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

Appello del 31/3/2009

Esercizio 1 (4 punti)

Definizione ed uso dei *generics* nelle collezioni Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[],int M) {
  int i, prod=1;
  for(i=0; i<2*M; i++)
    prod+=V[++i];
  return prod;
}

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

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` (suggerimento: si supponga `N` pari e si esprima `j`=2·`i`).
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (6 punti)

La società di consulenza "HOpinioning" vuole informatizzare la gestione dei team di consulenti associati a ciascun progetto svolto in azienda. In particolare, per ogni consulente dell'azienda viene rappresentato il nome, la qualifica (es., "I livello", "junior") e la data di assunzione. Si scriva una classe `Consulente` per la società di consulenza "HOpinioning"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 la descrizione del consulente.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Consulente` (l'uguaglianza va verificata unicamente sul nome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un `Consulente` passato come parametro (per anzianità decrescente).

Esercizio 4 (7 punti)

Si scriva una classe `Team` che memorizzi le informazioni relative ai vari team di progetto dell'azienda. In particolare, oltre all'elenco dei consulenti che partecipano al team, che vanno inserite all'interno di un insieme, occorre memorizzare il nome del progetto ed il consulente a capo del team. La classe `Team` deve inoltre:

1. Presentare un opportuno costruttore (inizialmente l'elenco dei consulenti è vuoto).
2. Possedere un metodo `getNome` che restituisca il nome del progetto.
3. Presentare un metodo `quanti` che restituisca il numero di consulenti del team.
4. Possedere un metodo `aggiungi` che, dato un oggetto `Consulente`, lo inserisca all'interno dell'insieme (a meno che tale consulente non faccia già parte del team).
5. Presentare un metodo `cerca` che, dato il nome di un consulente, indichi se esso fa parte del team o meno.
6. Possedere il metodo `toString` che restituisca una stringa che fornisca una descrizione del team, comprendendo anche tutti i consulenti che vi lavorano.

Esercizio 5 (8 punti)

Si scriva un'applicazione per la società di consulenza "HOpinioning" che:

1. Crei una lista (vuota) di oggetti `Team`.
2. Crei un oggetto `Consulente`, lette da tastiera le informazioni necessarie.
3. Letto da tastiera il nome di un progetto, crei un nuovo team, con a capo il consulente creato al punto 2, e lo inserisca in coda alla lista di cui al punto 1.
4. Letto da tastiera il nome di un consulente, provveda a stampare il nome di tutti i progetti di cui tale consulente fa parte.
5. Stampi a video il nome del progetto il cui team contiene più consulenti, tra quelli contenuti nella lista di cui al punto 1.

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:

- `boolean leggiBoolean()` Legge un boolean (delimitato da spazi).
- `char leggiChar()` Legge un singolo carattere.
- `double leggiDouble()` Legge un numero razionale (delimitato da spazi).
- `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
`for`	`M` + 1
`prod+=V[i++]`	`M`
`i++`	`M`
Totale	3 `M` + 3

Domanda 2:

2 assegnamenti	2
`while`	`N`/2 + 2
chiamata di `f`	`N`/2 + 1
complessità di `f`	$3N^2/4 + 3N + 3$
`j=N-i`	`N`/2 + 1
Totale	$3N^2/4 + 9N/2 + 7$

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

Soluzione Esercizio 3

```
class Consulente implements Comparable<Consulente> {
  private String nome, cognome, qualifica;
  private int g, m, a;

  public Consulente(String nome, String cognome, String qualifica,
      int g, int m, int a) {
    this.nome=nome; this.cognome=cognome; this.qualifica=qualifica;
    this.g=g; this.m=m; this.a=a;
  }

  public String getNome() { return nome+" "+cognome; }
  public String getQualifica() { return qualifica; }
  public String getData() { return g+"/"+m+"/"+a; }

  public String toString() {
    return getNome()+ " " + qualifica + "(" + getData() + ")";
  }

  public boolean equals(Object o) { return equals((Consulente) o); }
  public boolean equals(Consulente c) {
    return (getNome().equals(c.getNome()));
  }

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

Soluzione Esercizio 4

```
import java.util.*;
class Team {
  private String nome;
  private Consulente capo;
  private Set<Consulente> squadra;

  public Team(String nome, Consulente capo) {
    this.nome=nome;
    this.capo=capo;
    squadra = new HashSet<Consulente>();
  }

  public String getNome() { return nome; }
  public int quanti() { return squadra.size(); }
  public boolean aggiungi(Consulente c) { return squadra.add(c); }

  public boolean cerca(String nome) {
    for(Consulente c:squadra)
      if(c.getNome().equals(nome)) return true;
    return false;
  }

  public String toString() {
    return nome + " (" + capo + ")\n" + squadra;
  }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    List<Team> l = new LinkedList<Team>(); // domanda 1
    Consulente c= new Consulente(Lettore.in.leggiLinea(),
      Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt()); // domanda 2
    l.add(new Team(Lettore.in.leggiLinea(), c)); // domanda 3
    String nome=Lettore.in.leggiLinea();
    for(Team t:l)
      if(t.cerca(nome)) System.out.println(t.getNome()); // domanda 4
    int max=0;
    Team maxt=null;
    for(Team t:l) {
      int n=t.quanti();
      if(n>max) {
        maxt=t;
        max=n;
      }
    }
    System.out.println(maxt.getNome()); // domanda 5
  }
}
```