Java Come Linguaggio di Programmazione

- Programmare in Java
  - Un esempio elementare
  - Gli strumenti di Java
- Struttura di un programma Java
  - Metodi statici
  - Variabili
  - Tipi elementari
  - Istruzioni principali

Il linguaggio Java

- Formalismo ad alto livello...
  - Permette di descrivere programmi basandosi su concetti primitivi “sofisticati” (file, finestre, tabelle, liste, ecc.)
- ...basato su una notazione testuale familiare
  - Codice sorgente
  - Simile, nella sintassi e nelle istruzioni al linguaggio C

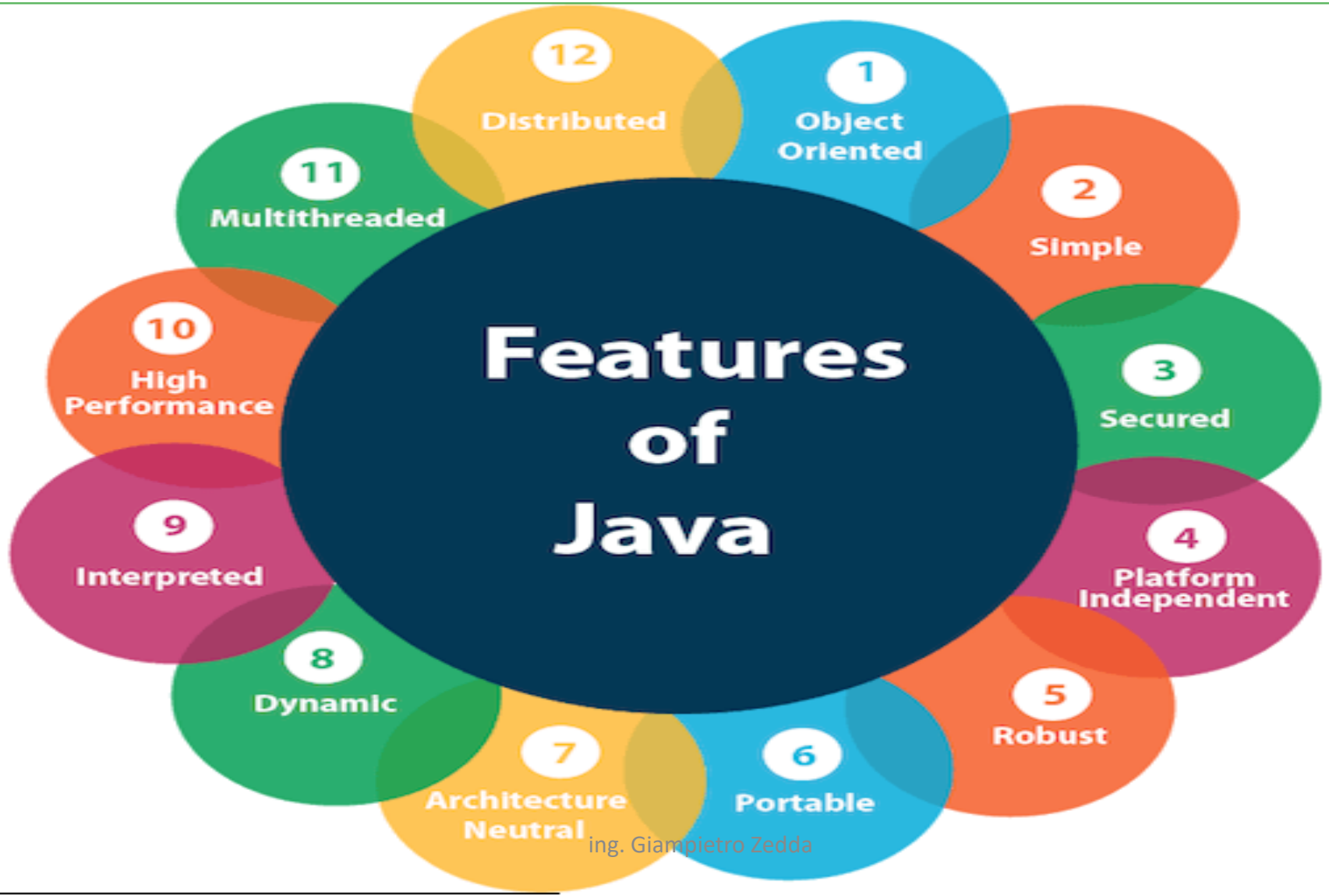
Codice Sorgente

- I programmi Java sono suddivisi in "classi"
  - Le classi sono descritte all'interno di file di testo con estensione ".java"
  - Ogni file contiene una sola classe
  - Il nome file deve coincidere con il nome della classe

