

SUPSI

# Primi Programmi

Loris Grossi, Fabio Landoni, Andrea Baldassari

Contenuto realizzato in collaborazione con: T. Leidi, A.E. Rizzoli, S. Pedrazzini

Fondamenti di Informatica

Bachelor in Ingegneria Informatica

# Obiettivo

Essere in grado di scrivere, compilare ed eseguire semplici programmi Java che utilizzano delle istruzioni di input, di output, di selezione e di ripetizione.

## Obiettivi della lezione:

- Conoscere la struttura di base di un programma Java.
- Conoscere gli elementi che compongono un programma Java.
- Conoscere le istruzioni di input e di output.
- Conoscere alcuni tipi di dato: int, double, String.
- Iniziare ad utilizzare istruzioni di selezione (`if .. else`) e di ripetizione (`while`).
- Saper compilare un programma Java.
- Saper eseguire un programma Java.

# Esempio di programma in Java

```
/**  
 * Esempio di programma in Java  
 */  
public class OperazioniAritmetiche {  
    public static void main(String[] args) {  
        int result = 1 + 2;  
        System.out.println(result);  
  
        result = result - 1;  
        System.out.println(result);  
  
        result = result * 2;  
        System.out.println(result);  
  
        result = result / 2;  
        System.out.println(result);  
    }  
}
```

# Compilazione ed esecuzione

Ma come si fa a compilare ed eseguire un programma Java?

# JRE e JDK

## Java Runtime Environment (JRE):

- È un ambiente che permette di eseguire applicazioni Java. Contiene la Java Virtual Machine (JVM) e le Java APIs.

## Java Development Kit (JDK):

- Collezione di tools per la programmazione in Java: compilatore, debugger e vari strumenti per analizzare il comportamento e le performances delle applicazioni.
- Librerie aggiuntive per lo sviluppo di programmi Java.
- Applicazioni e codice esempio.
- Codice sorgente delle Java APIs.
- Comprende una versione della JRE.

# Editore Visual Studio Code

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Buongiorno.java - lezione - Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer Sidebar:** Shows the file structure under LEZIONE:
  - Buongiorno.java (selected)
  - Buongiorno.class
  - Buongiorno.java
- Code Editor:** Displays the following Java code:

```
1 import java.util.Scanner;
2
3 public class Buongiorno {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6
7         System.out.print("Come ti chiami? ");
8         String nome = input.nextLine();
9         System.out.println("Buongiorno " + nome + " e benvenuto al corso");
10
11        System.out.println();
12        System.out.print("Inserisci due numeri interi: ");
13        int numero1 = input.nextInt();
14        int numero2 = input.nextInt();
15        input.close();
16
17        System.out.println(numero1 + " + " + numero2 + " = " + (numero1 +
18        System.out.println(numero1 + " - " + numero2 + " = " + (numero1 -
19        System.out.println(numero1 + " * " + numero2 + " = " + (numero1 *
20    }
21 }
22 }
```
- Terminal:** Shows the command-line output:

```
Lando@pop-os:~/Desktop/lezione$ javac Buongiorno.java
Lando@pop-os:~/Desktop/lezione$ java Buongiorno
Come ti chiami? ■
```
- Bottom Status Bar:** Ln 3, Col 24 (10 selected) Spaces: 4 UTF-8 LF Java ⚡ ⚡

# Esempio di programma in Java

```
import java.util.Scanner;

public class Buongiorno {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Come ti chiami? ");
        String nome = input.nextLine();
        System.out.println("Buongiorno " + nome + " e benvenuto al corso.");

        System.out.println();
        System.out.print("Inserisci due numeri interi: ");
        int numero1 = input.nextInt();
        int numero2 = input.nextInt();
        input.close();

        System.out.println(numero1 + " + " + numero2 + " = " + (numero1 + numero2));
        System.out.println(numero1 + " - " + numero2 + " = " + (numero1 - numero2));
        System.out.println(numero1 + " * " + numero2 + " = " + (numero1 * numero2));
    }
}
```

# Programmi

Nel codice sorgente, ogni programma Java è rappresentato da **una classe**.

