Metodologie di Programmazione

Lezione 11: Gli array

Lezione 11: Sommario

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Gli array
- Dichiarazione
- Inizializzazione
- Accesso agli elementi
- Array a due dimensioni
- Metodi con numero variabile di parametri

Array

- Un array rappresenta un gruppo di variabili (chiamate elementi) tutte dello stesso tipo
- Gli array **sono oggetti**
- Quindi le variabili di array contengono il **riferimento** all'array
- Gli elementi di un array possono essere valori di tipi primitivi (interi, double, ecc.) oppure riferimenti a oggetti (inclusi altri array!)

Un esempio di array

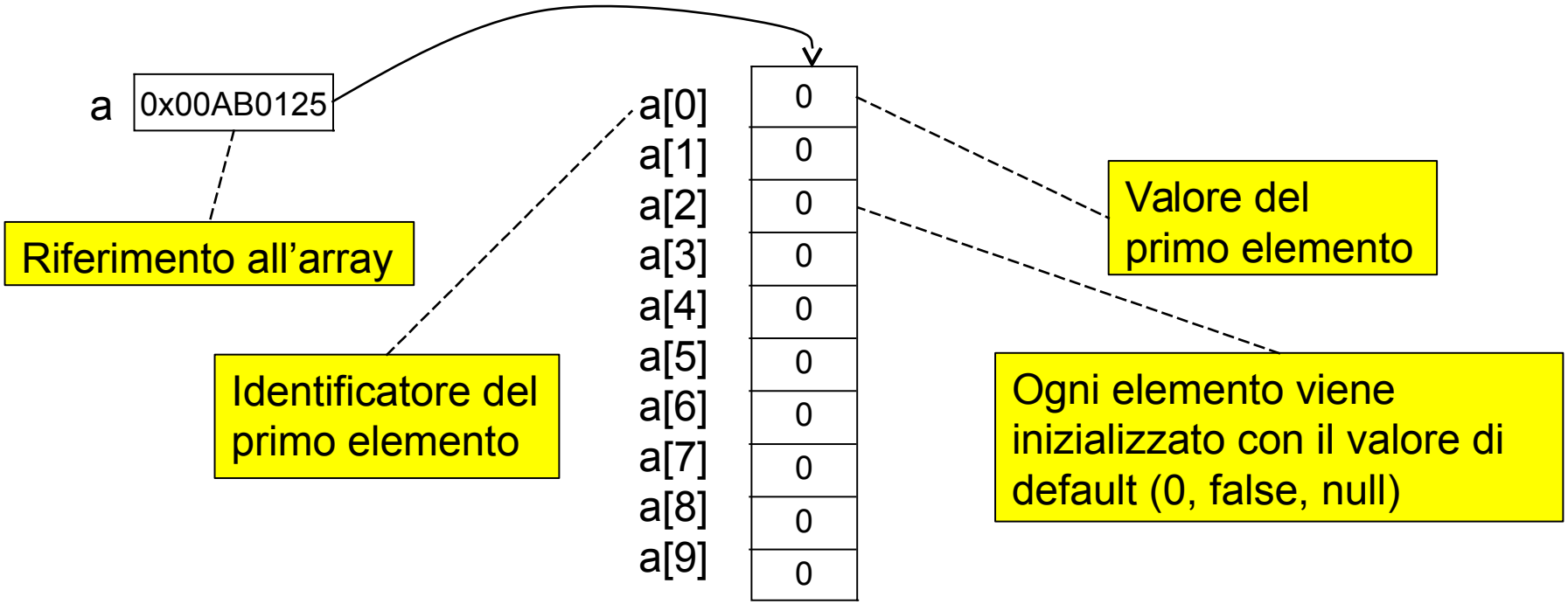
- Dichiarazione:

```
int[] a;
```

Nome dell'array

- Creazione senza valori:

