UNLP. Facultad de Informática.
Algoritmos y Estructuras de Datos

Práctica 3
Árboles Generales

Objetivos

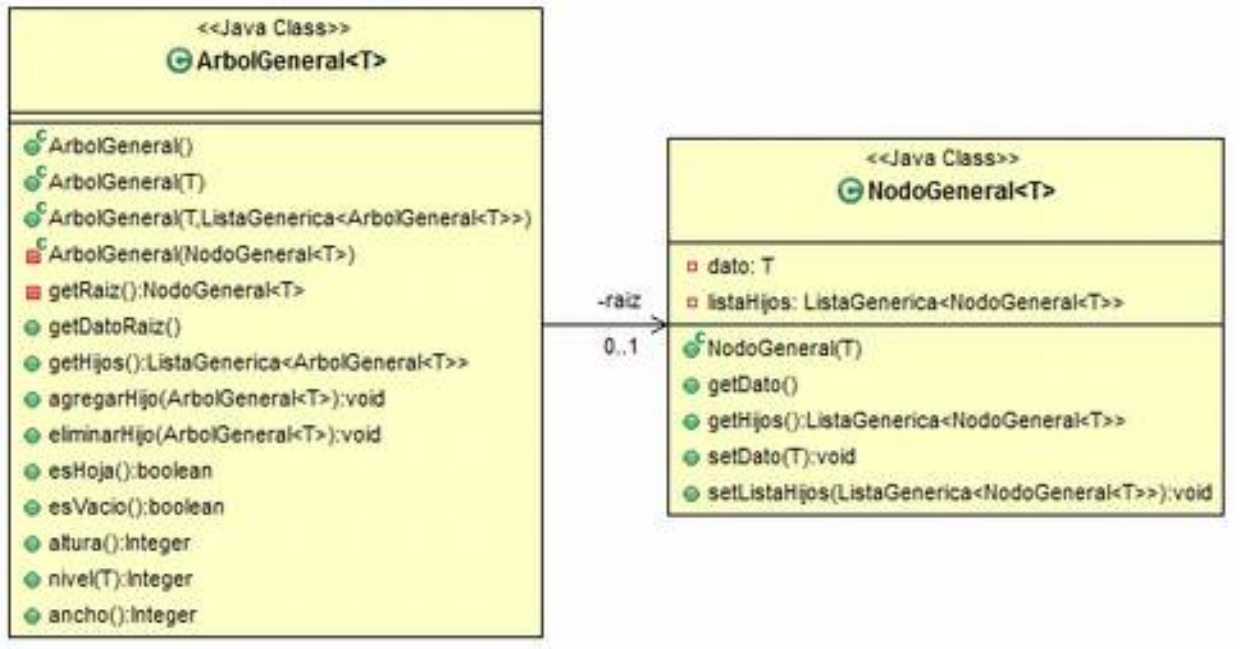
- Modelar un árbol n-ario
- Realizar recorridos sobre árboles generales
- Implementar ejercicios de aplicación sobre árboles generales.

Importante: Se recomienda continuar trabajando dentro del proyecto AyED y generar paquetes y subpaquetes para esta práctica.
Descargue el archivo **tp03_ag.zip** El archivo zip descargado desde la página de la cátedra no es un proyecto eclipse, por lo tanto:

1. Descomprima el archivo zip.
2. Sobre la carpeta **src** de su proyecto AyED haga click con el botón derecho del mouse y seleccione la opción *Import > FileSystem*.
3. Haga click en "Browse", busque la carpeta descomprimida y seleccione la carpeta **tp03_ag** (haga click para que aparezca el check seleccionado).
4. Haga click en el botón finalizar.

Ejercicio 1

Considere la siguiente especificación de la clase **ArbolGeneral** (con la representación de **Lista de Hijos**)



Nota: la clase **Lista** es la utilizada en la práctica 2.

El constructor **ArbolGeneral()** inicializa un árbol vacío, es decir, la raíz en null.

El constructor **ArbolGeneral(T dato)** inicializa un árbol que tiene como raíz un nodo general. Este nodo tiene el dato pasado como parámetro y una lista vacía.

