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

Appello del 27/6/2006

Compito A

Esercizio 1 (4 punti)

Gestione ed utilizzo di insiemi in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M) {
  int j=0;
  int i=M*2;
  while(i>0) {
    V[j]=i;
    i-=2;
    j++;
  }
  return V[j-1];
}

 int g(int V[], int N) {
  int i, res=0;

  for(i=0; i<N; i++)
    if(i<N/2) res+=f(V,i);
    else res-=f(V,i);
  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`.
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (10 punti)

La federazione calcistica dello stato caraibico di St. Marquez sta preparando il sistema informatico per la gestione del prossimo campionato nazionale. Per ogni squadra vengono memorizzati il nome, i punti ed il numero di reti segnate e subite.

Si scriva una classe `Squadra` per la federazione calcistica dello stato di St. Marquez che:

1. Possieda un opportuno costruttore con parametri (inizialmente la squadra non ha giocato alcuna partita).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda un metodo `partitaGiocata` che, dato il numero di reti segnate e subite in una partita, provveda ad aggiornare le variabili di istanza opportune (si ricorda che la vittoria vale 3 punti, mentre il pareggio 1 punto).
4. Presenti il metodo `toString` che fornisca una descrizione testuale della squadra.
5. Possieda un metodo `equals` per stabilire l'uguaglianza con un'altra squadra (l'uguaglianza va verificata esclusivamente sul nome).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un'altra squadra, effettuando il confronto prima sui punti (più punti meglio è), quindi sulla differenza reti fatte–subite (maggiore la differenza meglio è) ed infine sul numero di reti segnate (più reti è meglio).

Esercizio 4 (6 punti)

Si scriva una classe `Girone` che memorizzi all'interno di una lista le informazioni relative a tutte le squadre iscritte al campionato. La classe `Girone` deve:

1. Possedere un opportuno costruttore (inizialmente la lista è vuota).
2. Presentare il metodo `aggiungi` che, data una `Squadra`, la inserisca nella lista, solo se questa non è già presente.
3. Possedere un metodo `trova` che, dato il nome di una squadra, restituisca l'oggetto `Squadra` corrispondente, se tale oggetto esiste.
4. Presentare il metodo `classifica` che ordini la lista delle squadre secondo il punto 6. dell'esercizio 3. A tal scopo si utilizzi il metodo di classe `sort` della classe `Collections` che, data una collezione, la ordina in base al metodo `compareTo`.
5. Possedere il metodo `toString` che restituisca una descrizione di tutte le squadre.

Esercizio 5 (6 punti)

Si scriva un'applicazione per la federazione calcistica che:

1. Crei un oggetto di tipo `Girone`.
2. Lette da tastiera le informazioni relative a quattro nuove squadre, provveda ad inserire i relativi nuovi oggetti nel girone.
3. Letto da tastiera il risultato di una partita (es.: "Internacional Joventud 3 2"), provveda ad aggiornare in maniera opportuna le informazioni sulle squadre e riordini la classifica.
4. Stampi su video la classifica aggiornata.

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:

- `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
`while(i>0)`	$M + 1$
assegnamento	M
assegnamento	M
`j++`	M
Totale	$4M + 3$

Domanda 2:

2 assegnamenti	2
`for` eseguito N volte	$N + 1$
incremento di `i`	N
`if(i<N/2)`	N
chiamata di `f`	N
complessità di `f`	$3N + 2N(N - 1)$
Totale	$2N^2 + 5N + 3$

