Università degli Studi di Palermo

Corso di Laurea
Ingegneria Gestionale

Lezione 12

Gli Array

Ing. Massimo Cossentino

Sistemi Informativi Aziendali

a.a. 2008/2009

Array

- Array: sequenza di valori omogenei (cioè dello stesso tipo).
- Dichiarazione di una variabile array `double[ ]`

```
double[] data;
```

- Istanziare un array:

```
double[] data = new double[10];
```

- Il tipo di una variabile che fa riferimento a un array è il tipo dell'elemento.
- Memorizzare in una variabile il riferimento all'array.

Array

- Dichiarare e creare array:
  - Gli array sono oggetti che occupano memoria
  - Vengono creati dinamicamente attraverso la parola chiave new

```
int c[] = new int[ 12 ];
```

  - Equivalente a

```
int c[];  // dichiara una variabile array
c = new int[ 12 ]; // crea un array
```

- Possiamo anche creare array di oggetti:

```
String b[] = new String[ 100 ];
```

Array

- Nel momento in cui viene istanziato l'array, tutti i suoi valori sono inizializzati al valore
  - 0 (per un array di numeri come int[] o double[]),
  - false (per un array boolean[]),
  - null (per un array di riferimenti a oggetti).
- Nel dichiarare l'array non si specifica la dimensione che è invece obbligatoria al momento dell'instanziazione

Array

- Si accede agli elementi di un array tramite un indice di tipo intero, usando la notazione **`a[i]`**.

```
System.out.println(data[2]);
```

- I valori per gli indici di un array vanno da 0 a `length – 1`. L'accesso a un elemento non esistente provoca il lancio di un'eccezione per errori di limiti.
- Per conoscere il numero di elementi di un array usare il campo `length`.
- Gli array hanno un limite pesante: *la loro lunghezza è fissa*.

Sintassi

- Dichiarazione di array

```
 nomeTipo[] nomeVariabile;
Esempio:
 int[] arrayDiInteri;
```

- Instanziazione

```
 new nomeTipo[lunghezza]
Esempio:
 new double[10]
```

- Accesso agli elementi di un array

```
riferimentoAdArray[indice]
Esempio:
data[2]
```

Arrays

- Usare un inizializzatore di array
  - *initializer list*
    - Racchiudere gli elementi all'interno di parentesi({})
    - Separare gli elementi con virgole

```
int n[] = { 10, 20, 30, 40, 50 };
```

      - Crea un array di 5 elementi
      - Con valori dell'indice 0, 1, 2, 3, 4
  - Non c'è bisogno della parola chiave new

Esempio di uso dell'inizializzatore

```
public class InitArray
{
    private int[] vettoreInteri={10,20,30,40,50};

    public void stampaArray()
    {
        System.out.println("Indice-Valore");
        for (int i=0; i<vettoreInteri.length; i++)
       {
            System.out.println(i+" - "+vettoreInteri[i]);
       }
    }
}
```

Arrays

- Calcolare il valore da memorizzare in ogni elemento di un array
  - Inizializzare un array di 10 elementi con valori interi pari.
  - Ad esempio:
  - 2,4,6,8,...

```
public class RiempiArray
{
    private int[] vettoreInteri= new int[10];

    public void riempi()
    {
        for (int i=0; i<vettoreInteri.length; i++)
       {
           vettoreInteri[i]=2*i+2;
       }
        System.out.println(”Indice-Valore");
        for (int i=0; i<vettoreInteri.length; i++)
       {
            System.out.println(i+" - "+vettoreInteri[i]);
       }
    }
}
```

Il ciclo `for` generalizzato

- Il ciclo `for` generalizzato scandisce tutti gli elementi di una raccolta:

```
double[] data = . . .;
double sum = 0;
for (double e : data) // si legge "per ogni e in data"
{
   sum = sum + e;
}
```

Il ciclo `for` generalizzato

- Per scandire tutti gli elementi di un array non è obbligatorio utilizzare il ciclo `for` generalizzato: lo stesso ciclo può essere realizzato con un `for` normale e una variabile indice esplicita.

```
double[] data = . . .;
double sum = 0;
for (int i = 0; i < data.length; i++)
{
   double e = data[i];
   sum = sum + e;
}
```

Sintassi : Il ciclo `for` generalizzato

```
for (Tipo variabile : raccolta)
    enunciato
```

Esempio:

```
for (double e : data)
    sum = sum + e;
```

Obiettivo:

Eseguire un ciclo avente un'iterazione per ogni elemento appartenente a una raccolta. All'inizio di ciascuna iterazione viene assegnato alla variabile l'elemento successivo della raccolta, poi viene eseguito l'enunciato.