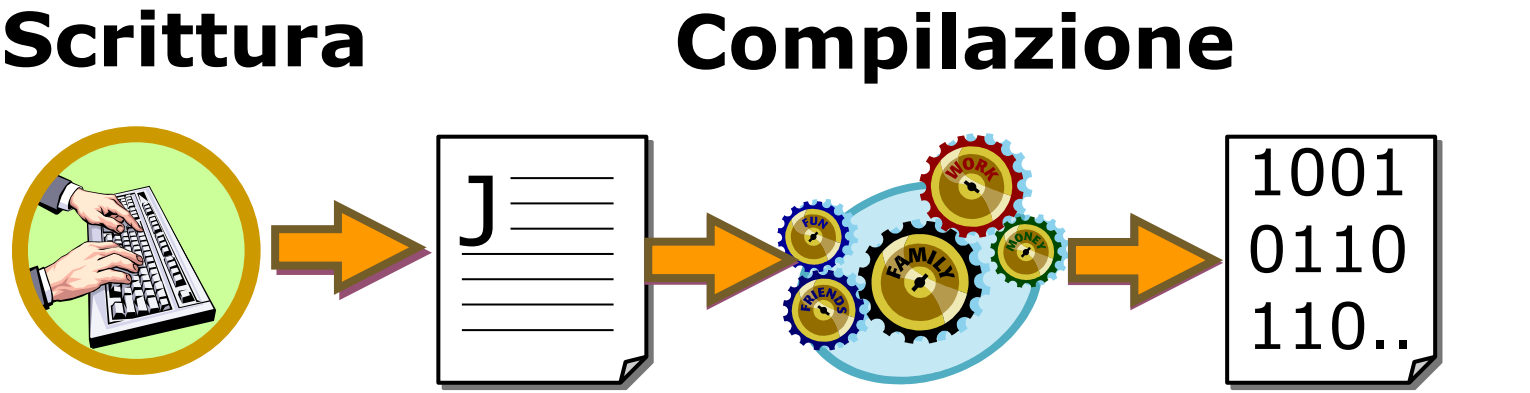
Un esempio

```
public class Hello {
  // Il programma più semplice
  public static void main(String[] args) {
    // Stampa un messaggio sul video
    System.out.println("Hello Java");
  }
}
```