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



Un esempio di array

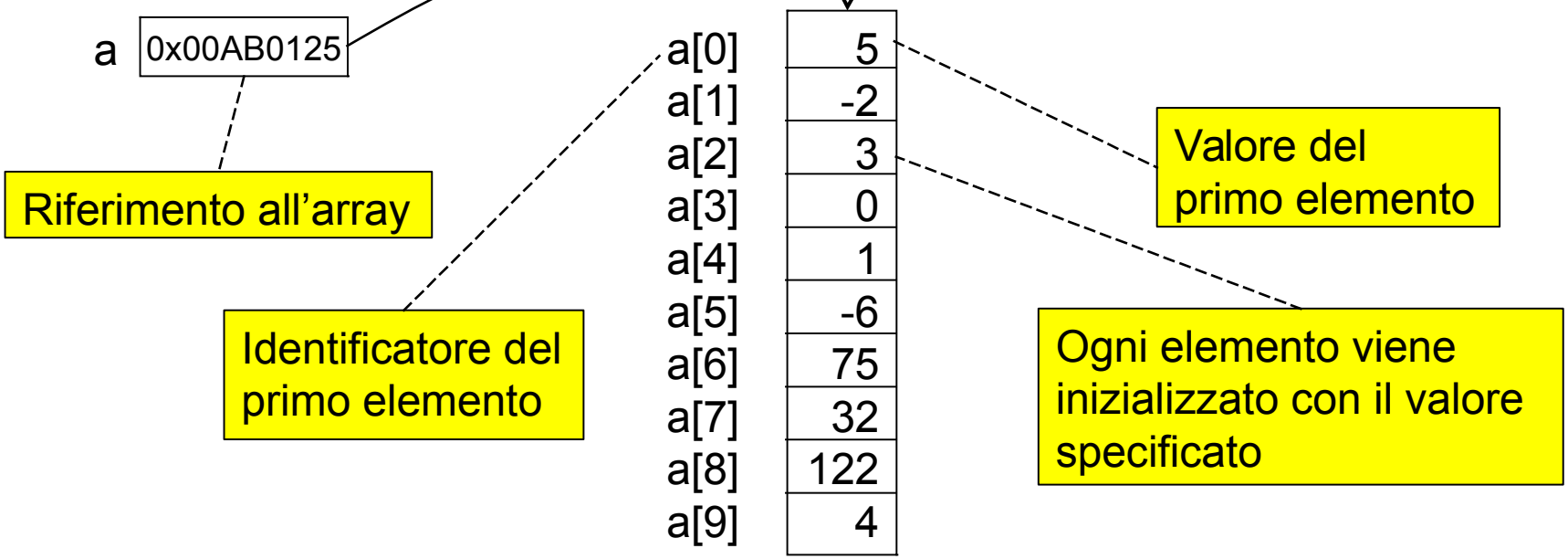
- Dichiarazione:

```
int[] a;
```

Nome dell'array

- Creazione con valori:

```
a = new int[] { 5, -2, 3, 0, 1, -6, 75, 32, 122, 4 };
```



Dichiarazioni valide di array

Un array di 10 interi. Tutti inizializzati a 0.	`int[] numeri = new int[10];`
Meglio utilizzare una COSTANTE per la dimensione di un array.	`final int NUMERO_DI_CIFRE = 10;` `int[] numeri = new int[NUMERO_DI_CIFRE];`
La lunghezza di un array può essere data dal valore di una variabile	`int numeroDiCifre = new Scanner(System.in).nextInt();` `int[] numeri = new int[numeroDiCifre];`
Un array di 10 interi, con valori assegnati	`int[] numeri = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };`
Un array di 3 riferimenti a stringa, tutti inizializzati a null	`String[] nomi = new String[3];`
Un array di 3 riferimenti a stringa, con valori assegnati	`String[] nomi = { "mario", "luigi", "wario" };`

Inizializzazione di un array

- E' possibile inizializzare un array con la **sintassi estesa** new in qualsiasi momento:

```
int[] a = new int[] { 1, 2, 3, 4 };
int[] b;
…
b = new int[] { 1, 2, 3 };
```

- Oppure con la **sintassi abbreviata**:

```
int[] a = { 1, 2, 3, 4 }
```

- Ma la sintassi abbreviata può essere usata **solo in fase di definizione di un array**, non successivamente:

```
int[] b;
…
b = { 1, 2, 3 };
```

Errori tipici con gli array!

