Sistemi di Elaborazione 2
Esercizi – Quinta lezione di laboratorio

Esercizio n. 1
Una lista è una struttura dati che implementa una sequenza ordinata di oggetti, permettendo di inserire e rimuovere elementi in un qualsiasi punto della sequenza. Una possibile interfaccia per una lista in Java è la seguente:

```
package it.unipd.stat.se2.lista;

import java.lang.IndexOutOfBoundsException;

public interface GenericList<E> {
	void add(int index, E element) throws IndexOutOfBoundsException;
	E get(int index) throws IndexOutOfBoundsException;
	E remove(int index) throws IndexOutOfBoundsException;
	int size();
}
```

Il metodo `add` permette di aggiungere un elemento alla lista, nella posizione `index`. Gli eventuali oggetti agli indici successivi vanno spostati avanti di una posizione. Il metodo `get` restituisce l'elemento all'indice `index`. Il metodo `remove` restituisce e rimuove l'elemento all'indice `index`, e "riempe il buco" lasciato libero. Le implementazioni dei metodi dell'interfaccia dovranno eseguire gli opportuni controlli sull'indice. Gli indici partono da zero [il primo elemento della lista è a indice `0`, l'ultimo elemento è all'indice `size()-1`]

Si chiede di fornire due diverse implementazioni dinamiche dell'interfaccia `GenericList<E>`.

1. `GenericDoubleLinkedList<E>` è una implementazione ottenuta mediante una lista con doppi link. Utilizzare la classe di supporto `NodoDoppio` vista in classe.
2. `GenericArrayList<E>` è una implementazione ottenuta mediante un `Array`. L'array dovrà essere creato inizialmente di dimensione 1. Ogni volta che una operazione `add` trova un array pieno, si provvederà a creare un nuovo array di dimensioni doppie, copiando tutti gli elementi dal vecchio al nuovo array.

Esercizio n. 2
Effettuare un test comparativo di prestazioni sulle implementazioni `GenericDoubleLinkedList` e `GenericArrayList`. In particolare, **per ciascuna della due implementazioni**, eseguire il seguente test:

1. Creare una lista di 100.000 elementi
2. Misurare il tempo necessario ad effettuare 50.000 accessi casuali ad elementi della lista
3. Misurare il tempo necessario ad inserire 50.000 elementi, in testa alla lista (posizione 0)

Per misurare il tempo, è possibile sfruttare la funzione Java `System.currentTimeMillis()`:

```
long start = System.currentTimeMillis();
// ... Effettuare gli inserimenti
long stop = System.currentTimeMillis();
long durata = stop – start;
```

Per quanto riguarda l'accesso ad un elemento casuale, è possibile utilizzare le funzionalità della classe `java.util.Random`. La funzione `nextInt(N)` restituisce un intero casuale compreso tra 0 (incluso) e N (escluso):

```
Random rand = new Random();
for (...) {
     list.get(rand.nextInt(N));
}
```

Attenzione! Durante i periodi misurati non eseguire stampe a schermo. Tali chiamate sono molto dispendiose e falserebbero il risultato.

Inviare i risultati del test a `martino.fornasa@dei.unipd.it` entro giovedì 18 febbraio alle ore 16.00.