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

Appello del 15/1/2016

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=M, sum=0;
  while (++i<=N)
    sum+=V[i];
  return sum;
}

public static int g(int V[], int N, int M) {
  int j=N, sum=0;
  while(j>0)
    sum+=f(V, j--, M);
  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 dei parametri `M` e `N` (si supponga `N` minore di `M`).
3. Calcolare la complessità asintotica del metodo `g` nei termini dei parametri `M` e `N`.

Esercizio 3 (5 punti)

Max Frate è il preparatore atletico della squadra di calcio amatoriale "La Mucca Maria". Per organizzare al meglio la ripresa degli allenamenti dalla pausa invernale, Max ha deciso di memorizzare le informazioni relative a ogni esercizio all'interno di un calcolatore. In particolare, occorre registrare il nome, la durata (in minuti) e il numero di calorie necessarie per lo svolgimento dell'esercizio stesso. Si scriva una classe `Esercizio` per Max Frate 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'esercizio.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Esercizio` (la verifica va fatta unicamente sul nome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Esercizio` passato come parametro (in ordine crescente di durata e, a parità, in ordine alfabetico per nome).

Esercizio 4 (7 punti)

Si scriva una classe `Allenamento` che memorizzi le informazioni riguardanti gli esercizi svolti in un determinato allenamento. Per ogni allenamento occorre memorizzare la data e il nome del campo di svolgimento, mentre gli esercizi vanno inseriti all'interno di una lista. La classe `Allenamento` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, non vi sono esercizi).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione dell'allenamento (inclusa la descrizione di tutti gli esercizi).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Allenamento` (la verifica va effettuata unicamente sulla data).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Esercizio`, lo inserisca all'interno della lista, mantenendo quest'ultima ordinata secondo il punto 5. dell'Esercizio 3.
6. Possedere il metodo `totaleCalorie` che restituisca il numero totale di calorie necessarie per l'allenamento.
7. Presentare il metodo `esercizio` che, dato il nome di un esercizio, indichi se esso è incluso nell'allenamento.

Esercizio 4 (7 punti)

Si scriva un'applicazione per Max Frate che:

1. Crei un insieme di oggetti `Allenamento`.
2. Crei un oggetto `Allenamento`, 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 `Esercizio`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno dell'allenamento di cui al punto 2.
6. Letto da tastiera il nome di un esercizio, stampi a video la descrizione di tutti gli allenamenti in cui tale esercizio è svolto.
7. Stampi a video la data dell'allenamento che richiede il maggior numero di calorie, tra quelli contenuti nell'insieme di cui al punto 1.

Soluzione Esercizio 2

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`j>0`	$N + 1$
`sum+=f(V, j--, M)`	N
complessità di `f`	$2MN - N^2 + 2N$
Totale	$2MN - N^2 + 6N + 3$

complessità di `f`: $\sum_{j=1}^{N}(2M - 2j + 3) = 2MN - N(N+1) + 3N$

Domanda 3:
Complessità asintotica: $O(MN)$

Soluzione Esercizio 3

```
class Esercizio implements Comparable<Esercizio> {
  private String nome;
  private int durata, calorie;

  public Esercizio(String nome, int durata, int calorie) {
    this.nome = nome;
    this.durata= durata;
    this.calorie= calorie;
  }

  public String getNome() { return nome; }
  public int getDurata() { return durata; }
  public int getCalorie() { return calorie; }

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

  public boolean equals(Object o) { return equals((Esercizio) o); }
  public boolean equals(Esercizio e) { return this.nome.equals(e.nome); }

  public int compareTo(Esercizio e) {
    int ret = this.durata - e.durata;
    if(ret==0) ret = this.nome.compareTo(e.nome);
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Allenamento {
  private List<Esercizio> l;
  private String campo;
  private int g, m, a;

  public Allenamento(String campo, int g, int m, int a) {
    this.campo = campo;
    this.g = g; this.m = m; this.a = a;
    l = new LinkedList<Esercizio>();
  }
  public String getCampo() { return campo; }
  public String getData() { return g+"/"+m+"/"+a; }
  public String toString() { return campo+ "(" + getData() + "):" + l;}
  public boolean equals(Object o) { return equals((Allenamento) o); }
  public boolean equals(Allenamento a){
    return getData().equals(a.getData());
  }

  public void aggiungi(Esercizio e) {
    int i = 0;
    while((i<l.size())&&(l.get(i).compareTo(e)<0)) i++;
    l.add(i, e);
  }

  public int totaleCalorie() {
    int totale = 0;
    for(Esercizio e: l) totale += e.getCalorie();
    return totale;
  }

  public boolean esercizio(String nome) {
    for(Esercizio e: l) if(e.getNome().equals(nome)) return true;
    return false;
  }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Allenamento> s = new TreeSet<Allenamento>();
    Allenamento a = new Allenamento(Lettore.in.leggiLinea(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(), Lettore.in.leggiInt());
    if(!s.add(a)) System.out.println("Allenamento già esistente!");
    Esercizio e = new Esercizio(Lettore.in.leggiLinea(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt());
    a.aggiungi(e);
    String nome = Lettore.in.leggiLinea();
    for(Allenamento x: s) if(x.esercizio(nome)) System.out.println(x);
    Allenamento max=null;
    int maxC=0;
    for(Allenamento x: s) {
      int totale=x.totaleCalorie();
      if(totale>maxC) { max=x; maxC=totale; }
    }
    System.out.println(max.getData());
  }
}
```