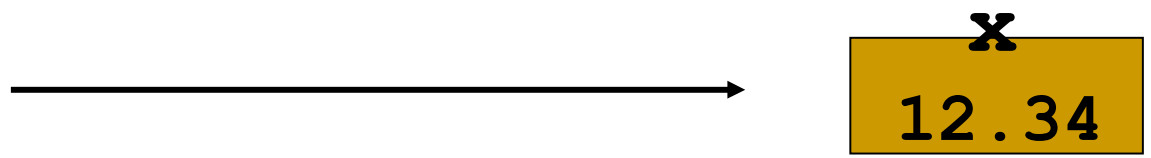
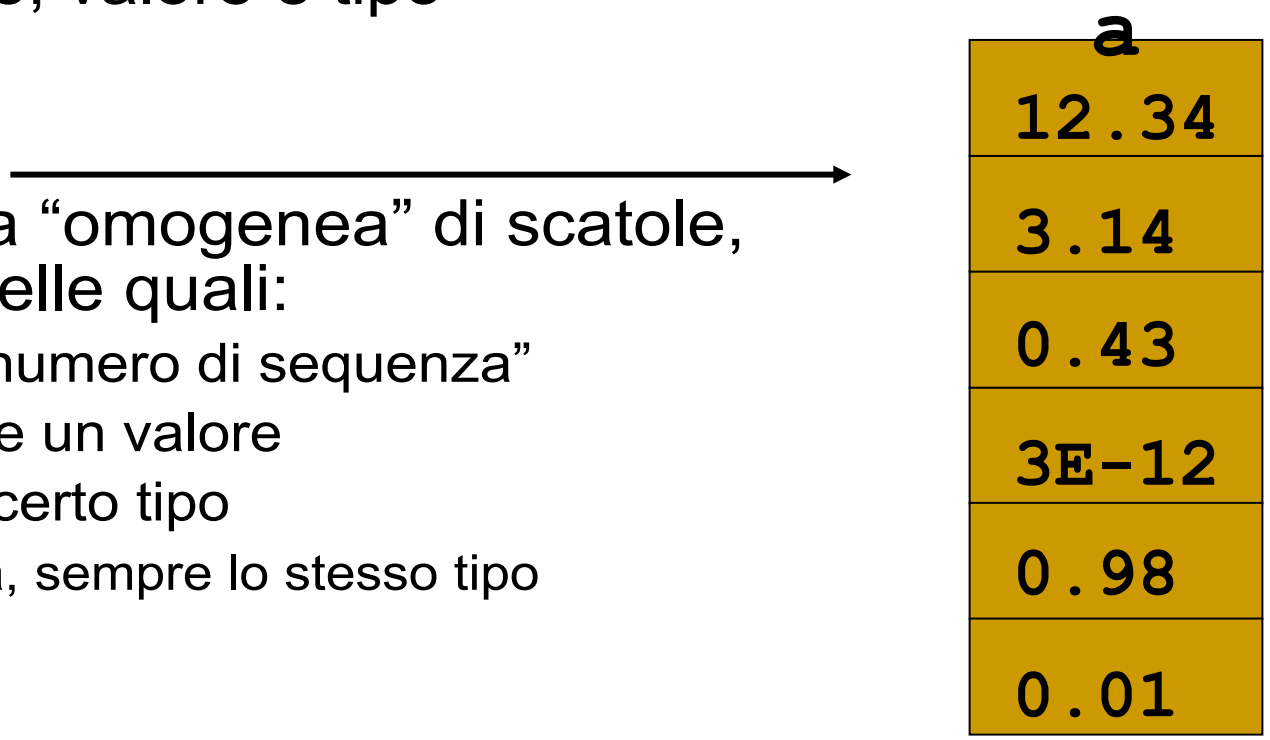
Array

