Actividad 2. Uso del TAD Árbol Binario

Objetivo

Saber usar el TAD árbol binario para resolver problemas de programación.

Procedimiento

1. Leer y comprender los apuntes de árboles binarios que están disponibles en **Tema**, sección **Documentos e Ligazóns** / **Teoría** / *ÁrbolBinario.pdf*
2. Emplear técnicas de aprendizaje colaborativo para resolver los *ejercicios de clase* que se indican en esta actividad, utilizando el lenguaje java. En concreto se utilizará la **técnica del rompecabezas.**
    a. Se formarán grupos base de 6 personas.
    b. Cada miembro del grupo será responsable de la resolución de un ejercicio (trabajo individual no presencial).
    c. En clase se reunirán los expertos de cada ejercicio para puesta en común.
    d. Después cada persona volverá a su grupo base para explicarles a cada compañero lo que aprendieron.
3. Los *otros ejercicios propuestos* se trabajarán de manera no presencial. Para probar su correcto funcionamiento se puede hacer uso del test disponible en Tema, sección Documentos e Ligazóns / Actividades / Test / **Actividad2Test.java**
4. Para resolver todos los ejercicios es necesario conocer la interfaz del TAD árbol binario, disponible en el anexo.

Evaluación

El trabajo colaborativo será evaluado de manera grupal el día 29 de septiembre.

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

Tiempo estimado

6 horas

Ejercicios para resolver en clase

1. Escribe un método que dado un árbol binario devuelva verdadero si el árbol es **completo** y falso en otro caso. Un árbol binario es completo si todos sus nodos tienen dos descendientes, excepto las hojas.

2. Escribe un método que dados dos árboles binarios A y B, determine si son **idénticos** o no.
3. Escribe un método que determine si un elemento **está** en un árbol binario.
4. Escribe un método que **cuente** el número de nodos que están en el **nivel** *n* de un árbol binario.
5. Escribe un método que dado un árbol binario A, cree un árbol binario B igual que A pero **sin las hojas**.
6. Escribe un método que calcule la **altura** de un árbol binario.

Otros ejercicios propuestos

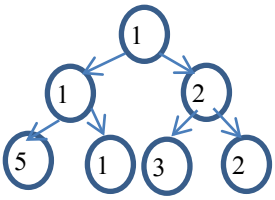
7. Escribe un método que dados dos árboles binarios A y B, determine si tiene la **misma forma** o no.

   public static <E> boolean mismaForma (ArbolBinario<E> a, ArbolBinario<E> b)

8. Un **árbol de selección** es un árbol binario donde cada nodo representa al menor de sus dos hijos, excepto las hojas. Construir un método que, dado un árbol binario, indique si es o no un árbol de selección.

   public static <E extends Comparable<E>> boolean arbolSeleccion (ArbolBinario<E> arbol)

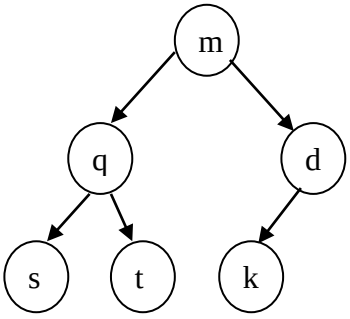
   Ejemplo:



9. Escribe un método booleano que dados un árbol binario y un camino expresado en forma de String determine si existe dicho **camino** en el árbol, teniendo en cuenta que el camino debe comenzar necesariamente en la raíz.

   public static <E> boolean esCamino(ArbolBinario<E> arbol, String camino)

   Por ejemplo, para el árbol que sigue existen los caminos m-q-t y m-d, pero no existen los caminos r-q-t ni d-k.



10. Escribe un método que dado un árbol binario y un nivel n del árbol, realice una **copia** del árbol hasta dicho nivel.

    public static <E> ArbolBinario<E> copiaHastaNivel(ArbolBinario<E> a, int nivel)

11. Cada nodo de un árbol binario A representa una letra alfabética. La concatenación de las mismas, en cada camino que va desde la raíz a una hoja representa una **palabra**. Realizar un procedimiento que visualice todas las palabras almacenadas en un árbol binario A.

```
public static <E> void visualizarPalabras(ArbolBinario<E> a) {
    String palabra = "";
    visualizarPalabras (a, palabra);
}
private static <E> void visualizarPalabras(ArbolBinario<E> a, String palabra)
```

12. Escribe un método que dado un árbol binario y un elemento devuelva el **padre** de dicho elemento, suponiendo que no hay elementos repetidos.

    public static <E> E padre(ArbolBinario<E> a, E elemento)

13. Un elemento de un árbol se dirá de nivel k si aparece en el árbol a distancia k de la raíz. Escribe un método que determine si un elemento está a **distancia k**.

    public static <E> boolean nivelK (ArbolBinario<E> a, E elem, int k)

Anexo:

- **TAD Árbol Binario:**

```
public interface ArbolBinario<E>{
   public boolean esVacio();
   public E raiz() throws ArbolVacioExcepcion;
   public ArbolBinario<E> hijoIzq()throws ArbolVacioExcepcion;
   public ArbolBinario<E> hijoDer()throws ArbolVacioExcepcion;
   public boolean esta(E elemento);
   public void setRaiz(E elemRaiz) throws ArbolVacioExcepcion;
   public void setHijoIzq(ArbolBinario<E> hi) throws ArbolVacioExcepcion, NullPointerException;
   public void setHijoDer(ArbolBinario<E> hd) throws ArbolVacioExcepcion, NullPointerException;
   public void suprimir();
}

public class EnlazadoArbolBinario<E> implements ArbolBinario<E>{
    public EnlazadoArbolBinario(){…}
    public EnlazadoArbolBinario(E elemento, EnlazadoArbolBinario<E> hi, EnlazadoArbolBinario<E>
hd) {…}
    …
}
```