Esercizio

- Utilizzare un ciclo for generalizzato per stampare gli elementi di un array

```
public class StampaElementi
{
   private int[]
   vettoreInteri={10,20,30,40,50};

    public void stampaArray()
    {
        for (int i : vettoreInteri)
       {
            System.out.println(i);
       }
    }
}
```

Esercizio

- Scrivere un programma che legga una serie di numeri interi, li memorizzi in un array e ne calcoli la somma a segni alterni.
- Per esempio:
  - In: 1 4 9 16 9 7 4 9 11
  - Out : 1-4+9-16+9-7+4-9+11=-2

- Utilizzare un ciclo for non generalizzato

```
public class Somma_alternata
{
    private int[] vettoreInteri={1,12,4,6,7,8,10,17};

    public void somma()
    {
        int somma=0;
        for (int i=0; i<vettoreInteri.length; i++)
       {
           if ((i%2)==0)
                somma = somma+vettoreInteri[i];
            else
                somma = somma-vettoreInteri[i];
           //La somma è il risultato dell'operazione
              //1-12+4-6+7-8+10-17=-21
       }
       System.out.println("La somma cercata è: "+somma);
    }
}
```

Array a due dimensioni

- Gli array bidimensionali rappresentano una tabella, una disposizione di elementi a due dimensioni. Si accede agli elementi di un array bidimensionale usando una coppia di indici, `a[i][j]`.

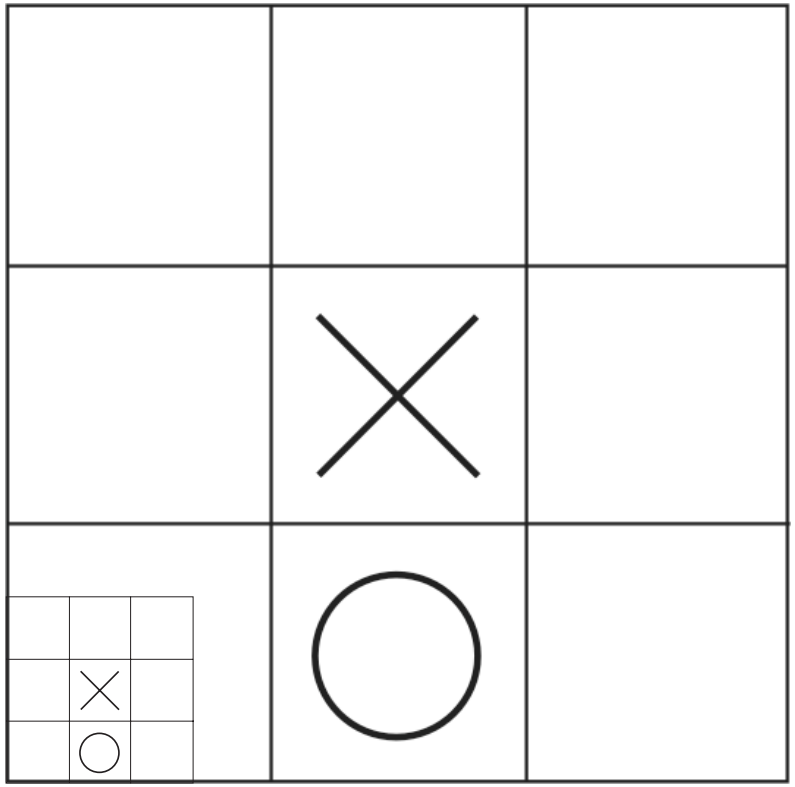
- Quando si costruisce un array bidimensionale, si deve specificare quante righe e quante colonne servono.

```
final int ROWS = 3;
final int COLUMNS = 3;
String[][] board = new String[ROWS][COLUMNS];
```

- Per accedere a un particolare elemento della matrice, si usano due indici tra parentesi quadre separate`a[i][j]`

```
board[i][j] = "x";
```

Una scacchiera per il gioco “Tic-Tac-Toe”



Array a due dimensioni

- Quando si inseriscono o si cercano dati in un array bidimensionale, di solito si usano due cicli annidati.
- Per esempio, questa coppia di cicli assegna a tutti gli elementi dell'array una stringa contenente il solo carattere di spaziatura.

```
for (int i = 0; i < ROWS; i++)
   for (int j = 0; j < COLUMNS; j++)
      board[i][j] = " ";
```