- Al contrario del linguaggio C, in Java non è possibile specificare la dimensione accanto al nome dell'array:

```
int a[10];
```

- Né specificare la dimensione e allo stesso tempo inizializzare i valori:

```
int[] a = new int[10] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

Accedere agli elementi di un array

- Si accede a un elemento dell'array specificando il nome dell'array seguito dalla posizione (indice) dell'elemento tra parentesi quadre

```
// stampa il sesto elemento dell'array
System.out.println(a[5]);

// memorizza la somma dei primi 3 elementi
int k = a[0] + a[1] + a[2];
```

- L'indice è sempre positivo, compreso da 0 e la dimensione dell'array-1
- L'indice può ess , essione:

```
int i = 2;
int j = 3;

a[i+j*2] += 2;
```

Esercizio: stampa di un array

- Scrivere un metodo che, dato un array di stringhe, ne stampi i valori in sequenza
- Ad esempio, dato l'array { "son", "buoni", "almeno?", "assaggi", "il", "vino" } stampi: ["son", "buoni", "almeno", "assaggi", "il", "vino"]

```
public void stampaArray(int[] a)
{
    System.out.print("[ ");

    for (int k = 0; k < a.length; k++)
        System.out.print("\"" + a[k] + "\"" + (k == a.length-1 ? " ]" : ", "));

    System.out.println();
}
```

Lunghezza dell'array (**attenzione:** .length **SENZA PARENTESI**)

Esercizio: somma dei valori di un array

- Scrivere un metodo che, dato un array di interi, restituisca la somma dei suoi elementi
- Ad esempio, dato l'array { 10, 12, 12, 8 } il metodo deve restituire: 42

```
public int sommaArray(int[] a)
{
    int val = 0;

    for (int k = 0; k < a.length; k++) val += a[k];

    return val;
}
```

Esercizio: media dei valori di un array

- Utilizzando il metodo scritto per l'esercizio precedente, scrivere un metodo che, dato un array di interi, restituisca la media (double) dei suoi elementi
- Ad esempio, dato l'array { 10, 12, 12, 8 } il metodo deve restituire: 10.5

```
public double mediaArray(int[] a)
{
    double somma = sommaArray(a);
    return somma/a.length;
}
```

- Scrivete anche la versione per array di double!

Altri esercizi semplici con gli array

- Scrivere un metodo che, dato un array di stringhe e una stringa in input, restituisca true se l'array contiene la stringa, false altrimenti
- Scrivere una seconda versione del metodo che restituisca la posizione della stringa trovata, -1 altrimenti
- Scrivere un metodo che, dato un array di double, restituisca il valore massimo dell'array

Esercizio: reimplementa conta vocali

- Scrivere un metodo che riceve una stringa e stampa a video il conteggio delle vocali in essa contenute
  - Utilizzare un array per il conteggio separato delle 5 vocali
- Ad esempio: data la stringa "le aiuole sono pulite", il metodo stampa:

```
a=1 e=3 i=2 o=3 u=2
```

- Come l'avevate implementato **SENZA** array?

Esercizio: array con fattoriale

- Scrivere un metodo che, dato un intero n in ingresso, restituisca un array di dimensione n contenente k! nella k-esima cella, per ogni valore di k

