Metodologie di Programmazione (M-Z)

II semestre - a.a. 2024 - 2025

Parte 1 - Introduzione**

a cura di Stefano Faralli*

SAPIENZA
UNIVERSITÀ DI ROMA

*Tutti i diritti relativi al presente materiale didattico ed al suo contenuto sono riservati a Sapienza e ai suoi autori (o docenti che lo hanno prodotto). È consentito l'uso personale dello stesso da parte dello studente a fini di studio. Ne è vietata nel modo più assoluto la diffusione, duplicazione, cessione, trasmissione, distribuzione a terzi o al pubblico pena le sanzioni applicabili per legge.

**I crediti sulle slide di questo corso sono riportati nell'ultima slide

La programmazione

La programmazione

- **Molto più semplice che apprendere una lingua straniera**
- Poche parole chiave da imparare (vs. migliaia di parole)
- Portabilità del bagaglio: la maggior parte di ciò che imparate in Java, potete applicarlo in Python, C, in C++ o nella maggior parte degli altri linguaggi di programmazione
- Come vi sentite in un paese straniero senza poter parlare correttamente?
- E' fondamentale apprendere a programmare (a oggetti), non (solo) apprendere Java:
  - sapere quali dettagli sono rilevanti in quale situazione
  - modellare la realtà mediante l'utilizzo di oggetti
  - astrarre e generalizzare per poter riutilizzare

La programmazione

- **Da dove venite…**
- Paradigma di programmazione procedurale visto attraverso il linguaggio Python (o C)
- Un tipo di programmazione imperativa in cui il programma è costituito da una o più procedure (funzioni)
- Tipi di base, costanti, espressioni, variabili, operatori, istruzioni di selezione (if) e iterative (for, while, do), array, funzioni, ricorsione, strutture, stringhe, puntatori, input/output, ecc.

La programmazione

- **Dove stiamo andando: grado di astrazione**
- Portare il grado di astrazione **al prossimo livello:**
  - In fondo anche il linguaggio C aumenta il grado di astrazione del linguaggio assembly (mov eax, ebx; jmp 0x00a30f0e)
  - Che a sua volta aumenta quello del linguaggio macchina

    (0100111001110111)

…

La programmazione

…

- La programmazione orientata agli oggetti fornisce nuovi strumenti per rappresentare elementi nello spazio del problema (qualsiasi esso sia!)
  - Gli elementi sono chiamati oggetti
  - Puoi descrivere il problema in termini del problema, non in termini del computer su cui gira il programma

La programmazione ad oggetti

Tutto è un oggetto

- Un oggetto è un po' come un piccolo computer
  - Ha uno stato
  - Puoi "farci delle cose" (= ha delle operazioni che puoi chiedergli di eseguire)

Esempio:

- Modellare una lampadina in una stanza
  - Stato: accesa vs. spenta
  - Operazioni: collega, scollega

La programmazione ad oggetti

Tutto è un oggetto

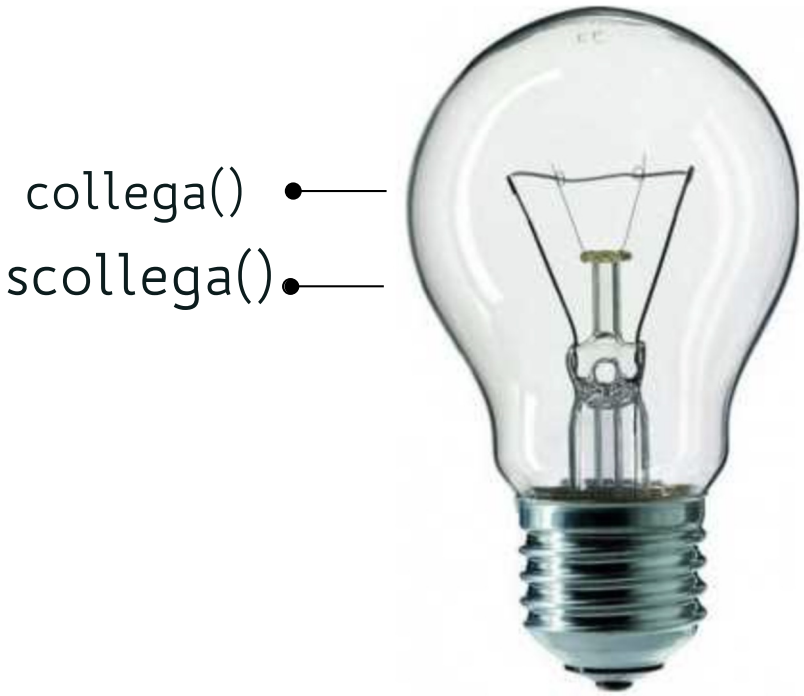
- Un oggetto è un po' come un piccolo computer
  - Ha uno stato
  - Puoi "farci delle cose" (= ha delle operazioni che puoi chiedergli di eseguire)

Esempio:

- Modellare una lampadina in una stanza
- Modellare un personaggio in un ambiente simulato

La programmazione ad oggetti

- Per effettuare una richiesta a un oggetto, si invia un messaggio a quell'oggetto
- Un messaggio è una richiesta di chiamata a una funzione (metodo) che appartiene a un particolare oggetto



La programmazione ad oggetti

Ogni oggetto ha una propria memoria "fatta" di altri oggetti (incapsulamento e information hiding)

- Un nuovo tipo di oggetto può essere creato utilizzando oggetti esistenti
- Un programma può nascondere la sua complessità mediante la semplicità degli oggetti

La programmazione ad oggetti

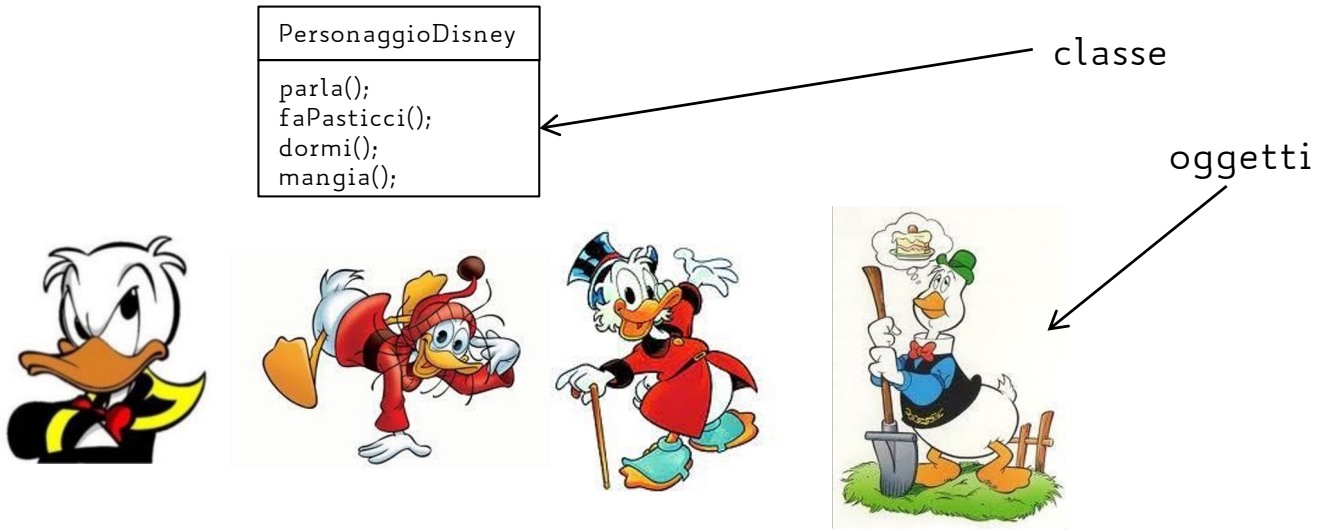
Ogni oggetto ha un suo tipo (detto classe)

Ogni oggetto è istanza di una classe

La classe è identificata dai messaggi (metodi) che essa possiede

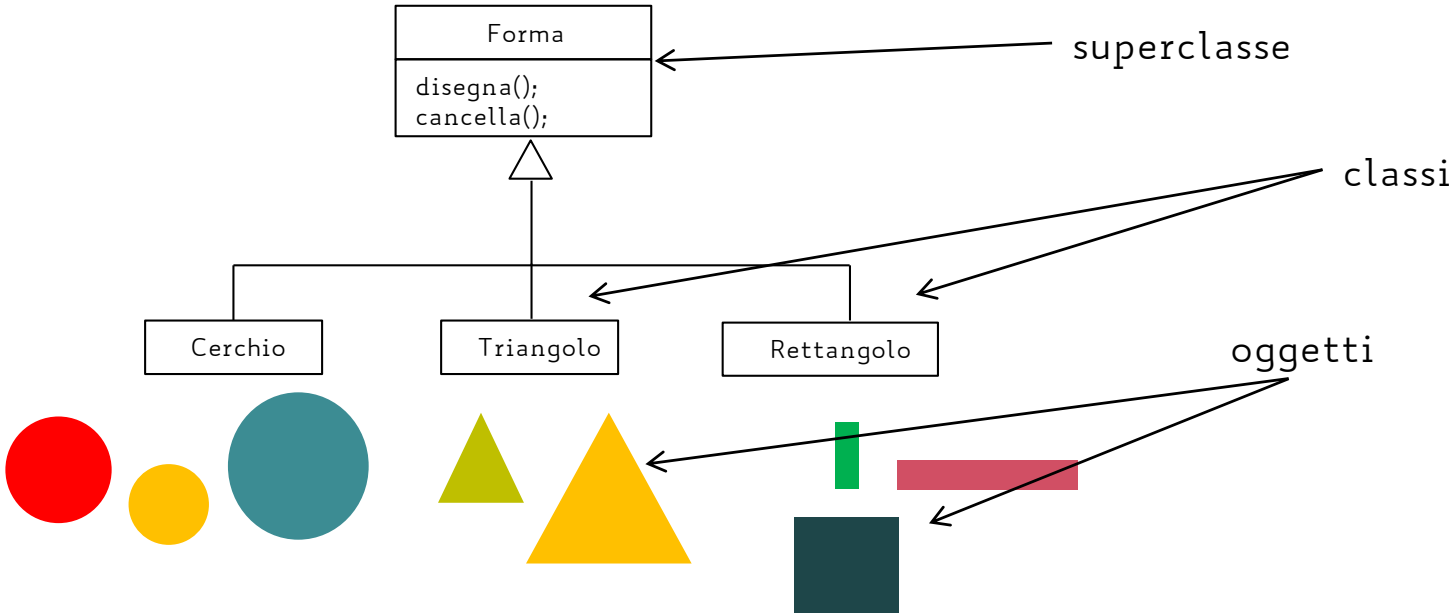
La programmazione ad oggetti

Tutti gli oggetti di uno stesso tipo possono ricevere gli stessi messaggi



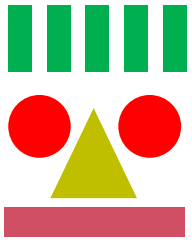
Ereditarietà

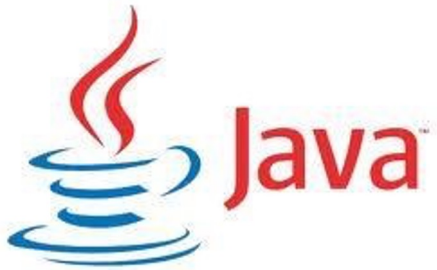
Vogliamo evitare di ricreare nuove classi di oggetti quando esse hanno funzionalità simili



Polimorfismo

- E' possibile utilizzare la classe base, senza dover conoscere necessariamente la classe specifica di un oggetto
- Permette di scrivere codice che non dipende dalla classe specifica
- Posso aggiungere nuove sottoclassi anche in seguito!
- Ad esempio, una **ImmagineVettoriale** è una collezione di oggetti di tipo **Forma** in determinate posizioni:




Java

Perché Java come linguaggio di riferimento?

