









# 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

## Código Fuente – Constructores, this()

```
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>> lista = new ListaEnlazadaGenerica<NodoGeneral<T>>(hijos.comenzar());
        while (!hijos.fin()) {
            ArbolGeneral<T> arbolTemp = hijos.proximo();
            lista.agregar(arbolTemp.getRaiz());
        }
        raiz.setListaHijos(lista);
    }
    private ArbolGeneral(NodoGeneral<T> nodo){
        raiz = nodo;
    }
    private NodoGeneral<T> getRaiz() {
        return raiz;
    }
    public ListaGenerica<ArbolGeneral<T>> getHijos() {
        ListaGenerica<ArbolGeneral<T>> lista = new ListaEnlazadaGenerica<ArbolGeneral<T>>();
        ListaGenerica<NodoGeneral<T>> hijos = this.getRaiz().getHijos();
        lista.comenzar(); hijos.comenzar();
        while (!hijos.fin()) {
            lista.agregarFinal(new ArbolGeneral<T>(hijos.proximo()));
        }
        return lista;
    }
}
```

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

    NodoGeneral(T dato) {
        this.dato=dato;
        listaHijos=new ListaEnlazadaGenerica<NodoGeneral<T>>();
    }
    public void setDato(T dato) {
        this.dato = dato;
    }
    public void setListaHijos(ListaGenerica<NodoGeneral<T>> lista) {
        this.listaHijos = lista;
    }
    public T getDato() {
        return this.dato;
    }
    public ListaGenerica<NodoGeneral<T>> getHijos() {
        return this.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.preOrden(lis);
        return lis;
    }
    private void preOrden(ListaGenerica<T> l) {
        l.agregarFinal(this.getDatoRaiz());
        ListaGenerica<ArbolGeneral<T>> lHijos = this.getHijos();
        lHijos.comenzar();
        while (!lHijos.fin()) {
            (lHijos.proximo()).preOrden(l);
        }
    }
}
```

### Caso de uso

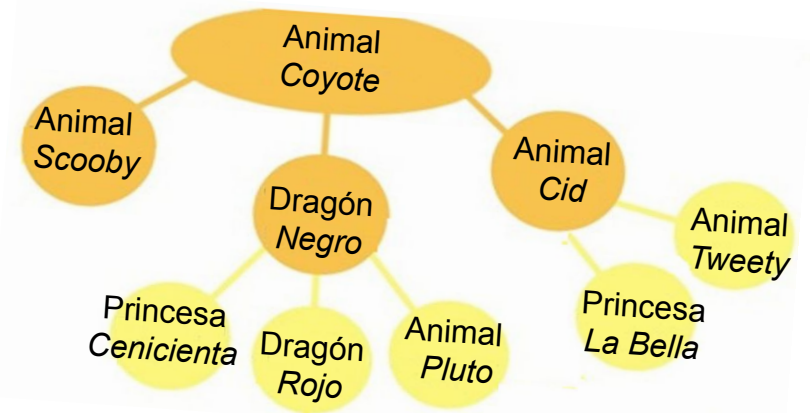
```
ArbolGeneral<String> a1 = new ArbolGeneral<String>("1");
ArbolGeneral<String> a2 = new ArbolGeneral<String>("2");
ArbolGeneral<String> a3 = new ArbolGeneral<String>("3");
ListaGenerica<ArbolGeneral<String>> hijos = new ListaEnlazadaGenerica<ArbolGeneral<String>>();
hijos.agregar(a1); hijos.agregar(a2);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

# Arboles Generales

## Ejercicio de parcial – Encontrar a la Princesa

Dado un árbol general compuesto por personajes, donde puede haber dragones, princesas y otros, se denominan nodos accesibles a aquellos nodos tales que a lo largo del camino del nodo raíz del árbol hasta el nodo (ambos inclusive) no se encuentra ningún dragón.

Implementar un método que devuelva una lista con un camino desde la raíz a una Princesa sin pasar por un Dragón –sin necesidad de ser el más cercano a la raíz-. Asuma que existe al menos un camino accesible.



