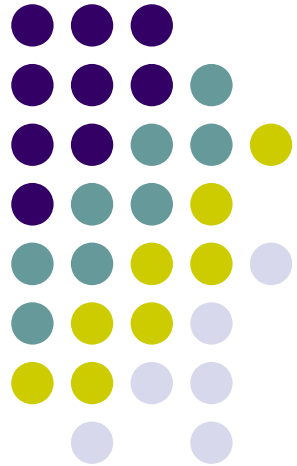
Università di Napoli Federico II - Dip. di Ingegneria Elettrica e delle Tecnologie dell'Informazione
Corso di Laurea in Ingegneria Informatica

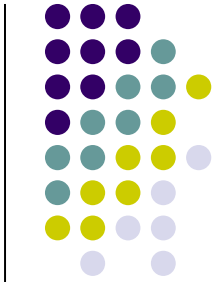
DIE TI

Corso di Programmazione

Array

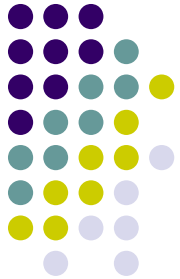


Array in Java



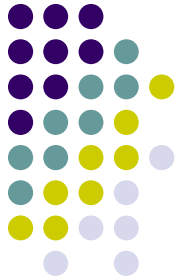
- Ricordiamo che i tipi in Java appartengono a due categorie:
  - Tipi primitivi
  - Riferimenti ad oggetti
- Gli array in Java sono oggetti
- Quindi aggregati di variabili in memoria tutti dello stesso tipo (primitivo o riferimento) a cui si arriva tramite un riferimento

Gli array



- Gli array possono essere monodimensionali (vettori) o avere più dimensioni, (gli array bidimensionali sono matrici)
- Gli elementi di un array vengono inizializzati ad un valore di default quando l'array viene creato (se non diversamente specificato)
  - null per i riferimenti
  - 0 per gli interi
  - false per i boolean
- Come in C l'accesso ai singoli elementi dell'array è diretto ed è effettuato mediante la notazione [] e la posizione dell'elemento nell'array
- Come in C *la prima posizione del vettore è la posizione 0*
- A differenza del C il compilatore effettua sempre il controllo sui limiti (non si può accedere al di fuori dell'intervallo degli indici)
- Nota: In C++ è disponibile anche la classe vector…

Array monodimensionali (vettori)

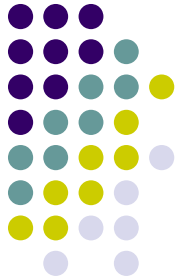


- Dichiarazione e Creazione:

```
int a[] = new int[10];
```

- La precedente istruzione dichiara il riferimento *a* ad un array monodimensionale e lo inizializza con il valore tornato da new
- new alloca in memoria un array di 10 interi e ritorna il riferimento all'area di memoria
- NOTA: int a[10] in Java è un errore di sintassi

Esempi di uso dei vettori



- Un vettore, una volta a disposizione il riferimento, si usa come in C/C++, ricordando sempre che in realtà utilizziamo un riferimento
- Esempi:

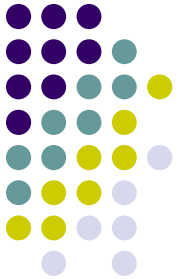
Lettura da tastiera del vettore *a:*

```
Scanner input = new Scanner(System.in);
for(int i=0; i<10; i++)
        a[i]=input.nextInt();
```

Visualizzazione del vettore a:

```
for(int i=0; i<10; i++)
  System.out.printf(" %d", a[i]);
```

Inizializzazione

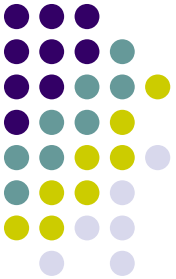


- E' possibile specificare il valore dei singoli elementi di un vettore all'atto della sua creazione:

```
int a[] = {4, 5, 1, 13, 0 };
```

