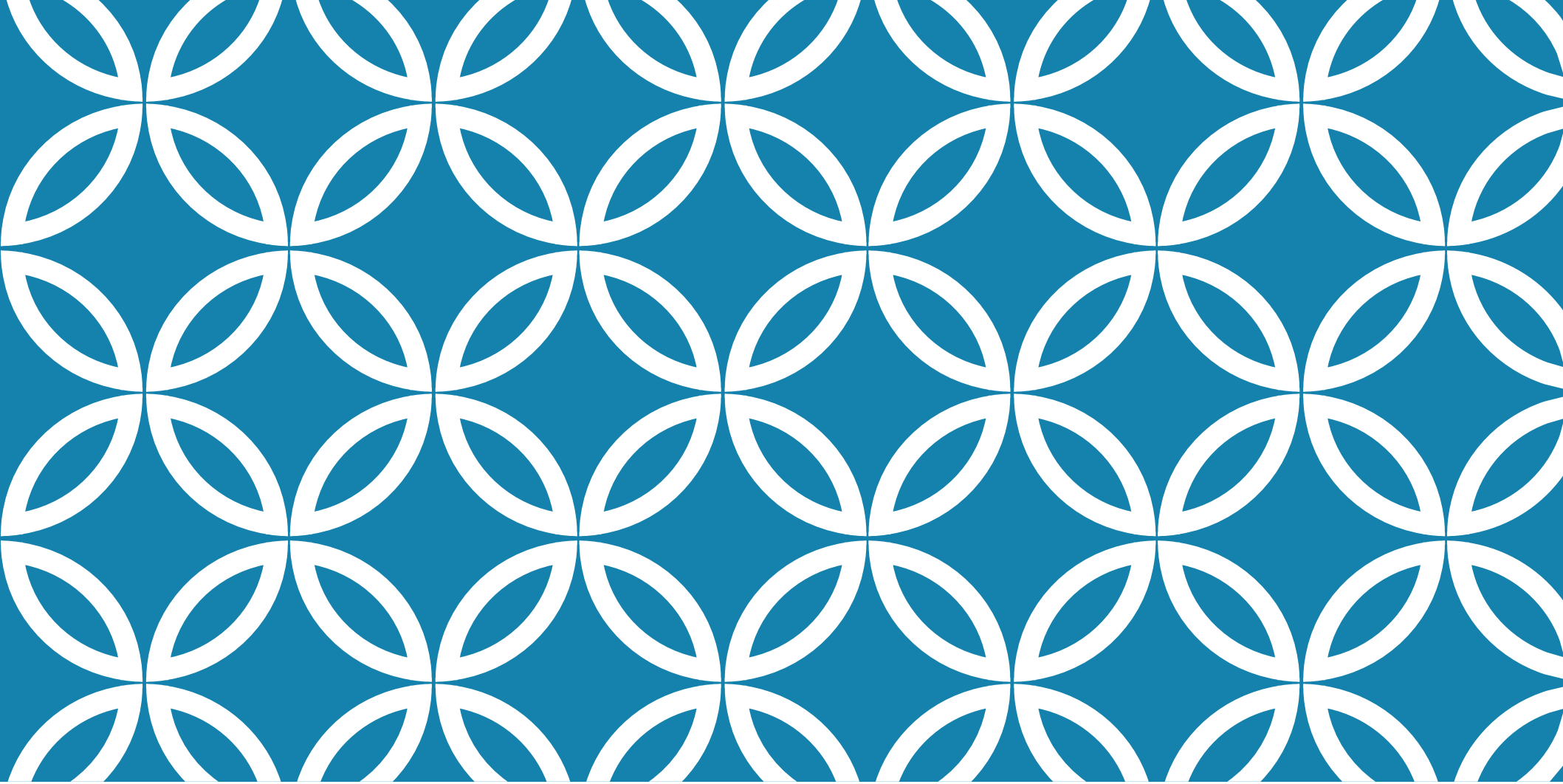


LABORATORIO DI PROGRAMMAZIONE

Corso di Laurea Ing. Gestionale 21/22

Ing. Antonio Luca Alfeo

luca.alfeo@ing.unipi.com

ARRAY (1/2)

- Un array o vettore rappresenta un insieme di elementi dello stesso tipo.
- Sintassi per la creazione di un array:

```
tipo_elementi[] nome;
nome = new tipo_elementi[numero_elementi];
```

- Esempio:

```
int[] v;          // riferimento all'array
v = new int[5]; // creazione di un array di 5 interi
                  // inizializzati a 0
```

- Accesso ad un elemento dell'array:

```
nome[indice]    // l'indice dell'array parte da 0
```

- Esempio:

```
v[0] = 2;         // assegnamento del primo elemento
v[2] = 4;         // assegnamento del terzo elemento
```

ARRAY (2/2)

- È possibile inizializzare un vettore all'atto della creazione. In tal caso, non va specificata la dimensione (che è implicita):

```
v = new int[] {1, 3, 5, 7, 9};
```

- La proprietà length dell'array contiene la sua lunghezza.

```
int l = v.length;          // l == 5;
```

Attenzione! A differenza di quanto accade nelle stringhe, length non è un metodo, quindi non va invocato usando le parentesi.