Un linguaggio di programmazione potente, orientato agli oggetti

- Creato da James Gosling e altri informatici di Sun Microsystems (ora Oracle)

- Relativamente recente (1995) – Python risale al 1991!
  - Precursori: Smalltalk (fine '70), C++ (inizio '80)

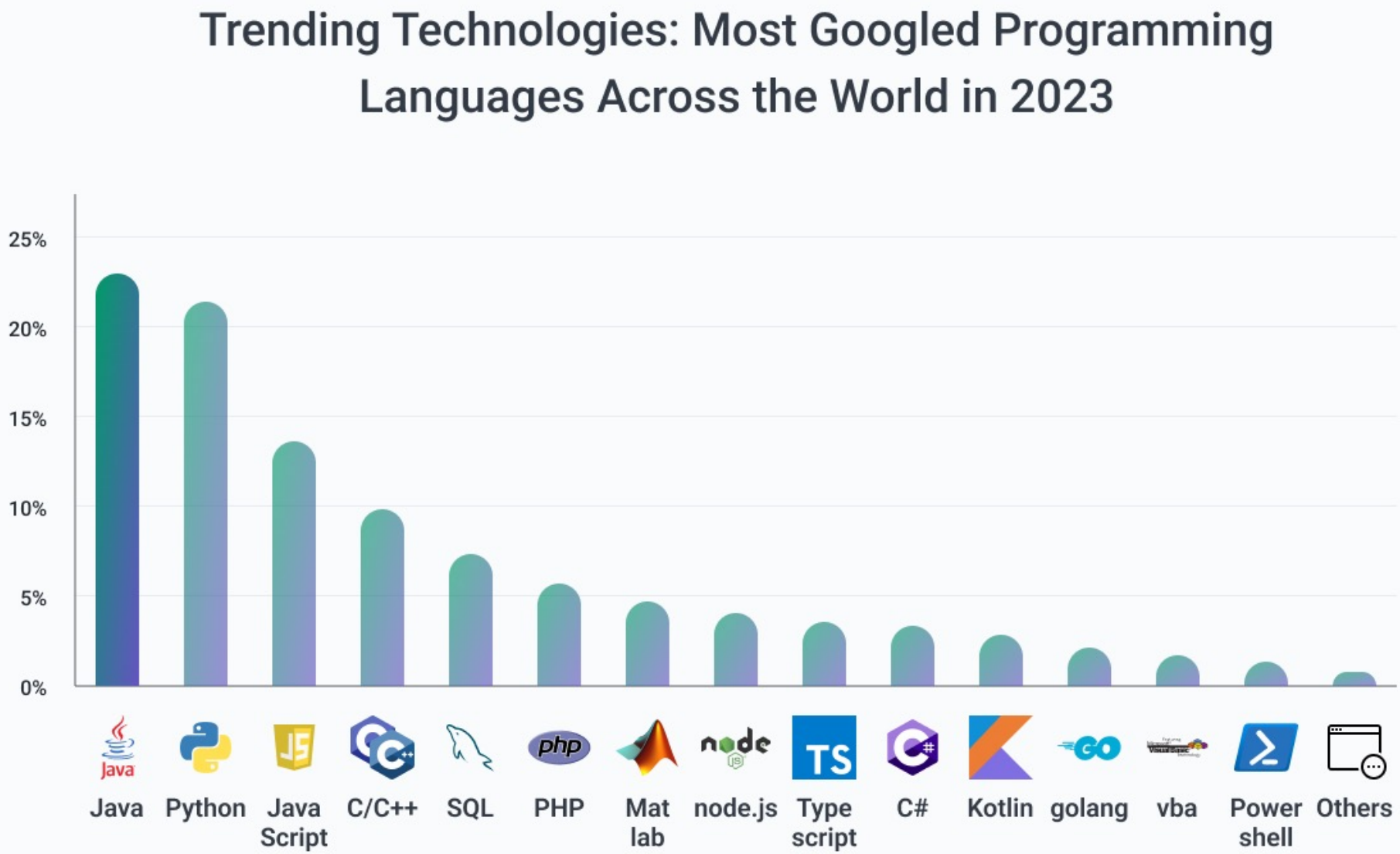
Costruito per essere "sicuro", cross-platform e internazionale

Perché Java come linguaggio di riferimento?

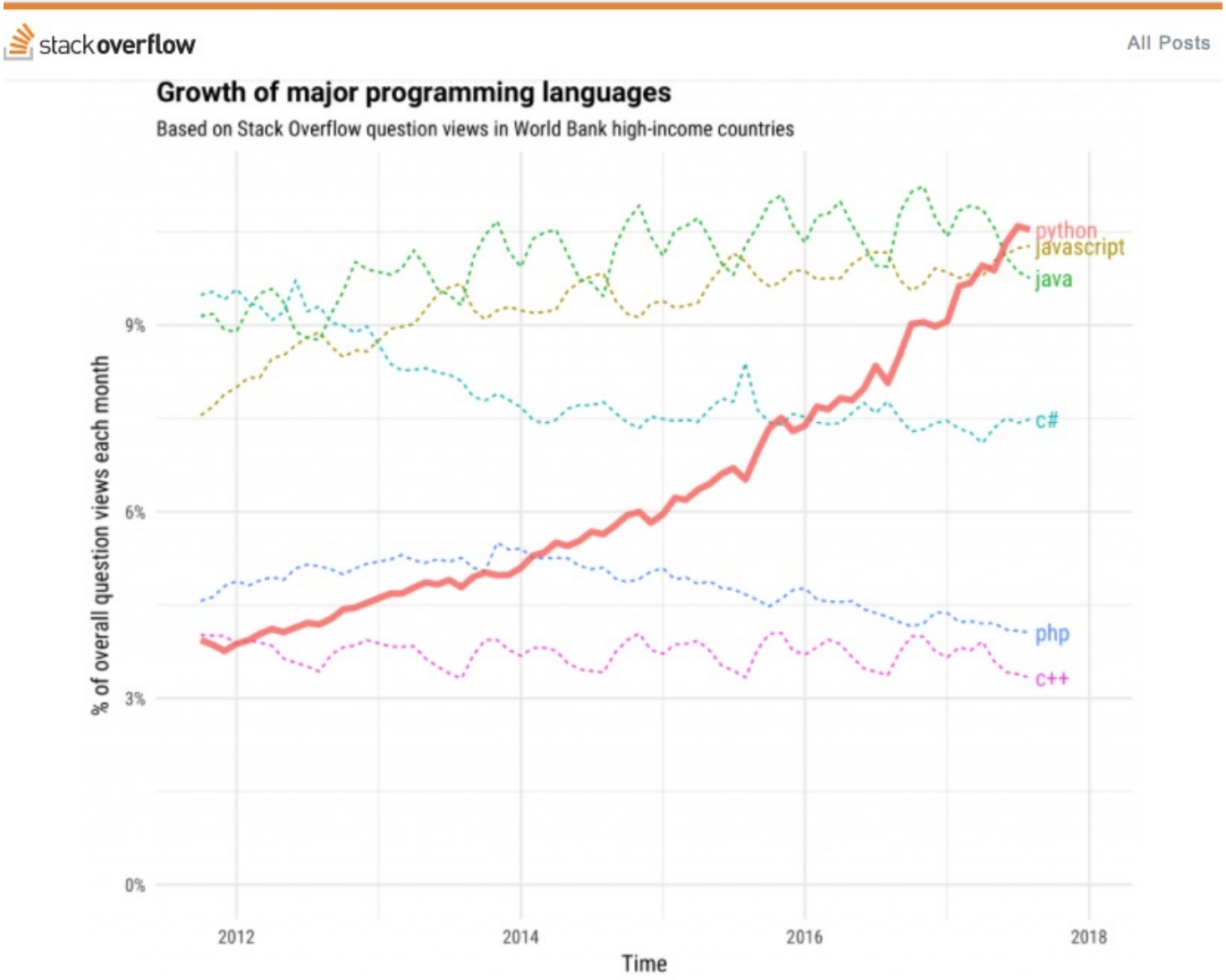
- Continuamente aggiornato con nuove feature e librerie:
  - Concorrenza
  - Accesso alle basi di dati
  - Programmazione di rete
  - Calcolo distribuito

- Portabile: "WORA" (write once, run anywhere)

Perché Java come linguaggio di riferimento?