Gli array

- Variabile:
  - Scatola
  - Con nome, valore e tipo

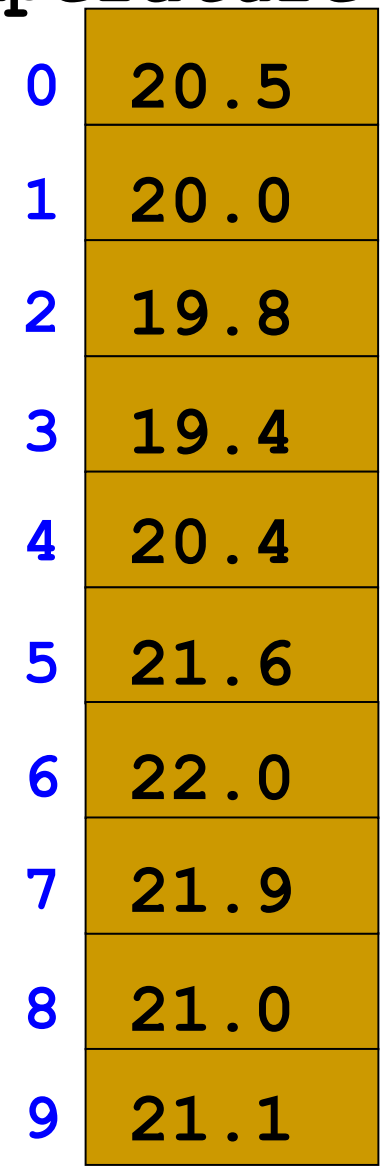


- Array:
  - Sequenza “omogenea” di scatole, ognuna delle quali:
    - Ha un “numero di sequenza”
    - Contiene un valore
    - È di un certo tipo
      - In Java, sempre lo stesso tipo



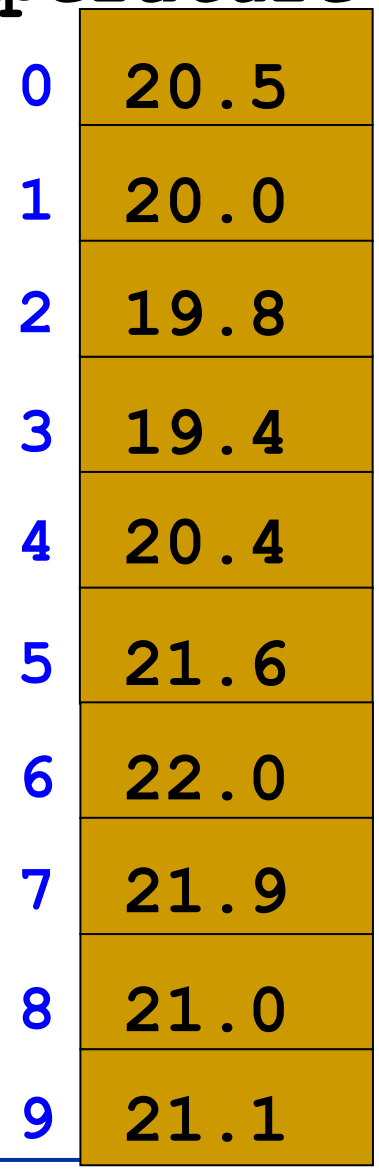
Esempio

- Programma che deve memorizzare 10 valori di temperatura
  - Invece di usare 10 variabili di tipo **`double temperatura1, temperatura2, ..., temperatura10`**
  - Uso un **array** **`temperature`** di 10 elementi di tipo **`double`**
- Posso comunque accedere ad ogni singola "scatola" tramite la posizione corrispondente (prima, seconda, …)
  - Occhio agli indici!