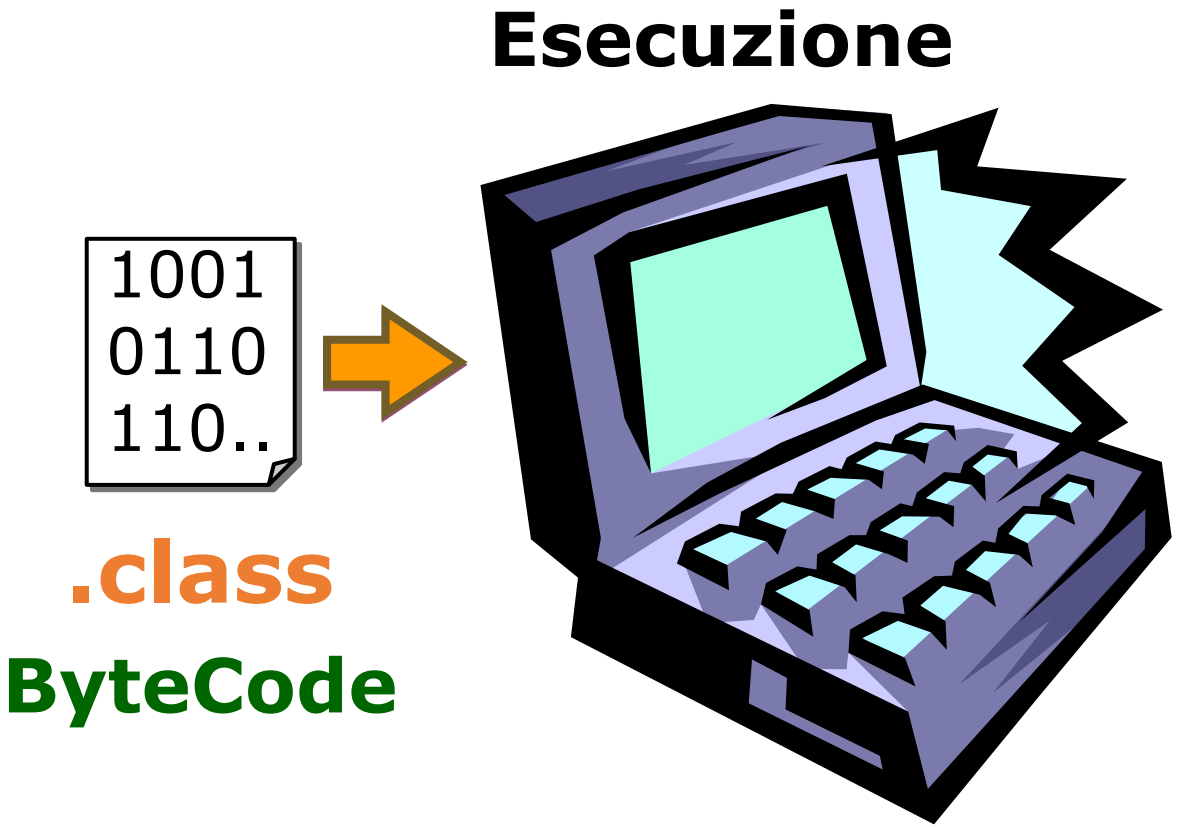
Il mondo java



Passi concettuali (1)



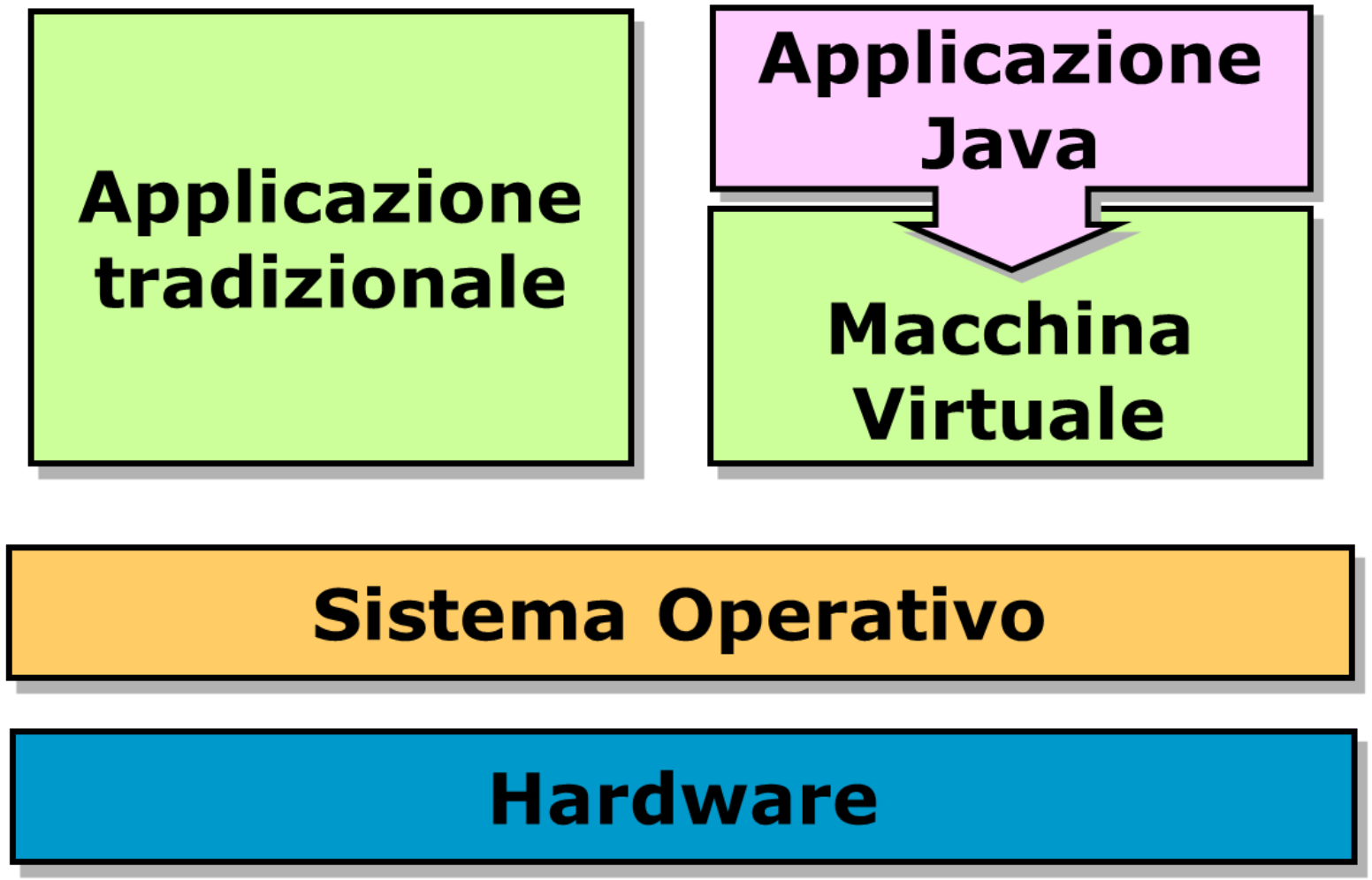
I passi concettuali (2)



