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

Appello del 13/9/2019

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

Esercizio 1 (4 punti)

1. Descrivere la gerarchia di memorie all'interno di un elaboratore.
2. Discutere l'utilizzo di metodi e proprietà statiche in Java.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **65,5** supponendo di utilizzare 8 bit per la mantissa (in modulo e segno) ed 8 bit per l'esponente (in complemento a 2). Si riconverta infine il numero rappresentato in base 10, motivando eventuali differenze con il numero originale.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

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

public static int g(int V[], int M, int N) {
  int j=0, sum=0;
  for(; ++j<M; j++)
    sum+=f(V, N, j--);
  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 maggiori di `N` da quelli in cui assume valori minori o uguali).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `M` e `N` (si supponga `M` maggiore di `N`).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (5 punti)

Arrigo Tonetti è decisamente un secchione: 6 esami superati con la media del 30! Ma ora arriva il difficile: l'esame di Informatica… Per prepararsi al meglio, Arrigo ha deciso di organizzare i testi degli esami precedenti all'interno di un calcolatore. Per questo, per ogni esercizio occorre memorizzare la categoria ("teoria", "linguaggi", "fondamenti", ecc.), il testo e il valore, in punti, dell'esercizio. Si scriva una classe `Esercizio` per Arrigo 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 su categoria e testo).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Esercizio` passato come parametro (in ordine alfabetico per categoria e, a parità, per valore decrescente).

Esercizio 5 (7 punti)

Si scriva una classe `Compito` che registri le informazioni riguardanti i compiti proposti nbegli appelli precedenti. Per ogni compito occorre memorizzare la data, mentre gli esercizi vanno inseriti all'interno di una lista. La classe `Compito` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la lista di esercizi è vuota).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del compito (inclusa la descrizione di tutti gli esercizi).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Compito` (la verifica va effettuata unicamente sulla data).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Esercizio`, lo inserisca in coda alla lista di esercizi.
6. Possedere il metodo `esercizi` che, data una categoria, restituisca un insieme contenente gli esercizi di tale categoria presenti all'interno del compito.
7. Possedere il metodo `punteggio` che restituisca il totale dei punteggi degli esercizi del compito.

Esercizio 6 (8 punti)

Si scriva un'applicazione per Arrigo Tonetti che:

1. Crei un insieme di oggetti `Compito`.
2. Crei un oggetto `Compito`, 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 del compito di cui al punto 2.
6. Stampi a video la descrizione di tutti i compiti aventi un punteggio totale superiore a 30.
7. Stampi a video la data del compito che prevede il maggior numero di esercizi di teoria.

Soluzione Esercizio 2

$65{,}5_{10} = 1000001{,}1_2$ quindi la mantissa è **(0)1000001**, l'esponente 7_{10} = **00000111**.
Il numero rappresentato è 65, diverso dall'originale poiché quest'ultimo ha un numero di cifre significative maggiore di 7.

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`sum+=V[i]`	$M-N$	o 1
`++i<N`	$M-N$	o 1
Totale	$2M-2N+2$	o 4

Domanda 2:

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

complessità. di `f`: $\sum_{j=1}^{N-1}(2N-2j+2)+\sum_{j=N}^{M-1}4 = 2N(N-1)-N(N-1)+2(N-1)+4(M-N) = N^2-3N+4M-2$

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

Soluzione Esercizio 4

```
class Esercizio implements Comparable<Esercizio> {
  private String categoria, testo;
  private int valore;

  public Esercizio(String categoria, String testo, int valore) {
    this.categoria = categoria;
    this.testo = testo;
    this.valore = valore;
  }

  public String getCategoria() { return categoria; }
  public String getTesto() { return testo; }
  public int getValore() { return valore; }

  public String toString() {
    return testo + " (" + categoria + ": " + valore;
  }

  public boolean equals(Object o) { return equals((Esercizio) o); }
  public boolean equals(Esercizio e) {
    return categoria.equals(e.categoria) && testo.equals(e.testo);
  }

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

Soluzione Esercizio 5

```
import java.util.*;

class Compito {
  private int g, m, a;
  private List<Esercizio> l;

  public Compito(int g, int m, int a) {
    this.g = g; this.m = m; this.a = a;
    l = new LinkedList<Esercizio>();
  }

  public String getData() { return g+"/"+m+"/"+a; }
  public String toString() { return getData() + "\n" + l;}
  public boolean equals(Object o) { return equals((Compito) o); }
  public boolean equals(Compito c){ return getData().equals(c.getData()); }

  public void aggiungi(Esercizio e) { l.add(e); }

  public Set<Esercizio> esercizi(String categoria) {
    Set<Esercizio> result = new HashSet<Esercizio>();
    for(Esercizio e: l)
      if(e.getCategoria().equals(categoria)) result.add(e);
    return result;
  }

  public int punteggio() {
    int result=0;
    for(Esercizio e: l) result+=e.getValore();
    return result;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Compito> s = new TreeSet<Compito>();
    Scanner scanner = new Scanner(System.in);
    Compito c = new Compito(scanner.nextInt(), scanner.nextInt(),
        scanner.nextInt());
    if(!s.add(c)) System.out.println("Inserimento non avvenuto!");
    Esercizio e = new Esercizio (scanner.nextLine(), scanner.nextLine(),
        scanner.nextInt());
    c.aggiungi(e);
    for(Compito x: s)
      if(x.punteggio()>30) System.out.println(x);
    Compito max = null;
    int maxN = 0;
    for(Compito x: s) {
      int n = x.esercizi("teoria").size();
      if(n > maxN) { max=x; maxN=n; }
    }
    System.out.println(max.getData());
  }
}
```