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

Appello del 19/1/2018

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 M, int N) {
  int i=N, sum=0;
  while(--i>M)
    sum+=V[i];
  return sum;
}

public static int g(int V[], int N) {
  int j=1, sum=0;
  while(j<N)
    sum+=f(V, j++, N);
  return sum;
}
```

1. Calcolare la complessità in passi base del metodo `f` nei termini dei parametri `M` e `N` (si distinguano i casi in cui `M` assume valori maggiori o uguali a `N` da quelli in cui assume valori minori).
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)

In vista delle prossime primarie che dovranno decidere il candidato premier per le prossime elezioni nello stato caraibico di S. Marquez, il partito "Izquierda Marqueña" ha affidato al suo segretario, l'ingegner Bruno Paragola, l'informatizzazione dell'intero sistema di voto. In particolare, per ogni candidato occorre memorizzare il nome, il cognome e il numero di voti ottenuti. Si scriva una classe `Candidato` per Bruno Paragola che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Possieda il metodo modificatore per il numero di voti ottenuti.
4. Presenti il metodo `toString` che fornisca una descrizione del candidato.
5. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Candidato` (la verifica va fatta su nome e cognome).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Candidato` passato come parametro (per ordine alfabetico su cognome e nome e, a parità, per numero di voti decrescente).

Esercizio 4 (7 punti)

Si scriva una classe `Sezione` che registri le informazioni riguardanti le votazioni svolte in una sezione del partito. Per ogni sezione occorre memorizzare il numero della sezione e l'indirizzo (via e CAP), mentre i candidati vanno memorizzati all'interno di un insieme. La classe `Sezione` deve:

1. Presentare un opportuno costruttore con parametri (l'insieme di candidati è inizialmente vuoto).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della sezione (inclusa la descrizione di tutti i candidati).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Sezione` (la verifica va effettuata esclusivamente sul numero di sezione).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Candidato`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `numeroVoti` che restituisca il numero totale di voti ottenuti dai candidati.
7. Possedere il metodo `piuVotato` che restituisca il candidato che ha avuto il maggior numero di voti nella sezione (si supponga non vi siano pari merito).

Esercizio 5 (8 punti)

Si scriva un'applicazione per Bruno Paragola che:

1. Crei una lista di oggetti `Sezione`.
2. Crei un oggetto `Sezione`, 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 `Candidato`, lette da tastiera le informazioni necessarie (il numero di voti va inizializzato a zero).
5. Inserisca l'oggetto di cui al punto 4. all'interno di tutte le sezioni all'interno della lista di cui al punto 1., indicando per quali sezioni l'inserimento non sia possibile.
6. Stampi a video il numero della sezione in cui si è votato di più, tra quelle della lista di cui al punto 1.
7. Stampi a video la descrizione del candidato che ha ottenuto il maggior numero di voti in una singola sezione.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2	o 2
`--i>=N`	$N-M$	o 1
`sum+=V[i]`	$N-M-1$	o 0
Totale	$2N-2M+1$	o 3

Domanda 2:

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

complessità. di `f`: $\sum_{j=1}^{N-1}(2N-2j+1) = 2N(N-1) - N(N-1) + (N-1) = N^2-1$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Candidato implements Comparable<Candidato> {
  private String nome, cognome;
  private int voti;

  public Candidato(String nome, String cognome,int voti) {
    this.nome = nome;
    this.cognome = cognome;
    this.voti = voti;
  }

  public String getNome() { return nome; }
  public String getCognome() { return cognome; }
  public int getVoti() { return voti; }
  public void setVoti(int voti) { this.voti=voti; }

  public String toString() {
    return nome + " " + cognome + "): " + voti;
  }

  public boolean equals(Object o) { return equals((Candidato) o); }
  public boolean equals(Candidato c) {
    return nome.equals(c.nome) && cognome.equals(c.cognome);
  }

  public int compareTo(Candidato c) {
    int ret = this.cognome.compareTo(c.cognome);
    if(ret==0) ret = this.nome.compareTo(c.nome);
    if(ret==0) ret = c.voti-this.voti;
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Sezione {
  private String indirizzo;
  private int numero, CAP;
  private Set<Candidato> s;
  public Sezione(String indirizzo, int numero, int CAP) {
    this.indirizzo = indirizzo;
    this.numero = numero;
    this.CAP = CAP;
    s = new TreeSet<Candidato>();
  }
  public int getNumero() { return numero; }
  public String getIndirizzo() { return indirizzo; }
  public int getCAP() { return CAP; }
  public String toString() { return numero + "(" + indirizzo + "," + CAP
    + ": " + s;}
  public boolean equals(Object o) { return equals((Sezione) o); }
  public boolean equals(Sezione s){ return this.numero==s.numero;}
  public boolean aggiungi(Candidato c) { return s.add(c); }
  public int numeroVoti() {
    int count=0;
    for(Candidato c: s) count+=c.getVoti();
    return count;
  }
  public Candidato piuVotato() {
    Candidato max=null;
    for(Candidato c: s) if(max==null||c.getVoti()>max.getVoti()) max=c;
    return max;
  }
}
```

Soluzione Esercizio 5

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

class Applicazione {
  public static void main(String[] args) {
    List<Sezione> l = new LinkedList<Sezione>();
    Sezione s = new Sezione(Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
        Lettore.in.leggiInt());
    l.add(s);
    Candidato c = new Candidato(Lettore.in.leggiLinea(),
        Lettore.in.leggiLinea(), 0);
    for(Sezione x: l) if(!x.aggiungi(c)) System.out.println(x.getNumero());
    Sezione maxS=null;
    int maxVoti=0;
    for(Sezione x: l) {
      int voti=x.numeroVoti();
      if(voti>maxVoti) { maxS=x; maxVoti=voti;}
    }
    System.out.println(maxS.getNumero());
    Candidato maxC = null;
    for(Sezione x: l) {
      Candidato max=x.piuVotato();
      if(maxC==null||max.getVoti()>maxC.getVoti()) maxC=max;
    }
    System.out.println(maxC.toString());
  }
}
```