Gli strumenti di Java

- La macchina virtuale
- Il compilatore
- Gli ambienti di sviluppo
- La documentazione
- Il debugger

Macchina virtuale (1)



Macchina virtuale (2)

- Astrazione di un elaboratore “generico”
  - Ambiente di esecuzione delle applicazioni Java
- Esempio:
  - *java Hello*
- Responsabilità:
  - Caricamento classi dal disco
  - Verifica consistenza codice
  - Esecuzione applicazione

Il compilatore

- Traduzione dei sorgenti testuali in bytecode
  - Linguaggio della macchina virtuale Java
- Esempio:

  javac Hello.java

Ambienti di sviluppo

- Il codice sorgente Java è composto da testo
  - Un editor qualsiasi è sufficiente
- Si possono usare ambienti integrati per lo sviluppo (IDE)
  - Offrono strumenti per la redazione, la compilazione, l'esecuzione e la ricerca degli errori

Esempi

- Eclipse
  - http://www.eclipse.org
- JCreator
  - http://www.jcreator.com
- Intellij
  - https://www.jetbrains.com
- VSCode
  - https://code.visualstudio.com/

Documentazione (1)

- Strumento **necessario** alla sopravvivenza del programmatore Java!!
- Raccolta di informazioni relative alle classi appartenenti alle librerie standard di Java
  - **https://docs.oracle.com/en/java/**

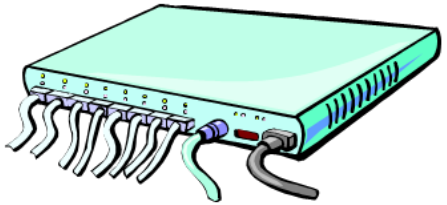
Documentazione (2)

- Per ogni classe:
  - Descrizione funzionale
  - Elenco di attributi:
    funzionalità, caratteristiche
  - Elenco dei metodi:
    funzionalità, parametri in ingresso, valori di ritorno, …

Il debugger

- Strumento ausiliario al programmatore
  - Monitorare l'esecuzione di una applicazione
  - Analisi dei valori assunti dalle variabili, i metodi della classe, …
- Comando jdb.exe

Dove si usa Java?