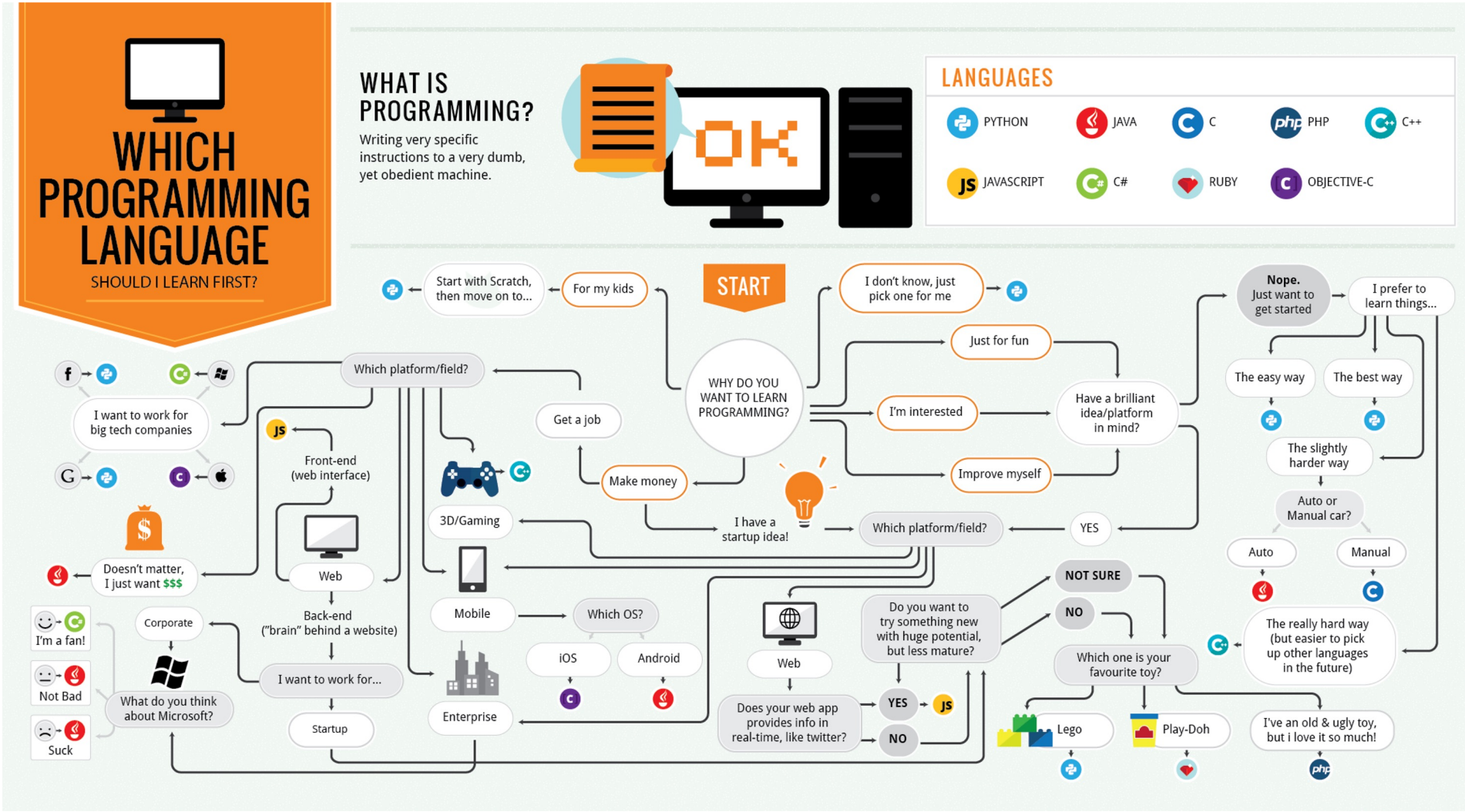


Perché Java come linguaggio di riferimento?



i

Perché Java come linguaggio di riferimento



http://carlcheo.com/startcoding

Perché Java come linguaggio di riferimento?

< Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program. >

– Linus Torvalds (creator of Linux)

Perché Java come linguaggio di riferimento?

Adozione su larghissima scala

E' il linguaggio delle applicazioni di business

- ecommerce (eBay, Amazon)
- streaming (Netflix)
- portali
- finanza
- sistemi di scambio/commercio
- online banking / pagamenti online (paypal)
- telco
- assicurazioni
- Flickr, IMDB, Walmart, Linkedin e molti altri

Google usa Java in gmail, Google+

Perché Java come linguaggio di riferimento?

- Android SDK usa(/va) Java come "cittadino di prima classe"
- Big data: Hadoop

Caratteristiche di Java (1)

- **Indipendenza dalla piattaforma:**
  - Portabile: "WORA" (write once, run anywhere)
  - Al contrario di linguaggi come il C o il C++ non viene compilato su una macchina o piattaforma, ma nel bytecode di una macchina virtuale – **che è anche meglio!**
  - Neutro rispetto all'architettura sottostante (architecture-neutral)
- **Sicurezza:**
  - Non permette manomissioni
  - Le tecniche di autenticazione sono basate su codifiche con chiavi pubbliche

Caratteristiche di Java (2)

- **Robustezza:**
  - Situazioni tipiche d'errore vengono eliminate il più possibile a tempo di compilazione
  - Laddove non possibile, gestite a tempo di esecuzione con appositi controlli

Caratteristiche di Java (3)

Multithreaded:

- Supporta nativamente programmi che gestiscono attività eseguite in contemporanea (thread)
- Facilita la costruzione di applicazioni interattive

Interpretato e compilato:

- Il byte code è tradotto "al volo" in istruzioni macchina native
- Rende più veloce e snello il processo di sviluppo

Caratteristiche di Java (4)

Alte prestazioni:

- Con l'uso dei compilatori Just-In-Time (JIT), le prestazioni sono le stesse se non addirittura SUPERIORI del codice nativo

Distribuito:

- Progettato per ambienti distribuiti come Internet

Dinamico

- Si adatta a un ambiente in evoluzione
- Porta con sé parecchie informazioni a tempo di esecuzione per verificare e risolvere gli accessi agli oggetti

Caratteristiche di Java (5)

Enterprise, Web e Mobile

- E' utilizzato a livello enterprise, web e mobile per applicazioni robuste, solide, sicure e distribuite

Installare JDK

- Installiamo il JDK
  https://www.oracle.com/it/java/technologies/downloads/

Tipi di dato di base (o primitivi)

Tipi di dato di base (o primitivi)

I tipi di dati di base sono built-in, ovvero sono predefiniti nel linguaggio

E' **fondamentale** essere a conoscenza di quali siano i tipi di base e quali non lo siano

Per ragioni di efficienza e di allocazione della memoria

Tipi di dato di base (o primitivi)

Un tipo di dati è un insieme di valori e di operazioni definite su tali valori:

- Interi (es. 27)
- Reali (es. 27.5)
- Booleani (true o false)
- Caratteri (es. 'a')
- Stringhe (es. "questa non è una stringa") – non è realmente "primitivo"

Principali tipi di dato di base

Tipo	Dominio	Operatori	Esempio
int	interi	+ - * / %	27 + 1
double	Numeri in virgola mobile	+ - * / %	3.14 * 5.01e23
boolean	valori booleani	&& || !	true || false
char	caratteri	+ -	‘a’
String	Sequenze di caratteri	+	“Hello”+ “ World”

ATTENZIONE:
String non è realmente un tipo primitivo

Tutti i tipi built-in

Tipo	Intervallo	Dimensione
double	parte intera: ±10308, parte frazionaria: circa 15 cifre decimali significative	8 byte
long	-9223372036854775808…9223372036854775807	8 byte
int	-2147483648…2147483647	4 byte
float	parte intera: ±1038, parte frazionaria: circa 7 cifre decimali significative	4 byte
boolean	true, false	1 byte
char	Rappresenta tutti i caratteri codificati con Unicode	2 byte
short	-32768…32767	2 byte
byte	-128…127	1 byte

Variabili, dichiarazioni e assegnazioni

- Una variabile è un nome usato per riferirsi a un valore di un tipo di dati (es. contatore)
- Una variabile è creata mediante una dichiarazione:

```
int contatore;
```

- Il valore viene assegnato a una variabile mediante una assegnazione:

```
contatore = 0;
```

- Un'istruzione può includere una dichiarazione e una assegnazione **allo stesso tempo**:

```
int contatore = 0;
```

Rispetto a python

- In Java siamo costretti a specificare il tipo della variabile
- Questo tipo non può più cambiare (ovvero è statico)
  - Tuttavia esistono alcuni elementi di dinamicità (che vedremo più avanti)

Non posso utilizzare una variabile senza prima dichiararla

NO: a = "stringa";
SI: String a = "stringa";

Non posso assegnare a una variabile tipi incompatibili tra loro

NO: int a = "stringa";
NO: String a = 50;
SI: String a = "stringa";

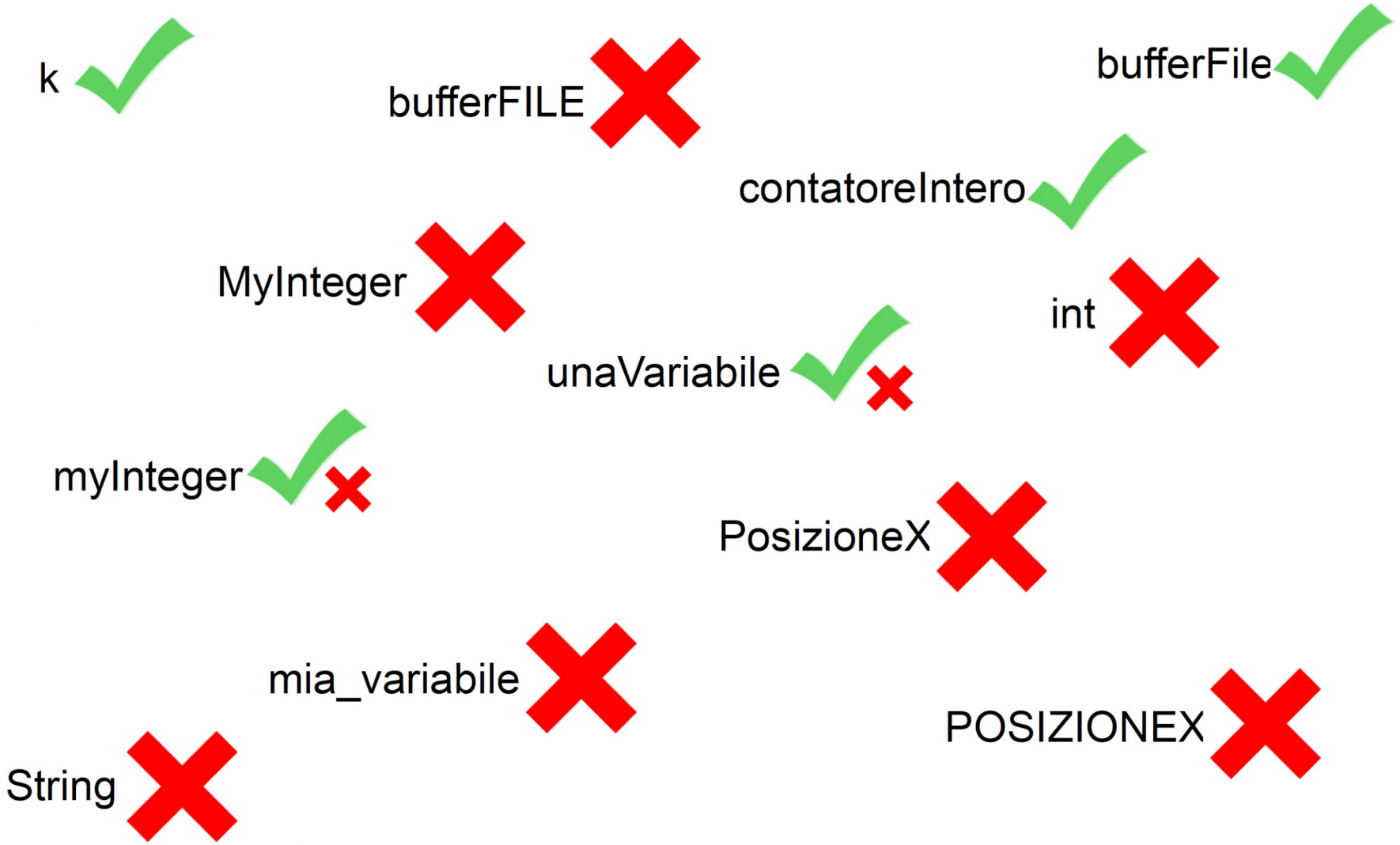
Variabili e identificatori

- Il nome assegnato a una variabile è detto identificatore, ovvero una sequenza di lettere, cifre, _ e $ la prima delle quali **non** è una cifra.