La classe è il contenitore del programma.

Il codice sorgente di un programma deve essere salvato all'interno di un file, con estensione **.java**, contenente **la classe**.

**Il nome del file deve coincidere con quello della classe.**

**Il file .java va compilato utilizzando il comando `javac`.**

# Programmi

Il risultato della compilazione di un file `.java` è un file con lo stesso nome, ma con estensione `.class`. Quindi, anche il nome del programma eseguibile coincide con quello della classe.

I programmi Java vengono eseguiti tramite la Java Virtual Machine (JVM) messa a disposizione dalla JRE.

Per lanciare l'esecuzione di un programma Java si utilizza il comando `java`.

## Riassumendo

Per compilare un programma Java bisogna utilizzare il comando `javac`:

```
javac IlMioProgramma.java
```

Per eseguire un programma bisogna utilizzare il comando `java`:

```
java IlMioProgramma
```

# Esempio di compilazione

```
user@ubuntu: ~/Formazione/Info/Esempi
user@ubuntu:~/Formazione/Info/Esempi$ ls
Buongiorno.java
user@ubuntu:~/Formazione/Info/Esempi$ javac Buongiorno.java
user@ubuntu:~/Formazione/Info/Esempi$ ls
Buongiorno.class  Buongiorno.java
user@ubuntu:~/Formazione/Info/Esempi$ █
```

## Esempio di esecuzione

```
user@ubuntu: ~/Formazione/Info/Esempi
user@ubuntu:~/Formazione/Info/Esempi$ ls
Buongiorno.class  Buongiorno.java
user@ubuntu:~/Formazione/Info/Esempi$ java Buongiorno
Come ti chiami? Luca
Buongiorno Luca e benvenuto al corso.

Inserisci due numeri interi: 13 2
13 + 2 = 15
13 - 2 = 11
13 * 2 = 26
user@ubuntu:~/Formazione/Info/Esempi$ █
```

# Struttura di un programma in Java

```
/**  
 * Commento di classe (opzionale)  
 */  
  
public class NomeProgramma {  
    public static void main(String[] args) {  
        // Riga di commento (opzionale)  
  
        dichiarazioni;  
        istruzioni;  
    }  
}
```

# Java keywords

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

# Commenti

In Java si possono utilizzare due tipi di commenti:

- **Commento end-of-line** (o per singola riga): **inizia con i caratteri //**. Tutto quello che si trova a destra dei caratteri // fino alla fine della riga viene considerato commento ed ignorato (come in C++).

```
// Commento end-of-line  
int raggio = 25; // Commento
```

- **Commento multiriga**: **inizia con la sequenza di caratteri /\* e termina con la sequenza \*/**. Tutto quello che è racchiuso tra queste sequenze di caratteri viene considerato commento ed ignorato (come in C e C++).

```
/* Commento multiriga che si  
estende sulla seconda riga  
ma anche sulla terza */  
int area = 0;
```

# Classe di un programma

Per definire la **classe di un programma** si deve scrivere `public class` seguito dal nome della classe.

La classe di un programma Java “deve” contenere la **procedura main**.

```
/**  
 * Esempio di programma Java  
 */  
public class ProgrammaEsempio {  
  
    /**  
     * Procedura main  
     */  
    public static void main(String[] args) {  
        // Aggiungere qui il codice del programma  
    }  
}
```

## Procedura main

La procedura `main` di Java va definita come segue:

```
public static void main(String[] args)
```

