

```
class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

In Java tutto il codice è dentro classi
In questo caso si tratta della classe Main, il nome può essere scelto liberamente dal programmatore
Per convenzione i nomi di classe iniziano con la lettera maiuscola
se il nome è composto allora LoStileEQuesto

La classe Main contiene un metodo

Il metodo si chiama main, si deve chiamare così il punto di ingresso per l'esecuzione del nostro programma

Java è un linguaggio case-sensitive

public: è un metodo pubblico e può essere "raggiunto" dall'esterno della classe

static: è un metodo della classe e non degli oggetti della classe

void: il metodo non restituisce alcun valore

String[] args: è l'argomento del metodo main
String[] ci dice che args è di tipo array di String (array di stringhe)

System è una classe che rappresenta l'ambiente in cui il programma viene eseguito
.out è un campo statico della classe System
.println invochiamo il metodo println sull'oggetto out
il metodo println ha un argomento di tipo stringa "Hello World"

*Salviemo il codice in un file
Main.java*

*↑ Il nome del file corrisponde al nome
della classe*

Per compilare:

\$ javac Main.java

*se non ci sono errori viene prodotto il
file oggetto Main.class che contiene
il bytecode*

Per eseguire:

\$ java Main

*↑ Solo il nome della classe,
senza .class*

In generale: java < nome della classe >

In Java ogni oggetto, ogni costante
ha un tipo

Due categorie di tipi:

- primitivi
- riferimenti

Tipi primitivi, rappresentano valori semplici

boolean	valori booleani, true e false
byte	interv., 8 bit, compl. a due
short	interv., 16 bit, compl. a due
int	interv., 32 bit, compl. a due
long	interv., 64 bit, compl. a due

short	intervi, 16 bit, Compl. a due
int	intervi, 32 bit, Compl. a due
long	intervi, 64 bit, Compl. a due
float	reali, prec. singola
double	reali, prec. doppie
char	caratteri, 16 bit, tutti i caratteri Unicode

Tipi riferimento

- array
- classe
- interfaccia

Caratteri

Unicode : ogni carattere di ogni lingua del mondo è associato a un numero detto code point

Basic Multilingual Plane
64 K caratteri

Codifica dei codepoint:

UTF-16 : ogni code point è rappresentato su 16 bit

UTF-8 : alcuni caratteri sono codificati su 8 bit
altri caratteri su 16 bit
altri ancora su 24 bit

UTF-32 : ogni carattere rappresentato su 32 bit

litterale carattere: 'a'

' \u20ac '

↑ Codice unicode
del carattere

Commenti:

// commento singola linea

