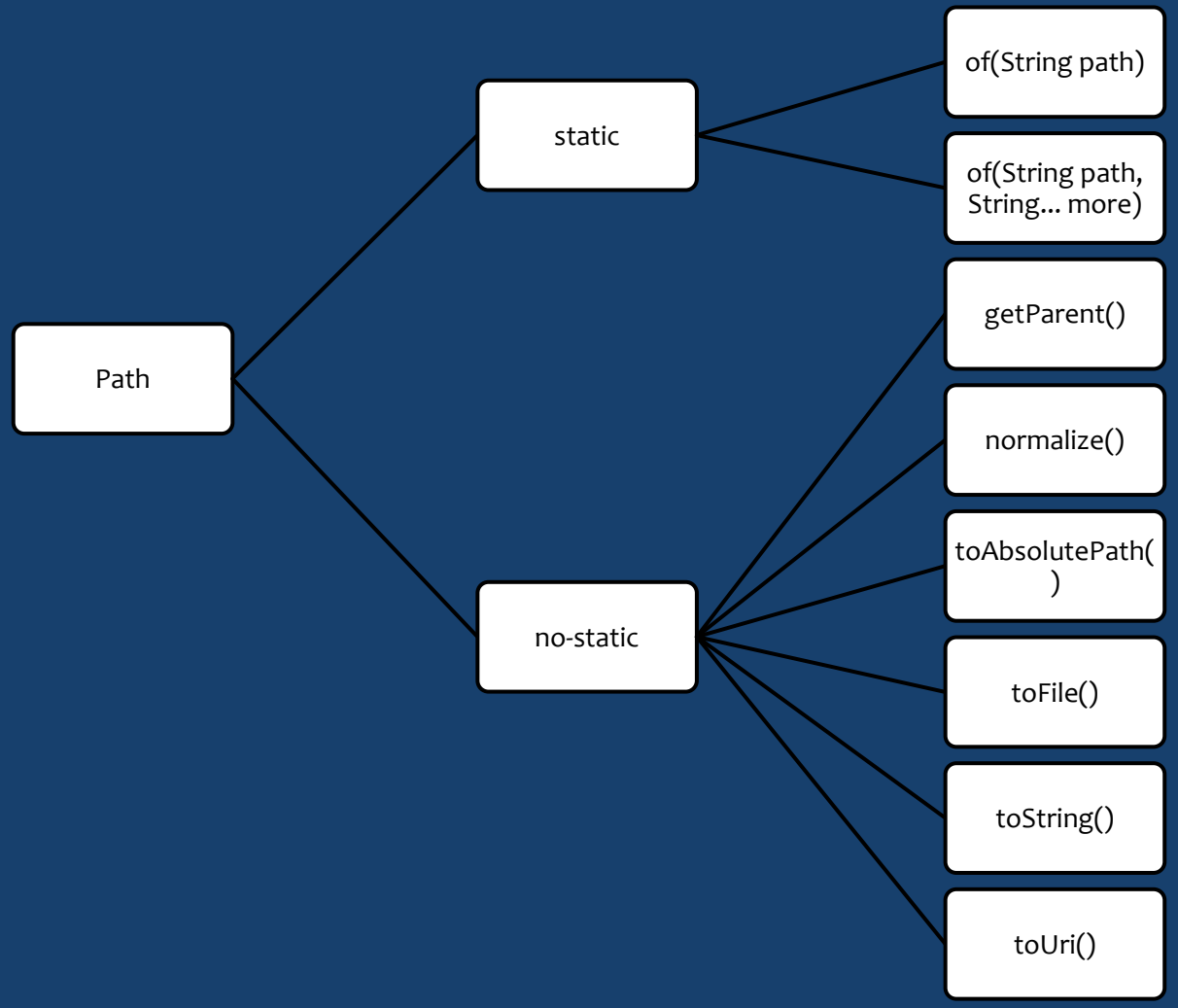
EJEMPLO RANDOMACCESFILE

```
File file = new File("teoria/aleatorio.txt");
try {
  RandomAccessFile randomAccessFile = new       RandomAccessFile(file, "rw");
  System.out.println("Posición del puntero: " + randomAccessFile.getFilePointer());
  String texto = "Linea 1\nLinea 2";
  randomAccessFile.seek(0);
  randomAccessFile
      .write(texto.getBytes(StandardCharsets.UTF_8));
  System.out.println("Posición del puntero: " + randomAccessFile.getFilePointer());
  randomAccessFile.seek(0);
  String line;
  while ((line= randomAccessFile.readLine())!=null){
    System.out.println("Texto leído del fichero:");
    System.out.println(line);
  }
  System.out.println("Posición del puntero: " + randomAccessFile.getFilePointer());
  randomAccessFile.seek(2);
  System.out.println("Posición del puntero: " + randomAccessFile.getFilePointer());
  randomAccessFile.close();
} catch (FileNotFoundException e) {
  System.err.println("No se ha encontrado el fichero: " + file.getName());
  e.printStackTrace();
} catch (IOException e) {
  System.err.println("Error al recuperar o insertar información");
  e.printStackTrace();
}
```

