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

Appello del 16/1/2015

Esercizio 1 (4 punti)

Gestione ed utilizzo di insiemi in Java.

Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

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

public static int g(int V[], int N) {
  int j, sum=0;
  for(j=0; j++<N; ++j)
    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` pari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 3 (5 punti)

La redazione del giornale satirico "Carlos Semanal" vuole informatizzare la gestione delle strisce fumettistiche satiriche pubblicate all'interno di ogni fascicolo. A tal scopo, per ogni striscia vengono memorizzati il titolo, una descrizione, il nome del disegnatore e la lunghezza in pagine. Si scriva una classe `Striscia` per il giornale "Carlos Semanal" 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 della striscia.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Striscia` (la verifica va fatta su titolo e disegnatore).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Striscia` passato come parametro (in ordine alfabetico per disegnatore e, a parità, la precedenza va data alla striscia più lunga).

Esercizio 4 (7 punti)

Si scriva una classe `Fascicolo` che memorizzi le informazioni relative alle strisce pubblicate in un numero del giornale. Per ciascun fascicolo occorre memorizzare il numero progressivo, la quantità di copie stampate e la data di pubblicazione, mentre le strisce vanno memorizzate all'interno di una lista. La classe `Fascicolo` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la lista di strisce è vuota).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del fascicolo (inclusa la descrizione di tutte le strisce contenute).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Fascicolo` (il numero progressivo è univoco).
5. Implementare l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Fascicolo` passato come parametro (in ordine crescente di data di pubblicazione).
6. Presentare il metodo `aggiungi` che, dato un oggetto `Striscia`, lo inserisca all'interno della lista, mantenendo la lista ordinata secondo il punto 5. dell'esercizio 3.
7. Possedere il metodo `disegnatore` che, dato il nome di un disegnatore, indichi se una sua striscia sia presente all'interno del fascicolo.

Esercizio 5 (8 punti)

Si scriva un'applicazione per il giornale "Carlos Semanal" che:

1. Crei un insieme di oggetti `Fascicolo`.
2. Crei un oggetto `Fascicolo`, 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 `Striscia`, lette da tastiera le informazioni necessarie.
5. Inserisca la striscia creata al punto 4. tra quelle contenute nel fascicolo di cui al punto 2.
6. Letti da tastiera il nome di un disegnatore, stampi a video la data di tutti i fascicoli che presentano strisce da lui disegnate.
7. Stampi a video le informazioni del fascicolo più recente, tra quelli dell'insieme di cui al punto 1., che contiene strisce disegnate dal disegnatore il cui nome è stato letto al punto 6.

Soluzione Esercizio 2

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`j++<N`	N + 1
`sum+=f(V, j--)`	N
`++j`	N
complessità di `f`	$N^2/2 + 3N$
Totale	$N^2/2 + 6N + 3$

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

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

Soluzione Esercizio 3

```
class Striscia implements Comparable<Striscia> {
  private String titolo, descr, dis;
  private int l;

  public Striscia(String titolo, String descr, String dis, int l) {
    this.titolo = titolo;
    this.descr = descr;
    this.dis = dis;
    this.l = l;
  }

  public String getTitolo() { return titolo; }
  public String getDescrizione() { return descr; }
  public String getDisegnatore() { return dis; }
  public int getLunghezza() { return l; }

  public String toString() {
    return titolo+ " (" + descr + "): " + dis+ ", " + l;
  }

  public boolean equals(Object o) { return equals((Striscia) o); }
  public boolean equals(Striscia s) {
    return titolo.equals(s.titolo) && dis.equals(s.dis);
  }

  public int compareTo(Striscia s) {
    int ret = this.dis.compareTo(s.dis);
    if(ret==0) ret = s.l - this.l;
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Fascicolo implements Comparable<Fascicolo> {
  private List<Striscia> l;
  private int num, g, m, a, copie;

  public Fascicolo(int num, int g, int m, int a, int copie) {
    this.num = num;
    this.g = g; this.m = m; this.a = a;
    this.copie = copie;
    l = new ArrayList<Striscia>();
  }

  public String getData() { return g + "/" + m + "/" + a; }
  public int getNumero() { return num; }
  public int getCopie() { return copie; }
  public String toString() {
    return num + "( " + getData() + "):" + l.toString();
  }
  public boolean equals(Object o) { return equals((Fascicolo) o); }
  public boolean equals(Fascicolo f) { return num==f.num; }
  public int compareTo(Fascicolo f) {
    int ret = this.a - f.a;
    if(ret==0) ret = this.m - f.m;
    if(ret==0) ret = this.g - f.g;
    return ret;
  }

  public void aggiungi(Striscia s) {
    int i = 0;
    while((i<l.size())&&(l.get(i).compareTo(s)<0)) i++;
    l.add(i, s);
  }
  public boolean disegnatore(String nome) {
    for(Striscia s: l) if(s.getDisegnatore().equals(nome)) return true;
    return false;
  }
}
```

Soluzione Esercizio 5

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

class Applicazione {
  public static void main(String[] args) {
    Set<Fascicolo> s=new HashSet<Fascicolo>();
    Fascicolo f=new Fascicolo(Lettore.leggiInt(), Lettore.leggiInt(),
      Lettore.leggiInt(), Lettore.leggiInt(), Lettore.leggiInt());
    if(!s.add(f)) System.out.println("Fascicolo già esistente!");
    Striscia x=new Striscia(Lettore.leggiLinea(), Lettore.leggiLinea(),
      Lettore.leggiLinea(), Lettore.leggiInt());
    f.aggiungi(x);
    String nome = Lettore.leggiLinea();
    Fascicolo max = null;
    for(Fascicolo n: s) if(n.disegnatore(nome)) {
      System.out.println(n.getData());
      if(max==null || n.compareTo(max)>0) max=n;
    }
    System.out.println(max);
  }
}
```