Apparati di rete

PC client

Cellulari e PC palmari

Server

Elettrodomestici

Java nei sistemi Client

- Vantaggi
  - Facilità di realizzazione delle interfacce grafiche
  - Indipendenza dalla piattaforma di sviluppo e di utilizzo
  - Ciclo di sviluppo rapido
  - Distribuzione immediata, a richiesta, tramite web
- Svantaggi
  - Prestazioni limitate rispetto ad applicazioni native
  - Ambienti di esecuzione in forte evoluzione

Java nei sistemi Server

- Varie tipologie di utilizzo:
  - Servlet / Java Server Pages / Java Server Faces
  - EnterpriseJavaBean
  - Application server
- È il contesto in cui Java sta ottenendo il massimo successo

Java nei sistemi Embedded

- Segmento di mercato in forte crescita:
  - Milioni di PC connessi ad Internet
  - Centinaia di milioni di telefoni cellulari
  - Miliardi di elettrodomestici
- Vengono usate librerie specifiche
  - JavaCard, Java Micro Edition, …

Struttura dei file (1)

- **<File>.java**
  - Contengono il codice sorgente Java
- **<File>.class**
  - Contengono il risultato della compilazione
  - Espressi in bytecode (formato binario)
  - All'interno sono presenti le istruzioni, la tabella dei simboli, le informazioni ausiliarie necessarie all'esecuzione

Struttura dei file (2)

- **<File>.jar**
  - Archivi compressi che consentono di aggregare più file
  - Contengono una o più classi Java, risorse e ulteriori informazioni ausiliarie (documentazione, parametri di configurazione, …)
- **<File>.properties**
  - Coppie chiave – valore
  - proprietà del sistema o dell'applicazione
  - Si usano per configurare il sistema
- **<File>.war**
  - Web application archive
- **File>.<ear**
  - Enterprise Web application archive

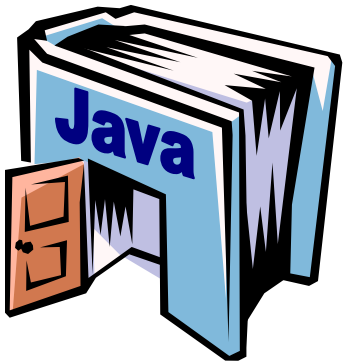
Programmi Java

- Java è un linguaggio ad oggetti
  - L'unità minima di programmazione è la classe
  - Un programma Java utilizza una o più classi
- Per ora, consideriamo programmi formati da una sola classe
  - Un solo file sorgente
  - Il nome del file coincide con il nome della classe
  - L'estensione del file è ".java"

Formato generale

main(...): la porta sul mondo Java

```
public static void
    main(String[] args){
    /* istruzioni ... */
}
```



- Punto di ingresso di ogni applicazione
  - Invocato automaticamente dalla VM Java

Metodi statici

- Sequenze di istruzioni che svolgono un determinato compito
  - Hanno un **nome**
  - Possono avere dei **parametri**
  - Possono ritornare un **valore**
  - Dovrebbero avere un **commento**!

```java
public class Test {

   static int m1(int a, int b) {
      int c = a*a+b*b;
      return c;
   }

   public static
      void main(String[] args) {
         int i = m1(3,4);
         System.out.println(i);
      }
}
```

Commenti

- Servono a documentare un programma (o una sua parte)
  - Racchiusi da “***/*” e “***/***”
  - “***//***” inizia un commento che termina a fine riga
- Usati sempre troppo poco!

Variabili locali

- I metodi possono utilizzare variabili:
  - Servono a memorizzare risultati intermedi
  - Dichiarate nel corpo del metodo
  - Durata temporanea: quando il metodo ritorna, la variabile è distrutta

Tipi: una forma per i dati

- Variabili, parametri, valori ritornati hanno un **tipo** che ne precisa /limita l'uso
  - Possono contenere solo valori conformi al proprio tipo
- Esistono tipi
  - Semplici (**primitivi**)
  - Compositi (**classi**) – descritti nelle lezioni successive

Tipi primitivi

- Numerici interi
  - **byte, short, int, long**
- Numerici reali
  - **float, double**
- Non numerici
  - **boolean, char**

Tipi numerici interi (1)

- Valori ammissibili
  - **byte** {-128 ... 127}
  - **short** {-32768 ... 32767}
  - **int** {-2147483648 ... 2147483647}
  - **long** {-9.22 E18 ... 9.22E18}

Tipi numerici interi (2)

- Operazioni consentite
  - aritmetiche
    +, -, *, /, %, ++, --, >>
  - logiche (bitwise bit a bit)
    ^, &, |, ~, <<, >>>
  - confronto
    ==, !=, >, >=, <, <=

