

**Lars-Åke Fredlund**

lfredlund@fi.upm.es

**Tonghong Li**

tonghong@fi.upm.es

**Manuel Carro Liñares**

mcarro@fi.upm.es

**Germán Puebla Sánchez**

german@fi.upm.es

**Pablo Nogueira**

pnogueira@fi.upm.es

Viernes 11:00-13:00

# Entrega

- ▶ La fecha límite para optar a la máxima nota es **Viernes 30 de noviembre de 2012, a las 13:00 horas**
- ▶ Los ficheros que hay que subir son `BinTree.java` y `BinTreeLeafIterator.java` (son dos ficheros)
- ▶ La entrega se hace a través de la siguiente URL:  
`http://lml.ls.fi.upm.es/~entrega`
- ▶ El paquete `binaryTrees` esta documentado con Javadoc en  
`http://babel.ls.fi.upm.es/~fred/courses/aed/binaryTrees/`
- ▶ El proyecto debe compilar sin errores, cumplir la especificación y pasar el Tester.

# Configuración

- ▶ Arrancad Eclipse.
- ▶ Cread un paquete `binaryTrees` en el proyecto `aed`, dentro de `src`.
- ▶ Aula Virtual → AED → Sesiones de laboratorio → Laboratorio 7 → `codigo_lab7.zip` (formato zip).
- ▶ Importad al paquete `binaryTrees` los fuentes que habéis descargado
- ▶ Ejecutad `Tester`. Veréis que lanza una excepción:

```
Testing insert...
```

```
The size of the tree should be 4 but is 0
```

```
Tree:
```

```
<<<empty>>>
```

```
Exception in thread "main" java.lang.Error  
at binaryTrees.Tester.doTest(Tester.java:60)  
at binaryTrees.Tester.main(Tester.java:37)
```

# Tareas para hoy

Hoy trabajaremos con arboles binarios de búsqueda. La clase `BinTree<E>` extiende la clase `LinkedBinaryTree<E>` del libro con dos métodos que debéis completar:

- ▶ `void insert(E element)` que inserta `element` en el árbol binario de búsqueda.
- ▶ `Iterator<E> leaves()` devuelve un iterador que itera sobre las hojas del árbol en orden ascendente. Concretamente hay que modificar los métodos `hasNext()` y `next()` del iterador, que se encuentra en el fichero `BinTreeLeafIterator.java`.


# Restricciones y permisos

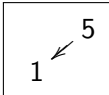
- ▶ Esta permitido añadir nuevos métodos, atributos privados, o variables locales.
- ▶ Solo esta permitido usar los siguientes métodos de la clase `LinkedBinaryTree<E>`:  
`addRoot`, `insertLeft`, `insertRight`, `hasLeft`, `hasRight`,  
`isEmpty`, `isExternal`, `isInternal`, `isRoot`,  
`left`, `right`, `root`, `parent`, `sibling`, `size`  
El uso de otros métodos de la clase queda prohibido.
- ▶ En la clase que implementa `Position` solo esta permitido llamar a `element()`.

## insert(Element e)

- ▶ El método debe insertar un elemento dentro del árbol implementado por la clase `BinTree<E>`.
- ▶ La inserción debe cumplir las siguientes propiedades:
  - ▶ El número de nodos dentro el árbol es igual al número de inserciones de elementos, y cada nodo almacena un único elemento.
  - ▶ El elemento de un nodo es mayor que todos los elementos de su subárbol izquierdo, y menor o igual que todos los elementos de su subárbol derecho.
  - ▶ Se comparan dos elementos usando el comparador `cmp` que se encuentra como atributo dentro la clase `BinTree<E>`.

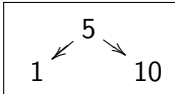
## insert(Element e): ejemplos

► insert(5)  $\Rightarrow$  

► insert(1)  $\Rightarrow$  

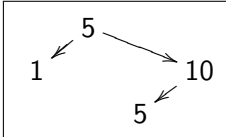
```
graph TD; 5 --> 1
```

*añade un nodo nuevo 1 a la izquierda porque  $1 < 5$*

► insert(10)  $\Rightarrow$  

```
graph TD; 5 --> 1; 5 --> 10
```

*añade un nodo nuevo 10 a la derecha porque  $5 \leq 10$*

► insert(5)  $\Rightarrow$  

```
graph TD; 5 --> 1; 5 --> 10; 10 --> 5
```


*ir a la derecha porque  $5 \leq 5$ , y después añade un nodo nuevo 5 a la izquierda porque  $5 < 10$*

## BinTreeLeafIterator<E>

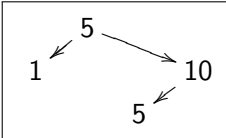
- ▶ Se pide completar los métodos boolean `hasNext()` y `E next()` de la clase `BinTreeLeafIterator<E>`, que implementa un iterador sobre las *hojas* del árbol.
- ▶ El iterador debe devolver las hojas en orden izquierda a derecha, es decir, de menores valores a mayores.
- ▶ Se pide que la implementación del iterador sea *eficiente*. Es decir, en la implementación del `next` o `hasNext` no se debería recorrer todo el árbol sino solo la parte del árbol que es necesaria para encontrar la siguiente hoja.
- ▶ Para conseguir una implementación eficiente queda prohibido la solución de recorrer todo el árbol y guardar los elementos en un `PositionList` o una estructura de datos parecida.



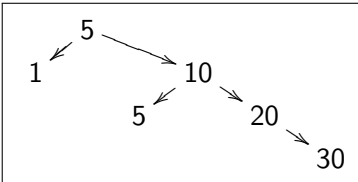
## BinTreeLeafIterator<E>: ejemplos

- ▶ Dado el árbol 

el iterador debería devolver el elemento 5.

- ▶ Dado el árbol 

el iterador debería devolver los elementos 1 y 5 en orden.

- ▶ Dado el árbol 

el iterador debería devolver los elementos 1, 5 y 30 en orden.