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

Appello del 15/2/2008

Esercizio 1 (4 punti)

Visibilità di variabili, metodi e classi in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int N, int M) {
  int res=0, i=N;
  while(i<M) {
    res+=V[i];
    i++;
  }
  return res;
}

int g(int V[], int M, int N) {
  int j, res=0;
  for(j=0; j<M; j++)
    res+=f(V, j, N);
  return res;
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini dei parametri `M` ed `N` (suggerimento: si supponga `N<M`).
2. Calcolare la complessità in passi base della funzione `g` nei termini dei parametri `M` ed `N`.
3. Calcolare la complessità asintotica della funzione `g` nei termini dei parametri `M` ed `N`.

Esercizio 3 (5 punti)

La società trasporti comunali del comune di Collate (Parma) desidera informatizzare la gestione del nuovo parcheggio coperto multilivello di Piazza Steli. Per questo, ogni posto parcheggio viene caratterizzato da un codice numerico identificativo e da un tipo (auto/moto).
Si scriva una classe `Posto` per la società trasporti comunali del comune di Collate (Parma) che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi per accedere alle proprietà del posto parcheggio.
3. Possieda il metodo `occupa` che provveda a contrassegnare il posto come occupato.
4. Presenti il metodo `adatto` che, data la descrizione di un mezzo (auto/moto), indichi se il posto è in grado di contenere tale mezzo (i posti auto possono contenere moto, ma non viceversa)
5. Presenti il metodo `toString` che fornisca una descrizione testuale del posto parcheggio.
6. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Posto` (l'uguaglianza va verificata unicamente sul codice identificativo).

Esercizio 4 (7 punti)

Si scriva una classe `Piano` che memorizzi le informazioni relative ad un piano del parcheggio multilivello. Per ogni piano occorre memorizzare, oltre ad un insieme che contenga i posti parcheggio presenti, il livello ed il prezzo del parcheggio in tale piano (ovviamente diversificato per auto/moto). La classe `Piano` deve inoltre:

1. Possedere un opportuno costruttore (all'atto della creazione, occorre specificare quanti posti auto e quanti posti moto deve contenere il piano; tali posti vanno poi creati ed inseriti nell'insieme).
2. Presenti un metodo `prezzo` che, dato un tipo di mezzo, restituisca il prezzo corrispondente.
3. Possedere il metodo `getPosto` che, data la descrizione di un mezzo (auto/moto), restituisca il primo posto disponibile adatto a contenere tale mezzo; nel caso che il mezzo sia una moto, la precedenza va data ai posti moto.
4. Possedere il metodo `prezzoTotale` che restituisca il prezzo totale dei posti attualmente occupati.
5. Presentare il metodo `toString` che restituisca una descrizione del piano.

Esercizio 5 (8 punti)

Si scriva un'applicazione per la società trasporti comunali del comune di Collate (Parma) che:

1. Crei una lista di oggetti di tipo `Piano`.
2. Lette da tastiera le informazioni relative ad un nuovo piano, provveda ad inserire tale oggetto in coda alla lista (che si deve supporre possa contenere già alcuni oggetti).
3. Crei un vettore di oggetti `Posto` per memorizzare un posto disponibile per il parcheggio per ciascun piano.
4. Letto da tastiera il tipo di un mezzo, memorizzi all'interno del vettore di cui al punto 3. le informazioni relative ai posti in cui è possibile parcheggiare tale mezzo su ciascun piano.
5. Trovi il posto a minor prezzo all'interno del vettore di cui al punto 4. (si verifichi anche l'esistenza di almeno un posto libero).
6. Stampi su video le informazioni relative al posto di cui al punto 5. e provveda ad occuparlo.

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
`while`	$M - N + 1$
`res+=V[i]`	$M - N$
`i++`	$M - N$
Totale	$3M - 3N + 3$

Domanda 2:

2 assegnamenti	2
`for`	$M + 1$
incremento di `j`	M
chiamata di `f`	M
complessità di `f`	$3MN - 3M^2/2 + 9M/2$
Totale	$3MN - 3M^2/2 + 15M/2 + 3$

Domanda 3:
Complessità asintotica: $O(MN)$

Soluzione Esercizio 3

```
class Posto {
  private String tipo;
  private int codice;
  private boolean libero;

  public Posto(String tipo, int codice) {
    this.tipo=tipo;
    this.codice=codice;
    this.libero=true;
  }
  public int getCodice() { return codice; }
  public String getTipo() { return tipo; }
  public boolean isLibero() { return libero; }
  public void occupa() { libero=false; }
  public boolean adatto(String s) {
    if(s.equals("moto")) return true;
    return s.equals(tipo);
  }
  public String toString() {
    String s=""+codice+" "+tipo;
    if(libero) s+=" (libero)";
    return s;
  }
  public boolean equals(Object o) { return equals((Posto) o); }
  public boolean equals(Posto p) { return (codice==p.codice); }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Piano {
  private Set<Posto> posti;
  private int livello;
  private float prezzoAuto, prezzoMoto;
  public Piano(int livello, float prezzoAuto, float prezzoMoto,
    int auto, int moto) {
    this.livello=livello;
    this.prezzoAuto=prezzoAuto; this.prezzoMoto=prezzoMoto;
    this.posti=new TreeSet<Posto>();
    for(int i=0; i<auto; i++) posti.add(new Posto("auto", i+1));
    for(int i=0; i<moto; i++) posti.add(new Posto("moto", i+auto+1));
  }
  public float prezzo(String tipo) {
    if(tipo.equals("moto")) return prezzoMoto;
    else return prezzoAuto;
  }

  public Posto getPosto(String tipo) {
    if(tipo.equals("moto"))
      for(Posto p:posti)
        if(p.getTipo().equals(tipo)&&p.isLibero()) return p;
    for(Posto p:posti) if(p.adatto(tipo)&&p.isLibero()) return p;
    return null;
  }

  public float prezzoTotale() {
    float totale=0;
    for(Posto p:posti)
      if(!p.isLibero())
        if(p.getTipo().equals("moto")) totale+=prezzoMoto;
        else totale+=prezzoAuto;
    return totale;
  }

  public String toString() { return "Livello: "+livello+"\n"+posti; }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    int i;
    List<Piano> l=new ArrayList<Piano>(); // domanda 1

    l.add(new Piano(Lettore.in.leggiInt(), Lettore.in.leggiFloat(),
      Lettore.in.leggiFloat(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt())); // domanda 2
    Posto[] v=new Posto[l.size()]; // domanda 3
    String tipo=Lettore.in.leggiString();
    for(i=0; i<l.size(); i++)
      v[i]=l.get(i).getPosto(tipo); // domanda 4
    i=0;
    while(i<l.size()&&v[i]==null) i++;
    if(i>= l.size()) System.out.println("non esistono posti liberi!");
    else {
      int min=i;
      while(++i<l.size())
        if(v[i]!=null&&l.get(min).prezzo(tipo)>l.get(i).prezzo(tipo))
          min=i; // domanda 5
      v[min].occupa();
      System.out.println(v[min]); // domanda 6
    }
  }
}
```