El constructor **ArbolGeneral(T dato, ListaGenerica<ArbolGeneral<T>> hijos)** inicializa un árbol que tiene como raíz un nodo general. Este nodo tiene el dato pasado como parámetro y tiene como hijos una copia de la lista pasada como parámetro. Tenga presente que la Lista pasada

como parámetro es una lista de árboles generales, mientras que la lista que se debe guardar en el nodo general es una lista de nodos generales. Por lo cuál, de la lista de árboles generales debe extraer la raíz (un Nodo General) y guardar solamente este objeto.

El constructor **ArbolGeneral(NodoGeneral<T> nodo)** inicializa un árbol donde el nodo pasado como parámetro es la raíz. Notar que este constructor NO es público.

El método **getRaiz():NodoGeneral<T>** retorna el nodo ubicado en la raíz del árbol. Notar que NO es un método público.

El método **getDatoRaiz():T** retorna el dato almacenado en la raíz del árbol (NodoGeneral).

El método **getHijos():ListaGenerica<ArbolGeneral<T>>**, retorna la lista de hijos de la raíz del árbol. Tenga presente que la lista almacenada en la raíz es una lista de nodos generales, mientras que debe devolver una lista de árboles generales. Para ello, por cada hijo, debe crear un árbol general que tenga como raíz el Nodo General hijo.

El método **agregarHijo(ArbolGeneral<T> unHijo)** agrega unHijo a la lista de hijos del árbol. Es decir, se le agrega a la lista de hijos de la raíz del objeto receptor del mensaje el Nodo General raíz de unHijo.

El método **eliminarHijo(ArbolGeneral<T> unHijo)** elimina unHijo del árbol. Con la misma salvedad indicada en el método anterior.

Los métodos **esHoja(), esVacio(), altura(), nivel(T) y ancho()** se resolverán más adelante.

Analice la implementación en JAVA de las clases **ArbolGeneral** y **NodoGeneral** brindadas por la cátedra.

Ejercicio 2

a) ¿Cuál/es es/son el/los recorrido/s que conoce para recorrer en profundidad un árbol general? Explique brevemente.
b) ¿Cuál/es es/son el los recorrido/s que conoce para recorrer por niveles un árbol general? Explique brevemente.
c) ¿Existe alguna diferencia entre los recorridos pre-orden, post-orden, en-orden para recorrer los árboles generales respecto de los árboles binarios? Justifique su respuesta.
d) ¿Existe alguna noción de orden entre los elementos de un árbol general? Justifique su respuesta.
e) En un árbol general se define el grado de un nodo como el número de hijos de ese nodo y el grado del árbol como el máximo de los grados de los nodos del árbol. ¿Qué relación encuentra entre los Árboles Binarios sin tener en cuenta la implementación? Justifique su respuesta.

Ejercicio 3

Implemente en la clase **ArbolGeneral<T>** los siguientes métodos:

```
public ListaEnlazadaGenerica<T> preOrden(){
//Método que retorna una lista con los elementos del Árbol receptor, en  pre order

//código
}

public ListaEnlazadaGenerica<T> inOrden(){
// Método que retorna una lista con los elementos del Árbol receptor, en in order

//código
```

}

public ListaEnlazadaGenerica<T> postOrden(){
// Método que retorna una lista con los elementos del Árbol receptor, en post order