In Java

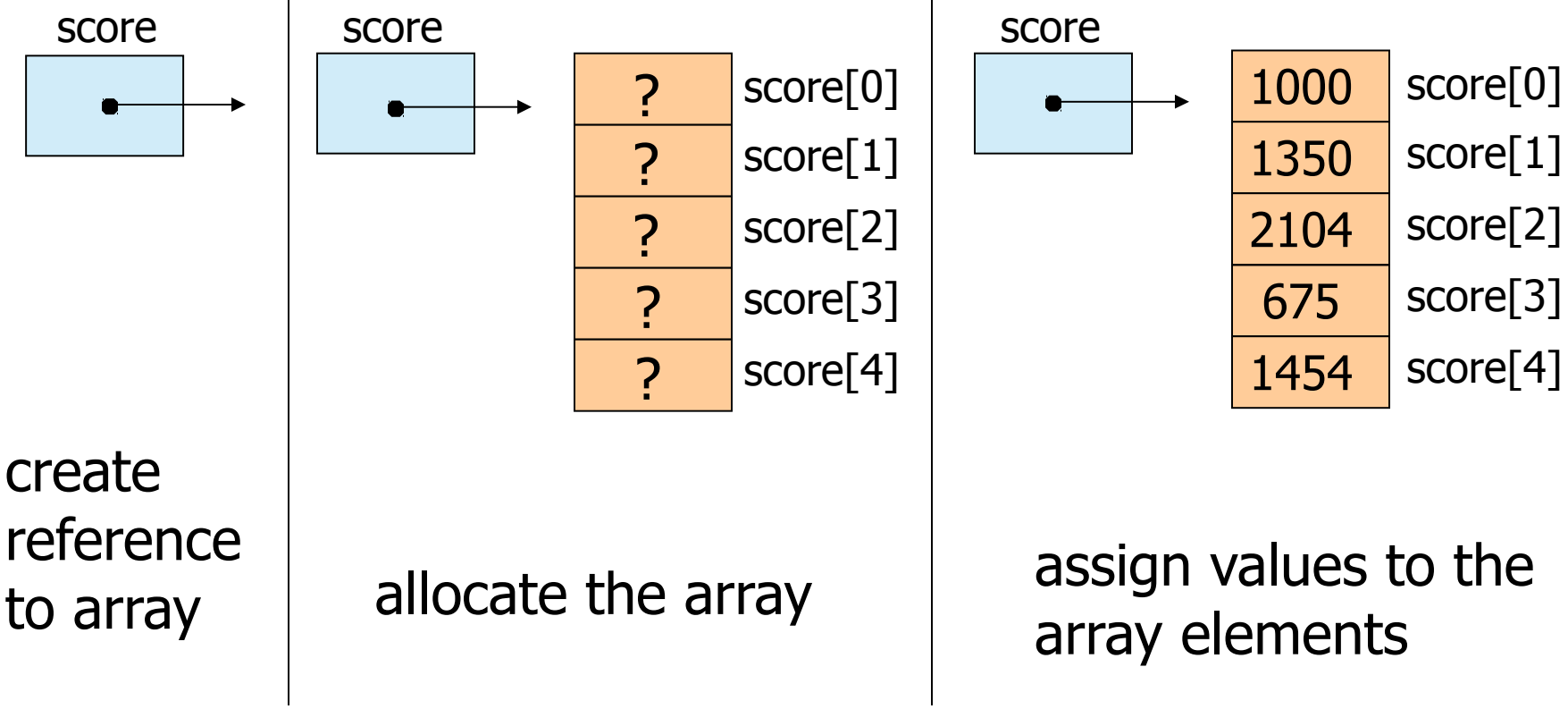
```
double[] temperature;
temperature = new double[10];
temperature[0] = 20.5;
temperature[1] = 20.0;
temperature[2] = 19.8;
temperature[3] = 19.4;
temperature[4] = 20.4;
temperature[5] = 21.6;
...
System.out.print(temperature[1]);
System.out.print(temperature[5]);
...
temperature[5] = temperature[4];
temperature[5] = temperature[5]+1;
temperature[5+1] = temperature[5];
...
```



Sintassi

- Dichiarazione
  - `tipo[] nome;`
  - Es.: `double[] temperature;`
- Allocazione
  - `nome = new tipo[lunghezza];`
  - Es.: `temperature = new double[10];`
- Uso
  - `nome[posizione]`
  - Es.: `temperature[i+j]=temperature[5];`

Rappresentazione



Vantaggio degli array

- Gestione uniforme
- Ad es., per azzerare tutte le temperature:

```
temperature[0] = 0;
temperature[1] = 0;
temperature[2] = 0;
...
temperature[9] = 0;
```

```
for (int i = 0; i < 10; i++)
  temperature[i] = 0;
```

Gestione uniforme

- Calcolare la media delle temperature

```
double[] temperature;
temperature = new double[10];
double media = 0;

...// temperature inizializzate

for (int i = 0; i < 10; i++)
  media = media + temperature[i];
media = media / 10;
```

- Implementare la media con 100 temperature è praticamente identico a farlo per 10 temperature...