# Arboles Generales

## Ejercicio de parcial – Encontrar a la Princesa

```
package parcial.juego;
```

```
public class Personaje {  
    private String nombre;  
    private String tipo;    //Dragon, Princesa, Animal, etc.
```

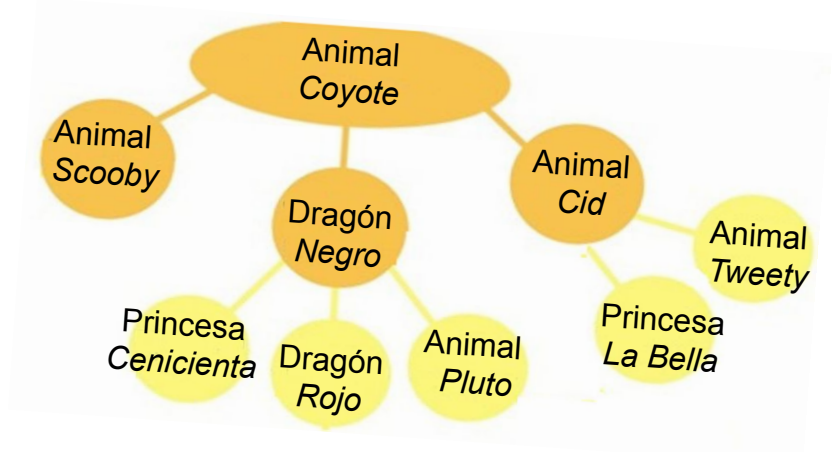
```
    public Personaje(String nombre, String tipo) {  
        this.nombre = nombre;  
        this.tipo = tipo;  
    }
```

```
    public String getNombre() {  
        return nombre;  
    }
```

```
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
    . . .
```

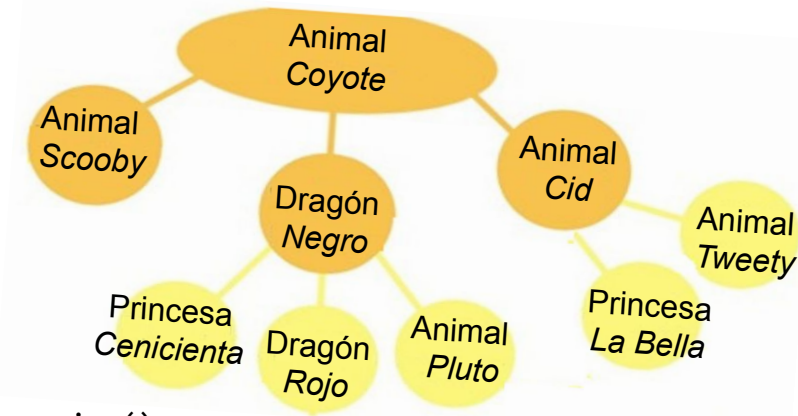
```
    public boolean esDragon(){  
        return this.getTipo().equals("Dragon");  
    }
```

```
    public boolean esPrincesa(){  
        return this.getTipo().equals("Princesa");  
    }  
}
```



# Arboles Generales

## Ejercicio de parcial – Encontrar a la Princesa



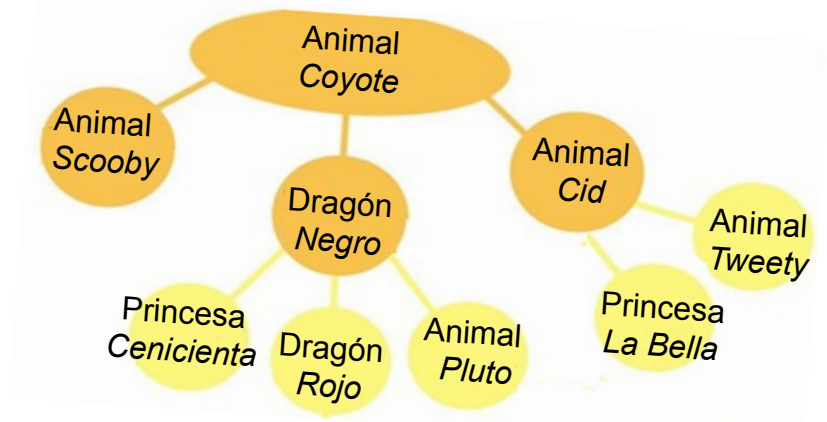
```
public class Juego {  
  
    public void encontrarPrincesa(ArbolGeneral<Personaje> arbol) {  
        ListaGenerica<Personaje> lista = new ListaEnlazadaGenerica<Personaje>();  
        lista.agregarInicio(arbol.getDatoRaiz());  
        ListaGenerica<Personaje> camino = new ListaEnlazadaGenerica<Personaje>();  
        encontrarPrincesa(arbol, lista, camino);  
        System.out.print("Se encontró a la Princesa en el camino: " + camino);  
    }  
  
    private void encontrarPrincesa(ArbolGeneral<Personaje> arbol, ListaGenerica<Personaje> lista,  
                                   ListaGenerica<Personaje> camino) {  
  
        Personaje p = arbol.getDatoRaiz();  
        if (p.esPrincesa()) {  
            clonar(lista, camino);  
        }  
        if (camino.esVacia()) {  
            ListaGenerica<ArbolGeneral<Personaje>> lHijos = arbol.getHijos();  
            lHijos.comenzar();  
            while (!lHijos.fin() && camino.esVacia()) {  
                ArbolGeneral<Personaje> aux = lHijos.proximo();  
                if (!aux.getDatoRaiz().esDragon()) {  
                    lista.agregarFinal(aux.getDatoRaiz());  
                    encontrarPrincesa(aux, lista, camino);  
                    lista.eliminarEn(lista.tamano());  
                }  
            }  
        }  
    }  
}
```

```
public void clonar(ListaGenerica<Personaje> origen,  
                  ListaGenerica<Personaje> destino) {  
    origen.comenzar();  
    while (!origen.fin()) {  
        destino.agregarFinal(origen.proximo());  
    }  
}
```

