Sistemi di Elaborazione 2
Esercizi – Lezione di laboratorio del 10/3/2011
Sorting, searching, profiling.

Esercizio 1

Implementare la classe `MyArrays`, contenente metodi statici per la manipolazione di array contenenti oggetti qualsiasi. I metodi della classe dovranno trattare correttamente il caso in cui il riferimento all'array dovesse essere `null`, lanciando una `NullPointerException`.

La classe dovrà fornire i seguenti metodi:

1) `public static int search(Object[] a, Object key)`

La funzione 1) cerca nell'array `a` l'oggetto `key`, utilizzando l'algoritmo di ricerca lineare, e restituisce l'indice di una delle occorrenze dell'oggetto cercato. Se l'oggetto non è contenuto nell'array, restituisce -1.

2) `public static int binarySearch(Object[] a, Object key)`
3) `public static int binarySearch(Object[] a, Object key, Comparator c)`

La funzione 2) cerca nell'array `a` l'oggetto `key`, utilizzando l'algoritmo di ricerca binaria. Prima della chiamata l'array deve essere già ordinato in ordine crescente secondo l'ordinamento naturale dei suoi elementi. La funzione restituisce l'indice di una delle occorrenze dell'oggetto cercato; se l'oggetto non è contenuto nell'array, restituisce il valore `(-(punto_di_inserimento)-1)`, dove `punto_di_inserimento` è l'indice dell'array in cui l'elemento cercato dovrebbe essere inserito per mantenere l'ordinamento.
La funzione 3) è analoga alla 2); è necessario però che l'arrray sia fornito ordinato in ordine crescente secondo la relazione d'ordine specificata nel comparatore `c` .

4) `public static boolean equals(Object[] o1, Object[] o2)`

La funzione 4) restituisce `true` se i due array sono uguali, ovvero se contengono gli stessi elementi nello stesso ordine. Due riferimenti ad array `null` vanno considerati come uguali; due elementi `null` negli array vanno considerati come uguali.

5) `public static void sort(Object[] a)`
6) `public static void sort(Object[] a, Comparator c)`

La funzione 5) ordina l'array `a` in ordine crescente secondo l'ordinamento naturale dei suoi elementi. Gli elementi dovranno quindi implementare `Comparable`. Implementare l'ordinamento mediante l'algoritmo *insertion sort*.
La funzione 6) è analoga alla 5), con l'ordinamento eseguito secondo la relazione d'ordine specificata dal comparatore `c`.

Implementare gli opportuni test di correttezza dei metodi implementati.

Continua sul retro →

Esercizio 2

Utilizzare le funzionalità della classe `Cronometro` vista in classe per eseguire delle misure di prestazioni sulle funzionalità della classe `MyArrays`. In particolare:

1. Misurare il tempo di esecuzione di `sort` per array in ingresso di dimensioni crescenti. Gli array in ingresso dovranno essere composti di elementi disordinati (suggerimento di esecuzione: utilizzare array di oggetti `Integer` ottenuti mediante generazione casuale tramite `java.util.Random`).
2. Misurare il tempo di esecuzione di `sort` per array in ingresso **già ordinati** di dimensioni crescenti.
3. Misurare il tempo di esecuzione di `binarySearch` per array in ingresso di dimensioni crescenti. In questo caso è opportuno eseguire numerose prove per ogni dimensione dell'array, e considerare poi la media dei tempi.

Presentare i risultati delle misure in forma di tabelle e grafici, e inviarli a `martino.fornasa@dei.unipd.it` entro il giorno 14/3/2011.