A. Investiga sobre las clases de DOM para el tratamiento de ficheros XML.

1. ¿Para qué se utiliza DOM en Java?

En Java, **DOM** es una interfaz de programación de aplicaciones (API) estándar que se utiliza para **representar documentos XML en memoria como un árbol de nodos**. Sus funciones principales son:

- **Lectura y navegación:** Cargar un documento XML completo en la memoria RAM y recorrerlo jerárquicamente (padres, hijos, hermanos) para consultar su estructura y datos.
- **Modificación:** Añadir, eliminar o modificar etiquetas, atributos y contenidos de forma dinámica mientras el documento reside en memoria.
- **Escritura (Persistencia):** Volcar de nuevo el árbol de nodos modificado desde la memoria a un fichero físico en disco o a un flujo de datos (`OutputStream`).

> **Nota:** Al cargar todo el documento en memoria, es ideal para ficheros pequeños o medianos, ya que ofrece un acceso muy flexible (a diferencia de otros enfoques secuenciales como SAX).

2. ¿En qué otros ámbitos, además de en programación con JAVA, se utiliza el enfoque DOM?

El concepto y enfoque DOM no es exclusivo de Java ni del ecosistema XML; se utiliza ampliamente en:

- **Desarrollo Web (JavaScript / Navegadores):** Es el ámbito más conocido. El navegador web crea el *HTML DOM* a partir de una página web, permitiendo a JavaScript manipular elementos visuales, estilos y eventos en tiempo real.
- **Otros lenguajes de programación:** Prácticamente todos los lenguajes modernos (Python con `minidom`/`lxml`, C#/.NET con `XmlDocument`, C++, PHP, etc.) implementan parsers basados en DOM para procesar XML y HTML.
- **Herramientas de interoperabilidad y APIs:** En sistemas donde se intercambian estructuras de datos jerárquicas y es necesario validarlas o transformarlas mediante lenguajes como XPath o XSLT.

3. Lectura de ficheros XML con DOM

¿Cuáles son las clases principales utilizadas para leer y cargar un fichero XML usando DOM?

Las clases e interfaces fundamentales dentro del paquete `javax.xml.parsers` y `org.w3c.dom` son:

1. **`DocumentBuilderFactory`**: Factoría abstracta encargada de crear instancias de `DocumentBuilder`. Permite configurar aspectos del análisis (como ignorar espacios en blanco o validar esquemas).
2. **`DocumentBuilder`**: La clase que realmente realiza el análisis (*parsing*) del fichero XML físico y genera la estructura en memoria.
3. **`Document`**: Representa el documento XML completo en la memoria. Actúa como la raíz del árbol y punto de entrada para buscar elementos.

¿Qué clases se usan para acceder a los nodos asociados a las etiquetas del fichero XML? ¿Y para acceder a la información que tienen?

- **Para acceder a los nodos (etiquetas):**
  - **`Element`**: Representa una etiqueta XML concreta.
  - **`NodeList`**: Colección de nodos (útil cuando se usa `getElementsByTagName()`).
  - **`Node`**: Interfaz genérica de la que heredan elementos, textos y atributos.
  - *Métodos clave:* `getDocumentElement()`, `getElementsByTagName()`, `getChildNodes()`, `getNextSibling()`.
- **Para acceder a la información (contenido y atributos):**
  - **`Text`** o directamente el método **`getTextContent()`** / **`getNodeValue()`** de la interfaz `Node` para extraer el texto interior de una etiqueta.
  - **`Attr`**: Clase que representa los atributos de una etiqueta.
  - *Métodos clave:* `getAttribute(nombre)`, `getAttributeNode()`.

4. Escritura de ficheros XML con DOM

¿Cuáles son las clases principales utilizadas para escribir un fichero XML?

Para serializar (volcar a disco) el árbol DOM que tenemos en memoria, se utilizan clases del paquete `javax.xml.transform`:

1. **`TransformerFactory`**: Factoría encargada de instanciar un `Transformer`.
2. **`Transformer`**: Objeto que realiza la transformación o volcado de los datos. Se encarga de convertir el árbol de memoria en un flujo de salida (fichero físico).
3. **`DOMSource`**: Actúa como origen de datos, envolviendo el objeto `Document` de Java que contiene el árbol DOM.
4. **`StreamResult`**: Actúa como el destino de la transformación (puede ser un `File`, un `OutputStream`, etc.).

¿Qué clases se usan para crear una etiqueta, el contenido asociado a una etiqueta o el contenido asociado a un atributo?

Todo el proceso de creación de la estructura se realiza invocando métodos directamente sobre la instancia de la interfaz **`Document`**:

- **Para crear una etiqueta (Elemento):** Se utiliza el método `document.createElement("nombreEtiqueta")`, que devuelve un objeto de tipo **`Element`**.
- **Para crear el contenido asociado a una etiqueta (Texto):** Se puede usar directamente el método `elemento.setTextContent("texto")` o crear un nodo de texto con `document.createTextNode("texto")` y luego añadirlo como hijo con `appendChild()`.
- **Para crear el contenido asociado a un atributo:** Se puede utilizar directamente el método `elemento.setAttribute("nombreAtributo", "valor")` o bien crear el atributo de forma explícita con `document.createAttribute("nombreAtributo")` (devolviendo un **`Attr`**), asignarle un valor con `.setValue("valor")` y vincularlo al elemento mediante `elemento.setAttributeNode(attr)`.