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

Appello del 16/7/2009

Esercizio 1 (4 punti)

Gestione ed utilizzo di collezioni ed iteratori in Java.

Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

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

1. Calcolare la complessità in passi base del metodo `f` nei termini dei parametri `M` ed `N` (suggerimento: si distinguano i casi in cui `M` assume valori minori di `N` da quelli in cui assume valori maggiori o uguali a `N`).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `M` ed `N` (suggerimento: si supponga `N>M`).
3. Calcolare la complessità asintotica del metodo `g` nei termini dei parametri `M` ed `N`.

Esercizio 3 (6 punti)

Il governo militare dello stato caraibico di St. Marquez, visto il sovraffollamento del proprio carcere di Ciudad Phanéll a causa dell'ondata di malcontento seguita al recente golpe, ha deciso di informatizzare la gestione dei cittadini detenuti all'interno del carcere. In particolare, occorre memorizzare i dati relativi ai reati commessi: tali dati includono il tipo di reato, la data in cui tale reato è stato commesso ed il numero di anni di detenzione associati a tale reato. Si scriva una classe `Reato` per il carcere di Ciudad Phanéll 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 reato.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Reato` (l'uguaglianza va verificata solo sul tipo di reato).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un `Reato` passato come parametro (la precedenza va data al reato più recente).

Esercizio 4 (8 punti)

Si scriva una classe `Detenuto` che memorizzi le informazioni relative ai cittadini detenuti all'interno del carcere. Oltre al nome del detenuto ed alla sua data di nascita, occorre memorizzare i reati contestati al cittadino all'interno di una lista. La classe `Detenuto` deve inoltre:

1. Presentare un opportuno costruttore (inizialmente la lista dei reati è vuota).
2. Possedere un metodo `aggiungi` che, dato un oggetto `Reato`, lo inserisca all'interno della lista mantenendola ordinata secondo il punto 5. dell'esercizio 3 (suggerimento: si utilizzi il metodo `add(int i, Reato r)` della classe `List<Reato>`).
3. Presentare un metodo `cerca` che restituisca una lista contenente tutti i reati di tale detenuto commessi dopo la data che è passata come parametro al metodo.
4. Possedere un metodo `anniDiGalera` che restituisca il numero totale di anni di detenzione associati al detenuto.
5. Possedere il metodo `toString` che restituisca una stringa che fornisca una descrizione del detenuto, compresi tutti i reati a lui contestati.

Esercizio 5 (7 punti)

Si scriva un'applicazione per il carcere di Ciudad Phanéll che:

1. Crei un insieme di oggetti `Detenuto`.
2. Crei un oggetto `Detenuto`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. all'interno dell'insieme di cui al punto 1.
4. Lette da tastiera le informazioni su un nuovo `Reato`, inserisca tale reato tra quelli del detenuto di cui punto 2.
5. Trovi, all'interno dell'insieme di cui al punto 1., il detenuto associato al maggior numero di anni di detenzione.

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

Domanda 2:

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

Domanda 3:

Complessità asintotica: $O(MN)$

Soluzione Esercizio 3

```
class Reato implements Comparable<Reato> {
  private String tipo;
  private int g, m, a, anni;

  public Reato(String tipo, int g, int m, int a, int anni) {
    this.tipo=tipo;
    this.g=g; this.m=m; this.a=a;
    this.anni=anni;
  }

  public String getTipo() { return tipo; }
  public String getData() { return g+"/"+m+"/"+a; }
  public int getAnni() { return anni; }

  public String toString() {
    return tipo + "(" + getData() + ") per " + anni + " anni";
  }

  public boolean equals(Object o) { return equals((Reato) o); }
  public boolean equals(Reato r) { return (tipo.equals(r.tipo)); }

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

Soluzione Esercizio 4

```
import java.util.*;
class Detenuto {
  private String nome;
  private int g, m, a;
  private List<Reato> fedina;

  public Detenuto(String nome, int g, int m, int a) {
    this.nome=nome; this.g=g; this.m=m; this.a=a;
    fedina=new LinkedList<Reato>();
  }

  public void aggiungi(Reato r) {
    int i=0;
    while((i<fedina.size())&&(fedina.get(i).compareTo(r)<0)) i++;
    fedina.add(i, r);
  }

  public List<Reato> cerca(int g, int m, int a) {
    List<Reato> l=new ArrayList<Reato>();
    Reato r=new Reato("", g, m, a, 0);
    for(Reato reato:fedina) if(reato.compareTo(r)<0) l.add(reato);
    return l;
  }

  public int anniDiGalera() {
    int anni=0;
    for(Reato reato:fedina) anni+=reato.getAnni();
    return anni;
  }

  public String toString() {
    return nome + " (" + g + "/" + m + "/" + a + ")\n" + fedina;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*; import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Set <Detenuto> set=new TreeSet<Detenuto>(); // domanda 1
    Detenuto d=new Detenuto(Lettore.in.leggiLinea(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt()); // domanda 2
    set.add(d); // domanda 3
    d.aggiungi(new Reato(Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt())); // domanda 4
    Detenuto maxd=null; int max=0;
    for(Detenuto det: set)
      if(det.anniDiGalera()>max) {
        max=det.anniDiGalera(); maxd=det;
      }
    System.out.println(maxd); // domanda 5 }
}
```