File TicTacToe.java

```
/**
    Una scacchiera 3x3 per il gioco tic-tac-toe.
 */
 public class TicTacToe
 {
    /**
       Costruisce una scacchiera vuota.
    */
    public TicTacToe()
    {
       board = new String[ROWS][COLUMNS];
       // riempi di spazi
       for (int i = 0; i < ROWS; i++)
          for (int j = 0; j < COLUMNS; j++)
             board[i][j] = " ";
    }
```

Segue

File TicTacToe.java

```
/**
   Imposta un settore della scacchiera.
      Il settore deve essere libero.
   @param i l'indice di riga
   @param j l'indice di colonna
   @param player il giocatore ("x" o "o")
*/
public void set(int i, int j, String player)
{
   if (board[i][j].equals(" "))
      board[i][j] = player;
}

/**
   Crea una rappresentazione della scacchiera in una stringa, ad esempio
   |x  o|
   |  x |
   |   o|
   @return la rappresentazione della stringa
*/
```

Segue

File TicTacToe.java

```
    public String toString()
    {
        String r = "";
        for (int i = 0; i < ROWS; i++)
        {
            r = r + "|";
            for (int j = 0; j < COLUMNS; j++)
                r = r + board[i][j];
            r = r + "|\n";
        }
        return r;
    }

    private String[][] board;
    private static final int ROWS = 3;
    private static final int COLUMNS = 3;
}
```

File TicTacToeRunner.java

```
import java.util.Scanner;

/**
   Questo programma esegue la classe TicTacToe
   chiedendo all'utente di selezionare posizioni sulla
   scacchiera e visualizzando il risultato.
*/
public class TicTacToeRunner
{
   public static void main(String[] args)
   {
      Scanner in = new Scanner(System.in);
      String player = "x";
      TicTacToe game = new TicTacToe();
      boolean done = false;
      while (!done)
      {
```

Segue

File TicTacToeRunner.java

```
            System.out.println(game.toString());
            System.out.print(
                  "Row for " + player + " (-1 to exit): ");
            int row = in.nextInt();
            if (row < 0) done = true;
            else
            {
               System.out.print("Column for " + player + ": ");
               int column = in.nextInt();
               game.set(row, column, player);
               if (player.equals("x"))
                  player = "o";
               else
                  player = "x";
            }
         }
      }
}
```

File TicTacToeRunner.java

Visualizza

```
|  |
|  |
|  |
Row for x (-1 to exit): 1

Column for x: 2

|   |

| x|

|

Row for o (-1 to exit): 0

Column for o: 0

|o |

| x|

|   |

Row for x (-1 to exit): -1
```

