

Tema 1: Lenguajes de marcas

1.- Introducción

Una de las tareas básicas de un ordenador es almacenar información para poder ser procesada posteriormente. Esta información puede ser de muchos tipos diferentes:

- Texto
- Imágenes
- Vídeos
- Música
- Etc.

1.1 Características de los datos

Entre las características de los datos se destacan sobre todo tres aspectos:

- A quién van dirigidos
- La posibilidad de reutilizarlos
- Que se puedan compartir

A quién van dirigidos los datos

- A personas: Tendrán una estructura concreta, con formatos determinados, con textos decorados de alguna manera. Aparecerán títulos, caracteres en negrita, etc.
- A programas: Los datos deben ser fácilmente identificables, debe quedar claro de qué tipo son y que haya alguna manera de determinar qué significan para poderlos tratar automáticamente.

Reutilización de los datos

Frecuentemente se querrá reutilizar los datos para poder hacer tareas diferentes. Por ejemplo, un archivo hecho con MS Office puede que queramos abrirlo más adelante con otra suite ofimática, como LibreOffice. El archivo de texto que guarda una estación meteorológica quizá queramos que se pueda abrir desde una web para hacer públicos los datos. Etc.

Es básico disponer de un sistema de almacenamiento que permita conseguir que los datos puedan ser reutilizados fácilmente. Mejor aún si pueden ser reutilizados tanto por personas como por programas.

Compartición de los datos

En el pasado, con los ordenadores centrales la información se generaba i procesaba en el mismo lugar. Pero la aparición de los ordenadores personales, las redes informáticas y, sobre todo, el éxito de internet, ha creado problemas que antes no existían: los datos generados en un lugar ahora pueden ser consumidos desde otro totalmente diferente, como:

- En sistemas operativos totalmente diferentes
- En máquinas que pueden funcionar de maneras muy diferentes

La conclusión es que es necesario almacenar los datos de manera que no haya problemas para usarlos en sistemas diferentes.

