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

Appello del 17/4/2007

Esercizio 1 (4 punti)

Visibilità di variabili, metodi e classi in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M) {
  int i=1, sum=0;

  while(i<=M) sum+=V[i-1];
  return sum;
}

void g(int U[], int V[], int N) {
  int j;

  for(j=0; j<N; j+=2) {
    U[j]=f(V,j--);
  }
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini del parametro `M`.
2. Calcolare la complessità in passi base 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)

L'ospedale "Minore" di Collate (Parma) vuole informatizzare la gestione degli interventi al pronto soccorso. Le informazioni sui vari interventi comprendono la tipologia (es. frattura, puntura d'insetto, ferita da taglio), la priorità dell'emergenza (un valore da 0 a 3), la data, il medico che ha gestito l'emergenza e una descrizione dell'intervento effettuato.

Si scriva una classe `Intervento` per l'ospedale "Minore" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda il metodo `toString` che fornisca una descrizione testuale dell'intervento, comprendente tutte le variabili istanza.
4. Presenti un metodo `equals` per stabilire l'uguaglianza con un altro intervento (l'uguaglianza va verificata unicamente sulla data).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` che effettui il confronto con un altro intervento (la precedenza va data all'intervento più recente; a parità di data, ha precedenza l'intervento con priorità più elevata).

Esercizio 4 (8 punti)

Si scriva una classe `Cartella` che memorizzi le informazioni relative agli interventi effettuati su ogni paziente. In particolare, per ogni paziente, oltre ai dati anagrafici (nome, cognome, anno di nascita) vanno memorizzati gli interventi subiti (contenuti all'interno di una lista). La classe `Cartella` deve inoltre:

1. Possedere un opportuno costruttore con parametri (inizialmente il paziente non ha subito interventi).
2. Presentare opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Presentare un metodo `aggiungi` che, dato un `Intervento`, lo inserisca all'interno della lista, mantenendo ordinata la lista secondo il criterio espresso al punto 5. dell'esercizio 3. (suggerimento: si utilizzi il metodo `add(int i, Intervento o)` della classe `List<Intervento>`).
4. Possedere un metodo `haOperato` che, dato il nome di un medico, indichi se tale medico ha effettuato almeno un intervento sul paziente.
5. Presentare un metodo `interventi` che restituisca l'insieme degli interventi effettuati in una certa data, passata come parametro.
6. Possedere il metodo `toString` che restituisca una stringa che includa le informazioni sul paziente (comprendendo anche le informazioni su tutti gli interventi subiti dal paziente).

Esercizio 5 (7 punti)

Si scriva un'applicazione per l'ospedale "Minore" che:

1. Crei un insieme di oggetti `Cartella`.
2. Lette da tastiera le informazioni relative ad un nuovo paziente, provveda ad inserire un nuovo oggetto `Cartella` nell'insieme, verificando che tale oggetto non sia già contenuto nell'insieme stesso.
3. Lette da tastiera le informazioni relative ad un nuovo intervento, provveda a creare un nuovo oggetto `Intervento` e lo inserisca nella cartella clinica di cui al punto 2.
4. Letto da tastiera il nome di un medico, provveda a stampare le informazioni su tutti i pazienti che hanno subito almeno un intervento da parte del medico.
5. Letta da tastiera la data odierna, stampi le informazioni relative a tutti gli interventi effettuati in tale data che abbiano priorità maggiore di 2.

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.
- `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(i<=M)`	`M` + 1
`sum+=V[i-1]`	`M`
Totale	3 + 2 `M`

Domanda 2:

1 assegnamento	1
`while` eseguito `N` volte	`N` + 1
incremento di `j`	`N`
`U[j]=f(V,j--);`	`N`
complessità di `f`	$N^2 + 2N$
Totale	$N^2 + 5N + 2$

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

Soluzione Esercizio 3

```
class Intervento implements Comparable {
  private String tipo, medico, descr;
  private int g, m, a, pr;

  public Intervento(int p, String t, String e, String d, int g, int m, int a) {
    tipo=t; medico=e; descr=d;
    this.g=g; this.m=m; this.a=a; pr=p;
  }

  public String getMedico() { return medico; }
  public String getData() { return ""+g+"/"+m+"/"+a; }
  public int getPriorita() { return pr; }

  public String toString() {
    return tipo+"-"+pr+" ("+medico+"): "+getData()+", "+descr;
  }

  public boolean equals(Intervento i) {
    return(getData().equals(i.getData()));
  }

  public int compareTo(Intervento i) {
    int ret=i.a-a;
    if(ret==0) ret=i.m-m;
    if(ret==0) ret=i.g-g;
    if(ret==0) ret=i.pr-pr;
    return ret;
  }

  public boolean equals(Object o) { return equals((Intervento) o); }
  public int compareTo(Object o) { return compareTo((Intervento) o); }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Cartella {
  private String nome, cognome;
  private int anno;
  private List<Intervento> l;

  public Cartella(String nome, String cognome, int anno) {
    this.nome=nome; this.cognome=cognome; this.anno=anno;
    l=new ArrayList<Intervento>();
  }

  public String getNome() { return nome+" "+cognome; }
  public void aggiungi(Intervento o) {
    int i=0;
    while(i<l.size() && l.get(i).compareTo(o)<0) i++;
    l.add(i, o);
  }

  public boolean haOperato(String m) {
  for(Intervento i:l) if(i.getMedico().equals(m)) return true;
    return false;
  }
  public Set<Intervento> interventi(int g, int m, int a) {
    Set<Intervento> s=new HashSet<Intervento>();
    String data=""+g+"/"+m+"/"+a;
    for(Intervento i:l) if(i.getData().equals(data)) s.add(i);
    return s;
  }
  public String toString() {
    return getNome()+"("+anno+"):\n"+l.toString();
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Cartella> pazienti=new TreeSet<Cartella>();
    Cartella c=new Cartella(Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
      Lettore.in.leggiInt());
    String medico;

    if(!pazienti.add(c)) System.out.println("Paziente già esistente!");
    c.aggiungi(new Intervento(Lettore.in.leggiInt(), Lettore.in.leggiString(),
      Lettore.in.leggiString(), Lettore.in.leggiString(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(), Lettore.in.leggiInt()));
    medico=Lettore.in.leggiString();
    for(Cartella cl:pazienti) if(cl.haOperato(medico)) System.out.println(cl);
    int g=Lettore.in.leggiInt(), m=Lettore.in.leggiInt(),
      a=Lettore.in.leggiInt();
    for(Cartella cl:pazienti) {
      Set<Intervento> s=cl.interventi(g, m, a);
      for(Intervento i:s) if(i.getPriorita()>2) System.out.println(i);
    }
  }
}
```