# Árboles Generales

# Implementación

```
public class ArbolGeneral<T> {

	private T dato;

	private ListaGenerica<ArbolGeneral<T>> hijos = new ListaGenericaEnlazada<ArbolGeneral<T>>();

	public ArbolGeneral(T dato) {
		this.dato = dato;
		this.hijos = new ListaGenericaEnlazada<ArbolGeneral<T>>();
	}
	public ArbolGeneral(T dato, ListaGenerica<ArbolGeneral<T>> hijos) {
		this(dato);
		this.hijos = hijos;
	}

	public T getDato() {
		return dato;
	}

	public void setDato(T dato) {
		this.dato = dato;
	}

	public void setHijos(ListaGenerica<ArbolGeneral<T>> hijos) {
		if (hijos==null)     this.hijos = new ListaEnlazadaGenerica<ArbolGeneral<T>>();
		else     this.hijos = hijos;

	}
```

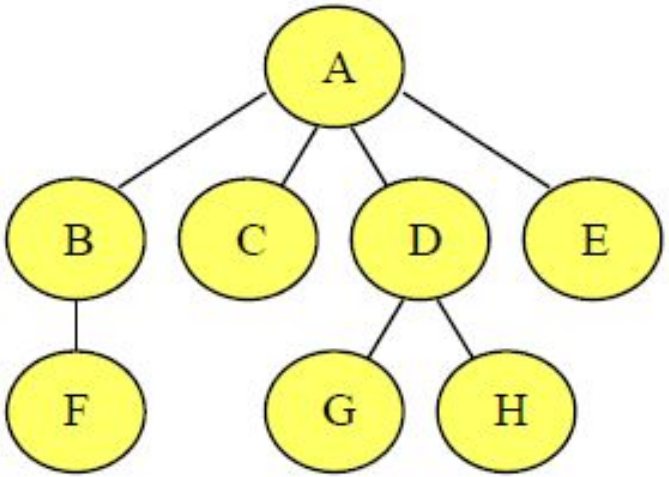
```
    public ListaGenerica<ArbolGeneral<T>> getHijos() {
        return this.hijos;
    }

    public void agregarHijo(ArbolGeneral<T> unHijo) {
        this.getHijos().agregarFinal(unHijo);
    }

    public boolean esHoja() {
        return !this.tieneHijos();
    }

    public boolean tieneHijos() {
        return this.hijos != null && !this.hijos.esVacia();
    }

    public boolean esVacio() {
        return this.dato == null && !this.tieneHijos();
    }

    public void eliminarHijo(ArbolGeneral<T> hijo) {
        if (this.tieneHijos()) {
            ListaGenerica<ArbolGeneral<T>> hijos = this.getHijos();
            if (hijos.incluye(hijo))
                hijos.eliminar(hijo);
        }
    }
}
```

# Método altura()

**Definición:** La **altura** de un nodo es la longitud del camino más largo desde el nodo hasta una hoja.

- Las hojas tienen altura cero.
- La altura de un árbol es la altura del nodo raíz

**¿Posibles soluciones?**



# Posibles Soluciones

1. **Recorrer en profundidad**, delegando en los **hijos árboles** al cálculo de la altura.

    a. si el árbol es una hoja devolverá 0 como altura.
    b. si el árbol no es una hoja, toma el valor más grande entre lo devuelto por sus hijos, le suma 1 y devuelve ese valor.

2. **Recorrer en profundidad**, delegando en los **hijos árboles** al cálculo de la altura pero resolviendo en las hojas.

a. necesito un método privado para poder enviar como argumento la altura acumulada y devolver el máximo.
b. sumo 1 a la altura acumulada cada vez que avanzo en profundidad y delego el cálculo.
c. al llegar a una hoja, verificar si el valor calculado es máximo y actualizar dicho máximo.

3. **Recorrer por niveles.**

a. necesito una variable donde almacenar el número de nivel actual
b. cuando alcance el último nivel, la variable almacenará el valor de altura.

# Método altura (opción 1)

```
public Integer altura() {
    if (!this.esVacio()) {
        if (this.esHoja())
            return 0;
        else {
            ListaGenerica<ArbolGeneral<T>> hijos = this.getHijos();
            ArbolGeneral<T> unHijo = null;
            int maximo = 0;
            int altCalc = 0;
            hijos.comenzar();
            while (!hijos.fin()) {
                unHijo = hijos.proximo();
                altCalc = unHijo.altura();
                if (maximo < altCalc)
                    maximo = altCalc;

            }
            return 1 + maximo;
        }
    }
    return 0;
}
```

# Método altura (opción 2)

La clase Resultado solo encapsula un atributo donde se almacenará el máximo valor

```
private void alturav2(int contador, Resultado max) {
	if (!this.esVacio()) {
		if (this.esHoja()) {
			if (max.getValor() < contador)
				max.setValor(contador);
		} else {
			ListaGenerica<ArbolGeneral<T>> hijos = this.getHijos();
			hijos.comenzar();
			while (!hijos().fin()) {
				hijos().proximo().alturav2(contador + 1, max);
			}

		}
	}
}
```