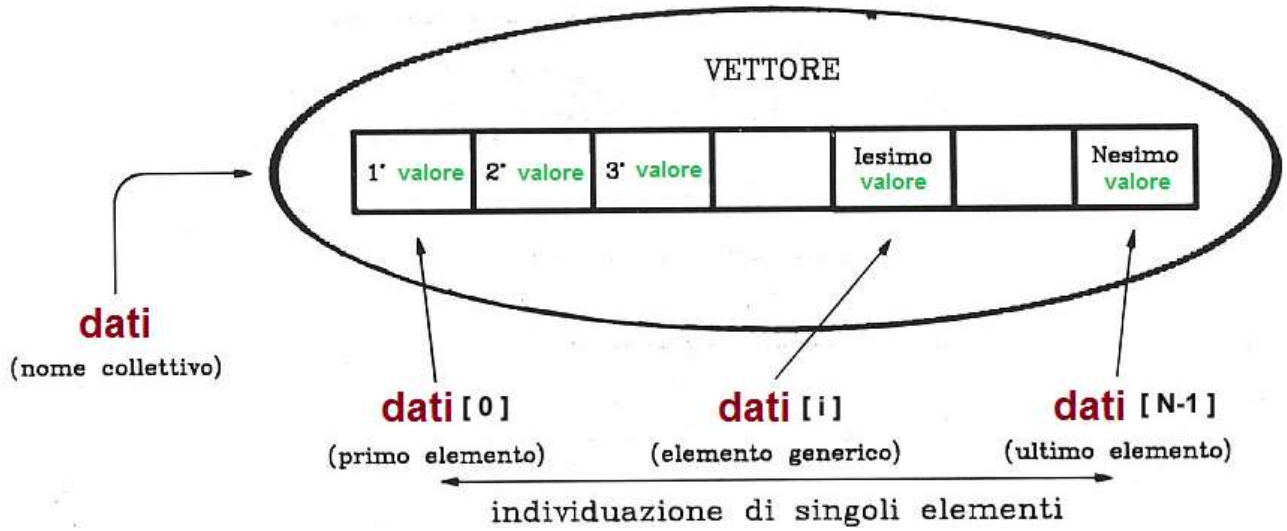
***Tipi aggregati: esercizi***

Si sono memorizzati temporaneamente in RAM valori relativi a misure acquisite durante un'esperienza di fisica. È noto il loro numero (**N**) e si è scelto come nome collettivo **dati**




**ESERCIZIO 1**:

Crea un'applicazione **Somma1.java** con lo scopo seguente:

INIZIALIZZATO il vettore ***da programma*** con valori di tipo double
Calcola la somma
Stampa la somma
Visualizza i valori, scandendo il vettore

**soluzione**

**ESERCIZIO 2**:

Crea un'applicazione **Somma2.java** con lo scopo seguente:

INIZIALIZZATO il vettore con valori di tipo double usando ***lettura da tastiera con controllo degli input***
Calcola la somma
Stampa la somma
Visualizza i valori, scandendo il vettore

**soluzione**

**ESERCIZIO 3**:

Crea un'applicazione **Cerca.java** con lo scopo seguente:

INIZIALIZZATO il vettore ***da programma*** con valori di tipo double
Letto un valore di tipo double usando ***lettura da tastiera con controllo degli input***
Cerca se il valore letto è presente nel vettore
Stampa il risultato della ricerca
Visualizza i valori, scandendo il vettore

*Nb: Con "Stampa il risultato della ricerca" si può intendere sia un messaggio con il valore trovato sia il numero di indice dell'elemento trovato.*

**per un confronto tra soluzioni**

**ESERCIZIO 4**:

Crea un'applicazione **Menu.java** con lo scopo seguente:

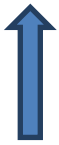
INIZIALIZZATO il vettore con valori di tipo double

CALCOLA con possibile ***scelta***:

1. la MEDIA ARITMETICA
2. il VALORE MASSIMO
3. il VALORE MINIMO

Proponi una soluzione modulare, usando i tre metodi sotto-illustrati dove i *parametri passati sono il vettore ed il numero dell'opzione scelta* (il cui valore si potrà stampare come prima informazione all'utente)

| scopo | Firma o Segnatura | Tipo ritorno |
|---|---|---|
| Trovare il massimo | massimo(double[], int) | double |
| Trovare il minimo | minimo (double[], int) | double |
| Calcolare la media | media (double[], int) | double |



*Nb: con **firma** o **segnatura di un metodo** nella OOP si intende*

*Il **nome** del metodo*
*ed il **tipo, numero ed ordine** dei parametri formali*

*il tipo di ritorno non è nella segnatura (wikipedia)*

Ricorda che esiste in tutti i linguaggi C_Like il costrutto ***switch – case***

*(pg.6 del manualetto)*

con sintassi:

| | |
|---|---|
| ***switch (espressione) {***<br>***case costante : istruzioni***<br>***.....***<br>***default:***<br>***}*** | ***switch (ch) {***<br>***case 1 :*** *<chiamata a metodo>*<br>***break;***<br>***.....***<br>***}*** |

che permette in modo efficiente di implementare una *sequenza di if*, dove la condizione da valutare per verificare se è vera è una ***espressione*** costante (ad esempio un valore intero letto da tastiera)

**soluzione**

**ESERCIZIO 5**:

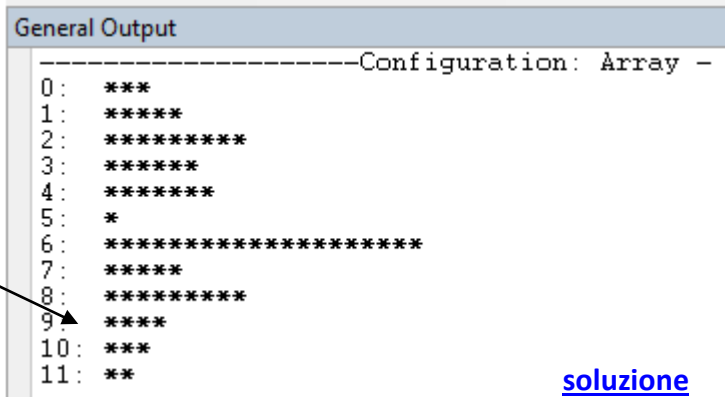
Crea un'applicazione **Istogramma.java** con lo scopo seguente:

a) disegnare l'istogramma di un array di interi: per ogni numero dell'array di nome **frequenze** viene stampata una riga di asterischi la cui lunghezza è pari al numero in esame.

b) Proporre una variante dell'applicazione **IstogrammaConMetodi.java** in cui si definisce un metodo **stampaRiga()** che ha come parametri un vettore **a** di interi ed un intero **r** (indice di un elemento) e stampa una riga di asterischi il cui numero è pari ad **a[r]**.

