Apellidos:_____

Nombre: _____

DNI: _____ **UO:** _____

Universidad de Oviedo
Escuela de Ingeniería Informática
Estructura de Datos
Grupo L5
22/11/2016 Evaluación continua Árboles

Instrucciones

1. **Incluye tus datos personales en todas las hojas, incluidas las de borrador.**
2. **Analiza el examen detenidamente. Se atenderán dudas solo durante los primeros 5 minutos del examen.**
3. **Está terminantemente prohibido acceder a Internet durante el examen.**
4. **Se debe utilizar la versión de los algoritmos y estructuras de datos vistas en clase de laboratorio.**
5. **Antes de abandonar este recinto deberás entregar el examen completo (incluso si está en blanco).**
6. **Tras el examen se deberán subir 2 ficheros (por separado) a la tarea de entrega del examen:**
   a. **El PROYECTO exportado a fichero zip El PROYECTO exportado a fichero zip al Campus Virtual Nombre del fichero exportado: : EX-Arboles-Apellido1Apellido2NombreUOxxxx.zip**
   b. **El fichero de la JUnit del examen: ExamenArboles_Apellido1Apellido2NombreUOxxxx.java**

Implementar una JUnit llamada ***ExamenArboles_Apellido1Apellido2NombreUOxxxx*** en la que a un objeto ***AVLTree<Double>*** se le deben añadir (add) y borrar (remove) nodos de forma que se produzcan la siguiente serie de **rotaciones** con los **desequilibrios** indicados, **en las operaciones** indicadas, en el **orden** indicado y con las **restricciones** indicadas:

Rotaciones:

1ª	`remove(  ).singleLeftRotation[-2][-1]`	6ª	`remove(  ).doubleRightRotation[2][-1]`
2ª	`add(  ).doubleLeftRotation[-2][1]`	7ª	`add(  ).singleRightRotation[2][1]`
3ª	`remove(  ).singleLeftRotation[-2][0]`	8ª	`remove(  ).singleRightRotation[2][0]`
4ª	`add(  ).singleLeftRotation[-2][-1]`	9ª	`add(  ).doubleRightRotation[2][-1]`
5ª	`remove(  ).doubleLeftRotation[-2][1]`	10ª	`remove(  ).singleRightRotation[2][1]`

Restricciones:

- Las 5 primeras operaciones sobre el árbol recién creado serán "**add**" y **NO PUEDEN** producir rotaciones.
- A partir de las 5 primeras operaciones **add** anteriores, el árbol debe tener **SIEMPRE** al menos 5 nodos.
- Entre las rotaciones indicadas, se pueden realizar **cualquier** número de añadidos y/o borrados, pero **que EN NINGÚN CASO, deben producir ROTACIONES**.

Antes de entregar:

- Rellenar los parámetros que producen las rotaciones, y que se pasan a los add y remove (en la tabla de arriba)
- En el código de la JUnit:
  Tenéis que rellenar el array con los mismos nodos que producen las rotaciones:
  Double[] nodosQueHacenRotar=new Double[]{// Estos valores son un ejemplo…
  /* rotaciones del 1 al 5 */ 30d,20d,10d,40d,90d,
  /* rotaciones del 6 al 10 */ 50d,60d,70d,80d,100d};

IMPRESCINDIBLE:

- **TODOS** los métodos de las clases BSTNode, BSTree, AVLNode, AVLTree deben ser los indicados por el profesor en las clases prácticas.
- Los métodos **toString()** de las clases BSTNode, BSTree, AVLNode, deben ser los proporcionados por el profesor, y el del BSTree será heredado por su clase derivada **AVLTree**, puesto que se utilizarán para verificar el correcto funcionamiento de las clases.
- Las invocaciones a métodos de las clases BSTree y AVLTree en las pruebas deben cumplir lo indicado anteriormente.

Duración: 45 minutos