- Esempio di output del contenuto di un vettore:

```
for(int i = 0; i < v.length; i++){
    System.out.println(v[i]);
}
```

STAMPA DEGLI ARRAY

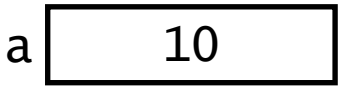
Stampa a video gli elementi di un array NON NULLO nella forma { 1, 2, 3, 4 }

```
public static void printValues(int[] v) {
          int n = v.length;
          System.out.print("{ ");
        for (int i = 0; i < n; i++) {
           if (i < n - 1) {
                System.out.print(v[i] + ", ");
            } else {
                System.out.print(v[i] + " ");
            }
           }
           System.out.println("}" + '\n');
    }
```

RIFERIMENTI

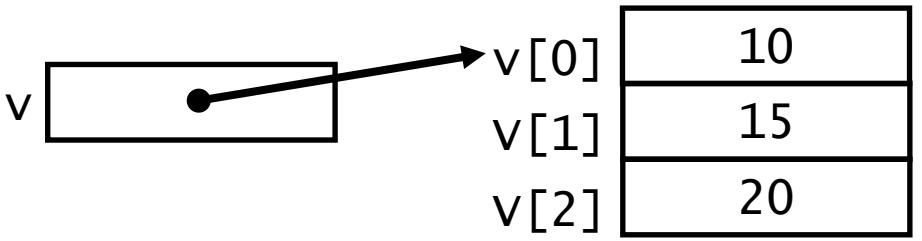
- Una variabile di un tipo primitivo (int, double, boolean, char, …) viene allocata automaticamente e contiene il valore che il programmatore le assegna.

```
int a = 10;
```



- Quando si definisce una variabile di tipo array, invece, viene allocato un riferimento che punterà alla zona di memoria dove verrà effettivamente allocato l'array tramite la parola chiave "new".

```
int[] v;
v = new int[] {10, 15, 20};
```



- Similmente a quanto accade per gli array, anche le stringhe sono gestite tramite riferimenti.

RIFERIMENTI NULLI

- La parola chiave `null` si usa per indicare un riferimento nullo.

```
int[] v = null;      // v non punta a nessun vettore
String str = null;   // str non punta a nessuna stringa
```

- Non è possibile utilizzare la proprietà "length" dei vettori, o i vari metodi delle stringhe (equals, length, charAt, …) su riferimenti nulli.
  - Prima di utilizzare i metodi o le proprietà di un oggetto, assicurarsi che il riferimento non sia null
  - Se si tenta di invocare un metodo o di accedere ad una proprietà attraverso un riferimento nullo, il programma dà errore quando eseguito.

```
Exception in thread "main"
java.lang.NullPointerException
```

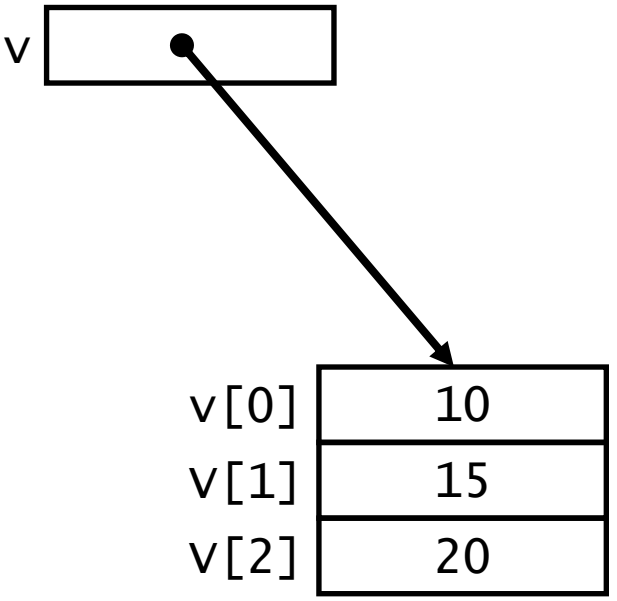
EFFETTO COLLATERALE (1/3)

- Utilizzando i riferimenti, è possibile modificare i valori di un array passato come parametro ad un metodo

```
public static void f(int[] fv){
    fv[0] = 5;
}

public static void main(String[] args) {
    int[] v;
    v = new int[] {10, 15, 20};

}
```



