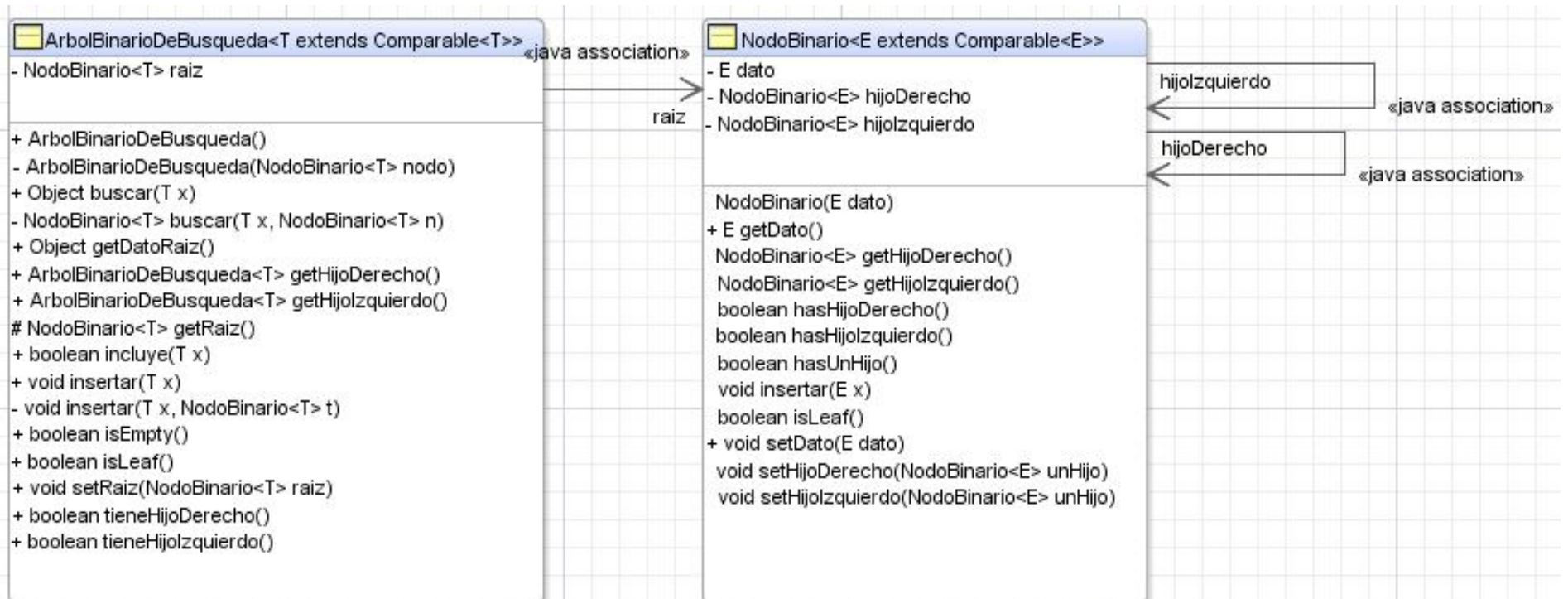


Arboles Binarios de Búsqueda con tipos genéricos y objetos comparables

Arboles Binarios de Búsqueda

Estructura



<T extends Comparable<T>> un objeto de tipo **T** que implemente la interface **Comparable<T>**

<E extends Comparable<E>> un objeto de tipo **E** que implemente la interface **Comparable<E>**

Se utilizó E para mostrar que son definiciones independientes pero podria haberse usado el mismo nombre de variable en ambas definiciones

Arboles Binarios de Búsqueda- Estructura

```
package ayed2010;
public class ArbolBinarioDeBusqueda<T extends Comparable<T>>{

    private NodoBinario<T> raiz;

    public ArbolBinarioDeBusqueda() {
        this.setRaiz(null);
    }
    private ArbolBinarioDeBusqueda(NodoBinario<T> nodo) {
        this.setRaiz(nodo);
    }
    NodoBinario<T> getRaiz() {
        return this.raiz;
    }
    public void setRaiz(NodoBinario<T> raiz) {
        this.raiz = raiz;
    }
    public T getDatoRaiz() {
        return (this.getRaiz().getDato());
    }
    public ArbolBinarioDeBusqueda<T> getHijoIzquierdo() {
        return
            new ArbolBinarioDeBusqueda<T>(this.getRaiz().getHijoIzquierdo());
    }
    public ArbolBinarioDeBusqueda<T> getHijoDerecho() {
        return
            new ArbolBinarioDeBusqueda<T>(this.getRaiz().getHijoDerecho());
    }
    . . .
}
```

```
package ayed2010;
public class NodoBinario<E extends Comparable<E>> {

    private E dato;
    private NodoBinario<E> hijoIzquierdo;
    private NodoBinario<E> hijoDerecho;

    NodoBinario(E dato) {
        this.dato = dato;
    }
    public E getDato() {
        return dato;
    }
    public void setDato(E dato) {
        this.dato = dato;
    }
    NodoBinario<E> getHijoIzquierdo() {
        return hijoIzquierdo;
    }
    NodoBinario<E> getHijoDerecho() {
        return hijoDerecho;
    }

    void setHijoIzquierdo(NodoBinario<E> unHijo) {
        hijoIzquierdo = unHijo;
    }
    void setHijoDerecho(NodoBinario<E> unHijo) {
        hijoDerecho = unHijo;
    }
    . . .
}
```

