

```
/**
ArrayOrdinato: raccolta di metodi statici per un array ORDINATO:
**** e' aggiunto FONDI ****
*/

public class Fusione

{ public static boolean èOrdinato(int[] a)
  { int n = a.length;
    int i = 1;
    while(i < n && a[i-1] <= a[i]) i++;
    return i==n;
  }

/**
ricerca sequenziale in array ORDINATO
restituente un booleano,
versione con il while
*/
public static boolean ricercaWh(int x, int[] a)
{ int n = a.length;
  int i = 0;
  while(i < n && x > a[i]) i++;
  return i < n && x == a[i];
}

/**
ricerca sequenziale in array ORDINATO
restituente un booleano,
versione con il while ottimizzata
*/
public static boolean ricercaWhOtt(int x, int[] a)
{ int n = a.length;
  int i = 0;
  while(i < n-1 && x > a[i]) i++;
  return x == a[i];
}

/**
ricerca sequenziale in array ORDINATO
restituente l'indice dell'elemento
oppure -1 se il valore cercato non c'è,
versione con il while
*/
public static int ricercaIndiceWh(int x, int[] a)
{ int n = a.length;
  int i = 0;
  while(i < n && x > a[i]) i++;
  //if(i < n && x == a[i]) return i;
  //else return -1;
  return i < n && x == a[i] ? i : -1;
}

/** ricerca binaria, restituente un booleano */
public static boolean ricercaBin(int x, int[] a)
{ int i, inf = 0, sup = a.length - 1;
  while(inf <= sup) {
    i = (inf + sup)/2;
    if(x < a[i]) sup = i-1;
    else if(x > a[i]) inf = i+1;
```

```

        else return true;
    }
    return false;
}

/** ricerca binaria, restituisce l'indice */
public static int binRicercaIndiceDi(int x, int[] a)
{
    int i, inf = 0, sup = a.length - 1;
    while(inf <= sup) {
        i = (inf + sup)/2;
        if(x < a[i]) sup = i-1;
        else if(x > a[i]) inf = i+1;
        else return i;
    }
    return -1;
}

/** metodo FONDI con parametri due array ordinati ciascuno con
    eventuali ripetizioni, e totalmente riempiti;
    restituisce un array ordinato, totalmente riempito,
    con eventuali elementi ripetuti, dato dalla FUSIONE
    ordinata dei due array-argomenti che non sono modificati
    */

public static int[] fondi(int[] a, int[] b)
{
    int m = a.length;
    int n = b.length;
    int[] c = new int[m+n];

    int i = 0, j = 0;
    while(i < m && j < n)
    {
        if(a[i] <= b[j])
        {
            c[i+j] = a[i];
            i++;
        }
        else
        {
            c[i+j] = b[j];
            j++;
        }
    }
    while(i < m)
    {
        c[i+j] = a[i];
        i++;
    }
    while(j < n)
    {
        c[i+j] = b[j];
        j++;
    }
    return c;
}

static final int N = 10000; // lunghezza dell'array da ordinare
static final int K = 3*N; // N/3; // numero di valori diversi possibili

public static void main(String args[])
{
    int[] ar1 = {3,5,21, 30,30, 65, 88, 90, 99};
    int[] ar2 = {-5, -2, 4, 11, 13, 21, 25, 30, 89, 90, 94, 97};

    int[] fuso = fondi(ar1,ar2);
}

```

```
        System.out.println("ricerca");  
        System.out.println(fuso[ricercaIndiceWh(89, fuso)]);  
        System.out.println(fuso[binRicercaIndiceDi(89, fuso)]);  
    }  
}
```