








Arboles Generales

Estructura

<<Java Class>>

 **ArbolGeneral<T>**

tp04.ejercicio1

-  **ArbolGeneral(T)**
-  **ArbolGeneral(T, ListaGenerica<ArbolGeneral<T>>)**
-  **ArbolGeneral(NodoGeneral<T>)**
-  **getRaiz():NodoGeneral<T>**
-  **setRaiz(NodoGeneral<T>):void**
-  **getDatoRaiz()**
-  **getHijos():ListaGenerica<ArbolGeneral<T>>**








-raiz

>

<<Java Class>>

 **NodoGeneral<T>**

tp04.ejercicio1

-  **dato: T**
-  **listaHijos: ListaGenerica<NodoGeneral<T>>**
-  **NodoGeneral(T)**
-  **getDato()**
-  **setDato(T):void**
-  **getListaHijos():ListaGenerica<NodoGeneral<T>>**
-  **setListaHijos(ListaGenerica<NodoGeneral<T>>):void**

Arboles Generales

```
package tp04;
public class ArbolGeneral<T> {

    private NodoGeneral<T> raiz;

    public ArbolGeneral(T dato) {
        raiz = new NodoGeneral<T>(dato);
    }

    public ArbolGeneral(T dato,
                        ListaGenerica<ArbolGeneral<T>> hijos) {
        this(dato);
        ListaGenerica<NodoGeneral<T>> newList =
            new ListaEnlazadaGenerica<NodoGeneral<T>>();
        hijos.comenzar();
        while (!hijos.fin()) {
            ArbolGeneral<T> arbolTemp = hijos.proximo();
            newList.agregar(arbolTemp.getRaiz());
        }
        raiz.setListaHijos(newList);
    }

    private ArbolGeneral(NodoGeneral<T> nodo) {
        raiz = nodo;
    }

    private NodoGeneral<T> getRaiz() {
        return raiz;
    }

    . . .
}
```

```
package tp04;
public class NodoGeneral<T> {
    private T dato;
    private ListaGenerica<NodoGeneral<T>> listaHijos;

    NodoGeneral(T dato) {
        this.dato = dato;
        listaHijos = new ListaGenerica<NodoGeneral<T>>();
    }

    public T getDato() {
        return dato;
    }

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

    public ListaGenerica<NodoGeneral<T>> getListaHijos()
    {
        return listaHijos;
    }

    void setListaHijos(Lista listaHijos) {
        this.listaHijos = listaHijos;
    }
}
```

Arboles Generales

Recorrido Preorden

Implementar un método en `ArbolGeneral` que retorne una lista con los datos del árbol recorrido en preorden

```
package ayed;
public class ArbolGeneral<T> {
    private NodoGeneral<T> raiz;
    . . .
    public ListaEnlazadaGenerica<T> preOrden() {
        ListaEnlazadaGenerica<T> lis = new ListaEnlazadaGenerica<T>();
        this.getRaiz().preOrden(lis);
        return lis;
    }
}
```

```
package ayed;
public class NodoGeneral<T> {
    private T dato;
    private ListaGenerica<NodoGeneral<T>> listaHijos;
    . . .
    void preOrden(ListaGenerica<T> l) {
        l.agregar(this.getDato());
        ListaGenerica<NodoGeneral<T>> lHijos = this.getListaHijos();
        lHijos.comenzar();
        while(!lHijos.fin()) {
            (lHijos.proximo()).preOrden(l);
        }
    }
}
```

Caso de uso

```
ArbolGeneral<String> a1 = new ArbolGeneral<String>("1");
ArbolGeneral<String> a2 = new ArbolGeneral<String>("2");
ArbolGeneral<String> a3 = new ArbolGeneral<String>("3");
ListaGenerica<ArbolGeneral<String>> hijos = new ListaEnlazadaGenerica<ArbolGeneral<String>>();
hijos.agregar(a1); hijos.agregar(a2); hijos.agregar(a3);
ArbolGeneral<String> a = new ArbolGeneral<String>("0", hijos);
System.out.println("Datos del Arbol: "+a.preOrden());
```

Arboles Generales

Contar cantidad de nodos

```
package ayed;

public class ArbolGeneral {
    private NodoGeneral<T> raiz;
    . . .

    public int contarEnPreOrden() {
        return this.getRaiz().contarEnPreOrden();
    }
}
```

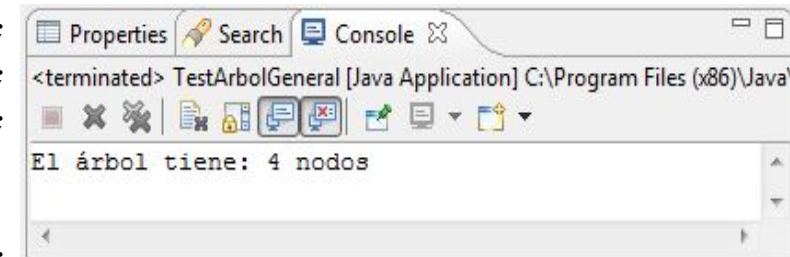
```
package ayed;

public class NodoGeneral<T> {
    private T dato;
    private ListaGenerica<NodoGeneral<T>> listaHijos
    ...
    int contarEnPreOrden() {
        int res=1;
        ListaGenerica<NodoGeneral<T>> lHijos =
            this.getListHijos();

        lHijos.comenzar();
        while(!lHijos.fin()){
            res = res + lHijos.proximo().contarEnPreOrden();
        }
        return res;
    }
}
```

Caso de uso

```
ArbolGeneral<String> a1 = new ArbolGeneral<String>("1");
ArbolGeneral<String> a2 = new ArbolGeneral<String>("2");
ArbolGeneral<String> a3 = new ArbolGeneral<String>("3");
ListaGenerica<ArbolGeneral<String>> hijos = new
ListaEnlazadaGenerica<ArbolGeneral<String>>();
hijos.agregar(a1); hijos.agregar(a2); hijos.agregar(a3);
ArbolGeneral<String> a = new ArbolGeneral<String>("0", hijos);
System.out.println("El árbol tiene:"+a.contarEnPreOrden()+" nodos");
```



Arboles Generales

Contar cantidad de nodos

```
package ayed;

public class ArbolGeneral {
    private NodoGeneral<T> raiz;
    . . .

    public int contarEnPreOrdenArray() {
        int[] a = new int[1];
        this.getRaiz().contarEnPreOrdenArray(a);
        return a[0];
    }
}
```

```
package ayed;

public class NodoGeneral<T>{
    private T dato;
    private ListaGenerica<NodoGeneral<T>> listaHijos;

    void contarEnPreOrdenArray(int[] x){
        x[0]++;
        ListaGenerica<NodoGeneral<T>> lHijos =
            this.getListaHijos();

        lHijos.comenzar();
        while(!lHijos.fin()){
            lHijos.proximo().contarEnPreOrdenArray(x);
        }
    }
    . . .
}
```

Caso de uso

```
ArbolGeneral<String> a1 = new ArbolGeneral<String>("1");
ArbolGeneral<String> a2 = new ArbolGeneral<String>("2");
ArbolGeneral<String> a3 = new ArbolGeneral<String>("3");
ListaGenerica<ArbolGeneral<String>> hijos = new
ListaEnlazadaGenerica<ArbolGeneral<String>>();
hijos.agregar(a1); hijos.agregar(a2); hijos.agregar(a3);
ArbolGeneral<String> a = new ArbolGeneral<String>("0", hijos);
System.out.println("El árbol tiene: "+a.sumarEnPreOrdenArray() +" nodos");
```