- In questo caso la chiamata di new è implicita, *il compilatore deriva la dimensione del vettore dal numero di elementi specificati* e esegue l'opportuna istruzione new

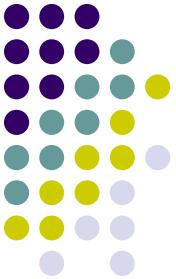
lenght



- E’ possibile determinare la dimensione massima di un vettore interrogando la variabile di istanza pubblica ***lenght***

```
int a[] = {4, 5, 1, 13, 0 };
for(int i=0; i<a.lenght; i++)
  System.out.printf(" %d", a[i]);
```

Istruzione for su array

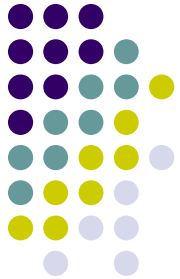


- Dalla JDK 5.0 in poi l'istruzione for è stata «potenziata» per iterare sugli elementi di una qualsiasi collezione, e in particolare sugli elementi di un array, *senza dover usare un contatore:*

```
for( parametro : nomeArray)
    blocco del for
```

- *parametro* è specificato da un tipo compatibile con il tipo degli elementi dell'array e da un identificatore

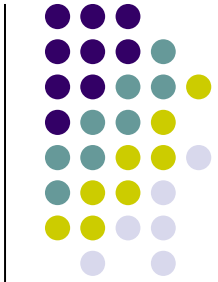
Esempio:



```
double array[] = {3.4, 2.5, 1.7, 9.0};
        double somma = 0.0;
        for(double elem : array)
            somma += elem;
        System.out.printf("somma degli elementi del vettore: %f \n", somma);
```

- Questo utilizzo del for semplifica il codice nel caso in cui si debba accedere *in lettura* a tutti gli elementi del vettore
- Non può essere utilizzato se gli elementi devono essere modificati
- Non può essere utilizzato se le operazioni non devono essere effettuate su tutti gli elementi del vettore

Copia



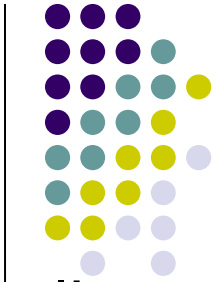
- Non è possibile copiare gli array con l' operatore =

```
Esempio:
int a[] = {4, 5, 1, 13, 0 };
Int b[] ;
b=a; // il riferimento a viene copiato in b
```

- Se si vuole ottenere una copia dell'oggetto è necessario creare l'oggetto puntato da *b* della stessa dimensione di quello puntato da *a* e copiare elemento per elemento

```
int a[] = {4, 5, 1, 13, 0 };
int b[]  = new[a.lenght];
for(int i=0; i<a.lenght; i++)
  b[i]=a[i];
```

Confronto

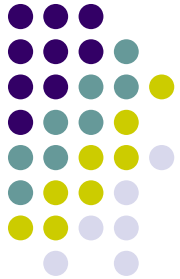


- Analogamente non è possibile confrontare gli array con l'operatore ==

Esempio:
int a[] = {4, 5, 1, 13, 0 };
Int b[] = {4, 5, 1, 13, 0 };
If(b==a) // il *riferimento b* viene confrontato con *a*
{//...}

- Nell'esempio i due oggetti hanno lo stesso stato ma sono oggetti diversi, quindi il valore dei loro riferimenti è diverso
- Anche in questo caso è necessario confrontare gli oggetti elemento per elemento

