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

Appello del 18/2/2009

Esercizio 1 (4 punti)

Discutere i concetti di classe astratta ed interfaccia in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M) {
  int j=0, sum=0;

  while(j<=M) {
    sum+=V[j];
    j++;
  }
  return sum;
}

void g(int U[], int V[], int N) {
  int j=0, i=N;

  for(; j<N; ) {
    U[j]=f(V, --i);
    j=N-i;
  }
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini del parametro `M`.
2. Calcolare la complessità in passi base della funzione `g` nei termini del parametro `N`.
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (6 punti)

In vista del proprio lancio sul mercato, la società di autonoleggi "Volt" sta preparando il sistema informatico per la gestione dei noleggi. In particolare, per ogni auto a disposizione vengono indicati la marca, il modello, la targa ed il chilometraggio percorso. Si scriva una classe `Auto` per la società di autonoleggi "Volt" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Possieda un metodo `aggiorna` che, dato il numero di chilometri percorsi durante un noleggio, aggiorni opportunamente le variabili di istanza dell'oggetto.
4. Presenti il metodo `toString` che fornisca la descrizione dell'auto.
5. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Auto` (l'uguaglianza va verificata unicamente sulla targa del veicolo).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un'`Auto` passata come parametro (l'ordine è dato dal chilometraggio).

Esercizio 4 (7 punti)

Si scriva una classe `Deposito` che memorizzi le informazioni relative ai vari veicoli parcheggiati all'interno di un deposito. In particolare, oltre all'elenco delle auto, che vanno inserite all'interno di una lista, occorre memorizzare la città e l'indirizzo del deposito. La classe `Deposito` deve inoltre:

1. Presentare un opportuno costruttore (inizialmente nessun veicolo è contenuto nel deposito).
2. Possedere un metodo `getIndirizzo` che restituisca l'indirizzo del deposito.
3. Presentare un metodo `getCitta` che restituisca il nome della città in cui si trova il deposito.
4. Possedere un metodo `aggiungi` che, dato un oggetto `Auto`, lo inserisca all'interno della lista che va mantenuta ordinata per valori di chilometraggio crescente (suggerimento: si utilizzi il metodo `add(int i, Auto a)` della classe `List<Auto>`).
5. Presentare un metodo `cerca` che restituisca l'`Auto` con meno chilometri presente in deposito.
6. Possedere il metodo `toString` che restituisca una stringa che fornisca una descrizione del deposito, comprendendo anche tutti i veicoli ivi presenti.

Esercizio 5 (8 punti)

Si scriva un'applicazione per la società di autonoleggi "Volt" che:

1. Crei un insieme (vuoto) di oggetti `Deposito`.
2. Crei un oggetto `Deposito`, lette da tastiera le informazioni necessarie, e lo inserisca all'interno dell'insieme di cui al punto 1.
3. Letto da tastiera il nome di una città, provveda ad inserire in un nuovo insieme tutti i depositi che si trovano in tale città.
4. Letto da tastiera un numero di chilometri massimo, stampi a video le informazioni del veicolo, tra quelli presenti nei depositi dell'insieme di cui al punto 3., che presenta il minor chilometraggio (se questo è inferiore al chilometraggio massimo).
5. Letto da tastiera il numero di chilometri indicato dal contachilometri, provveda ad aggiornare il chilometraggio del veicolo di cui al punto 4.

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:

- `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:

2 assegnamenti	2
`while`	`M` + 2
`sum+=V[i]`	`M` + 1
`j++`	`M` + 1
Totale	3 `M` + 6

Domanda 2:

2 assegnamenti	2
`for`	`N` + 1
chiamata di `f`	`N`
complessità di `f`	$3N^2/2 + 9N/2$
`j=N-i`	`N`
Totale	$3N^2/2 + 15N/2 + 3$

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

Soluzione Esercizio 3

```
class Auto implements Comparable<Auto> {
  private String marca, modello, targa;
  private int km;

  public Auto(String marca, String modello, String targa, int km) {
    this.marca = marca; this.modello = modello; this.targa = targa;
    this.km = km;
  }

  public String getTarga() { return targa; }
  public int getKM() { return km; }

  void aggiorna(int km) { this.km+=km; }

  public String toString() {
    return marca + " " + modello + "(" + targa+ ") " + km;
  }

  public boolean equals(Object o) { return equals((Auto) o); }
  public boolean equals(Auto a) { return (targa.equals(a.targa)); }

  public int compareTo(Auto a) { return this.km-a.km; }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Deposito {
  private String citta, indirizzo;
  private int cap;
  private List<Auto> veicoli;

  public Deposito(String citta, String indirizzo, int cap) {
    this.citta = citta; this.indirizzo = indirizzo;
    this.cap = cap;
    veicoli = new LinkedList< Auto >();
  }

  public String getIndirizzo() { return indirizzo; }
  public String getCitta() { return citta; }

  public void aggiungi(Auto a) {
    int i=0;
    while((i<veicoli.size())&&(veicoli.get(i).compareTo(a)<0)) i++;
    veicoli.add(i, a);
  }

  public Auto cerca() { return veicoli.get(0); }

  public String toString() {
    return indirizzo + "\n" + cap + ", "+ citta + "\n" + veicoli;
  }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Deposito> s = new HashSet<Deposito>(); // domanda 1
    s.add(new Deposito(Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
      Lettore.in.leggiInt())); // domanda 2
    String citta=Lettore.in.leggiLinea();
    Set<Deposito> c = new HashSet<Deposito>();
    for(Deposito d:s)
      if(d.getCitta().equals(citta)) c.add(d); // domanda 3
    int minkm = Lettore.in.leggiInt();
    Auto migliore=null;
    for(Deposito d:c) {
      Auto a=d.cerca();
      if(a.getKM()<minkm) {
        migliore=a;
        minkm=a.getKM();
      }
    }
    if(migliore!=null) System.out.println(migliore); // domanda 4
    migliore.aggiorna(Lettore.in.leggiInt()-migliore.getKM());
  }
}
```