Dichiarazione

- Simile alle dichiarazioni di variabili
- Si usano le `[]`
  - dopo il tipo (prefereibile... “Array di tipo…”)
  - o dopo il nome (lo eviterei...)

```
int[] a1;
int i;
char[] a2, a3;
char c;
double[] a4;
```

```
int a1[], i;
char a2[], a3[], c;
double a4[];
```

Allocazione

- Definisce lo spazio in memoria necessario per contenere gli elementi dell'array
- Si usano
  - **new**
  - **[*lunghezza*]** (numero di elementi)

```
a1 = new int[10];
a2 = new char[12];
a3 = new char[5];
a4 = new double[7];
```

- Dichiarazione e allocazione si possono fare insieme

```
int a1[] = new int[10];
char a2[] = new char[12], a3[] = new char[5];
double[] a4 = new double[7];
```

Inizializzazione

- È possibile assegnare valori iniziali a un array:

```
int[] giorniMese =
          {31,28,31,30,31,30,31,31,30,31,30,31};
```

```
int[] giorniMese;
giorniMese={31,28,31,30,31,30,31,31,30,31,30,31};
```

```
int[] giorniMese;
giorniMese =
 new int[] {31,28,31,30,31,30,31,31,30,31,30,31};
```

Ricapitoliamo con esempi

```
// Example 1: First declare a reference and then
// construct it.
int[] exampleArray1;
exampleArray1 = new int[24];

// Example 2: Short form for declaring and construction
// (probably preferable)
int[] exampleArray2 = new int[24];

// Example 3: Construct and assign an array using a
// single command.
String[] platforms = { "Sun Solaris" , "HP-UX" ,
          "Linux" , "MS Windows" , "Macintosh" };
```

Uso

- Per ora, possiamo vedere ogni posizione come una variabile
  - (si può anche lavorare su tutto un array)
  - Si usano le `[]` e un **indice** (un'*espressione* di tipo `int`)
- L'indice del primo elemento è **0** (zero)!!!

```
a1[4] = 5;
if (a1[4] > i) {
   a2[10] = 'p';
   a4[0] = a4[1];
}
```

Il `.length`

- C'è modo di sapere la lunghezza di un array (il numero di elementi)
- ***`nomearray`*`.length`**

```
char[] a = new char[12];
char[] b = {'a', 's', 'd'};

System.out.println(a.length);
System.out.println(b.length);
System.out.println(b[b.length - 2]);
System.out.println(b[a.length/6]);
```

Esempio

```
String[] platforms = {"Sun Solaris", "HP-UX",
  "Linux", "MS Windows", "Macintosh"};
for (int i = 0; i < platforms.length; i++)
  System.out.println(" - " + platforms[i]);
```

```
- Sun Solaris
- HP-UX
- Linux
- MS Windows
- Macintosh
```

Esempio

```
final int MAX = 5;
String[] platforms = new String[MAX];
platforms[0] = new String("Sun Solaris");
platforms[1] = new String("HP-UX");
platforms[2] = new String("RedHat Linux");
platforms[3] = new String("MS Windows");
platforms[4] = new String("Macintosh");

System.out.println("+----------------------+");
System.out.println("| - print platforms[] |");
System.out.println("+----------------------+");

for (int i = 0; i < MAX; i++)
  System.out.println(" - " + platforms[i]);
```

Esercizi

- (esercizio 0) Inizializzare un vettore con N valori interi arbitrari e stamparlo
- Azzerare un array **`a`** di **`int`** (ossia, assegnare zero a tutti gli elementi di **`a`**)
- Assegnare 0, 1, 2, … agli elementi di **`a`**
- Incrementare di 1 ogni elemento dell'array **`a`**
- Assegnare 1 agli elementi di **`a`** di indice dispari e 2 a quelli di indice pari
  - Provare senza usare if o espressioni condizionali

