



Java Gli array

Gli array

- In generale un array è una sequenza di locazioni di memoria, che contengono entità dello stesso tipo, e a cui si può fare riferimento con un nome comune
- Le entità che compongono un array prendono il nome di elementi dell'array
- Un elemento di un array è identificato dal nome dell'array e da un indice
- La notazione più comune, adottata anche da Java, fa uso delle parentesi quadre:

nome[indice]

- Nella maggior parte dei linguaggi (Java compreso) gli indici degli array partono da zero

Gli array in Java

- In Java gli array sono entità molto particolari: si comportano a tutti gli effetti come oggetti senza esserlo del tutto
- Infatti
 - Sono gestiti per riferimento: una variabile array è un riferimento
 - Vengono creati dinamicamente con `new`
 - Vengono distrutti automaticamente dal garbage collector quando non servono più (non esistono più riferimenti)
- Però
 - Non sono istanze di una classe: non esiste in Java una classe `Array`

Dichiarazione e creazione

- Partiamo dal caso più semplice: un array di elementi di tipo primitivo, per esempio `int`.
- Prima di tutto dichiariamo l'array:

```
int[] a;
```
- La presenza delle parentesi quadre indica che non stiamo dichiarando una variabile di tipo intero ma un array di interi
- **Attenzione:** abbiamo semplicemente creato un riferimento ad un array, non un array, infatti non è stata specificata la dimensione
- Per creare un'array usiamo l'operatore `new`, specificando la dimensione

```
a = new int[50];
```
- A questo punto abbiamo creato una array di 50 interi e `a` è un riferimento ad esso

Uso

- A questo punto possiamo accedere agli elementi dell'array come in C:

```
a[5] = 18;  
int n = a[7];
```
- Avendo creato un array di 50 elementi gli indici validi vanno da 0 a 49
- Cosa succede cerchiamo di accedere a elementi con indice al di fuori di questo intervallo?

```
a[51] o a[-1]
```
- In C avremmo avuto effetti imprevedibili perché non c'è alcuna verifica sugli indici: l'effetto è quello di sporcare la memoria
- In Java il sistema genera un errore e il programma si blocca: non è infatti possibile accedere in modo errato alla memoria.

Dimensioni

- Abbiamo visto che quando creiamo l'array, con l'operatore `new`, definiamo anche la sua dimensione
- Questo significa che possiamo dichiarare due variabili array dello stesso tipo e attribuirgli due dimensioni diverse:

```
int[] a,b;  
a = new int[50];  
b = new int[30];
```
- **Attenzione:** la dimensione di un array può essere stabilita solo al momento della creazione e non può essere più cambiata!

L'attributo length

- E' possibile conoscere la dimensione di un array usando l'attributo length:

```
int n;  
n = a.length; // n vale 50  
n = b.length; // n vale 30
```

- **Attenzione:** l'attributo length è di sola lettura, non è possibile assegnargli un valore.
- Infatti, come abbiamo detto, la dimensione di un array non può cambiare dopo la creazione.
- Quindi se scriviamo
- `a.length = 60;`
- Il compilatore darà errore

Inizializzatori

- Se vogliamo creare un array di 3 numeri reali e assegnargli un valore dobbiamo procedere così:

```
double[] a;  
a = new double[3];  
a[0] = 2.5; a[1] = 7.8; a[2] = 11.0;
```

- Per comodità Java prevede un'estensione sintattica che consente di fare tutto questo in un'unica istruzione:
- `double[] a = {2.5, 7, 8, 11.0};`
- L'elenco di numeri tra parentesi graffa prende il nome di inizializzatore e consente in un colpo solo di creare l'array, definendone la dimensione dell'array, e di attribuire i valori iniziali dei suoi elementi
- E' solo una scorciatoia, l'effetto è esattamente quello della serie di istruzioni in testa alla pagina

Array di oggetti

- Finora abbiamo visto esempi di array di tipi primitivi (interi e reali)
- Java consente di avere anche array di oggetti, o meglio array di riferimenti ad oggetti

- Possiamo quindi scrivere, per esempio:

```
Counter[] a;  
a = new Counter[4];
```

- **Attenzione:** a[] è un array di riferimenti. Creando l'array non ho creato automaticamente anche gli oggetti di tipo Counter

- Devo farlo esplicitamente:

```
a[0] = new Counter(); a[1] = new Counter();  
a[2] = new Counter(); a[3] = new Counter();
```

- O, ancor meglio:

```
for(int i=0; i<4; i++)  
    a[i] = new Counter();
```

Esempio: Orologio con gli array - 1

- Proviamo a riscrivere la classe orologio usando un array di due Counter per le ore e i minuti:

```
public class Orologio  
{  
    private Counter[] c;  
    public Orologio()  
    {  
        c = new Counter[2];  
        for (int i=0; i<2; i++)  
            c[i] = new Counter()  
        }  
    public void reset()  
    {  
        for (int i=0; i<2; i++)  
            c[i].reset();  
    }  
    ...  
}
```