JAVA.NIO

- Java.nio incluye nuevas clases para trabajar con ficheros
  - Path: Sirve para manejar las rutas de los ficheros.
  - Files: Sirve para controlar las operaciones básicas de los ficheros
  - FileSystem: Sirve para obtener referencias al sistema de archivos

PATH – FUNCIONES MÁS IMPORTANTES



FILES – FUNCIONES MÁS IMPORTANTES

Files

copy(Path source, Path target)

copy(Path source, OutputStream out)

~~copy(InputStream in, Path target)~~

createDirectory(Path dir)

createFile(Path path)

delete(Path path)

deleteIfExists(Path path)

exists(Path path)

readAllLines(Path path)

readAllBytes(Path path)

write(Path path, byte[] bytes)

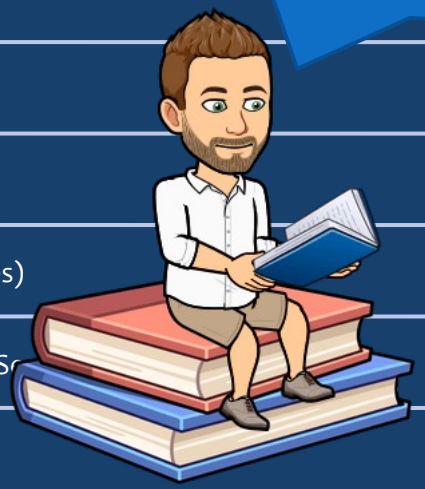
writeString(Path path, CharSequence chr)

FILES – FUNCIONES MÁS IMPORTANTES

Files

copy(Path source, Path target)
copy(Path source, OutputStream out)
~~copy(InputStream in, Path target)~~
createDirectory(Path dir)
createFile(Path path)
delete(Path path)
deleteIfExists(Path path)
exists(Path path)
readAllLines(Path path)
readAllBytes(Path path)
write(Path path, byte[] bytes)
writeString(Path path, CharS

Todas las funciones de la clase Files son estáticas.



EJEMPLO – LISTADO DE ARCHIVOS

```
Path path = Path.of("teoria");
try {
  Stream<Path> files = Files.list(path);
  for(Path path1: files.collect(Collectors.toList())){
    System.out.println(path1);
  }
} catch (IOException e) {
  System.err.println("Error al leer el directorio: "+path.getFileName());
  e.printStackTrace();
}
```

EJEMPLO – ESCRIBIR BYTES

```
Path path = Path.of("teoria/nio.txt");
byte[] a = { 20, 10, 30, 5 };
System.out.println("Byte[] inicial");
for (byte item: a){
  System.out.println(item);
}
try {
  Files.write(path, a);
} catch (IOException ex) {
  System.err.println("Error al escribir los bytes");
}
```

EJEMPLO – LEER BYTES

```
Path path = Path.of("teoria/nio.txt");
try {
  byte[] b = Files.readAllBytes(path);
  System.out.println("Byte[] recuperado");
  for (byte item: b){
    System.out.println(item);
  }
}catch (IOException ex){
  System.err.println("Error al leer los bytes");
}
```

