



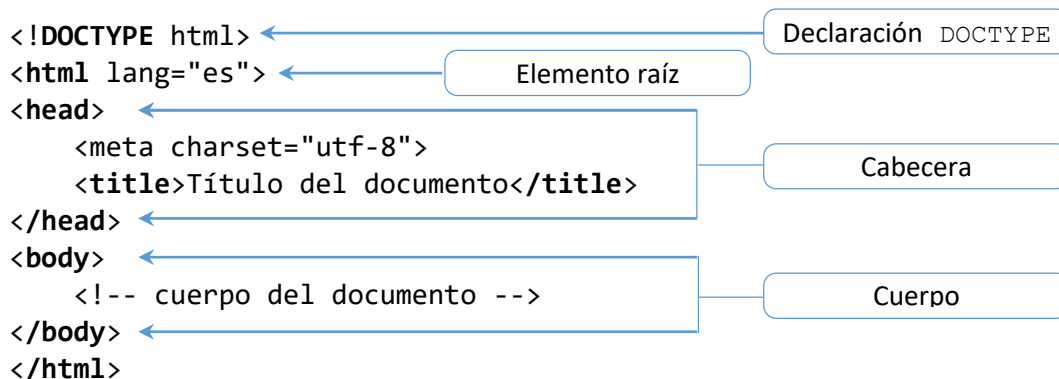
1 Sesión 2. HTML.

En esta práctica se van a escribir documentos (X)HTML5 válidos, introduciendo los elementos de uso más habitual e ignorando detalles de presentación de la información. La presentación de los documentos en un cliente Web (navegador) será la que establezca éste por defecto.

1.1 HTML5 y XHTML5

Un documento (X)HTML5 permite describir la estructura y la información de un documento. No está basado en SGML, como las versiones anteriores de HTML, y por eso la declaración `DOCTYPE` es más corta.

La estructura global de un documento HTML 5 es la siguiente:



donde todos los elementos que figuran en **negrita** son obligatorios. El cuerpo del documento contiene la información contenida en éste y que se mostrará en el cliente. La cabecera del documento, además del título de éste, contiene otra información que aporta o describe ciertas características del mismo (meta-información) como, por ejemplo, la codificación de caracteres utilizado para escribirlo, la cual se indica en el elemento `meta`.

Los documentos XHTML(5) tienen una sintaxis más estricta ya que también tienen que ser documentos XML bien formados. Mismamente la estructura global de un documento XHTML incluye otras partes y atributos:

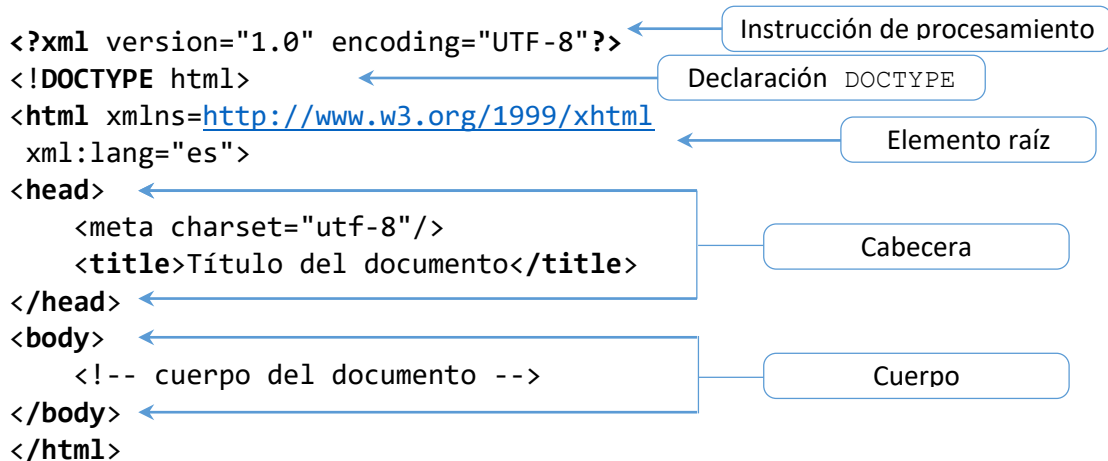
1. Una instrucción de procesamiento, donde se indica la versión XML y la codificación de caracteres utilizada en el documento. Si no se proporciona esta instrucción se asume que la codificación de caracteres es UTF-8.
2. El elemento raíz incluye un espacio de nombres (atributo `xmlns`). Este espacio de nombres establece que todos los elementos y atributos de XHTML deben escribirse en minúsculas. Y para indicar el idioma se utiliza el atributo cualificado `xml:lang`.