Esempio: Orologio con gli array - 2

```
...
public void tic()
{
    c[0].inc();
    if (c[0].getValue() == 60)
    {
        c[0].reset();
        c[1].inc();
    }
    if (c[1].getValue() == 24)
        c[1].reset();
}
public int getOre()
{return c[1].getValue();}
public int getMinuti()
{return c[0].getValue();}
}
```

Il parametro di main

- Avendo fatto array e stringhe siamo finalmente in grado di capire il significato del parametro dei metodi main()
`public static void main(String[] args)`
- `args` è un array di stringhe che contiene gli argomenti passati sulla riga di comando quando si lancia un programma java
- Il main riceve quindi un array di String in cui ogni stringa è un argomento
- Non c'è `argc` come nei programmi C, perché la dimensione dell'array si può ricavare dall'attributo `length`
- Ogni elemento di `args` è un riferimento ad un'istanza di String

Esempio con array e stringhe - 1

- Proviamo a scrivere un programma che stampa a video gli argomenti passati dalla linea di comando

```
public class EsempioMain
{
    public static void main(String[] args)
    {
        if (args.length==0)
            System.out.println("Nessun argomento");
        else
            for (int i=0; i<args.length; i++)
                System.out.println(
                    "argomento " + i + ": " + args[i])
    }
}
```

- Da notare l'uso del + per concatenare stringhe e interi

Esempio con array e stringhe - 2

- Esempio d'uso:

```
C:> java EsempioMain 34 e 56 "aaa eee"
```

- Output:

```
argomento 0: 34
argomento 1: e
argomento 2: 56
argomento 3: aaa eee
```

- Da notare che, a differenza del C, args[0] non è il nome del programma ma il primo argomento

Esempio: la classe Set – Specifiche

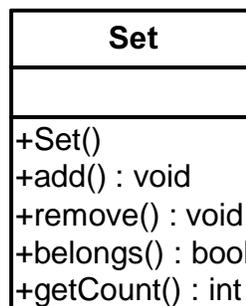
- Definiamo una classe che implementa un insieme di numeri interi, nel senso matematico del termine
- La classe si chiamerà Set (parola inglese per insieme)
- Partiamo come sempre dalle specifiche:
 - Dovrà contenere gli interi in un intervallo compreso fra zero e un numero fissato in fase di creazione
 - Ogni numero dovrà essere contenuto una volta sola
 - Dovrà consentire di aggiungere e togliere numeri
 - Dovrà consentire di stabilire se un numero appartiene o meno all'insieme
 - Dovrà consentire di sapere quanti numeri sono contenuti nell'insieme

Esempio: la classe Set – Comportamento

- Le specifiche ci portano a definire il comportamento della classe, ovvero i metodi pubblici:

```
public Set(int maxnum)
public void add(int n)
public void remove(int n)
public boolean belongs(int n)
public int getCount()
```

- Il diagramma UML sarà:

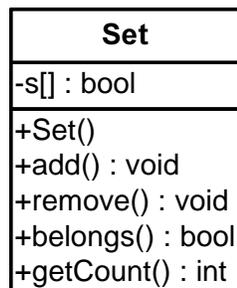


Esempio: la classe Set – Scelte implementative

- Il modo più semplice per rappresentare un insieme è quello di avere un array di boolean di dimensione pari al massimo numero ammissibile + 1
- Lo stato sarà quindi rappresentato da un solo attributo:

```
private boolean[] s;
```

- Il diagramma UML completo è quindi:



Esempio: la classe Set – Definizione

- Definiamo quindi la classe:

```
public class Set
{
    private bool[] s;
    public Set(int maxnum)
    { s = new bool[maxnum+1]; }
    public void add(int n)
    { if (n<s.length) s[n] = true; }
    public void remove(int n)
    { if (n<s.length) s[n] = false; }
    public boolean belongs(int n)
    { if (n<s.length) return false;
      else return s[n];
    }
    public int getCount()
    { int n = 0;
      for (int i=0; i<s.length; i++)
        if (s[i]==true) n++;
      return n;
    }
}
```

Esempio: la classe Set - Uso

- Definiamo una classe che implementa main e usa Set

```
public class EsempioSet
{
    public static void main(String[] args)
    {
        Set st;
        st = new Set(250);
        st.add(15);
        st.add(35);
        st.add(18);
        st.remove(35);
        System.out.println(
            "Elementi: "+st.getCount());
    }
}
```

- Il programma scriverà a video: **Elementi: 2**