```
/*
 * Commento
 * più
 * righe
 */
```

```
/***
 * Commento JavaDoc
 * più righe
 * @param ...
```

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

*/

↑

A *1

utili per produrre documentazione
usando lo strumento javadoc

"queste è una stringa"

"stringa molto" +
"molto lunga"

Definizione di variabili

tipo nomeDellaVariabile

Esempi:

int x;

float y;

double pippo;

int i1, i2, i3;

float variable-con-nome-lungo;

double variabileNomeLungo;

valori iniziali:

double d1 = 1.5;

double d2 = d1;

double d3 = d1 + 3.0;

double d4 = d2 + Math.sqrt(7.4);

Costanti:

si usa la parola chiave final

final int z = 3;

z = 5; // errore!

blank final:

final int w;

w = 3; // ok

$w = 3$; // errore

Interro: divisione per 0 genera ArithmeticException

Reale:

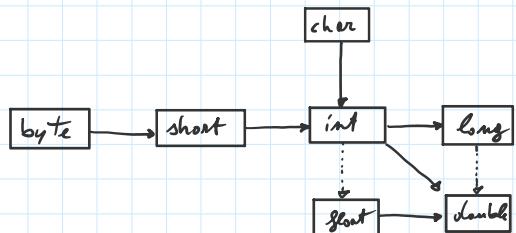
$1.1 / 0.0$ genera Double.POSITIVE_INFINITY

↑
"D maiuscola, è
la classe Double"

$0.0 / 0.0$ genera Double.NaN

Not a Number

Conversioni di tipo



→ Conversione automatica, senza perdita di precisione

---> Conversione automatica, con possibile perdita
di precisione

Attenzione: i boolean non sono convertibili
negli altri tipi e viceversa

È possibile convertire nel senso opposto
alla freccia, ma in modo esplicito
(cast)

(tipo e cast convertire)

int a = 1;

long b;

b = a; // ok conversione implicita

int c;

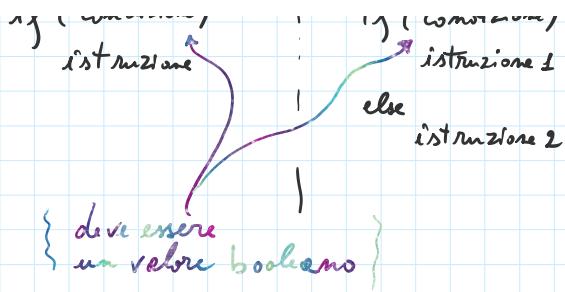
c = b; // errore

c = (int) b; // ok, cast esplicito

Istruzioni condizionali.

if (condizione)
 istruzione

: if (condizione)
 istruzione
| else



`switch`: come in C/C++

Istruzioni ripetitive

`for`: come C/C++

condizione:

`while`: " "

sempre

`do while`: " "

un booleano

C'è una versione "alternativa" del `for`
detta `for-each` o `enhanced for`:

`for (tipo variabile : set-expression)`

corpo

variabile: assume tutti i valori
identificati dalla set-expression

set-expression: definisce l'insieme di
valori da scorrere e
può essere

- array
- oggetto che implementa
l'interfaccia
`java.lang.Iterable`

tipi: tipo della variabile, deve corrispondere
al tipo degli elementi da scorrere

```

class Main {
    public static void main(String[] args) {
        int[] ar = new int[] {1, 2, 3};
        double r = media(ar);
        System.out.println(r);
    }

    static double media(int[] v) {
        if(v == null || v.length == 0)
            throw new IllegalArgumentException();
        double somma = 0.0;
        for(int valore: v) {
            somma += valore;
        }
        return somma / v.length;
    }
}
  
```

every h' 3
int

chiama il
metodo media()

restituisce
double

```

    }
}

double
"valori" assume
tutti i valori di v

```

Altro esempio:

```

public class Main {
    public static void main(String[] args) {
        // crea un array di 4 interi, con valori iniziali [0 0 0 0]
        int[] v = new int[4];
        // x assume tutti i valori contenuti in v
        for(int x: v) {
            x++; // ++ incrementa
        }
        System.out.println(v[0]);
    }
}

```

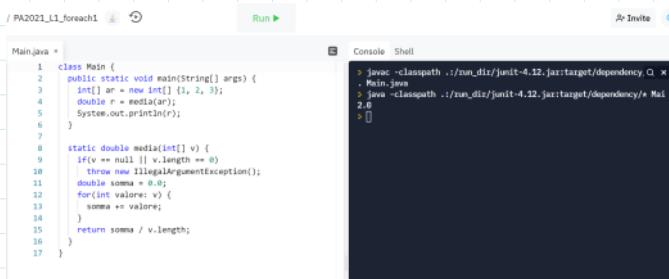
<sup>x assume il valore
di v[0], ma è
distinto da v[0]</sup>

v 

x 

Esempio eseguibile con for each

[PA2021_L1_foreach1](#)



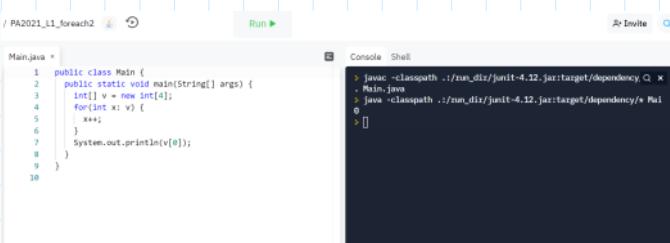
```

