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

Appello del 13/2/2012

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

Esercizio 1 (4 punti)

1. Illustrare i componenti interni di una CPU.
2. Discutere il concetto di classe astratta.

Esercizio 2 (2 punti)

Convertire in binario i numeri **112** e **–45**, supponendo di utilizzare una rappresentazione a 8 bit in complemento a 2. Si esegua infine la somma dei due numeri, riconvertendo il risultato in base 10, motivando eventuali differenze tra il risultato ottenuto e quello atteso.

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; i<M; )
    sum+=V[i++];
  return sum;
}

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

Esercizio 4 (5 punti)

La società di elfi "Snowmen" si occupa di costruire pupazzi di neve a domicilio. A tal scopo, i dati relativi ai pupazzi realizzati dagli elfi vengono memorizzati registrando il nome del pupazzo (come è noto, ogni pupazzo possiede un nome univoco), la sua altezza (in centimetri) e l'indirizzo del committente. Si scriva una classe `Pupazzo` per gli elfi di "Snowmen" 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 una descrizione del pupazzo.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Pupazzo` (l'uguaglianza va verificata sul nome del pupazzo).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Pupazzo` passato come parametro (la precedenza va data al pupazzo più alto, in caso di parità si procede per ordine alfabetico per nome).

Esercizio 5 (7 punti)

Si scriva una classe `Squadra` che memorizzi le informazioni relative alle squadre di elfi impegnate nella realizzazione dei pupazzi. Per ogni squadra vengono memorizzati la targa della slitta ed il nome della squadra. Inoltre, i pupazzi costruiti dalla squadra vanno inseriti all'interno di un insieme. La classe `Squadra` deve inoltre:

1. Presentare un opportuno costruttore con parametri (inizialmente, una squadra non ha realizzato alcun pupazzo).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della squadra (inclusa la descrizione di tutti i pupazzi realizzati).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Pupazzo`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
5. Presentare il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Squadra` (l'uguaglianza va verificata solamente sul nome della squadra).
6. Possedere il metodo `altezzaTotale` che restituisca l'altezza cumulativa dei pupazzi realizzati dalla squadra.
7. Presentare il metodo `quanti` che restituisca il numero di pupazzi costruiti dalla squadra.

Esercizio 6 (8 punti)

Si scriva un'applicazione per la società di elfi "Snowmen" che:

1. Crei una lista di oggetti `Squadra`.
2. Crei un oggetto `Squadra`, 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 `Pupazzo`, lette da tastiera le informazioni necessarie.
5. Inserisca il pupazzo creato al punto 4. tra quelli realizzati dalla squadra di cui al punto 2., controllando che tale inserimento sia possibile.
6. Stampi a video il nome della squadra che ha realizzato più pupazzi tra quelle della lista di cui al punto 1.
7. Stampi a video l'altezza cumulativa totale dei pupazzi realizzati da tutte le squadre contenute nella lista di cui al punto 1.

Soluzione Esercizio 2

$112_{10} = 01110000_2$
$-45_{10} = 11010011_2$

$01000011_2 = 67_{10}$

```
01110000+
11010011
01000011
```

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

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

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

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

Soluzione Esercizio 4

```
class Pupazzo implements Comparable<Pupazzo> {
  private String nome, indirizzo;
  private int altezza;

  public Pupazzo(String nome, String indirizzo, int altezza) {
    this.nome=nome;
    this.indirizzo=indirizzo;
    this.altezza=altezza;
  }

  public String getNome() { return nome; }
  public String getIndirizzo () { return indirizzo; }
  public int getAltezza () { return altezza; }

  public String toString() {
    return nome + "(" + altezza + "): " + indirizzo;
  }

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

  public int compareTo(Pupazzo p) {
    int ret=p.altezza-this.altezza;
    if(ret==0) ret=this.nome.compareTo(p.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Squadra {
  private String nome, targa;
  private Set<Pupazzo> s;

  public Squadra(String nome, String targa) {
    this.nome=nome;
    this.targa=targa;
    s=new HashSet<Pupazzo>();
  }

  public String getNome() { return nome; }
  public String getTarga() { return targa; }

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

  public boolean aggiungi(Pupazzo p) { return s.add(p); }

  public boolean equals(Object o) { return equals((Squadra) o); }
  public boolean equals(Squadra s) { return nome.equals(s.nome); }

  public int quanti() { return s.size(); }

  public int altezzaTotale() {
    int cm=0;
    for(Pupazzo p:s) cm+=p.getAltezza();
    return cm;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Squadra> l=new LinkedList<Squadra>();
    Scanner scanner=new Scanner(System.in);
    Squadra s=new Squadra(scanner.next(), scanner.next());
    l.add(s);
    Pupazzo p=new Pupazzo(scanner.next(), scanner.next(),
      scanner.nextInt());
    if(!s.aggiungi(p)) System.out.println("Pupazzo già presente!");
    Squadra max=null;
    for(Squadra x: l) {
      if(max==null || x.quanti()>max.quanti())
        max=x;
    }
    System.out.println(max.getNome());
    int tot=0;
    for(Squadra x: l) tot+=x.altezzaTotale();
    System.out.println(tot);
  }
}
```