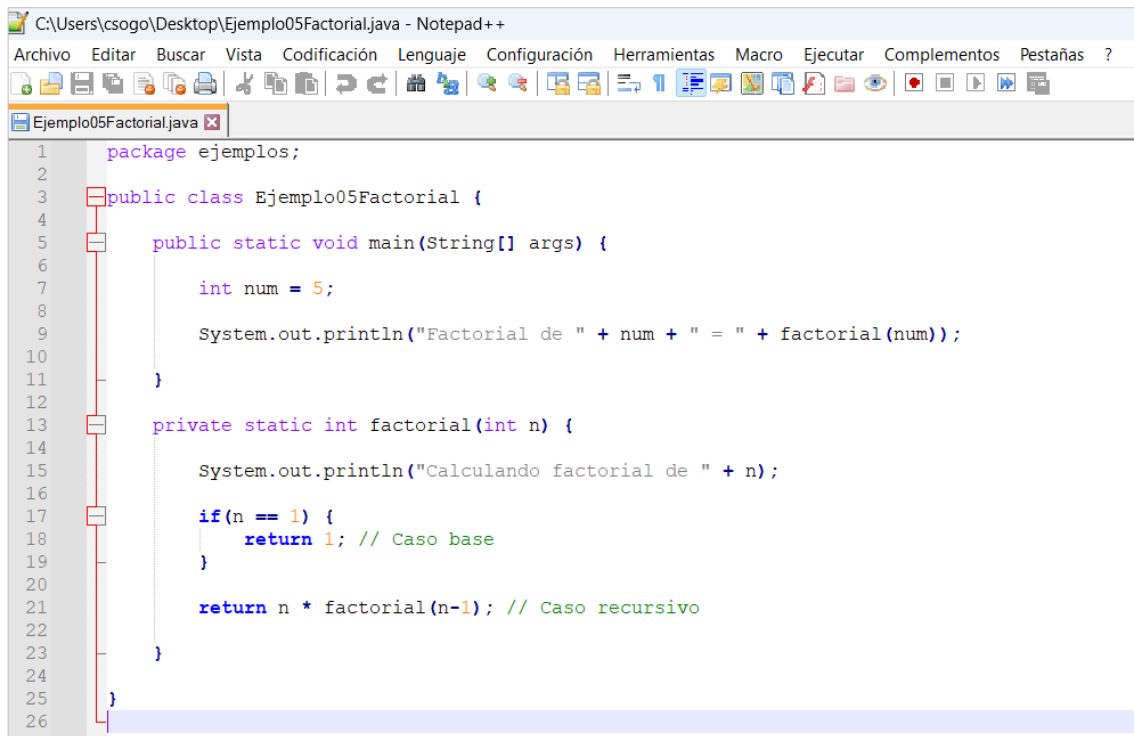
1.2 Almacenaje de datos en ordenadores

En cualquier archivo de un ordenador hay información codificada en sistema binario. Si lo abrimos veríamos una ristra de 0s y 1s. Sin embargo, según el formato en que se codifique la información en un archivo, hacemos esta distinción:

- **Archivos binarios:** son generados mediante programas y para cada tipo de archivo se necesita un programa que entienda el formato en que se ha guardado la información. Ejemplos de archivos binarios son los de imagen, música, vídeo, documentos con formatos (Word, Writer), etc. Si abrimos un archivo binario con un editor de texto sólo veremos información sin sentido, como esto:

- **Archivos de texto:** un archivo de texto es una secuencia de caracteres organizados en líneas terminadas por un carácter de nueva línea. Los archivos de texto se caracterizan por ser planos, es decir, todas las letras tienen el mismo tamaño y no hay palabras subrayadas, en negrita, o letras de distinto tamaño o ancho. Son generados mediante editores de texto plano, como Visual Studio Code, Notepad++, Bloc de notas, etc.

Cualquier editor de texto nos permitirá abrir correctamente un archivo de texto plano.



```
C:\Users\csogo\Desktop\Ejemplo05Factorial.java - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Complementos  Pestañas  ?

Ejemplo05Factorial.java
1  package ejemplos;
2
3  public class Ejemplo05Factorial {
4
5      public static void main(String[] args) {
6
7          int num = 5;
8
9          System.out.println("Factorial de " + num + " = " + factorial(num));
10     }
11
12     private static int factorial(int n) {
13
14         System.out.println("Calculando factorial de " + n);
15
16         if(n == 1) {
17             return 1; // Caso base
18         }
19
20         return n * factorial(n-1); // Caso recursivo
21     }
22
23 }
24
25
26
```

Códigos de caracteres

Representar los datos en un ordenador en forma de texto implica que para poder representar una palabra cualquiera en el ordenador, previamente tendrá que ser codificada para hacer que pueda ser representada en binario (recordamos que los ordenadores solo pueden representar datos en binario).

Esta codificación suele consistir en determinar una cantidad de bits predefinida para marcar un carácter y posteriormente se asocia un valor numérico a cada uno de los caracteres.

La equivalencia entre los caracteres y sus valores numéricos no se puede hacer de manera aleatoria, puesto que se estaría creando el mismo problema que hay con los datos binarios. Si se quiere conseguir que los datos se puedan leer en diferentes sistemas hay que seguir algún tipo de norma conocida por todo el mundo. Por este motivo aparecieron los estándares de codificación de caracteres.

Codificación ASCII (American Standard Code for Information Interchange)

Fue uno de los primeros estándares internacionales. Cada carácter de texto tiene asignado un número entre 0 y 127 (en binario, de 7 cifras). Para poder codificar caracteres no ingleses, se hicieron versiones añadiendo un bit más, con lo que se disponía de números de 0 a 255. Estos ASCII ampliados reciben nombres como ISO 8859-1, ISO 8859-2, etc.

ASCII Hex Símbolo	ASCII Hex Símbolo	ASCII Hex Símbolo	ASCII Hex Símbolo
0 0 NUL	16 10 DLE	32 20 (espacio)	48 30 0
1 1 SOH	17 11 DC1	33 21 !	49 31 1
2 2 STX	18 12 DC2	34 22 "	50 32 2
3 3 ETX	19 13 DC3	35 23 #	51 33 3
4 4 EOT	20 14 DC4	36 24 \$	52 34 4
5 5 ENQ	21 15 NAK	37 25 %	53 35 5
6 6 ACK	22 16 SYN	38 26 &	54 36 6
7 7 BEL	23 17 ETB	39 27 '	55 37 7
8 8 BS	24 18 CAN	40 28 (56 38 8
9 9 TAB	25 19 EM	41 29)	57 39 9
10 A LF	26 1A SUB	42 2A *	58 3A :
11 B VT	27 1B ESC	43 2B +	59 3B ;
12 C FF	28 1C FS	44 2C ,	60 3C <
13 D CR	29 1D GS	45 2D -	61 3D =
14 E SO	30 1E RS	46 2E .	62 3E >
15 F SI	31 1F US	47 2F /	63 3F ?

