Práctica 3: EclEmma AVL

En esta práctica partimos de una minuciosa especificación y de una detallada información sobre su implementación. El objetivo, como siempre, es encontrar el mayor número de errores posible a partir del análisis funcional y de la información que extraigamos a partir de las medidas de cobertura de la implementación, maximizando a continuación los valores de cobertura con los diferentes criterios analizados por EclEmma.

Para ayudarnos a sistematizar la realización de las pruebas, definid métodos factoría auxiliares que construyan árboles binarios y AVLs a partir de arrays.

```
public static <T extends Comparable<? super T>> ABB<T> abbArray(T[] valores)

public static <T extends Comparable<? super T>> AVL<T> avlArray(T[] valores)
```

Para sistematizar la comprobación de los árboles resultantes, definid un método que compruebe que el árbol es ordenado, equilibrado en altura y que su recorrido en in-orden coincide con el array pasado como argumento.

```
<T> boolean equals(T[] valores, Lista<T> lista)

<T extends Comparable<? super T>> void compruebaAVL(T[] valores, ArbolBinario<T> arbol)
```

Analizad qué tipo de errores pueden producirse debido a particularidades de Java, como el uso de equals(), hashCode() o compareTo(), y añade casos de prueba para comprobar su correcto funcionamiento.