Arboles Binarios de Búsqueda

Ejemplos de instanciación y uso de árboles binarios de búsqueda con diferentes tipos:

```
ArbolBinarioDeBusqueda<Integer> abb = new ArbolBinarioDeBusqueda<Integer>();  
abb.insertar(new Integer(2));  
abb.insertar(6);  
abb.insertar(1);  
abb.insertar(13);  
abb.insertar(5);
```

↑
Clases que
implementan la
interface Comparable
↓

```
ArbolBinarioDeBusqueda<Persona> abb = new ArbolBinarioDeBusqueda<Persona>();  
Persona p = new Persona("Paula","Gomez",16);  
abb.insertar(p);  
p = new Persona("Ana","Rios",6);  
abb.insertar(p);  
p = new Persona("Maria","Ferrer",55);  
abb.insertar(p);  
Alumno a = new Alumno("Luis", "Rios", 18); // Alumno subclase de Persona  
Abb.insertar(a);
```

Arboles Binarios de Búsqueda

Inserción de un dato (Caso 1: Weiss)

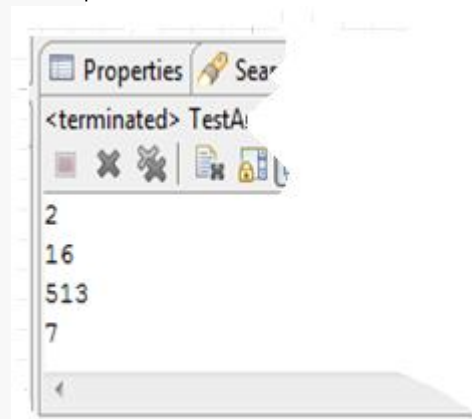
```
package ayed2010;

public class ArbolBinarioDeBusqueda
    <T extends Comparable<T>> {
    private NodoBinario<T> raiz;
    . . .

    public void insertar(T x) {
        raiz = this.insertar(x, raiz);
    }
    private NodoBinario<T> insertar(T x, NodoBinario<T> t){
        if (t == null) {
            t = new NodoBinario<T>(x);
        }
        else
            if (x.compareTo(t.getDato()) < 0)
                t.setHijoIzquierdo(this.insertar(x, t.getHijoIzquierdo()));
            else
                if (x.compareTo(t.getDato()) > 0)
                    t.setHijoDerecho(this.insertar(x, t.getHijoDerecho()));
        return t;
    }
}
```

```
package ayed2010;

public class NodoBinario<E> extends Comparable<E>> {
    private E dato;
    private NodoBinario<E> hijoIzquierdo;
    private NodoBinario<E> hijoDerecho;
    . . .
}
```



```
ArbolBinarioDeBusqueda<Integer> abb =
    new ArbolBinarioDeBusqueda<Integer>();
abb.insertar(2);
abb.insertar(6);
abb.insertar(1);
abb.insertar(13);
abb.insertar(5);
abb.insertar(7);
abb.insertar(5);
abb.mostrar();
```

Caso de uso

Arboles Binarios de Búsqueda

Inserción de un dato (Caso 2)

```
package ayed2010;

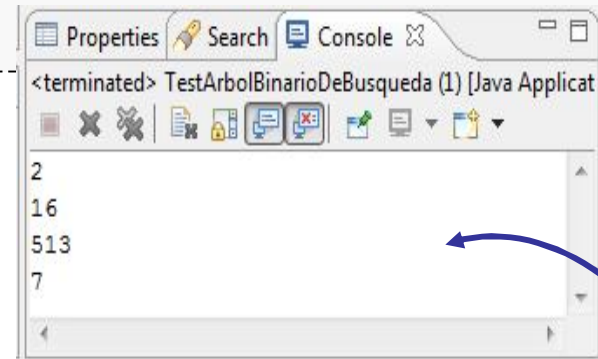
public class ArbolBinarioDeBusqueda<T
    extends Comparable<T>> {
    private NodoBinario<T> raiz;
    . . .

    public void insertar(T x) {
        if (raiz == null)
            raiz = new NodoBinario<T>(x);
        else
            this.insertar(x, raiz);
    }

    private void insertar(T x, NodoBinario<T> t) {
        if (x.compareTo(t.getDato()) < 0)
            if (t.getHijoIzquierdo()==null)
                t.setHijoIzquierdo(new NodoBinario<T>(x));
            else
                this.insertar(x, t.getHijoIzquierdo());
        else
            if (x.compareTo(t.getDato()) > 0)
                if (t.getHijoDerecho()==null)
                    t.setHijoDerecho(new NodoBinario<T>(x));
                else
                    this.insertar(x, t.getHijoDerecho());
            }
    }
}
```

```
package ayed2010;

public class NodoBinario<E extends Comparable<E>>
{
    private E dato;
    private NodoBinario<E> hijoIzquierdo;
    private NodoBinario<E> hijoDerecho;
    . . .
}
```



```
ArbolBinarioDeBusqueda<Integer> abb =
    new ArbolBinarioDeBusqueda<Integer>();
abb.insertar(2);
abb.insertar(6);
abb.insertar(1);
abb.insertar(13);
abb.insertar(5);
abb.insertar(7);
abb.insertar(5);
abb.mostrar();
```

