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

Appello del 14/12/2006

Esercizio 1 (4 punti)

Discutere i concetti di classe astratta ed interfaccia in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
void f(int U[], int V[], int N) {
  for(j=0; j<N; j++) {
    int i=0;
    while(i<j)
      U[i++]+=V[j];
  }
}

void g(int U[], int V[], int M) {
  int j;

  for(j=1; j<=M; j++) f(U,V,M);
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini del parametro `N`.
2. Calcolare la complessità in passi base della funzione `g` nei termini del parametro `M`.
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `M`.

Esercizio 3 (7 punti)

La ditta "Una casa a Betlemme" si occupa della realizzazione e della vendita di presepi. In previsione del prossimo Natale, la direzione ha deciso di informatizzare il progetto e la costruzione dei presepi. A questo scopo, per ogni statuina prodotta si devono memorizzare il tipo (es. San Giuseppe, pastore), il costo e la dimensione (piccola o grande); inoltre, alcune statuine possono esistere in un unico esemplare in ogni presepe (es. San Giuseppe o l'asinello), mentre altre possono essere presenti in diversi esemplari (es. pastori, angeli).

Si scriva una classe `Statuina` per la ditta "Una casa a Betlemme" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe
3. Presenti il metodo `toString` che fornisca una descrizione testuale della statuina.
4. Possieda un metodo `equals` per stabilire l'uguaglianza con un'altra statuina; in particolare, se la statuina può esistere in più esemplari, l'uguaglianza non è mai verificata, altrimenti questa va accertata sul tipo e sulla dimensione.
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` che effettui il confronto con un'altra statuina; l'ordine è quello alfabetico sul tipo delle statuine e, a parità di tipo, va data precedenza per dimensione crescente.

Esercizio 4 (8 punti)

Ogni presepe della ditta "Una casa a Betlemme" è caratterizzato da un codice e contiene, in un insieme, le informazioni sulle statuine al suo interno; inoltre, i presepi possiedono un proprio costo di base dovuto alla scenografia presente (es. capanna, montagne). Si scriva una classe `Presepe` che:

1. Possieda un opportuno costruttore con parametri (inizialmente il presepe è vuoto).
2. Presenti un metodo che permetta di accedere al codice del presepe.
3. Possieda il metodo `aggiungi` che, data una `Statuina`, la inserisca nell'insieme, se questo non contiene già una statuina uguale, e indichi se la statuina è stata effettivamente inserita o meno.
4. Presenti un metodo `quante` che indichi il numero totale di statuine presenti nel presepe.
5. Possieda un metodo `getCosto` che restituisca il costo totale del presepe (costo di base + costo di tutte le statuine).
6. Presenti un metodo `completo` che, dato in ingresso un insieme di statuine, controlli che tutte le statuine dell'insieme in ingresso siano contenute nel presepe.
7. Possieda un metodo `equals` per stabilire l'uguaglianza con un altro presepe (l'uguaglianza va verificata unicamente sul codice).

Esercizio 5 (7 punti)

Si scriva un'applicazione per la ditta "Una casa a Betlemme" che:

1. Crei una lista di oggetti di tipo `Presepe`.
2. Lette da tastiera le informazioni relative ad un nuovo presepe (codice e costo di base), provveda a creare un nuovo oggetto `Presepe` e lo inserisca nella lista di cui al punto 1.
3. Lette da tastiera le informazioni (costo e dimensione) relative alla statuina di San Giuseppe, provveda a creare un nuovo oggetto `Statuina` e lo inserisca nel presepe di cui al punto 2., indicando se l'inserimento è avvenuto correttamente.
4. Crei un insieme di oggetti di tipo `Statuina`.
5. Supponendo che l'insieme al punto 4. sia stato opportunamente inizializzato, provveda a stampare su schermo i codici dei presepi che non contengono tutte le statuine di tale insieme.

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).
- `double leggiDouble()` Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` Legge un intero (delimitato da spazi).
- `String leggiString()` Legge una parola senza spazi al suo interno.

Soluzione Esercizio 2

Domanda 1:

1 assegnamento	1
`for` eseguito `N` volte	$N + 1$
incremento	N
assegnamento	N
ciclo `while`	N^2
Totale	$N^2 + 3N + 2$

Domanda 2:

1 assegnamento	1
`for` eseguito `M` volte	$M + 1$
chiamata di `f`	M
complessità di `f`	$M^3 + 3M^2 + 2M$
incremento	M
Totale	$M^3 + 3M^2 + 5M + 2$

Domanda 3:

Complessità asintotica: $O(M^3)$

Soluzione Esercizio 3

```
class Statuina implements Comparable {
  private String tipo;
  private double costo;
  private boolean piccola, unica;

  public Statuina(String nome, double costo, boolean piccola, boolean unica) {
    this.tipo=tipo;
    this.costo=costo;
    this.piccola=piccola;
    this.unica=unica;
  }

  public String getTipo() { return tipo; }
  public double getCosto() { return costo; }
  public boolean isPiccola() { return piccola; }
  public boolean isUnica() { return unica; }

  public String toString() {
    String s=this.getTipo()+" ("+this.getCosto()+"), dim:";
    if(isPiccola()) s+="piccola";
    else s+="grande";
    return s;
  }

  public boolean equals(Statuina s) {
    if(!isUnica() || !s.isUnica()) return false;
    return (getTipo().equals(s.getTipo()) && isPiccola()==s.isPiccola());
  }

  public int compareTo(Statuina s) {
    int ret=getTipo().compareTo(s.getTipo());
    if(ret==0)
      if(isPiccola())
        if(s.isPiccola()) return 0;
        else return -1;
      else if(s.isPiccola()) return 1;
        else return 0;
    else return ret;
  }

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

Soluzione Esercizio 4

```
import java.util.*;
class Presepe {
  private int codice;
  private double costo;
  private Set<Statuina> s;

  public Presepe(int codice, double costo) {
    this.codice=codice;
    this.costo=costo;
    this.s=new HashSet<Statuina>();
  }

  public int getCodice() { return codice; }
  public boolean aggiungi(Statuina t) { return s.add(t); }
  public int quante() { return s.size(); }

  public double getCosto() {
    double c=costo;
    for(Statuina t:s) c+=t.getCosto();
    return c;
  }

  public boolean completo(Set<Statuina> set) {
    for(Statuina t:set) if(!s.contains(t)) return false;
    return true;
  }

  public boolean equals(Presepe p) { return getCodice()==p.getCodice(); }
  public boolean equals(Object o) { return equals((Presepe) o); }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    List<Presepe> l=new ArrayList<Presepe>(); // domanda 1
    Presepe p=new Presepe(Lettore.in.leggiInt(), Lettore.in.leggiDouble());
    l.add(p); // domanda 2

    Statuina t=new Statuina("San Giuseppe", Lettore.in.leggiDouble(),
      Lettore.in.leggiBoolean(), true);
    if(p.aggiungi(t)) System.out.println("Inserimento avvenuto");
    else System.out.println("Errore nell'inserimento"); // domanda 3

    Set<Statuina> s=new TreeSet<Statuina>(); // domanda 4

    for(Presepe pr: l)
      if(!pr.completo(s)) System.out.println(pr.getCodice()); // domanda 5
  }
}
```