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

Soluzione Esercizio 3

```
class Squadra implements Comparable {
  private String nome;
  private int punti, rf, rs;

  public Squadra(String nome) {
    this.nome=nome;
    this.punti=0;
    this.rf=0;
    this.rs=0;
  }

  public String getNome() { return nome; }
  public int getPunti() { return punti; }
  public int getFatte() { return rf; }
  public int getSubite() { return rs; }

  public void partitaGiocata(int rf, int rs) {
    this.rf+=rf;
    this.rs+=rs;
    if(rf>rs) this.punti+=3;
    else if (rf==rs) this.punti+=1;
  }

  public String toString() { return nome+" "+punti+" "+rf+" "+rs; }
  public boolean equals(Squadra s) { return(nome.equals(s.nome)); }

  public int compareTo(Squadra s) {
    int ret=(s.punti-punti);

    if(ret==0) ret=((s.rf-s.rs)-(rf-rs));
    if(ret==0) ret=(s.rf-rf);
    return ret;
  }

  public boolean equals(Object o) { return equals((Squadra) o); }
  public int compareTo(Object o) { return compareTo((Squadra) o); }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Girone {
  private List<Squadra> l;

  public Girone() {
    this.l=new LinkedList<Squadra>();
  }

  public boolean aggiungi(Squadra s) {
    if(l.contains(s)) return false;
    return l.add(s);
  }

  public Squadra trova(String nome) {
    for(Squadra s:l) if(s.getNome().equals(nome)) return s;
    return null;
  }

  public void classifica() {
    Collections.sort(l);
  }

  public String toString() {
    String ret="";

    for(Squadra s:l) ret+=s.toString()+"\n";
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Girone g=new Girone(); // domanda 1

    for(int i=0; i<4; i++)
      g.aggiungi(new Squadra(Lettore.in.leggiString())); // domanda 2
    String n1=Lettore.in.leggiString();
    String n2=Lettore.in.leggiString();
    int r1=Lettore.in.leggiInt();
    int r2=Lettore.in.leggiInt();
    Squadra s1=g.trova(n1);
    Squadra s2=g.trova(n2);
    if((s1!=null)&&(s2!=null)) {
      s1.partitaGiocata(r1, r2);
      s2.partitaGiocata(r2, r1);
      g.classifica();
    } // domanda 3
    System.out.println(g); // domanda 4
  }
}
```

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

Appello del 27/6/2006

Compito B

Esercizio 1 (4 punti)

Gestione ed utilizzo di liste in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M) {
  int j=0;
  int i=M*2;
  while(i>0) {
    V[j]=i;
    i-=2;
    j++;
  }
  return V[j-1];
}

 int g(int V[], int N) {
  int i, res=0;

  for(i=0; i<N; i++)
    if(i<N/2) res+=f(V,i);
    else res-=f(V,i);
  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`.
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (8 punti)

La federazione golfistica dello stato caraibico di St. Marquez sta preparando il sistema informatico per la gestione dei prossimi tornei nazionali. Per ogni giocatore vengono memorizzati il nome, il numero di colpi ed il numero di buche giocate.

Si scriva una classe `Giocatore` per la federazione golfistica dello stato di St. Marquez che:

1. Possieda un opportuno costruttore con parametri (inizialmente il giocatore non ha giocato alcuna buca).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda un metodo `bucaGiocata` che, dato il numero di colpi effettuati, provveda ad aggiornare opportunamente le variabili di istanza.
4. Presenti il metodo `toString` che fornisca una descrizione testuale del giocatore.
5. Possieda un metodo `equals` per stabilire l'uguaglianza con un altro giocatore (l'uguaglianza va verificata esclusivamente sul nome).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un altro giocatore, effettuando il confronto prima sul numero di colpi (meno colpi è meglio), quindi sul numero di buche giocate (più buche è meglio).

Esercizio 4 (8 punti)

Si scriva una classe `Torneo` che memorizzi, oltre al nome del torneo, le informazioni relative a tutti i giocatori iscritti al torneo all'interno di un vettore. La classe `Torneo` deve:

1. Possedere un opportuno costruttore che riceva in ingresso il nome del torneo ed il numero di partecipanti massimo (inizialmente il vettore è vuoto).
2. Presentare il metodo `getNome` che permetta di accedere alla relativa variabile d'istanza.
3. Presentare il metodo `aggiungi` che, dato un `Giocatore`, lo inserisca nel vettore.
4. Possedere un metodo `trova` che, dato il nome di un giocatore, restituisca l'oggetto `Giocatore` corrispondente, se tale oggetto esiste.
5. Presentare un metodo `migliore` che restituisca l'oggetto `Giocatore` che presenti il punteggio migliore (secondo quanto indicato dal punto 6. dell'esercizio 3).
6. Possedere il metodo `toString` che restituisca la descrizione di tutti i giocatori iscritti al torneo.

