Actividad 3. Implementación y uso del TAD Árbol Binario

Objetivo

Implementar el TAD Árbol Binario haciendo uso de una estructura enlazada y posteriormente utilizar dicha implementación para resolver ejercicios de uso del TAD Árbol Binario.

Procedimiento

1. Ver el video o leer la presentación sobre árboles binarios que están disponibles en Moodle, Tema 1/ Sección 1.2 TAD Árbol Binario/ Recursos didácticos.
2. Haciendo uso de una representación enlazada, implementar el TAD Árbol Binario.
3. Para probar el correcto funcionamiento de la implementación se puede hacer uso del test disponible en Moodle, Tema 1/ Sección 1.2 TAD Árbol Binario/ Actividades Grupo Reducido: *EnlazadoArbolBinarioTest.java*.
4. Haciendo uso de la implementación realizada, resolver los ejercicios de árboles binarios que se proponen.
5. Para probar el correcto funcionamiento de los ejercicios se puede hacer uso del test disponible en Moodle, Tema 1/ Sección 1.2 TAD Árbol Binario/ Actividades Grupo Reducido: *SolActividad3Test.java*.

Evaluación

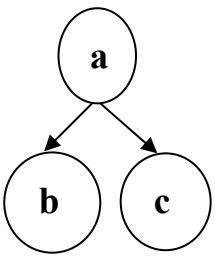
Estos contenidos serán evaluados mediante una prueba individual el 29 de octubre de 2024.

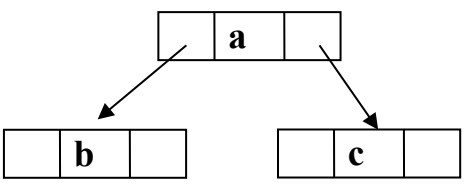
Tiempo estimado

5 horas

Ejercicios

1.- Ejercicio de implementación: Una de las formas más habituales de implementar un árbol binario es mediante una estructura enlazada, donde cada nodo (NodoBinario<E>) está compuesto de un dato genérico y dos enlaces, uno hacia el NodoBinario hijo izquierdo y otro hacia el NodoBinario hijo derecho.





Haciendo uso de la clase NodoBinario<E>, se pide crear un proyecto que implemente el TAD Árbol Binario (disponible en el Anexo).

2.- Ejercicios de uso:

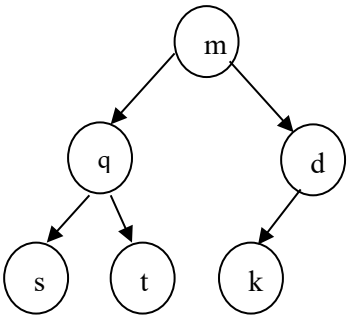
a) Escribe un método que dados los recorridos en preoden e inorden de un árbol binario, reconstruya el árbol. Suponemos que los recorridos son String y que no hay caracteres repetidos.

```
public static ArbolBinario<Character> construir(String preord, String inord)
```

b) 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 boolean esCamino(ArbolBinario<Character> 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.



c) Escribe un método que dados un árbol binario y un elemento devuelva el **padre** de dicho elemento en el árbol, suponiendo que no hay elementos repetidos. Si el elemento no está en el árbol o es el elemento raíz, el método devuelve null.

```
public static <E> E getPadre (ArbolBinario<E> a, E elemento)
```

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, NullPointerException;
   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, ArbolBinario<E> hi, ArbolBinario<E> hd) {…}
    …
}
```