3. El elemento `meta` no permite establecer la codificación de caracteres, y de incluirse éste, se ignora.

Además, para obtener un documento XML bien formado es necesario:

1. Que todos los elementos tengan etiqueta de inicio y etiqueta de fin, incluidos los elementos que no tienen contenido.
2. Que todos los elementos estén anidados, no solapados.
3. Que todos los atributos tengan valores entrecomillados (con comillas simples o dobles), incluidos los atributos indicativos.

La estructura global de un documento XHTML(5) es la siguiente:



La ventaja de (X)HTML5 es que ambas estructuras globales (HTML5 y XHTML5) se pueden combinar, prescindiendo de lo que no está permitido en HTML: *las instrucciones de procesamiento*. De esta forma, se puede escribir un documento único que el servidor podrá servir como HTML (`text/html`) o como XHTML (`application/xhtml+xml`) en función de la negociación que establezca con el cliente.

Por tanto, escribiremos documentos de acuerdo a la sintaxis más estricta requerida por XHTML y como éstos carecerán de instrucciones de procesamiento, sólo podrán escribirse utilizando la codificación de caracteres UTF-8 (codificación por defecto en XML). Además, el elemento raíz del documento (`html`) incluirá los tres atributos resultantes (`lang`, `xmlns` y `xml:lang`).

El archivo comprimido que se proporciona para esta práctica, `practica-02.zip`, contiene un documento (X)HTML5 completo: el archivo `noSoportado.html`. Prueba a validar el documento en: <https://validator.nu/>

Primero utiliza el *parser* HTML5 y luego el *parser* XML (`don't load external entities`) para comprobar que es un documento (X)HTML válido. Añade ahora como primera línea la instrucción de procesamiento:

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


Y repite la validación, no habrá problema alguno con el *parser* XML, pero el *parser* HTML5 dará un error en la línea añadida.

1.2 Un pequeño sitio Web

Escribe la página principal, `index.html`, de un sitio sobre tipos de datos. El documento contendrá los siguientes elementos:

1. Una cabecera del cuerpo del documento (`header`) con logo (`img`), título principal (`h1`) y subtítulo (`h2`).
2. Una sección de navegación (`nav`) con una lista de hipervínculos (`ul`).
3. Una sección principal (`section`) con encabezados (`hnúmero`) y que contendrá varios párrafos (`p`), el documento `noSoportado.html` incrustado (`iframe`), una tabla (`table`) y una imagen (`img`).
4. Todos los elementos característicos de la sección principal (tabla, imágenes, código incrustado) estarán contenidos en un elemento `figure` y un pie descriptivo establecido con `figcaption`.

Aunque el documento se realizará ignorando la información de presentación, el aspecto final del sitio (con la información de presentación incluida) sería similar al que se muestra en la imagen siguiente (también disponible en PDF, `index.pdf`):



Tipos de Datos

Números racionales

[Página inicial](#) [TAD Racional](#) [AbstractRacional](#) [RacionalNoMod](#) [RacionalMod](#) [Documentación](#) [Descargar](#)

TAD Racional

El tipo abstracto de dato (TAD) `Racional` se proporciona mediante una [interfaz Java](#), especificando cada una de sus operaciones por medio de Javadoc. La documentación se ha generado para el ámbito `package`; es decir, desde el punto de vista del desarrollador.

Por analogía con los tipos numéricos predefinidos en Java, un tipo de dato para números racionales seguramente debería ser inmutable, en cuyo caso, el TAD `Racional` no debería incluir operaciones básicas modificadoras. Sin embargo, se ha preferido incluir una operación de este tipo, `acumula`, para que sea posible implementar tipos con distintas características: *inmutables* y *mutables*.

```
default void acumula (Racional r) {  
    throw new UnsupportedOperationException();  
}
```

Código 1. Implementación de una operación no soportada

Para que la misma interfaz se pueda utilizar para realizar implementaciones de tipos de datos mutables y de tipos de datos inmutables, todas las operaciones modificadoras se implementan, por defecto, como operaciones no soportadas, tal como se indica en el cuadro adjunto.