Esercizio 5 (6 punti)

Si scriva un'applicazione per la federazione calcistica che:

1. Crei un insieme di oggetti di tipo `Torneo`.
2. Letto da tastiera il nome di un torneo, provveda a recuperare il torneo corrispondente all'interno dell'insieme.
3. Letto da tastiera il nome di un giocatore ed il numero di colpi con cui questi ha terminato una buca, provveda ad aggiornare in maniera opportuna le informazioni su tale giocatore nel torneo di cui al punto 2.
4. Crei un nuovo torneo, chiamato "Torneo dei Campioni", che contenga tutti i giocatori migliori dei tornei dell'insieme 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:

- `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
`while(i>0)`	`M` + 1
assegnamento	`M`
assegnamento	`M`
`j++`	`M`
Totale	4 `M` + 3

Domanda 2:

2 assegnamenti	2
`for` eseguito `N` volte	`N` + 1
incremento di `i`	`N`
`if(i<N/2)`	`N`
chiamata di `f`	`N`
complessità di `f`	3 `N` + 2 `N`(`N` – 1)
Totale	2 N^2 + 5 `N` + 3

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

Soluzione Esercizio 3

```
class Giocatore implements Comparable {
  private String nome;
  private int colpi, buche;

  public Giocatore(String nome) {
    this.nome=nome;
    this.colpi=0;
    this.buche=0;    }

  public String getNome() { return nome; }
  public int getColpi() { return colpi; }
  public int getBuche() { return buche; }

  public void bucaGiocata(int colpi) {
    this.colpi+=colpi;
    this.buche++;
  }

  public String toString() { return nome+" "+colpi+" ("+buche+")"; }
  public boolean equals(Giocatore g) { return(nome.equals(g.nome)); }

  public int compareTo(Giocatore g) {
    int ret=(colpi-g.colpi);

    if(ret==0) ret=(g.buche-buche);
    return ret;
  }

  public boolean equals(Object o) { return equals((Giocatore) o); }
  public int compareTo(Object o) { return compareTo((Giocatore) o); }
}
```

Soluzione Esercizio 4

```
class Torneo {
  private String nome;
  private Giocatore[] v;
  private int quanti;

  public Torneo(String nome, int n) {
    this.nome=nome;
    v=new Giocatore[n];
    quanti=0;
  }

  public String getNome() { return nome; }
  public void aggiungi(Giocatore g) {
    if(quanti<v.length) v[quanti++]=g;
  }
  public Giocatore trova(String nome) {
    for(int i=0; i<quanti; i++)
      if(v[i].getNome().equals(nome)) return v[i];
    return null;
  }

  public Giocatore migliore() {
    Giocatore migliore=v[0];
    for(int i=1; i<quanti; i++)
      if(v[i].compareTo(migliore)<0) migliore=v[i];
    return migliore;
  }

  public String toString() {
    String ret="";
    for(int i=0; i<quanti; i++) ret+=v[i].toString()+"\n";
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Torneo> s=new HashSet<Torneo>(); // domanda 1
    String nome= Lettore.in.leggiString();
    Torneo t=null;
    Giocatore g=null;
    for(Torneo i:s) if(i.getNome().equals(nome)) {
      t=i; break;
    } // domanda 2
    if(t!=null) g=t.trova(Lettore.in.leggiString());
    if(g!=null) g.bucaGiocata(Lettore.in.leggiInt()); // domanda 3
    t=new Torneo("Torneo dei Campioni", s.size());
    for(Torneo i:s) t.aggiungi(i.migliore()); // domanda 4
  }
}
```