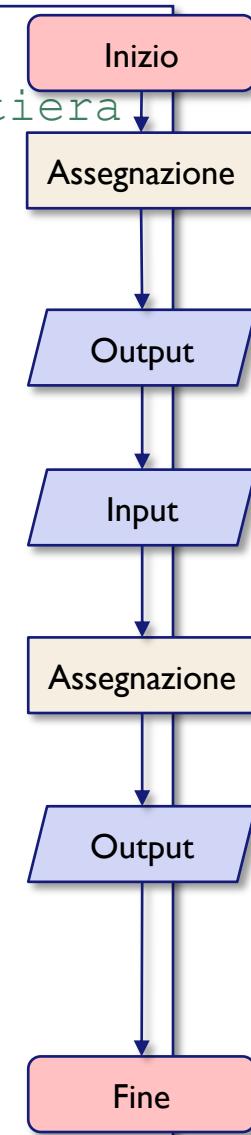
In seguito, va specificato il blocco di codice contenente il programma.

La procedura `main` è una procedura particolare e rappresenta il **punto di partenza del programma Java**.

Il blocco della procedura `main` può contenere una serie di dichiarazioni di variabili e di istruzioni che vengono eseguite in sequenza.

# Esempio

```
public static void main(String[] args) {  
    // Inizializza lo scanner per leggere dati da tastiera  
    Scanner input = new Scanner(System.in);  
  
    // Mostra a schermo la domanda "Come ti chiami?"  
    System.out.print("Come ti chiami? ");  
  
    // Legge il nome da tastiera  
    String nome = input.nextLine();  
  
    // Crea la frase di saluto  
    String saluto = "Ciao " + nome + "!";  
  
    // Mostra a schermo la frase finale  
    System.out.println(saluto);  
  
    // Finalizza lo scanner  
    input.close();  
}
```



## Blocchi di codice

I blocchi di codice vengono **specificati con le parentesi graffe ‘{’ e ‘}’** e servono a **raggruppare parti di codice sorgente**. In alcuni casi sono obbligatori, in altri casi sono opzionali.

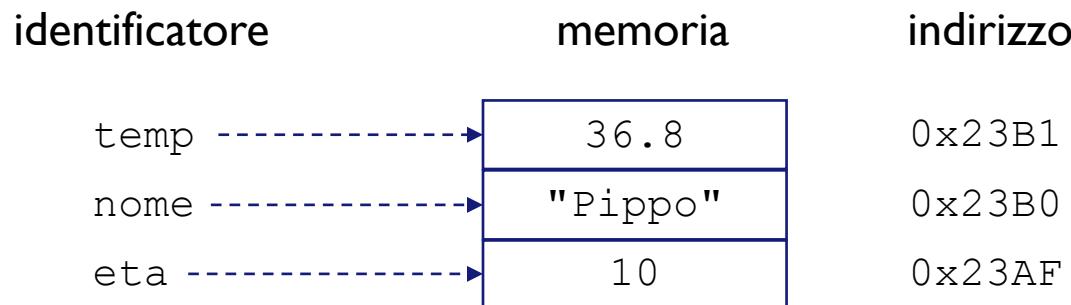
All'interno dei blocchi è buona abitudine **indentare il codice**.

## Esempio blocchi di codice

```
public class BlocchiDiCodice {
    public static void main(String[] args) {
        int valore = 0;
        {
            valore++;
            System.out.println(valore);
            {
                int nuovoValore = 2;
                nuovoValore++;
                System.out.println(nuovoValore);
            }
        }
        System.out.println("valore: " + valore);
    }
}
```

# Variabili

Una **variabile** è un elemento del programma che, in momenti diversi, può assumere valori differenti. Ad ogni variabile è associata una zona di memoria.



All'interno del programma, l'identificatore della variabile permette di leggere e/o modificare, durante l'esecuzione, il valore associato.

## Variabili

Prima di poter essere utilizzata, una variabile deve essere dichiarata. Ogni dichiarazione di variabile, in Java, è composta da un **tipo di dato** e da un **identificatore** (nome).

Quando viene dichiarata, la variabile può anche essere inizializzata.

