

Programmazione I corso A – Prova scritta del 12 gennaio 2005

Cognome e Nome: _____ Matr.: _____

Esercizio 1. (circa punti 9) Si consideri un'esecuzione del seguente programma Java:

```
class DatoMeteo {
    final String luogo;
    final int temperatura;

    DatoMeteo(String l, int t) {
        luogo = l;
        temperatura = t;
    }
}

class Meteo {
    private DatoMeteo[] datiMeteo = new DatoMeteo[10];
    private int numDatiMeteo;

    public DatoMeteo ricerca(String luogo) {
        for(int i = 0; i < numDatiMeteo; i++)
            if(datiMeteo[i].luogo.equals(luogo)) return datiMeteo[i];
        return null;
    }

    void aggiungiDato(DatoMeteo dm) {
        datiMeteo[numDatiMeteo] = dm;
        numDatiMeteo++;
    }

    void print() {
        for(int i = 0; i < numDatiMeteo; i++) {
            DatoMeteo dato = datiMeteo[i];
            System.out.println(dato.luogo + " " + dato.temperatura);
        }
    }
}

class ProvaMeteo {
    public static void main(String[] args) {
        Meteo meteo1 = new Meteo();
        Meteo meteo2 = new Meteo();
        meteo1.aggiungiDato(new DatoMeteo("Torino", 5));
        meteo2.aggiungiDato(meteo1.ricerca("Torino"));
        meteo1.print();
        meteo2.print();
    }
}
```

1.1 Si raffiguri lo stato della memoria durante l'esecuzione della seconda chiamata del metodo `aggiungiDato`; il disegno deve raffigurare lo stato della memoria nell'istante immediatamente precedente la deallocazione del frame di `aggiungiDato`.

1.2 Si scriva ciò che viene visualizzato sullo schermo durante l'esecuzione del programma.

NOTA. Si ricorda che una classe per cui non è stato definito nessun costruttore ha di default un costruttore vuoto.

Esercizio 2. (circa punti 4)

Si definisca un metodo statico il quale, preso come argomento un array *ar* di interi completamente riempito, lo modifichi in modo che, per ogni *i*, l'elemento *ar[i]* sia uguale alla somma dei valori originari di *ar[0]*, *ar[1]*, ..., *ar[i]*.

Esempio: *ar* = {5, 7, 2, 8, -3, 4}

ar modificato = {5, 12, 14, 22, 19, 23}

Il metodo deve avere complessità temporale ottimale.

Esercizio 3. (circa punti 9)

Si definisca una classe **Rettangolo** le cui istanze rappresentino ognuna un rettangolo di una data base e una data altezza; base e altezza non possono essere cambiate dopo la creazione dell'oggetto.

La classe contenga un metodo *area* che restituisce l'area del rettangolo *this*.

Si definisca poi una classe **ListaRettangoli** le cui istanze rappresentino ognuna una sequenza di rettangoli: la sequenza deve poter essere variata mediante l'aggiunta o la cancellazione di elementi, e deve essere mantenuta ordinata per ordine crescente di area. Oltre ai costruttori e ai metodi ritenuti necessari, si definisca un costruttore

ListaRettangoli(Rettangolo[] rettangoli)

il quale costruisca una *ListaRettangoli* contenente tutti gli elementi dell'array argomento, ma **ordinati** per ordine crescente di area.

La classe *ListaRettangoli* non deve contenere procedure di input-output.

Esercizio 4. (punti 6) Si consideri definito il metodo statico :

`public static int trovaPosMinimo(int m, int n, int p)`

che restituisce 1 se $m \leq n$ e $m \leq p$, 2 se $n < m$ e $n \leq p$, 3 se $p < m$ e $p < n$ (trova il posto del minore dei parametri *m,n,p*); ad esempio `trovaPosMinimo(8, -4, -4) = 2`

Usare `trovaPosMinimo` per scrivere un metodo statico

`fusioneATre(int[] a, int[] b, int[] c)`

che ha come parametri tre array di interi ordinati e completamente riempiti e restituisce un array ordinato e completamente riempito fusione degli array *a*, *b* e *c*.

Si chiede che il metodo `fusioneATre` **non** esegua quanto richiesto facendo la fusione di *a* con *b* in un array risultato e poi la fusione di questo risultato con *c*.