# Esame di Fondamenti di Informatica L-B
# Ingegneria Gestionale e dei Processi Gestionali

## Appello del 13/7/2018

***Esercizio 1 (4 punti)***

Gestione e utilizzo di liste in Java.

***Esercizio 2 (6 punti)***

Siano dati i seguenti metodi Java:

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

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

1. Calcolare la complessità in passi base del metodo `f` nei termini del parametro `N` (si distinguano i casi in cui `N` assume valori pari da quelli in cui assume valori dispari).
2. Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N` dispari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

***Esercizio 3 (5 punti)***

Julio Campos ha deciso di informatizzare la gestione del personale di sala e cucina del suo locale "Destra&Sinistra". In particolare, per ogni dipendente occorre memorizzare nome e cognome, mansione (es., "cuoco", "cameriere", "sguattero") e data di nascita. Si scriva una classe `Dipendente` per Julio Campos che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca una descrizione del dipendente.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Dipendente` (la verifica va fatta su nome e cognome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Dipendente` passato come parametro (per ordine alfabetico di cognome e nome e, a parità, per età crescente).

***Esercizio 4 (8 punti)***

Si scriva una classe `Turno` che registri le informazioni riguardanti i dipendenti assegnati a un particolare turno (i turni si ripetono con cadenza settimanale). Per ogni turno occorre memorizzare il giorno della settimana mentre i dipendenti vanno memorizzati all'interno di un insieme. La classe `Turno` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, l'insieme dei dipendenti è vuoto).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del turno (inclusa la descrizione di tutti i dipendenti).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Turno` (la verifica va effettuata unicamente sul giorno).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Dipendente`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `mansione` che, data una stringa, indichi il numero di dipendenti all'interno del turno che hanno tale mansione.
7. Possedere il metodo `dipendente` che, data una stringa, indichi se, all'interno del turno, esiste un dipendente con nome o cognome uguale a tale stringa.

***Esercizio 5 (8 punti)***

Si scriva un'applicazione per Julio Campos che:

1. Crei una lista di oggetti `Turno`.
2. Crei un oggetto `Turno`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. in testa alla lista di cui al punto 1.
4. Crei un oggetto `Dipendente`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. tra quelli contenuti nel turno di cui al punto 2., controllando che tale inserimento sia possibile.
6. Letti da tastiera una mansione e un numero intero, stampi a video la descrizione di tutti i turni aventi un numero di dipendenti aventi tale mansione inferiore al valore letto.
7. Stampi a video la descrizione del turno avente il minor numero di camerieri.

***Soluzione Esercizio 2***

**Domanda 1:**

| 2 assegnamenti | 2 | o 2 |
|---|---|---|
| `i++<N` | `N`/2 + 1 | o (`N` + 1)/2 + 1 |
| `sum+=V[i]` | `N`/2 | o (`N` + 1)/2 |
| `++i` | `N`/2 | o (`N` + 1)/2 |
| Totale | 3`N`/2 + 3 | o 3`N`/2 + 9/2 |

**Domanda 2:**

| 2 assegnamenti | 2 |
|---|---|
| `j<N` | `N` |
| `sum+=f(V, ++j)` | `N` |
| complessità di `f` | $3N^2/4 + 9N/2 + 3/4$ |
| Totale | $3N^2/4 + 13N/2 + 11/4$ |

complessità. di `f`: $\sum_{\substack{j=1 \\ j \text{ pari}}}^{N} \left(\frac{3j}{2}+3\right) + \sum_{\substack{j=1 \\ j \text{ dispari}}}^{N} \left(\frac{3j}{2}+\frac{9}{2}\right) = \sum_{\substack{j=1 \\ j \text{ pari}}}^{N} \left(\frac{3j}{2}+3\right) + \sum_{\substack{j=1 \\ j \text{ dispari}}}^{N} \left(\frac{3j}{2}+3\right) +$

$\sum_{\substack{j=1 \\ j \text{ dispari}}}^{N} \frac{3}{2} = \sum_{j=1}^{N}\left(\frac{3j}{2}+3\right) + \frac{3}{2}\frac{(N+1)}{2} = \frac{3}{2}\frac{N(N+1)}{2} + 3N + \frac{3}{2}\frac{(N+1)}{2} = \frac{3N^2}{4} + \frac{9N}{2} + \frac{3}{4}$

**Domanda 3:**

Complessità asintotica: $O(N^2)$

***Soluzione Esercizio 3***

```
class Dipendente implements Comparable<Dipendente> {
  private String nome, cognome, mansione;
  private int g, m, a;

  public Dipendente(String nome, String cognome, String mansione, int g,
    int m, int a) {
    this.nome = nome;
    this.cognome = cognome;
    this.mansione = mansione;
    this.g = g; this.m = m; this.a = a;
  }

  public String getNome() { return nome; }
  public String getCognome() { return cognome; }
  public String getMansione() { return mansione; }
  public String getData() { return g + "/" + m + "/" + a; }

  public String toString() {
    return nome + " " + cognome + " (" + getData() + "): " + mansione;
  }

  public boolean equals(Object o) { return equals((Dipendente) o); }
  public boolean equals(Dipendente d) {
    return this.nome.equals(d.nome) && this.cognome.equals(d.cognome);
  }

  public int compareTo(Dipendente d) {
    int result = this.cognome.compareTo(d.cognome);
    if(result==0) result = this.nome.compareTo(d.nome);
    if(result==0) result = d.a - this.a;
    if(result==0) result = d.m - this.m;
    if(result==0) result = d.g - this.g;
    return result;
  }
}
```

***Soluzione Esercizio 4***

```
import java.util.*;

class Turno {
  private String giorno;
  private Set<Dipendente> s;

  public Turno(String giorno) {
    this.giorno = giorno;
    s = new TreeSet<Dipendente>();
  }

  public String getGiorno() { return giorno; }
  public String toString() { return giorno + ": " + s;}
  public boolean equals(Object o) { return equals((Turno) o); }
  public boolean equals(Turno t){ return this.giorno.equals(t.giorno); }
  public boolean aggiungi(Dipendente d) { return s.add(d); }

  public int mansione(String mansione) {
    int totale = 0;
    for(Dipendente d: s) if(d.getMansione().equals(mansione)) totale++;
    return totale;
  }

  public boolean dipendente(String s) {
    for(Dipendente d: this.s)
      if(d.getNome().equals(s) || d.getCognome().equals(s)) return true;
    return false;
  }
}
```

***Soluzione Esercizio 5***

```
import fiji.io.*;
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Turno> l = new LinkedList<Turno>();
    Turno t = new Turno(Lettore.in.leggiLinea());
    l.add(0,t);
    Dipendente d = new Dipendente(Lettore.in.leggiLinea(),
      Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(), Lettore.in.leggiInt());
    if(!t.aggiungi(d)) System.out.println("Inserimento non avvenuto!");
    String mansione=Lettore.in.leggiLinea();
    int minimo=Lettore.in.leggiInt();
    for(Turno x: l)
      if(x.mansione(mansione)<minimo) System.out.println(x);
    Turno min = null;
    int minCam = 0;
    for(Turno x: l) {
      int camerieri=x.mansione("cameriere");
      if(min==null || camerieri<minCam) { min=x; minCam=camerieri; }
    }
    System.out.println(min.toString());
  }
}
```