ASCII Hex Símbolo	ASCII Hex Símbolo	ASCII Hex Símbolo	ASCII Hex Símbolo
64 40 @	80 50 P	96 60 `	112 70 p
65 41 A	81 51 Q	97 61 a	113 71 q
66 42 B	82 52 R	98 62 b	114 72 r
67 43 C	83 53 S	99 63 c	115 73 s
68 44 D	84 54 T	100 64 d	116 74 t
69 45 E	85 55 U	101 65 e	117 75 u
70 46 F	86 56 V	102 66 f	118 76 v
71 47 G	87 57 W	103 67 g	119 77 w
72 48 H	88 58 X	104 68 h	120 78 x
73 49 I	89 59 Y	105 69 i	121 79 y
74 4A J	90 5A Z	106 6A j	122 7A z
75 4B K	91 5B [107 6B k	123 7B {
76 4C L	92 5C \	108 6C l	124 7C
77 4D M	93 5D]	109 6D m	125 7D }
78 4E N	94 5E ^	110 6E n	126 7E ~
79 4F O	95 5F _	111 6F o	127 7F □

Los números que van de 0 a 31 representan caracteres de control (acciones) como salto de línea (LF, 10) o tabulación (TAB, 9).

Codificación UNICODE

Unicode es un intento de sustituir los códigos de caracteres existentes por uno genérico que sirva para todas las lenguas, y por tanto supere todos los problemas de incompatibilidad que se producían en entornos multilingües y permita añadir los caracteres no latinos. La idea básica de Unicode es dar a cada uno de los símbolos un identificador único universal de forma que se puedan usar en el mismo documento idiomas diferentes sin que esto comporte problemas de representación.

Unicode define tres formas de codificación básicas UTF (Unicode Transformation Format): UTF-8, UTF-16, UTF-32.

Múltiples codificaciones

Si un archivo de texto ha sido codificado con un juego de caracteres y lo abrimos con un editor que usa otro juego, veremos que los caracteres no ingleses no se muestran correctamente:

Què és l'IOC?

dijous, 10 de febrer de 2011 09:29

La raó de ser de l'Institut Obert de Catalunya es troba plenament vinculada als objectius del Departament d'Ensenyament

En primer lloc, l'extensió de l'educació, facilitant que pugui arribar al màxim de persones superant les limitacions de l'espai i del temps.

En segon lloc, una educació que ha de posar els interessos, les necessitats i les possibilitats de progrés de les persones en el centre d'un model educatiu flexible i adaptable.

En tercer lloc, una proposta educativa que, en col·laboració amb les altres, impulsa la formació al llarg de la vida.

I, finalment, el treball compartit per presentar una oferta educativa de qualitat i amb voluntat de millora, que vagi incorporant les innovacions en les tecnologies de la informació i de la comunicació amb l'objectiu de donar un millor servei educatiu.

1.3 Lenguajes de marcas

Un lenguaje de marcado o lenguaje de marcas es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación.

El lenguaje de marcas más extendido es el HTML (HyperText Markup Language, lenguaje de marcado de hipertexto), fundamento de la World Wide Web.