- Gli identificatori sono case-sensitive
- Non possono essere utilizzate alcune parole riservate (es. public, static, int, double, ecc.)
- Si utilizza la notazione a cammello (Camel case).

- Le variabili devono avere un **nome sensato**

Variabili e identificatori



Assegnazione e inizializzazione

- Alcuni esempi (**ci troviamo all'interno di un metodo**):

```
int a, b;
a = 5;
b = a+10;
int c = a+b;
a = c-3;
```

Assegnazione e inizializzazione

Alcuni esempi (**ci troviamo all'interno di un metodo**):

	a	b	c
`int a, b;`	non definita	non definita	
`a = 5;`			
`b = a+10;`			
`int c = a+b;`			
`a = c-3;`			

Al momento della dichiarazione, il valore iniziale della variabile non è definito!

Assegnazione e inizializzazione

- Alcuni esempi (**ci troviamo all'interno di un metodo**):

	a	b	c
`int a, b;`	non definita	non definita	
`a = 5;`	5	non definita	
`b = a+10;`			
`int c = a+b;`			
`a = c-3;`			

Assegnazione e inizializzazione

Alcuni esempi (**ci troviamo all'interno di un metodo**):

```
int a, b;
a = 5;
b = a+10;
int c = a+b;
a = c-3;
```

a	b	c
non definita	non definita	
5	non definita	
5	15	

Assegnazione e inizializzazione

Alcuni esempi (**ci troviamo all'interno di un metodo**):

	a	b	c
`int a, b;`	non definita	non definita	
`a = 5;`	5	non definita	
`b = a+10;`	5	15	
`int c = a+b;`	5	15	20
`a = c-3;`			

Assegnazione e inizializzazione

- Alcuni esempi (**ci troviamo all'interno di un metodo**):

	a	b	c
`int a, b;`	non definita	non definita	
`a = 5;`	5	non definita	
`b = a+10;`	5	15	
`int c = a+b;`	5	15	20
`a = c-3;`	17	15	20

Letterali

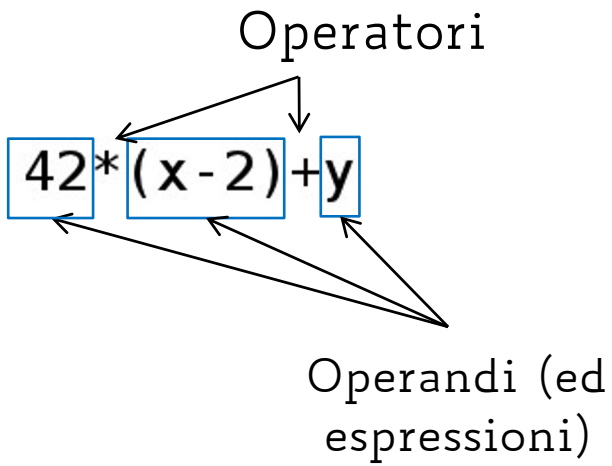
- Un letterale è una rappresentazione a livello di codice sorgente del valore di un tipo di dati
  - 27 o -32 sono letterali per gli interi
  - 3.14 è un letterale per i double
  - true o false sono gli unici due letterali per il tipo booleano
  - “Ciao, mondo” è un letterale per il tipo String

Distinguere tra costanti intere e in virgola mobile

- Interi:
  - Le costanti di tipo int sono semplicemente numeri nell'intervallo
    [-2, +2] miliardi circa
  - Le costanti di tipo long vengono specificate con il suffisso l o L (ad esempio, 10000000000000L)
- Numeri in virgola mobile:
  - Le costanti di tipo double sono semplicemente numeri con la virgola (punto)
  - Le costanti di tipo float hanno il suffisso f o F (ad esempio, 10.5f)
- In tutti i casi si può utilizzare il trattino basso (_) per separare le cifre (es. 100_000 per indicare 100000, 1_234 per indicare 1234 ecc.)
- Si può ottenere un intero da una rappresentazione binaria anteponendo 0b alla stringa binaria (ad esempio, 0b101 vale 5)

Espressioni

Un'espressione è un letterale, una variabile o una sequenza di operazioni su letterali e/o variabili che producono un valore



Assegnazione di un'espressione a una variabile

Supponiamo di voler effettuare l'assegnazione:
c = a*2+b

1) Calcola il valore dell'espressione destra (5*2+15)
2) Assegna il valore (25) alla variabile di destinazione (c)

a	b	c
5	15	

a	b	c
5	15	25

Precedenza degli operatori aritmetici

Operatori	Operazioni	Precedenza
*, /, %	Moltiplicazione, divisione, resto	Valutati per primi, da sinistra verso destra
+, -	Addizione, sottrazione	Valutati per secondi, da sinistra verso destra

- Quanto fa `5-2-3/2.0+2*2-5%2/2.0`?

Caratteri e stringhe

- Un char è un carattere alfanumerico o un simbolo
- Ci sono 2^{16} possibili valori di caratteri (più eventuali "caratteri supplementari", per esempio per il cinese)
  - Codifica Unicode basata su interi a 16 bit
- Racchiusi da apici (es. 'a', 'b', 'O', '1', ecc.)
- Caratteri di escape:
  - **Tab**: '\t'
  - **A capo**: '\n'
  - **Backslash**: '\\'
  - **Apice**: '\''
  - **Virgolette**: '\"'

- Una stringa è una sequenza di caratteri.

Tipi booleani

- Il tipo booleano ha solo due valori possibili: true (vero) e false (falso)
- Gli operatori disponibili sono && (and), || (or) e ! (not)
- Le tabelle di verità sono:

a	!a
true	false
false	true

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Esempio di istruzioni con variabile booleane

```
boolean piove = true;
boolean giocoAPallone = !piove;
boolean fuoriForma = true;
boolean vadoInPalestra = piove &&
fuoriForma;
```

Operatori di confronto

- Definiti sui tipi numerici primitivi
- Producono un valore booleano
  - **Uguaglianza**: ==
  - **Diversità**: !=
  - **Minore**: <
  - **Minore uguale**: <=
  - **Maggiore**: >
  - **Maggiore uguale**: >=

Operatori di confronto

operatore	significato	true	false
==	uguale	27 == 27	27 == 43
!=	diverso	27 != 43	27 != 27
<	minore	27 < 100	27 < 27
<=	minore o uguale	27 <= 27	43 <= 27
>	maggiore	100 > 27	27 > 100
>=	maggiore o uguale	43 >= 27	27 >= 43

Operatori di confronto

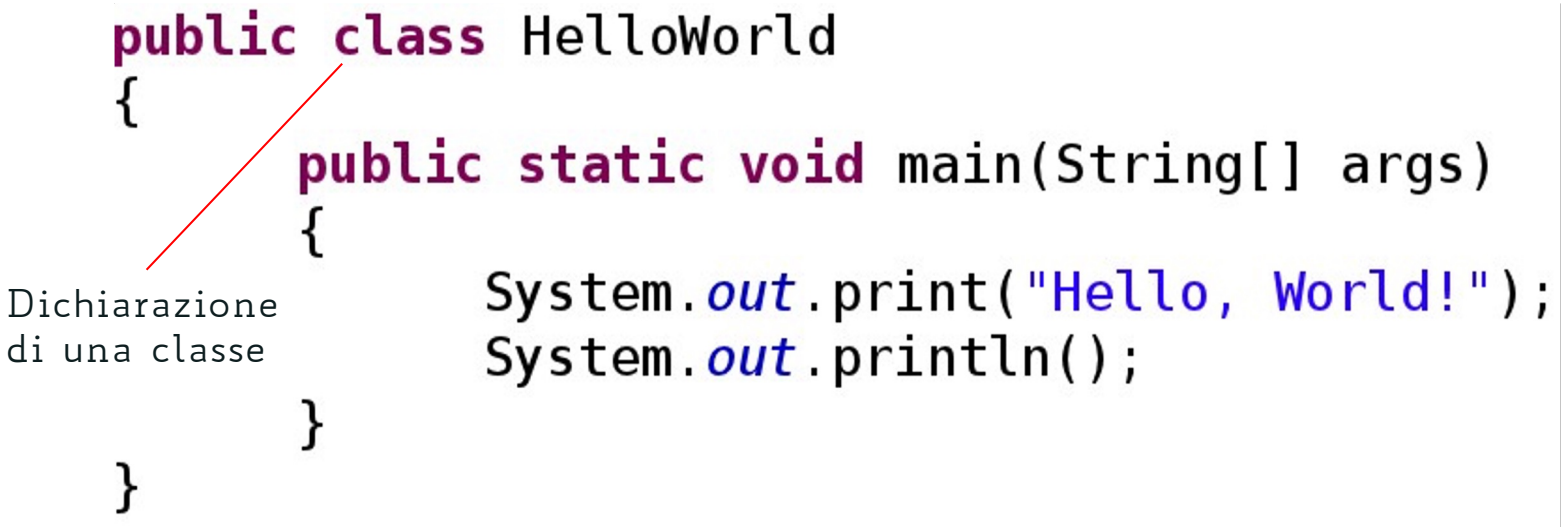
Operatori	Operazioni	
post-incremento e post-decremento	var++ var--	
pre-incremento e pre-decremento, operatori unari	++var --var +espr -espr	~ !
operatori moltiplicativi	* / %	
operatori additivi	+ -	
shift	<< >> >>>	
relazionali	< > <= >= instanceof	
confronto	== !=	
AND bit a bit	&	
OR esclusivo bit a bit (XOR)	^	
OR inclusivo bit a bit	|	
AND logico	&&	
OR logico	||	
operatore ternario	? :	
lambda	->	
assegnazione	= += -= *= /= %= &= ^= |= <<= >>= >>>=	

Hello, World!

