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

Appello del 16/1/2007

Esercizio 1 (4 punti)

Modalità di creazione di oggetti in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

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

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

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

  for(j=1; j<=N; j+=2) {
    U[j]=f(V,j-1);
  }
}
```

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

Esercizio 3 (6 punti)

Lo studio dentistico "Denti splendenti" vuole gestire tramite un sistema informatico le cartelle cliniche dei propri pazienti. A questo scopo, le informazioni sulle prestazioni offerte vengono memorizzate all'interno delle cartelle cliniche. Per ogni prestazione si devono memorizzare il codice, la descrizione, la data in cui è stata effettuata e il prezzo. Inoltre, le prestazioni possono essere pagate dai pazienti in tempi diversi, quindi occorre tenere traccia dell'avvenuto pagamento di ogni prestazione.
Si scriva una classe `Prestazione` per lo studio dentistico "Denti splendenti" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda il metodo `saldata` che contrassegni come pagata la prestazione.
4. Presenti il metodo `toString` che fornisca la descrizione completa della prestazione.
5. Possieda un metodo `equals` per stabilire l'uguaglianza con un'altra prestazione; l'uguaglianza va verificata unicamente sul codice.
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` che effettui il confronto con un'altra prestazione, dando la precedenza alla prestazione più datata.

Esercizio 4 (8 punti)

Ogni cartella clinica dello studio "Denti splendenti" è caratterizzata dal codice del paziente, dai suoi dati anagrafici (nome e cognome, indirizzo) e contiene, in una lista, le informazioni sulle prestazione effettuate al paziente. Si scriva una classe `Cartella` che:

1. Possieda un opportuno costruttore con parametri (inizialmente la cartella non contiene alcuna prestazione).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda il metodo `aggiungi` che, data una `Prestazione`, la inserisca in coda alla lista.
4. Presenti il metodo `costototale` che restituisca il costo totale delle prestazioni non ancora pagate.
5. Presenti un metodo `fattura` che fornisca una stringa contenente la fattura relativa a tutte le prestazioni non saldate (includendo anche l'importo totale da pagare).
6. Possieda un metodo `prestazioni` che restituisca un insieme contenente tutte le prestazioni effettuate sul paziente (una sola prestazione per ogni tipo, senza duplicati).
7. Presenti un metodo `saldata` che provveda a contrassegnare come pagate tutte le prestazioni non ancora saldate.

Esercizio 5 (7 punti)

Si scriva un'applicazione per lo studio dentistico "Denti splendenti" che:

1. Crei un insieme di oggetti di tipo `Cartella`.
2. Lette da tastiera le informazioni relative ad un nuovo paziente (codice e dati anagrafici), provveda a creare un nuovo oggetto `Cartella` e lo inserisca nell'insieme di cui al punto 1.
3. Lette da tastiera le informazioni relative ad una prestazione, provveda a creare un nuovo oggetto `Prestazione` e lo inserisca nella cartella di cui al punto 2.
4. Stampi a video il nome e cognome di tutti i pazienti che non hanno effettuato tutte le prestazioni del paziente di cui al punto 2.
5. Stampi a video le fatture relative a tutti i pazienti dell'insieme di cui al punto 1., solo per quelle fatture il cui importo ammonta a più di 50 euro.

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).
- `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` volte	`M` + 1
`sum+=V[--i];`	`M`
Totale	2 `M` + 3

Domanda 2:

1 assegnamento	1
`for` eseguito `N`/2 volte	`N`/2 + 1
complessità di `f`	$N^2/2 + N/2$
`U=f(V,j-1);`	`N`/2
Incremento di `j`	`N`/2
Totale	$2 + 2N + N^2/2$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Prestazione implements Comparable {
  private String codice, descrizione;
  private int costo, gg, mm, aa;
  private boolean pagata;

  public Prestazione(String codice, String descrizione, int costo, int gg,
    int mm, int aa) {
    this.codice=codice; this.descrizione=descrizione;
    this.costo=costo; this.gg=gg; this.mm=mm; this.aa=aa;
    pagata=false;
  }

  public String getCodice() { return codice; }
  public String getDescr() { return descrizione; }
  public int getCosto() { return costo; }
  public boolean isPagata() { return pagata; }
  public void saldata() { pagata=true; }

  public String toString() {
    String s;
    s=""+gg+"/"+mm+"/"+aa+"\t"+getCodice()+"\t"+getDescr()+"\t"+getCosto();
    return s;
  }

  public boolean equals(Prestazione p) {
    return (this.getCodice().equals(p.getCodice()));
  }

  public int compareTo(Prestazione p) {
    int ret=aa-p.aa;
    if(ret==0) ret=mm-p.mm;
    if(ret==0) ret=gg-p.gg;
    return ret;
  }

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

Soluzione Esercizio 4

```
import java.util.*;
class Cartella {
  private int codice;
  private String nome, cognome, indirizzo;
  private List<Prestazione> l;

  public Cartella(int codice, String nome, String cognome, String indirizzo) {
    this.codice=codice; this.indirizzo=indirizzo;
    this.nome=nome; this.cognome=cognome;
    this.l=new LinkedList<Prestazione>();
  }

  public int getCodice() { return codice; }
  public String getNome() { return nome+" "+cognome; }
  public String getIndirizzo() { return indirizzo; }
  public void saldata() { for(Prestazione p:l) p.saldata(); }
  public void aggiungi(Prestazione p) { l.add(p); }

  public int costototale() {
    int c=0;
    for(Prestazione p:l)
      if(!p.isPagata()) c+=p.getCosto();
    return c;
  }
  public String fattura() {
    String s="";
    for(Prestazione p:l)
      if(!p.isPagata()) s+=p.toString()+"\n";
    return s+costototale();
  }

  public Set<Prestazione> prestazioni() {
    Set<Prestazione> s=new HashSet<Prestazione>();
    for(Prestazione p:l) s.add(p);
    return s;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Cartella> s=new TreeSet<Cartella>(); // domanda 1
    Cartella c=new Cartella(Lettore.in.leggiInt(), Lettore.in.leggiLinea(),
      Lettore.in.leggiLinea(), Lettore.in.leggiLinea());
    s.add(c); // domanda 2

    Prestazione p=new Prestazione(Lettore.in.leggiLinea(),
      Lettore.in.leggiLinea(), Lettore.in.leggiInt(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt(),Lettore.in.leggiInt());
    c.aggiungi(p); // domanda 3
    Set<Prestazione> ps=c.prestazioni();
    for(Cartella cp:s) {
      Set<Prestazione> cps=cp.prestazioni();
      for(Prestazione pr:ps)
        if(!cps.contains(pr)) System.out.println(cp.getNome());
    } // domanda 4
    for(Cartella cp:s)
      if(cp.costototale()>50) System.out.println(cp.fattura()); // domanda 5
  }
}
```