Programmazione
II Compitino (Vers. B)

17 dicembre 2015

Cognome **Jackson** Nome **Michael**
Matricola **696969** Anno di corso **1**

- Nei seguenti quesiti, quando vi è richiesto di scrivere un programma, potete limitarvi al *corpo* del metodo main, assumendo se necessario che in sia una variabile di tipo Scanner, già dichiarate e inizializzate.

1. Scrivete un programma che legga un numero n, seguito da una sequenza di esattamente n stringhe; al termine, deve stampare il penultimo carattere di ciascuna stringa, dall'ultima alla prima, saltando le stringhe di lunghezza inferiore a 3.

   Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

   ```
   Quante stringhe:  6
   Stringa 1:  mammone
   Stringa 2:  p
   Stringa 3:  xyp231
   Stringa 4:  urka
   Stringa 5:  ur
   Stringa 6:  nannanni
   nk3n
   ```

(Svolgimento sul retro)

```
System.out.print("Quante stringhe: ");
int n = in.nextInt();
String x[] = new String[n];
for (int i=0; i<n; i++) {
 System.out.print("Stringa " + (i+1) + ": ");
 x[i]=in.nextLine();
}

for (int i=n; i>=0; i--)
 if (x[i].length()>=3)
  System.out.print(x[i].charAt(x[i].length()-2));
System.out.println();
```

2. Scrivete un programma che legga un numero n, seguito da una sequenza di esattamente n interi positivi; al termine, deve stampare la somma degli interi saltando quelli ripetuti.

   Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

```
Quanti interi:  6
Intero 1:  15
Intero 2:  152
Intero 3:  15
Intero 4:  1
Intero 5:  1500
Intero 6:  15
1653
```

```
System.out.print("Quanti interi: ");
int n = in.nextInt();
int x[] = new int[n];
for ( int i=0; i<n; i++ ) {
 System.out.print("Intero " + (i+1) + ": ");
 x[i]=in.nextInt();
}
int somma = 0;
for ( int i=0; i<n; i++ ) {
 int j;
 for ( j=0; j<n; j++ )
  if ( x[j] == x[i] && j != i )
   break;
 if ( j == i )
  somma += x[i]
}
System.out.println(somma);
```

3. Supponete di avere una classe astratta di nome Cane le cui istanze rappresentano dei cani. Nella classe Cane sono presenti i seguenti metodi:

   - un metodo abbaia() che restituisce (sottoforma di stringa) il rumore che il cane fa quando abbaia; questo metodo è astratto perché il rumore dipende dalla razza;
   - un metodo *statico* contaMaiuscole(String x) che restituisce il numero di lettere maiuscole[1] in una data stringa;
   - un metodo abbaiaPiuForteDi(Cane x) che restituisce true se il cane su cui è invocato abbaia più forte del cane passato come argomento; si dice che un cane abbaia più forte di un altro se il rumore che fa abbaiando contiene più lettere maiuscole.

   Implementate la classe Cane.

```
public abstract class Cane {

 public abstract void abbaia();

 public static int contaMaiuscole(String x) {
  int conta = 0;
  for (int i = 0; i < x.length(); i++)
   if (Character.isUpperCase(x.charAt(i))
    conta++;
  return conta;
 }

 public boolean abbaiaPiuForteDi(Cane x) {
  return contaMaiuscole(abbaia()) > contaMaiuscole(x.abbaia());
 }

}
```

[1] La classe Character contiene un metodo statico isUpperCase(**char** c) che restituisce true se il carattere passato come argomento è una lettera maiuscola, false altrimenti.

4. Scrivete una classe, di nome Chihuahua, che estenda la classe Cane con i seguenti costruttori e metodi

   - un costruttore che riceve il peso del cane in grammi (un valore double);
   - il metodo abbaia (il rumore che fa un chihuahua è *caiCAIcai*);
   - un metodo peso() che restituisce il peso;
   - il metodo toString() che restituisce la stringa *chihuahua (xxx g)* dove *xxx* sia il peso;
   - il metodo equals (facendo overriding del metodo della classe Object) che stabilisce che un chihuahua è uguale a un altro chihuahua purché abbiano lo stesso peso.

```
public class Chihuahua extends Cane {
 private double peso;

 public Chihuahua(double peso) {
  this.peso = peso;
 }

 public String abbaia() {
  return "caiCAIcai";
 }

 public double peso() {
  return peso;
 }

 @Override
 public String toString() {
  return "chihuahua␣(" + peso + "␣g)";
 }

 @Override
 public boolean equals(Object x) {
  if (x instanceof Chihuahua) {
   Chihuahua c = (Chihuahua)x;
   return peso == c.peso;
  }
  else
   return false;
 }
}
```