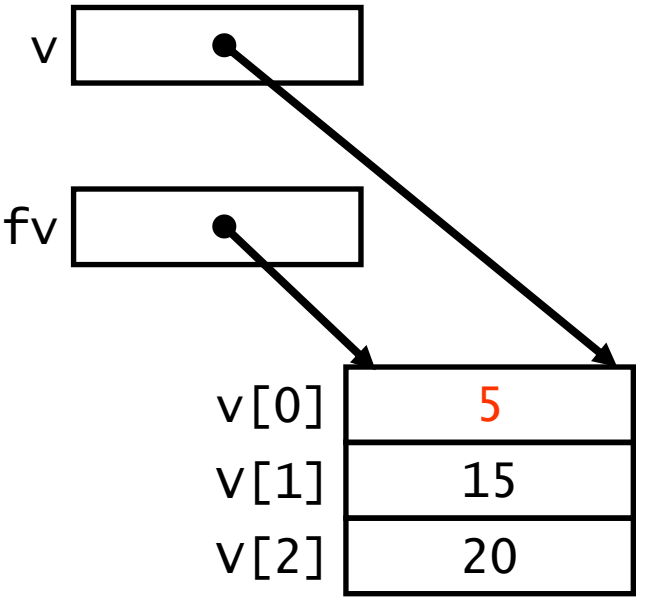
- Si dice che un metodo ha un "effetto collaterale" quando modifica un ambiente non locale.

EFFETTO COLLATERALE (2/3)

- Utilizzando i riferimenti, è possibile modificare i valori di un array passato come parametro ad un metodo

```
public static void f(int[] fv){
    fv[0] = 5;
}

public static void main(String[] args) {
    int[] v;
    v = new int[] {10, 15, 20};
    f(v);
    // il valore di v[0] viene cambiato!
}
```



EFFETTO COLLATERALE (3/3)

Per effetto collaterale, si possono scambiare gli elementi di una array all'interno di un metodo:

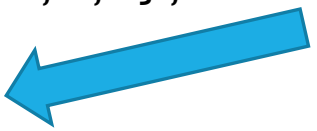
```
/* Per il vettore v passato come parametro, scambia l'elemento i-esimo
con quello j-esimo! */
public static void scambiaValori(int[] v, int i, int j){
    int tmp = v[i];
    v[i] = v[j];
    v[j] = tmp;
}

public static void main(String[] args) {
    int[] vettore = new int[] {2,3,5,1,8,6};

    int primoIndice = 0;
    int ultimoIndice = v.length - 1;
    printValues(vettore); // output {2,3,5,1,8,6}

    scambiaValori(vettore, primoIndice, ultimoIndice);

    printValues(vettore); // output {6,3,5,1,8,2}
}
```



ORDINAMENTO "SEMPLICE"

```
public class Ordinamento {

    public static void ordina(int[] v) {
        /* v viene ordinato per effetto collaterale */
        for (int i = 0; i < v.length - 1; i++) {
            for (int j = i + 1; j < v.length; j++) {
                if (v[i] > v[j]) {
                    // scambio degli elementi
                    int tmp = v[i];
                    v[i] = v[j];
                    v[j] = tmp;
                }
            }
        }
    }

    public static void main(String[] args) {
        int[] v = new int[] {2,5,3,1,8,6};
        ordina(v);
        printValues(v); /* output {1,2,3,5,6,8} */
    }

}
```

ORDINAMENTO PER SELEZIONE

```
/* Ordinamento con scambio nel ciclo esterno, dopo aver trovato
l'indice dell'elemento minimo */
  public static void selectionSort(int[] v) {

        int n = v.length;
      for (int i = 0; i < n-1; i++) {

          int min = i;
        for (int j = i + 1; j < n; j++) {
            if (v[min] > v[j]) {
                min = j
            }

          }

          // scambio degli elementi
        int tmp = v[i];
        v[i] = v[min];
        v[min] = tmp;
      }
  }
```

VERIFICA ORDINAMENTO

```
/* Verifico che l'array sia ordinato in maniera crescente,
assumendo che il riferimento v sia non nullo */
  public static boolean verifyOrder(int[] v){

        boolean order = true;

        int n = v.length;
      for (int i = 0; i < n-1 && order; i++) {

              if (v[i] > v[i+1]) {
                 order = false;
            }

        }

        return order;
  }
```

ESERCIZIO "PRODOTTOSCALARE"

- Scrivere un metodo "*prodotto*" che ha come parametri due array di interi di pari lunghezza e restituisce un intero calcolato come prodotto scalare tra i due array.

- Scrivere il metodo main che legga da tastiera un intero "n" rappresentante la dimensione dei due array, gli "n" interi del primo array e gli "n" interi del secondo array e stampi il risultato del metodo "prodotto", invocato con i due array letti come parametri.

