

## TAREA 8.

# APLICACIÓN DE LAS ESTRUCTURAS DE ALMACENAMIENTO.

### ENUNCIADO.

Imagina que te proporcionan, en el siguiente formato, los datos de un cliente, con todos sus teléfonos y todos sus direcciones de correo electrónico:

DNI, "nombre", "apellidos", teléfono 1, teléfono 2, email 1, teléfono 3, email 2,...

En una misma línea se encuentran, separados por comas, todos los datos del cliente: DNI (o NIE), nombre, apellidos, teléfonos y direcciones de correo electrónico. Fíjate que los teléfonos y los correos electrónicos pueden aparecer desordenados, y que pueden ser más de uno. La idea es meter dichos datos en un documento XML que contenga los datos de contacto del cliente. **No es necesario leer los datos de un archivo, basta con que se capturen del teclado. Además, el documento XML solo contendrá los datos de un cliente, por lo que no debes preocuparte de procesar múltiples líneas (solo una).**

A la hora de procesar los datos tienes que tener en cuenta los siguientes aspectos:

- El DNI o NIE siempre aparecerá en primer lugar.
- El nombre siempre aparecerá en segundo lugar, entre comillas. Las comillas habrá que quitarlas, no pueden aparecer en el documento XML.
- Los apellidos siempre aparecerán en tercer lugar, entre comillas. Nuevamente, las comillas habrá que quitarlas.
- Los teléfonos y los correos electrónicos aparecerán desordenados y mezclados entre sí, pero en el XML resultante deben aparecer separados en dos partes: teléfonos y correos electrónicos por separado.
- Los teléfonos pueden tener paréntesis para simbolizar el prefijo ("(91)2345678") o en el código del país. Estos paréntesis habrá que eliminarlos al almacenarlo en el documento XML.
- Los teléfonos además pueden tener el símbolo "+" delante. Este hay que dejarlo, dado que simboliza que el número es un número internacional.
- Los teléfonos deben aparecer ordenados. Primero aparecerán ordenados de mayor a menor los números locales (sin símbolo "+" delante) y después los internacionales, también ordenados de mayor a menor.
- Los correos electrónicos deben almacenarse en minúsculas, aunque en la entrada vayan en mayúsculas.
- La lista de teléfonos y de correos no debería tener duplicados (tomando mayúsculas y minúsculas como lo mismo). Ten en cuenta que los paréntesis sobre un mismo número de teléfono no deben provocar la existencia de teléfonos diferentes en el XML.
- Es importante eliminar espacios sobrantes antes y después de cada trozo de texto extraído para procesarlo adecuadamente.

Las características del documento XML a generar son las siguientes:

- El elemento raíz se llamará "datos\_cliente".
- El DNI o el NIE se almacenará en un elemento llamado "id".
- El nombre siempre se almacenará en un elemento llamado "nombre".
- Los apellidos siempre se almacenarán en un elemento llamado "apellidos".
- Los teléfonos se almacenarán todos en un elemento llamado "telefonos" (sin acento). Dentro de dicho elemento, habrá sub-elementos, llamados "telefono" (sin acento), cada uno de los cuales contendrá un único teléfono. Los teléfonos deberán aparecer ordenados tal y como se comentó en el párrafo anterior.
- La etiqueta "telefonos" deberá tener un atributo llamado "total" que contendrá el número total de teléfonos de la lista.
- Los correos electrónicos se almacenarán en un elemento llamado "mails". Dentro de dicho elemento, habrá sub-elementos llamados "mail" destinados a almacenar cada uno de los correos electrónicos del usuario.

Puedes usar los siguientes datos de prueba en tu aplicación (cada línea debería generar un documento XML):

X12345678F,"nombre","apellidos",+(82)12345678, 612345678,test@TEST.com,(91)23456789  
,prueba@prueba.com

12345678Z,"nombre","apellidos", prueba@prueba.com,(952)333333,test@test.com ,952333333,test@TEST.com

No olvides ser descriptivo o descriptiva en los errores. Si la línea proporcionada contiene errores, el programa no debería quedarse bloqueado, sino que debería informar de que se ha producido un error, e intentar ser descriptivo en el error detectado.

## Criterios de puntuación.

La tarea tiene una puntuación total de 10 puntos repartidos de la siguiente forma:

- Si el programa es capaz de procesar los datos de entrada y almacenarlos en un documento XML, separando la información de la forma descrita en el enunciado: 5 puntos.
- Si el programa además elimina caracteres sobrantes (espacios, comillas y paréntesis) y convierte las direcciones de correo a minúsculas: 1 punto.
- Si el programa además elimina duplicados de teléfonos y correos electrónicos: 1 punto.
- Si el programa además ordena los teléfonos de la forma descrita en el enunciado: 1,5 puntos.
- Si el programa además muestra los errores detectados en la cadena de entrada con un alto nivel de detalle (indicando en que lugar estuvo el error encontrado, así como duplicidades existentes): 1,5 puntos.

Se valorará positivamente que el resultado se guarde en un archivo XML de disco preguntando al usuario dónde almacenarlo, aunque no es obligatorio.

## Recursos necesarios para realizar la Tarea.

Un ordenador con NetBeans y Java 6 o superior instalado.

## Consejos y recomendaciones.

Te recomendamos, que para llegar a realizar con éxito la tarea, sigas las siguientes indicaciones:

1. Estudia con detenimiento el ejemplo proporcionado en los casos prácticos de los contenidos. El ejemplo proporcionado es algo más complicado que lo que tienes que realizar, con lo que te servirá de gran ayuda.
2. Divide la cadena de entrada usando como separador la coma a través de la función `split`.
3. Crea expresiones regulares para detectar cuando se trata del DNI o NIE (esta la puedes encontrar en los contenidos), también para el nombre o los apellidos entre comillas (recuerda que el uso de los caracteres de escape), y también para detectar teléfonos y correos electrónicos. No busques expresiones regulares complicadas, sino aquellas que de forma básica detecten de que se trata, en especial para el correo electrónico, cuya expresión regular puede tener varios niveles de complejidad.
4. Es mejor empezar comprobando si el primer campo es el DNI o NIE, si el segundo y tercero son los nombres y apellidos, y después, mirar con que expresión regular encajan el resto de los campos (si con teléfono o con correo electrónico).
5. Usa listas para disponer de una capacidad de crecimiento dinámico, es lo más adecuado para almacenar y ordenar la lista de teléfonos y correos electrónicos.
6. Usa la interfaz `Comparator` para crear un comparador adecuado. Una opción para comparar los teléfonos es realizar conversión a entero en determinadas situaciones. En dicho caso, no uses el tipo `Integer`, es mejor usar el tipo `Long`, dado que los teléfonos pueden ser números mayores que lo que el tipo `Integer` soporta.
7. Para ahorrarte trabajo, puedes poner los errores detectados como comentarios en el documento XML.
8. Encontrar un teléfono o un correo duplicado no debería detener la ejecución del programa, sino simplemente no insertarlo en el documento XML final. Si encuentras un duplicado, es conveniente insertar un comentario en el documento XML con la incidencia ocurrida, te servirá para depurar tu código.
9. La solución ronda las 200 líneas de código, aunque se puede hacer en menos de forma elegante. Si tu código supera esa cantidad de líneas, antes de estar completamente operativo, posiblemente te estés complicando en sobremanera. Puedes usar `JOptionPane` para ahorrar trabajo de entrada y salida de información.
10. Usa la clase `DOMUtil` facilitada en los contenidos para que te sea más fácil manejar el archivo XML.