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

Appello del 17/1/2014

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

Esercizio 1 (4 punti)

1. Illustrare i componenti interni di una CPU.
2. Descrivere i tipi primitivi del linguaggio Java.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **33,75** supponendo di utilizzare 8 bit per la mantissa (in modulo e segno) ed 8 bit per l'esponente (in complemento a 2). Si motivino infine eventuali differenze tra il numero originale e quello rappresentato.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[], int N) {
  int sum=0;
  for(int i=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 4 (5 punti)

L'alchimista StranoMadus, alla ricerca della pietra filosofale, ha deciso di memorizzare all'interno di un calcolatore i risultati dei suoi esperimenti. A tal scopo, per tutti i composti chimici utilizzati negli esperimenti vengono memorizzati il nome, la quantità (arrotondata al grammo) utilizzata ed il grado di purezza (in percentuale). Si scriva una classe `Composto` per StranoMadus 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 composto.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Composto` (la verifica va fatta sul nome e sul grado di purezza).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Composto` passato come parametro (in ordine alfabetico per nome e, a parità, per quantità decrescente).

Esercizio 5 (7 punti)

Si scriva una classe `Esperimento` che memorizzi le informazioni relative ad uno specifico esperimento. Per ogni esperimento occorre memorizzare la data e la temperatura dell'alambicco in cui vengono miscelati i composti, mentre i composti stessi vanno memorizzati all'interno di un insieme. La classe `Esperimento` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, nell'alambicco non c'è alcun composto).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione dell'esperimento (inclusa la descrizione di tutti i composti utilizzati).
4. Presentare il metodo `aggiungi` che, dato un oggetto `Composto`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
5. Possedere il metodo `quanti` che restituisca il numero di composti utilizzati nell'esperimento.
6. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Esperimento` (la verifica va fatta sulla data e sull'insieme di composti utilizzati).
7. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Esperimento` passato come parametro (in ordine decrescente di data).

Esercizio 6 (8 punti)

Si scriva un'applicazione per l'alchimista StranoMadus 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 coda alla lista di cui al punto 1.
4. Crei un oggetto `Composto`, lette da tastiera le informazioni necessarie.
5. Inserisca il composto creato al punto 4. tra quelli utilizzati nell'esperimento di cui al punto 2.
6. Stampi a video la data dell'esperimento, tra quelli della lista di cui al punto 1., che utilizza il numero massimo di composti.
7. Stampi a video la data dell'esperimento meno recente tra quelli della lista di cui al punto 1.
8. Verifichi se, all'interno della lista di cui al punto 1., esiste un altro esperimento che abbia la stessa data e utilizzi gli stessi composti dell'esperimento di cui al punto 2.

Soluzione Esercizio 2

$33{,}75_{10} = -100001{,}11_2$ quindi la mantissa è **(0)1000011**, l'esponente 6_{10} = **00000110**.
Il numero rappresentato è 33,5, diverso dal numero originale in quanto quest'ultimo è un numero che richiede un numero di cifre binarie maggiore rispetto a quelle disponibili.

Soluzione Esercizio 3

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	3N/2 + 3	o 3N/2 + 3/2

Domanda 2:

2 assegnamenti	2
`sum+=f(V, ++j)`	(N + 1)/2
`++j<N`	(N + 1)/2
complessità di `f`	$3N^2/8 + 3N/2+9/8$
Totale	$3N^2/8 + 5N/2 + 33/8$

complessità di `f`: $\sum_{\substack{j=0 \\ (j \text{ pari})}}^{N-1} \left(\frac{3j}{2}+3\right) = \sum_{i=0}^{(N-1)/2}(3i+3) = 3\frac{N^2}{8} - \frac{3}{8} + 3\frac{N+1}{2}$

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

Soluzione Esercizio 4

```
class Composto implements Comparable<Composto> {
  private String nome;
  private int quantita;
  private float purezza;

  public Composto(String nome, int quantita, float purezza) {
    this.nome = nome;
    this.quantita = quantita;
    this.purezza = purezza;
  }

  public String getNome() { return nome; }
  public int getQuantita() { return quantita; }
  public float getPurezza() { return purezza; }

  public String toString() {
    return nome + " (" + quantita + ", " + purezza+ ")";
  }

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

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

Soluzione Esercizio 5

```
import java.util.*;

class Esperimento implements Comparable<Esperimento> {
  private Set<Composto> s;
  private int g, m, a;
  private float temperatura;

  public Esperimento(int g, int m, int a, float temperatura) {
    this.g = g; this.m = m; this.a = a;
    this.temperatura = temperatura;
    s=new HashSet<Composto>();
  }

  public String getData() { return g + "/" + m + "/" + a; }
  public float getTemperatura() { return temperatura; }
  public String toString() {
    return getData() + "( " + temperatura + "):" + s.toString();
  }

  public boolean aggiungi(Composto c) { return s.add(c); }
  public int quanti() { return s.size(); }

  public boolean equals(Object o) { return equals((Esperimento) o); }
  public boolean equals(Esperimento e) {
    return getData().equals(e.getData()) && s.equals(e.s);
  }

  public int compareTo(Esperimento e) {
    int ret = e.a - a;
    if(ret==0) ret = e.m - m;
    if(ret==0) ret = e.g - g;
    return ret;
  }
}
```

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.nextInt(), scanner.nextInt(),
        scanner.nextInt(), scanner.nextFloat());
    l.add(e);
    Composto c=new Composto(scanner.nextLine(), scanner.nextInt(),
        scanner.nextFloat());
    e.aggiungi(c);
    Esperimento max = null;
    for(Esperimento x: l) if(max==null || x.quanti()>max.quanti()) max = x;
    System.out.println(max.getData());
    max=null;
    for(Esperimento x: l) if(max==null || x.compareTo(max)>0) max = x;
    System.out.println(max.getData());
    for(Esperimento x: l) if(x.equals(e) && x!=e)
      System.out.println("Esiste!");
  }
}
```