Esame di Fondamenti di Informatica L-B
Ingegneria Gestionale e dei Processi Gestionali (L-Z)

Appello del 13/9/2007

Esercizio 1 (4 punti)

Discutere le principali differenze esistenti tra gli array e le collezioni in Java.

Esercizio 2 (7 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int K) {
  int i=0;
  while(i<K/2) {
    if(V[i]!=V[K-i-1]) return 0;
    i++;
  }
  return 1;
}

int g(int V[][], int N, int M) {
  for(int j=0; j<N; j++)
    if(!f(V[j], M)) return 0;
  return 1;
}
```

1. Calcolare la complessità in passi base (nel caso peggiore) della funzione `f` nei termini del parametro `K` (suggerimento: si distinguano i casi in cui `K` assume valori pari da quelli in cui assume valori dispari).
2. Calcolare la complessità in passi base (nel caso peggiore) della funzione `g` nei termini dei parametri `N` ed `M` (suggerimento: si distinguano i casi in cui `M` assume valori pari da quelli in cui assume valori dispari).
3. Calcolare la complessità asintotica della funzione `g` nei termini dei parametri `N` ed `M`.
4. Indicare per quali valori della matrice `V` la funzione `g` restituisce il valore 1.

Esercizio 3 (9 punti)

Gli alieni mutanti del pianeta Croma si stanno preparando ad invadere nuovamente il pianeta Eldenia. Per riscattare la debacle dello scorso Luglio, i capi dell'esercito stanno approntando delle squadre di robot invasori. Tali robot sono caratterizzati da un nome, un modello ed un colore. Inoltre, ogni robot mantiene memoria di tutte le battaglie che ha sostenuto all'interno di un insieme: ogni battaglia è rappresentata da una descrizione testuale.

Si scriva una classe `Robot` per il pianeta Croma che:

1. Possieda un opportuno costruttore con parametri (inizialmente, il robot non ha mai combattuto).
2. Presenti opportuni metodi per accedere al nome, al modello ed al colore del robot.
3. Possieda un metodo `combatte` che, data la descrizione di una battaglia, la inserisca all'interno dell'insieme delle battaglie combattute, se questa non è già presente.
4. Possieda un metodo `haCombattuto` che, data la descrizione di una battaglia, indichi se il robot ha combattuto o meno in tale battaglia.
5. Presenti il metodo `toString` che fornisca una descrizione testuale del robot (incluse le descrizioni delle battaglie combattute).
6. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Robot` (l'uguaglianza va verificata esclusivamente sul nome).
7. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un altro oggetto `Robot`, dando la precedenza al robot con maggiore esperienza in battaglia.

Esercizio 4 (7 punti)

Si scriva una classe `Squadra` che memorizzi le informazioni riguardanti le squadre di robot. Oltre al nome della squadra, vanno memorizzati i robot partecipanti all'interno di un vettore. La classe `Squadra` deve inoltre:

1. Possedere un opportuno costruttore (all'atto della costituzione della squadra occorre specificare il numero massimo di robot all'interno della squadra stessa).
2. Presentare un metodo `completa` che indichi se la squadra è al completo.
3. Possedere un metodo `aggiungi` che, dato un oggetto `Robot`, lo inserisca all'interno della squadra, se il numero massimo di partecipanti non è già stato raggiunto. Se, viceversa, la squadra è al completo, il nuovo robot deve sostituire il robot con minor esperienza della squadra, a patto che questo non abbia un'esperienza superiore al nuovo robot.
4. Presentare il metodo `toString` che restituisca una descrizione della squadra, compresi tutti i robot.

Esercizio 5 (5 punti)

Si scriva un'applicazione per il pianeta Croma che:

1. Crei una lista di oggetti di tipo `Squadra`.
2. Lette da tastiera le informazioni relative ad una nuova squadra, provveda ad inserire tale oggetto all'interno della lista.
3. Lette da tastiera le informazioni relative ad un nuovo robot, provveda ad inserire tale oggetto all'interno della squadra di cui al punto 2.
4. Lette da tastiera le descrizioni di alcune battaglie provveda ad aggiornare le battaglie combattute dal robot di cui al punto 3.
5. Stampi su video le informazioni relative a tutte le squadre della lista.

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `char leggiChar()` Legge un singolo carattere.
- `double leggiDouble()` Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` Legge un intero (delimitato da spazi).
- `String leggiLinea()` Legge una linea di testo.
- `String leggiString()` Legge una parola senza spazi al suo interno.

Soluzione Esercizio 2

Domanda 1:

1 assegnamento	1
`while(i<K/2)`	K/2 + 1 o K/2 + 1/2
`if(V[i]!=V[K-i-1])`	K/2 o K/2 – 1/2
`i++`	K/2 o K/2 – 1/2
Totale (K pari)	3K/2 + 2
Totale (K dispari)	3K/2 + 1/2

Domanda 2:

1 assegnamento	1
`for`	N + 1
incremento di `j`	N
chiamata di `f`	N
complessità di `f`	3MN/2 + 2N o 3MN/2 + N/2
Totale (M pari)	3MN/2 + 5N + 2
Totale (M dispari)	3MN/2 + 7N/2 + 2

Domanda 3:
Complessità asintotica: *O*(MN)

Domanda 4:
La funzione `g` restituisce il valore 1 solo se tutte le righe della matrice `V` sono palindrome, ovvero sono indistinguibili se lette sia da destra che da sinistra.

Soluzione Esercizio 3