In entrambi i casi si desidera l'**output** mostrato in figura:

*Nb: con attenzione alla diversa formattazione nel caso di punti dell'elenco espressi come unità o decine*

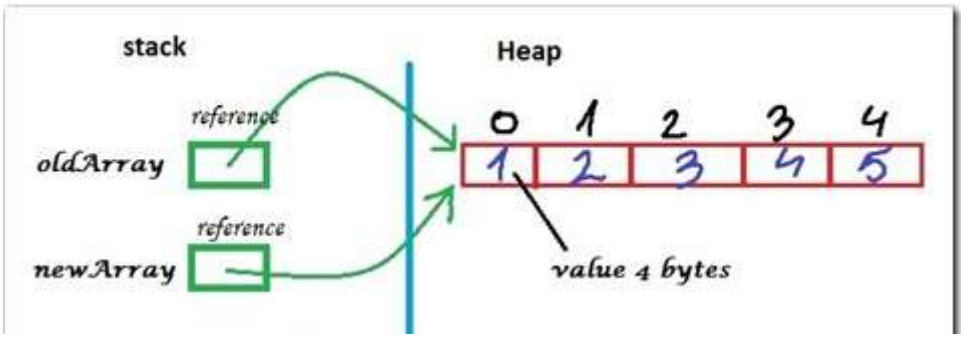



[soluzione]

**ESERCIZIO 6**[1]:

Verificare, stampando a monitor, che il nome di un vettore è un indirizzo e verificare che l'istruzione evidenziata nel seguente segmento di programma non copia elementi ma produce l'effetto sotto-illustrato; verificare infine, creando un altro vettore di interi con dimensione letta da tastiera, che tutti gli elementi sono inizializzati a zero

```
//initial array
int[] oldArray = {1,2,3,4,5};
//new array
int[] newArray;

//shallow copy - copy reference values
newArray = oldArray;

//print a value using second reference
//output: prints value 3
System.out.println(newArray[2]);
```




[soluzione]

[1] *Da* http://www.itcsolutions.eu/2011/01/03/tutorial-java-7-reference-data-type-variables/

**ESERCIZIO 7**:

In una gara il punteggio di ciascun atleta è dato dal pubblico.

I voti possono andare da 1 a 10.

Scrivere un'applicazione che per ogni atleta rilevi il numero di occorrenze dei vari voti.

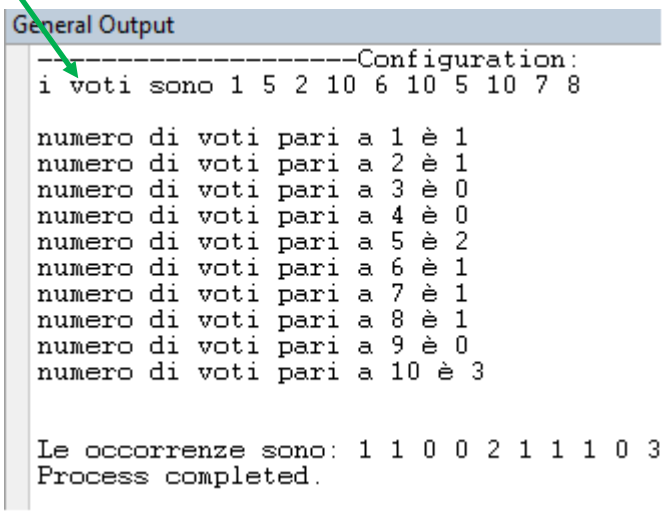
***Suggerimento:***

*Si intenda: scrivere un'applicazione che per un dato atleta conti il numero di occorrenze dei vari voti.*

*Si ipotizzi che il pubblico sia composto da 10 persone e si memorizzino i voti (quindi 10 voti) in un vettore di interi.*

*Per semplicità si inizializzi da programma tale vettore.*

**Si vuole un output come in figura:**




Il numero dei voti, uguale al numero delle persone,
è la dimensione del vettore di nome **voti**

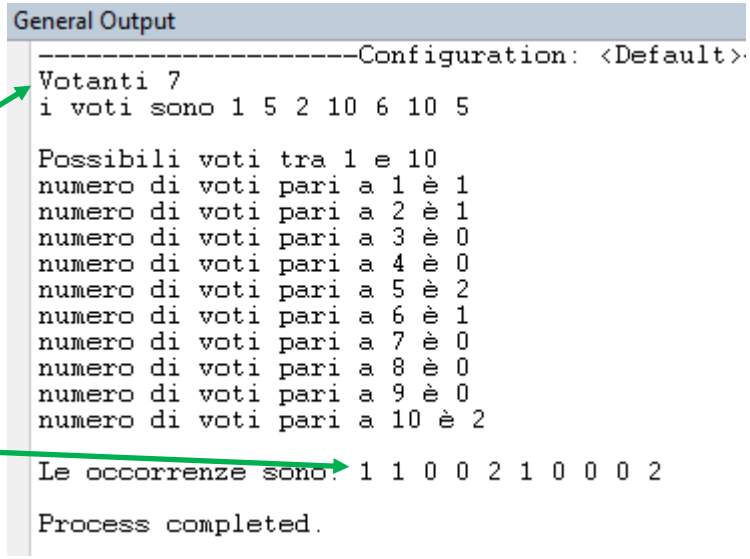
E si pensa di usare un secondo vettore di nome **occ**, che conterrà le occorrenze di ogni voto.

**soluzione**

Si verifichi
la correttezza di funzionamento
anche nel caso di un numero di persone
diverso dal numero massimo dei voti:

***nb**: la dimensione del vettore* **voti**
*cambia (in base al numero di votanti)*

*ma la dimensione del vettore*
*che contiene le occorrenze di ogni voto è*
*sempre pari a 10 (infatti i voti sono*
*sempre nell'intervallo 1-10)*



**ESERCIZIO 7B**: (miglioramenti successivi: *intervallo dei voti variabile, controlli)*

In una gara il punteggio di ciascun atleta è dato dal pubblico (*un numero di persone noto*).

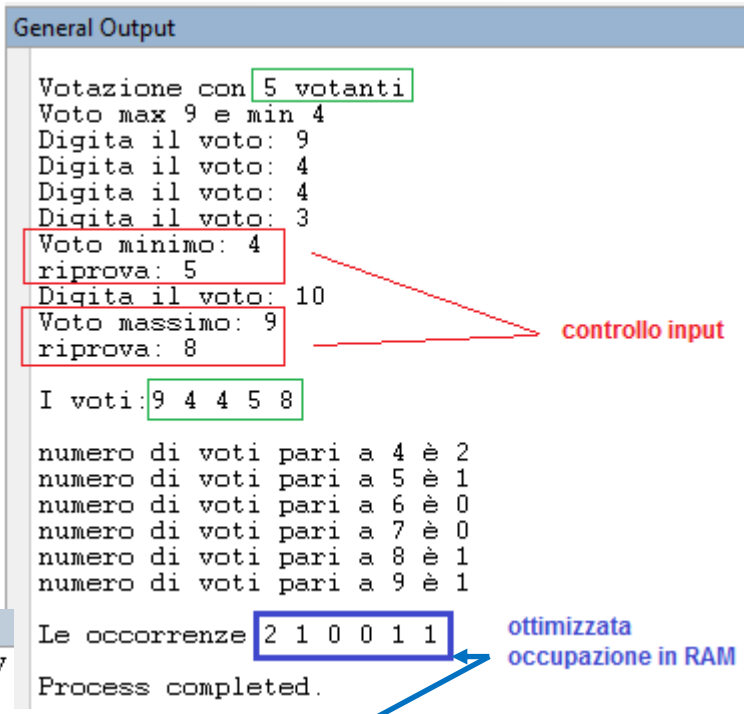
I voti possono andare da un *voto minimo* (maggiore di 0) ad un *voto massimo*.

Scrivere un'applicazione che, per un dato atleta, rilevi il numero di occorrenze dei vari voti.