HelloWorld.java

```
public class HelloWorld
{
	public static void main(String[] args)
	{
		System.out.print("Hello, World!");
		System.out.println();
	}
}
```

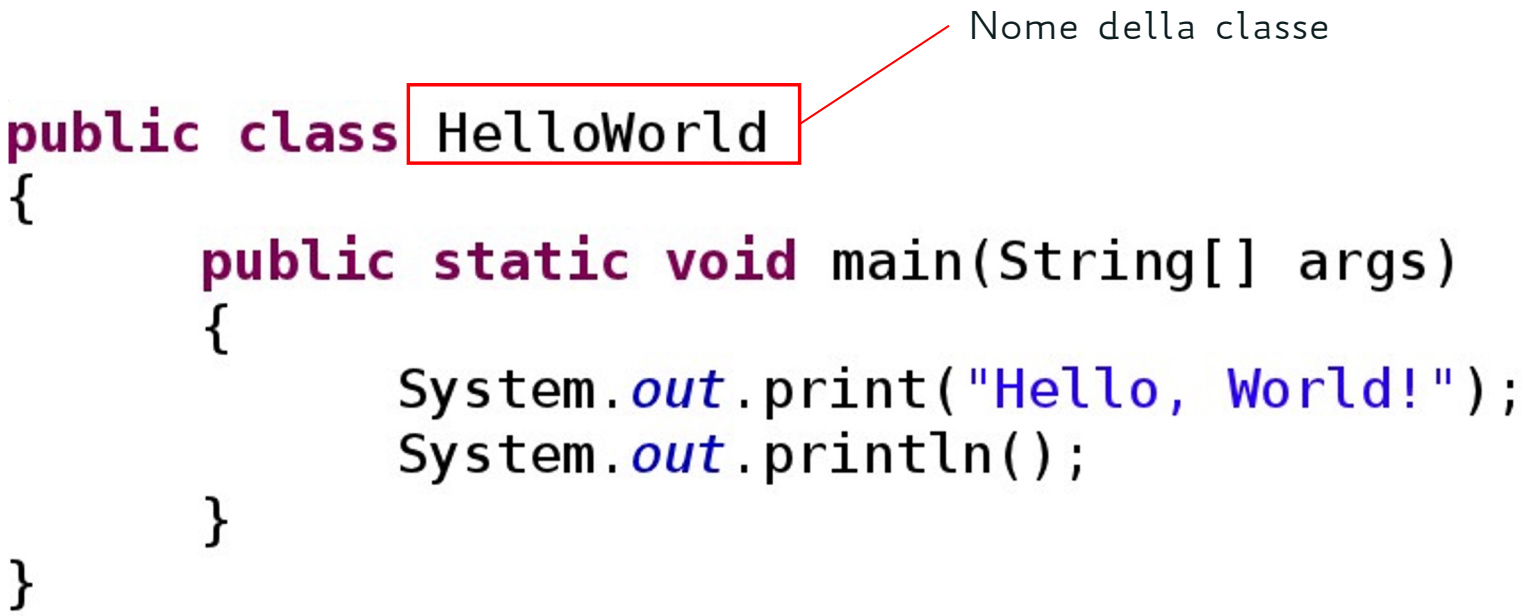
Anatomia di HelloWorld.java



public, class, static, void: sono parole chiave

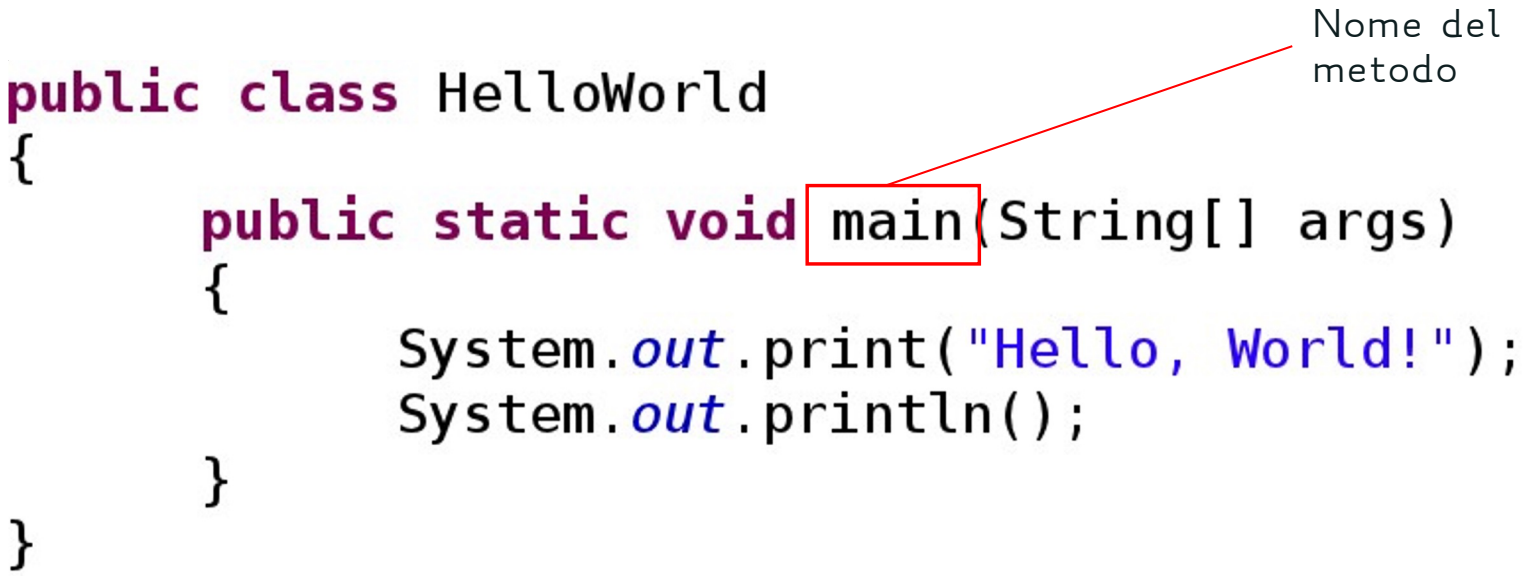
System, out sono …

Anatomia di HelloWorld.java



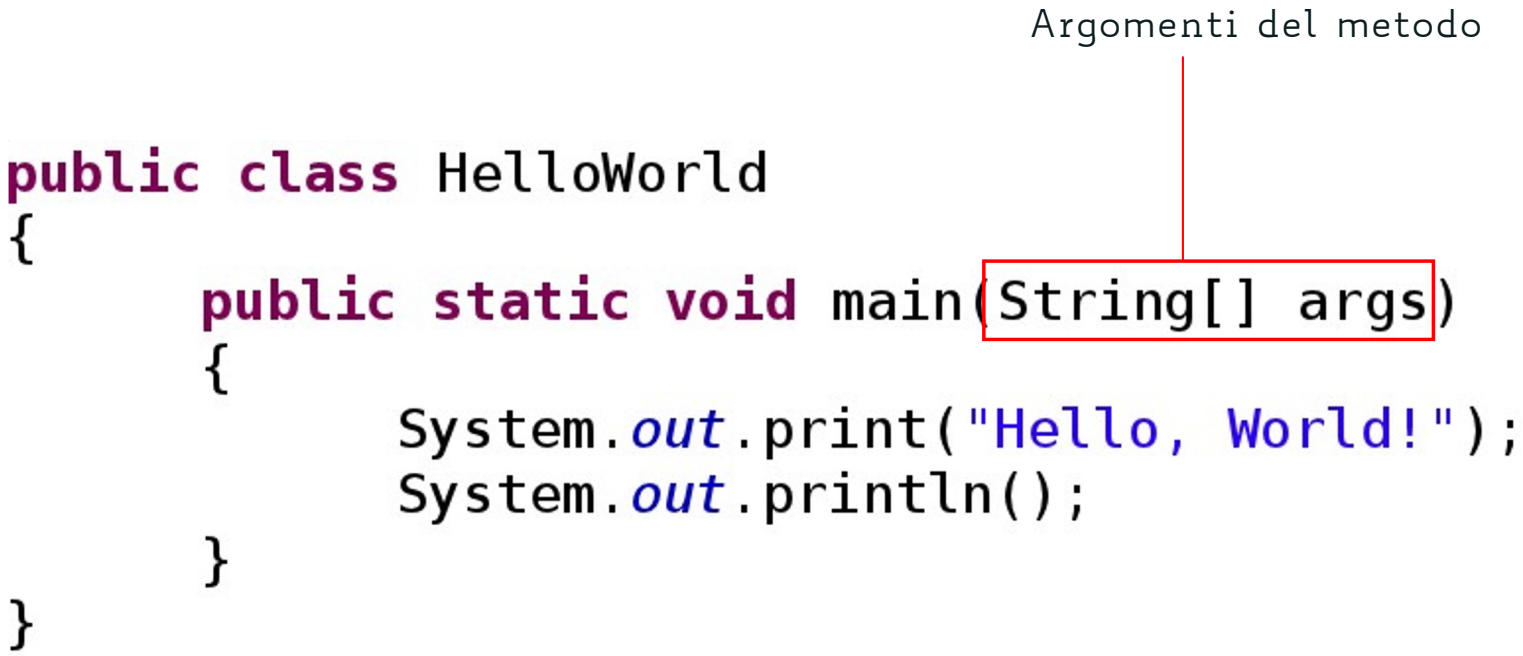
Il programma (**meglio**: la classe) Java risiede in un file che ha lo stesso nome della classe creata (HelloWorld) più l'estensione .java (HelloWorld.java)

