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

Appello del 14/1/2010

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

Esercizio 1 (4 punti)

1. Descrivere i tipi primitivi del linguaggio Java.
2. Descrivere la codifica dei numeri interi all'interno di un elaboratore elettronico.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **–0,12** supponendo di utilizzare 8 bit per la mantissa (in modulo e segno) ed 8 bit per l'esponente (in complemento a 2).

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

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

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

1. Calcolare la complessità in passi base del metodo `f` nei termini dei parametri `M` ed `N` (suggerimento: si supponga `N>M`).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `M` ed `N` (suggerimento: si supponga `N` pari e `N>M`).
3. Calcolare la complessità asintotica del metodo `g` nei termini dei parametri `M` ed `N`.

Esercizio 4 (6 punti)

Il sito di scommesse online "spoilingU" vuole memorizzare le informazioni sui propri iscritti all'interno di un calcolatore. In particolare, per ogni giocatore viene registrato il nome, l'indirizzo e-mail, l'anno di nascita (per scommettere occorre essere maggiorenni) ed il numero di carta di credito. Si scriva una classe `Giocatore` per il sito "spoilingU" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca la descrizione del giocatore.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Giocatore` (l'uguaglianza va verificata unicamente sull'indirizzo e-mail).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Giocatore` passato come parametro (la precedenza va verificata in ordine alfabetico per nome e, in caso di omonimia, per età crescente).

Esercizio 5 (7 punti)

Si scriva una classe `Scommessa` che memorizzi le informazioni relative ad una scommessa. Oltre al giocatore che ha realizzato la scommessa occorre memorizzare la descrizione dell'evento su cui si scommette e la somma scommessa. La classe `Scommessa` deve inoltre:

1. Presentare un opportuno costruttore.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca la descrizione della scommessa (includendo il solo indirizzo mail del giocatore che ha effettuato la scommessa).
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Scommessa` (l'uguaglianza va verificata sulla descrizione dell'evento e sull'identicità del giocatore).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Scommessa` passato come parametro (la precedenza va verificata in ordine di giocatore e, in caso di parità, per somma scommessa crescente).

Esercizio 6 (8 punti)

Si scriva un'applicazione per il sito di scommesse "spoilingU" che:

1. Crei un insieme di oggetti `Giocatore`.
2. Crei un oggetto `Giocatore`, 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 il giocatore sia maggiorenne e che non sia già contenuto nell'insieme.
4. Crei una lista di oggetti `Scommessa`.
5. Lette da tastiera le informazioni su una nuova `Scommessa`, effettuata dal giocatore di cui al punto 2., inserisca tale scommessa in coda alla lista di cui al punto 4.
6. Letto da tastiera il nome di un evento, stampi le informazioni sulla scommessa dalla somma più elevata relativa a tale evento e contenuta nella lista di cui al punto 4.

Soluzione Esercizio 2

$-0{,}12_{10} = -0{,}0001111010_2$ quindi la mantissa è **(1)1111010** e l'esponente -3_{10} = **11111101**.

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2
`i<N+M`	2 M+1
`sum+=V[i]`	2 M
`i++`	2 M
Totale	6 M + 3

Domanda 2:

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

complessità di `f`: $\sum_{j=0}^{M-1}(6j+3) = \frac{6}{2}M(M-1)+3M$

Domanda 3:

Complessità asintotica: $O(M^2)$

Soluzione Esercizio 4

```
class Giocatore implements Comparable<Giocatore> {
  private String nome, mail, carta;
  private int anno;

  public Giocatore(String nome, String mail, int anno, String carta) {
    this.nome=nome;
    this.mail=mail;
    this.anno=anno;
    this.carta=carta;
  }

  public String getNome() { return nome; }
  public String getMail() { return mail; }
  public int getAnno() { return anno; }
  public String getCarta() { return carta; }

  public String toString() {
    return nome + "(" + mail+ "), " + anno;
  }

  public boolean equals(Object o) { return equals((Giocatore) o); }
  public boolean equals(Giocatore g) {
    return mail.equals(g.mail);
  }

  public int compareTo(Giocatore g) {
    int ret=nome.compareTo(g.nome);
    if(ret==0) ret=g.anno-anno;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
class Scommessa implements Comparable<Scommessa> {
  private Giocatore g;
  private String evento;
  private float somma;

  public Scommessa(Giocatore g, String evento, float somma) {
    this.g=g;
    this.evento=evento;
    this.somma=somma;
  }

  public Giocatore getGiocatore() { return g; }
  public String getEvento () { return evento; }
  public float getSomma() { return somma; }

  public String toString() {
    return g.getMail() + "(" + evento+ "): " + somma;
  }

  public boolean equals(Object o) { return equals((Scommessa) o); }
  public boolean equals(Scommessa s) {
    return (evento.equals(s.evento)&&g==s.g);
  }

  public int compareTo(Scommessa s) {
    int ret=g.compareTo(s.g);
    if(ret==0)
      if(somma>s.somma) ret=1;
      else if(somma<s.somma) ret=-1;
    return ret;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Giocatore> s=new HashSet<Giocatore>(); // domanda 1
    Scanner scanner=new Scanner(System.in);
    Giocatore g=new Giocatore(scanner.next(), scanner.next(),
      scanner.nextInt(), scanner.next()); // domanda 2
    if((2010-g.getAnno()<18)||!s.add(g))
      System.out.println("Giocatore non inserito!"); // domanda 3
    List<Scommessa> l=new ArrayList<Scommessa>(); // domanda 4
    Scommessa sc=new Scommessa(g, scanner.next(), scanner.nextFloat());
    l.add(sc); // domanda 5
    String evento=scanner.next();
    Scommessa max=null;
    float somma=0;
    for(Scommessa x: l)
      if(x.getEvento().equals(evento)&&x.getSomma()>somma) {
        somma=x.getSomma(); max=x;
      }
    System.out.println(max);
  } // domanda 6
}
```