```
public int[] getArrayFattoriali(final int n)
{
    int[] fatt = new int[n];

    fatt[0] = 1;
    for (int k = 1; k < n; k++) fatt[k] = fatt[k-1]*k;

    return fatt;
}
```

Esercizio: stampa di istogrammi

- Progettare una classe Istogramma che rappresenta la distribuzione di dati (es. voti degli studenti) in un intervallo da i a j fornito in input (es. da 0 a 31 (trenta e lode))
- La classe permette di incrementare il conteggio in corrispondenza di ciascun elemento dell'intervallo (es. memorizzando così un nuovo voto di uno studente)
- La classe può stampare a video l'istogramma corrispondente
  - Più facile in orizzontale
  - Provate a stampare in verticale!!!

Esercizio: mescolare e distribuire un mazzo di carte da gioco

- Progettare una classe Carta che rappresenti una singola carta da gioco (con seme e valore)
- La classe deve restituire su richiesta la propria rappresentazione sotto forma di stringa
- Progettare quindi una classe MazzoDiCarte che rappresenti un intero mazzo da 52 carte
- La classe deve implementare i seguenti metodi:
  - mescola il mazzo di carte
  - distribuisci la prossima carta
- Infine si progetti una classe di collaudo che crea un mazzo, mescoli le carte e ne distribuisca carte fino ad esaurimento del mazzo

Modificare le dimensioni di un array

- Un array ha dimensioni prefissate che **NON** possono essere modificate
- Tuttavia è possibile creare un nuovo array con nuove dimensioni a partire da un array preesistente
- Con il metodo statico copyOf della classe java.util.Arrays

```
import java.util.Arrays;


public class MyArrays
{
    public static void main(String[] args)
    {
        // array di dimensione 10
        int[] array = { 1, 5, 8, 2, 3, 4, 7, 6, 9 };

        System.out.println(Arrays.toString(array));

        // restringe l'array a dimensione 5
        array = Arrays.copyOf(array, 5);
        System.out.println(Arrays.toString(array));

        // allarga l'array a dimensione 8 (gli ultimi 3 valori sono inizializzati a 0)
        array = Arrays.copyOf(array, 8);
        System.out.println(Arrays.toString(array));
    }
}
```

Crea un nuovo array con i primi 5 elementi dell' array in input

Crea un nuovo array di 8 elementi (i primi elementi dall' array in input)

Metodo per la formattazione a stringa di un array

Output

```
[1, 5, 8, 2, 3, 4, 7, 6, 9]
[1, 5, 8, 2, 3]
[1, 5, 8, 2, 3, 0, 0, 0]
```

Esercizio: implementare un filtro

- Progettare una classe Filtro costruita con un array di interi
- La classe implementa operazioni che permettono di ottenere nuovi sotto-array dell'array iniziale:
  - passaBasso(): restituisce tutti gli elementi <= k nell'ordine iniziale
  - passaAlto(): restituisce tutti gli elementi >= k nell'ordine iniziale
  - filtra(): restituisce l'array iniziale da cui sono state eliminate tutte le occorrenze dell'intero passato in input
  - filtra(): una seconda versione del metodo che restituisce l'array iniziale da cui vengono eliminate tutte le occorrenze di interi presenti nell'array passato in input
  - Se Filtro viene costruito con l'array { 1, 2, 10, 2, 42, 7, 8 }:
    - passaBasso(8) restituisce { 1, 2, 2, 7, 8 }
    - passaAlto(9) restituisce { 10, 42 }
    - filtra(2) restituisce { 1, 10, 42, 7, 8 }
    - filtra(new int[] { 2, 7, 42 }) restituisce { 1, 10, 8 }

Esercizio: reimplementare Arrays.copyOf() e Arrays.toString()

- Implementare un metodo statico copyOf che, analogamente a java.util.Arrays.copyOf, copi un array in un nuovo array delle dimensioni specificate (troncando l'array in input, se più grande)

Esercizio: sequenza di cifre estensibile

- Progettare una classe SequenzaDiCifre che espone un metodo che, data in input una stringa e un intero N, aggiunga alla sequenza inizialmente vuota (rappresentata mediate un array) le prime N cifre contenute nella stringa (si assuma che ne contenga comunque almeno N). La classe espone anche un metodo toString che fornisce una rappresentazione sotto forma di stringa della sequenza.
- Ad esempio:

```
SequenzaDiCifre s = new SequenzaDiCifre();
s.aggiungiCifre("abc1--23", 2);
s.aggiungiCifre("xx0a8b76543100", 4);
System.out.println(s.toString());
```

- stampa: [1,2,0,8,7,6]

Soluzione 1: SequenzaDiCifre


```
public class SequenzaDiCifre
{
    /**
     * Array della sequenza, inizialmente vuoto
     */
    private int[] sequenza = new int[0];

    /**
     * Aggiunge all'array le prime n cifre della stringa
     * @param s stringa in input
     * @param N massimo numero di cifre da aggiungere
     */
    public void aggiungiCifre(String s, int N)
    {
        int[] array = new int[N];
        int count = 0;
        for (int i = 0; i < s.length(); i++)
        {
            if (Character.isDigit(s.charAt(i)))
            {
                // in alternativa: array[count] = Character.getNumericValue(s.charAt(i));
                array[count] = s.charAt(i)-'0';
                count++;
            }
            if (count == N) break;
        }

        // crea il nuovo array (vecchia dimensione + dim. nuovo array)
        int[] array2 = new int[sequenza.length+array.length];

        // copia il vecchio array (potevo usare anche Arrays.copyOf)
        for (int i=0; i < sequenza.length; i++)
            array2[i] = sequenza[i];
        // copia la parte nuova dell'array, a partire dalla posizione sequenza.length
        for (int i = sequenza.length; i < sequenza.length+array.length; i++)
            array2[i] = array[i-sequenza.length];

        // sovrascrive il vecchio riferimento all'array con il nuovo
        sequenza = array2;
    }

    /**
     * Restituisce una rappresentazione sotto forma di stringa della sequenza di cifre
     */
    public String toString()
    {
        String stringa = "[";

        for (int i=0; i<sequenza.length; i++)
            stringa += sequenza[i]+",";

        // elimina l'ultima virgola
        stringa = stringa.substring(0,stringa.length()-1);
        stringa += "]";

        return stringa;
    }
}
```

Soluzione 2: SequenzaDiCifre


```java
import java.util.Arrays;

public class SequenzaDiCifre
{
    /**
     * Array della sequenza, inizialmente vuoto
     */
    private int[] array = new int[0];

    /**
     * Aggiunge all'array le prime n cifre della stringa
     * @param s stringa in input
     * @param N massimo numero di cifre da aggiungere
     */
    public void aggiungiCifre(String stringa, int N)
    {
        // crea il nuovo array (vecchia dimensione + dim. nuovo array)
        // copiando nella prima parte il vecchio array
        int[] temp = Arrays.copyOf(array, array.length + N);

        int k = array.length;
        for (int i = 0; i < stringa.length(); i++)
        {
            if (Character.isDigit(stringa.charAt(i)))
                temp[k++] = Character.getNumericValue(stringa.charAt(i));
            if (k == N + array.length) break;
        }

        // sovrascrive il vecchio riferimento all'array con il nuovo
        array = temp;
    }