SOLUZIONE (1/3)

```
import fiji.io.Lettore;

 public class ProdottoScalare {

     public static int prodotto(int[] a, int[] b) {
         int r = 0;
         for(int i = 0; i < a.length; i++){
             r += a[i] * b[i];
         }
         return r;
     }
```

SOLUZIONE (2/3)

```
public static void main(String[] args) {
        int n, r;
        int[] a, b;
        System.out.println("Inserisci la lunghezza degli array");
        n = Lettore.in.leggiInt();
        a = new int[n];
        System.out.println("Inserisci " + n + " interi (primo array)");
        for (int i = 0; i < n; i++){
            a[i] = Lettore.in.leggiInt();
        }
```

SOLUZIONE (3/3)

```
        b = new int[n];
         System.out.println("Inserisci " + n +
                 " interi (secondo array)");
         for (int i = 0; i < n; i++){
             b[i] = Lettore.in.leggiInt();
         }


         r = prodotto(a, b);

         System.out.println("Il risultato è: " + r);

    }

}
```

ESERCIZIO "SOMMAUGUALI"

- Scrivere un metodo "somma" che ha come parametri due array di interi di pari lunghezza e restituisce un array di interi della stessa lunghezza i cui elementi sono calcolati in modo tale che ogni elemento dell'array di output sia pari alla somma dei corrispondenti elementi nei due array di input se questi sono uguali, altrimenti sia 0.
- Scrivere il metodo main che legga da tastiera un intero "n" rappresentante la dimensione dei due array, gli "n" interi del primo array e gli "n" interi del secondo array e stampi il risultato del metodo "somma", invocato con i due array letti come parametri.

SOLUZIONE (1/3)

```
import fiji.io.Lettore;

public class SommaUguali {

    public static int somma(int[] a, int[] b) {
        int[] r = new int[a.length];
        for(int i = 0; i < a.length; i++){
            if(a[i] == b[i]){
                r[i] = a[i] + b[i];
            }else{
                r[i] = 0;
            }
        }
        return r;
    }
```

SOLUZIONE (2/3)

```
public static void main(String[] args) {
        int n;
        int[] a, b, r;
        System.out.println("Inserisci la lunghezza degli array");
        n = Lettore.in.leggiInt();
        a = new int[n];
        System.out.println("Inserisci " + n +
                " interi (primo array)");
        for (int i = 0; i < n; i++){
            a[i] = Lettore.in.leggiInt();
        }
```

SOLUZIONE (3/3)

```
        b = new int[n];
        System.out.println("Inserisci " + n +
                " interi (secondo array)");
        for (int i = 0; i < n; i++){
            b[i] = Lettore.in.leggiInt();
        }
        r = somma(a, b);

        System.out.println("Il risultato è:");
        for (int i = 0; i < n; i++){
            System.out.println(r[i]);
        }
    }

}
```

ESERCIZIO "CONFRONTOARRAYSTRINGHE" (1/2)

- Scrivere il metodo "confronta" per il confronto robusto tra due array di stringhe con la seguente intestazione:

```
public static boolean confronta(String[] a, String[] b)
```

- Il metodo deve ritornare true se i due riferimenti sono entrambi nulli, oppure se puntano ad array della stessa lunghezza ed aventi in posizioni corrispondenti due stringhe uguali oppure due riferimenti a stringa nulli.

ESERCIZIO "CONFRONTOARRAYSTRINGHE" (2/2)

- Testare il metodo con il seguente main:

```
public static void main(String[] args) {
    String[] s1 = new String[] { "hello", "world" };
    String[] s2 = new String[] { "hello", "world" };
    String[] s3 = null;
    String[] s4 = new String[] { "hello", "world", null };
    String[] s5 = new String[] { "hello", null };

    System.out.println(confronta(s1, s1));      // true
    System.out.println(confronta(s1, s2));      // true
    System.out.println(confronta(s1, s3));      // false
    System.out.println(confronta(s1, s4));      // false
    System.out.println(confronta(s1, s5));      // false
    System.out.println(confronta(s3, null));    // true
}
```

SOLUZIONE

```
public static boolean confronta(String[] a, String[] b) {
     boolean uguali = true;
     /* Se a == b i riferimenti sono entrambi null oppure puntano
      * allo stesso array. In entrambi i casi va ritornato true
 */
     if(a != b){
         if(a == null || b == null || a.length != b.length){
             uguali = false;
         }else{
             for(int i = 0; i < a.length && uguali; i++){
                 if(a[i] != b[i]
                         && (a[i] == null
                             || b[i] == null
                             || !a[i].equals(b[i])) ){
                     uguali = false;
                 }
             }
         }
     }
     return uguali;
 }
```