La variabile è accessibile, tramite l'identificatore, dal momento in cui viene dichiarata e fino alla fine del blocco in cui essa è contenuta.

```
int abc; // non inizializzata
int x = 20 + 30;

int y = 0, z;
y = x * 4;
z = x * 8;

double d = z / (y * 1.75);
String mioNome = "Pippo";
```

## Esempio variabili

```
public class UsoDiVariabili {  
    public static void main(String[] args) {  
        int valore = 0;  
        System.out.println("Valore iniziale: " + valore);  
  
        valore = valore + 1;  
        System.out.println("Valore incrementato: " + valore);  
  
        String str1 = "Buongiorno";  
        String str2 = "studenti";  
  
        double altroValore = valore + 5.316;  
  
        System.out.println(str1 + " " + str2 + " il valore "  
                           + "finale è: " + altroValore);  
    }  
}
```



# Costanti

Per le **costanti**, in Java bisogna utilizzare le **variabili static final** da dichiarare al di fuori (di regola prima) della procedura main.

```
static final int LA_MIA_COSTANTE = 42;
```



## Esempio con costanti

```
public class Sfera {  
  
    static final double PI = 3.141592653589793;  
  
    public static void main(String[] args) {  
        double raggio = 4.;  
        double vol = (4. / 3.) * PI * raggio * raggio * raggio;  
        System.out.println(vol);  
    }  
}
```

# Identifieri

Gli identifieri sono i **nomi delle variabili, delle procedure, delle classi, ...**

Un identificatore è una **sequenza senza spazi** e di lunghezza illimitata di **lettere, cifre** e i simboli ‘\_’ e ‘\$’. All’inizio della sequenza ci deve essere una lettera o un simbolo (‘\_’ o ‘\$’). Inoltre, un identificare non può essere identico ad una keyword, ad un valore booleano (true o false) o al valore null.

Esempi:

**OK:** TheName, theName, THENAME, \_thename, \$theName, x, i

**KO:** The Name, 9Name, 9, cip+ciop, Alpha&Beta

# Identifieri

Java è case sensitive!

L'identificatore `HelloWorld` è diverso da `HELLOWORLD`, diverso da `helloworld`, diverso da `hELLOWORLD`, e così via.

Per convenzione, in Java:

