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

Appello del 16/6/2017

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

Esercizio 1 (4 punti)

1. Illustrare il ciclo di esecuzione di un'istruzione da parte della CPU.
2. Discutere il concetto di ereditarietà.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **0,0315** 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 come somma di potenze di 2, motivando eventuali differenze con il numero originale.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

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

public static int g(int V[], int N) {
  int j=1, sum=0;
  while(j<N)
    sum+=f(V, 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 pari da quelli in cui assume valori dispari).
2. Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N` dispari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (5 punti)

Il comune di Collate (Parma) ha deciso di affidare al dott. Antonio LaSporta la manutenzione degli alberi presenti nelle strade urbane. Per questo, il dott. LaSporta ha deciso di gestire all'interno di un calcolatore le informazioni relative agli alberi presenti sul territorio viabile comunale. In particolare, per ogni albero occorre registrare la matricola, il genere (es. Pino, Abete) e l'anno in cui è stato piantato. Si scriva una classe `Albero` per il dott. LaSporta 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'albero.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Albero` (la verifica va fatta esclusivamente su genere e matricola).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Albero` passato come parametro (per ordine alfabetico del genere e, a parità, per età decrescente).

Esercizio 5 (7 punti)

Si scriva una classe `Strada` che registri le informazioni riguardanti una singola strada del comune di Collate (Parma). Per ogni strada occorre memorizzare il nome e il quartiere, mentre gli alberi vanno inseriti all'interno di un insieme. La classe `Strada` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, l'elenco degli alberi 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 della strada (inclusa la descrizione di tutti gli alberi).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Strada` (la verifica va effettuata esclusivamente sul nome).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Albero`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `genere` che, dato il nome di un genere, restituisca un insieme contenente tutti gli alberi di tale genere inclusi nella strada.
7. Possedere il metodo `vecchi` che, dato un anno, restituisca il numero totale di alberi contenuti nella strada piantati prima di tale anno.

Esercizio 6 (9 punti)

Si scriva un'applicazione per il dott. LaSporta che:

1. Crei una lista di oggetti `Strada`.
2. Crei un oggetto `Strada`, 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 `Albero`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno della strada di cui al punto 2., controllando che tale inserimento sia possibile.
6. Stampi a video il numero totale di alberi delle strade del comune di Collate (Parma) piantati prima dell'anno 2000.
7. Letto da tastiera il nome di un genere, stampi a video il nome di tutte le strade, all'interno della lista di cui al punto 1., che contengono alberi di tale genere.
8. Stampi a video la descrizione della matricola dell'albero più vecchio, tra quelli del genere di cui al punto 7.

Soluzione Esercizio 2

$-0{,}0315_{10} = -0{,}00001000000\ldots_2$ quindi la mantissa è **(0)1000000**, l'esponente -4_{10} = **11111100**.
Il numero rappresentato è $2^{-5} = 0.03124$, 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/2	o (N + 1)/2
`++i<N`	N/2	o (N + 1)/2
Totale	N + 2	o N + 3

Domanda 2:

2 assegnamenti	2
`j<N`	N
`sum+=f(V, j++)`	N – 1
complessità di `f`	$N^2/2 + 2N - 5/2$
Totale	$N^2/2 + 4N - 3/2$

complessità. di `f`: $\sum_{\substack{j=1 \\ j \text{ dispari}}}^{N-1} (j+3) + \sum_{\substack{j=1 \\ j \text{ pari}}}^{N-1} (j+2) = \sum_{j=1}^{N-1}(j+2) + \sum_{\substack{j=1 \\ j \text{ dispari}}}^{N-1} 1 =$

$$\frac{N(N-1)}{2} + 2(N-1) + \frac{N-1}{2} = \frac{N^2}{2} + 2N - \frac{5}{2}$$

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

Soluzione Esercizio 4

```
class Albero implements Comparable<Albero> {
  private String genere;
  private int matricola, anno;

  public Albero(String genere, int matricola, int anno) {
    this.genere = genere;
    this.matricola = matricola;
    this.anno = anno;
  }

  public String getGenere() { return genere; }
  public int getMatricola() { return matricola; }
  public int getAnno() { return anno; }

  public String toString() {
    return matricola + " (" + anno + "): " + genere;
  }

  public boolean equals(Object o) { return equals((Albero) o); }
  public boolean equals(Albero a) {
    return genere.equals(a.genere) && matricola==a.matricola;
  }

  public int compareTo(Albero a) {
    int ret = this.genere.compareTo(a.genere);
    if(ret==0) ret = this.anno - a.anno;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Strada {
  private String nome, quartiere;
  private Set<Albero> s;

  public Strada(String nome, String quartiere) {
    this.nome = nome;
    this.quartiere = quartiere;
    s = new TreeSet<Albero>();
  }

  public String getNome() { return nome; }
  public String getQuartiere() { return quartiere; }
  public String toString() { return nome + " (" + quartiere + ": " + s;}

  public boolean equals(Object o) { return equals((Strada) o); }
  public boolean equals(Strada s){ return nome.equals(s.nome); }
  public boolean aggiungi(Albero a) { return s.add(a); }

  public Set<Albero> genere(String s) {
    Set<Albero> result = new HashSet<Albero>();
    for(Albero a: this.s) if(a.getGenere().equals(s)) result.add(a);
    return result;
  }

  public int vecchi(int anno) {
    int totale=0;
    for(Albero a: this.s) if(a.getAnno()<anno) totale++;
    return totale;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Strada> l = new ArrayList<Strada>();
    Scanner scanner = new Scanner(System.in);
    Strada s = new Strada(scanner.nextLine(), scanner.nextLine());
    l.add(s);
    Albero a = new Albero(scanner.nextLine(), scanner.nextInt(),
        scanner.nextInt());
    if(!s.aggiungi(a)) System.out.println("Inserimento non riuscito!");
    int totale=0;
    for(Strada x: l) totale+=x.vecchi(2000);
    System.out.println(totale);
    String nome = scanner.nextLine();
    Albero max = null;
    for(Strada x: l) {
      Set<Albero> t=x.genere(nome);
      if(!t.isEmpty()) System.out.println(x.getNome());
      for(Albero z: t)
        if(max==null||z.compareTo(max)<0) max=z;
    }
    System.out.println(max);
  }
}
```