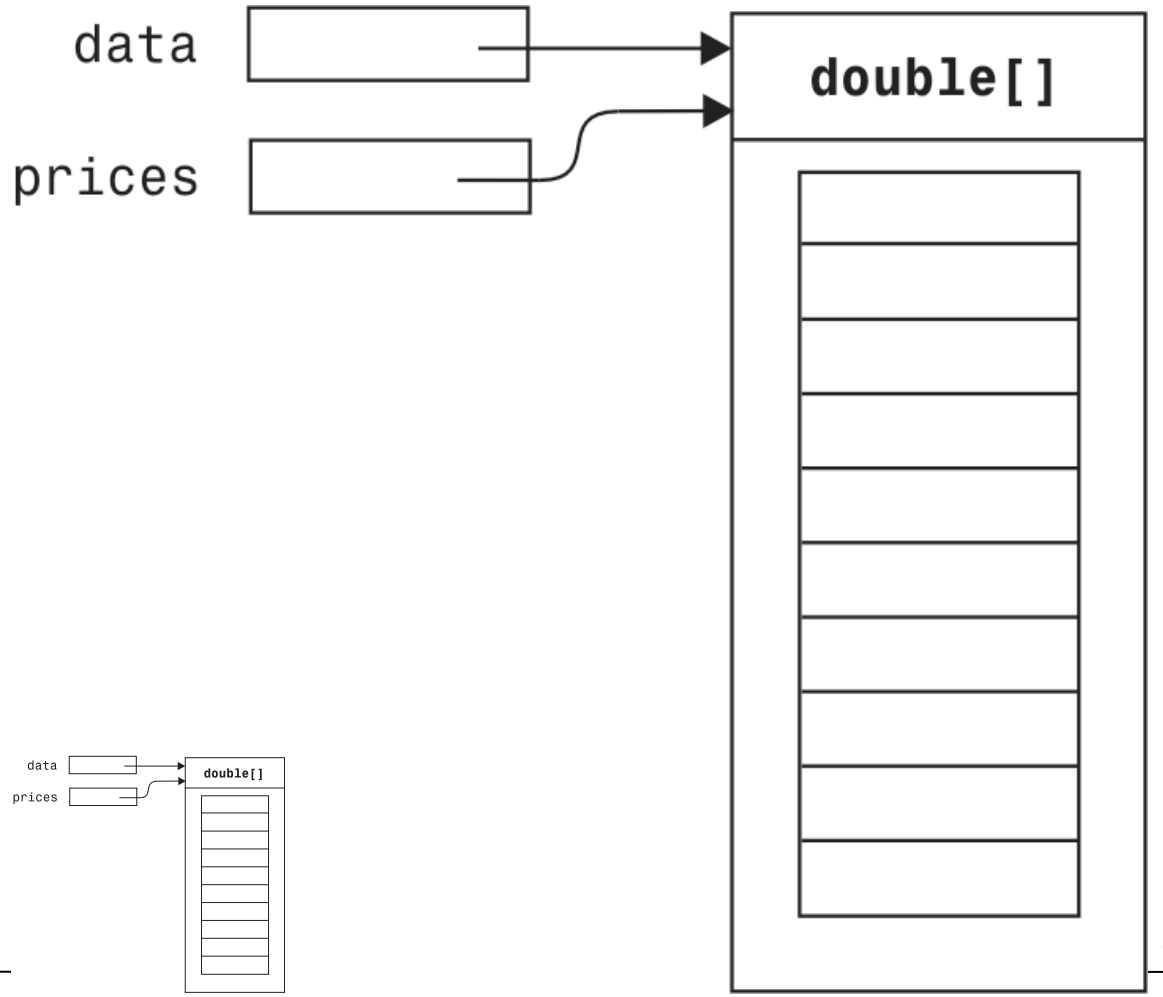
Operazioni sugli array

Copiare array:
Copiare il riferimento a un array

- Una variabile di tipo array memorizza un riferimento all'array. Copiando la variabile si ottiene un secondo riferimento al medesimo array.
- Le variabili array funzionano esattamente come le variabili oggetto: contengono un riferimento.

```
double[] data = new double[10];
 . . . // riempimento dell'array
double[] prices = data;
```

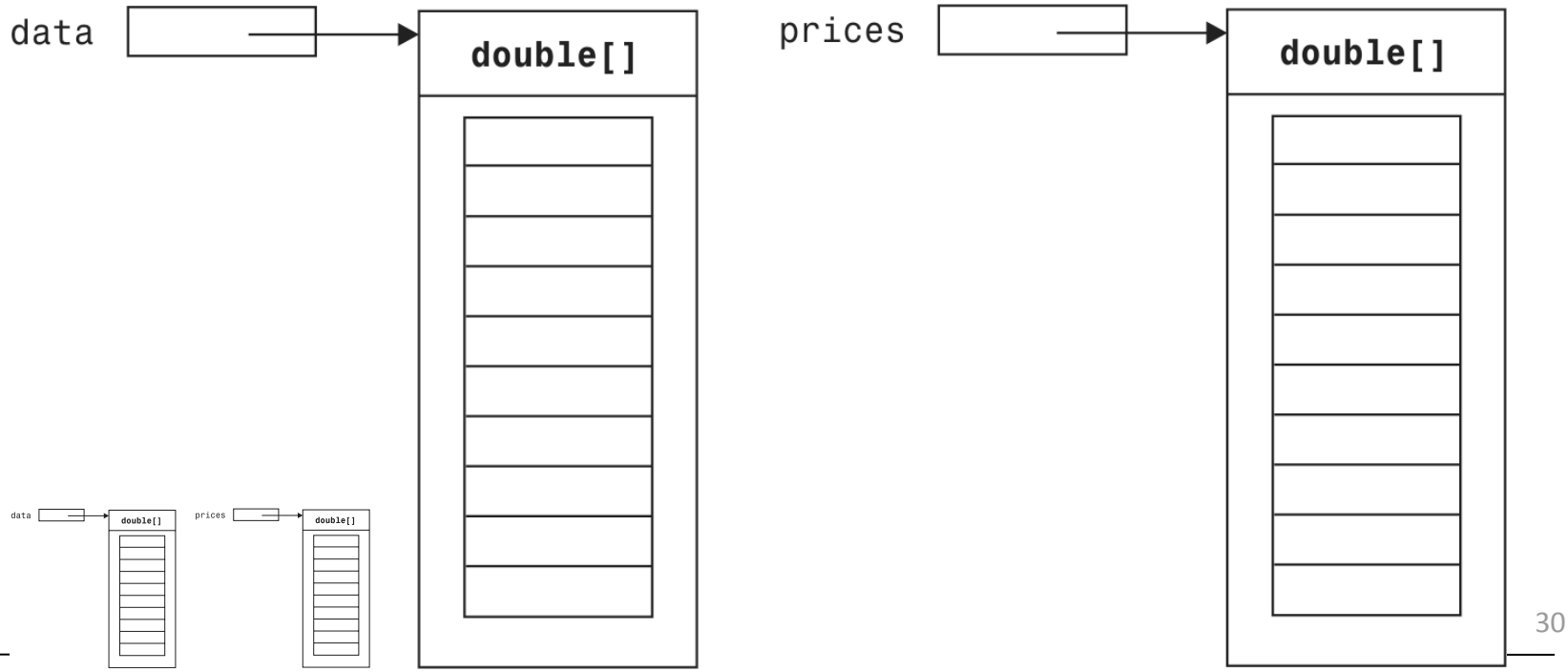
Copiare array:
Copiare il riferimento a un array



