Algoritmos y Estructuras de Datos

Recursion III

Lars-Åke Fredlund

lfredlund@fi.upm.es

Universidad Politécnica de Madrid

Curso 2021/2022

Recursion: pregunta 1

Se pide: Implementar **de forma recursiva** en Java el método:

```
static <E> int countApariciones(PositionList<E> list, E elem
    )
```

que recibe como argumento una lista `list` y un elemento `elem`. El método `countApariciones` debe devolver el número de apariciones de `elem` en la lista `list` (y 0 en caso de que `elem` no esté en `list`). El parámetro `list` podrá ser `null` (en cuyo caso contiene cero apariciones de cualquier elemento) y podrá contener elementos `null`. Igualmente, el parámetro `elem` también podrá ser `null`. El uso de bucles (`while`, `for`, `do-while` o `for-each`) así como de iteradores NO está permitido. La implementación debe realizarse mediante un **método auxiliar** que sea **recursivo**. No está permitido modificar el contenido de `list`.
Por ejemplo, dado `list = [1,2,3,null,2,4,5,null]`, la llamada `countApariciones(list,18)` debe devolver 0; `countApariciones(list,2)` debe devolver 2; `countApariciones(list,null)` debe devolver 2.

Recursion: pregunta 2

Recursion: pregunta 2

Se pide: Implementar de forma **recursiva** en Java el método:

```
static <E>  PositionList<E> quitarIguales (PositionList<E>
    list, E elem)
```

que recibe como parámetro una `PositionList` y un elemento `elem` y devuelve una **nueva** lista con los elementos contenidos en `list` (en el mismo orden) que sean distintos de `elem`. La lista `list` podrá ser `null`, en cuyo caso, el método debe devolver `null`. El uso de bucles (while, for o do-while, for-each) así como de iteradores NO está permitido, se debe implementar mediante un **método auxiliar** que sea **recursivo**. Si `elem` no está en la lista se devolverá una copia de `list`. Se dispone de la clase `NodePositionList<E>`, que implementa el interfaz `PositionList<E>` y que dispone de un constructor sin parámetros para crear una lista vacía.
Por ejemplo, dada la lista `list = [1,2,3,4,5]`, la llamada a `quitarIguales(list,3)` devolverá una **nueva** lista que contendrá `[1,2,4,5]` o `quitarIguales(list,8)` devolverá una **nueva** lista que contendrá `[1,2,3,4,5]`.

Recursion: ejemplo 3

- Implementa

```
static <E> PositionList<E> eliminarRepetidos(
    PositionList<E> l)
```

  que dado una lista `l`, devuelve una lista nueva donde todos los elementos repetidos estan borrados.

- Ejemplos:

```
eliminarRepetidos([]) => []
eliminarRepetidos([1]) => [1]
eliminarRepetidos([1,1]) => [1]
eliminarRepetidos([1,2,3,1,2,3,1,1,4]) => [1,2,3,4]
```

- Se puede asumir que `l` no es null, y no contiene elementos null.