Esame di Fondamenti di Informatica T-1/T-A
Ingegneria Gestionale (A-K)

Appello del 20/1/2017

NOTA: Per il superamento dell'esame è **necessario** ottenere la sufficienza nello svolgimento dell'Esercizio 1.

Esercizio 1 (4 punti)

1. Descrivere le principali differenze tra memoria principale e secondaria.
2. Descrivere l'utilizzo degli iteratori per la visita di collezioni in Java.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **–0,0635** supponendo di utilizzare 8 bit per la mantissa (in modulo e segno) ed 8 bit per l'esponente (in complemento a 2). Si riconverta infine il numero rappresentato in base 10 (esprimendolo come somma di potenze di due), motivando eventuali differenze con il numero originale.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

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

public static int g(int V[], int M, int N) {
  int sum=0;
  for(int j=N; j>0; --j)
    sum+=f(V, j, M);
  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 minori di `N` da quelli in cui assume valori maggiori o uguali a `N`).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `M` e `N` (si supponga `N` minore di `M`).
3. Calcolare la complessità asintotica del metodo `g` nei termini dei parametri `M` e `N`.

Esercizio 4 (5 punti)

Tempi duri per il rettore dell'Università "La Sorbolona": la protesta degli studenti, con a capo il combattivo Joseph Palmier, per il rincaro dei prezzi della mensa ha portato alla necessità di una riorganizzazione del sistema di gestione della mensa stessa. Per questo, il rettore ha deciso di gestire all'interno di un calcolatore le informazioni relative ai piatti serviti nella mensa universitaria. In particolare, per ogni piatto occorre registrare il nome, il numero di calorie e il prezzo. Si scriva una classe `Piatto` per il rettore de "La Sorbolona" 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 del piatto.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Piatto` (la verifica va fatta esclusivamente sul nome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Piatto` passato come parametro (in ordine decrescente di prezzo e, a parità, per ordine alfabetico del nome).

Esercizio 5 (8 punti)

Si scriva una classe `Menu` che memorizzi le informazioni riguardanti un menu giornaliero proposto dalla mensa universitaria de "La Sorbolona". Per ogni menu occorre la data, mentre i piatti proposti vanno inseriti all'interno di una lista. La classe `Menu` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, il menu è vuoto).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del menu (inclusa la descrizione di tutti i piatti inclusi).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Menu` (la verifica va effettuata esclusivamente sulla data).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Piatto`, lo inserisca all'interno della lista, mantenendo quest'ultima ordinata secondo il punto 5. dell'Esercizio 4.
6. Possedere il metodo `piatto` che, dato il nome di un piatto, indichi se tale piatto è contenuto o meno nel menu.
7. Possedere il metodo `costo` che restituisca la somma totale dei prezzi dei piatti inclusi nel menu.

Esercizio 6 (7 punti)

Si scriva un'applicazione per il rettore de "La Sorbolona" che:

1. Crei un insieme di oggetti `Menu`.
2. Crei un oggetto `Menu`, 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 `Piatto`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno del menu di cui al punto 2.
6. Letto da tastiera il nome di un piatto, stampi a video la data dei menu che contengono tale piatto all'interno dell'insieme di cui al punto 1.
7. Stampi a video la descrizione del menu più costoso tra quelli dell'insieme di cui al punto 1.

Soluzione Esercizio 2

$-0{,}0635_{10} = -0{,}00010000010\ldots_2$ quindi la mantissa è **(1)1000001**, l'esponente -3_{10} = **11111101**.
Il numero rappresentato è diverso dall'originale poiché quest'ultimo possiede un numero di cifre significative maggiore di 7.

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`j>0`	$N+1$
`sum+=f(V, j, M)`	N
`--j`	N
complessità di `f`	$2MN-N^2+N$
Totale	$2MN-N^2+4N+3$

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

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

Soluzione Esercizio 4

```
class Piatto implements Comparable<Piatto> {
  private String nome;
  private float calorie, prezzo;

  public Piatto(String nome, float calorie, float prezzo) {
    this.nome = nome;
    this.calorie = calorie;
    this.prezzo = prezzo;
  }

  public String getNome() { return nome; }
  public float getCalorie() { return calorie; }
  public float getPrezzo() { return prezzo; }

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

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

  public int compareTo(Piatto p) {
    if(prezzo>p.prezzo) return -1;
    if(prezzo<p.prezzo) return 1;
    return nome.compareTo(p.nome);
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Menu {
  private int g, m, a;
  private List<Piatto> l;

  public Menu(int g, int m, int a) {
    this.g = g; this.m = m; this.a = a;
    l = new LinkedList<Piatto>();
  }

  public String getData() { return g+"/"+m+"/"+a; }
  public String toString() { return getData() + ": " + l; }

  public boolean equals(Object o) { return equals((Menu) o); }
  public boolean equals(Menu m){ return getData().equals(m.getData()); }

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

  public boolean piatto(String s) {
    for(Piatto p: l) if(p.getNome().equals(s)) return true;
    return false;
  }

  public float costo() {
    float costo=0;
    for(Piatto p: l) costo+=p.getPrezzo();
    return costo;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Menu> s = new TreeSet<Menu>();
    Scanner scanner = new Scanner(System.in);
    Menu m = new Menu(scanner.nextInt(), scanner.nextInt(),
        scanner.nextInt());
    if(!s.add(m)) System.out.println("Menu già esistente");
    Piatto p = new Piatto(scanner.nextLine(), scanner.nextFloat(),
        scanner.nextFloat());
    m.aggiungi(p);
    String nome = scanner.nextLine();
    for(Menu x: s)
      if(x.piatto(nome)) System.out.println(x.getData());
    float maxC = 0;
    Menu max = null;
    for(Menu x: s) {
      float costo=x.costo();
      if(costo>maxC) { maxC=costo; max=x; }
    }
    System.out.println(max);
  }
}
```