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

Appello del 21/7/2006

Esercizio 1 (4 punti)

Discutere il concetto di ereditarietà.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M) {
  int i;
  int res=0;

  for(i=M; i>0; i--)
    res+=V[--i];
  return res;
}

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

  while(++i<N)
    U[i]=f(V,i++);
}
```

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 esprima `i`=2·`j`+1 e si supponga `N` pari)
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (7 punti)

La motorizzazione civile dello stato caraibico di St. Marquez intende informatizzare la gestione delle contravvenzioni in vista della prossima entrata in vigore del sistema di patente a punti. Per ogni contravvenzione vengono memorizzati il codice, una descrizione testuale, la data ed il luogo in cui si è verificata l'infrazione ed il numero di punti da decurtare.

Si scriva una classe `Multa` per la motorizzazione civile dello stato di St. Marquez 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 contravvenzione.
4. Possieda un metodo `equals` per stabilire l'uguaglianza con un'altra multa (l'uguaglianza va verificata sulla data ed il luogo dell'infrazione).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` che effettui il confronto con un'altra multa, dando la precedenza alla contravvenzione più recente.

Esercizio 4 (7 punti)

Si scriva una classe `Patente` che memorizzi i dati di una patente. Oltre al nome ed all'indirizzo dell'intestatario, occorre memorizzare il numero della patente, l'anno di rilascio e le informazioni sulle contravvenzioni ricevute dall'intestatario. Le contravvenzioni vanno memorizzate all'interno di una lista. Ogni `Patente` deve:

1. Possedere un opportuno costruttore con parametri (inizialmente la lista delle multe è vuota).
2. Presentare il metodo `aggiungi` che, data una `Multa`, la inserisca nella lista, mantenendo ordinata la lista secondo il criterio espresso al punto 5. dell'esercizio 3. (suggerimento: si utilizzi il metodo `add(int i, Multa m)` della classe `List<Multa>`).
3. Possedere un metodo (privato) `puntiIniziali` che calcoli il numero di punti iniziali della patente; tale numero è pari agli anni trascorsi dal rilascio della patente, con un minimo di 10 punti.
4. Presentare un metodo `puntiResidui` che calcoli il numero di punti residui sulla patente.
5. Possedere un metodo `equals` per stabilire l'uguaglianza con un'altra patente (l'uguaglianza va verificata unicamente sul numero).
6. Presentare il metodo `toString` che restituisca una descrizione della patente, compresa la lista ordinata delle multe.

Esercizio 5 (8 punti)

Si scriva un'applicazione per la motorizzazione civile che:

1. Crei un insieme di oggetti di tipo `Patente`.
2. Lette da tastiera le informazioni relative ad una nuova patente, provveda ad inserire tale patente nell'insieme, controllando che questa non sia già presente.
3. Letto da tastiera il numero di una patente, provveda a ricercare all'interno dell'insieme l'oggetto `Patente` che presenti tale numero
4. Lette da tastiera le informazioni relative ad una nuova contravvenzione, provveda a creare un nuovo oggetto `Multa` e ad aggiungerlo nella lista delle contravvenzioni della patente trovata al punto 3.
5. Stampi a video le informazioni su tutte le patenti che presentino un saldo di punti non positivo.

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
`for` eseguito M/2 volte	M/2 + 1
decremento di `i`	M/2
`res+=V[--i]`	M/2
Totale	3 M/2 + 3

Domanda 2:

1 assegnamento	1
`while` eseguito N/2 volte	N/2 + 1
assegnamento	N/2
complessità di `f`	9N/4 + 3/2 N/2 (N/2 – 1)
Totale	3/8 N^2 + 5/2 N + 2

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

Soluzione Esercizio 3

```
class Multa implements Comparable {
  private int codice, punti, gg, mm, aa;
  private String descr, luogo;

  public Multa(int cod, String dd, int g, int m, int a, String l, int p) {
    codice=cod; descr=dd;
    gg=g; mm=m; aa=a;
    luogo=l; punti=p;
  }

  public int getCodice() { return codice; }
  public String getDescrizione() { return descr; }
  public String getData() { return ""+gg+"/"+mm+"/"+aa; }
  public String getLuogo() { return luogo; }
  public int getPunti() { return punti; }

  public String toString() { return descr+" ("+codice+"), "+punti+"\n"+luogo
    +", il "+getData(); }

  public boolean equals(Multa m) {
    return(getData().equals(m.getData()) && getLuogo().equals(m.getLuogo())); }

  public int compareTo(Multa m) {
    int ret=(m.aa-this.aa);
    if(ret==0) ret=(m.mm-this.mm);
    if(ret==0) ret=(m.gg-this.gg);
    return ret;
  }

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

Soluzione Esercizio 4

```
import java.util.*;
class Patente {
  private String nome, ind;
  private int numero, anno;
  private List<Multa> l;

  public Patente(String nome, String ind, int numero, int anno) {
    this.nome=nome;
    this.ind=ind;
    this.numero=numero;
    this.anno=anno;
    this.l=new ArrayList<Multa>();
  }

  public void aggiungi(Multa m) {
    int i=0;
    while(i<l.size() && l.get(i).compareTo(m)<0) i++;
    l.add(i, m);
  }

  private int puntiIniziali() {
    int punti=2006-anno;
    if(punti<10) punti=10;
    return punti;
  }

  public int puntiResidui() {
    int punti=puntiIniziali();
    for(Multa m:l) punti-=m.getPunti();
    return punti;
  }

  public boolean equals(Patente p) { return this.numero==p.numero; }
  public boolean equals(Object o) { return equals((Patente) p); }

  public String toString() {
    String ret=""+numero+", "+nome+", "+ind+" ("+anno+")\n";
    for(Multa m:l) ret+=m.toString()+"\n";
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Patente> s=new HashSet<Patente>(); // domanda 1

    if(!s.add(new Patente(Lettore.in.leggiString(), Lettore.in.leggiString(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt())))
        System.out.println("Patente già presente"); // domanda 2
    int numero=Lettore.in.leggiInt();
    Patente p=new Patente("", "", numero, 0), pat=null;
    for(Patente o: s) if(o.equals(p)) {
      pat=o;
      break;
    } // domanda 3
    Multa m=new Multa(Lettore.in.leggiInt(), Lettore.in.leggiString(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(), Lettore.in.leggiInt(),
      Lettore.in.leggiString(), Lettore.in.leggiInt());
    if(pat!=null) pat.aggiungi(m); // domanda 4
    for(Patente o: s)
      if(o.puntiResidui()<=0) System.out.println(o); // domanda 5
  }
}
```