//código
}

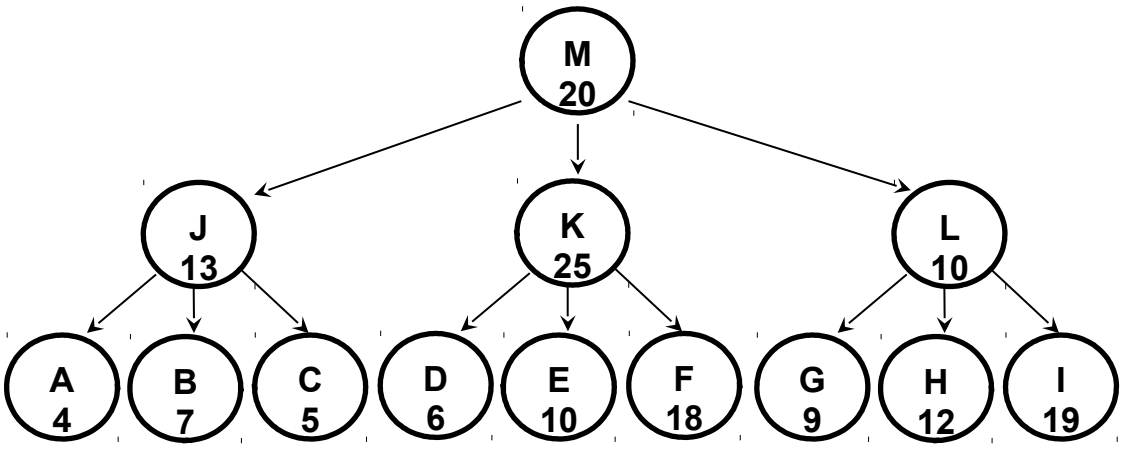
public ListaEnlazadaGenerica<T> recorridoPorNiveles (){
// Método que retorna una lista con los elementos del Árbol receptor, en orden de recorrido por
//niveles de arriba hacia abajo y de izquierda a derecha, incluyendo algún elemento que indique
// el fin de cada nivel

//código
}

Ejercicio 4
Implemente en la clase **ArbolGeneral** los siguientes métodos **(** incisos a) y b) con sintaxis Java y los incisos c), d) y e) con al menos pseudocódigo**)**:

a) **public boolean esHoja()** devuelve true si el nodo en cuestión no tiene hijos, false en caso contrario.
b) **public boolean esVacio()** devuelve true si el nodo en cuestión no tiene contenido (dato raíz es null).
c) **public int altura(): int** devuelve la altura del árbol, es decir, la longitud del camino más largo desde el nodo raíz hasta una hoja.
d) **public int nivel(T dato)** devuelve la profundidad o nivel del dato en el árbol. El nivel de un nodo es la longitud del único camino de la raíz al nodo.
e) **public int ancho(): int** la amplitud (ancho) de un árbol se define como la cantidad de nodos que se encuentran en el nivel que posee la mayor cantidad de nodos.

Ejercicio 5
Usted está encargado de la comunicación de una empresa. El esquema de comunicación está organizado en una estructura jerárquica, en donde cada nodo envía el mensaje a sus descendientes. Cada nodo posee el tiempo que tarda en transmitir el mensaje.



Usted debe devolver el mayor promedio por nivel. Por ejemplo, el promedio del nivel 0 es 20, el del

nivel 1 es 16 y el del nivel 2 es 10. Por lo tanto, debe devolver 20.

a) Indique qué tipo de recorrido utilizará y justifique por qué el recorrido elegido le ayuda a resolver el problema.
b) Implementar en java en una clase nueva, el método **public Integer devolverMaximoPromedio (ArbolGeneral<AreaEmpresa> arbol)** donde **AreaEmpresa** es una clase que representa a un área de la empresa mencionada y que contiene una identificación de la misma representada con un **String** y una tardanza de transmisión de mensajes interna representada con **Integer**.

Ejercicio 6
Se dice que un nodo n es ancestro de un nodo m si existe un camino desde n a m.
Se dice que un nodo n es descendiente de un nodo m si existe un camino desde m a n.
Implemente un método en la clase ArbolGeneral con la siguiente firma:

public Boolean esAncestro(T a, T b)

El cual determine si un valor a es ancestro de un valor b.

Ejercicio 7
Sea una red de agua potable, la cual comienza en un caño maestro y el mismo se va dividiendo sucesivamente hasta llegar a cada una de las casas.
Por el caño maestro ingresan x cantidad de litros y en la medida que el caño se divide, el caudal se divide en partes iguales en cada una de las divisiones. Es decir, si un caño maestro recibe 1000 litros y se divide en 4 partes, cada división tiene un caudal de 250 litros.
Luego, si una de esas divisiones se vuelve a dividir en 5 partes, cada una tendrá un caudal de 50 litros y así sucesivamente. Usted debe implementar una clase **RedDeAguaPotable** que contenga un método **public double minimoCaudal()**, que calcule el caudal de cada nodo y determine cuál es el mínimo caudal que recibe una casa.