Anatomia di HelloWorld.java



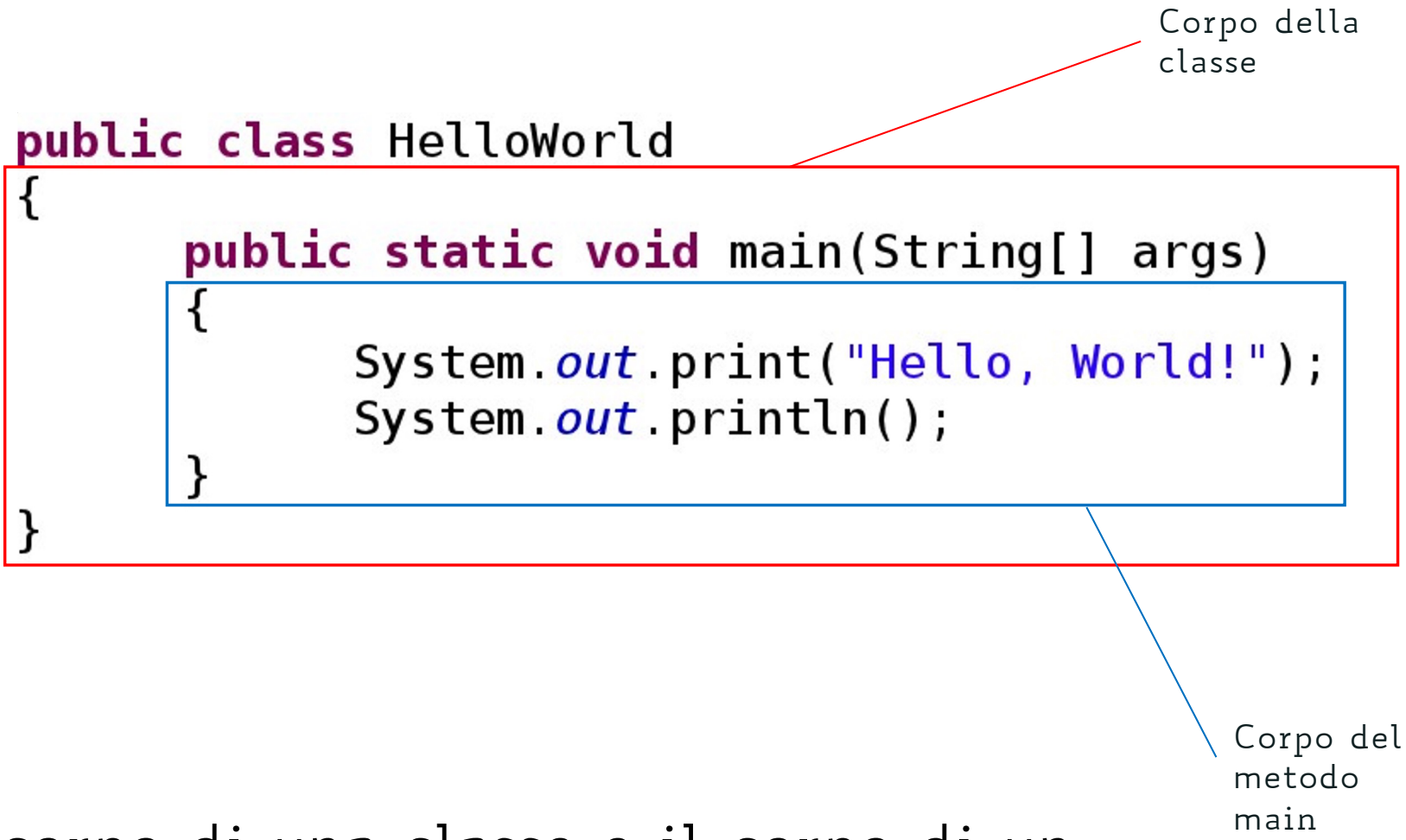
La classe contiene un metodo che si chiama main

Anatomia di HelloWorld.java



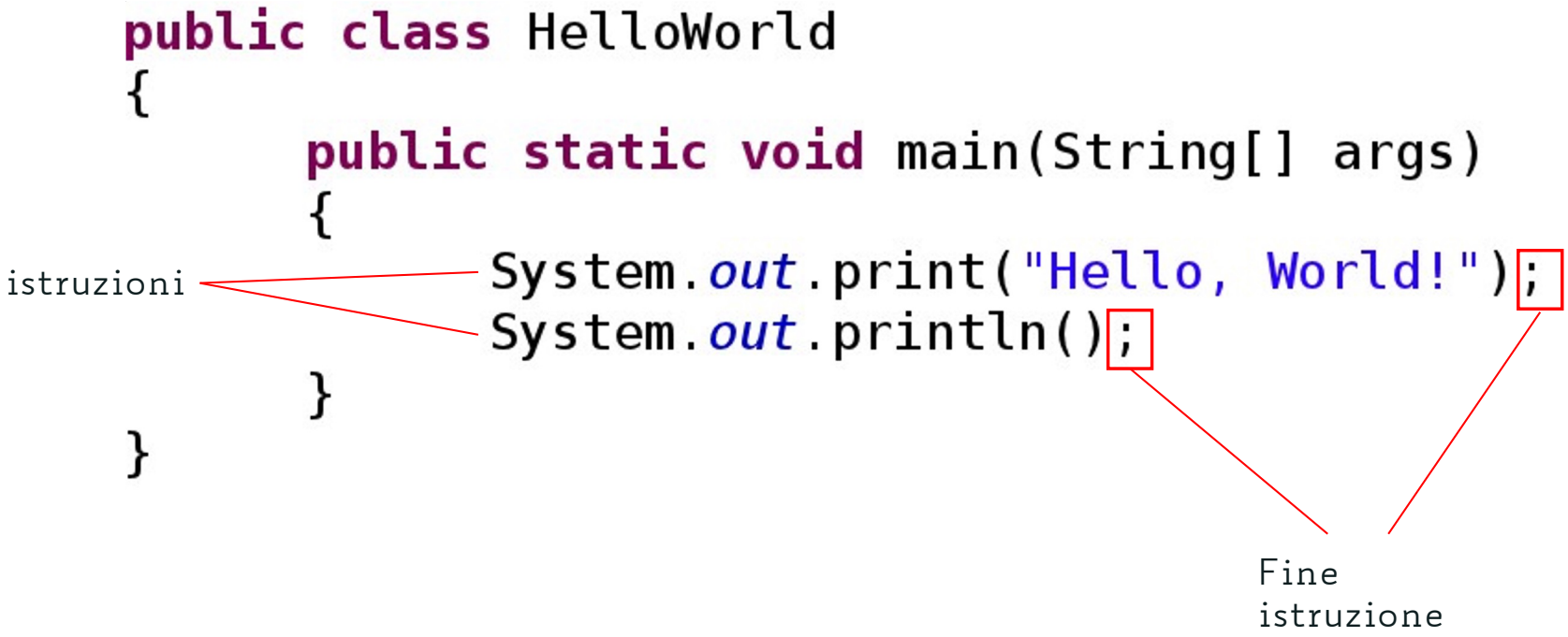
(String[] args) definisce gli argomenti del metodo

Anatomia di HelloWorld.java



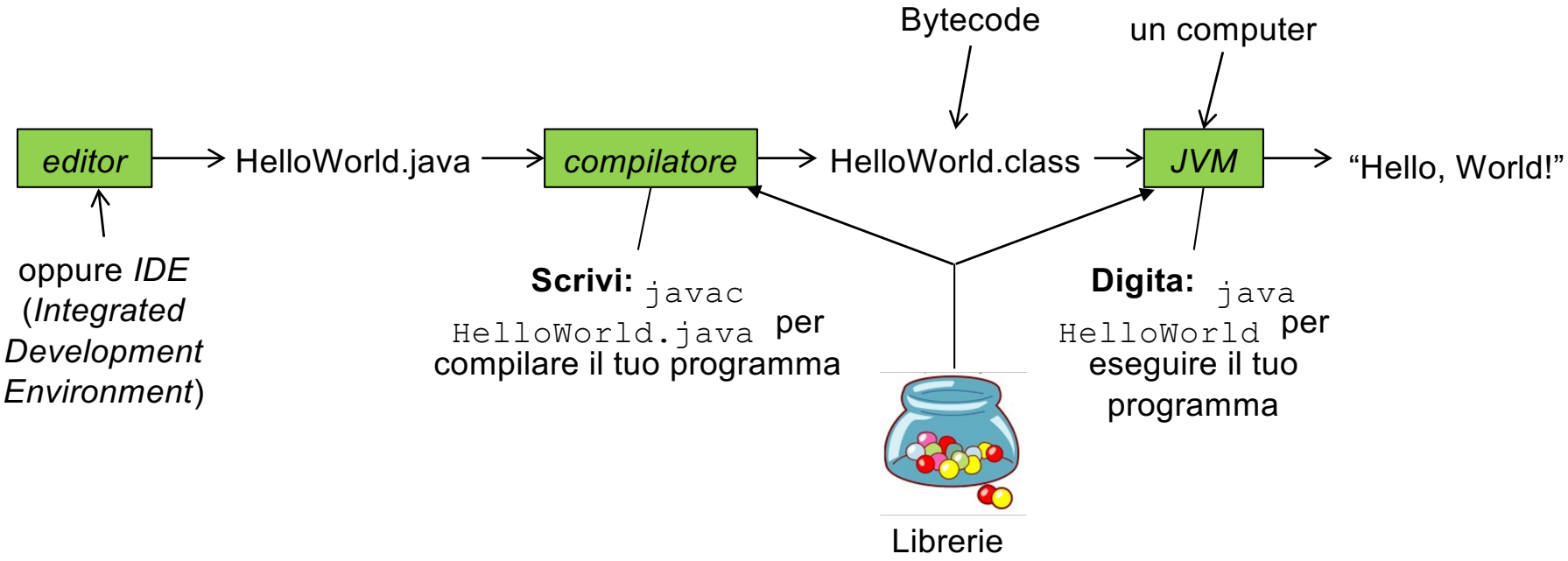
Il corpo di una classe e il corpo di un metodo sono delimitati da { }

Anatomia di HelloWorld.java



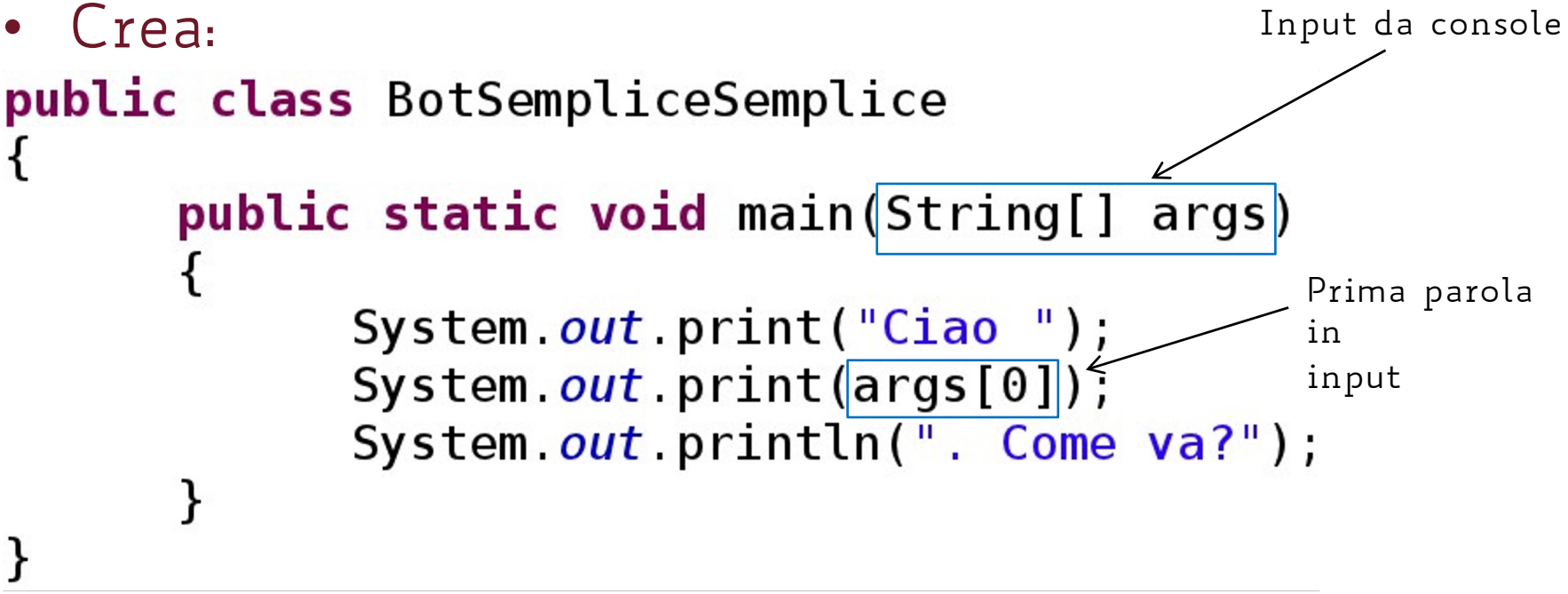
Tutte le istruzioni sono terminate con ;

