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

Appello del 19/9/2013

Esercizio 1 (4 punti)

Discutere i concetti di classe astratta ed interfaccia 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;
  while(--i>1;)
    sum+=V[--i];
  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` dispari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 3 (5 punti)

L'Ospedale Minore di Collate (Parma) ha deciso di informatizzare la gestione dei pazienti in attesa al pronto soccorso (che, come è noto, è chiuso nelle ore notturne). A tal scopo, per ogni paziente vengono memorizzati nome e cognome, orario di inserimento nel sistema, descrizione della patologia e urgenza dell'intervento (un numero da 1 a 5, per valori crescenti di urgenza). Si scriva una classe `Paziente` per l'Ospedale Minore 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 paziente.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Paziente` (la verifica va fatta sul nome e sul cognome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Paziente` passato come parametro (in ordine decrescente di urgenza e, a parità, per orario di inserimento crescente).

Esercizio 4 (7 punti)

Si scriva una classe `Reparto` che memorizzi le informazioni relative ai pazienti di pronto soccorso in attesa a ogni reparto. Per ciascun reparto occorre memorizzare il nome (es. "Ortopedia", "Cardiologia, ecc.), il cognome del medico di turno nel giorno in questione, mentre i pazienti vanno memorizzati all'interno di una lista. La classe `Reparto` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la lista dei pazienti è vuota).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del reparto (inclusa la descrizione di tutti i pazienti in attesa).
4. Possedere il metodo `posizione` che, dato un valore di urgenza, restituisca il numero di pazienti in lista aventi un'urgenza non inferiore.
5. Presentare il metodo `aggiungi` che, dato un oggetto `Paziente`, lo inserisca all'interno della lista, mantenendo la lista ordinata secondo il punto 5. dell'esercizio 3.
6. Possedere il metodo `listadAttesa` che restituisca il numero di pazienti in attesa al reparto.
7. Presentare il metodo `cerca` che, dati il nome e il cognome di un paziente, indichi se tale paziente è in attesa nel reparto.

Esercizio 5 (8 punti)

Si scriva un'applicazione per l'Ospedale Minore di Collate (Parma):

1. Crei un insieme di oggetti `Reparto`.
2. Crei un oggetto `Reparto`, 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 `Paziente`, lette da tastiera le informazioni necessarie.
5. Inserisca il paziente creato al punto 4. tra quelli in attesa nel reparto di cui al punto 2.
6. Stampi a video il numero totale di pazienti in attesa nei reparti del pronto soccorso.
7. Stampi a video il numero totale di pazienti con urgenza 5 in attesa nei reparti del pronto soccorso.
8. Letti da tastiera il nome e il cognome di un paziente, stampi a video, se esiste, il cognome del medico di turno nel reparto in cui tale paziente è in attesa.

Soluzione Esercizio 2

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`sum+=f(V, ++j)`	(N + 1)/2
`j<N`	(N + 1)/2 + 1
`++j`	(N + 1)/2
complessità di `f`	$N^2/4 + 3N/2 + 5/4$
Totale	$N^2/4 + 3N + 23/4$

complessità di `f`: $\sum_{\substack{j=1 \\ (j \text{ dispari})}}^{N} (j+2) = \sum_{i=0}^{(N-1)/2} (2i+3) = \frac{N^2}{4} - \frac{1}{4} + \frac{3N}{2} + \frac{3}{2}$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Paziente implements Comparable<Paziente> {
  private String nome, cognome, patologia;
  private int urgenza, ora, minuto;

  public Paziente(String nome, String cognome, String patologia,
    int urgenza, int ora, int minuto) {
    this.nome = nome;
    this.cognome = cognome;
    this.patologia = patologia;
    this.urgenza = urgenza;
    this.ora = ora;
    this.minuto = minuto;
  }

  public String getOrario() { return "" + ora + ":" + minuto; }
  public String getNome() { return nome + " " + cognome; }
  public String getPatologia() { return patologia; }
  public int getUrgenza() { return urgenza; }

  public String toString() {
    return getNome() + " (" + urgenza + "-" + getOrario() + "):"
      + patologia;
  }

  public boolean equals(Object o) { return equals((Paziente) o); }
  public boolean equals(Paziente p) {
    return nome.equals(p.nome) && cognome.equals(p.cognome);
  }

  public int compareTo(Paziente p) {
    int ret = p.urgenza - this.urgenza;
    if(ret==0) ret = this.ora – p.ora;
    if(ret==0) ret = this.minuto – p.minuto;
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Reparto {
  private List<Paziente> l;
  private String nome, medico;

  public Reparto(String nome, String medico) {
    this.nome = nome;
    this.medico = medico;
    l=new LinkedList<Paziente>();
  }
  public String getNome() { return nome; }
  public String getMedico() { return medico; }

  public String toString() {
    return nome + ": " + medico + "," + l.toString();
  }

  public int posizione(int urgenza) {
    int i=0;
    while((i<l.size())&&(l.get(i).getUrgenza()>=urgenza)) i++;
    return i;
  }
  public void aggiungi(Paziente p) {
    l.add(posizione(p.getUrgenza()), p);
}
  public int listadAttesa() { return l.size(); }

  public boolean cerca(String nome, String cognome) {
    Paziente p = new Paziente(nome, cognome, "", 0, 0, 0);
    return l.contains(p);
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Reparto> s=new TreeSet<Reparto>();
    Reparto r=new Reparto(Lettore.in.LeggiLinea(),
      Lettore.in.LeggiLinea());
    if(!s.add(r)) System.out.println("Reparto già esistente!");
    Paziente p=new Paziente(Lettore.in.LeggiLinea(),
      Lettore.in.LeggiLinea(), Lettore.in.LeggiLinea(),
      Lettore.in.LeggiInt(), Lettore.in.LeggiInt(), Lettore.in.LeggiInt());
    r.aggiungi(p);
    int totale = 0;
    for(Reparto z: s) totale += z. listadAttesa();
    System.out.println(totale);
    totale = 0;
    for(Reparto z: s) totale += z.posizione(5);
    System.out.println(totale);
    String nome = Lettore.in.LeggiLinea();
    String cognome = Lettore.in.LeggiLinea();
    for(Reparto z: s)
      if(z.cerca(nome, cognome)) System.out.println(z.getMedico());
  }
}
```