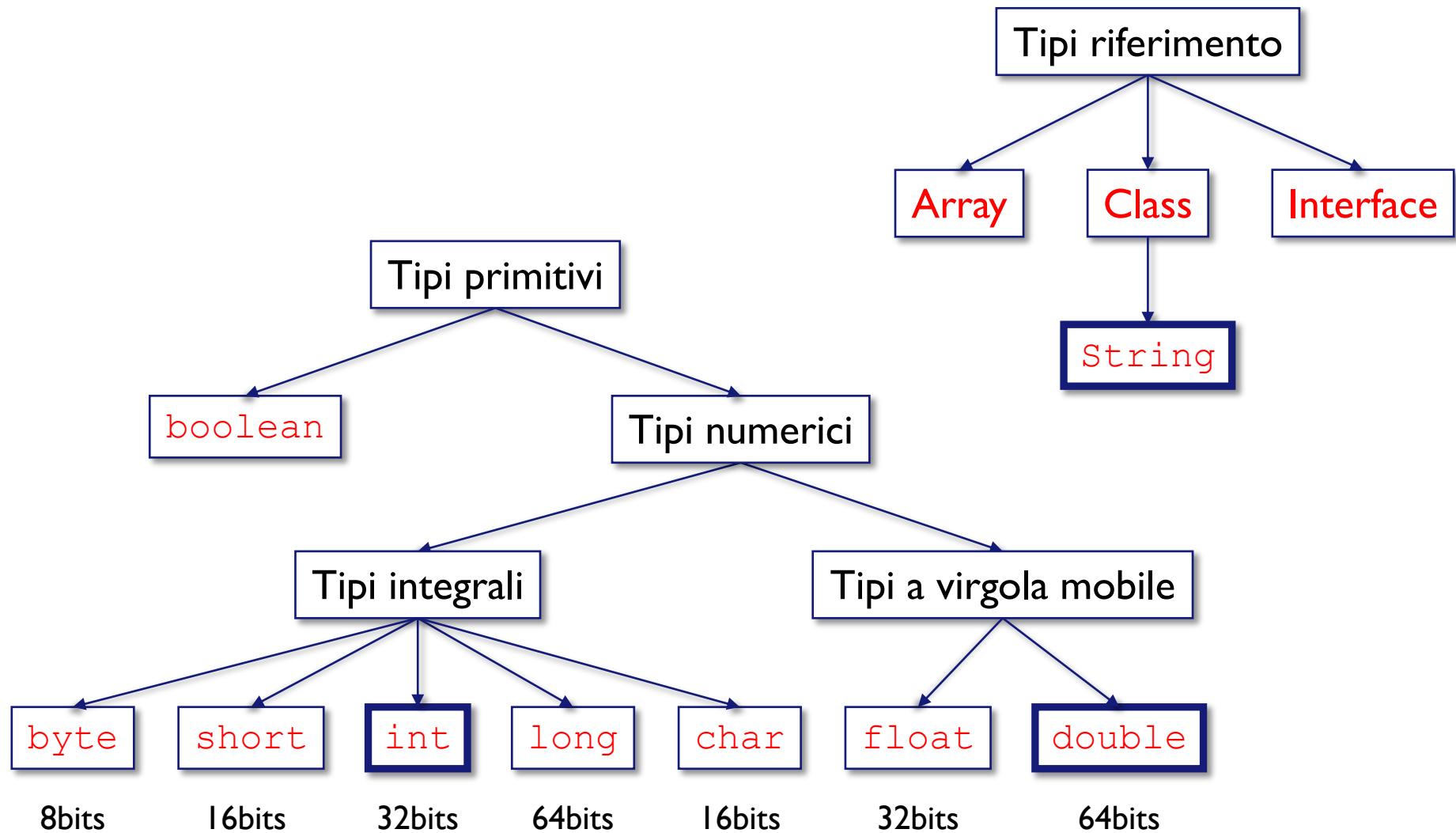
- Per i **nomi delle classi** è abitudine utilizzare la sintassi **CamelCase** (notazione a cammello) con la **prima lettera maiuscola** (ad esempio: `NomeDellaClasse`).
- Per i **nomi delle variabili** è abitudine utilizzare la sintassi **CamelCase** (notazione a cammello) con la **prima lettera minuscola** (ad esempio: `nomeDellaVariabile`).

## Tipi di dato

“Il linguaggio di programmazione Java è **strongly typed**. Questo significa che ogni variabile e ogni espressione **ha un tipo conosciuto al momento della compilazione**. I tipi **limitano i valori** che una variabile può assumere o che un’espressione può produrre, **limitano le operazioni** supportate su quei valori e determinano il significato di quelle operazioni. Lo strong typing aiuta ad **individuare gli errori** durante la compilazione.”

— fonte: *Java language specification*

# Tipi di dato



# Operatori

## *Operatore di assegnazione semplice*

= Assegnazione

## *Operatori aritmetici*

+ Somma  
Concatenazione di stringhe

- Sottrazione

\* Moltiplicazione

/ Divisione

% Resto della divisione

## *Altri operatori*

- Negativo

++ Incremento

-- Decremento

## *Operatori d'egualanza e relazionali*

== Uguale a

!= Differente da

> Maggiore di

>= Maggiore o uguale a

< Minore di

<= Minore o uguale a

## *Operatori logici (booleani)*

! Negazione (NOT)

&& AND

|| OR

Fonte:

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/opsummary.html>

# Separatori

- ( ) Raggruppare i parametri nelle definizioni e chiamate dei metodi.  
Specificare le precedenze nelle espressioni aritmetiche.  
Usate intorno ai type casts.  
Delimitare le condizioni d'esecuzione delle istruzioni per il controllo del flusso di codice (ad esempio if e while).
- { } Definire blocchi di codice.  
Inizializzare automaticamente gli arrays.
- [ ] Dichiarare e dereferenziare gli arrays.
- ; Terminatore di istruzione.  
Utilizzato nel for per delimitare le varie espressioni.
- , Separare le dichiarazioni di variabili dello stesso tipo.  
Utilizzato nelle espressioni dei for loops.
- . Selezionare un campo o un metodo di un oggetto.  
Separare i nomi dei packages e delle classi.
- : Utilizzato nello switch e dopo le loop labels.