Create (Crea), Compile (Compila), Run (Esegui)



Ricevere un input in fase di esecuzione e generare un output

- Crea:

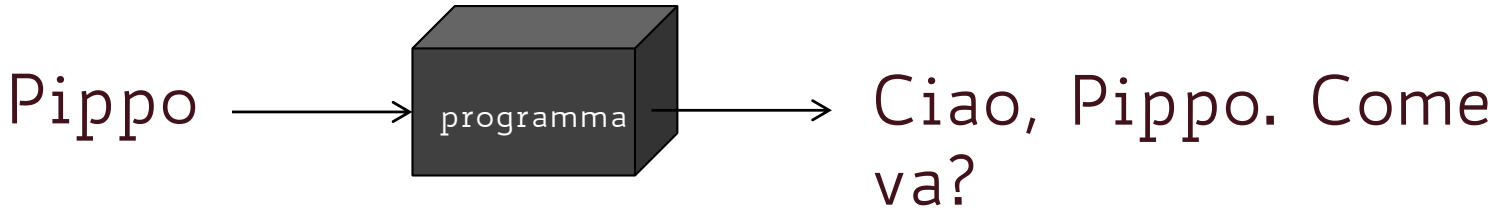



```
public class BotSempliceSemplice
{
	public static void main(String[] args)
	{
		System.out.print("Ciao ");
		System.out.print(args[0]);
		System.out.println(". Come va?");
	}
}
```

- Compila: javac BotSempliceSemplice.java
- Esegui: java BotSempliceSemplice Pippo
- Output: Ciao Pippo. Come va?

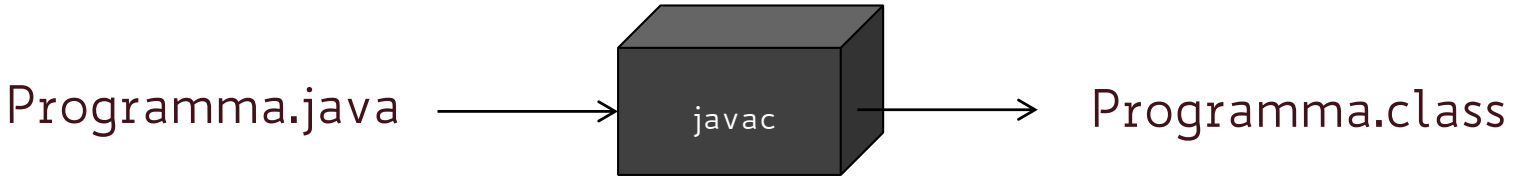
Scatola Nera

Abbiamo implementato un programma che prende una stringa in input e ne restituisce un'altra in output



Scatola Nera

Javac prende in input una classe .java (una stringa di testo) e ne restituisce la versione "compilata" .class (una stringa di testo) in bytecode



Regole di stile nella scrittura del codice

Posso scrivere HelloWorld.java in questo modo?

```
public            class
          HelloWorldAnarchico
{ public static void main(String

            [] args){        System.out.print("Hello, World!");
                                 System.out.println();     }
             }
```

Regole di stile nella scrittura del codice

Posso scrivere HelloWorld.java in questo modo?

```
public            class
          HelloWorldAnarchico
{ public static void main(String

            [] args){        System.out.print("Hello, World!");
                                System.out.println();     }
             }
```

Potenzialmente Sì: non è una questione di spazi e di "a capo".

Regole di stile nella scrittura del codice

Posso scrivere HelloWorld.java in questo modo?

```
public            class
          HelloWorldAnarchico
{ public static void main(String

            [] args){        System.out.print("Hello, World!");
                                System.out.println();     }
             }
```

Potenzialmente Sì: non è una questione di spazi e di "a capo".

Decisamente No!: il codice è scritto per essere **riletto** e **riusato** da altri (e da noi stessi) dopo qualche settimana, mese, anno…

Esempio di espressione nel codice main di una classe

```
public class Espressioni
{
    public static void main(String[] args)
    {
        // esempi di espressioni
        int k = 10;
        int j = k+20;
        int h;

        System.out.print("k == ");
        System.out.println(k);
        System.out.print("j == ");
        System.out.println(j);
        System.out.print("h == ");
        System.out.println(h);  // non definito: errore di compilazione!

        String s1 = "una stringa ";
        String s2 = "e un'altra stringa";
        String s3 = s1+s2;
        System.out.println(s3);

        double d = j+5.3;
        System.out.println("d == "+d);

        boolean b = (d == 35.3) && (j == 30);
        System.out.println("b == "+b);
    }
}
```

Questa riga va eliminata!

Output:

```
k == 10
j == 30
h == una stringa e un'altra stringa
d == 35.3
b == true
```

Errori tipici

```java
public class AssegnazioniCheNonVanno
{
    public static void main(String[] args)
    {
        // ERRORE DI COMPILAZIONE: manca il tipo di k!!!
        k = 50;

        // ERRORE DI COMPILAZIONE: non posso assegnare una string a un intero!
        int k = "50";

        // ERRORE DI COMPILAZIONE: non posso assegnare un intero a una stringa!
        String s = 20;
    }
}
```

operatori di confronto, booleani e anno bisestile

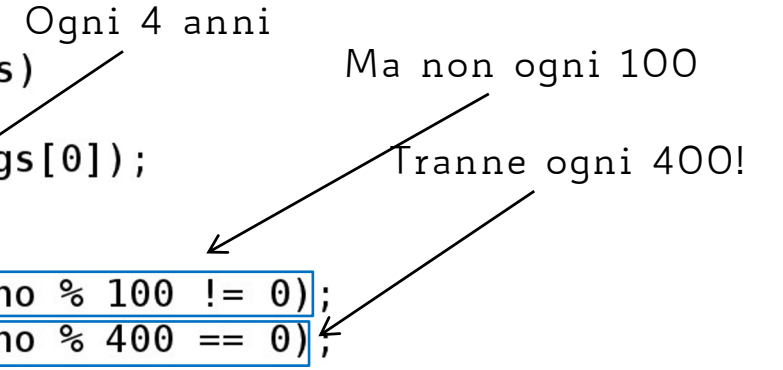
```
public class AnnoBisestile
{
	public static void main(String[] args)
	{
		int anno = Integer.parseInt(args[0]);
		boolean bBisestile;
		bBisestile = anno % 4 == 0;
		bBisestile = bBisestile && (anno % 100 != 0);
		bBisestile = bBisestile || (anno % 400 == 0);

		System.out.println("L'anno "+anno+" e' bisestile? "+bBisestile);
	}
}
```

operatori di confronto, booleani e anno bisestile

```
public class AnnoBisestile
{
	public static void main(String[] args)
	{
		int anno = Integer.parseInt(args[0]);
		boolean bBisestile;
		bBisestile = anno % 4 == 0;
		bBisestile = bBisestile && (anno % 100 != 0);
		bBisestile = bBisestile || (anno % 400 == 0);

		System.out.println("L'anno "+anno+" e' bisestile? "+bBisestile);
	}
}
```



- Compila: javac AnnoBisestile.java
- Esegui: java AnnoBisestile 2012
- Output: L'anno 2012 e' bisestile? true

Esercizi sulle espressioni

- Si riscriva l'espressione bBisestile della diapositiva precedente utilizzando un'unica istruzione di assegnazione
- Siano date a e b variabili di tipo double; si scriva l'istruzione che definisce c pari al valore true se a è maggiore di b, false altrimenti
- Siano date una variabile intera a e una variabile double b; si scriva l'istruzione che assegna a c la parte intera della differenza tra a e b
- Qual è il valore di x dopo l'esecuzione di quanto segue?

```
int b = 10;
double a = 5;
int x = (int)a/b + 5;
```

Alcune funzioni matematiche utili

Alcuni metodi della classe Math di Java

- double **abs**(double a)
- double **max**(double a, double b)
- double **min**(double a, double b)
- double **sin**(double theta)
- double **cos**(double theta)
- double **tan**(double theta)
- double **exp**(double a)
- double **log**(double a)
- double **pow**(double a, double b)
- long **round**(double a)
- double **random**()
- double **sqrt**(double a)

Alcune costanti di base:

- double **E**
- double **PI**

Da stringhe a dati primitivi

Supponiamo di voler utilizzare l'input di un programma per effettuare dei calcoli (es. somme)

Come convertiamo da stringa a intero?

```
public class SommaInteri
{
	public static void main(String[] args)
	{
		int a = Integer.parseInt(args[0]);
		int b = Integer.parseInt(args[1]);
		System.out.print("La somma vale: ");
		System.out.println(a+b);
	}
}
```

Da stringhe a dati primitivi

Supponiamo di voler utilizzare l'input di un programma per effettuare dei calcoli (es. somme)

Come convertiamo da stringa a intero?

```
public class SommaInteri
{
	public static void main(String[] args)
	{
		int a = Integer.parseInt(args[0]);
		int b = Integer.parseInt(args[1]);
		System.out.print("La somma vale: ");
		System.out.println(a+b);
	}
}
```

Da stringhe a dati primitivi

e da stringa a double?

```
public class SommaVirgolaMobile
{
	public static void main(String[] args)
	{
		double a = Double.parseDouble(args[0]);
		double b = Double.parseDouble(args[1]);
		System.out.print("La somma vale: ");
		System.out.println(a+b);
	}
}
```

Da dati primitivi a stringhe

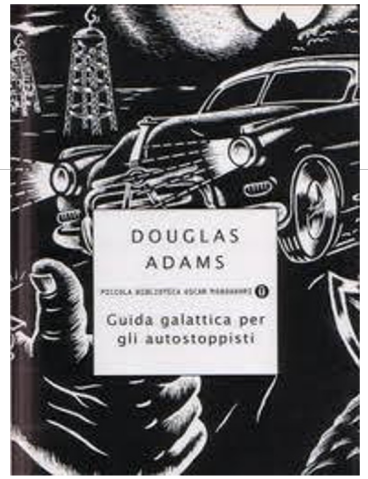
Java definisce l'operatore + sul tipo di dato "built-in" String

