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

Appello del 17/1/2008

Esercizio 1 (4 punti)

Modalità di gestione delle stringhe in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M, int N) {
  int res=0, i=M-1;
  while(i>=N) {
    res+=V[i];
    i--;
    if(i<0) return;
  }
  return res;
}

int g(int V[], int N) {
  int j, res=1;
  for(j=0; j<N; j++)
    res*=f(V, N, j);
  return res;
}
```

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

Esercizio 3 (5 punti)

Per facilitare la gestione degli esami sostenuti dai propri studenti, il Prof. Croma Pellata dell'università "La Sorbolona" ha deciso di memorizzare i dati dei propri appelli d'esame in forma elettronica. A tal fine, ogni esame sostenuto da uno studente viene caratterizzato dai dati dello studente (nome, cognome e numero di matricola), dal voto ottenuto e dall'esito (ad es., promosso, respinto, ritirato).

Si scriva una classe `Esame` per il Prof. Croma Pellata che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi per accedere alle proprietà dell'esame.
3. Presenti il metodo `toString` che fornisca una descrizione testuale dell'esame.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Esame` (l'uguaglianza va verificata unicamente sul numero di matricola).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un altro oggetto `Esame`; in particolare, la precedenza è determinata, nell'ordine, per cognome, per nome e, infine, per numero di matricola.

Esercizio 4 (8 punti)

Si scriva una classe `Appello` che memorizzi le informazioni relative ad un determinato appello di esame. Per ogni appello occorre memorizzare, oltre alla lista degli esami sostenuti in tale appello, l'anno accademico ed il numero della sessione d'esame (le sessioni sono al massimo 8 per A.A.). La classe `Appello` deve inoltre:

1. Possedere un opportuno costruttore (ovviamente, all'atto della creazione, l'appello non è stato sostenuto da alcuno studente).
2. Possedere un metodo `aggiungi` che, dato un oggetto `Esame`, lo inserisca all'interno della lista, se questo non esiste già, mantenendo la lista ordinata secondo il punto 5. dell'esercizio 3 (suggerimento: si utilizzi il metodo `add(int i, Esame e)` della classe `List<Esame>`).
3. Presentare il metodo `getEsito` che, data la matricola di uno studente, restituisca l'esito dell'appello per tale studente, se questo ha sostenuto l'appello.
4. Possedere il metodo `media` che calcoli la media ottenuta dagli studenti nell'appello, non considerando gli studenti ritirati (si supponga che per ogni appello esista almeno uno studente non ritirato).
5. Presentare il metodo `toString` che restituisca una descrizione dell'appello.

Esercizio 5 (8 punti)

Si scriva un'applicazione per il Prof. Croma Pellata che:

1. Crei un vettore di oggetti di tipo `Appello` per un certo anno accademico.
2. Lette da tastiera le informazioni relative ad un nuovo appello, provveda ad inserire tale oggetto in coda al vettore (che si deve supporre possa contenere già alcuni oggetti).
3. Lette da tastiera le informazioni relative ad alcuni esami sostenuti, provveda ad inserire tali oggetti all'interno dell'appello creato in precedenza.
4. Letto da tastiera il numero di matricola di uno studente, stampi a video tutti gli esiti relativi a tale studente per gli appelli contenuti nel vettore.
5. Stampi su video le informazioni relative all'appello che presenta la media inferiore tra tutti quelli all'interno del vettore.

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:

- `char leggiChar()` Legge un singolo carattere.
- `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
`while`	`M` – `N` + 1 o `M` + 1
`res+=V[i]`	`M` – `N` o `M`
`i--`	`M` – `N` o `M`
`if(i<0)`	`M` – `N` o `M`
Totale	4`M` – 4`N` + 3 o 4`M` + 3

Domanda 2:

2 assegnamenti	2
`for`	`N` + 1
incremento di `j`	`N`
chiamata di `f`	`N`
complessità di `f`	$2N^2 + 5N$
Totale	$2N^2 + 8N + 3$

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

Soluzione Esercizio 3

```
class Esame implements Comparable<Esame> {
  private String nome, cognome, esito;
  private int matricola, voto;

  public Esame(String nome, String cognome, int matricola, int voto,
      String esito) {
    this.nome=nome;
    this.cognome=cognome;
    this.matricola=matricola;
    this.voto=voto;
    this.esito=esito;
  }

  public String getNome() { return cognome+" "+nome; }
  public int getMatricola() { return matricola; }
  public int getVoto() { return voto; }
  public String getEsito() { return esito; }

  public String toString() {
    return getNome()+"("+matricola+"): "+voto+", "+esito;
  }

  public boolean equals(Object o) { return equals((Esame) o); }
  public boolean equals(Esame e) { return (matricola==e.matricola); }

  public int compareTo(Esame e) {
    int res=cognome.compareTo(e.cognome);
    if(res==0) res=nome.compareTo(e.nome);
    if(res==0) res=matricola-e.matricola;
    return res;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Appello {
  private List<Esame> esami;
  private int anno, sessione;
  public Appello(int anno, int sessione) {
    this.anno=anno; this.sessione=sessione;
    this.esami=new ArrayList<Esame>();
  }
  public boolean aggiungi(Esame e) {
    int i=0;
    while(i<esami.size() && e.compareTo(esami.get(i))>0) i++;
    if(i<esami.size() && e.equals(esami.get(i))) return false;
    esami.add(i, e);
    return true;
  }
  public String getEsito(int n) {
    for(Esame e:esami) if(e.getMatricola()==n) return e.getEsito();
    return null;
  }
  public double media() {
    int somma=0, quanti=0;
    for(Esame e:esami) if(!e.getEsito().equals("ritirato")) {
      somma+=e.getVoto();
      quanti++; }
    return ((double)somma)/quanti;
  }
  public String toString() {
    return "Anno: "+anno+", sessione: "+sessione+"\n"+esami;
  } }
```

Soluzione Esercizio 5

```
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    int i, n, quanti=0;
    Appello[] v=new Appello[8]; // domanda 1
    v[quanti]=new Appello(Lettore.in.leggiInt(), Lettore.in.leggiInt());
    quanti++; // domanda 2
    n=Lettore.in.leggiInt();
    for(i=0; i<n; i++)
      v[quanti-1].aggiungi(new Esame(Lettore.in.leggiLinea(),
        Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
        Lettore.in.leggiInt(),Lettore.in.leggiString())); // domanda 3
    n=Lettore.in.leggiInt();
    for(i=0; i<quanti; i++) {
      String s=v[i].getEsito(n);
      if(s!=null) System.out.println(s); } // domanda 4
    double min=0;
    for(i=0; i<quanti; i++) {
      double media=v[i].media();
      if(media<min) {
        min=media;
        n=i; }}
    System.out.println(v[n]); // domanda 5
  } }
```