```
import java.util.*;
class Robot implements Comparable<Robot> {
  private String nome, modello, colore;
  private Set<String> battaglie;

  public Robot(String nome, String modello, String colore) {
    this.nome=nome;
    this.modello=modello;
    this.colore=colore;
    this.battaglie=new HashSet<String>();
  }

  public String getNome() { return nome; }
  public String getModello() { return modello; }
  public String getColore() { return colore; }

  public boolean haCombattuto(String battaglia) {
    return battaglie.contains(battaglia);
  }

  public boolean combatte(String battaglia) {
    return battaglie.add(battaglia);
  }

  public String toString() {
    return nome+" ("+modello+", "+colore+")\n"+battaglie;
  }

  public boolean equals(Robot r) { return (nome.equals(r.nome)); }
  public boolean equals(Object o) { return equals((Robot) o); }

  public int compareTo(Robot r) {
    return (r.battaglie.size()-battaglie.size());
  }
}
```

Soluzione Esercizio 4

```
class Squadra {
  private String nome;
  private Robot[] v;
  private int quanti;

  public Squadra(String nome, int quanti) {
    this.nome=nome;
    this.v=new Robot[quanti];
    this.quanti=0;
  }

  public boolean completa() { return quanti==v.length; }

  public void aggiungi(Robot r) {
    if(!completa()) v[quanti++]=r;
    else {
      int peggiore=0;
      for(int i=1; i<quanti; i++)
        if(v[i].compareTo(v[peggiore])>0) peggiore=i;
      if(r.compareTo(v[peggiore])<0) v[peggiore]=r;
    }
  }

  public String toString() {
    String s=nome+"\n";
    for(int i=0; i<quanti; i++) s+=v[i].toString()+", ";
    return s;
  }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    List<Squadra> l=new ArrayList<Squadra>(); // domanda 1
    Squadra s=new Squadra(Lettore.in.leggiLinea(),
      Lettore.in.leggiInt());
    Robot r;
    int n;

    l.add(s); // domanda 2
    r=new Robot(Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
      Lettore.in.leggiLinea());
    s.aggiungi(r); // domanda 3
    n=Lettore.in.leggiInt();
    for(int i=0; i<n; i++)
      r.combatte(Lettore.in.leggiLinea()); // domanda 4
    System.out.println(l); // domanda 5
  }
}
```