Esercizi

- Azzerare un array di **`int`** di nome **`a`**

```
for (int i = 0; i < a.length; i++)
  a[i] = 0;
```

- Assegnare 0, 1, 2, … agli elementi di **`a`**

```
for (int i = 0; i < a.length; i++)
  a[i] = i;
```

- Incrementare di 1 ogni elemento dell'array **`a`**

```
for (int i = 0; i < a.length; i++)
  a[i] = a[i] + 1; // o a[i]++, o ++a[i]
```

Esercizi

- Assegnare 1 agli elementi di **a** di indice dispari e 2 a quelli di indice pari

```
for (int i = 0; i < a.length; i++)
  if (i % 2 == 0)
    a[i] = 2;
  else
    a[i] = 1;
```

```
for (int i = 0; i < a.length; i++)
  a[i] = (i % 2 == 0) ? 2 : 1;
```

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

Osservazioni

- Il primo elemento di un array ha indice ZERO
- L'ultimo elemento di un array **`a`** ha indice **`a.length - 1`**
- Nei cicli for che "scorrono" un array **`a`**
  - **`i >= 0`**
  - **`i < a.length`**
- Cosa succede se "esco dai confini" dell'array?
  - Es. **`a[-1]`** oppure **`a[a.length]`**

Index out of bound

```
int[] a = {2,3};
System.out.println(a[1]);
System.out.println(a[a.length]);

3
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException:
Index 2 out of bounds for length 2
     at Prova.main(Prova.java:14)
```

- Il codice compila correttamente, ma si ha un errore a *runtime* (quando eseguo il codice)
  - Viene sollevata un *eccezione*

Il **`final`** e le dimensioni degli array

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

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

- Modifiche più semplici nel secondo caso

```
for (int i = 0; i < N; i++)
    ...
```

- In Java è comodo usare **`.length`**

```
for (int i = 0; i < a.length; i++)
    ...
```

Lettura Interattiva di un Array

```
…
// Read size of array and declare array

System.out.println("Enter number of elements");
int size = scan.nextInt();
double[] score = new double[size];

// read elements and store in array

for (int k = 0; k < score.length; k++) {
   System.out.println("Enter element " + k);
   score[k] = scan.nextDouble();
}
```

Esercizio 1

- Scrivere un programma Java, che riceva in ingresso una sequenza di N numeri interi. Il valore N è inserito dall'utente. I numeri dovranno essere memorizzati in un array *x*.

- Successivamente il programma crea un array y in cui vengono memorizzate le somme parziali degli elementi di x, fino all'indice considerato
  - es: se x = [1 3 4 5]
  - deve risultare y = [1 4 8 13]

Esercizio 2

- Scrivere un programma Java, che riceva in ingresso una sequenza di N numeri interi. Il valore N è inserito dall'utente. I numeri dovranno essere memorizzati in un array.

- Successivamente, il programma esegue le seguenti operazioni:
  - visualizza l'array
  - esegue uno spostamento (shift) a sinistra di una posizione del contenuto dell'array. Pertanto ogni elemento dell'array deve assumere il valore dell'elemento immediatamente successivo all'interno dell'array. L'elemento di indice N-1 deve assumere il valore zero.
    Ad esempio dato l'array: 1 10 15 18 Il programma deve trasformare l'array in: 10 15 18 0.
  - visualizza l'array ottenuto.

- Nota. Nella definizione di "destra" e "sinistra" si immagini il vettore stampato orizzontalmente, a partire dalla cella di indice 0.

Esercizio 3

- Scrivere un programma che, letto in input un numero intero *n*, stampi a video la sequenza dei numeri di Fibonacci (fino all'n-simo valore) sapendo che:
- F(1) = 1
- F(2) = 1
- se i > 2: F(i) = F(i-1) + F(i-2)

- Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, …
- Nota: costruire un array con i primi N numeri della serie di Fibonacci e stamparlo