Ejercicios de Pseudo TDA parciales - 1

Ejercicio #1

Tipo Alumno
- CI : int
- Nombre : String
- Apellido : String
- Carrera : String

Descripción de alto nivel

El método armarIndiceCarrera (UnaCarrera) recorre el arbol original (indexado por el CI) y devuelve un nuevo árbol de búsqueda que solo contiene las referencias a los Alumnos cuya Propiedad "Carrera" coincida con una Carrera La clave de ordenación en el árbol será por apellido, de modo que un Recorrido en orden por este, devuelve la lista de Alumnos de una carrera ordenados alfabeticamente.

Pre Condiciones :

1. El Arbol recibido es un Arbol binario de busqueda bien construido, cuyos nodos contienen :
   - → Clave : Cadena que es el CI del Alumno.
   - → dato : Referencia no nula a un objeto Alumno que contiene su CI, Nombre, apellido y carrera.

2. Una Carrera es una cadena No Vacía.

Pseudo

```
Tipo Alumno
    CI : Int
    Nombre : Str
    Apellido : Str
    Carrera : Str

Constructor Alumno ( CI, nom , ape , car )
    CI ← CI
    Nombre ← nom
    Apellido ← ape
    Carrera ← car
```

```
Clase ElementoAB                    * DE AQUI EN MAS SON COSAS DE ElementoAB
    clave : Str
    dato : Alumno
    izquierdo, derecho : ElementoAB     // Son de tipo ElementoAB

Constructor ElementoAB
    this.clave ← clave
    this.dato ← dato
    izquierdo ← null
    derecho ← null
```

```
método insertar ( elemento : ElementoAB )
    Si: elemento.clave < this.clave
      { Si izquiendo == null
          { izquiendo = elemento }
        Sino { izquiendo.insertar ( elemento ) }
    Sino si elemento.clave > this.clave
      { Si derecho == null
          { derecho = elemento }
        Sino { derecho.insertar ( elemento )
    Sino
      { Tinar excepción : " Elemento repetido " }
Fin método

método inOrden ( accion )   // acción es lo que se
    Si izquiendo != null    // va a hacer. Ej: imprimir datos.
      { izquiendo.inorden (accion) }
    accion (this)
    Si derecho != null
      { derecho.inOrden (accion) }
Fin método
```

* Terminar ElementoAB

```
Clase ArbolBB
    Raiz ← ElementoAB

Constructor ( ) :
    Raiz ← null

método insertar ( Clave , dato: Alumno)   // El dato es Alumno
    Nodete ← ElementoAB ( Clave , dato )
    Si Raiz == null
      { Raiz ← Nodete }
    Sino
       { Raiz . insertar ( Nodete ) }
    Fin sí
Fin método

método inOrden (accion)
    Si Raiz != null
      { Raiz . inOrden (accion) }
Fin método                                  método Java

método armarIndice Carrera ( una Carrera : Str ) → ArbolBB
    Arbolito ← ArbolBB ( )

    Procedimiento Filtrar ( nodo : ElementoAB )
    Si !nodo == null
       { Retornar }          // Si no hay NADA, me voy
      Filtrar ( nodo . izquierdo )
      Alumno ← nodo . dato
      Si Alumno . Carrera == una Carrera
         { Arbolito . insertar (Alumno . Apellido , Alumno )
       Fin si
      Filtrar (nodo . derecho )
Fin Procedimiento
```

Sigue el método

Filtrar (this. Raíz)
Retornar Arbolito
Fin método

Post Condiciones

- Arbolito se devuelve correctamente.
- Arbolito. Raíz puede ser null (Caso de que no haya estudiantes en una carrera dada).

- No hay duplicados

- Los Alumnos insertados en el Arbol quedan ordenados por su apellido y no su CI.