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

Appello del 20/7/2007

Compito A

Esercizio 1 (4 punti)

Discutere i concetti della programmazione modulare.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
void f(int V[], int M) {
  int i=0, temp;
  while(i<M/2) {
    temp=V[i];
    V[i]=V[M-i-1];
    V[M-i-1]=temp;
    i++;
  }
}

int g(int V[], int N) {
  for(int j=0; j<N; j+=2) f(V, j);
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini del parametro `M` (suggerimento: si distinguano i casi in cui `M` assume valori pari da quelli in cui assume valori dispari).
2. Calcolare la complessità in passi base della funzione `g` nei termini del parametro `N` (suggerimento: si esprima `j`=2·`i` e si supponga `N` pari).
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (9 punti)

Gli abitanti del pianeta Eldenia devono fronteggiare una nuova invasione da parte degli alieni mutanti del pianeta Croma. Tali forme viventi sono in grado di assumere la forma di diversi mezzi civili e non; per questo è necessario catalogare gli avvistamenti in maniera tale da studiare le forme aliene. Gli alieni risultano essere contraddistinti da una sigla, da un insieme di possibili forme (es. “pick-up”, “elicottero”, ecc.) e dalla data dell’ultimo avvistamento.
Si scriva una classe `Alieno` per il pianeta Eldenia che:

1. Possieda un opportuno costruttore con parametri (inizialmente, l’alieno possiede solo la sigla, in quanto deve essere ancora avvistato).
2. Presenti opportuni metodi per accedere alla sigla e alla data dell'ultimo avvistamento.
3. Possieda un metodo `assumeForma` che, data una forma, controlli se questa è presente all’interno dell’insieme delle forme note.
4. Presenti un metodo `avvista` che, data una forma ed una data di avvistamento, aggiorni in maniera opportuna le variabili d’istanza.
5. Presenti il metodo `toString` che fornisca una descrizione testuale dell’alieno (incluse le possibili forme mutanti).
6. Possieda il metodo `equals` per stabilire l’uguaglianza con un altro oggetto `Alieno` (l’uguaglianza va verificata esclusivamente sulla sigla).
7. Implementi l’interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un altro oggetto `Alieno`, dando la precedenza all’alieno avvistato più di recente.

Esercizio 4 (7 punti)

Si scriva una classe `Avvistamenti` che memorizzi le informazioni riguardanti tutti gli avvistamenti all’interno di una lista. Tale classe deve:

1. Possedere un opportuno costruttore (inizialmente non ci sono avvistamenti).
2. Presentare un metodo `aggiungi` che, dato un oggetto `Alieno`, lo inserisca in coda alla lista.
3. Possedere il metodo `cerca` che, data una sigla, restituisca l’oggetto `Alieno` corrispondente all’interno della lista, se questo esiste.
4. Presentare un metodo `alieni` che restituisca un insieme contenente le sigle di tutti gli alieni avvistati.
5. Possedere il metodo `toString` che restituisca una descrizione di tutti gli avvistamenti.

Esercizio 5 (6 punti)

Si scriva un’applicazione per il pianeta Eldenia che:

1. Crei un oggetto di tipo `Avvistamenti`.
2. Lette da tastiera le informazioni relative ad una nuova forma aliena, provveda ad inserire tale oggetto tra gli avvistamenti.
3. Letta da tastiera una sigla, provveda a cercare l’alieno corrispondente tra gli avvistamenti e ad aggiornarne opportunamente i dati (letti da tastiera) relativi ad un nuovo avvistamento.
4. Stampi su video le informazioni relative all’alieno avvistato più di recente.

