Esercitazione 7: strutture dinamiche generiche a nodi concatenati

Esercizio 1. Implementare in una classe `NodeUtil`, i seguenti metodi generici[1] in modo iterativo o ricorsivo, a scelta.

Nell'implementazione, si assuma che, per il tipo generico `T` cui si fa riferimento nei metodi, sia disponibile il metodo `equals` che ne implementa il criterio di uguaglianza (nel senso che per ogni `x` e `y` di tipo `T`, `x.equals(y)` restituisce `true` se e solo se `x` è uguale a `y`).[2]

1. `public static <T> int size(Node<T> p)`
   ritorna il numero di elementi in `p`. Per esempio, se `p` rappresenta la lista `[3, 1, 2, 1, 0]` il metodo deve ritornare `5`.

2. `public static <T> int  occurrences(Node<T> p, T x)`
   ritorna il numero di occorrenze di `x` in `p`. Per esempio, se `p` rappresenta la lista `[3, 1, 2, 1, 0]` e `x` è `1`, il metodo deve ritornare `2`.

3. `public static <T> boolean included(Node<T> p, Node<T> q)`
   ritorna `true` se tutti gli elementi nella lista `p` compaiono nello stesso ordine anche nella lista `q`, e `false` altrimenti. Per esempio, se `p` e `q` rappresentano rispettivamente le liste `[1, 2, 3]` e `[0, 1, 2, 2, 0, 3, 4]` il metodo deve ritornare `true`.

4. `public static <T> Node<T> reverse(Node<T> p)` ☠
   ritorna la lista che contiene gli stessi elementi di `p`, ma in ordine inverso. Per esempio, se `p` rappresenta la lista `[1, 2, 3]` il metodo deve ritornare la lista `[3, 2, 1]`. Attenzione: la versione ricorsiva è più difficile di quella iterativa.

5. `public static <T> void printList(Node<T> p)`
   stampa la lista.[3]

6. un metodo che accetta in input un nodo che rappresenta una lista di liste di oggetti di un tipo generico `T` non noto a priori e restituisce la lista delle cardinalità delle singole liste che occorrono nella lista di liste. Ad esempio, se il nodo in input rappresenta la lista `[[ ][ 4 1 2 1 ][ 3 4 1 3 2 3 1 3 ]]`, il metodo deve restituire la lista `[ 0 4 8 ]`.

   Per questo metodo non viene specificato il propotipo, in quanto l'individuazione del prototipo corretto è parte dell'esercizio. ☠

Tali metodi **non devono modificare** in alcun modo le liste prese in ingresso come parametri. In particolare, il metodo `reverse` deve ritornare una **nuova** lista.

Si implementi, inoltre, una classe `TestNodeUtil` che testa tutte le funzionalità di `NodeUtil` su liste di `Integer`, di `Double`, di `String` e di oggetti di tipo `Person` (la classe `Person` è data e il suo codice è disponibile Moodle)

[1] Alcuni di essi sono la generalizzazione di metodi implementati in un'esercitazione precedente.

[2] Java fornisce un'implementazione di default per il metodo `equals`, la quale, se necessario, può essere opportunamente ridefinita(sovrascritta) in ogni classe. Il meccanismo di ridefinizione è il cosiddetto *overriding dei metodi* che vedrete in seguito. Qui possiamo accontentarci di assumere che per il tipo `T` sia stata in qualche modo specificata l'opportuna implementazione per il metodo `equals`, il che consente l'uso di tale metodo, quando necessario.

[3] Si assuma che per ogni oggetto `x` di tipo `T`, le istruzioni `System.out.print(x)` e `System.out.println(x)` stampino adeguatamente l'oggetto `x`.