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

Appello del 19/7/2013

Esercizio 1 (4 punti)

Definire il concetto di ricorsione.

Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

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

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

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

Esercizio 3 (5 punti)

Il programma televisivo "Non si gioca col cuoco", prodotto dalla società "LeMonde", propone menù realizzati da aspiranti cuochi non professionisti. Giunti alla quattordicesima stagione, la "LeMonde" ha deciso di informatizzare i dati relativi ai prodotti culinari presentati durante il programma. Ogni piatto è caratterizzato dal nome, dal tempo necessario per la preparazione (in minuti) e dalla tipologia (es. "Primo", "Dessert", ecc.). Si scriva una classe `Piatto` per la "LeMonde" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca una descrizione del piatto.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Piatto` (la verifica va fatta sul nome e sulla tipologia).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Piatto` passato come parametro (in ordine alfabetico di categoria e, quindi, per durata decrescente).

Esercizio 4 (7 punti)

Si scriva una classe `Puntata` che memorizzi le informazioni relative ai prodotti culinari proposti durante una puntata del programma "Non si gioca col cuoco". Per ogni puntata occorre memorizzare la data di messa in onda e il nome del cuoco in gara, mentre i piatti proposti vanno memorizzati all'interno di una lista. La classe `Puntata` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la lista dei piatti è vuota).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della puntata (inclusa la descrizione di tutti i piatti presentati).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Piatto`, lo inserisca all'interno della lista, mantenendo la lista ordinata secondo il punto 5. dell'esercizio 3.
5. Presentare il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Puntata` (la verifica va fatta sulla data di messa in onda).
6. Possedere il metodo `tempoTotale` che restituisca il tempo totale necessario per la preparazione di tutti i piatti proposti nella puntata.
7. Presentare il metodo `tipologia` che, data una tipologia di piatti, indichi se all'interno della puntata è proposto almeno un piatto di tale tipologia.

Esercizio 5 (8 punti)

Si scriva un'applicazione per la "LeMonde" che:

1. Crei un insieme di oggetti `Puntata`.
2. Crei un oggetto `Puntata`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. all'interno dell'insieme di cui al punto 1., controllando che tale inserimento sia possibile.
4. Crei un oggetto `Piatto`, lette da tastiera le informazioni necessarie.
5. Inserisca il piatto creato al punto 4. tra quelli proposti nella puntata di cui al punto 2.
6. Letta da tastiera la durata massima di una puntata, stampi a video la data di tutte le puntate che hanno una durata maggiore di tale valore.
7. Letta una tipologia di piatto, stampi a video il nome dei cuochi che non propongono piatti di tale categoria nella puntata a loro dedicata.
8. Stampi a video la data della puntata più breve tra tutte quelle dell'insieme di cui al punto 1.

Soluzione Esercizio 2

Domanda 1:

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

Domanda 2:

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

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

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Piatto implements Comparable<Piatto> {
  private String nome, tipologia;
  private int durata;

  public Piatto(String nome, String tipologia, int durata) {
    this.nome = nome;
    this.tipologia = tipologia;
    this.durata = durata;
  }

  public String getNome() { return nome; }
  public String getTipologia() { return tipologia; }
  public int getDurata() { return durata; }

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

  public boolean equals(Object o) { return equals((Piatto) o); }
  public boolean equals(Piatto p) {
    return nome.equals(p.nome) && tipologia.equals(p.tipologia);
  }

  public int compareTo(Piatto p) {
    int ret = this.tipologia.compareTo(p.tipologia);
    if(ret==0) ret = p.durata – this.durata;
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Puntata {
  private List<Piatto> l;
  private String cuoco;
  private int g, m, a;
  public Puntata(String cuoco, int g, int m , int a) {
    this.cuoco = cuoco;
    this.g = g; this.m = m; this.a = a;
    l=new ArrayList<Piatto>();
  }
  public String getData() { return g + "/" + m + "/" + a; }
  public String getCuoco() { return cuoco; }
  public String toString() {
    return "Puntata del " + getData() + ": " + cuoco + "," +  l.toString();
  }
  public void aggiungi(Piatto p) {
    int i=0;
    while((i<l.size())&&(l.get(i).compareTo(p)<0)) i++;
    l.add(i, p);
  }
  public boolean equals(Object o) { return equals((Puntata) o); }
  public boolean equals(Puntata p) { return getData().equals(p.getData()); }
  public int tempoTotale() {
    int tot = 0;
    for (Piatto p : l) tot+= p.getDurata();
    return tot;
  }
  public boolean tipologia(String tipologia) {
    for(Piatto p: l) if(p.getTipologia().equals(tipologia)) return true;
    return false;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Puntata> s=new HashSet<Puntata>();
    Puntata p=new Puntata(Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt());
    if(!s.add(p)) System.out.println("Puntata già esistente!");
    Piatto x=new Piatto(Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
      Lettore.in.leggiInt());
    p.aggiungi(x);
    int durata = Lettore.in.leggiInt();
    for(Puntata z: s)
      if(z.tempoTotale() > durata) System.out.println(z.getData());
    String tipologia = Lettore.in.leggiLinea();
    for(Puntata z: s)
      if(!z.tipologia(tipologia)) System.out.println(z.getCuoco);)
    Puntata min = null;
    int minT = 0;
    for(Puntata z: s) {
      int t = z.tempoTotale();
      if(min == null || t < minT) { minT = t; min = z; }
    }
    System.out.println(min.getData()); }
}
```