Corso di Programmazione
Anno Accademico 2014–2015
Prof. Giovanni Pighizzini

Note ed esercizi aggiuntivi

9. Ancora array

Gli esercizi proposti sono utili per rivedere gli esempi riportati, che sono stati sviluppati e discussi in dettaglio a lezione.

Esempio. Calcolo del numero di occorrenze di ciascuna lettera dell'alfabeto inglese in una stringa letta da tastiera.

```
import prog.io.ConsoleInputManager;
import prog.io.ConsoleOutputManager;

class ContaLettere {

  public static void main(String[] a) {
    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    int[] n = new int['z' - 'a' + 1];  //array dei contatori
    String s = in.readLine("Stringa da esaminare? ");

    for (int i = 0; i < s.length(); i ++) {
      char ch = Character.toLowerCase(s.charAt(i));
      if (ch >= 'a' && ch <= 'z')
        n[ch - 'a']++;
    }

    for (char ch = 'a'; ch <= 'z'; ch++)
      out.println("Numero di occorrenze della lettera " + ch
                  + ": " + n[ch - 'a']);
  }
}
```

Note
La soluzione presentata è piuttosto a basso livello e per nulla object oriented, tuttavia è interessante per l'uso degli array e del tipo `char`. L'array contiene una posizione per ogni lettera dell'alfabeto: la posizione `0` per `'a'`, `1` per `'b'`, `2` per `'c'`, ecc. Utilizzando le operazioni aritmetiche sul tipo `char`, se la variabile `ch` contiene una lettera minuscola, la posizione corrispondente è il risultato dell'espressione intera `ch - 'a'`. Si noti anche il ciclo `for` e l'uso dell'operatore di incremento su `ch` per esaminare tutte le lettere da `'a'` a `'z'`.

Gli array sono oggetti: in mancanza di altre indicazioni i dati all'interno degli oggetti sono inizializzati automaticamente (nel caso del tipo `int` i dati vengono inizializzati a `0`). Per questa ragione non è necessario scrivere esplicitamente un ciclo che inizializzi gli elementi dell'array `n` a `0`.

Esercizio 9.1
Modificate il codice precedente in modo che esamini una sequenza di stringhe, anziché una sola stringa, indicando per ciascuna lettera il numero totale di occorrenze nella sequenza di stringhe letta. L'inserimento della stringa vuota indica la fine della sequenza. Fate inoltre in modo che la fase di visualizzazione elenchi solo il numero di occorrenze delle lettere che appaiono almeno una volta.

Esempio. Elenca i nomi dei mesi dell'anno indicando il numero di giorni di ciascun mese.

```
import prog.utili.MeseDellAnno;

class ElencaMesi {
  public static void main(String[] a) {
    MeseDellAnno[] mesi = MeseDellAnno.values();
    for (MeseDellAnno m: mesi)
      System.out.println(m + " ha " + m.numeroGiorni() + " giorni");
  }
}
```

Note

- Le classi enumerative (`enum`) forniscono un metodo *statico* di nome `values` che restituisce un array contenente i riferimenti agli oggetti della classe (nell'ordine fissato nella definizione della classe stessa). Ad esempio, il metodo `values` di `MeseDellAnno` fornisce il riferimento ad un array contenente i riferimenti ai 12 oggetti della classe `MeseDellAnno` che rappresentano i 12 mesi, da gennaio a dicembre.

- Il ciclo *for-each* nel codice precedente scandisce tale array (il cui riferimento è stato memorizzato nella variabile `mese`). Avremmo potuto evitare di dichiarare la variabile `mesi`, scrivendo più sinteticamente all'intero del metodo `main` esclusivamente il ciclo, come segue:

  ```
  for (MeseDellAnno m: MeseDellAnno.values())
    System.out.println(m + " ha " + m.numeroGiorni() + " giorni");
  ```

- Segue una soluzione alternativa in cui al posto del ciclo *for-each* si utilizza un poco elegante ciclo `for`:

  ```
  import prog.utili.MeseDellAnno;

  class ElencaMesi {
    public static void main(String[] a) {
      MeseDellAnno[] mesi = MeseDellAnno.values();
      for (int i = 0; i < mesi.length; i++)
        System.out.println(mesi[i] + " ha " + mesi[i].numeroGiorni() + " giorni");
    }
  }
  ```

