UNLP. Facultad de Informática.
**Algoritmos y Estructuras de Datos 2006.** **Trabajo Práctico 6**

## Arboles Generales

Objetivos
- Modelar un árbol n-ario con distintas representaciones.
- Recorrer, calcular la altura e implementar aplicaciones varias sobre árboles n-arios.
- Calcular el orden de la complejidad algorítmica de los métodos anteriores, en base a la altura o cantidad de nodos.

Ejercicio 1.
Un árbol general con la representación de **Lista de Hijos** puede implementarse de dos maneras:

**A-**

```
public class ArbolGeneral {
        private NodoGeneral raíz;
}
```

```
public class NodoGeneral {
        private Object dato;
        private Lista: listaHijos;  //listaHijos es una Lista de
                                    //objetos NodoGeneral
}
```

**B-**

```
public class ArbolGeneral {
        private Object dato;
        private Lista listaHijos;  // listaHijos es una Lista de objetos ArbolGeneral
}
```

***Nota:*** *la clase Lista es la utilizada en la práctica 5.*

Análogamente se puede implementar de dos formas un árbol general con la representación de **Hijo más izquierdo – Hermano derecho**.

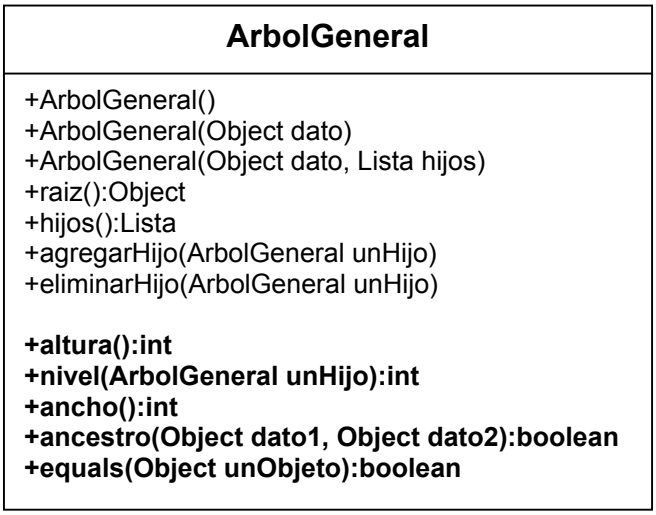
Considere la siguiente especificación de la clase **ArbolGeneral**:

| **ArbolGeneral** |
|---|
| +ArbolGeneral()<br>+ArbolGeneral(Object dato)<br>+ArbolGeneral(Object dato, Lista hijos)<br>+raiz():Object<br>+hijos():Lista<br>+agregarHijo(ArbolGeneral unHijo)<br>+eliminarHijo(ArbolGeneral unHijo) |

El método **raiz():Object**, retorna el dato almacenado en la raíz del árbol.
El método **hijos():Lista**, retorna la lista de hijos del árbol
El método **agregarHijo(ArbolGeneral unHijo)**, agrega un hijo a la lista de hijos del árbol
El método **eliminarHijo(ArbolGeneral unHijo)**, elimina un hijo a la lista de hijos del árbol

Utilice la estructura presentada en el inciso **B** para:
a) Implementar la clase **ArbolGeneral** en el paquete **estructuradedatos.arbolesgenerales**, con la representación lista de hijos.
b) Implementar la clase **ArbolGeneral** en el paquete **estructuradedatos.arbolesgenerales**, con la representación hijo más izquierdo – hermano derecho.

Ejercicio 2.
Agregue a la clase **ArbolGeneral** los siguientes métodos:




a) **altura(): int** que devuelve la altura del árbol. La altura del árbol es la longitud del camino más largo desde el nodo raíz a una hoja.
b) **nivel(ArbolGeneral unHijo):int** que devuelve la profundidad o nivel del nodo en el árbol. El nivel de un nodo es la longitud del único camino de la raíz al nodo.
c) **ancho():int** La amplitud de un árbol se define como la cantidad de nodos que se encuentran en el nivel que posee la mayor cantidad de nodos.
d) **ancestro(Object dato1, Object dato2):boolean** que determina si dato1 es ancestro de dato2, es decir si hay un camino de dato1a dato2. Considere que no se repite el contenido de los nodos.
e) **equals(Object unObjeto):bolean** que determina si dos árboles son o no similares. Dos árboles son similares si son estructuralmente iguales, aunque no coincidan los valores que se almacenan en los nodos respectivos de los árboles.

***Nota:*** *tenga presente que no importa que representación interna utilizó. Estas operaciones deberían estar implementadas a partir de las operaciones especificadas en el ejercicio 1.*

Ejercicio 3.
Modelizar e implementar en JAVA una estructura de directorio de DOS: existe un directorio raíz; un directorio puede contener archivos y a su vez otros directorios. Tanto los archivos como los directorios tienen un nombre. Los archivos también tienen un tamaño. Para este ejercicio realice métodos que al menos resuelvan los siguientes problemas:
a.- devolver los archivos de un directorio.
b.- devolver los subdirectorios de un directorio.
c.- retornar el tamaño de un directorio: sólo la suma de los tamaños de los archivos que contiene.
d.- retornar el path (concatenación de los nombres de los directorios) con mayor cantidad de caracteres comenzando en el directorio raíz.

Ejercicio 4.
Modelizar e implementar en JAVA la siguiente situación:
En una empresa los empleados están categorizados con un número que va del 1 al 10, siendo el 1 el presidente, el 2 el vicepresidente, 3 los gerentes y así siguiendo. También los mismos poseen una antigüedad. Esta información está dispuesta en un árbol general. Para este ejercicio realice métodos que al menos resuelvan los siguientes problemas:
a.- devolver la cantidad de trabajadores por categoría.
b.- determinar que categoría posee la mayor cantidad de trabajadores.
c.- contar la cantidad total de trabajadores.
d.- Sea la situación en donde el presidente deja su función, reemplazarlo por la persona más antigua de sus subordinados, quién a su vez es reemplazada de la misma forma.

Ejercicio 5.
Implemente una clase en Java llamada **Imagen** que permita crear imágenes en blanco y negro y utilice como representación interna una matriz.
Una forma de comprimir una imagen es transformarla a un árbol 4-ario. Si toda la matriz tiene un mismo color, se debe definir un nodo con ese color. En caso contrario, se divide la matriz en cuatro partes, se define un nodo con 4 hijos, donde cada hijo es la conversión de cada una de las partes de la matriz.
Realice un método de instancia llamado **imagenComprimida()** que devuelva la representación de la imagen en su árbol correspondiente.