Tipi numerici reali

- Valori
  - **float** {±3.403E38}
    6 cifre significative
  - **double** {±1.798E308}
    15 cifre significative
- Operazioni
  - aritmetiche (**+**, **-**, ***, */**)
  - confronto (**==**, **!=**, **>**, **>=**, **<**, **<=**)

Tipi non numerici: boolean

- Valori
  - {**false**, **true**}
- Operazioni
  - Logiche short-circuit (**&&**, **||**, **!**)
  - confronto (**==**, **!=**)

Operatori bitwise /short-circuit

&	AND bitwise	A&B
|	OR bitwise	A|B
^	XOR bitwise	A^B
~	NOT bitwise	A~
&=	AND bitwise con assegnazione	A&=B
|=	OR bitwise con assegnazione	A|=B
^=	XOR bitwise con assegnazione	A^=B
>>	sposta n bit a destra con estensione	A>>n
>>=	sposta n bit a destra con estensione e assegna	A>>=n
<<	sposta bit a sinistra	A<<n
<<=	sposta bit a sinistra e assegna	A<<=n
>>>	sposta bit a destra senza segno	A>>>n
>>>=	sposta bit a destra senza segno e assegna	A>>>=n

&&	AND logico	A&&B
||	OR logico	A||B
!	NOT logico	A!

Tipi non numerici: char

- Rappresentazione dei caratteri secondo lo standard Unicode
  - Lettere di vari alfabeti
    arabo, armeno, bengalese, cirillico, greco, latino, ...
  - Simboli
    diacritici, punteggiatura, matematici, tecnici, ...

Tipi non numerici: char

- Rappresentazione su due byte
  - Le costanti sono indicate tra apici semplici (‘a’)
  - I valori non stampabili sono rappresentati nel formato unicode (‘\u27AF’)
- Operazioni
  - confronto (==, !=, >,>=, <, <=)

Operatori – riassunto 1

Gli operatori logici

&&	AND logico	A&&B
||	OR logico	A||B
^	XOR	A^B
!	NOT logico	A!
&	AND bitwise	A&B
|	OR bitwise	A|B
A?B:C	if A then B else C	A?B:C

Gli operatori di confronto

==	Uguale a	A==B
<	Minore di	A<B
>	Maggiore di	A>B
<=	Minore o uguale a	A<=B
>=	Maggiore o uguale a	A>=B
!=	Diverso da	A!=B

Gli operatori di incremento e decremento

++	Incremento	A++ ++AA
--	Decremento	A-- --A

Operatori – riassunto 2

Gli operatori matematici

+	Somma	A+B
-	Sottrazione	A-B
*	Moltiplicazione	A*B
/	Divisione	A/B
%	Modulo	A%B

Gli operatori di assegnazione

=	Assegnazione singola	A=B
+=	Somma e assegnazione	A+=B
-=	Sottrazione e assegnazione	A-=B
*=	Moltiplicazione e assegnazione	A*B
/=	Divisione e assegnazione	A/B
%=	Modulo e assegnazione	A%=B

Operatori – riassunto 3

Gli operatori bitwise

&	AND bitwise	A&B
|	OR bitwise	A|B
^	XOR bitwise	A^B
~	NOT bitwise	A~
&=	AND bitwise con assegnazione	A&=B
|=	OR bitwise con assegnazione	A|=B
^=	XOR bitwise con assegnazione	A^=B
>>	sposta bit a destra con estensione	

>>=	sposta bit a destra con estensione e assegna	
<<	sposta bit a sinistra	
<<=	sposta bit a sinistra e assegna	
>>>	sposta bit a destra senza segno	
>>>=	sposta bit a destra senza segno e assegna	