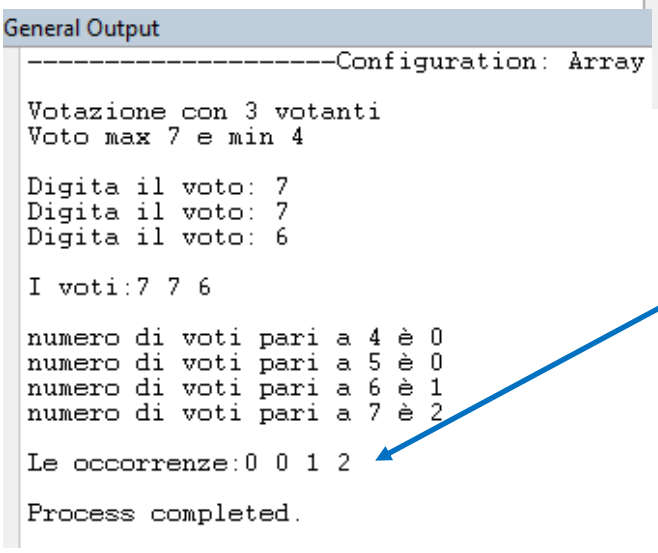
Si preveda ***controllo degli input*** nell' ipotesi di lettura da tastiera di tali voti.

Testare la proposta risolutiva per più casi, variando il numero di votanti, l'intervallo dei voti e prevedere controllo sull'involontaria inversione tra valori massimi e minimi dei voti.

***Effetto desiderato*** *nel caso di* ***5 persone****, voto* ***minimo 4*** *e* ***massimo 9****:*




***Effetto desiderato*** *nel caso di* ***3 persone****, voto* ***minimo 4*** *e* ***massimo 7****:*




**soluzione**

**ESERCIZIO 7C**: (miglioramenti successivi: *uso di array di oggetti*)

In una gara il punteggio di ciascun atleta è dato dal pubblico (*un numero di persone noto*). I voti possono andare da un *voto minimo* (maggiore di 0) ad un *voto massimo*.

Scrivere un'applicazione che usi oggetti di tipo Atleta (almeno 2) - caratterizzati almeno da un numero di gara - ed oggetti di tipo Votante – caratterizzati almeno dalla capacità di votare. Tale applicazione deve rilevare, per ogni atleta, il numero di occorrenze dei vari voti.

**Possibile soluzione dell'Esercizio 1** *(paradigma tradizionale):*

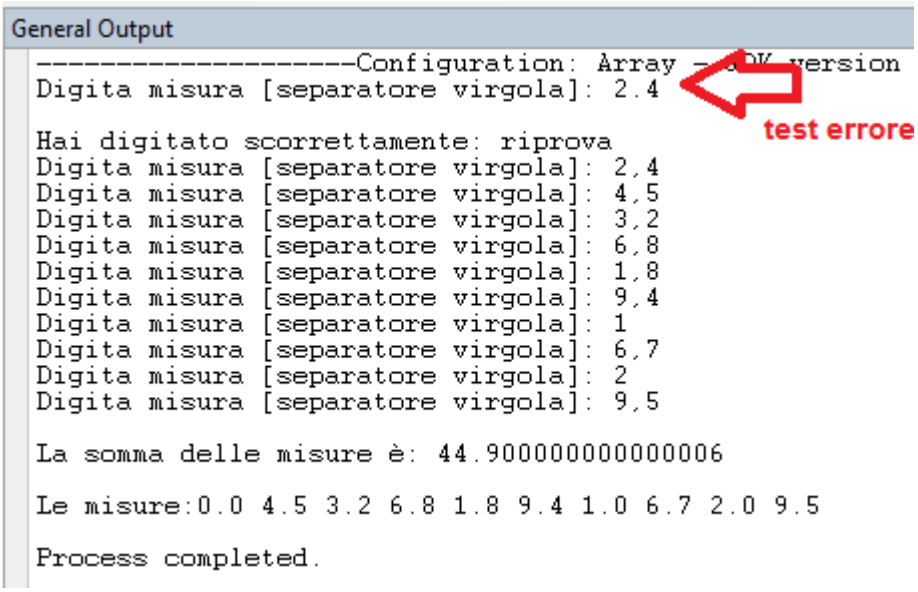
```
/**
 *
 * Somma1 application
 *
 * @author 3AI
 * @version 1.00 2017/3/17
 */

public class Somma1 {

   public static void main (String[]args) {

            double dati[] = {1.2, 4.5, 6.8, 7.9};   // inizializzazione da programma
                                                    // contemporanea alla definizione del vettore
            double somma = 0.0;
            for(int i=0; i<dati.length; i++)        // calcolo della somma, scandendo gli elementi
                  somma=somma+dati[i];

            System.out.println("\nLa somma delle misure è: " + somma);  // stampa della somma

            System.out.print("\nLe misure:");
            for(int i=0; i<dati.length; i++)              // visualizza i valori, scandendo gli elementi
                 System.out.print(dati[i] + " ");
            System.out.print("\n");
   }

} // fine applicazione
```

```
General Output
--------------------Configuration: <Default>---

La somma delle misure è: 20.4

Le misure:1.2 4.5 6.8 7.9

Process completed.
```

**Possibile soluzione orientata agli oggetti dell'Esercizio 2** :

```
/**
 * Somma2 application
 *
 * @author 3AI
 * @version 1.00 2017/3/20
 */
 import java.util.Scanner;

public class Somma2 {

   private double dati[];
                                                                    // metodi di accesso al vettore
   public double[] getDati(){
          return dati;                    // ritorna il reference cioè l'indirizzo
   }
   public void setDati(double[] v){    // ora sia dati che v puntano alla stessa area di RAM
          dati = v;                        // cioè dati[0] e v[0]  sono la stessa variabile ecc
   }
   public Somma2(){                 // costruttore di default
          final int N = 1;              // almeno un elemento … se N=0 non si accede a nessun elemento
          dati = new double [N];
  }
  public Somma2(int n){           // costruttore parametrico
           dati = new double [n];
  }
  public double leggiD (){  // lettura da tastiera con controllo degli input

          double d = 0.0;
          Scanner in = new Scanner (System.in);
          System.out.print("Digita misura [separatore virgola]: ");
          try {
                    d = in.nextDouble();
          }catch(Exception e){
                              System.out.println("\nHai digitato scorrettamente: riprova");
                              leggiD();
                          }
           return d;
   }
   public void inizializza(){                    // inizializzazione
            for(int i=0; i<dati.length; i++)
                    dati[i] = leggiD();
   }
   public double somma(){
             double somma = 0.0;
             for(int i=0; i<dati.length; i++)        // calcolo della somma, scandendo gli elementi
                    somma=somma+dati[i];
              return somma;
   }
   public void visualizza(){
             System.out.print("\nLe misure:");
             for(int i=0; i<dati.length; i++)              // visualizza i valori, scandendo gli elementi
                    System.out.print(dati[i] + " ");
             System.out.print("\n");
   }
```

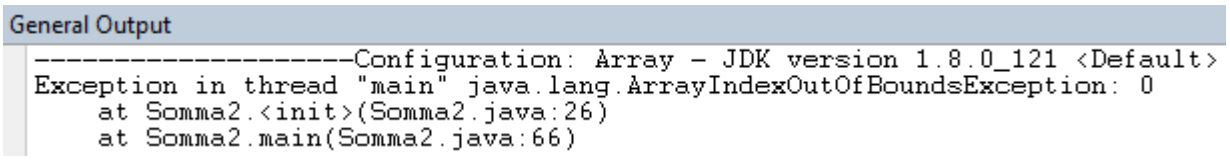
```
   public static void main (String[]args) {

         final int N = 10;
         Somma2 ogg = new Somma2 (N);  // se N elementi … ad esempio 10
         ogg. inizializza();
          double somma = ogg.somma();
          System.out.println("\nLa somma delle misure è: " + somma);  // stampa della somma
          ogg.visualizza();
   }
} // fine applicazione
```




*Nb:* ricordando la motivazione all'uso di array, cioè organizzare più variabili dello stesso tipo in un'unica struttura con nome collettivo ed evitare di inventare diversi nomi identificatori per Individuarle, si penserà tale insieme costituito da più elementi, sicuramente ***almeno uno …***

