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

Appello del 25/10/2019

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

Esercizio 1 (4 punti)

1. Descrivere i tipi primitivi del linguaggio Java.
2. Discutere i diversi tipi di complessità computazionale di un algoritmo.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **0,025** 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 come somma di potenze di due, motivando eventuali differenze con il numero originale.

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=N, sum=0;
  for(; --j>0; ++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)

Safiria è una ragazza estremamente talentuosa e intraprendente! Strumentista poliedrica, ha deciso di organizzare nella sua scuola un concorso tra bande musicali. Per organizzare il tutto, Safiria ha deciso di organizzare le informazioni su bande e musicisti all'interno di un calcolatore. Per questo, per ogni strumentista occorre memorizzare il nome e il cognome, la data di nascita e il nome dello strumento suonato. Si scriva una classe `Strumentista` per Safiria 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 dello strumentista.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Strumentista` (la verifica va fatta su nome e cognome e data di nascita).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Strumentista` passato come parametro (per età decrescente e, a parità, in ordine alfabetico per cognome e nome).

Esercizio 5 (7 punti)

Si scriva una classe `Banda` che registri le informazioni riguardanti le bande che partecipano al concorso. Per ogni banda occorre memorizzare la classe (es. "5C", "3A"), mentre gli strumentisti vanno inseriti all'interno di un insieme. La classe `Banda` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, l'insieme di strumentisti è vuoto).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della banda (inclusa la descrizione di tutti i suoi strumentisti).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Banda` (la verifica va effettuata unicamente sulla classe).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Strumentista`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `strumento` che, dato il nome di uno strumento, restituisca il numero di strumentisti all'interno della banda che suonano tale strumento.
7. Possedere il metodo `piuGiovane` che restituisca lo strumentista più giovane tra quelli della banda.

Esercizio 6 (8 punti)

Si scriva un'applicazione per Safiria che:

1. Crei una lista di oggetti `Banda`.
2. Crei un oggetto `Banda`, 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 `Strumentista`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno della banda di cui al punto 2., controllando che tale inserimento sia possibile.
6. Stampi a video la descrizione di tutte le bande che non possiedono violinisti.
7. Stampi a video la classe della banda che presenta lo strumentista più giovane.

Soluzione Esercizio 2

$0{,}025_{10} = 0{,}000001100110_2$ quindi la mantissa è **(0)1100110**, l'esponente -5_{10} = **11111011**.
Il numero rappresentato è $2^{-6}+2^{-7}+2^{-10}+2^{-11}$, diverso dall'originale poiché quest'ultimo è periodico in base 2.

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`--j>0`	N
`sum+=f(V, j--)`	N – 1
`++j`	N – 1
complessità di `f`	$N^2/2 + 2N - 5/2$
Totale	$N^2/2 + 5N - 5/2$

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

$$\frac{N(N-1)}{2} + 2(N-1) + \frac{N-1}{2} = \frac{N^2}{2} + 2N - \frac{5}{2}$$

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

Soluzione Esercizio 4

```
class Strumentista implements Comparable<Strumentista> {
  private String nome, cognome, strumento;
  private int g, m, a;

  public Strumentista(String nome, String cognome, String strumento, int g,
      int m, int a) {
    this.nome = nome;
    this.cognome = cognome;
    this.strumento = strumento;
    this.g = g;
    this.m = m;
    this.a = a;
  }

  public String getNome() { return nome + " " + cognome; }
  public String getStrumento() { return strumento; }
  public String getData() { return g+"/"+m+"/"+a; }

  public String toString() {
    return getNome() + ", " + getData() + ": " + strumento;
  }

  public boolean equals(Object o) { return equals((Strumentista) o); }
  public boolean equals(Strumentista s) {
    return nome.equals(s.nome) && cognome.equals(s.cognome) &&
        getData().equals(s. getData());
  }

  public int compareTo(Strumentista s) {
    int ret = this.a-s.a;
    if(ret==0) ret = this.m-s.m;
    if(ret==0) ret = this.g-s.g;
    if(ret==0) ret = this.cognome.compareTo(s.cognome);
    if(ret==0) ret = this.nome.compareTo(s.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Banda {
  private String classe;
  private Set<Strumentista> s;

  public Banda(String classe) {
    this.classe = classe;
    s = new HashSet<Strumentista>();
  }

  public String getClasse() { return classe; }
  public String toString() { return classe + ": " + s; }

  public boolean equals(Object o) { return equals((Banda) o); }
  public boolean equals(Banda b){ return classe.equals(b.classe); }

  public boolean aggiungi(Strumentista s) { return this.s.add(s); }

  public int strumento(String strumento) {
    int result = 0;
    for(Strumentista s: this.s)
      if(s.getStrumento().equals(strumento)) result++;
    return result;
  }

  public Strumentista piuGiovane() {
    Strumentista min=null;
    for(Strumentista s: this.s) if(min==null||s.compareTo(min)>0) min=s;
    return min;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Banda> l = new ArrayList<Banda>();
    Scanner scanner = new Scanner(System.in);
    Banda b = new Banda(scanner.next());
    l.add(0,b);
    Strumentista s = new Strumentista(scanner.nextLine(),
        scanner.nextLine(), scanner.nextLine(), scanner.nextInt(),
        scanner.nextInt(), scanner.nextInt());
    if(!b.aggiungi(s)) System.out.println("Inserimento non avvenuto!");
    for(Banda x: l)
      if(x.strumento("violino")==0) System.out.println(x);
    Strumentista giovane = null;
    Banda min = null;
    for(Banda x: l) {
      Strumentista g = x.piuGiovane();
      if(giovane ==null || g.compareTo(giovane)>0) { giovane=g; min=b;}
    }
    System.out.println(min.getClasse());
  }
}
```