Copiare array:
Clonare un array

- Per copiare gli elementi di un array si usa il metodo clone.

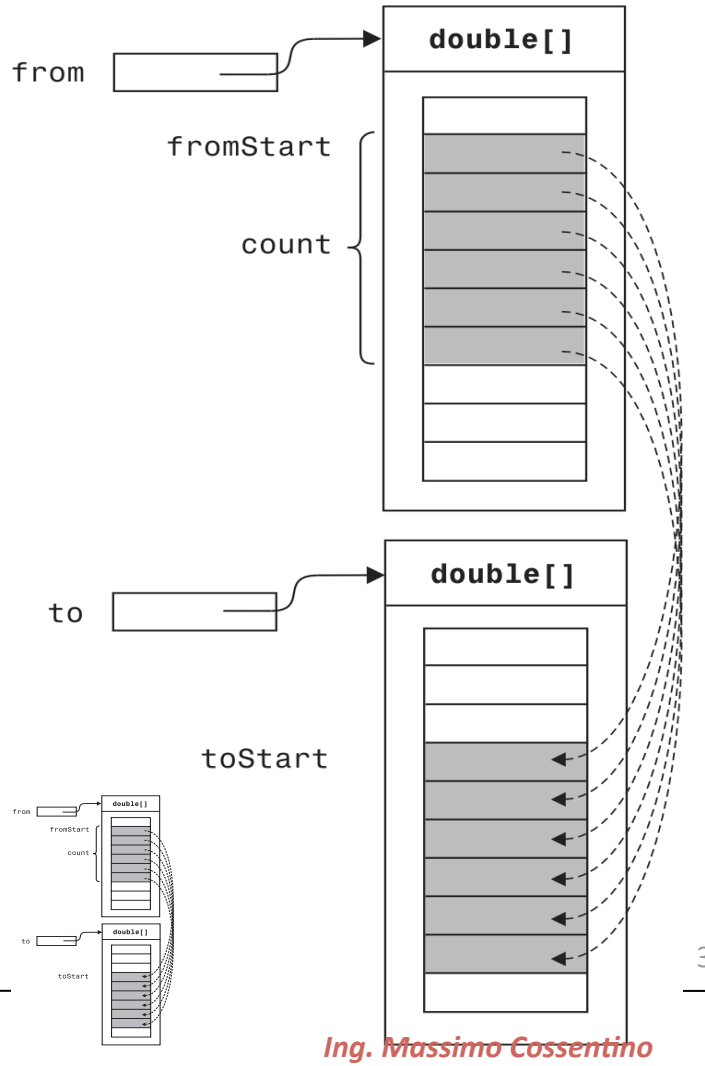
```
double[] prices = (double[]) data.clone();
```