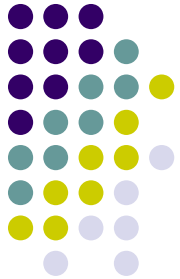
Dimensione e riempimento



- All'atto della creazione di un array bisogna specificare la sua dimensione MASSIMA
- La dimensione fornita a new può anche essere una variabile (il cui valore è noto a tempo di esecuzione)
- Ovviamente il valore deve essere assunto dalla variabile prima del suo uso in new
- In generale però non è necessario o opportuno utilizzare tutti gli elementi allocati, come anche in C/C++ si può utilizzare il concetto di riempimento
- In riempimento sarà a sua volta una variabile che in ogni istante fornisce il numero corrente di elementi effettivamente presenti nell'array

```
int a[] = new int[200];  //dimensione massima: 200 elementi
int n; //riempimento
Scanner input = new Scanner(System.in);
System.out.printl("Quanti elementi vuoi nel vettore? ");
n = input.nextInt();
System.out.printl("Inserisci il vettore:");
for(int i=0; i<n; i++)  //vengono usati solo n elementi su 200
      a[i]=input.nextInt();
```

Array bidimensionali



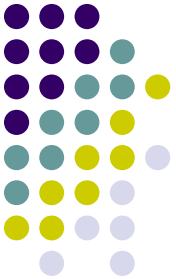
- Per le matrici valgono tutte le considerazioni già fatte per i vettori

```
int m1[][] = new int[4][5];                  //crea una matrice 4x5
int m2[][] = {{ 4, 5, 9 } , {1, 13, 0 } };  //crea una matrice 2x3
                                             // e la inizializza ai valori specificati
int m3[][] = {{ 4, 5 } , { 8 },  {1, 13, 0 } };    ????
```

- E' possibile avere array bidimensionali in cui ogni riga ha un diverso numero di elementi!

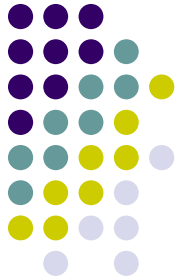
```
int m[][] = new int[2][];  //crea 2 righe
m[0] = new int[5];    // la prima riga ha 5 colonne
m[1] = new int[3];   //  la seconda riga ha 3 colonne
```

Esempio: visualizzazione degli elementi



```
int m3[][] = {{ 4, 5 } , { 8 },  {1, 13, 0 } };
        for(int r = 0; r < m3.length; r++){
            for(int c = 0; c < m3[r].length; c++)
                System.out.printf(" %d", m3[r][c]);
         System.out.println();
        }
```

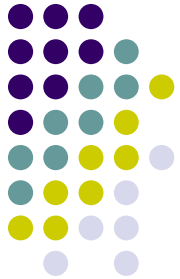
Esempio: manipolazioni



- E’ possibile manipolare gli array bidimensionali per righe, il for seguente imposta a -1 tutti gli elementi della terza riga della matrice m3

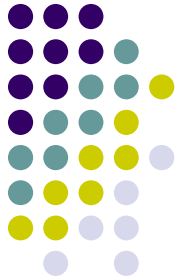
```
for(int i = 0; i<m3[2].length; i++)
        m3[2][i] = -1;
```

La classe Arrays



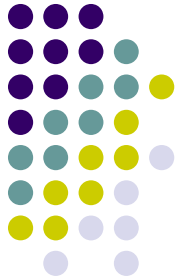
- Fornisce metodi statici per le operazioni più comuni sugli array.
- Questi metodi includono
  - **sort** per l'ordinamento di un array (per esempio, la disposizione degli elementi in ordine crescente),
  - **binarySearch** per la ricerca in array ordinati (per esempio, per determinare se un array contiene un valore specifico e, in tal caso, dove si trova il valore),
  - **equals** per il confronto di array,
  - **fill** per l'inserimento di valori in un array.
- Questi metodi, grazie all'overloading, sono applicabili sia agli array di tipi primitivi sia agli array di oggetti.