EJEMPLO – ESCRIBIR CARACTERES

```
Path path = Path.of("teoria/nio.txt");
try {
   Files.writeString(path, "Esto es un texto de prueba");
} catch (IOException ex) {
   System.err.println("Error al escribir en el fichero: " +
path.getFileName());
}
```

EJEMPLO – ESCRIBIR CARACTERES

```
Path path = Path.of("teoria/nio.txt");
try {
   String texto = Files.readString(path);
   System.out.println(texto);
}catch (IOException ex){
   System.err.println("Error al leer el fichero: "+path.getFileName());
}
```

EXTRAS CLASE FILES

- Adicionalmente la clase Files nos ofrece mecanismos para copiar y mover archivos que no se encontraba en la clase File.
- Estas operaciones son muy sencillas de realizar y no necesitan de librerías externas ~~(FileUtils)~~.
    - Copia:
    - Movimiento:

```
Path path = Path.of("teoria/nio.txt");
Path copy = Path.of("teoria/nio_copia.txt");
try {
   Files.copy(path, copy);
}catch (IOException ex){
   System.err.println("No se ha podido copiar el
fichero");
}
```

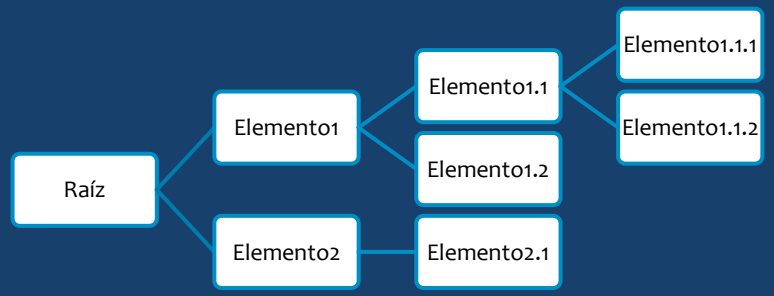
EXTRAS CLASE FILES

- Adicionalmente la clase Files nos ofrece mecanismos para copiar y mover archivos que no se encontraba en la clase File.
- Estas operaciones son muy sencillas de realizar y no necesitan de librerías externas ~~(FileUtils)~~.
    - Copia:
    - Movimiento:

```
Path path = Path.of("teoria/nio.txt");
Path move = Path.of("teoria/tema1/nio.txt");
try {
   Files.move(path, move);
} catch (IOException ex) {
   System.err.println("No se ha podido mover el
fichero");
}
```

FICHEROS XML

- XML: Las siglas de XML vienen de eXtensible Markup language.
- Este nos permite crear estructuras anidando los elementos unos dentro de otros y definir el contenido de cada elemento.
- Algo muy importante de XML es que sigue una jerarquía en forma de árbol, por lo que todo elemento (excepto la ráiz) tiene un único padre, pero cada elemento puede tener muchos hijos.
    - De la misma forma no se pueden dar elementos cíclicos, un elemento no puede ser padre e hijo de otro elemento



FICHEROS XML

- Los documentos XML son muy fáciles de leer, ya que tiene una sintaxis muy sencilla.
    - Está basado en etiquetas que tienen un nombre y a su vez pueden tener atributos
- Son muy usados en ficheros de configuración de algunos programas o protocolos SOAP para enviar información a los servidores y ejecutar diferentes rutinas.
-
- Etiquetas XML
    - Cada etiqueta XML tendrá un inicio que se definirá dentro de los símbolos "<*etiqueta*>"
    - De la misma forma tendrá un fin que se definirá dentro de los símbolos </etiqueta>
    - Entre la etiqueta de inicio y de fin pueden ir otras etiquetas creando así la estructura
- Atributos XML
    - A su vez una etiqueta puede tener 0..* atributos, elementos opcionales que dotan al XML de mayor información.
    - Algo muy importante es que toda información en atributos puede ser representada en elementos, mientras que no todos los elementos pueden ser atributos.
        - ¿Cómo saber donde va la información?
        - La única forma de determinarlo es saber si el elemento estará repetido y si tendrá hijos. En estos dos casos la información no puede ir en atributos