Copiare array:
Copiare gli elementi di un array

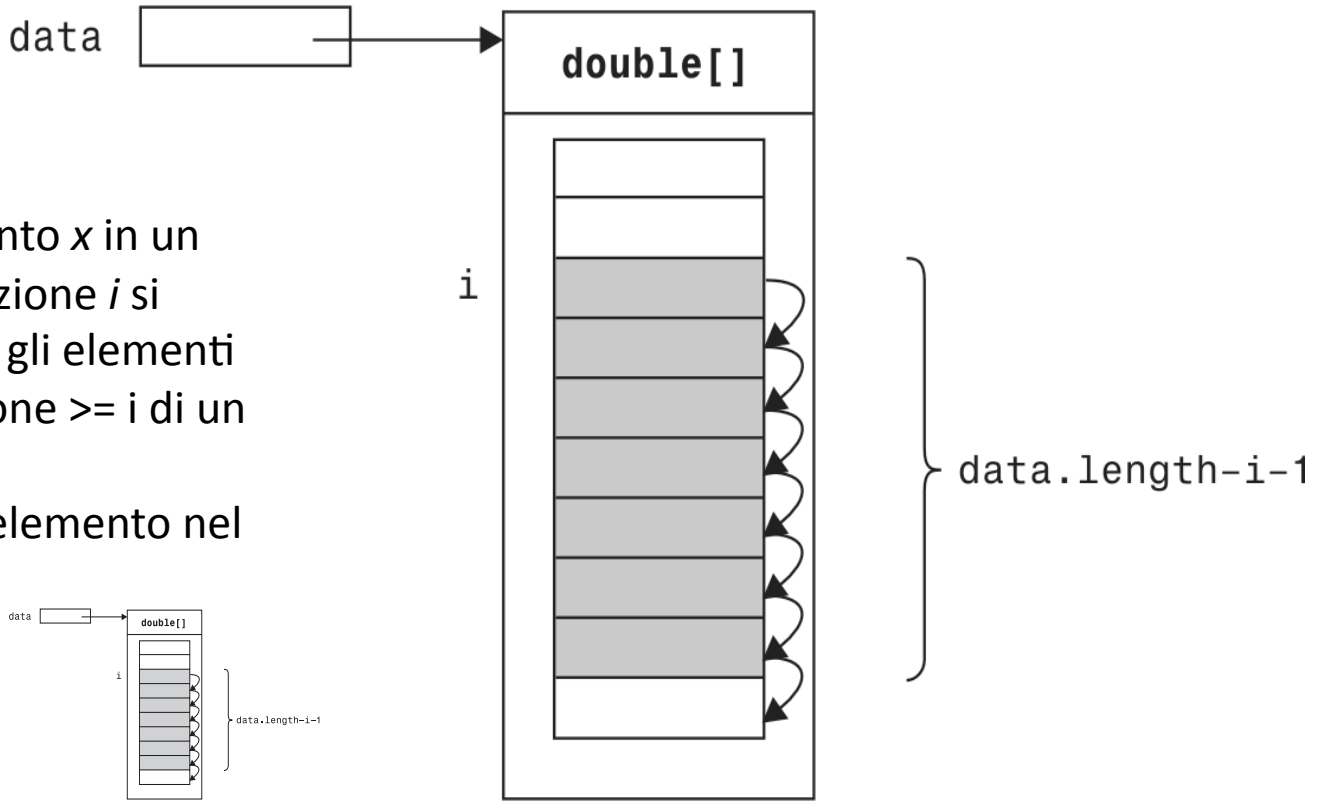
- Usate il metodo `System.arraycopy` per copiare elementi da un array a un altro.

```
System.arraycopy(
from, fromStart, to, toStart, count);
```



- from=array da cui copiare
- fromStart=indice primo elemento da copiare
- to=array in cui copiare
- toStart=indice primo elemento in cui copiare
- count=numero elementi da copiare