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<M/2)`	$M/2 + 1$ o $M/2 + 1/2$
4 assegnamenti	$2M$ o $2M – 2$
Totale (M pari)	$5M/2 + 2$
Totale (M dispari)	$5M/2 – 1/2$

Domanda 2:

1 assegnamento	1
`for`	$N/2 + 1$
incremento di `j`	$N/2$
chiamata di `f`	$N/2$
complessità di `f`	$5N^2/8 – N/4$
Totale	$5N^2/8 + 5N/4 + 2$

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

Soluzione Esercizio 3

```
import java.util.*;
class Alieno implements Comparable<Alieno> {
  private String sigla;
  private int g, m, a;
  private Set<String> forme;

  public Alieno(String sigla) {
    this.sigla=sigla;
    this.g=0; this.m=0; this.a=0;
    this.forme=new HashSet<String>();
  }

  public String getSigla() { return sigla; }
  public String getData() {
    if(g>0) return ""+g+"/"+m+"/"+a; }
    return "(non avvistato)";
  }

  public boolean assumeForma(String forma) {
    return forme.contains(forma);
  }

  public void avvista(String forma, int g, int m, int a) {
    forme.add(forma);
    this.g=g;
    this.m=m;
    this.a=a;
  }

  public String toString() { return sigla+" "+ getData()+"\n"+forme; }

  public boolean equals(Alieno p) { return (sigla.equals(p.sigla));   }

  public int compareTo(Alieno p) {
    int ret=(p.a-a);
    if(ret==0) ret=(p.m-m);
    if(ret==0) ret=(p.g-g);
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Avvistamenti {
  private List<Alieno> l;

  public Avvistamenti() { this.l=new LinkedList<Alieno>(); }

  public void aggiungi(Alieno a) { l.add(a); }

  public Alieno cerca(String sigla) {
    for(Alieno a:l)
      if(a.getSigla().equals(sigla)) return a;
    return null;
  }

  public Set<String> alieni() {
    Set<String> s=new TreeSet<String>();
    for(Alieno a:l) s.add(a.getSigla());
    return s;
  }

  public String toString() { return l.toString(); }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    Avvistamenti a=new Avvistamenti(); // domanda 1

    a.aggiungi(new Alieno(Lettore.in.leggiLinea()));// domanda 2
    Alieno p=a.cerca(Lettore.in.leggiLinea());
    if(p!=null)
      p.avvista(Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
        Lettore.in.leggiInt(), Lettore.in.leggiInt()); // domanda 3
    Set<String> s=a.alieni();
    Alieno recente=null;
    for(String t:s) {
      p=a.cerca(t);
      if((recente==null)||(p.compareTo(recente)<0)) recente=p;
    }
    System.out.println(recente); // domanda 4
  }
}
```

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

Appello del 20/7/2007

Compito B

Esercizio 1 (4 punti)

Discutere i concetti della programmazione modulare.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
void f(int V[], int M) {
  for(int i=M/2; i>0; i--) {
    int temp=V[M-i-1];
    V[M-i-1]=V[i];
    V[i]=temp;
  }
}

int g(int V[], int N) {
  int j=1;
  while(j<N) {
    f(V, j);
    j+=2;
  }
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini del parametro `M` (suggerimento: si distinguano i casi in cui `M` assume valori pari da quelli in cui assume valori dispari).
2. Calcolare la complessità in passi base della funzione `g` nei termini del parametro `N` (suggerimento: si esprima `j`=2·`i` + 1 e si supponga `N` dispari).
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (8 punti)

Il professor Selti sta preparando la nuova edizione della sua opera più famosa, "Il dizionario dei sinonimi". A tal scopo, per ogni voce, oltre al termine principale, va memorizzato un insieme di sinonimi.

Si scriva una classe `Voce` per il professor Selti che:

1. Possieda un opportuno costruttore con parametri (inizialmente la voce non ha sinonimi).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda un metodo `quanti` che indichi quanti sinonimi possiede la voce.
4. Possieda un metodo `aggiungiSinonimo` che, dato un termine, lo aggiunga tra i sinonimi della voce.
5. Presenti un metodo `isSinonimo` che controlli se il termine corrispondente è all'interno dei sinonimi dell'oggetto `Voce` dato.
6. Presenti il metodo `toString` che fornisca una descrizione della voce (inclusi tutti i sinonimi).
7. Possieda un metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Voce` (l'uguaglianza va verificata esclusivamente sul termine).
8. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un altro oggetto `Voce`, secondo l'ordine lessicografico dei termini.

