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

Appello del 16/2/2018

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

public static int g(int V[], int M, int N) {
  int j=0, sum=0;
  while(++j<M)
    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 dei parametri `M` e `N` (si supponga `M<N`).
3. Calcolare la complessità asintotica del metodo `g` nei termini dei parametri `M` e `N`.

Esercizio 3 (5 punti)

Al fine di mitigare lo scandalo dei rimborsi fasulli che rischia di stravolgere i risultati delle prossime elezioni, il partito "Izquierda Marqueña" ha affidato al suo segretario, l'ingegner Bruno Paragola, l'informatizzazione dell'intero sistema di gestione dei bonifici che gli eletti versano alle sezioni del partito. In particolare, per ogni bonifico occorre memorizzare la data, l'ammontare (in $) e il numero della sezione ricevente. Si scriva una classe `Bonifico` 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. Presenti il metodo `toString` che fornisca una descrizione del bonifico.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Bonifico` (la verifica va fatta sulla data e sulla sezione ricevente).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Bonifico` passato come parametro (per ordine decrescente di data e, a parità, per numero di sezione crescente).

Esercizio 4 (7 punti)

Si scriva una classe `Eletto` che registri le informazioni riguardanti i bonifici effettuati da un eletto del partito. Per ogni eletto occorre memorizzare il nome e il cognome, mentre i bonifici vanno memorizzati all'interno di una lista. La classe `Eletto` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la lista di bonifici 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 dell'eletto (inclusa la descrizione di tutti i suoi bonifici).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Eletto` (la verifica va effettuata su nome e cognome).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Bonifico`, lo inserisca all'interno della lista, mantenendo quest'ultima ordinata secondo il punto 5. dell'Esercizio 3.
6. Possedere il metodo `numeroBonifici` che, dato il numero di una sezione, restituisca il numero di bonifici effettuati dall'eletto a tale sezione.
7. Possedere il metodo `totale` che restituisca l'ammontare totale dei bonifici effettuati.

Esercizio 5 (8 punti)

Si scriva un'applicazione per Bruno Paragola che:

1. Crei un insieme di oggetti `Eletto`.
2. Crei un oggetto `Eletto`, 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 `Bonifico`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. tra quelli relativi alleletto di cui al punto 2.
6. Letto da tastiera il numero di una sezione, stampi a video il numero totale di bonifici effettuati a tale sezione dagli eletti dell'insieme lista di cui al punto 1.
7. Stampi a video la descrizione dell'eletto che ha versato il maggiore ammontare tra quelli dell'insieme lista di cui al punto 1.

Soluzione Esercizio 2

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`++j<M`	M
`sum+=f(V, j, N)`	$M - 1$
complessità di `f`	$2MN^2 - 2N - M^2 + 4M - 3$
Totale	$2MN^2 - 2N - M^2 + 6M - 2$

complessità. di `f`: $\sum_{j=1}^{M-1}(2N - 2j + 3) = 2N(M-1) - M(M-1) + 3(M-1) = 2MN - 2N - M^2 + 4M - 3$

Domanda 3:

Complessità asintotica: $O(MN)$

Soluzione Esercizio 3

```
class Bonifico implements Comparable<Bonifico> {
  private int g, m, a, sezione;
  private float ammontare;

  public Bonifico(int g, int m, int a, int sezione, float ammontare) {
    this.g = g;
    this.m = m;
    this.a = a;
    this.sezione = sezione;
    this.ammontare = ammontare;
  }

  public String getData() { return g+"/"+m+"/"+a; }
  public int getSezione() { return sezione; }
  public float getAmmontare() { return ammontare; }

  public String toString() {
    return getData() + " (" + sezione + "): " + ammontare;
  }

  public boolean equals(Object o) { return equals((Bonifico) o); }
  public boolean equals(Bonifico b) {
    return getData().equals(b.getData()) && sezione==b.sezione;
  }

  public int compareTo(Bonifico b) {
    int ret = b.a - this.a;
    if(ret==0) ret = b.m - this.m;
    if(ret==0) ret = b.g - this.g;
    if(ret==0) ret = this.sezione - b.sezione;
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Eletto {
  private String nome, cognome;
  private List<Bonifico> l;
  public Eletto(String nome, String cognome) {
    this.nome = nome;
    this.cognome = cognome;
    l = new ArrayList<Bonifico>();
  }
  public String getNome() { return nome; }
  public String getCognome() { return cognome; }
  public String toString() { return nome + " " + cognome+ ": " + l;}
  public boolean equals(Object o) { return equals((Eletto) o); }
  public boolean equals(Eletto e){
    return this.nome.equals(e.nome) && this.cognome.equals(e.cognome);
  }
  public void aggiungi(Bonifico b) {
    int i = 0;
    while((i<l.size())&&(l.get(i).compareTo(b)<0)) i++;
    l.add(i, b);
  }
  public int numeroBonifici(int sezione) {
    int count=0;
    for(Bonifico b: l) if(b.getSezione()==sezione) count++;
    return count;
  }
  public float totale() {
    float totale=0;
    for(Bonifico b: l) totale+=b.getAmmontare();
    return totale;
  }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Eletto> s = new HashSet<Eletto>();
    Eletto e=new Eletto(Lettore.in.leggiLinea(), Lettore.in.leggiLinea());
    if(!s.add(e)) System.out.println("Inserimento non riuscito!");
    Bonifico b=new Bonifico (Lettore.in.leggiInt(), Lettore.in.leggiInt(),
        Lettore.in.leggiInt(), Lettore.in.leggiInt(),
        Lettore.in.leggiFloat());
    e.aggiungi(b);
    int sezione = Lettore.in.leggiInt();
    int count=0;
    for(Eletto x: s) count+=x.numeroBonifici(sezione);
    System.out.println(count);
    Eletto max = null;
    float maxA = 0;
    for(Eletto x: s) {
      float ammontare=x.totale();
      if(ammontare>maxA) { max=x; maxA=ammontare; }
    }
    System.out.println(max.toString());
  }
}
```