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

Appello del 16/10/2009

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

Esercizio 1 (4 punti)

1. Discutere l'utilizzo di metodi e proprietà statiche in Java.
2. Descrivere le principali differenze tra memoria principale e secondaria.

Esercizio 2 (2 punti)

Convertire in binario i numeri **57** e **–93**, supponendo di utilizzare una rappresentazione a 8 bit in complemento a 2. Si esegua infine la somma dei due numeri, riconvertendo il risultato in base 10.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

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

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

1. Calcolare la complessità in passi base del metodo `f` nei termini dei parametri `M` ed `N` (suggerimento: si distinguano i casi in cui `M` assume valori minori di `N` da quelli in cui assume valori maggiori o uguali a `N`).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `M` ed `N` (suggerimento: si supponga `N<M`).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (6 punti)

Il vivaio "Clorofilla" vuole organizzare all'interno di un calcolatore le informazioni sulle piante che vengono coltivate dall'azienda. In particolare, per ogni pianta viene registrata la specie, l'anno di produzione ed il prezzo di vendita. Si scriva una classe `Pianta` per il vivaio "Clorofilla" 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 la descrizione della pianta.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Pianta` (l'uguaglianza va verificata sulla specie e sull'anno di produzione).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Pianta` passato come parametro (la precedenza va verificata in ordine alfabetico per specie e, a parità di specie, per età crescente).

Esercizio 5 (8 punti)

Si scriva una classe `Serra` che memorizzi le informazioni relative alle piante contenute all'interno di ciascuna serra. Oltre al codice della serra, occorre memorizzare i dati relativi alle piante all'interno di un vettore. La classe `Serra` deve inoltre:

1. Presentare un opportuno costruttore (il numero massimo di piante che la serra è in grado di contenere deve essere passato come parametro).
2. Possedere un metodo `aggiungi` che, dato un oggetto `Pianta`, lo inserisca all'interno del vettore, controllando che tale inserimento sia possibile.
3. Presentare un metodo `cerca` che, dato il nome di una specie di pianta, restituisca l'oggetto `Pianta` di tale specie più giovane, se almeno un esemplare di tale specie è presente nella serra.
4. Possedere un metodo `valoreTotale` che restituisca il prezzo totale di tutte le piante contenute all'interno della serra.
5. Possedere il metodo `toString` che restituisca una stringa che fornisca una descrizione della serra, compresi i dati di tutte le piante ivi contenute.

Esercizio 6 (7 punti)

Si scriva un'applicazione per il vivaio "Clorofilla" che:

1. Crei una lista di oggetti `Serra`.
2. Crei un oggetto `Serra`, 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. Lette da tastiera le informazioni su una nuova `Pianta`, inserisca tale pianta nella prima serra, tra quelle nella lista di cui al punto 1., in grado di contenerla.
5. Letto da tastiera il nome di una specie, stampi le informazioni della serra, tra quelle nella lista di cui al punto 1., che contiene l'esemplare più giovane di tale specie.

Soluzione Esercizio 2

$57_{10} = 00111001_2$
$-93_{10} = 10100011_2$

$11011100_2 = -36_{10}$

```
00111001+
10100011
11011100
```

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`i<N`	`N-M`+1	o 1
`sum+=V[i++]`	`N-M`	o 0
Totale	2 `N` – 2 `M` + 3	o 3

Domanda 2:

2 assegnamenti	2
`while(j<M)`	`M` + 1
`sum+=f(V, ++j, N)`	`M`
complessità di `f`	N^2 - `N` + 3 `M`
Totale	N^2 - `N` + 5 `M` + 3

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

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

Soluzione Esercizio 4

```
class Pianta implements Comparable<Pianta> {
  private String specie;
  private int anno;
  private float prezzo;

  public Pianta(String specie, int anno, float prezzo) {
    this.specie=specie;
    this.anno=anno;
    this.prezzo=prezzo;
  }

  public String getSpecie() { return specie; }
  public int getAnno() { return anno; }
  public float getPrezzo() { return prezzo; }

  public String toString() {
    return specie + "(" + anno + "): " + prezzo;
  }

  public boolean equals(Object o) { return equals((Pianta) o); }
  public boolean equals(Pianta p) {
    return (specie.equals(p.specie)&&anno==p.anno);
  }

  public int compareTo(Pianta p) {
    int ret=specie.compareTo(p.specie);
    if(ret==0) ret=p.anno-anno;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
class Serra {
  private String codice;
  private int quante;
  private Pianta piante[];
  public Serra(String codice, int max) {
    this.codice=codice; quante=0;
    piante=new Pianta[max];
  }
  public boolean aggiungi(Pianta p) {
    if(quante>=piante.length) return false;
    piante[quante++]=p;
    return true;
  }
  public Pianta cerca(String s) {
    int anno=-1;
    Pianta max=null;
    for(int i=0; i<quante; i++)
      if(piante[i].getSpecie().equals(s)&&piante[i].getAnno()>anno) {
        anno=piante[i].getAnno(); max=piante[i];
      }
    return max;
  }
  public float valoreTotale() {
    float v=0;
    for(int i=0; i<quante; i++) v+=piante[i].getPrezzo();
    return v;
  }
  public String toString() {
    String s="Serra " + codice + ":\n";
    for(int i=0; i<quante; i++) s+=piante[i].toString()+"\n";
    return s;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    List<Serra> l=new LinkedList<Serra>();
    Scanner scanner=new Scanner(System.in);
    l.add(new Serra(scanner.next(), scanner.nextInt()));
    int i=0;
    Pianta p=new Pianta(scanner.next(), scanner.nextInt(),
      scanner.nextFloat());
    while(i<l.size()&&!l.get(i).aggiungi(p)) i++;
    String specie=scanner.next();
    Serra max=null;
    int anno=0;
    for(Serra s: l) {
      p=s.cerca(specie);
      if(p!=null&&p.getAnno()>anno) {
        anno=p.getAnno(); max=s;
      }
    }
    System.out.println(max);
  }
}
```