[illegible] de Software

(Universidad [illegible])

Nombre:	
	LU:
Cantidad de hojas entregadas (sin enunciado):	

Observaciones generales:

- REALICE LOS EJERCICIOS EN HOJAS SEPARADAS.
- Lea todo el ejercicio antes de comenzar a desarrollarlo.
- Numere y ponga su nombre a todas las hojas. Indique cuántas hojas entrega.
- Si considera que existe alguna ambigüedad en el enunciado, tome una decisión, documéntela y continúe resolviendo el ejercicio correspondiente.

Referencias: [GT] Michael Goodrich & Roberto Tamassia. *Data Structures and Algorithms in Java*, Fourth Edition, John Wiley and Sons, 2006.

Ejercicio 1

a) Dado el siguiente diagrama de clases:

En Java, **defina todas las interfaces necesarias y complete las clases** *Nodo* y *Lista* (u otra que considere necesaria) para implementar una lista simplemente enlazada (no circular, sin centinelas y con posición directa). Los elementos de la lista son de tipo genérico E.

- Para la clase *Nodo*, defina los atributos, implemente el o los constructores y escriba las signaturas pero **no** implemente el cuerpo de las consultas y comandos. La interfaz *Position* corresponde a la dada en [GT].
- Para la clase *Lista*, defina los atributos e implemente el o los constructores. No defina [illegible] ni implemente el cuerpo de consultas ni comandos. La interfaz *PositionList* corresponde a la dada en clase y **no es iterable**.
- **Implemente** [illegible].

b) Implemente la operación *remove(p)* del TDA Lista que elimina el elemento en la posición p de la lista [illegible].

c) [illegible] (INVERTIR) [illegible] elementos. Para realizar esta operación deberá utilizar [illegible] las operaciones del TDA Lista. [illegible] **deberá implementar los métodos auxiliares que defina.**

d) Justificando adecuadamente, indique el orden del tiempo de ejecución del procedimiento [illegible] (c) asumiendo que las operaciones del tipo [illegible] son de tiempo constante.

PositionList<E>	Queue<E>	Stack<E>	Tree<E>	
size()	size()	size()	size()	
isEmpty()	isEmpty()	isEmpty()	isEmpty()	isExternal(p)
first()	front()	top()	iterator()	isRoot(p)
last()	enqueue(e)	push(e)	positions()	createRoot(e)
next(p)	dequeue()	pop()	replace(p, e)	addFirstChild(p,e)
prev(p)			root()	addLastChild(p,e)
addFirst(e)			parent(p)	addBefore(p, rb,e)
addLast(e)			children(p)	addAfter(p,lb,e)
addAfter(p, e)			isInternal(p)	removeNode(p)
addBefore(p, e)				
remove(p)				

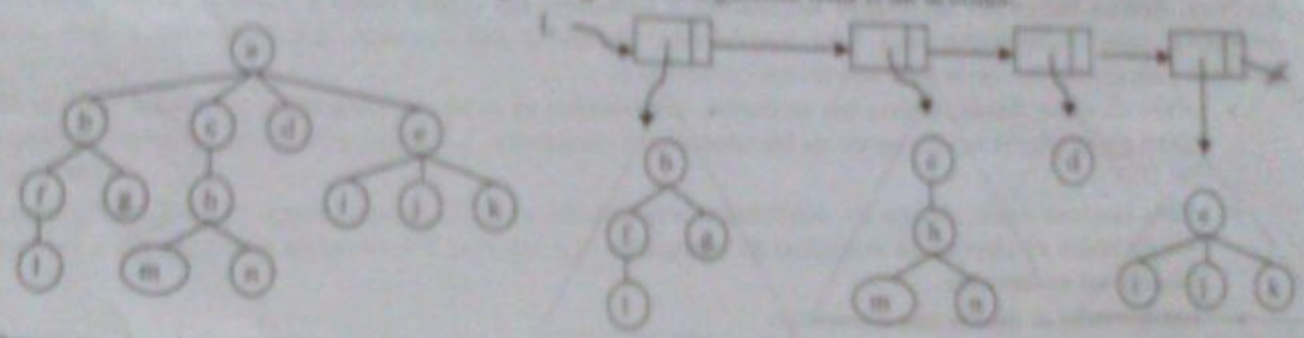
Sigue atrás...

Suponiendo que posee la implementación del TDA *Lista* presentado en [GT] y las interfaces *Posicion* y *Tree* de [GT], complete las clases *TNodo* y *Arbol* (la otra que considere necesaria) para implementar un árbol general, cuyos rótulos son de tipo genérico E, utilizando la representación de colección de hijos y referencia al padre.

- Para la clase *TNodo*, defina los atributos, implemente el constructor y escriba las signaturas pero **no** implemente el cuerpo de las consultas y comandos.
- Para la clase *Arbol*, defina los atributos e implemente solamente los constructores. No defina signaturas de consultas ni comandos ni implemente su cuerpo a menos que los vaya a utilizar para resolver el inciso (b).

De considerarlo necesario, puede asumir que la lista es iterable.

b) Agregue una operación a la clase *Arbol* definida en el inciso (a) que retorne una lista de árboles formada por la copia de aquellos subárboles de T cuya raíz sea un hijo del nodo raíz de T, donde T es el árbol que recibe el mensaje. Esta operación ***no debe*** modificar el árbol que recibe el mensaje.
Ejemplo: para un árbol T como el que sigue se genera la siguiente lista L de árboles:



Para este ejercicio no tiene acceso a la estructura de la clase lista, pero sí puede utilizarla como un TDA. Recuerde que está agregando un método a la clase Árbol, por lo tanto tiene total acceso a su estructura, si utiliza otras operaciones del TDA Arbol **debe implementarlas, como así también los métodos auxiliares utilizados** (pero no es necesario implementar las operaciones usadas del TDA lista).

c) Justificando adecuadamente, indique el orden del tiempo de ejecución del procedimiento solución al inciso (b) asumiendo que las operaciones del tipo E son de tiempo constante, e indicando apropiadamente el orden del tiempo de ejecución de las operaciones utilizadas del TDA Lista. Para ello justificar indicando desde su punto de vista cuál es la estructura de datos subyacente al TDA Lista.

Ejercicio 3:

a) Defina en Java, una interfaz para el TDA *Pila* donde los elementos son de tipo *Genérico E*.

b) Defina en Java todas las estructuras de datos (clases y atributos) necesarias para implementar pilas utilizando un arreglo. Implemente además el constructor de la Pila y las operaciones push(e) y pop() de acuerdo a la estructura propuesta.

c) Supo niendo que posee la implementación de los TDA Pila y TDACola (como se definen en [GT]) implemente, solamente en términos de las operaciones de estos, un método que recibe una cola de caracteres Q y un caracter X y determina si Q respeta el siguiente formato: AXAA', donde A es una cadena de caracteres formada por 1 o más caracteres y A' corresponde al inverso de A. Puede asumir que el carácter X pasado por parámetro no está presente en A.

d) Justificando adecuadamente, indique el orden del tiempo de ejecución del procedimiento solución al inciso c). Para ello justificar indicando desde su punto de vista cuál es la estructura de datos subyacente al TDA Cola.