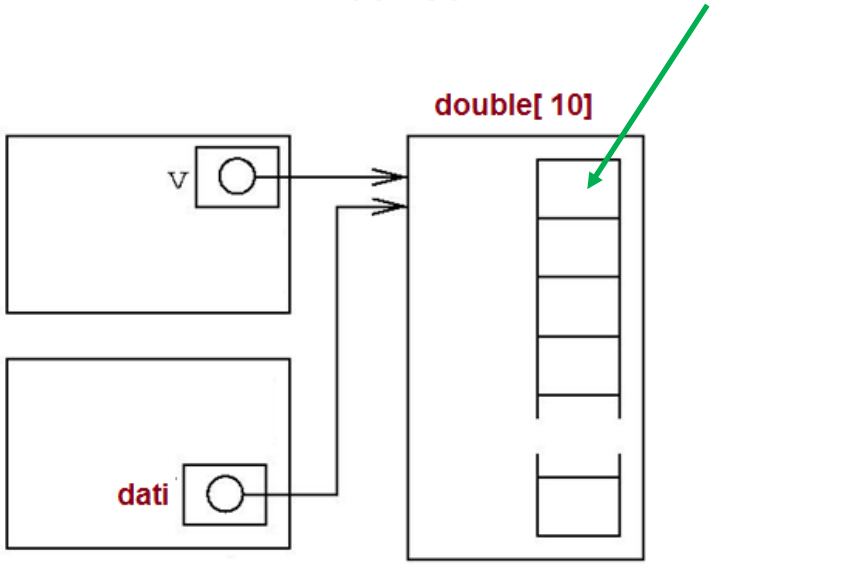
Se la dimensione è nulla ***non si accede a nessun elemento*** ed il tentativo di accesso causa eccezione di tipo **ArrayIndexOutOfBoundsException**

```
public Somma2(){      // costruttore di default
    final int N = 0;                  // possibile 0 ma non si accede a nessun elemento
    dati = new double [N];
    System.out.println("dato " + dati[0]);   // se N = 0
                                             // eccezione ArrayIndexOutOfBoundsException: 0
 }
```



**Metodi di accesso al vettore:**

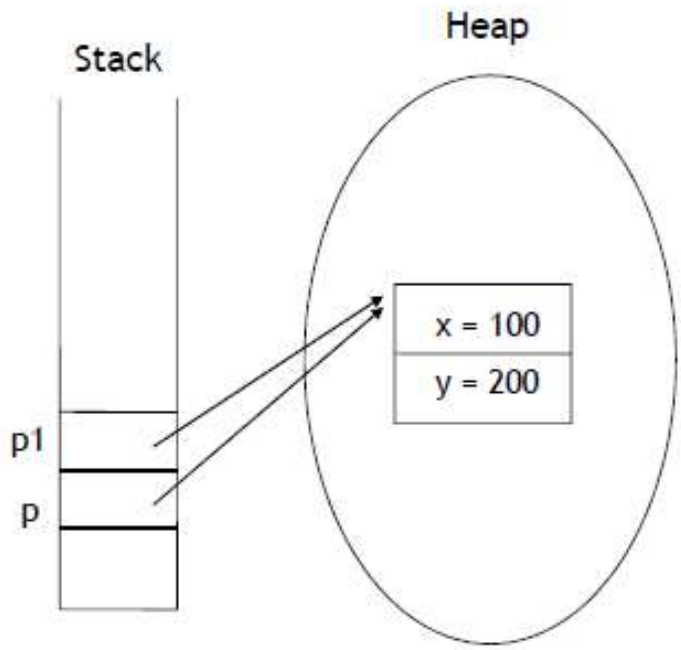
```
public double[] getDati(){
        return dati;                    // ritorna il reference cioè l'indirizzo
}

public void setDati(double[] v){        // ora sia dati che v puntano alla stessa area di RAM
        dati = v;                       // cioè dati[0] e v[0]  sono la stessa variabile ecc
}
```




***In generale per qualsiasi oggetto:***

- ***il nome è l'indirizzo di RAM dove è allocato cioè il valore del reference***
- ***l'istruzione nomeOggetto1 = nomeOggetto significa che entrambi i nomi puntano alla stessa area di RAM***

```
public class Point {
    private int x;
    private int y;

    public Point(int dx, int dy) {
            x = dx;
            y = dy;
    }
}
public class Test {
    public static void main(String[] args) {
            Point p = new Point( 100, 200 );
            Point p1;
            p1 = p;
    }
}
```

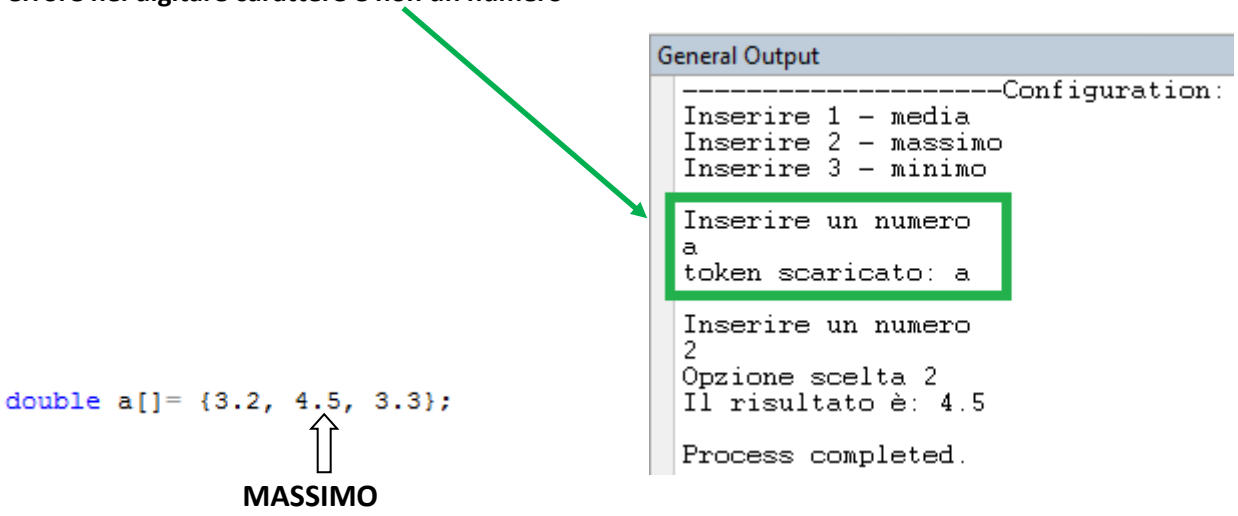


**Possibile soluzione dell'Esercizio 4**

```
import java.util.*;
public class MenuC{

        private int n;
        public int leggi(){

                Scanner in = new Scanner(System.in);
                System.out.println("\nInserire un numero");
                try{
                        n = in.nextInt();
                } catch(InputMismatchException e){
                                                        System.out.println ("token scaricato: " + in.next ());
                                                        // pulizia del buffer anche  in.nextLine();
                        leggi();
                }
                return n;
        }
        static double massimo(double a[], int n){
                System.out.println("Opzione scelta " + n);
                double m = 0.0;      // scelta valore iniziale temporaneo non valida se tutti valori negativi ….
                for (int i=0;i<a.length;i++){
                        if (a[i] > m){
                                m = a[i];
                }
                }
                return m;
        }
        static double minimo(double a[], int n){
                System.out.println("Opzione scelta " + n);
                double m = a[0];                                // …. scelta valore iniziale temporaneo migliore
                for (int i=1;i<a.length;i++){
                        if (a[i] < m){
                                m = a[i];
                }
                }
                return m;
        }
        static double media(double a[], int n){
                System.out.println("Opzione scelta " + n);
                double media = 0.0;
                double somma = 0.0;
                for (int i=0;i<a.length;i++){
                        somma=(somma+a[i]);
                }
                media=(somma/a.length);
                return media;
        }
        public static void main (String[]args){
                MenuC o = new MenuC ();
                double a[]= {3.2, 4.5, 3.3};
                System.out.println("Inserire 1 - media");
                System.out.println("Inserire 2 - massimo");
                System.out.println("Inserire 3 - minimo");
```

```
                double ris =0.0;

                int m=o.leggi();  // metodo richiesto all'oggetto come servizio

                switch(m){  // chiamata diretta di metodi static

                        case 1: ris = media(a, m);
                                        break;
                           case 2: ris = massimo(a, m);
                                        break;
                            case 3: ris = minimo(a, m);
                                    break;
                          default: System.out.println("Opzione non prevista");
                                    System.exit(0);
                }
                System.out.println("Il risultato è: " + ris);
        }
}
```