Esercizio 9.2
Modificate il codice precedente in modo che riceva in ingresso l'anno e stampi l'elenco dei mesi dell'anno considerato con i relativi giorni. L'unica differenza è che nel caso di febbraio dovrà essere stampato 29 se l'anno è bisestile.
Attenzione: non occorre introdurre alcuna selezione! La classe `MeseDellAnno` *offre altri metodi di nome* `numeroGiorni` *che permettono di risolvere facilmente il problema.*

Esempio. Costruzione di una tabella di grandezza definita dall'utente, riempita con numeri ottenuti mediante lanci di dato. Visualizzazione della tabella, insieme alla somma di ciascuna riga e di ciascuna colonna.

```
import prog.io.*;
import java.util.Random;

class TabellaDadi {
  public static void main(String[] a) {

    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    Random dado = new Random();  //generatore

    int nRighe = in.readInt("Quante righe? ");
    int nColonne = in.readInt("Quante colonne? ");

    int[][] tabella = new int[nRighe][nColonne];

    // riempie la matrice per righe
    for (int i = 0; i < nRighe; i++)
      for (int j = 0; j < nColonne; j++)
        tabella[i][j] = dado.nextInt(6) + 1;  //numero casuale tra 1 e 6

    // visualizza il contenuto
    out.println("E' stata generata la seguente tabella: ");
    for (int i = 0; i < nRighe; i++) {
      for (int j = 0; j < nColonne; j++)
        out.print(tabella[i][j] + " ");
      out.println();
    }

    // visualizza la somma di ciascuna riga
    for (int i = 0; i < nRighe; i++) {
      int somma = 0;
      for (int j = 0; j < nColonne; j++)
        somma += tabella[i][j];
      out.println("Somma riga " + (i + 1) + ": " + somma);
    }

    // visualizza la somma di ciascuna colonna
    for (int j = 0; j < nColonne; j++) {
      int somma = 0;
      for (int i = 0; i < nRighe; i++)
```

```
        somma += tabella[i][j];
      out.println("Somma colonna " + (j + 1) + ": " + somma);
    }
  }
}
```

Note

- Ecco un esempio di esecuzione:

  ```
  Quante righe? 3
  Quante colonne? 5
  E' stata generata la seguente tabella:
  4 4 2 5 6
  5 1 2 1 3
  1 5 2 5 6
  Somma riga 1: 21
  Somma riga 2: 12
  Somma riga 3: 19
  Somma colonna 1: 10
  Somma colonna 2: 10
  Somma colonna 3: 6
  Somma colonna 4: 11
  Somma colonna 5: 15
  ```

  Si osservi che nei messaggi le righe e le colonne sono numerate a partire da 1 e non da 0, in quanto un utente comune normalmente conta da 1 e non da 0. A tale scopo nei messaggi di visualizzazione il numero di riga o colonna è aumentato di 1. Cosa succede se si tolgono le parentesi intorno a `i + 1` e `j + 1`?

- Si confrontino le ultime due porzioni di codice: quella la visualizzazione della somma di ciascuna riga e quella per la visualizzazione della somma di ciascuna colonna. Ciascuna di queste parti utilizza due cicli innestati. Nel caso della somma per righe, il ciclo più esterno esamina ciascuna riga e quello più interno ciascun elemento nella riga. Nel caso della somma per colonne, invece, il ciclo più esterno esamina ciascuna colonna e quello più interno ciascun elemento della colonna.

- Consultate la documentazione della classe `Random` per il contratto del metodo `nextInt` utilizzato per simulare il lancio del dado.

Esercizio 9.3
Aggiungete al metodo `main` precedente del codice per visualizzare l'elemento minimo di ciascuna riga e di ciascuna colonna.

Esercizio 9.4
Aggiungete al metodo `main` precedente del codice per visualizzare la media (anche con cifre decimali) di *tutti* gli elementi nella tabella.

Esercizio 9.5
Aggiungete al metodo `main` precedente del codice per visualizzare il numero di caselle della tabella che contengono 1, il numero di quelle che contengono 2, ecc. Può essere utile costruire un array di 6 contatori interi, per memorizzare quante volte ciascun valore appare nella tabella.