



En este taller obligatorio implementaremos un tipo de datos **Conjunto** mediante Árboles Binarios de Búsqueda. Para aprobar el taller, todos los tests que les proveemos deben pasar. Recuerden subir únicamente el archivo **ABB.java** al repositorio de entrega.

**Última fecha de entrega:** domingo 25 de mayo.

## 1. Introducción

En un Árbol Binario de Búsqueda (ABB), cada nodo contiene un valor y la referencia a sus nodos descendientes izquierdo y derecho. Los ABB respetan la siguiente restricción: el valor de todos los nodo del árbol izquierdo debe ser menor al valor del nodo actual, y el valor de todos los nodo del árbol derecho debe ser mayor al valor del nodo actual. Para facilitar los algoritmos, se recomienda que los nodos también apunten a su padre.

El ABB debe soportar cualquier tipo de datos comparables (como **Integer**, **String**, **Byte**). Todos los tipos de datos comparables tienen definido el método **compareTo()**. Sean **elem1** y **elem2** dos instancias de un mismo tipo de datos comparable, luego **elem1.compareTo(elem2)** devuelve un entero: 1) mayor a 0 si **elem1 > elem2**, 2) menor a 0 si **elem1 < elem2** y 3) 0 si **elem1 = elem2**.

Para resolver el taller cuentan con tres archivos: **Conjunto.java**, **Iterador.java**, y **ABB.java**. El primero define la interfaz de una **Conjunto<T>**, mientras que el segundo define la interfaz de un iterador **Iterador<T>**. Estos dos archivos definen los métodos a implementar. El tercero corresponde a su implementación, bajo la clase **ABB<T>**, la cual debe respetar la estructura de representación de un ABB y constituye el archivo que deben completar.

## 2. Consigna

La interfaz **Conjunto.java** declara los siguientes métodos:

- **int ABB();**  
Constructor por defecto de la clase ABB. Construye un conjunto ABB vacío.
- **int cardinal();**  
Devuelve la cantidad de elementos que tiene el conjunto.
- **T minimo();**  
Devuelve el menor elemento del conjunto.
- **T maximo();**  
Devuelve el mayor elemento del conjunto.
- **void insertar(T elem);**  
Inserta un elemento en el conjunto. Si este ya existe, el conjunto no se modifica.
- **boolean pertenece(T elem);**  
Determina si un elemento pertenece al conjunto.
- **void eliminar(T elem);**  
Elimina un elemento del conjunto. Si este no existe, el conjunto no se modifica.
- **Iterador<T>iterador;**  
Devuelve un iterador de conjunto.
- **String toString();**  
Devuelve un **String** con los elementos del conjunto en orden, escritos entre llaves y separados por coma (e.g., {1, 3, 4, 6, 10}).