    /**
     * Restituisce una rappresentazione sotto forma di stringa della sequenza di cifre
     */
    public String toString()
    {
        return Arrays.toString(array);
    }
}
```

Esercizio: implementare una lista mediante array

- Che cos'è una lista? E' una sequenza di oggetti
- Implementare una classe ListaDiInteri che permetta le seguenti operazioni:
  - Restituisce l'elemento i-esimo della lista
  - Restituisce l'indice della posizione dell'intero fornito in input
  - Restituisce una stringa formattata contenente la lista di interi
  - Restituisce la dimensione della lista
  - Contiene un determinato intero (true o false)?
  - Aggiunge un intero in coda alla lista
  - Aggiunge un intero nella posizione specificata
  - Elimina la prima occorrenza di un intero dalla lista
  - Elimina l'elemento i-esimo della lista

Metodi con un numero di parametri variabile

- Si possono dichiarare metodi con un numero variabile di parametri (a partire da Java 5)
- Mediante la sintassi: tipo…

```
public class SommaDouble
{
    public static double sum(double... valori)
    {
        double somma = 0.0;
        for (int k = 0; k < valori.length; k++) somma += valori[k];
        return somma;
    }

    public static double sumFirstN(final int N, double... valori)
    {
        double somma = 0.0;
        for (int k = 0; k < valori.length && k < N; k++) somma += valori[k];
        return somma;
    }

    public static void main(String[] args)
    {
        System.out.println(sum());
        System.out.println(sum(1, 2, 3, 4));
        System.out.println(sumFirstN(3, 1, 2, 3, 4));
    }
}
```

Zero, uno o più valori double

Di fatto è un riferimento ad array

E' possibile specificare altri parametri, ma **PRIMA** dell'**UNICA** sequenza variabile di parametri

Output:
0
10
6

Array a 2 dimensioni

- Possiamo specificare un array a 2 dimensioni (o matrice) semplicemente specificando due coppie di parentesi quadre []:

```
String[][] matrice = new String[RIGHE][COLONNE];
```

- L'accesso avviene specificando le due dimensioni dell'array:

```
System.out.println(matrice[y][x]);
```

Inizializzazione di un array a 2 dimensioni

- Inizializzazione con **sintassi estesa**
- Ad es. una matrice 3x3:

```
int[][] a = new int[][] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
```

- oppure:

```
int[][] a;
…
a = new int[][] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
```

- Con **sintassi abbreviata**:

```
int[][] a = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
```

Un array bidimensionale è in realtà un array di array

- E' possibile avere righe di **lunghezza diversa**!
- Ad esempio:

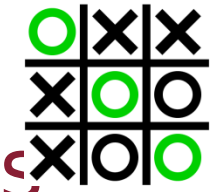
```
int[][] a = { { 1 }, { 2,3 }, { 4,5,6 } };
System.out.println(a[0].length); // stampa 1
System.out.println(a[1].length); // stampa 2
System.out.println(a[2].length); // stampa 3
```

Esercizio: la tavola pitagorica

- Scrivere una classe che rappresenti la tavola pitagorica NxN (dove l'intero N è un parametro di costruzione della classe)
- La classe deve, su richiesta, restituire il valore della tabella in corrispondenza della posizione (i, j)
- La classe deve poter stampare l'intera tavola

x	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	12	14	16	18	20
3	0	3	6	9	12	15	18	21	24	27	30
4	0	4	8	12	16	20	24	28	32	36	40
5	0	5	10	15	20	25	30	35	40	45	50
6	0	6	12	18	24	30	36	42	48	54	60
7	0	7	14	21	28	35	42	49	56	63	70
8	0	8	16	24	32	40	48	56	64	72	80
9	0	9	18	27	36	45	54	63	72	81	90
10	0	10	20	30	40	50	60	70	80	90	100

Esercizio: il gioco del tris



- Progettare una classe ScacchieraTris che implementi la scacchiera del gioco del tris
- La classe deve memorizzare la scacchiera i cui elementi possono essere:
  - " " (se non è stata ancora occupata la casella)
  - "X" oppure "O" (secondo il giocatore che ha occupato la casella)
- La classe deve stampare in qualsiasi momento la situazione della scacchiera
- Deve permettere di occupare una casella con un simbolo "X" o "O"
- Progettare quindi una classe Tris che implementi il gioco utilizzando la scacchiera appena progettata