Main.java *
1  class Main {
2      public static void main(String[] args) {
3          int[] v = new int[] {1, 2, 3};
4          double r = media(v);
5          System.out.println(r);
6      }
7
8      static double media(int[] v) {
9          if(v == null || v.length == 0)
10              throw new IllegalArgumentException();
11          double somma = 0.0;
12          for(int valore: v) {
13              somma += valore;
14          }
15          return somma / v.length;
16      }
17  }

```

Esempio eseguibile con for each

[PA2021_L1_foreach2](#)



```

Main.java *
1  public class Main {
2      public static void main(String[] args) {
3          int[] v = new int[4];
4          for(int x: v) {
5              x++;
6          }
7          System.out.println(v[0]);
8      }
9  }

```

Ingresso/uscita con le classi `System`

A ogni programma Java in esecuzione
sono associati tre flussi:

System.in ingresso standard, tastiera

System.out uscita standard, video

System.err uscita errore, video

\$ java Main > out.txt

Per l'uscita

↓ valore
System.out.println(...)

System.out.print(...) no "a capo"

Per l'ingresso usiamo la classe Scanner

Scanner s = new Scanner(System.in);

Per crea un oggetto di tipo Scanner
che legge i dati che gli arrivano da
System.in

Alcuni metodi della classe Scanner

TIPO RESTITUITO	NOME METODO
boolean	nextBoolean()
float	nextFloat()
double	nextDouble()
int	nextInt()
String	nextLine() // intera linea
String	next() // una sola parola
boolean	hasNext() // altro da leggere?

per poter usare la classe Scanner
all'inizio del file dobbiamo scrivere

import java.util.Scanner;

Alcuni esempi con Scanner

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Primo valore? ");
        int a = sc.nextInt();
        System.out.print("Secondo valore? ");
        int b = sc.nextInt();
        int c = a + b;
        System.out.print("Risultato: ");
        System.out.println(c);
    }
}
```

```
Main.java
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args){
5         Scanner sc = new Scanner(System.in);
6         System.out.print("Primo valore? ");
7         int a = sc.nextInt();
8         System.out.print("Secondo valore? ");
9         int b = sc.nextInt();
10        int c = a + b;
11        System.out.print("Risultato: ");
12        System.out.println(c);
13    }
14}
```

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        float m = 0;
        boolean primo = true;
        Scanner sc = new Scanner(System.in);
        System.out.println("Inserisci i valori (ctrl-d per terminare)");
        while(sc.hasNext()) {
            float v = sc.nextFloat();
            if(primo) {
                m = v;
                primo = false;
            } else if(v>m)
                m = v;
        }

        if(primo)
            System.out.println("Nessun valore introdotto");
        else {
            System.out.print("Il massimo è: ");
            System.out.println(m);
        }
    }
}
```

PA2021_L1_max

```
Main.java
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args){
5         float m = 0;
6         boolean primo = true;
7         Scanner sc = new Scanner(System.in);
8         System.out.println("Inserisci i valori (ctrl-d per terminare)");
9         while(sc.hasNext()) {
10            float v = sc.nextFloat();
11            if(primo) {
12                m = v;
13                primo = false;
14            } else if(v>m)
15                m = v;
16        }

17        if(primo)
18            System.out.println("Nessun valore introdotto");
19        else {
20            System.out.print("Il massimo è: ");
21            System.out.println(m);
22        }
23    }
24}
```

Class'

In Jave tutto il codice è dentro class

Una classe contiene

- la definizione della struttura degli oggetti della classe attraverso le variabili istanza
- meccanismi per la costruzione di oggetti (i costruttori)
- codice che "manipola" lo stato

oggetti (i costruttori)

- codice che "manipola" lo stato degli oggetti (i metodi)

Esempio, una classe che rappresenta punti in un piano

