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

Appello del 14/7/2017

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

Esercizio 1 (4 punti)

1. Illustrare i concetti di over-riding e di late binding di un metodo.
2. Definire il concetto di complessità asintotica di un algoritmo.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **–0,0235** 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 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 j=0, sum=0;
  while(j<N)
    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).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `M` e `N` (si supponga `M<N`).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `M`.

Esercizio 4 (5 punti)

Dopo anni di gavetta, finalmente l'ingegnere civile Riccardo Insidio ha coronato il suo sogno: un'impresa di costruzioni tutta sua! Per questo, l'ing. Insidio desidera gestire all'interno di un calcolatore le informazioni relative alle squadre di costruzione dell'azienda. In particolare, per ogni lavoratore occorre registrare il cognome, la data di nascita e il codice fiscale. Si scriva una classe `Lavoratore` per l'ing. Insidio 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 lavoratore.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Lavoratore` (la verifica va fatta esclusivamente sul codice fiscale).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Lavoratore` passato come parametro (per età crescente e, a parità, per ordine alfabetico sul cognome).

Esercizio 5 (7 punti)

Si scriva una classe `Squadra` che registri le informazioni riguardanti una singola squadra di costruzione dell'ing. Insidio. Per ogni squadra occorre memorizzare l'indirizzo del cantiere, mentre i lavoratori vanno memorizzati all'interno di una lista. La classe `Squadra` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, a lista dei lavoratori è vuota).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della squadra (inclusa la descrizione di tutti i lavoratori).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Squadra` (la verifica va effettuata esclusivamente sull'indirizzo).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Lavoratore`, lo inserisca all'interno della lista, mantenendo quest'ultima ordinata secondo il punto 5. dell'Esercizio 4.
6. Possedere il metodo `lavora` che, dato il cognome di un lavoratore, indichi se tale squadra contiene o meno un lavoratore con tale nome.
7. Possedere il metodo `vecchi` che, dato un anno, restituisca un insieme contenente tutti i lavoratori nati prima di tale anno.

Esercizio 6 (9 punti)

Si scriva un'applicazione per l'ing. Insidio che:

1. Crei un array di 10 oggetti `Squadra`.
2. Crei un oggetto `Squadra`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. nella prima posizione dell'array di cui al punto 1.
4. Crei un oggetto `Lavoratore`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno della squadra di cui al punto 2.
6. Stampi a video il cognome di tutti i lavoratori nati prima dell'anno 1950.
7. Letto da tastiera un cognome, stampi a video l'indirizzo di tutti i cantieri in cui lavorano squadre cui partecipa un lavoratore con tale cognome.
8. Stampi a video il codice fiscale del lavoratore più giovane tra quelli della squadra di cui al punto 2. (si supponga che tutti i lavoratori siano nati prima del 2002).

Soluzione Esercizio 2

$-0{,}0235_{10} = -0{,}000001100000\ldots_2$ quindi la mantissa è **(1)1100000**, l'esponente -5_{10} = **11111011**.
Il numero rappresentato è $-2^{-6} + 2^{-7}$, 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]`	$N - M + 1$	o 1
`i++<N`	$N - M + 1$	o 1
Totale	$2N - 2M + 4$	o 4

Domanda 2:

2 assegnamenti	2
`j<N`	$N + 1$
`sum+=f(V, j++, M)`	N
complessità di `f`	$M^2 + M + 4N$
Totale	$M^2 + M + 6N + 3$

complessità. di `f`: $\sum_{j=0}^{M-1}(2M - 2j + 4) + \sum_{j=M}^{N-1} 4 = 2M^2 - M(M-1) + \sum_{j=0}^{N-1} 4 = M^2 + M + 4N$

Domanda 3:

Complessità asintotica: $O(M^2)$

Soluzione Esercizio 4

```
class Lavoratore implements Comparable<Lavoratore> {
  private String cognome, cf;
  private int g, m, a;

  public Lavoratore(String cognome, String cf, int g, int m, int a) {
    this.cognome = cognome;
    this.cf = cf;
    this.g = g; this.m = m; this.a = a;
  }

  public String getCognome() { return cognome; }
  public String getCF() { return cf; }
  public String getData() { return g+"/"+m+"/"+a; }

  public String toString() {
    return cognome + " (" + cf + "): " + getData();
  }

  public boolean equals(Object o) { return equals((Lavoratore) o); }
  public boolean equals(Lavoratore l) { return cf.equals(l.cf); }

  public int compareTo(Lavoratore l) {
    int ret = l.a-this.a;
    if(ret==0) ret = l.m - this.m;
    if(ret==0) ret = l.g - this.g;
    if(ret==0) ret = this.cognome.compareTo(l.cognome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Squadra {
  private String indirizzo;
  private List<Lavoratore> l;
  public Squadra(String indirizzo) {
    this.indirizzo = indirizzo;
    l = new LinkedList<Lavoratore>();
  }

  public String getIndirizzo() { return indirizzo; }
  public String toString() { return indirizzo + ": " + l;}
  public boolean equals(Object o) { return equals((Squadra) o); }
  public boolean equals(Squadra s){
    return indirizzo.equals(s.indirizzo);
  }

  public void aggiungi(Lavoratore l) {
    int i = 0;
    while((i<this.l.size())&&(this.l.get(i).compareTo(l)<0)) i++;
    this.l.add(i, l);
  }
  public boolean lavora(String cognome) {
    for(Lavoratore l: this.l)
      if(l.getCognome().equals(cognome)) return true;
    return false;
  }
  public Set<Lavoratore> vecchi(int anno) {
    Set<Lavoratore> result = new TreeSet<Lavoratore>();
    Lavoratore target = new Lavoratore("", "", 1, 1, anno);
    for(Lavoratore l: this.l) if(l.compareTo(target)>0) result.add(l);
    return result;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Squadra[] v = new Squadra [10];
    Scanner scanner = new Scanner(System.in);
    v[0] = new Squadra(scanner.nextLine());
    Lavoratore l = new Lavoratore(scanner.nextLine(),scanner.nextLine(),
      scanner.nextInt(),scanner.nextInt(), scanner.nextInt());
    v[0].aggiungi(l);
    for(int i=0; i<10; i++)
      for(Lavoratore l: v[i].vecchi(1950))
        System.out.println(l.getCognome());
    String cognome = scanner.nextLine();
    for(int i=0; i<10; i++)
      if(v[i].lavora(cognome)) System.out.println(v[i].getIndirizzo());
    Lavoratore max = null;
    Set<Lavoratore> s = v[0].vecchi(2002);
    for(Lavoratore l: s) if(max==null || l.compareTo(max)<0) max=l;
    System.out.println(max.getCF());
  }
}
```