EJEMPLO XML

```
<pizzas>
  <pizza>
    <nombre>Barbacoa</nombre>
    <ingredientes>
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza>
    <nombre>Cuatro Quesos</nombre>
    <ingredientes>
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

Equivalentes, a excepción de la cantidad en ingredientes

```
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

EJEMPLO XML

```
<pizzas>
  <pizza>
    <nombre>Barbacoa</nombre>
    <ingredientes>
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza>
    <nombre>Cuatro Quesos</nombre>
    <ingredientes>
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

```
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

EJEMPLO XML

ETIQUETA

```
<pizzas>
  <pizza>
    <nombre>Barbacoa</nombre>
    <ingredientes>
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza>
    <nombre>Cuatro Quesos</nombre>
    <ingredientes>
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

```
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

EJEMPLO XML

ETIQUETA

ATRIBUTOS

```
<pizzas>
  <pizza>
    <nombre>Barbacoa</nombre>
    <ingredientes>
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza>
    <nombre>Cuatro Quesos</nombre>
    <ingredientes>
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

```
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

EJEMPLO XML

```
<pizzas>
  <pizza>
    <nombre>Barbacoa</nombre>
    <ingredientes>
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza>
    <nombre>Cuatro Quesos</nombre>
    <ingredientes>
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

ETIQUETA

ATRIBUTOS

VALORES

```
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

EJEMPLO XML

```
<pizzas>
  <pizza>
    <nombre>Barbacoa</nombre>
    <ingredientes>
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza>
    <nombre>Cuatro Quesos</nombre>
    <ingredientes>
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

ETIQUETA

ATRIBUTOS

VALORES

```
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

EJEMPLO XML

```
<pizzas>
  <pizza>
    <nombre>Barbacoa</nombre>
    <ingredientes>
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza>
    <nombre>Cuatro Quesos</nombre>
    <ingredientes>
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

ETIQUETA

ATRIBUTOS

VALORES

```
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

EJEMPLO XML

```
<pizzas>
  <pizza>
    <nombre>Barbacoa</nombre>
    <ingredientes>
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza>
    <nombre>Cuatro Quesos</nombre>
    <ingredientes>
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

ETIQUETA

ATRIBUTOS

VALORES

```
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa"</ingrediente>
      <ingrediente>Mozzarella"</ingrediente>
      <ingrediente>Pollo"</ingrediente>
      <ingrediente>Bacon"</ingrediente>
      <ingrediente>Ternera"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>"Tomate"</ingrediente>
      <ingrediente>"Queso Azul"</ingrediente>
      <ingrediente>"Queso gorgonzola"</ingrediente>
      <ingrediente>"Queso cremoso"</ingrediente>
      <ingrediente>"Queso parmesano"</ingrediente>
      <ingrediente>"Aceite Oliva"</ingrediente>
      <ingrediente>"Orégano"</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

FICHEROS XML

- Existe formas diferentes de poder leer ficheros XML, ver su estructura y atributos. Estas herramientas son conocidas como XML-parser
- Las más utilizadas son DOM y SAX
    - DOM: Document Object Model, el procesador lee todo el documento XML y lo almacena en memoria RAM.
        - Es muy útil cuando queremos acceder de forma rápida a un elemento del árbol, ya que tiene toda la información precargada
        - Contra más grande es el documento más tiempo y memoria necesita para procesarlo
    - SAX: Simple Api for XML, el procesador va leyende de forma secuencial el fichero y va lanzando eventos que nuestro programa debe capturar. Para cada etiqueta de inicio/fin, atributo y valor emitirá un evento.
        - Es mucho más rápido que DOM y consume menos memoria.
        - En contra si queremos acceder a un elemento en concreto debemos de recorrer todo el documento.
-

FICHEROS XML - DOM

- Clases más importantes:
  - DocumentBuilderFactory: Es una clase especial, nos da la capacidad de poder crear parsers para nuestro programa.
  - DocumentBuilder: Define el parser DOM que se va a utilizar.
  - Document: Objeto que contiene la lectura completa del XML.
  - Node: Representa a cualquier Nodo del árbol.
  - NodeList: Lista que contiene todos los nodos hijos de un nodo.
  - Element: Es un tipo de nodo, representa un elemento del XML
  - Attr: Representa un atributo de un Nodo
  - Text: Representa el texto de un elemento
-
- Gracias a estas clases podemos leer/escribir documentos XML utilizando DOM

