Título: Lectura de modelos de coches filtrados por marca y ordenados por cilindrada con DOM

Descripción:

Este ejercicio implementa un programa en Java que permite al usuario introducir una **marca de coche** y muestra los **modelos de esa marca junto con su cilindrada**, leyendo un fichero XML (`coches.xml`) mediante **DOM (Document Object Model)**. Los resultados se muestran **ordenados de mayor a menor cilindrada**. Si el fichero no existe, se crea automáticamente con coches de ejemplo.

Funcionamiento paso a paso:

1. **Comprobación de fichero:**
    - El programa verifica si existe `coches.xml`.
    - Si no existe, se crea automáticamente con varios coches de ejemplo, incluyendo Seat, Renault, BMW y Ford, cada uno con su cilindrada.
2. **Entrada del usuario:**
    - Se solicita al usuario que introduzca la marca de coche a buscar.
3. **Carga y parseo del XML:**
    - Se utiliza `DocumentBuilderFactory` y `DocumentBuilder` para parsear el fichero XML y obtener un objeto `Document`.
    - Se normaliza el documento con `normalize()` para asegurar la consistencia de la estructura.
4. **Filtrado por marca:**
    - Se obtiene un `NodeList` con todos los elementos `<coche>`.
    - Se recorren los nodos y se seleccionan aquellos cuya marca coincide con la introducida por el usuario.
    - Se extraen el **modelo** y la **cilindrada** de cada coche coincidente.
5. **Ordenamiento descendente:**
    - Se almacena la información en objetos `Coche` (modelo + cilindrada).
    - Se ordena la lista de resultados de manera descendente por cilindrada usando `Comparator`.

6. **Salida de resultados:**
    - Si no se encuentran coches de la marca indicada, se muestra un mensaje de advertencia.
    - Si se encuentran, se muestran en consola los modelos de la marca y su cilindrada, ordenados de mayor a menor cilindrada.

Gestión de errores:

- Se controlan excepciones de parsing y escritura (`ParserConfigurationException`, `Exception`).
- La creación automática del fichero XML garantiza que siempre haya datos válidos para procesar.
- Se controla la conversión de cilindrada a entero (`Integer.parseInt`) para evitar errores de formato.

Aspectos técnicos:

- DOM carga todo el XML en memoria, permitiendo acceso directo a los nodos `<coche>` y sus subelementos.
- La creación del fichero XML se realiza mediante `Transformer`, con formato indentado y legible.
- El filtrado por marca se realiza ignorando mayúsculas/minúsculas (`equalsIgnoreCase`).
- La lista de coches se ordena usando `Comparator.comparingInt(...).reversed()` para obtener el orden descendente por cilindrada.