**Effetto con errore nel digitare carattere e non un numero**



**Possibile soluzione dell'Esercizio 5** *(paradigma tradizionale):*

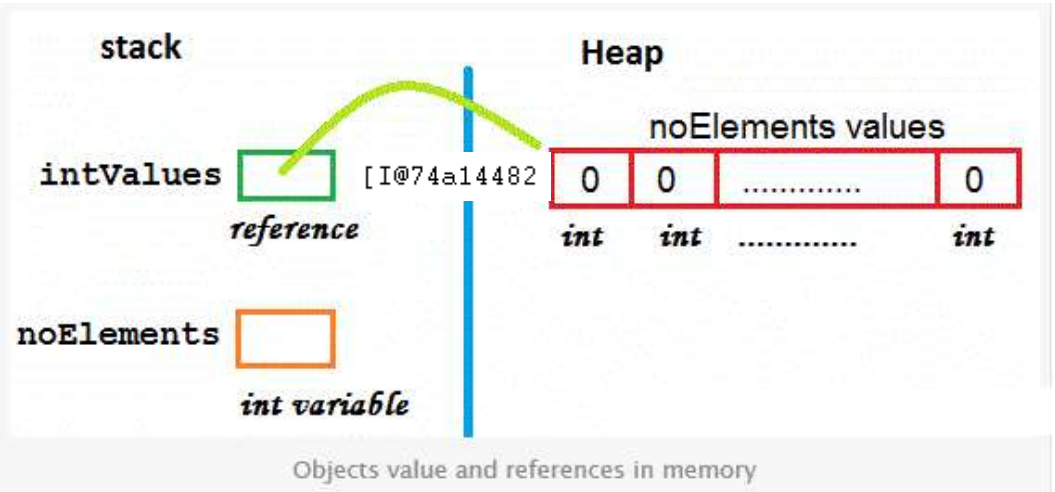
```
/**
 *
 * Istogramma application
 *
 * @author 3AI
 * @version 1.00 2017/3/20
 */
class Istogramma {
  public static void main (String[] args) {
    int i, j;
    int[] frequenze = {3, 5, 9, 6, 7, 1, 20, 5, 9, 4, 3, 2};   // inferiori a 100

    for (i = 0; i < frequenze.length; i++) {
            if (i<=9)
              System.out.print(i+": \t");   // unità nell'elenco
       else
            System.out.print(i+":\t");       // decine nell'elenco
       for (j = 1; j <= frequenze[i]; j++)
                  System.out.print('*');
      System.out.println();
    }
  }
}
```

**Possibile soluzione modulare dell'Esercizio 5:**

```
/**
 *
 * IstogrammaConMetodi application
 *
 * @author 3AI
 * @version 1.00 2017/3/20
 */
public class IstogrammaConMetodi {

  static void stampaRiga(int[] a, int r) { // stampa a[r] asterischi

    if (r<=9)
              System.out.print(r+": \t");   // unita nell'elenco
       else
            System.out.print(r+":\t");       // decine nell'elenco

    for (int j = 1; j <= a[r]; j++)
       System.out.print('*');
    System.out.println();
  }

  public static void main (String[] args) {
    int i;
    int[] frequenze = {3, 5, 9, 6, 7, 1, 20, 5, 9, 4, 3, 2};

    for (i = 0; i < frequenze.length; i++)
      stampaRiga(frequenze, i);
  }
}
```

**Possibile soluzione modulare dell'Esercizio 6:**

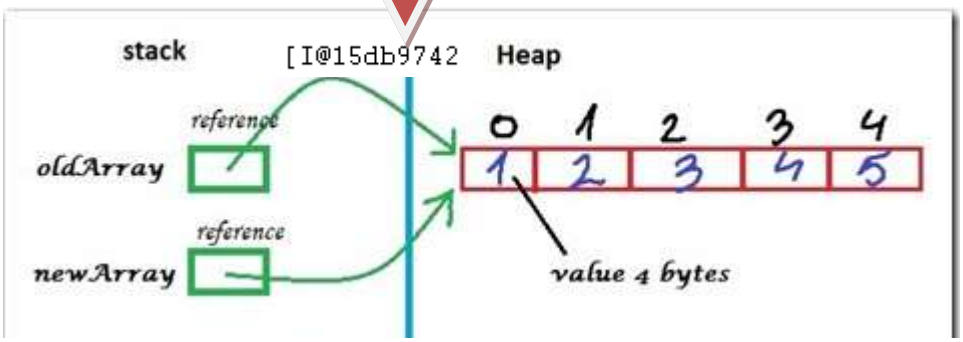
```
/**
 * Reference application
 *
 * @author 3AI
 * @version 1.00 2017/3/20
 */
import java.util.Scanner;

public class Reference {

        public static void print(int [] v){
                for(int i=0; i <v.length; i++)
                 System.out.print(v[i] + " ");
        }

        public static void main(String [] args){

                try {
                        int[] oldArray = {1,2,3,4,5};           //vettore iniziale
                        print(oldArray);   //stampa i valori
                        int[] newArray;  //nuovo vettore

                        newArray = oldArray;              //copia i valori dei riferimenti

                                                //stampa un valore - quello con indice 2 - usando il secondo riferimento
                        System.out.println("\nil terzo dato è: " + newArray[2]);
                                                //stampa i nomi dei vettori  cioè gli indirizzi o references
                        System.out.println("\nIndirizzo vecchio "+oldArray);
                        System.out.println("\nIndirizzo nuovo "+newArray);

                        int[] intValues;                  // altro vettore
                        int noElements = 0;
                        System.out.println("\nDigita il numero di elementi del vettore(0 - 255):");
                        Scanner in = new Scanner (System.in);       //leggi il numero di elementi
                        noElements =in.nextInt(); // se negativo  stamperà null ... se zero non stampa nulla
                         intValues = new int[noElements];
                                                //stampa i valori - tutti pari a 0 cioè il valore di default
                        print(intValues);
                                                                //stampa il nome del vettore  cioè l'indirizzo o reference
                        System.out.println("\n\nIndirizzo "+intValues);
                         } catch(Exception ex){
                                                        System.out.println(ex.getMessage());
                                        }
        }
}
```



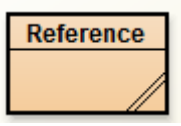
Objects value and references in memory

***Effetto***

```
General Output
----------Configuration: Array - JDK version
1 2 3 4 5
il terzo dato è: 3

Indirizzo vecchio [I@15db9742

Indirizzo nuovo [I@15db9742

Digita il numero di elementi del vettore(0 - 255):
4
0 0 0 0

Indirizzo [I@74a14482

Process completed.
```



