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

Appello del 16/2/2015

Esercizio 1 (4 punti)

Gestione e utilizzo di liste in Java.

Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

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

public static int g(int V[],int N) {
  int j=N, sum=0;
  for (; j>0; )
    sum+=f(V, N, --j);
  return sum;
}
```

1. Calcolare la complessità in passi base del metodo `f` nei termini dei parametri `M` e `N` (si distinguano i casi in cui `M` assume valori maggiori o uguali a `N` da quelli in cui assume valori minori di `N`).
2. Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N` pari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 3 (5 punti)

Il signor A. Cane, proprietario della pizzeria "Pavarotti" di via del Giardino, ha deciso di informatizzare la gestione delle pizze prodotte e vendute ai propri clienti. A tal scopo, degli ingredienti utilizzati vengono memorizzati il nome, il costo unitario e il numero di calorie (la pizzeria "Pavarotti" è sempre molto attenta alla linea dei propri clienti). Si scriva una classe `Ingrediente` per la pizzeria "Pavarotti" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca una descrizione dell'ingrediente.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Ingrediente` (la verifica va fatta sul nome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Ingrediente` passato come parametro (in ordine decrescente di apporto calorico e, a parità, in ordine alfabetico per nome).

Esercizio 4 (7 punti)

Si scriva una classe `Pizza` che memorizzi le informazioni relative alle pizze prodotte. Per ciascun tipo di pizza occorre memorizzare il nome e la farina utilizzata per produrre l'impasto (es. "integrale", "kamut", "doppia lievitazione"), mentre gli ingredienti vanno memorizzati all'interno di un insieme. La classe `Pizza` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la pizza non contiene alcun ingrediente).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della pizza (inclusa la descrizione di tutti i suoi ingredienti).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Pizza` (la verifica va effettuata sul nome).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Ingrediente`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `calorie` che restituisca il numero totale di calorie degli ingredienti della pizza (l'apporto calorico dell'impasto è da considerarsi trascurabile).
7. Presentare il metodo `ingrediente` che, data una stringa, indichi se la pizza contiene o meno un ingrediente con tale nome.

Esercizio 5 (8 punti)

Si scriva un'applicazione per la pizzeria "Pavarotti" che:

1. Crei una lista di oggetti `Pizza`.
2. Crei un oggetto `Pizza`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. in coda alla lista di cui al punto 1.
4. Crei un oggetto `Ingrediente`, lette da tastiera le informazioni necessarie.
5. Inserisca l'ingrediente creato al punto 4. tra quelli della pizza di cui al punto 2, controllando che tale inserimento sia possibile.
6. Crei un insieme di oggetti pizza e vi inserisca tutte le pizze, tra quelle della lista di cui al punto 1., che non contengono l'ingrediente "mozzarella".
7. Tra tutte le pizze dell'insieme di cui al punto 6., stampi a video le informazioni della pizza con meno calorie.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2	o 2
`i++<N`	1	o $N - M$
`sum+=V[i]`	0	o $N - M - 1$
Totale	3	o $2N - 2M + 1$

Domanda 2:

2 assegnamenti	2
`j>0`	$N + 1$
`sum+=f(V, N, --j)`	N
complessità di `f`	$3N$
Totale	$5N + 3$

complessità di `f`: $\sum_{j=0}^{N-1} 3 = 3N$

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

Soluzione Esercizio 3

```
class Ingrediente implements Comparable<Ingrediente> {
  private String nome;
  private float costo, calorie;

  public Ingrediente(String nome, float costo, float calorie) {
    this.nome = nome;
    this.costo = costo;
    this.calorie = calorie;
  }

  public String getNome() { return nome; }
  public float getCosto() { return costo; }
  public float getCalorie() { return calorie; }

  public String toString() {
    return nome + " (" + calorie + "): " + costo;
  }

  public boolean equals(Object o) { return equals((Ingrediente) o); }
  public boolean equals(Ingrediente i) { return this.nome.equals(i.nome); }

  public int compareTo(Ingrediente i) {
    float ret = i.calorie - this.calorie;
    if(ret<0) return -1;
    if(ret>0) return 1;
    return this.nome.compareTo(i.nome);
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Pizza {
  private Set<Ingrediente> s;
  private String nome, impasto;

  public Pizza(String nome, String impasto) {
    this.nome = nome;
    this.impasto = impasto;
    s = new HashSet<Ingrediente>();
  }

  public String getNome() { return nome; }
  public String getImpasto() { return impasto; }

  public String toString() {
    return nome + "( " + impasto + "):" + s.toString();
  }

  public boolean equals(Object o) { return equals((Pizza) o); }
  public boolean equals(Pizza p) { return this.nome.equals(p.nome); }

  public boolean aggiungi(Ingrediente i) { return s.add(i); }

  public float calorie() {
    int c=0;
    for(Ingrediente i: s) c+=i.getCalorie();
    return c;
  }

  public boolean ingrediente(String nome) {
    for(Ingrediente i: s) if(i.getNome().equals(nome)) return true;
    return false;
  }
}
```

Soluzione Esercizio 5

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

class Applicazione {
  public static void main(String[] args) {
    List<Pizza> l = new LinkedList<Pizza>();
    Pizza p = new Pizza(Lettore.in.leggiLinea(), Lettore.in.leggiLinea());
    l.add(p);
    Ingrediente i = new Ingrediente(Lettore.in.leggiLinea(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt());
    if(!p.aggiungi(i)) System.out.println("Ingrediente già presente!");
    Set<Pizza> s = new TreeSet<Pizza>();
    for(Pizza x: l) if(!x.ingrediente("mozzarella")) s.add(x);
    Pizza min = null;
    float minc = 0;
    for(Pizza x: s) {
      float cal = x.calorie();
      if(min==null || cal < minc) {
        min = x;
        minc = cal;
      }
    }
    System.out.println(min);
  }
}
```