# Arboles Generales

## Ejercicio de parcial – Encontrar a la Princesa

```
public class Juego {  
  
    public ListaEnlazadaGenerica<Personaje> encontrarPrincesa1(ArbolGeneral<Personaje> arbol){  
        ListaEnlazadaGenerica<Personaje> lista = new ListaEnlazadaGenerica<Personaje>();  
        if (arbol.getDatoRaiz().esPrincesa() || arbol.getDatoRaiz().esDragon() || arbol.esHoja()){  
            if (arbol.getDatoRaiz().esPrincesa()){  
                Personaje p = arbol.getDatoRaiz();  
                lista.agregarInicio(p);  
            }  
            return lista;  
        }  
        ListaGenerica<ArbolGeneral<Personaje>> lHijos = arbol.getHijos();  
        lHijos.comenzar();  
        while(!lHijos.fin() && lista.esVacia()){  
            lista = encontrarPrincesa1(lHijos.proximo());  
            if(!lista.esVacia()){  
                lista.agregarInicio(arbol.getDatoRaiz());  
                //break; // o lista.esVacia() en el while  
            }  
        }  
        return lista;  
    }  
}
```





# Arboles Generales

## Ejercicio de parcial – Encontrar a la Princesa

```
package parcial.juego;
...
public class JuegoTest {
public static void main(String[] args) {
    Personaje p0 = new Personaje("Scooby", "Animal");
    Personaje p1 = new Personaje("Cenicienta", "Princesa");
    Personaje p2 = new Personaje("Rojo", "Dragon");
    Personaje p3 = new Personaje("Pluto", "Animal");
    Personaje p4 = new Personaje("Negro", "Dragon");
    Personaje p5 = new Personaje("La Bella", "Princesa");
    Personaje p6 = new Personaje("Tweety", "Animal");
    Personaje p7 = new Personaje("Cid", "Animal");
    Personaje p8 = new Personaje("Coyote", "Animal");
    ArbolGeneral<Personaje> a1 = new ArbolGeneral<Personaje>(p0);
    ArbolGeneral<Personaje> a21 = new ArbolGeneral<Personaje>(p1);
    ArbolGeneral<Personaje> a22 = new ArbolGeneral<Personaje>(p2);
    ArbolGeneral<Personaje> a23 = new ArbolGeneral<Personaje>(p3);
    ListaGenerica<ArbolGeneral<Personaje>> hijosa2 = new ListaEnlazadaGenerica<ArbolGeneral<Personaje>>();
    hijosa2.agregar(a21, hijosa2.tamano());
    hijosa2.agregar(a22, hijosa2.tamano());
    hijosa2.agregar(a23, hijosa2.tamano());
    ArbolGeneral<Personaje> a2 = new ArbolGeneral<Personaje>(p4, hijosa2);
    . . .
    ArbolGeneral<Personaje> a = new ArbolGeneral<Personaje>(p8, hijos);
    Juego juego = new Juego();
    juego.encontrarPrincesa(a);
}
}
```

