

# Esame di Fondamenti di Informatica L-B

## Ingegneria Gestionale e dei Processi Gestionali

Appello del 21/6/2013

### Esercizio 1 (4 punti)

Discutere i concetti della programmazione modulare.

### Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[], int M, int N) {
    int i, sum=0;
    for(i=M; --i>N;)
        sum+=V[i];
    return sum;
}

public static int g(int V[], int N) {
    int j=1, sum=0;
    do
        sum+=f(V, N, j);
    while(++j<N)
    return sum;
}
```

- Calcolare la complessità in passi base del metodo `f` nei termini dei parametri `M` e `N` (si distinguono i casi in cui `N` assume valori minori di `M` da quelli in cui assume valori maggiori o uguali a `M`).
- Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N` pari e maggiore di 0).
- Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

### Esercizio 3 (5 punti)

Il talent show “Fattore C” è una gara tra aspiranti cantanti non professionisti della società di produzione televisiva “Michelia”. Giunti alla settima stagione, la “Michelia” ha deciso di informatizzare i dati relativi ai cantanti in gara. Ogni cantante è caratterizzato dal nome, dalla data di nascita e dalla categoria (es. “Under uomini”, “Over donne”, ecc.). Si scriva una classe `Cantante` per la “Michelia” che:

- Possieda un opportuno costruttore con parametri.
- Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell’oggetto.
- Presenti il metodo `toString` che fornisca una descrizione del cantante.
- Possieda il metodo `equals` per stabilire l’uguaglianza con un altro oggetto `Cantante` (la verifica va fatta sul nome e sulla data di nascita).
- Implementi l’interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Cantante` passato come parametro (in ordine alfabetico di categoria e, quindi, per nome).

### Esercizio 4 (7 punti)

Si scriva una classe `Puntata` che memorizzi le informazioni relative agli aspiranti cantanti che parteciperanno a una puntata del talent show “Fattore C”. Per ogni puntata si memorizzi la data di messa in onda e un insieme che conterrà gli aspiranti cantanti. La classe `Puntata` deve:

- Presentare un opportuno costruttore con parametri (inizialmente a una puntata non partecipa alcun cantante).
- Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell’oggetto.
- Presentare il metodo `toString` che fornisca la descrizione della puntata (inclusa la descrizione di tutti i suoi cantanti).
- Possedere il metodo `aggiungi` che, dato un oggetto `Cantante`, lo aggiunga a quelli presenti nella puntata, controllando che tale inserimento sia possibile.
- Presentare il metodo `equals` per stabilire l’uguaglianza con un altro oggetto `Puntata` (la verifica va fatta sulla data di messa in onda).
- Possedere il metodo `cantanti` che, data una categoria, restituisca un insieme contenente tutti gli aspiranti cantanti di tale categoria presenti nella puntata.
- Implementare l’interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Puntata` passato come parametro (in ordine di data di messa in onda).

### Esercizio 5 (8 punti)

Si scriva un’applicazione per la “Michelia” che:

- Crei una lista di oggetti `Puntata`.
- Crei un oggetto `Puntata`, lette da tastiera le informazioni necessarie.
- Inserisca l’oggetto di cui al punto 2. all’interno della lista di cui al punto 1, mantenendo tale lista ordinata secondo il punto 7. dell’esercizio 4.
- Crei un oggetto `Cantante`, lette da tastiera le informazioni necessarie.
- Inserisca il cantante creato al punto 4. tra quelli partecipanti alla puntata di cui al punto 2., controllando che tale inserimento sia possibile
- Letta una categoria, stampi a video i nomi di tutti i cantanti di tale categoria che partecipano alla puntata di cui al punto 2.
- Stampi a video la data della puntata che contiene il massimo numero totale di cantanti della categoria letta al punto 6.

## Soluzione Esercizio 2

### Domanda 1:

2 assegnamenti	2	o 2
--i < N	M - N	o 1
sum += V[i]	M - N - 1	o 0
Totale	2M - 2N + 1	o 3

### Domanda 2:

2 assegnamenti	2
sum += f(V, N, j)	N - 1
++j < N	N - 1
complessità di f	$N^2 - 1$
Totale	$N^2 + 2N - 1$

Complessità di f:  $\sum_{j=1}^{N-1} (2N - 2j + 1) = 2N^2 - 2N - N^2 + N + N - 1 = N^2 - 1$

### Domanda 3:

Complessità asintotica:  $O(N^2)$

## Soluzione Esercizio 3

```
class Cantante implements Comparable<Cantante> {
    private String nome, categoria;
    private int g, m, a;

    public Cantante(String nome, String categoria, int g, int m, int a) {
        this.nome = nome;
        this.categoria = categoria;
        this.g = g; this.m = m; this.a = a;
    }

    public String getNome() { return nome; }
    public String getCategoria() { return categoria; }
    public String getData() { return g + "/" + m + "/" + a; }
    public String toString() {
        return nome + " (" + getData() + "): " + categoria;
    }

    public boolean equals(Object o) { return equals((Cantante) o); }
    public boolean equals(Cantante c) {
        return this.nome.equals(c.nome) && this.getData().equals(c.getData());
    }

    public int compareTo(Cantante c) {
        int ret = this.categoria.compareTo(c.categoria);
        if(ret==0) ret = this.nome.compareTo(c.nome);
        return ret;
    }
}
```

## Soluzione Esercizio 4

```
import java.util.*;

class Puntata implements Comparable<Puntata> {
    private Set<Cantante> s;
    private int g, m, a;

    public Puntata(int g, int m, int a) {
        this.g = g; this.m = m; this.a = a;
        s=new HashSet<Cantante>();
    }

    public String getData() { return g + "/" + m + "/" + a; }
    public String toString() {
        return "Puntata del " + getData() + ": " + s.toString();
    }

    public boolean aggiungi(Cantante c) { return s.add(c); }

    public boolean equals(Object o) { return equals((Puntata) o); }
    public boolean equals(Puntata p) { return getData().equals(p.getData()); }

    public Set<Cantante> cantanti(String categoria) {
        Set<Cantante> i=new TreeSet<Cantante>();
        for(Cantante c: s)
            if(c.getCategoria().equals(categoria))
                i.add(c);
        return i;
    }

    public int compareTo(Puntata p) {
        int ret = this.a - p.a;
        if(ret==0) ret = this.m - p.m;
        if(ret==0) ret = this.g - p.g;
        return ret;
    }
}
```

## Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
    public static void main(String[] args) {
        List<Puntata> l=new LinkedList<Puntata>();
        Puntata p=new Puntata(Lettore.in.leggiInt(), Lettore.in.leggiInt(),
            Lettore.in.leggiInt());
        int i=0;
        while((i<l.size())&&(l.get(i).compareTo(p)<0)) i++;
        l.add(i, p);
        Cantante c=new Cantante(Lettore.in.leggiLinea(),
            Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
            Lettore.in.leggiInt(), Lettore.in.leggiInt());
        if(!p.aggiungi(c)) System.out.println("Cantante già presente!");
        String categoria = Lettore.in.leggiLinea();
        for(Cantante z: p.cantanti(categoria)) System.out.println(z.getNome());
        Puntata max = null; int maxC = 0;
        for(Puntata x: l) {
            int tot = x.cantanti(categoria).size();
            if(tot > maxC) { maxC = tot; max = x; }
        }
        System.out.println(max.getData());
    }
}
```