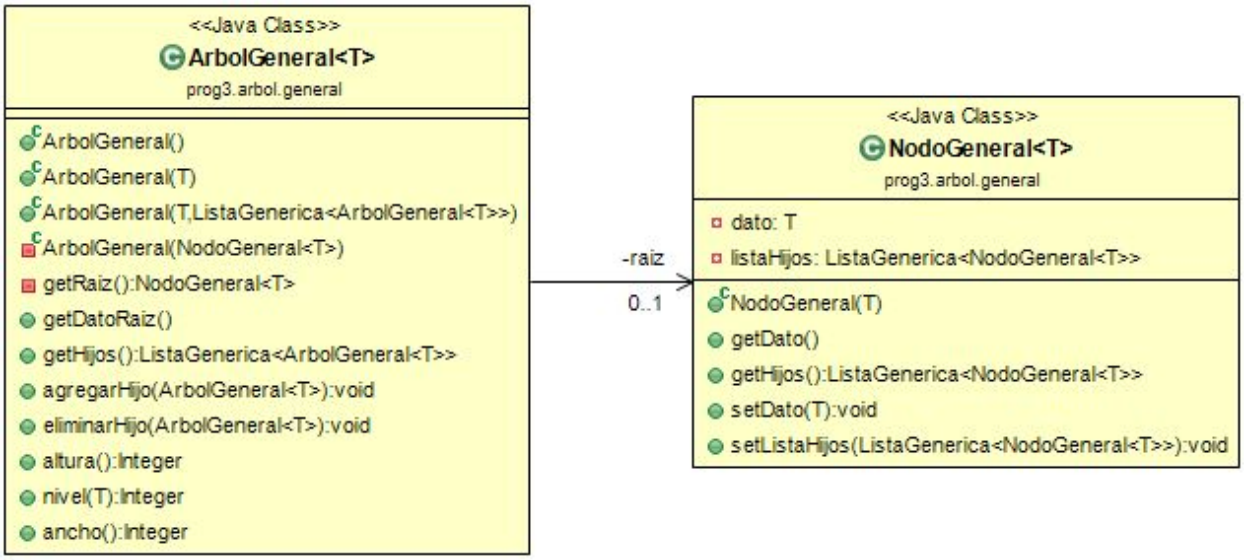
Programación III
TEMA 5: Árbol general
Práctica nº 5 - A

Puede continuar trabajando en su proyecto Programacion3. El archivo zip descargado desde la página de la cátedra no es un proyecto eclipse, por tanto:

1. descomprima el archivo zip
2. sobre la carpeta **src** de su proyecto Programacion3 haga click con el botón derecho del mouse y seleccione la opción *Import* > *FileSystem*.
3. Haga click en "Browse" y busque la carpeta descomprimida y seleccione la carpeta **src** (haga click para que aparezca el check seleccionado)
4. Haga click en el botón finalizar

1. Considere la siguiente representación de árboles generales provista por la cátedra:



Nota: la clase **Lista** es la utilizada en la práctica 3, vamos a utilizar la representación **lista de hijos**.

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, Lista<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():Lista<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.

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

  a. Agregue los métodos públicos **esHoja()** y **esVacio()** a la implementación de ArbolGeneral. Tenga en cuenta que un árbol es vacío cuando **no posee dato, ni hijos** (que no es lo mismo que decir que el árbol es null).

  b. Implemente el método **altura(): int** que devuelve la altura del árbol, es decir, la longitud del camino más largo desde el nodo raíz hasta una hoja. ***Pista****: el mensaje altura debe chequear si el árbol es una sola hoja o no. Si el árbol es una sola hoja, se devuelve 0. Si no, se utiliza el mensaje getHijos() para obtener la lista de hijos (recuerde que devuelve una lista de* ***árboles hijos****). Luego, debe iterar por cada uno de los hijos, delegando el mensaje* ***altura()*** *hasta llegar a una hoja, donde evaluará si se alcanzó una altura máxima. A medida que se avanza en profundidad sobre el árbol, se va sumando 1.*

  c. Implemente el método **include(T dato): boolean** que devuelve **true** si dato se encuentra en el árbol, y **false** en caso contrario.

  d. Implemente el método **nivel(T dato): int** que devuelve el nivel donde se encontró el dato en el árbol. El nivel de un nodo es la longitud del único camino de la raíz al nodo. ***Pista:*** *si el nodo raíz posee el mismo dato que pasado como parámetro, se retorna 0. En caso contrario, se debe buscar en cuáles de los subárboles hijos se encuentra el dato.*

  e. Implemente el método **ancho(): int** que devuelve la amplitud (ancho) de un árbol, es decir la cantidad de nodos que se encuentran en el nivel que posee la mayor cantidad de nodos. ***Pista:*** *realice un recorrido por niveles. Encole inicialmente la raíz del árbol y luego una marca null (o un nodo vacío ó el número de nivel) para indicar el fin de nivel. Mientras la cola no se vacía, itere. En cada iteración extraiga el tope de la cola, y con la operación getHijos() encole los mismos. Cuando encuentra la marca de fin de nivel cuente si los elementos del nivel es mayor a la máxima cantidad que poseía.*