La interfaz incluye las operaciones siguientes:

Operación	Descripción	Coste temporal
<code>default void acumula (int dato)</code>	Modifica este racional sumándole el entero especificado (opcional)	Constante
<code>default void acumula(Racional r)</code>	Modifica este racional sumándole el racional especificado (opcional)	
<code>int denominador ()</code>	El denominador de este racional	
<code>int numerador ()</code>	El numerador de este racional	
<code>Racional reduce ()</code>	Un nuevo racional equivalente a éste e irreducible	
<code>Racional suma(Racional r)</code>	Retorna el racional suma de este racional y el racional especificado	

Tabla 1. Operaciones de la interfaz

Tipo `RacionalNoMod`

La primera implementación del TAD `Racional` es un tipo de dato no modificable (o inmutable) dado por la clase `RacionalNoMod`.

Para almacenar la información de los valores racionales del tipo de dato, se utiliza un campo estático, `array datos`. Los racionales se almacenan en posiciones consecutivas del `array`, por lo que cada componente de éste es otro `array` de dos enteros, uno para el numerador y otro para el denominador de un racional. El número de racionales guardados en el `array datos` se mantiene el campo estático `numRacionales`.

Para representar un racional se utiliza un campo de tipo entero, `rep`. El valor de este campo es el número de componente del `array datos` en el que se almacena la información (numerador y denominador de éste).

```
Racional r = new RacionalNoMod(1, 2); // R(r) = 1/2
```

código cliente

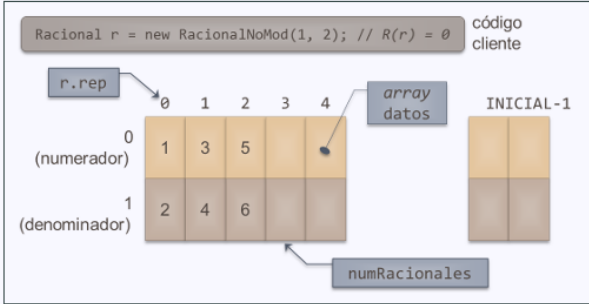


Fig. 1. Almacenamiento para el tipo de dato `RacionalNoMod`

El resto de páginas del sitio, a las que se accede desde el menú de navegación, tendrán la misma cabecera del cuerpo del documento y barra de navegación que la página principal. Además, para cada una de las referencias URL de la barra de navegación, exceptuando la opción `Descargar`, el recurso recuperado mostrará un documento XHTML o HTML incrustado correspondiente al código Java formateado de la interfaz o clase seleccionada, o bien, de la documentación Javadoc del *package* correspondiente. Si la opción seleccionada es la interfaz o una clase, el documento incrustado irá precedido de un encabezado indicando el tipo de código, interfaz o clase, y su nombre, tal y como se muestra en la imagen siguiente:

Tipos de Datos
Números racionales

[Página inicial](#) [TAD Racional](#) [AbstractRacional](#) [RacionalNoMod](#) [RacionalMod](#) [Documentación](#) [Descargar](#)

Clase RacionalNoMod

```

1 package estDatos;
2
3 /**
4  * Implementación de la interfaz {@code Racional}. El tipo de
5  * dato es no modificable o inmutable.
6  *
7  * @author Pedro Hernández
8  */
9
10 public class RacionalNoMod extends AbstractRacional {

```

Por último, la opción `Descargar` del menú de navegación permitirá descargar todo el código fuente. Éste está disponible en el archivo comprimido `Racionales.zip`.

En el archivo comprimido proporcionado para la realización de la práctica, están disponibles todos los documentos de código formateado y la documentación Javadoc del *package*. Supuesto que las páginas del sitio se guardan en la misma carpeta que los recursos proporcionados para la práctica, para recuperar éstos deberás utilizar las referencias URL relativas que se indican en la tabla siguiente:

Recurso	Referencia URL
Interfaz Racional	Racional.xhtml
Clase AbstractRacional	AbstractRacional.xhtml
Clase RacionalNoMod	RacionalNoMod.xhtml
Clase RacionalMod	RacionalMod.xhtml
Documentación Javadoc	doc/index.html
Archivos de código Java	Racionales.zip

NOTA: Es necesario validar todos los documentos XHTML5 realizados mediante el recurso en línea citado en el apartado previo (<https://validator.nu/>). Un documento se considerará correcto si no se produce error o aviso alguno al utilizar tanto el parser HTML5 como el parser XML.