# Actividad 4. Uso del TAD Árbol Binario de Búsqueda

## Objetivo

Entender y saber usar el TAD árbol binario de búsqueda.

## Procedimiento

1.- Leer y comprender los apuntes sobre árboles binarios de búsqueda que están disponibles en **Tema**, sección **Documentos e Ligazóns** / **Teoría** / *ÁrbolBinarioBúsqueda.pdf*

2.- Resolver el ejercicio que se indica en esta actividad. Para ello el alumno hará uso de horas presenciales y no presenciales.

## Evaluación

Estos contenidos serán evaluados mediante una prueba individual el 3 de noviembre.

## Tiempo estimado

4 horas

## Ejercicio

El TAD Conjunto<E> está formado por una colección de elementos junto con sus operaciones, tal que todos los elementos del conjunto son del mismo tipo, no hay elementos repetidos y no existe ningún tipo de ordenación de los elementos (es decir, no hay un primero, segundo, etc). A continuación se presenta una especificación del TAD Conjunto<E>, en forma de interface (faltaría considerar el constructor **public Conjunto()** que crea el conjunto vacío).

```
public interface Conjunto<E>
        public int cardinal();
        //Produce: el número de elementos del conjunto
        public boolean pertenece(E x);
        //Produce: cierto si x está en el conjunto y falso en otro caso
        public boolean inserta(E x);
        //Modifica: this
        //Produce: devuelve cierto si añade x al conjunto, falso en caso contrario
        public boolean elimina(E x);
        //Modifica: this
        //Produce: devuelve cierto si elimina x del conjunto, falso en caso contrario
        public E elige() throws IllegalArgumentException;
        //Produce: lanza la excepción cuando el conjunto está vacío;  en otro caso, devuelve un elemento
        del conjunto
        public Conjunto<E> union(Conjunto<E> conj);
        //Produce: devuelve un conjunto unión de los conjuntos this y conj;
```

```
        public Conjunto<E> interseccion(Conjunto<E> conj);
        //Produce: devuelve un conjunto intersección de los conjuntos this y conj;
        public Conjunto<E> diferencia(Conjunto<E> conj);
        //Produce: devuelve un conjunto diferencia de los conjuntos this y conj;
}
```

Entre las posibles estructuras de implementación de dicho TAD, una de las más eficientes es la implementación utilizando un árbol binario de búsqueda, que exige que entre los elementos exista una relación de orden total.

Se pide:

1- implementar el TAD Conjunto haciendo uso del TAD Árbol Binario de Búsqueda, disponible en *aed2.jar* y cuya interfaz se puede consultar en el anexo.

2- probar el correcto funcionamiento de la clase implementada haciendo uso del test disponible en Tema**,** sección Documentos e Ligazóns / Actividades / Test / ***ConjuntoABBTest.java***

# Anexo:

- **TAD Árbol Binario Búsqueda:**

```
public interface ArbolBusqueda <E extends Comparable<E>> {
        public boolean esVacio();
        public E raiz() throws ArbolVacioExcepcion;
        public ArbolBusqueda<E> hijoIzq() throws ArbolVacioExcepcion;
        public ArbolBusqueda<E> hijoDer() throws ArbolVacioExcepcion;
        public void insertar(E elemento);
        public void eliminar(E elemento) throws ElementoIncorrecto;
        public boolean buscar(E elemento);
}
public class ArbolBinarioBusqueda<E extends Comparable<E>> implements ArbolBusqueda<E> {

        // atributos

        public ArbolBinarioBusqueda(){

        // crea un árbol vacío;

        }

        …
}
```