Esame di Fondamenti di Informatica T-1/T-A
Ingegneria Gestionale (A-K)

Appello del 15/9/2017

NOTA: Per il superamento dell'esame è **necessario** ottenere la sufficienza nello svolgimento dell'Esercizio 1.

Esercizio 1 (4 punti)

1. Descrivere l'architettura di un elaboratore elettronico.
2. Discutere l'utilizzo di metodi e proprietà statiche in Java.

Esercizio 2 (2 punti)

Convertire in binario i numeri **43** e **88**, supponendo di utilizzare una rappresentazione a 8 bit in complemento a 2. Si esegua infine la somma dei due numeri, riconvertendo il risultato in base 10, motivando eventuali differenze tra il risultato ottenuto e quello atteso.

Esercizio 3 (5 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 4 (5 punti)

Silvio Torso è un giovane laureando magistrale in Ingegneria Chimica. Per meglio organizzare i molteplici esperimenti richiestigli dalla sua relatrice, ha deciso di gestire all'interno di un calcolatore le informazioni relative ai composti usati nelle reazioni. In particolare, per ogni composto occorre registrare il nome, la data di produzione e la quantità in ml. Si scriva una classe `Composto` per Silvio Torso 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 composto.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Composto` (la verifica va fatta sul nome e sulla quantità).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Composto` passato come parametro (per ordine alfabetico sul nome e, a parità, per quantità decrescente).

Esercizio 5 (7 punti)

Si scriva una classe `Esperimento` che registri le informazioni riguardanti un singolo esperimento compiuto da Silvio Torso. Per ogni esperimento occorre memorizzare il titolo e una nota sintetica, mentre i composti utilizzati vanno memorizzati all'interno di un insieme. La classe `Esperimento` deve:

1. Presentare un opportuno costruttore con parametri (l'insieme dei composti è inizialmente vuoto).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione dell'esperimento (inclusa la descrizione di tutti i composti utilizzati).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Esperimento` (la verifica va effettuata esclusivamente sul titolo).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Composto`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `quantita` che, dato il nome di un composto, restituisca la quantità totale di tale composto presente nell'esperimento.
7. Possedere il metodo `totale` che restituisca la quantità totale di composti presenti nell'esperimento.

Esercizio 6 (8 punti)

Si scriva un'applicazione per Silvio Torso che:

1. Crei una lista di oggetti `Esperimento`.
2. Crei un oggetto `Esperimento`, 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 `Composto`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno dell'esperimento di cui al punto 2., controllando che tale inserimento sia possibile.
6. Letto da tastiera il nome di un composto, stampi a video il titolo di tutti gli esperimenti che comprendono tale composto.
7. Stampi a video la descrizione dell'esperimento che usa la maggior quantità totale di composti.

Soluzione Esercizio 2

$43_{10} = 00101011_2$
$88_{10} = 01011000_2$

00101011+
01011000
10000011

$10000011_2 = -125_{10}$
Il numero rappresentato è diverso da quello atteso (131), in quanto quest'ultimo non è rappresentabile con 8 bit.

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`j<N`	(N + 3)/2
`sum+=f(V, ++j)`	(N + 1)/2
`++j`	(N + 1)/2
complessità di `f`	$N^2/4 + 3N/2 + 5/4$
Totale	$N^2/4 + 3N + 23/4$

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

Domanda 3:
Complessità asintotica: $O(N^2)$

Soluzione Esercizio 4

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

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

  public String getNome() { return nome; }
  public int getQuantita() { return quantita; }
  public String getData() { return g+"/"+m+"/"+a; }

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

  public boolean equals(Object o) { return equals((Composto) o); }
  public boolean equals(Composto c) {
    return nome.equals(c.nome) && quantita==c.quantita;
  }

  public int compareTo(Composto c) {
    int ret = this.nome.compareTo(c.nome);
    if(ret==0) ret = c.quantita - this.quantita;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Esperimento {
  private String titolo, note;
  private Set<Composto> s;
  public Esperimento(String titolo, String note) {
    this.titolo = titolo;
    this.note = note;
    s = new HashSet<Composto>();
  }

  public String getTitolo() { return titolo; }
  public String getNote() { return note; }
  public String toString() { return titolo+ "(" + note + "): " + s;}
  public boolean equals(Object o) { return equals((Esperimento) o); }
  public boolean equals(Esperimento e){ return titolo.equals(e.titolo);}
  public boolean aggiungi(Composto c) { return s.add(c); }

  public int quantita(String nome) {
    for(Composto c: s)
      if(c.getNome().equals(nome)) return c.getQuantita();
    return 0;
  }

  public int totale() {
    int totale=0;
    for(Composto c: s) totale+=c.getQuantita();
    return totale;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Esperimento> l = new ArrayList<Esperimento>();
    Scanner scanner = new Scanner(System.in);
    Esperimento e = new Esperimento(scanner.nextLine(),
        scanner.nextLine());
    l.add(0,e);
    Composto c = new Composto(scanner.nextLine(), scanner.nextInt(),
        scanner.nextInt(),scanner.nextInt(), scanner.nextInt());
    if(!e.aggiungi(c)) System.out.println("Inserimento non riuscito!");
    String nome = scanner.nextLine();
    for(Esperimento x: l)
      if(x.quantita(nome)>0) System.out.println(x.getTitolo());
    int maxQ = 0;
    Esperimento max = null;
    for(Esperimento x: l) {
      int quantita = x.totale();
      if(quantita>maxQ) { max=e; maxQ=quantita; }
    }
    System.out.println(max.toString());
  }
}
```