

Tipo di dato = insieme di valori e insieme di operazioni

Tipi di dato in Java

Primitivi

- byte
- short
- int
- long

- float
- double

- char

- boolean

Reference

tipi di dato reference e variabili

variabili di tipi riferimento →
memorizzano i riferimenti all'area di memoria dove si trovano i valori

```
int v[] = new int [3];
v[0] = 10;
v[1] = 50;
v[2] = 30;
```

	heap						
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">v[0]</td> <td style="border: 1px solid black; padding: 2px;">v[1]</td> <td style="border: 1px solid black; padding: 2px;">v[2]</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px; text-align: center;">10</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">50</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">30</td> </tr> </table>	v[0]	v[1]	v[2]	10	50	30	
v[0]	v[1]	v[2]					
10	50	30					
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">main</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px; text-align: center;">v</td> </tr> </table>	main	v	stack				
main							
v							

tipi di dato primitivi e variabili

variabili di tipi primitivi
→ memorizzano i valori

```
int a=5;
int b=7;
```

	heap			
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">main</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px; text-align: center;">7 b</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px; text-align: center;">5 a</td> </tr> </table>	main	7 b	5 a	stack
main				
7 b				
5 a				

Variabili di tipo primitivo come parametri

- nel metodo chiamato viene effettuata una copia dei valori delle corrispondenti variabili del metodo chiamante
- il metodo chiamato può eventualmente modificare le proprie variabili locali, ma al termine dell'esecuzione del metodo chiamato i nuovi valori vengono persi, il metodo chiamato non li può utilizzare
- il metodo chiamante può eventualmente restituire un valore al metodo chiamato mediante l'istruzione return

```
public class Prova {
    public static void main(String[] args) {
        int a=2, b=3, c=4, d=5;
        int s=0;
        s = somma(a,b,c,d);
        System.out.println(s); // stampa 14
    } // fine metodo main

    static int somma(int x, int y, int w, int z) {
        int totale;
        totale = x+y+w+z;
        return totale;
    }
} //fine classe
```

variabili di tipo primitivo → un metodo lavora su una copia dei valori

tipi primitivi come parametri

Variabili di tipo reference come parametri

- nel metodo chiamato viene effettuata una copia del riferimento ai valori che sono nella heap (viene passato come valore il riferimento)
- NON viene effettuata una copia dei valori
- Il metodo chiamata può modificare i valori della heap
- Al termine dell'esecuzione del metodo chiamato il metodo chiamante può accedere ai valori modificati

```
public class Prova {
    public static void main(String[] args) {
        int v[] = {10, 50, 30, 7};
        m(v);
        System.out.println(v[0]); // stampa 6
    } // fine metodo main

    static void m(int a[]) {
        a[0] = 6;
    }
} //fine classe
```

variabili di tipo riferimento → un metodo può modificare i valori del vettore, che restano modificati anche quando si torna al main

tipi reference come parametri