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

Appello del 12/2/2016

Esercizio 1 (4 punti)

Modalità di gestione delle stringhe in Java.

Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[], int N) {
  int i, sum=0;
  for(i=0; i++<N; ++i)
    sum+=V[i];
  return sum;
}

public static int g(int V[], int N) {
  int j=N, sum=0;
  do
    sum+=f(V, --j);
  while(j-->0);
  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'ingegner Alberto Dritto è un appassionato di viaggi e, per risparmiare, organizza sempre le proprie vacanze prenotando da solo i voli presso le compagnie low cost. Per organizzare al meglio le prossime vacanze, l'ingegner Dritto ha deciso di memorizzare le informazioni relative a ogni volo all'interno di un calcolatore. In particolare, occorre registrare le città di partenza e arrivo, la data di partenza (non si consideri l'anno, visto che l'ing. Dritto non vola mai durante le feste natalizie) e l'orario d'imbarco. Si scriva una classe `Volo` per l'ing. Dritto 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 volo.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Volo` (la verifica va fatta unicamente sulle città di partenza e arrivo).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Volo` passato come parametro (in ordine di data di partenza e, a parità, per orario d'imbarco).

Esercizio 4 (7 punti)

Si scriva una classe `Vacanza` che memorizzi le informazioni riguardanti i voli prenotati per una vacanza. Per ogni vacanza occorre memorizzare una descrizione e il budget stanziato, mentre i voli vanno inseriti all'interno di una lista. La classe `Vacanza` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, non vi sono voli).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della vacanza (inclusa la descrizione di tutti gli voli).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Vacanza` (la verifica va effettuata unicamente sulla descrizione).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Volo`, lo inserisca in coda alla lista, solo se questo segue, secondo il punto 5. dell'esercizio 3., l'ultimo volo della lista (o se la lista è vuota).
6. Presentare il metodo `tappa` che, dato il nome di una città, indichi se esiste un volo in partenza o in arrivo in tale città.

Esercizio 4 (7 punti)

Si scriva un'applicazione per l'ing. Dritto che:

1. Crei un insieme di oggetti `Vacanza`.
2. Crei un oggetto `Vacanza`, 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 `Volo`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno della vacanza di cui al punto 2. , controllando che tale inserimento sia possibile.
6. Letto da tastiera il nome di una città, stampi a video la descrizione di tutte le vacanze in cui tale città è stata visitata.
7. Stampi a video la descrizione della vacanza che, tra tutte quelle in cui viene visitata la città di cui al punto 6., prevede il budget minore.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2	o 2
`i++<N`	$N/2 + 1$	o $(N + 1)/2 + 1$
`sum+=V[i]`	$N/2$	o $(N + 1)/2$
`++i`	$N/2$	o $(N + 1)/2$
Totale	$3N/2+3$	o $3N/2+9/2$

Domanda 2:

2 assegnamenti	2
`sum+=f(V, --j)`	$(N + 1)/2$
`j-->0`	$(N + 1)/2$
complessità di `f`	$3N^2/8 + 3N/2 + 9/8$
Totale	$3N^2/8 + 5N/2 + 33/8$

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

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Volo implements Comparable<Volo> {
  private String partenza, arrivo;
  private int g, m, ora, min;

  public Volo(String p, String a, int g, int m, int ora, int min) {
    this.partenza = p;
    this.arrivo = a;
    this.g = g; this.m = m;
    this.ora = ora; this.min = min;
  }

  public String getPartenza() { return partenza; }
  public String getArrivo() { return arrivo; }
  public String getData() { return g + "/" + m; }
  public String getOra() { return ora + ":" + min; }

  public String toString() {
    return partenza + "-" + arrivo + ": " + getData() + ", " + getOra();
  }

  public boolean equals(Object o) { return equals((Volo) o); }
  public boolean equals(Volo v) {
    return partenza.equals(v.partenza) && arrivo.equals(v.arrivo);
  }

  public int compareTo(Volo v) {
    int ret = this.m - v.m;
    if(ret==0) ret = this.g - v.g;
    if(ret==0) ret = this.ora - v.ora;
    if(ret==0) ret = this.min - v.min;
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Vacanza {
  private List<Volo> l;
  private String descrizione;
  private int budget;

  public Vacanza(String descrizione, int budget) {
    this.descrizione = descrizione;
    this.budget = budget;
    l = new LinkedList<Volo>();
  }

  public String getDescrizione() { return descrizione; }
  public int getBudget() { return budget; }

  public String toString() { return descrizione + "(" + budget + "):" + l;}

  public boolean equals(Object o) { return equals((Vacanza) o); }
  public boolean equals(Vacanza v){
    return descrizione.equals(v.descrizione);
  }

  public boolean aggiungi(Volo v) {
    if(l.isEmpty()||(l.get(l.size()-1).compareTo(v)<0)) return l.add(v);
    return false;
  }

  public boolean tappa(String nome) {
    for(Volo v: l)
    if(v.getPartenza().equals(nome)||v.getArrivo().equals(nome))
      return true;
    return false;
  }
}
```

Soluzione Esercizio 5

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

class Applicazione {
  public static void main(String[] args) {
    Set<Vacanza> s = new HashSet<Vacanza>();
    Vacanza v = new Vacanza(Lettore.in.leggiLinea(),Lettore.in.leggiInt());
    if(!s.add(v)) System.out.println("Vacanza già esistente!");
    Volo c = new Volo(Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(), Lettore.in.leggiInt(),
        Lettore.in.leggiInt());
    if(v.aggiungi(c)) System.out.println("Volo inserito!");
    String nome = Lettore.in.leggiLinea();
    Vacanza min=null;
    for(Vacanza x: s) if(x.tappa(nome)) {
      System.out.println(x.getDescrizione());
      if(min==null||x.getBudget()<min.getBudget()) min=x;
    }
    System.out.println(min.getDescrizione());
  }
}
```