```
class Punto {  
    double x;  
    double y;  
}  
  
Punto (double a, double b){  
    x=a;  
    y=b;  
}  
  
double distanza(){  
    return Math.sqrt(x*x + y*y);  
}  
  
void trasla (double t){  
    x+=t;  
    y+=t;  
}
```

x e y sono le variabili istanza

definizione della struttura degli oggetti di tipo Punto

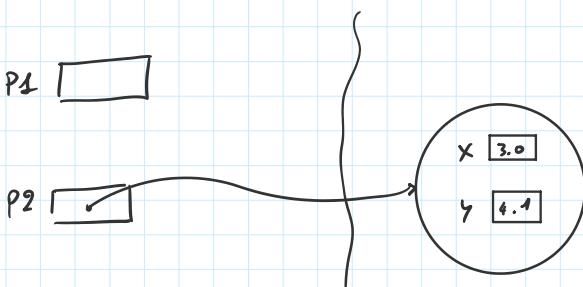
↳ Costruttore

metodo

Punto p1; ↗
p1 è un riferimento a oggetto di tipo Punto

Non ci sono ancora oggetti di tipo Punto!

Punto p2 = new Punto(3.0, 4.1);
↑
crea un nuovo oggetto di tipo Punto

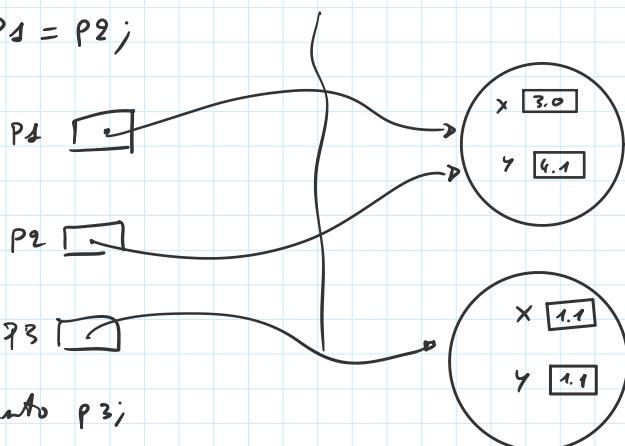


Tutti gli oggetti sono in una zona di memoria
HEAP

oltre HEAP

Se scriviamo

`P1 = P2;`



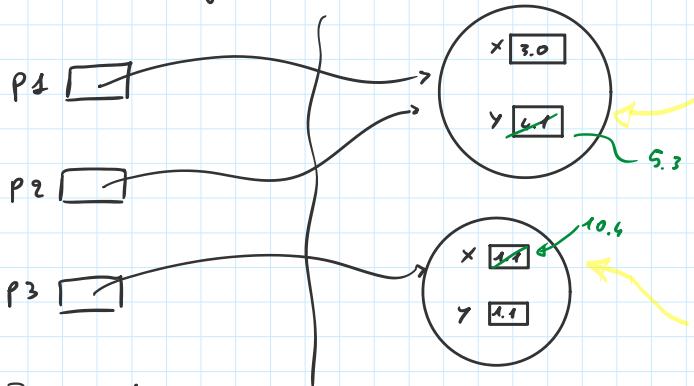
Punto p3;

`P3 = new Punto(1.1, 1.1);`

Per accedere alle variabili istanza e per invocare metodi si usa la notazione a punto (dot notation)

nomeDelRiferimento. nomeDellaVariabileIstanza

nomeDelRiferimento. nomeDelMetodo()



`P3.x = 10.4;`

`P1.y = 5.3;`

`double a = P2.y; // a vale 5.3`

`double d = P3.distanza();`

`P1.trash(3.5);`

Esempio di classe più complessa

```
class A {  
    double x;  
    int y;  
    char w;  
    AltraClasse z; questo è un riferimento  
};
```

3

A a1;

a1 = new A();

→ a1.z = new AlterClass()

