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

Appello del 19/10/2012

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, i=M;
  do
    sum+=V[i];
  while(--i>N)
  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 del parametro `N` (si distinguano i casi in cui `N` assume valori maggiori di `M` da quelli in cui assume valori minori o uguali a `M`).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `N` e `M` (si supponga `N`>`M`).
3. Calcolare la complessità asintotica del metodo `g` nei termini dei parametri `N` e `M`.

Esercizio 3 (6 punti)

L'agenzia di riscossione crediti "VAM Piro" intende gestire all'interno di un sistema informatico le fatture emesse. Ogni fattura è caratterizzata da una descrizione del servizio svolto, da un importo in euro e dal mese e anno di emissione. Per ogni fattura, inoltre, occorre indicare se essa è stata pagata o meno. Si scriva una classe `Fattura` per il sistema informatico della "VAM Piro" che:

1. Possieda un opportuno costruttore con parametri (inizialmente, una fattura non è pagata).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Possieda il metodo `pagata` che modifica la fattura indicandone l'avvenuto pagamento.
4. Presenti il metodo `toString` che fornisca una descrizione della fattura.
5. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Fattura` (l'uguaglianza va verificata sulla data di emissione e sull'importo della fattura).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Fattura` passato come parametro (la precedenza va data alla fattura meno recente, in caso di parità si utilizzi l'ordine alfabetico sulla descrizione).

Esercizio 4 (7 punti)

Si scriva una classe `Cliente` che memorizzi le informazioni relative ad un cliente della "VAM Piro". Per ogni cliente si memorizzi il nome, l'indirizzo di residenza ed una lista di fatture. La classe `Cliente` deve inoltre:

1. Presentare un opportuno costruttore con parametri (inizialmente, un cliente non ha fatture).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del cliente (inclusa la descrizione di tutte le fatture ad esso associate).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Fattura`, lo inserisca all'interno della lista, mantenendo la lista ordinata secondo il punto 6. dell'esercizio 3.
5. Presentare il metodo `totaleInevase` che restituisca l'importo totale delle fatture del cliente non pagate.
6. Possedere il metodo `ultima` che restituisca la fattura più recente del cliente.

Esercizio 5 (7 punti)

Si scriva un'applicazione per l'agenzia "VAM Piro" che:

1. Crei un insieme di oggetti `Cliente`.
2. Crei un oggetto `Cliente`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. all'interno dell'insieme di cui al punto 1., controllando che tale inserimento sia possibile.
4. Crei un oggetto `Fattura`, lette da tastiera le informazioni necessarie.
5. Inserisca la fattura creata al punto 4. tra quelle associate al cliente di cui al punto 2.
6. Stampi a video la descrizione del cliente con il maggior importo di fatture non pagate tra quelli dell'insieme di cui al punto 1.
7. Stampi a video la fattura più recente tra quelle dei clienti nell'insieme di cui al punto 1.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2	o 2
`sum+=V[i]`	`M – N`	o 1
`--i>N`	`M – N`	o 1
Totale	`2M – 2N + 2`	o 4

Domanda 2:

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

complessità di `f`: $\sum_{j=1}^{M}(2N - 2j + 2) = 2NM - M^2 - M + 2M$

Domanda 3:

Complessità asintotica: $O(NM)$

Soluzione Esercizio 3

```
class Fattura implements Comparable<Fattura> {
  private String descrizione;
  private int mese, anno;
  private float importo;
  private boolean pagata;

  public Fattura(String descrizione, int mese, int anno, float importo) {
    this.descrizione = descrizione;
    this.mese = mese;
    this.anno = anno;
    this.importo = importo;
    this.pagata = false;
  }
  public String getDescrizione() { return descrizione; }
  public float getImporto() { return importo; }
  public String getData() { return mese + "/" + anno; }
  public boolean isPagata() { return pagata; }
  public void pagata() { pagata = true; }
  public String toString() {
    return descrizione + "\t" + getData() + ": " + importo;
  }
  public boolean equals(Object o) { return equals((Fattura) o); }
  public boolean equals(Fattura f) {
    return getData().equals(f.getData()) && importo==f.importo;
  }
  public int compareTo(Fattura f) {
    int ret=anno-f.anno;
    if(ret==0) ret=mese-f.mese;
    if(ret==0) ret=descrizione.compareTo(f.descrizione);
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Cliente {
  private String nome, indirizzo;
  private List<Fattura> l;
  public Cliente(String nome, String indirizzo) {
    this.nome = nome;
    this.indirizzo = indirizzo;
    l=new LinkedList<Fattura>();
  }
  public String getNome() { return nome; }
  public String getIndirizzo() { return indirizzo; }
  public String toString() {
    return nome+ ", " + indirizzo + ": " + l.toString();
  }
  public void aggiungi(Fattura f) {
    int i=0;
    while((i<l.size())&&(l.get(i).compareTo(f)<0)) i++;
    l.add(i, f);
  }
  public float totaleInevase() {
    float totale=0;
    for (Fattura f: l) if(!f.isPagata()) totale+=f.getImporto();
    return totale;
  }
  public Fattura ultima() {
    if (l.size() > 0) return l.get(l.size()-1);
    return null;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Cliente> s=new TreeSet<Cliente>();
    Cliente c=new Cliente(Lettore.in.leggiLinea(),Lettore.in.leggiLinea());
    if(!s.add(c)) System.out.println("Cliente già presente!");
    Fattura f=new Fattura(Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt(), Lettore.in.leggiFloat());
    c.aggiungi(f);
    Cliente max=null;
    float maxD=-1;
    for(Cliente x: s) {
      float totale=x.totaleInevase();
      if(totale>maxD) { max=x; maxD=totale; }
    }
    System.out.println(max.toString());
    f=null;
    for(Cliente x: s) {
      Fattura ultima=x.ultima();
      if(ultima!=null && (f==null || ultima.compareTo(f)>0)) f=ultima;
    }
    System.out.println(f);
  }
}
```