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

Appello del 16/1/2013

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 array e collezioni in Java.
2. Descrivere le modalità di studio della complessità temporale di un algoritmo.

Esercizio 2 (2 punti)

Convertire in binario i numeri **15** e **114**, 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 N) {
  int i, sum=0;
  for(i=0; i<N; i++)
    sum+=V[++i];
  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` pari e si esprima `j=2·i`).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (6 punti)

L'agenzia di spedizioni "Tirapacchi" sta riprogettando il suo sistema informativo per tenere traccia delle spedizioni in corso. Ogni spedizione è caratterizzata dal nome e dall'indirizzo del destinatario e dal numero di giorni entro cui deve essere consegnata (valore negativo in caso di ritardi). Si scriva una classe `Spedizione` per l'agenzia "Tirapacchi" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Possieda il metodo `giornoTrascorso` che riduce di una unità il tempo limite di consegna.
4. Presenti il metodo `toString` che fornisca una descrizione della fattura.
5. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Spedizione` (verificata sul nome e l'indirizzo del destinatario e sul tempo limite di consegna).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Spedizione` passato come parametro (la precedenza per ordine crescente di limite di consegna. In caso di parità si procede in ordine alfabetico per il nome e quindi per l'indirizzo del destinatario).

Esercizio 5 (7 punti)

Si scriva una classe `Filiale` che memorizzi le informazioni relative alle spedizioni gestite da una specifica filiale dell'agenzia "Tirapacchi" all'interno di un insieme. Per ogni filiale si memorizzi inoltre l'indirizzo della sede operativa ed il numero di spedizioni che può di gestire. La classe `Filiale` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, una filiale non ha spedizioni).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della filiale (inclusa la descrizione di tutte le spedizioni che gestisce).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Spedizione`, lo aggiunga a quelli gestiti dalla filiale. L'aggiunta non è possibile se il numero di spedizioni gestite ha raggiunto il limite massimo: in questo caso il metodo restituisce `false`.
5. Presentare il metodo `giornoTrascorso` che riduca di una unità il tempo limite di consegna per tutte le spedizioni gestite.
6. Possedere il metodo `urgente` che restituisca la spedizione con il tempo limite di consegna più basso.

Esercizio 6 (7 punti)

Si scriva un'applicazione per l'agenzia "Tirapacchi" che:

1. Crei una lista di oggetti `Filiale`.
2. Crei un oggetto `Filiale`, 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 `Spedizione`, lette da tastiera le informazioni necessarie.
5. Inserisca la spedizione creata al punto 4. tra quelle associate alla filiale di cui al punto 2, controllando che l'inserimento sia possibile.
6. Simuli il passaggio di un giorno, decrementando opportunamente il tempo limite di consegna per tutte le spedizioni gestite da tutte le filiali.
7. Stampi a video la spedizione più urgente tra quelle delle filiali nella lista di cui al punto 1.

Soluzione Esercizio 2

$15_{10} = 00001111_2$
$114_{10} = 01110010_2$

$10000001_2 = -127_{10}$

```
00001111+
01110010
10000001
```

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`i<N`	N/2 + 1	o N/2 + 3/2
`sum+=V[++i]`	N/2	o N/2 + 1/2
`i++`	N/2	o N/2 + 1/2
Totale	3N/2 +3	o 3N/2 +9/2

Domanda 2:

2 assegnamenti	2
`while(j<M)`	N/2 + 2
`sum+=f(V, j++)`	N/2 + 1
complessità di `f`	$3N^2/8 + 9N/4 + 3$
Totale	$3N^2/8 + 13N/4 + 8$

complessità di `f`: $\sum_{\substack{j=0 \\ (j \text{ pari})}}^{N} (3j/2+3) = \sum_{i=0}^{N/2}(3i+3) = 3N^2/8 + 9N/4 + 3$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 4

```
class Spedizione implements Comparable<Spedizione> {
  private String nomeDest;
  private String indDest;
  private int limite;

  public Spedizione(String nomeDest, String indDest, int limite) {
    this.nomeDest=nomeDest;
    this.indDest=indDest;
    this.limite=limite;
  }

  public String getNomeDest() { return nomeDest; }
  public String getIndDest() { return indDest; }
  public int getLimite() { return limite; }
  public void giornoTrascorso() { limite--; }

  public String toString() {
    return nomeDest + " (" + indDest + ") entro " + limite + " giorni";
  }

  public boolean equals(Object o) { return equals((Spedizione) o); }
  public boolean equals(Spedizione s) {
    return nomeDest.equals(s.nomeDest) &&
    indDest.equals(s.indDest) && limite==s.limite;
  }

  public int compareTo(Spedizione s) {
    int ret=limite-s.limite;
    if(ret==0) ret=nomeDest.compareTo(s.nomeDest);
    if(ret==0) ret=indDest.compareTo(s.indDest);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Filiale {
  private String sede;
  private int capacita;
  private Set<Spedizione> s;

  public Filiale(String sede, int capacita) {
    this.sede = sede;
    this.capacita = capacita;
    s=new TreeSet<Spedizione>();
  }

  public String getSede() { return sede; }
  public String toString() {
    return "sede di " + sede + ": " + s.toString();
  }

  public boolean aggiungi(Spedizione x) {
    if(s.size()==capacita) return false;
    return s.add(x);
  }

  public void giornoTrascorso(){ for(Spedizione x: s) x.giornoTrascorso(); }

  public Spedizione urgente() {
    Spedizione max = null;
    for (Spedizione x : s) if(max==null || x.compareTo(max)<0) max = x;
    return max;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Filiale> l=new ArrayList<Filiale>();
    Scanner scanner=new Scanner(System.in);
    Filiale f=new Filiale(scanner.nextLine(), scanner.nextInt());
    Spedizione s=new Spedizione(scanner.nextLine(), scanner.nextLine(),
      scanner.nextInt());

    l.add(f);
    if(!f.aggiungi(s)) System.out.println("Spedizione già presente!");
    for(Filiale x : l) x.giornoTrascorso();

    Spedizione max=null;
    for(Filiale x: l) {
      Spedizione p=x.urgente();
      if(max==null || p.compareTo(max)<0) max=p;
    }
    System.out.println(max);
  }
}
```