Práctica 2: Árboles genéricos

Ejercicio 1: Para aumentar la funcionalidad de las estructuras de datos que derivan de Tree se desea disponer de mecanismos para obtener su tamaño y un iterador que recorrerá únicamente las hojas del árbol. Para ello se pide implementar las siguientes clases:

- `TreeSizeCalculator`
- `LeafIterator`

Para ello *solo* se deben modificar los ficheros: TreeSizeCalculator.java LeafIterator.java. Los test incluidos ayudan a comprender el funcionamiento de las clases.

Ejercicio 2: Implementar un árbol de tipo *Left-Child Right-Sibling* (LCRS) como estrategia alternativa de diseño de árboles genéricos para tener una funcionalidad equivalente a `LinkedTree`. Para que el ejercicio se considere válido las dos clases deben tener los *mismos métodos y el mismo comportamiento*. Es obligatorio utilizar el prototipo de clase disponible en el Aula Virtual.

Ejercicio 3: Se desea desarrollar en Java la clase `HtmlCrawler`. Esta clase permitirá analizar el árbol de enlaces que se origina a partir de una página web inicial. Para ello, la clase recibirá en su constructor una URL y un entero p ($p > 0$) que representa la máxima profundidad de exploración.
`HtmlCrawler` construirá un árbol genérico con dicha página como raíz y la profundidad proporcionada. Cada nodo del árbol, que se corresponderá a una página web, dispondrá de tantos hijos como enlaces externos contenga dicha página. Los enlaces dentro de una página vienen identificados por la secuencia `href=http://...` de HTML. No se considerarán enlaces que no estén precedidos por la cabecera http. El árbol resultante tendrá una profundidad máxima de p.
Para evitar que un nodo contenga información de una página ya analizada (y por tanto se entre en un bucle) se deberá utilizar una estructura de datos auxiliar que permita detectar tales repeticiones. En ese caso la página no se añadirá como hija de la página analizada en el nodo.
`HtmlCrawler` dispondrá de los siguientes métodos.

- `int getSize()`:devuelve el número de nodos del árbol que contiene el árbol (excluidos los que formen bucles).
- `Iterable<String> getFrontier()`: devuelve las URL de las páginas de máxima profundidad o de aquellas que no tienen enlaces.
- `String getFirstCommonAntecesor(String, String)`: dadas dos URLs de nodos que pertenecen al árbol devuelve la URL del primer nodo padre común.

Los test incluidos en el material suministrado ayudan a entender cómo debe funcionar esta clase. Para hacer este ejercicio solo debe modificarse el fichero `HtmlCrawler.java`, y no deberán modificarse los métodos públicos definidos.
Nota:
En esta página (http://docs.oracle.com/javase/tutorial/networking/urls/readingURL.html) se documenta cómo leer cadenas desde una URL.