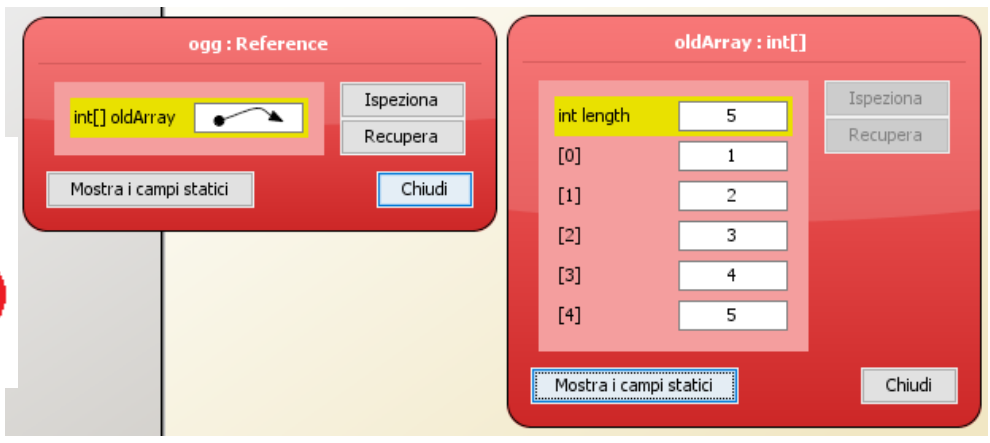

Creando in ambiente BlueJ, nell' ispezionare lo **stato** dell' oggetto di tipo Reference, si visualizzano valori e dimensione del vettore definito come attributo (*pur se la rappresentazione può risultare ambigua*)




```
import java.util.Scanner;

public class Reference {
        int[] oldArray = {1,2,3,4,5};   //vettore iniziale attributo pubblico della classe
        public void print(int [] v){
                for(int i=0; i <v.length; i++)
                        System.out.print(v[i] + " ");
        }
        public static void main(String [] args){
                Reference ogg = new Reference();
                ogg.print(ogg.oldArray);                    //stampa i valori
                int[] newArray;  //nuovo vettore
                 newArray = ogg.oldArray;            //copia i valori dei riferimenti
                                                    //stampa un valore - quello con indice 2 - usando il secondo riferimento
                System.out.println("\nil terzo dato è: " + newArray[2]);
                                                                    //stampa i nomi dei vettori  cioè gli indirizzi o references
                System.out.println("\nIndirizzo vecchio "+ogg.oldArray);
                System.out.println("\nIndirizzo nuovo "+newArray);
        }
}
```

***Effetto***

```
1 2 3 4 5
il terzo dato è: 3

Indirizzo vecchio [I@6662b071

Indirizzo nuovo [I@6662b071
```



**Possibile prima soluzione dell'Esercizio 7**: (*problema aperto - intervallo dei voti variabile)*

```
/**
 * Atleta application
                        applicazione che per un dato atleta conta il numero di occorrenze dei vari voti (da 1 a 10).
                        Si ipotizza che il pubblico sia composto da 10 persone (o 7 o 20)  e si memorizzano i voti in un vettore
                        Per semplicità si inizializza da programma tale vettore di interi di nome vett.
                        Si pensa di usare un secondo vettore, di nome occ, che conterrà le occorrenze di ogni voto
 * @author 3AI
 * @version 1.00 2017/3/24
 */
 public class Atleta{
     private int voti[] = {1,5,10,10,5,6,7,5,8,9};   //  voti, in numero pari alle persone (10  o 7 o …), con diverse occorrenze
                        //{1,5,10,10,5,6,7};        // se 7 persone …… se lettura da tastiera, necessario controllo degli input
     private final int DIM = 10;            // intervallo fisso dei voti da 1 a 10
     private int occ[] = new int [DIM]; // le occorrenze …. si memorizzeranno quanti 1, 2, 3, … 10
     public int[] getVoti(){                                                                    // metodi di accesso ai vettori
          return voti; // ritorna il reference cioè l'indirizzo
     }
     public void setVoti(int[] v){ // ora sia voti che v puntano alla stessa area di RAM
          voti = v;
     }
     public int[] getOcc(){
          return occ; // ritorna il reference cioè l'indirizzo
     }
     public void setOcc(int[] v){ // ora sia occ che v puntano alla stessa area di RAM
          occ = v;
     }
     public void calcolaOcc(){
          for(int i=0; i<DIM; i++) {                    // voto minimo 1  e  DIM ha il significato di voto massimo 10
                    occ[i] = conta(i+1);                                  ↓
                    System.out.println ("numero di voti pari a "+ (i+1) + " è " + occ[i]);   // occ[0] contatore dei voti 1 ecc…
          }
     }
     private int conta(int n){          // n voto tra  1 e 10
          int count = 0;
          for(int i=0; i<voti.length; i++)
                    if(voti[i] == n)
                     count = count + 1;
            return count;
     }
     public void visualizzaV(){
         System.out.print("\nI voti:");
              for(int i=0; i<voti.length; i++)              // visualizza i valori, scandendo gli elementi
                        System.out.print(voti[i] + " ");
              System.out.print("\n");
     }
     public void visualizzaO(){
        System.out.print("\nLe occorrenze:");
              for(int i=0; i<occ.length; i++)               // visualizza i valori, scandendo gli elementi
                        System.out.print(occ[i] + " ");
              System.out.print("\n");
     }
     public static void main (String[]args) {
              Atleta a = new Atleta();
              a.visualizzaV();
              System.out.print("\n");
              a.calcolaOcc();
              a.visualizzaO();
    }
} // fine applicazione
```