DOM – LEER ARCHIVOS

- Antes de empezar, debemos tener en cuenta que los espacios en blanco dentro de una etiqueta los lee como texto y debemos tener mucho cuidado al momento de tratar el fichero.
    - Recomiendo utilizar algún simplificador de XML como por ejemplo https://codebeautify.org/xmlviewer, donde podremos eliminar todos los espacios en blanco dándole al botón de *Minify*
- Pasos para leer un documento:
    1. Crear el DocumentBuilderFactory
    2. Crear el DocumentBuilder
    3. Crear el Document, este punto es muy importante ya que cargará todo el documento en memoria.
    4. Leer las propiedades del documento deseadas
        1. Este último paso variará según el documento y las propiedades que deseamos leer

EJEMPLO – DOM LECTURA

```
Path path = Path.of("teoria/tema1/pizzas.xml");
File file = path.toFile();
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder;
try {
   builder = factory.newDocumentBuilder();
} catch (ParserConfigurationException ex) {
   System.err.println("Error al crear el parser");
   ex.printStackTrace();
   return;
}
Document document = null;
try {
   document = builder.parse(file);
} catch (IOException | SAXException ex) {
   System.err.println("Error al parsear el fichero " + file.getName());
}
```

```
NodeList pizzas =
document.getElementsByTagName("pizzas").item(0).getChildNodes();
System.out.println("Pizzas en el menu");
for (int i = 0; i < pizzas.getLength(); i++) {
   Node node = pizzas.item(i);
   if (node.getNodeType() == Node.ELEMENT_NODE)
     switchElement(node);
}
```

```
private static void switchElement(Node node) {
   Element element = (Element) node;
   switch (element.getNodeName()) {
      case "pizza":
         System.out.print("Pizza: \t");
         System.out.println(getAttribute(element,
"nombre"));
         switchElement(element.getFirstChild());
         break;
      case "ingredientes":
         System.out.println("\tLista de ingredientes:");
         NodeList ingredientes = element.getChildNodes();
         for (int i = 0; i < ingredientes.getLength(); i++) {
            switchElement(ingredientes.item(i));
         }
         break;
      case "ingrediente":
         System.out.print("\t\tIngrediente: ");
         System.out.println(getText(element));
         break;
      default:
   }
}
```

```
private static String getAttribute(Element element,
String name) {
   return element.getAttribute(name);
}

private static String getText(Element element) {
   return element.getTextContent();
}
```

EJEMPLO – DOM LECTURA

```
Path path = Path.of("teoria/tema1/pizzas.xml");
File file = path.toFile();
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder;
try {
   builder = factory.newDocumentBuilder();
} catch (ParserConfigurationException ex) {
   System.err.println("Error al crear el parser");
   ex.printStackTrace();
   return;
}
Document document = null;
try {
   document = builder.parse(file);
} catch (IOException | SAXException ex) {
   System.err.println("Error al parsear el fichero " + file.getName());
}
```

```
private static void switchElement(Node node) {
   Element element = (Element) node;
   switch (element.getNodeName()) {
      case "pizza":
         System.out.print("Pizza: \t");
         System.out.println(getAttribute(element,
"nombre"));
         switchElement(element.getFirstChild());
         break;
      case "ingredientes":
         System.out.println("\tLista de ingredientes:");
         NodeList ingredientes = element.getChildNodes();
         for (int i = 0; i < ingredientes.getLength(); i++) {
            switchElement(ingredientes.item(i));
         }
         break;
      case "ingrediente":
         System.out.print("\t\tIngrediente: ");
         System.out.println(getText(element));
         break;
      default:
   }
}
```

```
private static String getAttribute(Element element,
String name) {
   return element.getAttribute(name);
}

private static String getText(Element element) {
   return element.getTextContent();
}
```