Ejemplo de documento HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mi web</title>
</head>
<body>
  <h1>Mi web</h1>
  <p>Esto es el primer párrafo de mi web</p>
</body>
</html>
```

Ejemplo de documento XML:

```
<?xml version="1.0" encoding="UTF-8">

<cliente id="1002">
  <nombre>Carlos</nombre>
  <apellidos>Martínez</apellidos>
  <dni>11000222J</dni>
</cliente>
```

Ejemplo de documento JSON:

```
{
  "cliente": {
    "id": 1002,
    "nombre": "Carlos",
    "apellidos": "Martínez",
    "dni": "11000222J"
  }
}
```

Ejemplo de documento TeX:

```
\section{Persones}
\begin{itemize}
\item Manel Puig Garcia
\item Pere González Puigdevall
\item Maria Pous Canadell
\end{itemize}
```

Ejemplo de documento Wiki Markup:

```
= Persones =  
* Manel Puig Garcia  
* Pere González Puigdevall  
* Maria Pous Canadell
```

Características de los lenguajes de marcas

Las características más interesantes de los lenguajes de marcas son:

- Que se basan en texto plano (sin ningún formato).
- Que permiten usar metadatos.
- Que son fáciles de interpretar y procesar.
- Que son fáciles de crear y bastante flexibles para representar datos muy diversos.

Las aplicaciones de Internet y muchos de los programas de ordenador que se usan habitualmente utilizan de alguna manera u otra algún lenguaje de marcas.

Clasificación de los lenguajes de marcas

Podemos dividir los lenguajes de marcas en dos tipos:

- Procedimentales y de presentación: están orientados a especificar cómo se representa la información. Ejemplos de este tipo son: HTML (para representar una página web), Wiki markup (usado en la Wikipedia), Tex (para representar textos matemáticos y técnicos).
- Descriptivos o semánticos: orientados a describir la estructura de los datos que contiene. Ejemplos de este tipo son: XML, JSON, YAML.

2. El lenguaje XML

2.1 Historia de XML

A principios de los años 80 la empresa IBM necesitaba una manera de compartir información entre distintas plataformas y acabó creando el estándar **SGML** (Standard Generalized Markup Language). Este modelo permitía definir formalmente un documento electrónico, independientemente de la aplicación o el sistema que lo utilizara. Como metalenguaje que era, permitía definir diferentes estructuras mediante marcas con el fin de describir los contenidos de los documentos.

El mayor éxito de SGML fue **HTML** (HyperText Markup Language), que es una aplicación concreta de SGML. HTML fue creado por Tim Berners-Lee en 1991, y se usa para describir la visualización de una página web. A diferencia de SGML, tiene un número limitado de etiquetas, y al usarlo no es posible definir marcas nuevas.

El lenguaje SGML era complicado. Su especificación ocupaba más de 150 páginas técnicas. A menudo programas que usaban un subconjunto de SGML eran incompatibles con programas que usaban un subconjunto diferente. Como consecuencia, en el año 1998 fue lanzado **XML 1.0**, una versión potente de SGML que eliminaba muchas de sus características redundantes, demasiado complicadas de implementar, confusas o que no habían sido útiles en los veinte años anteriores.

Las siglas XML significan *Extensible Markup Language* (Lenguaje de marcas extensible).

2.2 Características del XML

Extensibilidad

XML permite que cada persona cree su propio conjunto de etiquetas. Sólo es obligado ceñirse a un pequeño número de reglas.

Estructura de los datos

Una de las ideas más importantes de XML es separar los datos de la presentación. XML se diseñó con la idea de dar estructura a los datos, y olvidarse de cómo se presentarán al usuario. Para hacerlo se desarrollaron otras tecnologías como CSS, XML-FO, etc.

Transporte de los datos

El hecho de que el XML se concentre en la estructura de los datos hace que sea fácil determinar qué tipo de datos contiene y que sea un sistema ideal para el transporte de datos entre diferentes plataformas.

Creación de lenguajes

XML permite que cada persona invente las etiquetas y atributos que necesite para almacenar su información. Esto ha hecho que hayan surgido multitud de vocabularios estandarizados que son usados en ámbitos concretos. En la tabla podemos ver algunos ejemplos:

Nombre	Uso
SVG	Gráficos vectoriales
MathML	Representar fórmulas matemáticas
CML	Intercambio de fórmulas químicas
ChessGML	Representar partidas de ajedrez

En los dos temas siguientes veremos cómo, una vez tenemos claro el formato que usaremos en nuestros documentos XML, podremos definir la estructura que deben cumplir para considerarse correctos.

Extensible

Una vez hemos creado un vocabulario, nada nos impide mezclar nuestras etiquetas con las de otros vocabularios existentes.

Desventajas de XML

Los ficheros XML tienen tendencia a ser muy grandes. Gran parte del espacio ocupado por un archivo XML la forman las propias etiquetas. Esto tiene un impacto negativo en el rendimiento de un programa, ya que para poder trabajar con él, debe descargarlo antes por la red o leerlo desde el disco.

Esta desventaja de XML ha hecho que surjan los lenguajes de marcas ligeros, cuyo objetivo es ocupar mucho menos espacio. El lenguaje de marcas ligero más usado es JSON (JavaScript Object Notation).

3. Documentos XML

3.1 Estructura de un documento XML

Un archivo de lenguaje de marcado extensible (XML) es un documento basado en texto que se puede guardar con la extensión .xml. Para crear o editar un archivo XML, se puede usar cualquier editor de texto plano: Bloc de notas, Notepad++, Gedit, VS Code, Sublime Text, etc.

Cualquier archivo XML incluye los siguientes componentes:

Declaración XML

Un documento XML comienza con alguna información sobre el propio XML. Generalmente, la versión XML (1.0) y la codificación que usará el archivo:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Elementos

Una etiqueta en XML es un nombre envuelto en los símbolo <>. Un elemento es un bloque envuelto entre una etiqueta de apertura y otra de cierre. Por ejemplo:

```
<nombre>Mario López</nombre>
```

Atributos

Podemos añadir información a un elemento dentro de su etiqueta mediante atributos. Por ejemplo, aquí añadimos al nombre un atributo "cargo":

```
<nombre cargo="director">Mario López</nombre>
```

No podemos repetir el mismo atributo dentro de la misma etiqueta. Esto sería erróneo:

```
<nombre cargo="director" cargo="accionista">Mario López</nombre>
```

Si queremos indicar que Mario tiene dos cargos lo podríamos hacer de distintas maneras:

```
<nombre cargo="director accionista">Mario López</nombre>
```

```
<nombre director="si" accionista="si">Mario López</nombre>
```

Contenido

Se llama contenido a la información que hay entre una etiqueta de apertura y la correspondiente de cierre. Por ejemplo, en el elemento "nombre" anterior el contenido sería "Mario López".

Entidades

Existen ciertos caracteres que no pueden usarse, llamados *entidades*. Son los que tienen un significado especial en XML:

Símbolo	Entidad
<	<
>	>
”	"
,	'
&	&

Las entidades empiezan siempre por "&" y terminan en ";".

Ejemplo: la siguiente línea no sería correcta:

```
<info>Para aprobar la asignatura se debe cumplir que nota > 5</info>
```

Para que lo sea sustituiremos el símbolo ">" por la entidad ">":

```
<info>Para aprobar la asignatura se debe cumplir que nota &gt; 5</info>
```

Secciones CDATA

Si queremos guardar un texto en el documento XML que contiene muchos símbolos prohibidos, en vez de sustituirlos uno a uno por sus entidades, podemos meter todo el texto entre los símbolos:

```
<![CDATA[
    ...
]]>
```

Por ejemplo, aquí tenemos un trozo de documento XML en el que hemos puesto un trozo de programa en una sección CDATA:

```
<comprobacion>
    <![CDATA[
        if (x <y and y>z){
            printf("Correcto!");
        }
    ]]>
</comprobacion>
```

Comentarios

Podemos escribir comentarios en nuestros documentos para dar alguna información extra a la persona que lea el archivo. Los programas ignoran estos comentarios.

Los comentarios se escriben entre las etiquetas `<!--` y `-->`. Por ejemplo, en el siguiente documento XML hemos añadido dos comentarios explicativos:

```
<?xml version="1.0" encoding="UTF-8" ?>

<clase>
    <!-- Lista de alumnos en la clase -->
    <alumno>Juan Antonio López</alumno>
    <alumno>María Arregui</alumno>
    <alumno>Pedro Jiménez</alumno>
    <alumno>Raúl Perales</alumno>

    <!-- Lista de profesores que dan clase en este grupo -->
    <profesor>Mario López</profesor>
    <profesor>Manuela Sánchez</profesor>