Caso de uso

Arboles Binarios de Búsqueda

Inserción de un dato (Caso 3): Delegando a la clase NodoBinario

```
package ayed2010;

public class ArbolBinarioDeBusqueda<T>
    extends Comparable<T> {
    private NodoBinario<T> raiz;
    . . .

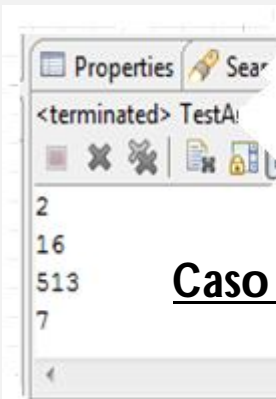
    public void insertar(T x) {
        if (raiz == null)
            raiz = new NodoBinario<T>(x);
        else
            raiz.insertar(x);
    }
}
```

```
package ayed2010;

public class NodoBinario<E extends Comparable<E>> {
    private E dato;
    private NodoBinario<E> hijoIzquierdo;
    private NodoBinario<E> hijoDerecho;
    . . .

    void insertar(E x) {
        if (x.compareTo(this.getDato()) < 0)
            if (this.getHijoIzquierdo() == null)
                this.setHijoIzquierdo(new NodoBinario<E>(x));
            else
                this.getHijoIzquierdo().insertar(x);
        else
            if (x.compareTo(this.getDato()) > 0)
                if (this.getHijoDerecho() == null)
                    this.setHijoDerecho(new NodoBinario<E>(x));
                else
                    this.getHijoDerecho().insertar(x);
    }
}
```

```
ArbolBinarioDeBusqueda<Integer> abb =
    new ArbolBinarioDeBusqueda<>(Integer);
abb.insertar(2);
abb.insertar(6);
abb.insertar(1);
abb.insertar(13);
abb.insertar(5);
abb.insertar(7);
abb.insertar(5);
abb.mostrar();
```



Caso de uso

Arboles Binarios de Búsqueda

Inserción de un dato (caso 4: sin recursión)

```
package ayed2010;

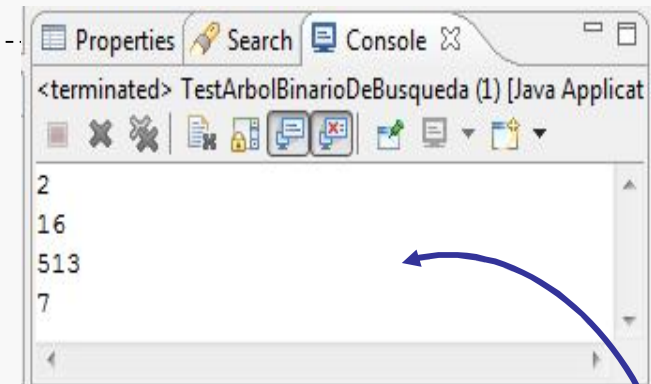
public class ArbolBinarioDeBusqueda<T>
    extends Comparable<T>> {
    private NodoBinario<T> raiz;
    . . .
    public void insertar(T x) {
        NodoBinario<T> act = raiz, ant = null;

        if (raiz == null)
            raiz = new NodoBinario<T>(x);
        else {
            while (act != null) {
                ant = act;
                if (x.compareTo(act.getDato()) < 0)
                    act = act.getHijoIzquierdo();
                else if (x.compareTo(act.getDato()) > 0)
                    act = act.getHijoDerecho();
                else act=null;
            }

            if (x.compareTo(ant.getDato()) < 0)
                ant.setHijoIzquierdo(new NodoBinario<T>(x));
            else if (x.compareTo(ant.getDato()) > 0)
                ant.setHijoDerecho(new NodoBinario<T>(x));
        }
    }
}
```

```
package ayed2010;

public class NodoBinario<E extends Comparable<E>> {
    private E dato;
    private NodoBinario<E> hijoIzquierdo;
    private NodoBinario<E> hijoDerecho;
    . . .
}
```



```
ArbolBinarioDeBusqueda<Integer> abb =
    new ArbolBinarioDeBusqueda<Integer>();
abb.insertar(2);
abb.insertar(6);
abb.insertar(1);
abb.insertar(13);
abb.insertar(5);
abb.insertar(7);
abb.insertar(5);
abb.mostrar();
```

Caso de uso