EJEMPLO – DOM LECTURA

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pizzas>
  <pizza nombre="Barbacoa">
    <ingredientes cantidad="6">
      <ingrediente>Salsa Barbacoa</ingrediente>
      <ingrediente>Mozzarella</ingrediente>
      <ingrediente>Pollo</ingrediente>
      <ingrediente>Bacon</ingrediente>
      <ingrediente>Ternera</ingrediente>
      <ingrediente>Aceite Oliva</ingrediente>
    </ingredientes>
  </pizza>
  <pizza nombre="Cuatro Quesos">
    <ingredientes cantidad="7">
      <ingrediente>Tomate</ingrediente>
      <ingrediente>Queso Azul</ingrediente>
      <ingrediente>Queso gorngonzola</ingrediente>
      <ingrediente>Queso cremoso</ingrediente>
      <ingrediente>Queso parmesano</ingrediente>
      <ingrediente>Aceite Oliva</ingrediente>
      <ingrediente>Oregano</ingrediente>
    </ingredientes>
  </pizza>
</pizzas>
```

TRANSFORMACIÓN

```
Pizzas en el menu
Pizza:   Barbacoa
Ingredientes
        Lista de ingredientes:
                Ingrediente: Salsa Barbacoa
                Ingrediente: Mozzarella
                Ingrediente: Pollo
                Ingrediente: Bacon
                Ingrediente: Ternera
                Ingrediente: Aceite Oliva
Pizza:   Cuatro Quesos
Ingredientes
        Lista de ingredientes:
                Ingrediente: Tomate
                Ingrediente: Queso Azul
                Ingrediente: Queso gorngonzola
                Ingrediente: Queso cremoso
                Ingrediente: Queso parmesano
                Ingrediente: Aceite Oliva
                Ingrediente: Oregano
```

DOM – ESCRIBIR ARCHIVOS

- Escribir un documento XML con DOM es muy sencillo (**pero bastante tedioso**), únicamente debemos ir línea a línea estableciendo las características que deseamos.
- Pasos para escribir un documento:
    1. Crear el DocumentBuilderFactory
    2. Crear el DocumentBuilder
    3. Crear el DOMImplementation (**Difiere de la lectura donde no hace falta**)
    4. Crear el Document, este punto es muy importante ya que cargará todo el documento en memoria.
    5. Escribir las propiedades deseadas
    6. Transformar el DOM en memoria en un archivo del disco
        1. Se necesita un objeto Source (origen de los datos) y un Result (destino de los datos)
        2. Se enviará del origen al destino mediante la clase Transform

EJEMPLO – DOM ESCRITURA

```
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder = null;
try {
   builder = factory.newDocumentBuilder();
} catch (ParserConfigurationException e) {
   e.printStackTrace();
}
assert builder != null;
DOMImplementation implementation =
builder.getDOMImplementation();
Document document = implementation.createDocument(null, null, null);
document.setXmlVersion("1.0");
document.setXmlStandalone(true);
```

```
Element pizzas =
document.createElement("Pizzas");
document.appendChild(pizzas);
crearPizza(pizzas, document);
```

```
Source source = new DOMSource(document);
Result result = new StreamResult(new
File("teoria/tema1/pizzas_dom.xml"));
Transformer transformer =
TransformerFactory.newInstance().newTransformer();
transformer.transform(source, result);
```

