Altri esercizi su **array bidimensionali**

array bidimensionali: un caso di array di array

Esempio: **Tavola Pitagorica**[1]

```
public class Pitagora {
   private int[][] tabella;

   public Pitagora(int n) { // costruttore parametrico
      tabella = new int[n][n];                    // matrice quadrata: righe = colonne
      for (int i=0; i<tabella.length; i++){       // anche i<n
         for (int j=0; j<tabella[i].length; j++){ // anche j<n
            tabella[i][j] = (i+1)*(j+1);
         }
      }
   }

   public void stampa() {
      for (int i=0; i<tabella.length; i++) {
         for (int j=0; j<tabella[i].length; j++) {   // per matrice (righe con ugual
                                                     // numero di elementi)
                                                     // anche  tabella[0].length
            System.out.print(tabella[i][j] + "\t");
         }
         System.out.println();
      }
   }
}

public class ProvaPitagora {                  // applicazione di test
   public static void main(String[] args) {
      Pitagora p = new Pitagora(10);
      System.out.println("Tavola Pitagorica");
      p.stampa();
   }
}
```

```
General Output
--------------------Configuration: Matrice
Tavola Pitagorica
1   2   3   4   5   6   7   8   9   10
2   4   6   8   10  12  14  16  18  20
3   6   9   12  15  18  21  24  27  30
4   8   12  16  20  24  28  32  36  40
5   10  15  20  25  30  35  40  45  50
6   12  18  24  30  36  42  48  54  60
7   14  21  28  35  42  49  56  63  70
8   16  24  32  40  48  56  64  72  80
9   18  27  36  45  54  63  72  81  90
10  20  30  40  50  60  70  80  90  100
```

1 da http://informatica.abaluth.com/programmazione/java/java-array-a-due-dimensioni-matrici/

Esempio: ricerca di occorrenze in un array bidimensionale[2]

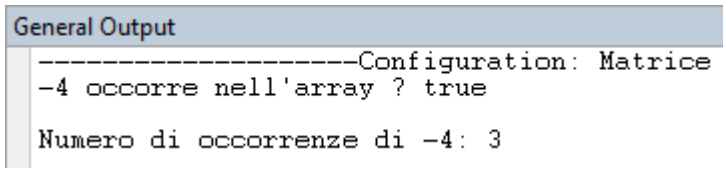
```
class Occorrenze {

// metodo che, dati un array bidimensionale di interi a ed un intero n, restituisce
//  true se n compare in a, false altrimenti

/**
 *   @param a array bidimensionale
 *   @param n numero da ricercare nell'array
 *   @return valore booleano
 */
public  boolean occorreBi (int[][] a, int n) {
        int i = 0;
        while (i < a.length) {
                int j = 0;
                while (j < a[i].length) {
                        if (a[i][j] == n)
                                return true;
                        j++;
                }
                i++;
        }
        return false;
}
// metodo che, dati un array bidimensionale di interi a ed un intero n, restituisce
// il numero delle occorrenze di n in a

/**
 *   @param a arrray bidimensionale
 *   @param n numero da ricercare nell'array
 *   @return cont - numero di occorrenze
 */
public  int occorrenzeBi (int[][] a, int n) {
        int cont = 0;
        for (int i=0; i<a.length; i++) {
                for (int j=0; j<a[i].length; j++) {
                        if (a[i][j] == n) {
                                cont++;
                        }
                }
        }
        return cont;
}
public static void main (String[] args) {
        int[][] a = {{3,-5,7,-4},{7,-4,-2},{8,7},{11,3,-4}};  // righe con numero di elementi diversi
        Occorrenze o = new Occorrenze();
        int n = -4;
        System.out.println(n + " occorre nell'array ? " +
        o.occorreBi(a,n));
        System.out.println("\nNumero di occorrenze di "+ n + ": " + o.occorrenzeBi(a,n));
}

} // fine applicazione
```



2 da http://www.di.univaq.it/monica/Lab-Progr/AA-07-08/lez07nov8.pdf