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

Appello del 15/6/2012

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

Esercizio 1 (4 punti)

1. Descrivere l'architettura egli elaboratori secondo Von Neumann.
2. Discutere il concetto di interfaccia in Java, evidenziandone le differenze con una classe astratta.

Esercizio 2 (2 punti)

Convertire in binario i numeri **-71** e **–38**, 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, motivando eventuali differenze tra il risultato ottenuto e quello atteso.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[], int N) {
  int sum=0;
  for(int i=N; i-->=0; )
    sum+=V[--i];
  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 `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 (6 punti)

La ditta "B-cool" si occupa dell'installazione e manutenzione di condizionatori di ultima generazione. A tal scopo, i dati relativi ai condizionatori installati vengono memorizzati registrando la marca ed il modello del condizionatore, l'anno e l'indirizzo di installazione. Si scriva una classe `Condizionatore` per la "B-cool" 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 del condizionatore.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Condizionatore` (l'uguaglianza va verificata sulla marca e sul modello del condizionatore).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Condizionatore` passato come parametro (la precedenza va data per ordine alfabetico su marca e modello, in caso di parità si procede per data di installazione crescente).

Esercizio 5 (7 punti)

Si scriva una classe `Squadra` che memorizzi le informazioni relative ai condizionatori che necessitano di manutenzione da parte di una specifica squadra di installatori della "B-cool". Per ogni squadra vengono memorizzati il nome della squadra ed una lista che include i condizionatori associati a tale squadra. La classe `Squadra` deve inoltre:

1. Presentare un opportuno costruttore con parametri (inizialmente, una squadra non ha nessun condizionatore da gestire).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della squadra (inclusa la descrizione di tutti i condizionatori ad essa associati).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Condizionatore`, lo inserisca all'interno della lista, mantenendo la lista ordinata secondo il punto 5. dell'esercizio 4.
5. Presentare il metodo `quanti` che restituisca il numero di condizionatori associati alla squadra.
6. Possedere il metodo `cerca` che restituisca una lista contenente tutti i condizionatori associati alla squadra e installati non prima dell'anno passato come parametro al metodo.

Esercizio 6 (8 punti)

Si scriva un'applicazione per la ditta "B-cool" che:

1. Crei un insieme di oggetti `Squadra`.
2. Crei un oggetto `Squadra`, 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 `Condizionatore`, lette da tastiera le informazioni necessarie.
5. Inserisca il condizionatore creato al punto 4. tra quelli associati alla squadra di cui al punto 2.
6. Stampi a video il nome della squadra che ha associati più condizionatori tra quelle dell'insieme di cui al punto 1.
7. Stampi a video la descrizione di tutti i condizionatori installati non prima del 2000.

Soluzione Esercizio 2

$-71_{10} = 10111001_2$
$-38_{10} = 11011010_2$

$10010011_2 = -109_{10}$

```
10111001+
11011010
10010011
```

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`i-->=0`	N/2 + 2	o (N+3)/2
`sum+=V[--i]`	N/2 + 1	o (N+1)/2
Totale	N + 5	o N + 4

Domanda 2:

2 assegnamenti	2
`while(++j<N)`	N
`sum+=f(V, N, j)`	N – 1
complessità di `f`	$N^2/2 + 4N - 9/2$
Totale	$N^2/2 + 6N - 7/2$

complessità di `f`: $\sum_{\substack{j=1 \\ (j \text{ pari})}}^{N-1} (j+5) + \sum_{\substack{j=1 \\ (j \text{ dispari})}}^{N-1} (j+4) = \sum_{j=1}^{N-1} j + 5\frac{N-1}{2} + 4\frac{N-1}{2} = \frac{N^2}{2} + 4N - \frac{9}{2}$

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

Soluzione Esercizio 4

```
class Condizionatore implements Comparable<Condizionatore> {
  private String marca, modello, indirizzo;
  private int anno;

  public Condizionatore (String marca, String modello, String indirizzo,
    int anno) {
    this.marca=marca;
    this.modello=modello;
    this.indirizzo=indirizzo;
    this.anno=anno;
  }
  public String getMarca() { return marca; }
  public String getModello() { return modello; }
  public String getIndirizzo() { return indirizzo; }
  public int getAnno() { return anno; }
  public String toString() {
    return marca + ", " + modello + "(" + indirizzo + "," + anno+ ")";
  }
  public boolean equals(Object o) { return equals((Condizionatore) o); }
  public boolean equals(Condizionatore c) {
    return marca.equals(c.marca) && modello.equals(c.modello);
  }
  public int compareTo(Condizionatore c) {
    int ret=marca.compareTo(c.marca);
    if(ret==0) ret=modello.compareTo(c.modello);
    if(ret==0) ret=anno-c.anno;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
class Squadra {
  private String nome;
  private List<Condizionatore> l;
  public Squadra(String nome) {
    this.nome=nome;
    l=new ArrayList<Condizionatore>();
  }
  public String getNome() { return nome; }
  public String toString() {
    return nome + ":" + l.toString();
  }
  public void aggiungi(Condizionatore c) {
    int i=0;
    while((i<l.size())&&(l.get(i).compareTo(c)<0)) i++;
    l.add(i, c);
  }
  public int quanti() { return l.size(); }
  public List<Condizionatore> cerca(int anno) {
    List<Condizionatore> res=new LinkedList<Condizionatore>();
    for(Condizionatore c:l) if(c.getAnno()>=anno) res.add(c);
    return res;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Squadra> s=new TreeSet<Squadra>();
    Scanner scanner=new Scanner(System.in);
    Squadra q=new Squadra(scanner.next());
    if(!s.add(q)) System.out.println("Squadra già presente!");
    Condizionatore c=new Condizionatore(scanner.nextLine(), scanner.next(),
      scanner.nextLine(), scanner.nextInt());
    q.aggiungi(c);
    Squadra max=null;
    for(Squadra x: s) {
      if(max==null || x.quanti()>max.quanti())
        max=x;
    }
    if(max!=null) System.out.println(max.getNome());
    for(Squadra x: s) System.out.println(x.cerca(2000));
  }
}
```