**Possibile soluzione dell'Esercizio 7B**: (*algoritmo più generale – intervallo dei voti variabile, controlli)*

```
/**
 * AtletaOOP application

    applicazione che per un dato atleta conta il numero di occorrenze dei vari voti  (da votoMin - maggiore di 0 - a votoMax).
    Si ipotizza che il pubblico sia composto da un numero di persone noto numVotanti  e si memorizzano i voti interi, letti da tastiera, in un vettore di nome vett.
    Si prevede controllo degli input (intervallo dei voti solo tra votoMin e votoMax) nell' inizializzare tale vettore.
    Si pensa di usare un secondo vettore, di nome occ, che conterrà le occorrenze di ogni voto  ottimizzandone la dimensione (senza sovradimensionarlo).

* @author 3AI
* @version 1.00 2017/3/27
*/
import java.util.Scanner;
 public class AtletaOOP{

     private int numVotanti;
     private int voti[];
     private int votoMin;
     private int votoMax;
     private int limite; // dimensione del vettore delle occorrenze non sovradimensionato
     private int occ[];  // le occorrenze … ogni elemento memorizza il conteggio di un dato voto
     private Scanner in;
     private int letto;

     public AtletaOOP(){
           numVotanti = 3;                  // per test veloce solo tre votanti
           votoMin = 1;
           votoMax =10;
            ini();      // per riuso codice ... da gestire con attenzione, evitando consenso a modifica attributi senza controllo (*)
           }
     public AtletaOOP(int nV, int min, int max){ // per semplicità: si propone solo costruttore parametrico opportuno        (*)
            numVotanti = nV;
            setRange(min, max);                    // prevedendo controllo nel settare il range
            ini();       // per riuso codice ...
     }
     private void setRange(int min, int max){          // metodo privato – chiamato solo da altro metodo della classe
            if (min <= max){
                        votoMax = max;          // controllo che il voto minimo sia davvero tale
                        votoMin = min;
            }
            else {      votoMax = min;          // scambio
                        votoMin = max;
            }
     }
     public void setAll(int nV, int min, int max){ // metodo di accesso per " scrittura" contemporanea del range voti
            numVotanti = nV;                        // e numero votanti                                                  (*)
            setRange(min, max);     // con controllo che il voto minimo sia davvero tale
            ini();
     }
      private void ini(){                                  // metodo privato – chiamato solo da altro metodo della classe
           voti = new int [numVotanti];
           limite = (votoMax - votoMin) + 1;
           occ = new int [limite];
            in = new Scanner (System.in);
     }
```

```
                                                                    // metodi di accesso " in lettura" utilizzati
public int getNumVotanti(){
        return numVotanti;
}
public int getVotoMax(){
        return votoMax ;
}
 public int getVotoMin(){
        return votoMin;
}
private int leggiI(){     // metodo privato – chiamato solo da altro metodo della classe
    try{
            letto = in.nextInt();
    }catch (Exception e){    System.out.print("Digita un intero: ");
                             leggiI();
              }
    if (letto < votoMin){                                  // controllo:  intervallo dei voti solo tra votoMin e votoMax
                    System.out.println("Voto minimo: " + votoMin);
                    System.out.print("riprova: " );
                    leggiI();
    }
    else if (letto > votoMax){
                       System.out.println("Voto massimo: " + votoMax);
                       System.out.print("riprova: " );
                       leggiI();
    }
    return letto;
}
public void inizializza(){
    for(int i=0; i<voti.length; i++){
            System.out.print("Digita il voto: ");
            voti[i] = leggiI();
    }
}
public void calcolaOcc(){   // algoritmo efficiente (evitando calcoli inutili)
     for(int i=0; i<limite; i++) {
            occ[i] = conta(i+votoMin);
            System.out.println ("numero di voti pari a "+ (i+votoMin) + " è " + occ[i]);
    }
}
private int conta(int n){   // metodo privato – chiamato solo da altri metodi della classe
                            // con passaggio di parametro  n voto tra votoMin e votoMax
     int count = 0;
     for(int i=0; i<voti.length; i++)
            if(voti[i] == n)
              count = count + 1;
       return count;
}
 public void visualizzaV(){
          System.out.print("\nI voti:");
              for(int i=0; i<voti.length; i++)       // visualizza i valori, scandendo gli elementi
                      System.out.print(voti[i] + " ");
              System.out.print("\n");
}
 public void visualizzaO(){
          System.out.print("\nLe occorrenze:");
              for(int i=0; i<occ.length; i++)          // visualizza i valori, scandendo gli elementi
                      System.out.print(occ[i] + " ");
              System.out.print("\n");
       }
```

```
public static void main (String[]args) {
    AtletaOOP a = new AtletaOOP();  //  3 votanti ed intervallo dei voti tra 1 e 10

    //a =new AtletaOOP(5,4,9);   // per test veloce .. in alternativa
    //a =new AtletaOOP(3,4,7);   // per test veloce .. in alternativa
    // a =new AtletaOOP(10,1,10);   // Esercizio 7   per test veloce .. in alternativa

    a.setAll(2,7,4);                                   // 2 votanti …. test errore min max → li scambia

    System.out.println("\nVotazione con " + a.getNumVotanti() + " votanti");
    System.out.println("Voto max " +a.getVotoMax() +" e min " + a.getVotoMin() );
    System.out.print("\n");
     a.inizializza();
    a.visualizzaV();
     System.out.print("\n");
    a.calcolaOcc();
    a.visualizzaO();
    }
} // fine applicazione
```

```
General Output
--------------------Configuration: Array

Votazione con 2 votanti
Voto max 7 e min 4

Digita il voto: 3
Voto minimo: 4
riprova: 5
Digita il voto: 8
Voto massimo: 7
riprova: 7

I voti:5 7

numero di voti pari a 4 è 0
numero di voti pari a 5 è 1
numero di voti pari a 6 è 0
numero di voti pari a 7 è 1

Le occorrenze:0 1 0 1
```

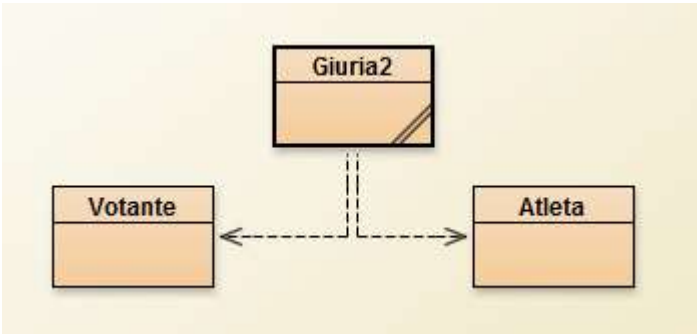
invertiti correttamente

controllo input

***Nb:***
***proposte risolutive da migliorare***
*specie nell'impostare analisi OOP*
*(progetto di classe Votante,*
*classe Atleta*
*ed applicazione che ne usa istanze)*
*e nell'inserire opportuni commenti*
*per documentazione con Javadoc*

**Possibile guida alla soluzione dell'Esercizio 7C**:

*Listing online*




**Constructor Summary**

| |
|---|
| Votante() |
| Votante(int min, int max) |

**Method Summary**

| | |
|---|---|
| void | decidiVoto() |
| int | getRange() |
| int | getVoto() |
| int | getVotoMax() |
| int | getVotoMin() |
| void | setRange(int min, int max) |

**Constructor Summary**

| |
|---|
| Giuria2() |

**Method Summary**

| | |
|---|---|
| void | calcolaOcc(Votante[] giuria, int j) |
| void | elabora(Atleta[] corridori, Votante[] giuria) |
| static void | main(java.lang.String[] args) |

**Constructor Summary**

| |
|---|
| Atleta() |
| Atleta(int n) |

**Method Summary**

| | |
|---|---|
| java.lang.String | getCognome() |
| int | getNum() |
| void | setCognome(java.lang.String s) |
| void | setNum(int n) |

***APPENDICE***

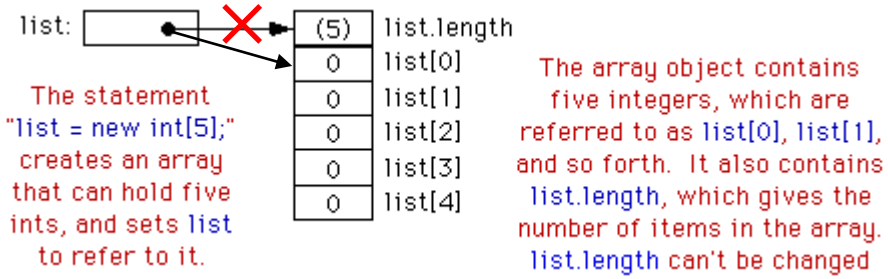
***Dispense della docente***:

**Array**: http://new345.altervista.org/Dispense/Array_java.pdf
con scheda introduttiva

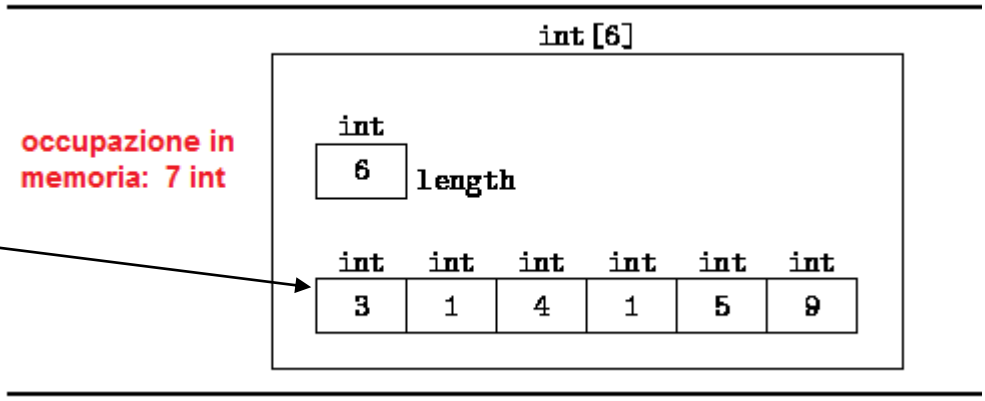
**Ricerca** in array **non ordinato**; in array ordinato (sequenziale o binaria)

