Práctica 2: Árboles genéricos

El alumno dispone de código de apoyo con los prototipos a desarrollar, así como de test unitarios para validar su correcto funcionamiento y también para aclarar los requisitos funcionales que se piden. En ningún caso pasar los casos de prueba garantiza que la práctica esté bien y se recomienda que el alumno desarrolle sus propios test unitarios.

Ejercicio 1

Aumentar la funcionalidad de la clase `LinkedTree` implementando el método moveSubtree, cuya cabecera está definida en la clase `LinkedTree`.

Ejercicio 2

Implementar un árbol de tipo *Left-Child Right-Sibling* (LCRS) como estrategia alternativa de diseño de árboles genéricos para tener funcionalidad equivalente a `LinkedTree`. Para que el ejercicio se considere válido las dos clases deben presentar exactamente el mismo comportamiento (incluyendo el método `moveSubtree` del ejercicio 1).

Ejercicio 3

Implementar el iterador `PreorderIterator`, cuyo prototipo está disponible en el Aula Virtual. El ejercicio se considerará que está correctamente implementado si SOLO se utilizan los métodos de la interfaz `Tree` (de tal forma que puedan ser utilizados para cualquier tipo de árbol).

Además, se debe aumentar la funcionalidad de este iterador mediante un tercer constructor que acepta un objeto de la clase `Predicate<Position<E>>` de forma que solo itere por aquellos elementos que cumplan la condición de dicho predicado, es decir, aquellos que para el método `test()` del predicado devuelvan true.

Ejercicio 4

Se desea desarrollar un programa en Java que permita representar virtualmente un sistema de ficheros. El programa recibirá la ruta de un directorio raíz y cargará todos los archivos accesibles a partir de dicha ruta. Ninguna operación de modificación afectará al sistema de ficheros real, solo a su representación virtual en nuestro programa. El programa debe contar con la siguiente funcionalidad:

1. Cargar directorio raíz: Permite introducir un directorio raíz para inicializar nuestro sistema virtual de ficheros. Se cargarán todos los archivos contenidos en el directorio raíz excepto los archivos ocultos.
2. Visualizar el sistema de ficheros: Se devolverá un *String* con la estructura del sistema de ficheros virtual de forma jerárquica. Además, se visualizará un identificador numérico asignado a cada elemento del árbol que permita referenciarlo con acceso O(1). El resultado de llamar este método será como el siguiente:

```
0    ./DirectorioRaiz/
1        SubdirectorioA/
2            ArchivoA.ext
3            ArchivoB.ext
4        SubdirectorioB/
5            SubdirectorioC/
```

```
6           ArchivoC.ext
7       SubdirectorioD/
8       SubdirectiorioE/
9           SubdirectorioF/
10              ArchivoD.ext
```

3. Mover un fichero dentro del árbol: Permite seleccionar un archivo/directorio y moverlo hacia otro directorio.
4. Eliminar fichero: Permite seleccionar un archivo/directorio y eliminarlo del árbol. Si es un directorio se eliminará también su contenido.
5. Filtrar ficheros por nombre: Devolverá una colección iterable que incluye el identificador y el nombre de todos los archivos que contengan en su nombre una cadena de texto. La búsqueda se realizará a partir de un archivo cualquiera dado su identificador.
6. Filtrar ficheros por tamaño: Al igual que en el punto 5 se devolverá una colección con todos los ficheros cuyo tamaño esté en los límites establecidos. La búsqueda se realizará a partir de un archivo cualquiera dado su identificador.
7. Obtener ruta virtual de un fichero: Se devolverá la ruta virtual del fichero según la estructura del árbol. No necesariamente se corresponde con la ruta física de ese fichero porque el árbol puede haber sido modificado.
8. Obtener ruta real de un fichero: Se devolverá la ruta real del archivo; aquella desde dónde fue leído originalmente.
9. De forma opcional (no será evaluable) se puede elaborar un menú interactivo que permita usar este programa. Esto puede ayudar a comprobar su correcto funcionamiento.