```
private static void crearPizza(Element element, Document document) {
   Element pizza = document.createElement("Pizza");
   pizza.setAttribute("nombre", "Barbacoa");
   ArrayList<Ingrediente> ingredienteList = new ArrayList<>() {{
      add(new Ingrediente("Salsa barbacoa", 200.0));
      add(new Ingrediente("Mozzarella", 100));
      add(new Ingrediente("Pollo", 20));
      add(new Ingrediente("Bacon", 20));
      add(new Ingrediente("Ternera", 20));
      add(new Ingrediente("Aceite de Oliva", 10));
   }};
   Element ingredientes = document.createElement("ingredientes");
   ingredienteList.forEach(ingrediente -> {
      Element element1 = document.createElement("ingrediente");
      Text nombreIngrediente =
document.createTextNode(ingrediente.nombre);
      element1.appendChild(nombreIngrediente);
      Attr cantidadIngrediente = document.createAttribute("cantidad");
      cantidadIngrediente.setValue(String.valueOf(ingrediente.cantidad));
      element1.setAttributeNode(cantidadIngrediente);
      ingredientes.appendChild(element1);
   });
   pizza.appendChild(ingredientes);
   element.appendChild(pizza);
}
```

FICHEROS XML - SAX

- Los parsers de tipo SAX funcionan de forma diferente a los de DOM, van leyende el documento poco a poco y cada vez que encuentren un elemento, atributo, texto, etc., lanzarán un evento para que lo capturemos y podamos actuar en consecuencia.
- La API de SAX es mucho más compleja que la de DOM pero vamos a analizarla.
- Clases más importantes:
    - SAXParserFactory → Clase especial, nos permitirá crear nuevos SAX parsers
    - SAXParser → Creación de un nuevo parser para SAX, utiliza la clase anterior
    - XMLReader → Objeto que permite leer el documento XML elemento a lemento
    - DefaultHandler → Clase abstracta que debemos implementar, contiene las llamadas a los eventos

FICHEROS XML - SAX

- La clase DefaultHandler entre otras, contiene las siguientes funciones que son las que más nos interesan
  - startDocument()
  - endDocument()
  - startElement() → Elemento que empieza, podemos en este punto podemos leer los atributos si los contiene
  - endElement()
  - characters() → Caracteres que contiene el elemento

FICHEROS XML - SAX

- Para leer un fichero XML con SAX seguiremos los siguientes pasos:
    1. Crear SAXParserFactory
    2. Crear el SAXParser
    3. Crear el XMLReader
    4. Implementar el DefaultHandler
        1. Dar funcionalidad a las diferentes funciones según nuestros requisitos
    5. Parsear el documento

EJEMPLO – SAX LECTURA

```
SAXParserFactory parserFactory = SAXParserFactory.newInstance();
SAXParser parser = parserFactory.newSAXParser();
XMLReader reader = parser.getXMLReader();
```

1

```
reader.parse("teoria/tema1/pizzas.xml");
```

3

```
reader.setContentHandler(new DefaultHandler() {
  @Override
  public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
    switch (qName) {
      case "pizzas":
        System.out.println(qName);
        break;
      case "pizza":
        System.out.println("\tNombre: " + attributes.getValue(0));
        break;
      case "ingredientes":
        System.out.println("\t\tListado de ingredientes");
        break;
      case "ingrediente":
        System.out.print("\t\t\tIngrediente: ");
        break;
    }
  }
  @Override
  public void characters(char[] ch, int start, int length) throws SAXException {
    String text = new String(ch, start, length);
    System.out.println(text);
  }
});
```

2

EJERCICIOS

- A partir del siguiente documento XML, https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms762271(v=vs.85)
    12. Muestra por pantalla los diferents id de cada libro utilizando la librería DOM
    13. Muestra por pantalla una lista de autores y los títulos de sus libros
    14. Muestra por pantalla los títulos de los libros y sus precios. Ordena de más económico a más caro.
    15. Muestra los libros por su genero
    16. Traduce todas las etiquetas del XML y guardarlo en un fichero llamado libros.xml
        - Catalog → Catalogo
        - Book → Libro
        - Title → Título
        - Genre → Genero
        - Price → Precio
        - Public_date → Fecha de publicación
        - Description → Descripción

PRACTICA COMPLETA TEMA 1

- Realiza la siguiente practica, puedes preguntarme todas las dudas que tengas.
- Crea un programa que genere la siguiente estructura de directorios:
    - Directorio ejercicios
        - Crea el fichero pizzas.xml y rellénalo con diferentes pizzas junto con sus ingredientes
        - Lee el fichero libros.xml y muestra únicamente los autor
    - Directorio alumnos
        - Crea los 3 alumnos, guardando los objetos en ficheros para luego poder recuperarlo.
            - 1 alumno → 1 fichero
    - Directorio profesores
        - Crea un fichero de texto con el listado de nombres de los profesores
        -

EJERCICIOS TEMA 1