***Link*** considerato utile da un gruppo, nel lavoro di venerdì:

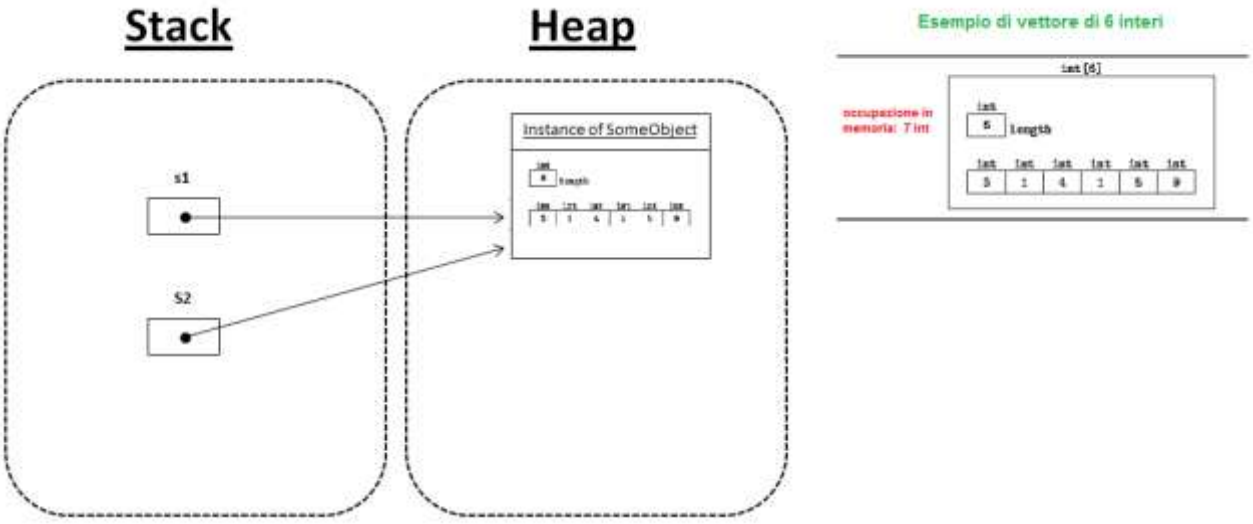
http://www.di.unipi.it/~dipierro/Didattica/LIP-C/Controllo/Array/main.html
dispense datate (corso 2001-2002) ma valide; può confondere una rappresentazione grafica, tipica anche in ambiente **BlueJ** che induce a non comprendere che il nome del vettore è l'indirizzo del primo elemento:




Correttamente si evidenzia che in RAM si occupa spazio per memorizzare sia la dimensione sia i valori degli elementi (ad esempio ti tipo intero nella figura a lato)




In modo più completo, nell'illustrazione successiva, si crea un vettore cioè un oggetto di **nome s1** che contiene 6 elementi di tipo intero:

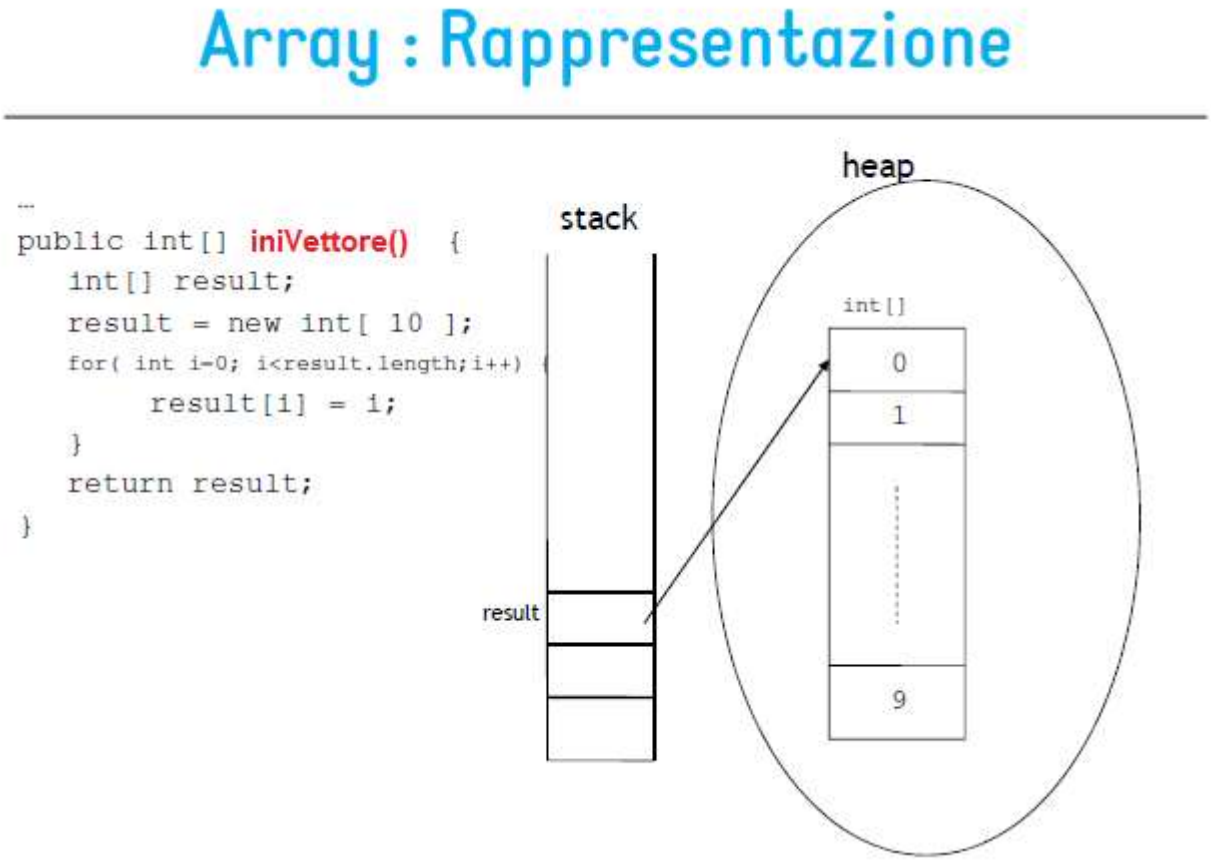



Poi si crea, con **nome s2**, una copia dell'indirizzo o *reference*: entrambi puntano allo stesso oggetto

**s2** = **s1**;

***Esempio di metodo per inizializzare un vettore***:

Si consideri un vettore di 10 elementi di tipo intero con nome collettivo **result**: ogni elemento viene inizializzato col valore dell'indice




Tratto da risorsa non più attiva (per approfondire: il *ciclo di vita di un oggetto*)

***Percorso di approfondimento successivo dalla pagina sul sito della docente***

- matrici a più dimensioni
- array vs vettori (array dinamici) da C. Horstmann "Fondamenti di Programmazione e Java 2" Apogeo