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

Appello del 18/9/2015

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

Esercizio 1 (4 punti)

1. Illustrare il ciclo di esecuzione di un'istruzione da parte della CPU (fetch/decode/execute).
2. Descrivere le principali differenze tra array e collezioni in Java.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **173,0** 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 N) {
  int j=N, sum=0;
  for (; j>0; )
    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 minori o uguali a `N` da quelli in cui assume valori maggiori di `N`).
2. Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N` pari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (5 punti)

In vista delle sue prossime nozze, il dott. Leonardo Di Francesco desidera organizzare al meglio il ricevimento nuziale. A tal scopo, le informazioni relative a ogni invitato vanno inserite all'interno di un calcolatore. In particolare, occorre memorizzare il nome, l'indirizzo e il numero di invitati (ogni invitato, infatti, può venire accompagnato da parenti, figli, ecc.). Si scriva una classe `Invitato` per il dott. Di Francesco 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'invitato.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Invitato` (la verifica va fatta unicamente per nome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Invitato` passato come parametro (in ordine alfabetico per nome e, a parità, in ordine crescente di numero di invitati).

Esercizio 5 (8 punti)

Si scriva una classe `Tavolo` che memorizzi le informazioni riguardanti i tavoli in cui saranno sistemati gli invitati. Per ciascun tavolo occorre memorizzare il numero e i posti a sedere disponibili, mentre gli invitati vanno memorizzati all'interno di un insieme. La classe `Tavolo` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, al tavolo non siedono invitati).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del tavolo (inclusa la descrizione di tutti gli invitati).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Tavolo` (la verifica va effettuata unicamente sul numero).
5. Presentare il metodo `postiLiberi` che indichi il numero di posti a sedere non ancora occupati da invitati.
6. Possedere il metodo `aggiungi` che, dato un oggetto `Invitato`, lo inserisca all'interno del tavolo, controllando anche che vi siano posti sufficienti.
7. Presentare il metodo `invitato` che, dato il nome di un invitato, indichi se esso è incluso nel tavolo.
8. Possedere il metodo `cancella` che, dato un oggetto `Invitato`, lo cancelli dall'elenco degli invitati, se esso è presente all'interno del tavolo (il metodo deve indicare se la cancellazione è avvenuta o meno).

Esercizio 6 (7 punti)

Si scriva un'applicazione per il dott. Di Francesco che:

1. Crei una lista di oggetti `Tavolo`.
2. Crei un oggetto `Tavolo`, 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 `Invitato`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. nel primo tavolo dell'insieme di cui al punto 1. in grado di contenerlo.
6. Letto da tastiera il nome di un invitato stampi a video la descrizione del tavolo (se esiste) in cui tale invitato è inserito.
7. Lette da tastiera le informazioni di un invitato lo sposti dal tavolo in cui si trova attualmente al tavolo di cui al punto 2. (controllando che ci siano posti liberi a sufficienza).

Soluzione Esercizio 2

$173{,}0_{10}$ = $10101101{,}0_2$ quindi la mantissa è **(0)1010110**, l'esponente 8_{10} = **00001000**.
Il numero rappresentato è 172, diverso dall'originale poiché quest'ultimo possiede 8 cifre rappresentative.

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`j>0`	$N+1$
`sum+=f(V, N, j--)`	N
complessità di `f`	N^2+N+2
Totale	N^2+3N+5

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

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

Soluzione Esercizio 4

```
class Invitato implements Comparable<Invitato> {
  private String nome, indirizzo;
  private int invitati;

  public Invitato(String nome, String indirizzo, int invitati) {
    this.nome = nome;
    this.indirizzo = indirizzo;
    this.invitati = invitati;
  }

  public String getNome() { return nome; }
  public String getIndirizzo() { return indirizzo; }
  public int getInvitati() { return invitati; }

  public String toString() {
    return nome + ", " + indirizzo + " (" + invitati + ")";
  }

  public boolean equals(Object o) { return equals((Invitato) o); }
  public boolean equals(Invitato i) { return this.nome.equals(i.nome); }

  public int compareTo(Invitato i) {
    int ret = this.nome.compareTo(i.nome);
    if(ret==0) ret = this.invitati - i.invitati;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Tavolo {
  private Set<Invitato> s;
  private int numero, posti;

  public Tavolo(int numero, int posti) {
    this.numero = numero;
    this.posti = posti;
    s = new HashSet< Invitato >();
  }
  public int getNumero() { return numero; }

  public String toString() { return numero + " (" + posti + "):" + s; }

  public boolean equals(Object o) { return equals((Tavolo) o); }
  public boolean equals(Tavolo t) { return this.numero == t.numero; }

  public int postiLiberi() {
    int n=posti;
    for(Invitato i: s) n-=i.getInvitati();
    return n;
  }

  public boolean aggiungi(Invitato i) {
    if(i.getInvitati()>postiLiberi()) return false;
    return s.add(i);
  }

  public boolean invitato(String nome) {
    for(Invitato i: s) if(i.getNome().equals(nome)) return true;
    return false;
  }

  public boolean cancella(Invitato i) { return s.remove(i); }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Tavolo> l = new LinkedList<Tavolo>();
    Scanner scanner = new Scanner(System.in);
    Tavolo t = new Tavolo(scanner.nextInt(), scanner.nextInt());
    l.add(t);
    Invitato i = new Invitato(scanner.nextLine(), scanner.nextLine(),
        scanner.nextInt());
    for(Tavolo x: l) if(x.aggiungi(i)) break;
    String nome = scanner.nextLine();
    for(Tavolo x: l) if(x.invitato(nome)) System.out.println(x);
    i = new Invitato(scanner.nextLine(), scanner.nextLine(),
        scanner.nextInt());
    for(Tavolo x: l) if(x.cancella(i)) break;
    if(!t.aggiungi(i)) System.out.println("Tavolo pieno!");
  }
}
```