5. Scrivete una classe, di nome Molosso, che estenda la classe Cane con i seguenti costruttori e metodi

   - un costruttore che riceve l'altezza del cane in centimetri (un valore double) e una stringa che rappresenta il colore del molosso;
   - il metodo abbaia (il rumore che fa un molosso è *BAUBAUBAU*);
   - un metodo altezza() che restituisce l'altezza;
   - un metodo colore() che restituisce il colore;
   - il metodo toString() che restituisce la stringa *molosso xxx (yyy cm)* dove *xxx* sia il colore e *yyy* l'altezza;
   - il metodo equals (facendo overriding del metodo della classe Object) che stabilisce che un molosso è uguale a un altro se il loro colore è stesso, *indipendentemente dall'altezza.*

```
public class Molosso extends Cane {
 private double altezza;
 private String colore;

 public Chihuahua(double altezza, String colore) {
  this.altezza = altezza;
  this.colore = colore;
 }

 public String abbaia() {
  return "BAUBAUBAU";
 }

 public double altezza() { return altezza; }

 public double colore() { return colore; }

 @Override
 public String toString() {
  return "molosso␣" + colore + "␣(" + altezza + "␣cm)";
 }

 @Override
 public boolean equals(Object x) {
  if (x instanceof Molosso) {
   Molosso m = (Molosso)x;
   return colore.equals(m.colore);
  }
  else
   return false;
 }
}
```

6. Considerate la classe Rubrica le cui istanze rappresentano delle rubriche di contatti. La classe possiede, fra gli altri, i seguenti metodi di istanza:

   - **int** length(): restituisce il numero di contatti presenti nella rubrica;
   - Contatto get(**int** i): restituisce l'i-esimo contatto (i contatti sono numerati da 0 a $n-1$, dove n sia il numero di contatti esistenti).

   La classe Contatto ha, fra gli altri, i metodi:

   - String tel(): restituisce il numero di telefono del contatto;
   - String nominativo(): restituisce il nominativo del contatto.

   Scrivete una classe UtilitaRubrica contenente due metodi statici:

   - conta: data una Rubrica e una stringa rappresentante un numero di telefono, restituisce quanti contatti in rubrica hanno quel numero;
   - stampaDuplicati: data una Rubrica stampa i nominativi di tutti i contatti che hanno numeri di telefono duplicati (cioè, il cui numero compare più di una volta nella rubrica). Per scrivere questo metodo, utilizzate il precedente.

```
public class UtilitaRubrica {
 public static int conta(Rubrica r, String numero) {
  int n = r.length();
  int conta = 0;
  for (int i = 0; i < n; i++)
   if (r.get(i).tel().equals(numero))
    conta++;
  return conta;
 }

 public static void stampaDuplicati(Rubrica r) {
  int n = r.length();
  for (int i = 0; i < n; i++)
   if (conta(r.get(i).tel())>1)
    System.out.println(r.get(i).nominativo());
 }
}
```

7. Considerate la seguente funzione definita ricorsivamente sugli interi:

$$f(x) = \begin{cases} f(x/10) & \text{se } x \geq 10 \\ x & \text{altrimenti,} \end{cases}$$

dove $x/10$ indica la divisione intera per 10. Scrivete un metodo statico con intestazione

```
public static int f( int x )
```

per il calcolo di f.

Rispondete inoltre alle seguenti domande:

- quanto vale $f(5000)$? 5
- quanto vale $f(15)$? 1
- riuscite a dire quanto vale $f(x)$ in generale? la prima cifra di x

```
public static int f(int x) {
 if (x>10) return f(x/10);
 else return x;
}
```

8. Scrivete un metodo statico che, preso come argomento un array di Cane, restituisca un array di due elementi: il primo dei due è il cane che abbaia più forte, il secondo è quello che abbaia più piano.

```
public static Cane[] estremi(Cane[] x) {
 Cane[] risultato = new Cane[2];
 risultato[0] = risultato[1] = x[0];
 for (int i = 1; i < x.length; i++) {
  if (x[i].abbaiaPiuForte(risultato[0]))
   risultato[0]=x[i];
  if (risultato[1].abbaiaPiuForte(x[i]))
   risultato[1]=x[i];
 }
 return risultato;
}
```