Operatori – riassunto 4

La gerarchia degli operatori

. [] ()
++ -- ! ~ instanceof
* / %
+ -
<< >> >>>>
< > <= >=
== !=
&
^
|

&&
||
? :
= += -= *= /= &= |= ^= %= >>= <<= >>>>=
,

Esempi operatori

```
package esempiVari;

public class TestJavaOperators {

    public static void main(String[] args) {

        // AND bitwise
        int a=60;             // = 0011 1100
        int b=13;             // = 0000 1101
        int c = a&b;          // = 0000 1100 ossia c=12
        System.out.println("1) a&b="+c);

        // OR Bitwise
        c=a|b; // = 0011 1101 ossia c=61
        System.out.println("2) a|b="+c);

        // NOT Bitwise
        c=~a;                // = 1100 0011 ossia c=-61
        System.out.println("3) ~a="+c);

        // << shift bitwise sinistra (moltiplica)
        a=60;                // = 0011 1100
        c=a<<2;              // = 1111 0000 ossia c=240
        System.out.println("4) a<<2="+c);

        // >> shift bitwise destra (divide)
        a=60;                // = 0011 1100
        c=a>>2;              // = 0000 1111 ossia c=15
        System.out.println("5) a>>2="+c);

        // >>> shift bitwise destra con segno (divide)
        a=60;                // = 0011 1100
        c=a>>>2;             // = 0000 1111 ossia c=15
        System.out.println("6) a>>>2="+c);

    }
}
```

Usare le variabili (1)

- Si può dichiarare una variabile ovunque nel codice di un metodo
  - Bisogna farlo, però, prima di usarla
  - Visibilità limitata al blocco di istruzioni in cui è dichiarata
- Le variabili devono essere inizializzate!
  - int a;
  - int b = a;

Errore!
La variabile “a” non è stata inizializzata, e il suo valore è casuale

Usare le variabili (2)

- Ad una variabile può essere assegnato:
  - Un valore costante
  - Il risultato di una formula
- Esempi
  - int i = **1**, j = **5**;
  - i = **(5*j) % 10**;
  - long l = **10L**;
  - char c1 = **‘a’**, c2 = **‘\u0062’**;
  - boolean b = **false**;

Convenzioni sui nomi

- Le **classi** hanno nomi che iniziano con la lettera **maiuscola**
- **Metodi**, **attributi** e **variabili** hanno l'iniziale **minuscola**
  - Se un nome è composto da più parole giustapposte, l'iniziale di ogni parola successiva alla prima è maiuscola
  - int contatoreNumeroOccorrenze = 0;
  - class CentraleTelefonica { }

Costrutti di programmazione

- Istruzioni semplici
  - Scelta
  - Ripetizione
  - Salto strutturato
- Invocazione di metodi
  - Trattata più in dettaglio successivamente
- Blocchi di istruzioni
  - Racchiusi tra parentesi graffe **{...}**

Istruzioni di scelta (1)

```
if (condizione)...
if (condizione)... else ...
```

- Verifica della condizione
  - **Vera** si esegue l'istruzione o il blocco di istruzioni successivo
  - **Falsa** si salta all'istruzione successiva oppure viene eseguito il blocco "**else**"

Istruzioni di scelta (2)

```
switch (var) {
    case val1: … break;
    case valN: … break;
    default: …
}
```

- Struttura di selezione multipla
  - Valutazione del valore di una variabile
  - `val1`, …, `valN`:
    espressioni costanti (interi o caratteri)

Istruzioni di ripetizione (1)

```
for(espr1; espr2; espr3){
  … //istruzioni da ripetere
}
```

- ***espr1***: inizializzazione variabile/i di controllo
- ***espr2***: condizione di continuazione
- ***espr3***: modifica della/e variabili di controllo

Istruzioni di ripetizione (2)

```
while (cond ){
  …
}
```

```
do {
  …
} while(cond );
```

- Esecuzione di un blocco di istruzioni finché la condizione rimane vera
- do/while garantisce almeno una iterazione

Istruzioni di ripetizione forEach (3)

```
for (element:structure ){
  …
}
```

- Esecuzione di un blocco di istruzioni per ogni elmento (primitivo o oggetto) in struttura (array o iterable)
- La struttura può essere un Array, Un Vector, Una Collection

  Quale ArrayList, Map, List etc.

Salto strutturato

- Istruzioni che permettono di alterare il flusso di esecuzione:
  - **`break`** : provoca l'uscita immediata dal blocco
  - **`continue`**: salto delle rimanenti istruzioni del blocco, procede con l'interazione successiva del ciclo
  - Usati nei costrutti **`while`**, **`do/while`**, **`for`**, **`switch`** (quest'ultimo, solo *break*)

Esempi if/switch/while/for