</clase>
```

3.2 Corrección de un documento XML

Un documento XML está **bien formado** (es correcto) cuando cumple estas cinco reglas:

1. Solo puede haber un elemento raíz.
2. Todas las etiquetas que se abren se tienen que cerrar.
3. Las etiquetas tienen que estar anidadas correctamente.
4. Los nombres de las etiquetas tienen que ser correctos.
5. Los valores de los atributos tienen que estar entre comillas.

Regla nº 1: Sólo puede haber un elemento raíz

El siguiente documento no está bien formado, porque no hay ningún elemento dentro del que estén **todos** los elementos del documento:

```
<?xml version="1.0" encoding="UTF-8" ?>

<clientes>
  <nombre>Juan García</nombre>
  <nombre>Manuel Tendero</nombre>
</clientes>

<empleados>
  <empleado>Ana Rico</empleado>
  <empleado>Lucía Sánchez</empleado>
</empleados>
```

En cambio, éste sí es correcto, ya que hemos añadido un elemento *personal* que agrupa a todos los demás:

```
<?xml version="1.0" encoding="UTF-8" ?>

<personal>
  <clientes>
    <nombre>Juan García</nombre>
    <nombre>Manuel Tendero</nombre>
  </clientes>

  <empleados>
    <empleado>Ana Rico</empleado>
    <empleado>Lucía Sánchez</empleado>
  </empleados>
</personal>
```

Regla nº 2: Todas las etiquetas que se abren se tienen que cerrar

Si dejamos una etiqueta sin su correspondiente etiqueta de cierre (con la barra, /), el documento no estará bien formado. Por ejemplo, aquí vemos un documento incorrecto porque no se ha cerrado la etiqueta *clientes*:

```
<?xml version="1.0" encoding="UTF-8" ?>

<personal>
  <clientes>
    <nombre>Juan García</nombre>
    <nombre>Manuel Tintero</nombre>

    <empleados>
      <empleado>Ana Rico</empleado>
      <empleado>Lucía Sánchez</empleado>
    </empleados>
  </personal>
```

Podemos tener una etiqueta **vacía** (sin contenido):

```
<proveedores></proveedores>
```

Y en tal caso normalmente la escribiremos una sola vez con la barra de cierre al final:

```
<proveedores />
```

Regla nº 3: Las etiquetas tienen que estar anidadas correctamente

Cada vez que aparezca una etiqueta de cierre, tiene que corresponder a la última etiqueta de apertura. Por ejemplo, este trozo no es válido porque antes que <empleados> debería aparecer </clientes>:

```
<clientes>
  <nombre>Juan García</nombre>
  <nombre>Manuel Tintero</nombre>
  <empleados>
</clientes>
  <empleado>Ana Rico</empleado>
  <empleado>Lucía Sánchez</empleado>
</empleados>
```

Si escribimos nuestros XML correctamente sangrados será más fácil evitar estos fallos.

Regla nº 4: Los nombres de las etiquetas tienen que ser correctos

Los nombres que usemos para etiquetas y atributos deben cumplir lo siguiente:

- Deben empezar por una letra o guión bajo (_).
- Pueden usarse caracteres alfanuméricos (letras y números), además de guión (-), guión bajo (_) y punto (.).
- No puede haber espacios.
- Las mayúsculas y minúsculas se consideran diferentes.
- No se pueden usar nombres que empiecen por "xml" (están reservados).

Regla nº 5: Los valores de los atributos tienen que estar entre comillas

Los valores de los atributos deben ir entre comillas dobles o simples. Por ejemplo:

```
<nombre edad="30">Juan García</nombre>
```

```
<nombre edad='30'>Juan García</nombre>
```

En cambio esto no sería correcto:

```
<nombre edad=30>Juan García</nombre>
```

3.3 Estructura en árbol de un documento XML

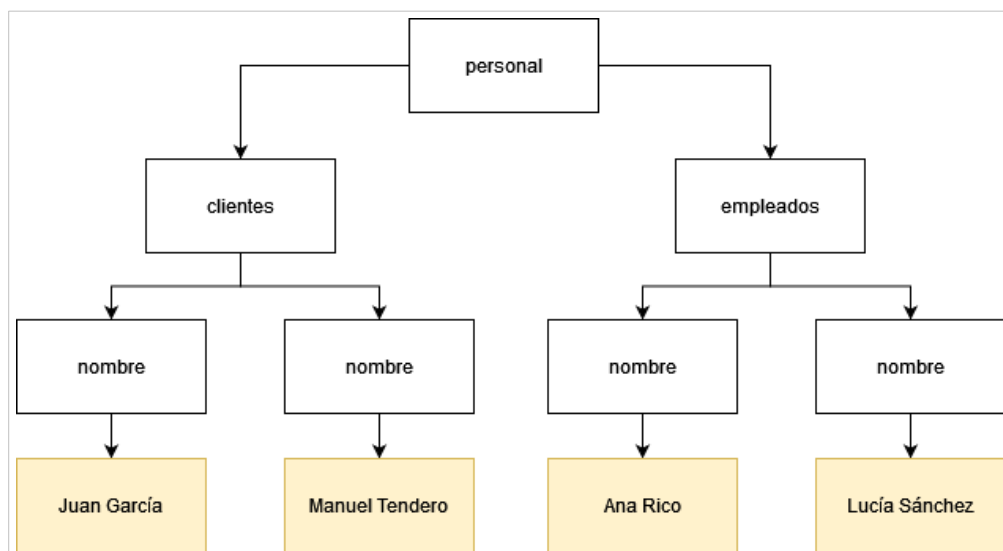
Un documento XML siempre puede representarse como un árbol donde los elementos van formando ramas.

Por ejemplo, dado el siguiente XML:

```
<personal>
  <clientes>
    <nombre>Juan García</nombre>
    <nombre>Manuel Tintero</nombre>
  </clientes>

  <empleados>
    <nombre>Ana Rico</empleado>
    <nombre>Lucía Sánchez</empleado>
  </empleados>
</personal>
```

Podemos representarlo en forma de árbol así:



Los rectángulos se llaman nodos. Si un rectángulo desciende de otro, decimos que es su nodo hijo. Si desciende de algún nodo hijo, decimos que es su descendiente.

3.4 Procesadores XML

Se llama *procesadores XML* a los programas encargados de detectar si un documento XML está bien formado. A menudo se usa la palabra inglesa *parsers*.

Navegadores web

Todos los navegadores incluyen un parser de manera que si abrimos un archivo xml nos mostrarán su contenido y nos dirán si hay algún error en el mismo (es decir, si está bien formado o no).

Librería libxml2

Existen varios procesadores independientes de XML, pero en este curso usaremos una biblioteca de código para analizar XML llamada **libxml2**.

libxml es un procesador open source escrito en lenguaje C. Puede ser usado desde muchos otros lenguajes de programación (Perl, Ruby, Python, C#, PHP, etc.) y sistemas operativos. Entre las utilidades que incluye está la aplicación **xmllint**, que nos permitirá analizar y validar documentos XML.

Si queremos analizar un archivo, sólo tenemos que abrir una terminal en la carpeta donde está ubicado y ejecutar la orden:

```
xmllint archivo.xml
```

Si el archivo es correcto, *xmllint* nos lo volverá a mostrar. Si no es correcto, en cambio, nos mostrará una descripción del error y la línea donde cree que está.

Si no queremos que *xmllint* nos devuelva el contenido del archivo cuando éste es correcto, podemos usar el argumento `--noout`:

```
xmllint archivo.xml --noout
```

Instalación de la librería libxml2 en Linux

Basta con ejecutar la orden *apt install libxml2-utils*.

Instalación de la librería libxml2 en Windows

En el Moodle puedes ver un videotutorial con todos los pasos.

3.5 Editores XML

Como lenguaje de marcas que és, podríamos escribir archivos XML incluso con el Bloc de notas de Windows. Sin embargo, si usamos un editor específico, podremos usar varias herramientas y ayudas.

Algunos editores comerciales para XML conocidos son: *oXygen XML Editor*, *Altova XMLSpy XML editor*, *Liquid XML Studio*, etc. En nuestra asignatura usaremos Eclipse con el plugin "Eclipse XML Editors and tools".

3.6 Espacios de nombres

Los espacios de nombres nos permiten crear grupos de elementos y atributos. Esto nos permite evitar que en un documento XML complejo se mezclen etiquetas con el mismo nombre pero que realmente significan cosas distintas.

Por ejemplo, en este XML hay dos elementos `<id>`, pero no representan lo mismo:

