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

## Appello del 18/1/2019

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

Discutere le principali differenze esistenti tra gli array e le collezioni in Java.

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

Siano dati i seguenti metodi Java:

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

public static int g(int V[],int N) {
  int j, sum=0;
  for(j=0; ++j<N; ++j)
    sum+=f(V, j);
  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)***

Andrea Fintana vuole lanciare, in società col fratello, un nuovo locale specializzato nel pokè, un piatto di origine hawaiana. Ogni piatto che verrà proposto dal locale è basato su una serie di ingredienti, tra cui pesce crudo, riso e altri condimenti assortiti. Andrea ha perciò deciso di informatizzare la gestione dei menù che verranno proposti nel locale di prossima apertura. Per questo, per ogni ingrediente che farà parte di un pokè, occorre memorizzare il nome, la tipologia (es.: pesce, base, condimento) e le calorie per porzione (il pokè si caratterizza per essere un piatto leggero, fresco e a basso contenuto calorico). Si scriva una classe `Ingrediente` per Andrea Fintana 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 dell'ingrediente.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Ingrediente` (la verifica va fatta su nome e tipologia).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Ingrediente` passato come parametro (per calorie decrescenti e, a parità, in ordine alfabetico per nome).

***Esercizio 4 (7 punti)***

Si scriva una classe `Piatto` che registri le informazioni riguardanti i piatti che verranno presentati nei menù. Per ogni piatto occorre memorizzare il nome mentre gli ingredienti vanno inseriti all'interno di un insieme. La classe `Piatto` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, l'insieme degli ingredienti 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 piatto (inclusa la descrizione di tutti gli ingredienti).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Piatto` (la verifica va effettuata unicamente sul nome).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Ingediente`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `valido` che indichi se il piatto contiene o meno un ingrediente di tipo pesce.
7. Possedere il metodo `calorie` che restituisca il numero totale di calorie del piatto.

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

Si scriva un'applicazione per Andrea Fintana che:

1. Crei una lista di oggetti `Piatto`.
2. Crei un oggetto `Piatto`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. in coda alla lista di cui al punto 1.
4. Crei un oggetto `Ingrediente`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno del piatto di cui al punto 2., controllando che tale inserimento sia possibile.
6. Stampi a video il nome dei piatti "non validi", ovvero che non contengono pesce.
7. Stampi a video la descrizione del piatto avente il maggior numero di calorie.

***Soluzione Esercizio 2***

**Domanda 1:**

| 2 assegnamenti | 2 | o 2 |
|---|---|---|
| `i-->0` | N/2 + 1 | o (N + 1)/2+ 1 |
| `sum+=V[i--]` | N/2 | o (N + 1)/2 |
| Totale | N + 3 | o N + 4 |

**Domanda 2:**

| 2 assegnamenti | 2 |
|---|---|
| `++j<N` | (N – 1)/2 + 1 |
| `sum+=f(V, j)` | (N – 1)/2 |
| `++j` | (N – 1)/2 |
| complessità di `f` | $N^2/4 + 3N/2 – 7/4$ |
| Totale | $N^2/4 + 3N – 1/4$ |

complessità. di `f`: $\sum_{\substack{j=1 \\ j \text{ dispari}}}^{N-2} (j+4) = \sum_{i=0}^{\frac{N-3}{2}} (2i+5) = \frac{(N-3)}{2}\frac{(N-1)}{2} + 5\frac{N-1}{2} = \frac{N^2}{4} + \frac{3N}{2} - \frac{7}{4}$

**Domanda 3:**

Complessità asintotica: $O(N^2)$

***Soluzione Esercizio 3***

```
class Ingrediente implements Comparable<Ingrediente> {
  private String nome, tipologia;
  private int calorie;

  public Ingrediente(String nome, String tipologia, int calorie) {
    this.nome = nome;
    this.tipologia = tipologia;
    this.calorie = calorie;
  }

  public String getNome() { return nome; }
  public String getTipologia() { return tipologia; }
  public int getCalorie() { return calorie; }

  public String toString() {
    return nome + " (" + tipologia + "): " + calorie;
  }

  public boolean equals(Object o) { return equals((Ingrediente) o); }
  public boolean equals(Ingrediente i) {
    return this.nome.equals(i.nome) && this.tipologia.equals(i.tipologia);
  }

  public int compareTo(Ingrediente i) {
    int ret = i.calorie - this.calorie;
    if(ret==0) ret =  this.nome.compareTo(i.nome);
    return ret;
  }
}
```

***Soluzione Esercizio 4***

```
import java.util.*;

class Piatto {
  private String nome;
  private Set<Ingrediente> s;

  public Piatto(String nome) {
    this.nome = nome;
    s = new HashSet<Ingrediente>();
  }

  public String getNome() { return nome; }
  public String toString() { return nome + ": " + s;}
  public boolean equals(Object o) { return equals((Piatto) o); }
  public boolean equals(Piatto p){ return this.nome.equals(p.nome); }
  public boolean aggiungi(Ingrediente i) { return s.add(i); }

  public boolean valido() {
    for(Ingrediente i: s) if(i.getTipologia().equals("pesce")) return true;
    return false;
  }

  public int calorie() {
    int calorie = 0;
    for(Ingrediente i: s) calorie += i.getCalorie();
    return calorie;
  }
}
```

***Soluzione Esercizio 5***

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

class Applicazione {
  public static void main(String[] args) {
    List<Piatto> l = new LinkedList<Piatto>();
    Piatto p = new Piatto(Lettore.in.leggiLinea());
    l.add(p);
    Ingrediente i = new Ingrediente(Lettore.in.leggiLinea(),
        Lettore.in.leggiLinea(),Lettore.in.leggiInt());
    if(!p.aggiungi(i)) System.out.println("Inserimento non avvenuto!");
    for(Piatto x: l)
      if(!x.valido()) System.out.println(x.getNome());
    Piatto max = null;
    int maxCalorie = 0;
    for(Piatto x: l) {
      int calorie = x.calorie();
      if(max==null || calorie>maxCalorie) {
        max=x;
        maxCalorie=calorie;
      }
    }
    System.out.println(max.toString());
  }
}
```