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

Appello del 12/9/2011

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

Esercizio 1 (4 punti)

1. Discutere il concetto di ereditarietà.
2. Descrivere le principali differenze tra array e collezioni in Java.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **0,0235** supponendo di utilizzare 8 bit per la mantissa (in modulo e segno) ed 8 bit per l'esponente (in complemento a 2).

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

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

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

1. Calcolare la complessità in passi base del metodo `f` nei termini del parametro `M` (si distinguano i casi in cui `M` 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` pari e si esprima `j=2 i+1`).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (5 punti)

Il comune di Collate (Parma) ha fornito la concessione per il servizio di trasporto scolastico alla ditta "Alt! E sali su me!". Tale azienda ha deciso di informatizzare la gestione dei pullmini che si occupano di trasportare i bambini a scuola e delle relative fermate. A tal scopo, ogni fermata viene memorizzata in un calcolatore, registrandone il nome, l'indirizzo ed il numero di bambini che verranno caricati in tale fermata. Si scriva una classe `Fermata` per la ditta "Alt! E sali su me!", 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 della fermata.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Fermata` (l'uguaglianza va verificata solamente sul nome della fermata).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Fermata` passato come parametro (la precedenza va data alla fermata che presenta un numero di bambini superiore e, in caso di parità, si procede in ordine alfabetico per nome).

Esercizio 5 (8 punti)

Si scriva una classe `Pullmino` che memorizzi le informazioni relative ai pullmini utilizzati per il trasporto. Per ogni pullmino vengono memorizzati la targa, il cognome dell'autista e la capienza massima. Inoltre, le fermate previste per ogni pullmino vanno inserite all'interno di una lista. La classe `Pullmino` deve inoltre:

1. Presentare un opportuno costruttore con parametri (inizialmente, un pullmino non presenta alcuna fermata).
2. Presentare opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Possedere il metodo `toString` che fornisca la descrizione del pullmino (inclusa la descrizione di tutte le fermate).
4. Presentare il metodo `postiLiberi` che restituisca il numero di posti ancora liberi.
5. Possedere il metodo `aggiungi` che, dato un oggetto `Fermata`, lo inserisca in coda alla lista di fermate, controllando che tale inserimento non violi il vincolo di capienza del pullmino.
6. Presentare il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Pullmino` (l'uguaglianza va verificata solamente sulla targa).
7. Possedere il metodo `servita` che, dato il nome di una fermata, indichi se tale fermata è servita dal pullmino.

Esercizio 6 (7 punti)

Si scriva un'applicazione per la ditta "Alt! E Sali su me!" che:

1. Crei un insieme di oggetti `Pullmino`.
2. Crei un oggetto `Pullmino`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. all'interno dell'insieme di cui al punto 1., verificando che tale insieme non contenga già un oggetto uguale.
4. Crei un oggetto `Fermata`, lette da tastiera le informazioni necessarie.
5. Inserisca la fermata creata al punto 4. nel primo pullmino, tra quelli contenuti nell'insieme di cui al punto 1., in grado di servire tale fermata.
6. Stampi a video la descrizione di tutti i pullmini che hanno esaurito la propria capacità.
7. Stampi a video la targa del pullmino, tra quelli contenuti nell'insieme di cui al punto 1., avente più posti liberi.

Soluzione Esercizio 2

$0{,}0235_{10} = 0{,}000001100000_2$ quindi la mantissa è **(0)1100000**, l'esponente -5_{10} = **11111011**.

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`sum+=V[i++]`	M/2	o M/2 + 1/2
`++i<M/2`	M/2	o M/2 + 1/2
Totale	M + 2	o M + 3

Domanda 2:

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

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

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 4

```
class Fermata implements Comparable<Fermata> {
  private String nome, indirizzo;
  private int bambini;

  public Fermata(String nome, String indirizzo, int bambini) {
    this.nome=nome;
    this.indirizzo=indirizzo;
    this.bambini=bambini;
  }

  public String getNome() { return nome; }
  public String getIndirizzo() { return indirizzo; }
  public int getBambini() { return bambini; }

  public String toString() {
    return nome+ " (" + indirizzo + "): " + bambini;
  }

  public boolean equals(Object o) { return equals((Fermata) o); }
  public boolean equals(Fermata f) { return nome.equals(f.nome); }

  public int compareTo(Fermata f) {
    int ret=f.bambini-this.bambini;
    if(ret==0) ret=this.nome.compareTo(f.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Pullmino {
  private String targa, autista;
  private int capienza, postiliberi;
  private List<Fermata> l;

  public Pullmino(String targa, String autista, int capienza) {
    this.targa=targa;
    this.autista=autista;
    this.capienza=capienza;
    this.postiliberi=capienza;
    l=new ArrayList<Fermata>();
  }

  public String getTarga() { return targa; }
  public String getAutista() { return autista; }
  public int postiLiberi() { return postiliberi; }

  public String toString() {
    return targa + " (" + autista + "):" + l.toString();
  }

  public boolean aggiungi(Fermata f) {
    if(postiliberi>=f.getBambini()) {
      l.add(f);
      postiliberi-=f.getBambini();
      return true;
    }
    return false;
  }

  public boolean equals(Object o) { return equals((Pullmino) o); }
  public boolean equals(Pullmino p) { return targa.equals(p.targa); }

  public boolean servita(String nome) {
    for(Fermata f:l) if(f.getNome().equals(nome)) return true;
    return false;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Pullmino> s=new TreeSet<Pullmino>();
    Scanner scanner=new Scanner(System.in);
    Pullmino p=new Pullmino(scanner.next(),scanner.next(),
      scanner.nextInt());
    if(!s.add(p)) System.out.println("Pullmino già presente!");
    Fermata f=new Fermata(scanner.next(), scanner.next(),
      scanner.nextInt());
    for(Pullmino x: s) if(x.aggiungi(f)) break;
    for(Pullmino x: s) if(x.postiLiberi()==0) System.out.println(x);
    Pullmino max=null;
    for(Pullmino x:s)
      if(max==null || x.postiLiberi()>max.postiLiberi()) max=x;
    System.out.println(max.getTarga());
  }
}
```