Esempio 1: ordinamento



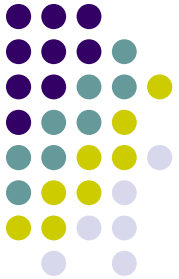
```
public static void main(String[] args) {

        double[] doubleArray = {8.4, 9.3, 0.2, 7.9, 3.4};
        //ordiniamo il vettore doubleArray
        Arrays.sort(doubleArray);

        for (double value : doubleArray) {
            System.out.printf("%.1f ", value);
         }
}
```

Esempio 2: copia di array



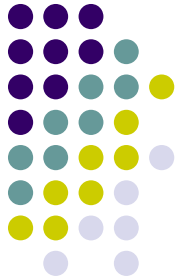
```
public static void main(String[] args) {
        int[] arrayDiInteri = {3, 9, 11, 4, 77};
        //copiamo arrayDiInteri in nuovoArray
        int[] nuovoArray = new int[arrayDiInteri.length];
        System.arraycopy(arrayDiInteri,0, nuovoArray, 0, arrayDiInteri.length);
        displayArray(arrayDiInteri, "arrayDiInteri");
        displayArray(nuovoArray, "nuovoArray");
}

     // visualizza valori in ogni array
    public static void displayArray(int[] array, String description) {
        System.out.printf("%n%s: ", description);
        for (int value : array) {
            System.out.printf("%d ", value);
        }
    }
```

Esempio 3: ricerca binaria



```
public static void main(String[] args) {
        int[] arrayDiInteri = {3, 9, 11, 9, 77};
        //cerchiamo il valore 9 in arrayDiInteri
        int valore=9;
        int posizione = Arrays.binarySearch(arrayDiInteri, valore);
        if (posizione >= 0) {
            System.out.printf("Valore %d trovato in posizione %d%n",valore, posizione);
        }
}
```

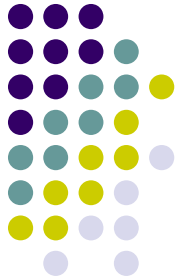
Esempio 4: verifica uguaglianza



```
public static void main(String[] args) {
        int[] primoArrayDiInteri = {3, 9, 11, 9, 77};
        int[] secondoArrayDiInteri = {3, 9, 11, 9, 18};
        //confrontiamo i due Array
        boolean verifica = Arrays.equals(primoArrayDiInteri,
secondoArrayDiInteri);
        if (verifica) {
            System.out.println("I due array sono uguali");
        } else {
            System.out.println("I due array sono diversi");
        }
 }
```

Quando si confrontano i contenuti degli array occorre utilizzare sempre Arrays.equals(array1, array2), che confronta i contenuti dei due array, anziché array1.equals(array2), che confronta se array1 e array2 si riferiscono allo stesso oggetto array.

Liste di argomenti di lunghezza variabile



- Con le **liste di argomenti di lunghezza variabile** è possibile creare metodi che prendono una quantità non specificata di argomenti.
- Un tipo di argomento seguito da **tre punti** (**...**) **nell'elenco dei parametri di un metodo** indica che il metodo può ricevere una quantità variabile di argomenti di quel particolare tipo.
  - I tre punti possono essere inclusi una sola volta in un elenco dei parametri, e necessariamente in fondo.
  - ***Errore tipico:*** *Inserire i tre punti che indicano una lista di argomenti di lunghezza variabile in mezzo alla lista dei parametri di un metodo è un errore di sintassi. I tre punti possono essere inseriti solo alla fine della lista dei parametri.*

Esempio

```
public static double media (double... numeri){
        double somma = 0.0;
        //calcola la somma usando il for potenziato
        for(double d: numeri) {
            somma += d;
        }
        return somma/numeri.length;
    }

    public static void main(String[] args) {
        double a1 = 10.0;
        double a2 = 40.0;
        double a3 = 50.0;
        double a4 = 75.0;
        System.out.printf("media di %.1f e %.1f: %.1f%n",
a1, a2, media(a1,a2));
        System.out.printf("media di %.1f, %.1f e %.1f: %.1f
%n", a1, a2, a3, media(a1,a2, a3));
        System.out.printf("media di %.1f, %.1f, %.1f e %.1f:
%.1f%n", a1, a2, a3, a4, media(a1,a2, a3, a4));
    }
```

Riferimenti