Esame di Fondamenti di Informatica L-B
Ingegneria Gestionale e dei Processi Gestionali (L-Z)

Appello del 16/6/2011

Esercizio 1 (4 punti)

Descrivere le modalità di studio della complessità temporale di un algoritmo.

Esercizio 2 (6 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, sum=0;
  for(j=1; j<=N; j++)
    sum+=f(V, ++j);
  return sum;
}
```

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

Esercizio 3 (5 punti)

Il Museo Opere Riciclate di Bologna (MORBO), di recentemente costituzione e specializzato in opere del secolo scorso, deve essere all'avanguardia per quanto riguarda la catalogazione e l'esposizione delle opere in esso esposte. A tal scopo, ogni opera viene catalogata in un computer, memorizzandone il titolo, il nome dell'autore, la corrente artistica e l'anno di realizzazione. Si scriva una classe `Opera` per il MORBO, 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 dell'opera.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Opera` (l'uguaglianza va verificata sul titolo e sul nome dell'autore).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Opera` passato come parametro (la precedenza va data per ordine alfabetico della corrente artistica e, in caso di parità, si procede per anno di produzione decrescente).

Esercizio 4 (8 punti)

Si scriva una classe `Sala` che memorizzi le informazioni relative alle opere esposte in ciascuna sala del museo. Oltre a memorizzare il numero della sala, le opere vanno inserite all'interno di un insieme. La classe `Sala` deve inoltre:

1. Presentare un opportuno costruttore (inizialmente una sala non contiene alcun'opera).
2. Presentare il metodo `getNumero` che restituisca il numero della sala.
3. Possedere il metodo `toString` che fornisca la descrizione della sala (inclusa la descrizione di tutte le opere ivi esposte).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Opera`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
5. Presentare il metodo `corrente` che, dato il nome di una corrente artistica, indichi se la sala contiene almeno un'opera di tale corrente.
6. Possedere il metodo `listaMuseale` che, dato il nome di un autore, restituisca una lista contenente tutte le opere di tale autore presenti nella sala.

Esercizio 5 (7 punti)

Si scriva un'applicazione per il MORBO che:

1. Crei una lista di oggetti `Sala`.
2. Crei un oggetto `Sala`, 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 `Opera`, lette da tastiera le informazioni necessarie, e lo aggiunga alle opere incluse nella sala di cui al punto 2., indicando se l'inserimento è andato a buon fine o meno.
5. Letto da tastiera il nome di un autore, stampi a video la descrizione dell'opera più recente realizzata da tale autore ed esposta all'interno della sala di cui al punto 2.
6. Letto da tastiera il nome di una corrente artistica, stampi a video il numero di sale che contengono opere relative a tale corrente.

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `double leggiDouble()` Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` Legge un intero (delimitato da spazi).
- `String leggiLinea()` Legge una linea di testo.
- `String leggiString()` Legge una parola senza spazi al suo interno.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2
`sum+=V[i]`	`M`
`++i<M`	`M`
Totale	`2M + 2`

Domanda 2:

2 assegnamenti	2
`j<=N`	`(N+1)/2 + 1`
`sum+=f(V, N, ++j)`	`(N+1)/2`
`j++`	`(N+1)/2`
complessità di `f`	$N^2/2 + 3N + 5/2$
Totale	$N^2/2 + 9N/2 + 7$

complessità di `f`: $\sum_{\substack{j=2 \\ (j \text{ pari})}}^{N+1}(2j+2) = \sum_{i=1}^{(N+1)/2}(4i+2) = \frac{4}{2}\frac{N+1}{2}\frac{N+3}{2} + N + 1 = \frac{N^2}{2} + 3N + \frac{5}{2}$

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

Soluzione Esercizio 3

```
class Opera implements Comparable<Opera> {
  private String titolo, autore, corrente;
  private int anno;

  public Opera(String titolo, String autore, String corrente, int anno) {
    this.titolo=titolo;
    this.autore=autore;
    this.corrente=corrente;
    this.anno=anno;
  }

  public String getTitolo() { return titolo; }
  public String getAutore() { return autore; }
  public String getCorrente() { return corrente; }
  public int getAnno() { return anno; }

  public String toString() {
    return titolo+ " (" + autore + ", " + anno + "): " + corrente;
  }

  public boolean equals(Object o) { return equals((Opera) o); }
  public boolean equals(Opera o) {
    return titolo.equals(o.titolo) && autore.equals(o.autore);
  }

  public int compareTo(Opera o) {
    int ret=corrente.compareTo(o.corrente);
    if(ret==0) ret=o.anno-this.anno;
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Sala {
  private int numero;
  private Set<Opera> opere;

  public Sala(int numero) {
    this.numero=numero;
    opere=new HashSet<Opera>();
  }

  public int getNumero() { return numero; }

  public String toString() {
    return numero + ":" + opere.toString();
  }

  public boolean aggiungi(Opera o) { return opere.add(o); }

  public boolean corrente(String corrente) {
    for(Opera o:opere)
      if(o.getCorrente().equals(corrente)) return true;
    return false;
  }

  public List<Opera> listaMuseale(String nome) {
    List<Opera> l=new ArrayList<Opera>();
    for(Opera o:opere)
      if(o.getAutore().equals(nome)) l.add(o);
    return l;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;

class Applicazione {
  public static void main(String[] args) {
    List<Sala> l=new LinkedList<Sala>(); // domanda 1
    Sala s=new Sala(Lettore.in.leggiInt()); // domanda 2
    l.add(s); // domanda 3
    if(!s.aggiungi(new Opera(Lettore.in.leggiLinea(),
      Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
      Lettore.in.leggiInt())))
        System.out.println("Opera già presente!"); // domanda 4
    String nome= Lettore.in.leggiLinea();
    List<Opera> opere=s.listaMuseale(nome);
    Opera recente=null;
    for(Opera o:opere) // domanda 5
      if(recente==null||o.getAnno()>recente.getAnno())
        recente=o;
    if(recente!=null) System.out.println(recente); // domanda 5
    String corrente= Lettore.in.leggiLinea();
    int n=0;
    for(Sala sala:l)
      if(sala.corrente(corrente)) n++;
    System.out.println(n); // domanda 6
  }
}
```