```
<pedido>

  <cliente>
    <id>DNI</id>
    <id_valor>10200200R</id_valor>
    <nombre>Lucía Ortiz</nombre>
  </cliente>

  <articulo>
    <id>205</id>
    <descripcion>Libreta Oxford A4</descripcion>
    <precio>4.50</precio>
  </articulo>

  <articulo>
    <id>401</id>
    <descripcion>Bolígrafo Pilot Extra</descripcion>
    <precio>2.15</precio>
  </articulo>

</pedido>
```

La etiqueta `<id>` de cliente representa el tipo de documento que usa para identificarse (DNI o Pasaporte, por ejemplo). En cambio, la etiqueta `<id>` de artículo se refiere a un número identificativo que se le da a cada artículo.

Una solución para este problema es poner un prefijo a las etiquetas de cliente (y así forman parte todas de un espacio de nombres) y otro a las etiquetas para artículos (y así pertenecerán a otro espacio de nombres diferentes).

```
<c:pedido>

  <c:cliente>
    <c:id>DNI</c:id>
    <c:id_valor>10200200R</c:id_valor>
    <c:nombre>Lucía Ortiz</c:nombre>
  </c:cliente>

  <t:articulo>
    <t:id>205</t:id>
    <t:descripcion>Libreta Oxford A4</t:descripcion>
    <t:precio>4.50</t:precio>
  </t:articulo>

  <t:articulo>
    <t:id>401</t:id>
    <t:descripcion>Bolígrafo Pilot Extra</t:descripcion>
    <t:precio>2.15</t:precio>
  </t:articulo>

</c:pedido>
```

No pasa nada porque haya dos elementos con nombre id, ya que uno es c:id y el otro es t:id.

Ahora estamos usando elementos de dos espacios de nombres, y debemos declarar estos espacios al principio). El estándar XML exige que sus nombres sean URI (identificadores de recursos universales), y la forma de hacer esto es juntar la URL de la empresa y añadir algo al final que los diferencie. Suponiendo que nuestra empresa se llama "Papelería Plus":

```
xmlns:c="http:www.papeleriaplus.com/clientes"
xmlns:t="http:www.papeleriaplus.com/articulos"
```

Estas declaraciones se ponen en el elemento raíz de la clase (o donde se necesite empezar a usar cada espacio de nombres).

```
<c:pedido xmlns:c="http://www.papeleriaplus.com/clientes"
          xmlns:t="http://www.papeleriaplus.com/articulos">

  <c:cliente>
    <c:id>DNI</c:id>
    <c:id_valor>10200200R</c:id_valor>
    <c:nombre>Lucía Ortiz</c:nombre>
  </c:cliente>

  <t:articulo>
    <t:id>205</t:id>
    <t:descripcion>Libreta Oxford A4</t:descripcion>
    <t:precio>4.50</t:precio>
  </t:articulo>

  <t:articulo>
    <t:id>401</t:id>
    <t:descripcion>Bolígrafo Pilot Extra</t:descripcion>
    <t:precio>2.15</t:precio>
  </t:articulo>

</c:pedido>
```

También podemos establecer un espacio de nombres por defecto, de manera que las etiquetas que pertenezcan al mismo no necesitarán prefijo. Por ejemplo, si decidimos que el espacio de nombres de las etiquetas de cliente sea el espacio por defecto, el documento quedaría así:

```
<pedido xmlns="http://www.papeleriaplus.com/clientes"
        xmlns:t="http://www.papeleriaplus.com/articulos">

  <cliente>
    <id>DNI</id>
    <id_valor>10200200R</id_valor>
    <nombre>Lucía Ortiz</nombre>
  </cliente>

  <t:articulo>
    <t:id>205</t:id>
    <t:descripcion>Libreta Oxford A4</t:descripcion>
    <t:precio>4.50</t:precio>
  </t:articulo>

  <t:articulo>
    <t:id>401</t:id>
    <t:descripcion>Bolígrafo Pilot Extra</t:descripcion>
    <t:precio>2.15</t:precio>
  </t:articulo>

</pedido>
```