Quando usiamo + con almeno un operando String, Java converte automaticamente l'altro operando a String, restituendo una stringa:

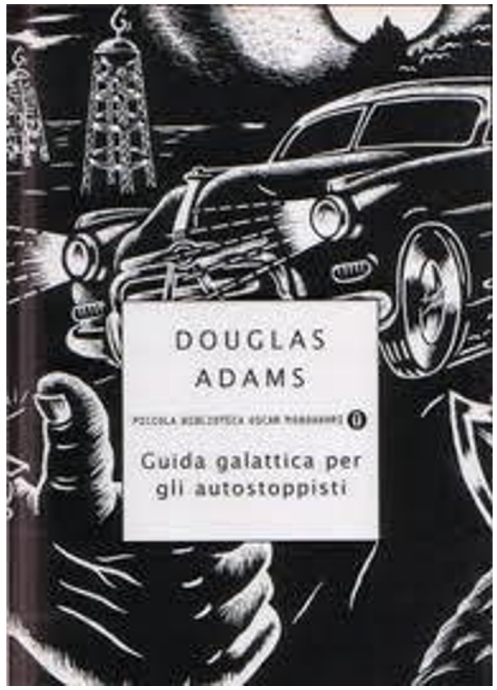
```
public class PensieroProfondo
{
	public static void main(String[] args)
	{
		String s = "La risposta alla domanda fondamentale sulla vita, l'universo e tutto quanto e' ";
		int v = 42;

		String risposta = s+v;
		System.out.println(risposta);
	}
}
```

https://it.wikipedia.org/wiki/Risposta_alla_domanda_fondamentale_sulla_vita,_l'universo_e_tutto_quanto



https://it.wikipedia.org/wiki/Risposta_alla_domanda_fondamentale_sulla_vita,_l'universo_e_tutto_quanto



La tua consapevolezza del tipo di dati

- DEVI SEMPRE ESSERE CONSAPEVOLE del tipo di dati che il tuo programma sta elaborando
- Quali valori può assumere ciascuna variabile?
- Tuttavia, lavoriamo tipicamente con molteplici tipi di dati
- **Come convertire un tipo di dati?**

Conversioni di tipo

Conversione esplicita:

- Utilizzando un metodo che prende in ingresso un argomento di un tipo e restituisce un valore di un altro tipo
- Integer.parseInt(), Double.parseDouble(), Math.round(), Math.floor(), Math.ceil() ecc.

Cast esplicito:

- Anteponendo il tipo desiderato tra parentesi
- (int)2.71828 produce un intero di valore 2
- Se il tipo di partenza è più preciso (es. double), le informazioni aggiuntive vengono eliminate nel modo più ragionevole (es. da double a int viene eliminata la parte frazionaria)

Cast implicito:

- Se il tipo di partenza è meno preciso, Java converte automaticamente il valore al tipo più preciso
- `double d = 2;`
- **Attenzione:** la somma di due caratteri dà un intero!

Regole per il cast implicito

Il **cast implicito** avviene **in fase di assegnazione**:

- byte, short e char possono essere promossi a int
- int può essere promosso a long
- float può essere promosso a double

o in fase di **calcolo di un'espressione**:

- se uno dei due operandi è double, l'intera espressione è promossa a double
- altrimenti, se uno dei due operandi è float, l'intera espressione è promossa a float

Conversioni di tipo

espressione	tipo	valore
(int)2.71828	int	2
Math.round(2.71828)	long	3
(int) Math.round(2.71828)	int	3
(int) Math.round(3.14159)	int	3
Integer.parseInt(“42”)	int	42
“42” + 99	String	“4299”
42 * 0.4	double	16.8
(int)42 * 0.4	double	16.8
42 * (int)0.4	int	0
(int)(42 * 0.4)	int	16

Il cast ha precedenza più elevata!

Quanto fa 5-2-3/2.0+2*2-5%2/2.0?

Prima si effettuano i prodotti, le divisioni e i moduli da sinistra verso destra

- Quando si effettua un'operazione aritmetica tra tipi diversi ma compatibili, avviene un cast implicito

5-2-**3/2.0**+2*2-5%2/2.0
5-2-**3.0/2.0**+2*2-5%2/2.0
5-2-**1.5**+2*2-5%2/2.0
5-2-1.5+**2*2**-5%2/2.0
5-2-1.5+**4**-5%2/2.0
5-2-1.5+4-**5%2**/2.0
5-2-1.5+4-**1**/2.0
5-2-1.5+4-**1.0/2.0**
5-2-1.5+4-**0.5**

Infine si calcolano somme e sottrazioni da sinistra verso destra

3-1.5+4-0.5 -> **1.5**+4-0.5 -> **5.5**-0.5 -> **5.0**

Esercizi per cominciare

Scrivere una classe Moltiplica che, dati in input 2 numeri interi, ne restituisca a video il prodotto

Scrivere una classe StampaNome che, dato in input un nome, lo stampi tra due righe di trattini. Ad es.:

```
+--------------------+
Pippo
+--------------------+
```

Soluzioni degli esercizi

```java
public class Moltiplica
{
    public static void main(String[] args)
    {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);

        System.out.println(a*b);
    }
}
```

```java
public class StampaNome
{
    public static void main(String[] args)
    {
        final String trattini = "+----------+";

        System.out.println(trattini);
        System.out.println(args[0]);
        System.out.println(trattini);
    }
}
```

L'intestazione del main è scolpita nella pietra!

```java
public static void main(int a, int b)
{
    // NO!!!
}
```

```java
public static void main(int[] args)
{
    // NOOOO!!!!
}
```

```java
public static void main(String[] args)
{
    // OK!
}
```

Esercizi per cominciare

Scrivere una classe Variabili che, all'interno del metodo main, dichiari una variabile intera i, una variabile di tipo stringa s e una variabile double d. Quindi vengono svolte le seguenti tre operazioni:

- La stringa viene inizializzata al valore del primo argomento fornito in input al main
- All'intero viene assegnato il valore intero della stringa
- Al double viene assegnata la metà del valore di i (ad es. se i è pari a 3, d sarà pari a 1.5)
- I valori di s, i e d vengono stampati a video

Soluzione dell'esercizio

```
public class Variabili
{
    public static void main(String[] args)
    {
        // primo argomento in input
        String s = args[0];

        // converte il primo argomento in intero
        int i = Integer.parseInt(s);

        // occhio al 2.0 (l'operatore converte al tipo più potente)
        double d = i/2.0;

        System.out.println("s = "+s+", i = "+i+", d = "+d);
    }
}
```

Un generatore di numeri

```
public class GeneraNumeri
{
	static public void main(String[] args)
	{
		// numero in input
		int max = Integer.parseInt(args[0]);

		// numero (pseudo)casuale compreso tra 0 e 1
		double r = Math.random();

		// intero (pseudo)casuale tra 0 e max-1
		int n = (int) (r*max);

		System.out.println(n);
	}
}
```

Commento su singola linea

Escluso 1

Phrase-O-Matic

- Progettare una classe i cui oggetti contengono tre elenchi di parole l_1, l_2 e l_3
- La classe è in grado di emettere nuove espressioni costruite creando stringhe del tipo "a b c" scegliendo casualmente dai tre rispettivi elenchi a $\in l_1$, b $\in l_2$, c $\in l_3$

Ad esempio, dati i seguenti elenchi:

- l_1 = { "salve", "ciao", "hello", "buongiorno", "scialla" }
- l_2 = { "egregio", "eclettico", "intelligentissimo", "astutissimo" }
- l_3 = { "studente", "ragazzo", "giovane", "scapestrato", "fannullone", "studioso" }

Esempi di output sono:

- "salve egregio fannullone"
- "ciao eclettico scapestrato"

Dichiarazione di un array di stringhe:

String[] a = { "a", "b", "c" };

Accesso all'i-esimo elemento dell'array:

String s = a[i];

```java
/**
 * Versione 1 (DA DIMENTICARE dalla prossima settimana in poi!!!)
 *
 * @author navigli
 */
public class PhraseOMatic
{
    public static void main(String[] args)
    {
        final String[] l1 = { "salve", "ciao", "hello", "buongiorno", "scialla" };
        final String[] l2 = { "egregio", "eclettico", "intelligentissimo", "astutissimo" };
        final String[] l3 = { "studente", "ragazzo", "giovane", "scapestrato", "fannullone", "studioso" };

        String s1 = l1[(int)(Math.random()*l1.length)];
        String s2 = l2[(int)(Math.random()*l2.length)];
        String s3 = l3[(int)(Math.random()*l3.length)];

        System.out.println(s1+" "+s2+" "+s3);
    }
}
```

N.B.: Sarebbe errore scrivere:
(int)Math.random()*l3.length
perché il cast ha precedenza sul prodotto!

Soluzione (ideale)

```java
/**
 * Versione 2: orientata a oggetti!
 *
 * @author navigli
 */
public class PhraseOMatic
{
    private String[] l1 = { "salve", "ciao", "hello", "buongiorno", "scialla" };
    private String[] l2 = { "egregio", "eclettico", "intelligentissimo", "astutissimo" };
    private String[] l3 = { "studente", "ragazzo", "giovane", "scapestrato", "fannullone", "studioso" };

    /**
     * Genera una stringa casuale
     * @return la stringa generata
     */
    public String genera() { return l1[rand(l1.length)]+" "+l2[rand(l2.length)]+" "+l3[rand(l3.length)]; }

    /**
     * Metodo di comodo per la generazione di un numero casuale tra 0 e max-1
     * @param max il massimo valore (escluso) nell'intervallo degli interi da cui campionare
     * @return il numero casuale
     */
    private int rand(int max) { return (int)(Math.random()*max); }

    public static void main(String[] args)
    {
        // crea un oggetto di tipo PhraseOMatic
        PhraseOMatic p = new PhraseOMatic();
        // genera una stringa casuale
        System.out.println(p.genera());
    }
}
```

Credits

Le slide di questo corso sono il frutto di una personale rielaborazione delle slide del Prof. Navigli.

In aggiunta, le slide sono state revisionate dagli studenti borsisti della Facoltà di Ingegneria Informatica, Informatica e Statistica: Mario Marra e Paolo Straniero.

Font ad alta leggibilità **Biancoenero©** di biancoenero edizioni srl, disegnata da Umberto Mischi, con la consulenza di Alessandra Finzi, Daniele Zanoni e Luciano Perondi (Brevetto n. RM2011O0O0128).

Disponibile gratuitamente per tutte le istituzioni e i privati che la utilizzino per scopi non commerciali.