Inserire un elemento in un array

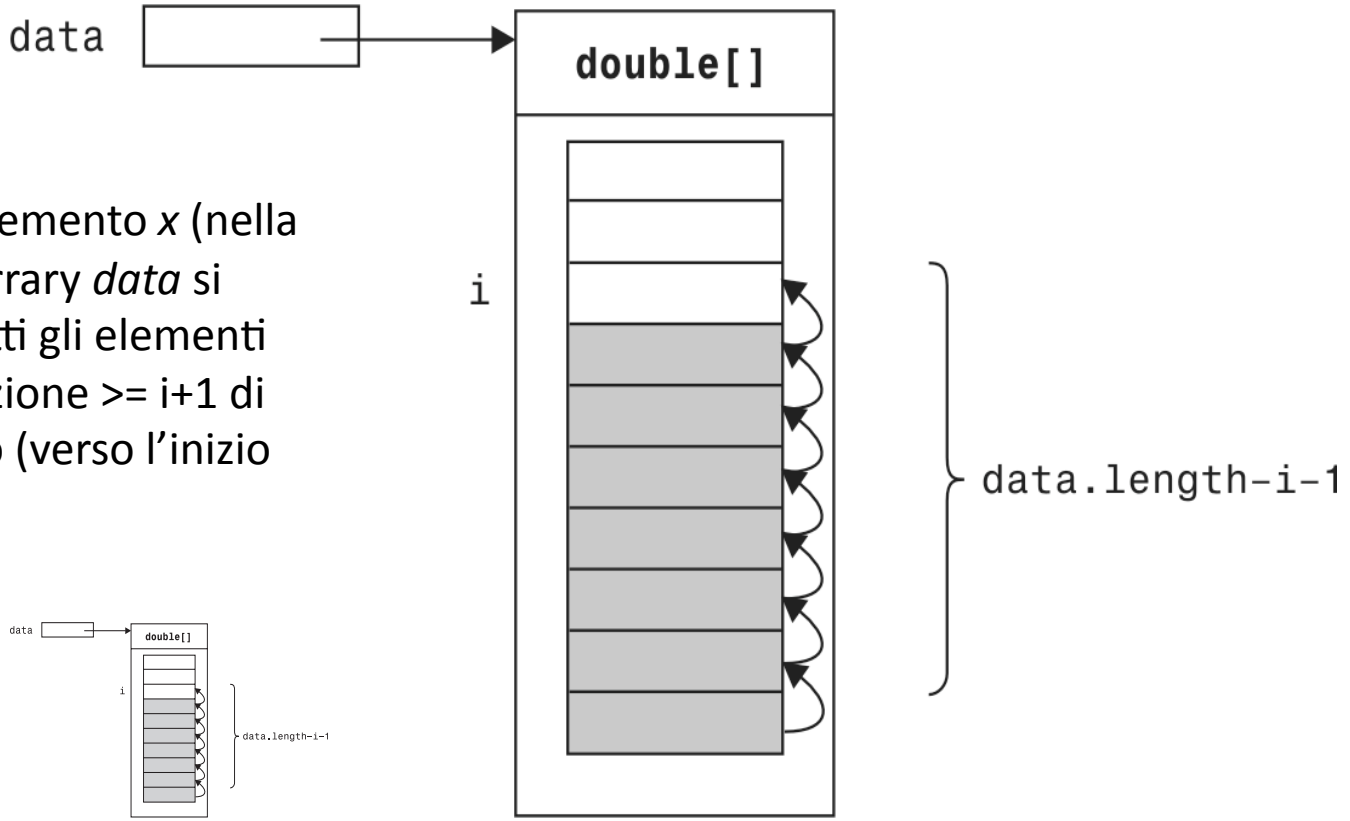


Per inserire un elemento *x* in un arrary *data* nella posizione *i* si devono spostare tutti gli elementi dell'insieme di posizione >= i di un posto.
Poi si potrà inserire l'elemento nel posto i-esimo


```
System.arraycopy(data, i, data, i + 1, data.length - i - 1);
data[i] = x;
```

Rimuovere un elemento da un array

Per rimuovere un elemento *x* (nella posizione *i*) da un arrary *data* si devono spostare tutti gli elementi dell'insieme di posizione >= i+1 di un posto all'indietro (verso l'inizio dell'array).




```
System.arraycopy(data, i + 1, data, i, data.length - i - 1);
```

Far crescere un array

- Il metodo `System.arraycopy` viene anche utilizzato per far crescere di dimensione un array che non ha più spazio, seguendo queste fasi operative:

1. Creare un nuovo array, di dimensione maggiore

```
double[] newData = new double[2 * data.length];
```

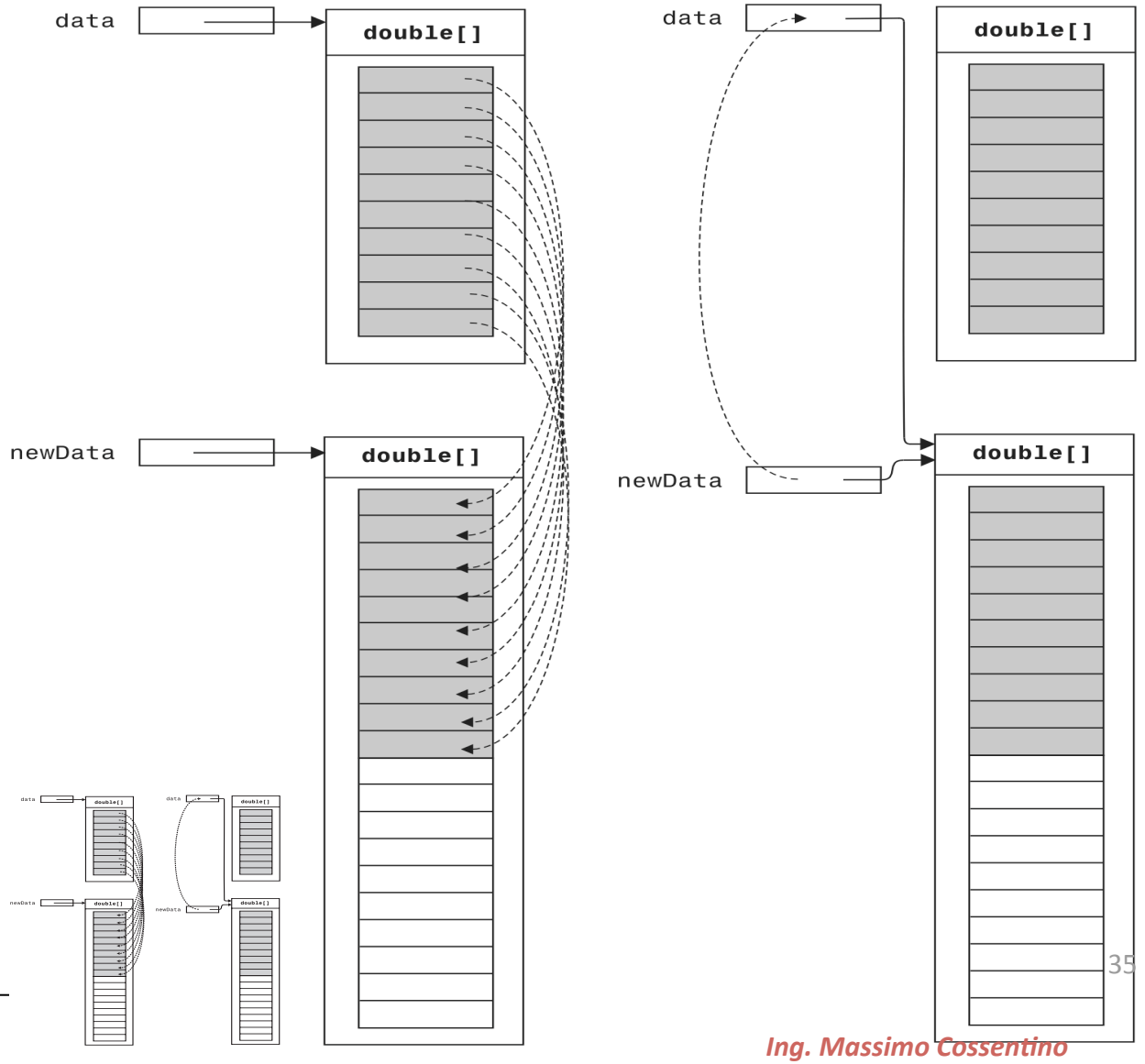
2. Copiare tutti gli elementi nel nuovo array

```
System.arraycopy(data, 0, newData, 0, data.length);
```

3. Memorizzare nella variabile array il riferimento al nuovo array

```
data = newData;
```

Far crescere un array



Array riempiti solo in parte

- La dimensione dell'array va impostata prima di sapere quanti sono gli elementi di cui si ha bisogno e non può più essere modificata.
- Si può creare un array che sia sicuramente più grande del numero massimo possibile di voci e poi riempirlo solo parzialmente.
- Usare una variabile complementare che dica quanti elementi dell'array sono realmente utilizzati.
- Assegnare sempre a tale variabile complementare un nome ottenuto aggiungendo il suffisso `Size` al nome dell'array.

```
final int DATA_LENGTH = 100;
double[] data = new double[DATA_LENGTH];
int dataSize = 0;
```

Array riempiti solo in parte

- `data.length` è la capacità dell'array `data`, mentre `dataSize` è la dimensione reale dell'array (vedasi prossima slide).
  Continuando ad aggiungere elementi all'array, bisogna incrementare di pari passo la variabile dimensione.

```
data[dataSize] = x;
dataSize++;
```

Array riempiti solo in parte