Esercizio 4 (9 punti)

Si scriva una classe `Dizionario` che memorizzi le informazioni relative a tutte le voci del dizionario all'interno di una lista. Tale classe deve:

1. Possedere un opportuno costruttore (inizialmente non ci sono voci).
2. Presentare un metodo `aggiungiVoce` che, dato un oggetto `Voce`, lo inserisca all'interno della lista, mantenendo ordinata la lista secondo il criterio espresso al punto 8. dell'esercizio 3. (suggerimento: si utilizzi il metodo `add(int i, Voce v)` della classe `List<Voce>`).
3. Possedere il metodo `cercaVoce` che, dato un termine, restituisca l'oggetto `Voce` corrispondente all'interno della lista, se questo esiste.
4. Presentare un metodo `sinonimi` che, dato un oggetto `Voce`, restituisca un insieme contenente tutte le voci che hanno tale voce come sinonimo.
5. Possedere il metodo `toString` che restituisca un elenco delle descrizioni di tutte le voci all'interno del dizionario.

Esercizio 5 (5 punti)

Si scriva un'applicazione per il professor Selti che:

1. Crei un oggetto di tipo `Dizionario`.
2. Lette da tastiera le informazioni relative ad alcune nuove voci, provveda ad inserire tali oggetti nel dizionario.
3. Letti da tastiera due termini, provveda a cercare il primo termine all'interno del dizionario e, se questo è presente, vi aggiunga il secondo termine come sinonimo.
4. Letto da tastiera un termine, provveda a stampare tutte le voci che lo contengono come sinonimo.

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
`for`	$M/2 + 1$ o $M/2 + 1/2$
decremento di `i`	$M/2$ o $M/2 – 1/2$
3 assegnamenti	$3M/2$ o $3M/2 – 3/2$
Totale (M pari)	$5M/2 + 2$
Totale (M dispari)	$5M/2 – 1/2$

Domanda 2:

1 assegnamento	1
`while(j<N)`	$N/2 + 1/2$
incremento di `j`	$N/2 – 1/2$
chiamata di `f`	$N/2 – 1/2$
complessità di `f`	$5N^2/8 – N/4$
Totale	$5N^2/8 + 5N/4 + 1/2$

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

Soluzione Esercizio 3

```
import java.util.*;
class Voce implements Comparable<Voce> {
  private String termine;
  private Set<String> sinonimi;

  public Voce(String termine) {
    this.termine=termine;
    this.sinonimi=new HashSet<String>();
  }

  public String getTermine() { return termine; }
  public int quanti() { return sinonimi.size(); }

  public void aggiungiSinonimo(String s) { sinonimi.add(s); }

  public boolean isSinonimo(Voce v) {
    return v.sinonimi.contains(termine);
  }

  public String toString() { return termine+"\n"+sinonimi; }
  public boolean equals(Voce v) { return (termine.equals(v.termine)); }
  public int compareTo(Voce v) { return termine.compareTo(v.termine); }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Dizionario {
  private List<Voce> l;

  public Dizionario() { this.l=new LinkedList<Voce>(); }

  public void aggiungiVoce(Voce v) {
    int i=0;
    while((i<l.size())&&(l.get(i).compareTo(v)<0)) i++;
    l.add(i, v); }

  public Voce cercaVoce(String termine) {
    for(Voce v:l)
      if(v.getTermine().equals(termine)) return v;
    return null;
  }

  public Set<Voce> sinonimi(Voce v) {
    Set<Voce> s=new TreeSet<Voce>();
    for(Voce p:l) if(p.isSinonimo(v)) s.add(p);
    return s;
  }

  public String toString() { return l.toString(); }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    Dizionario d=new Dizionario(); // domanda 1

    int n=Lettore.in.leggiInt();
    for(int i=0; i<n; i++)
      d.aggiungiVoce(new Voce(Lettore.in.leggiLinea()));// domanda 2
    Voce v=d.cercaVoce(Lettore.in.leggiLinea());
    if(v!=null)
      v.aggiungiSinonimo(Lettore.in.leggiLinea()); // domanda 3
    v=d.cercaVoce(Lettore.in.leggiLinea());
    Set<Voce> s=d.sinonimi(v);
    for(Voce t:s)
      System.out.println(t); // domanda 4
  }
}
```