## Letterali

“Un **letterale** è la rappresentazione in codice sorgente di un valore di tipo primitivo, una stringa o il valore null.”

— fonte: *Java language specification*

Esempi:

```
0  
34528  
22.5  
0x00FF00FF  
5e-28  
true  
'm'  
"Hello Students"  
null
```

# Espressioni

Un'espressione è un costrutto (porzione di codice) che, quando viene elaborato, assume un singolo valore. Può essere composta da: variabili, operatori, letterali, separatori, invocazioni di funzioni.

Esempi:

```
result = 100 + x // Tipo int
theVar == 500 // Tipo boolean (true o false)
isBlue && isDark // Tipo boolean (true o false)
"Current is: " + pos // Tipo String
myVect[20] = 1.5 // Tipo double
val = cos(x) + 1 // Tipo double
```

Il tipo di dato di un'espressione dipende dagli elementi che la compongono.

# Assegnazione

L'assegnazione è l'operazione tramite la quale si assegna il valore di un'espressione ad una variabile.

Esempi:

```
result = 100 + x;  
val = cos(x) + 1;
```

Il tipo della variabile deve coincidere (o essere compatibile) con il tipo dell'espressione.

Esempi:

```
double voto = 4.25; // OK  
int valutazione = "Ottimo"; // Errore!!
```

# Precedenze

Priorità	Operatori	Operazione	Associatività
1	[ ] ( ) . . ++ --	Accesso elemento di un array Invocazione metodo Accesso ad un membro di un oggetto Post-incremento e post-decremento	Sinistra
2	++ -- + - ~ !	Pre-incremento e pre-decremento + / - unario Bitwise NOT Boolean NOT (logico)	Destra
3	(tipo) new	Type cast Creazione oggetto	Destra
4	* / %	Moltiplicazione, divisione, resto	Sinistra
5	+ - +	Somma, sottrazione Concatenazione di stringhe	Sinistra
6	<< >> >>>	Bit shift a sinistra (signed) Bit shift a destra (signed) Bit shift a destra (unsigned)	Sinistra

# Precedenze

Priorità	Operatori	Operazione	Associatività
7	< <= > >= instanceof	Minore di, minore o uguale a Maggiore di, maggiore o uguale a Controllo tipo referenza	Sinistra
8	== !=	Uguale a Diverso da	Sinistra
9	&	Bitwise AND, boolean AND (logico)	Sinistra
10	^	Bitwise XOR, boolean XOR (logico)	Sinistra
11		Bitwise OR, boolean OR (logico)	Sinistra
12	&&	Boolean AND (logico)	Sinistra
13		Boolean OR (logico)	Sinistra
14	? :	Operatore ternario	Destra
15	= *= /= += -= %= <<= >>= >>>= &= ^=  =	Assegnazione Assegnamento combinato (operazione ed assegnamento)	Destra

# Istruzioni

L'istruzione è l'**unità di esecuzione** in Java.

Ogni istruzione **deve essere completata con il punto e virgola (';')**, fatta eccezione per le istruzioni di controllo del flusso di codice.

Le istruzioni si suddividono in:

- **Istruzioni di dichiarazione**: utilizzate per dichiarare le variabili.
- **Istruzioni d'espressione**: espressioni d'assegnazione, invocazione di procedure o funzioni, espressioni di incremento o di decremento (++ o --), espressioni per la creazione d'oggetti.
- **Istruzioni per il controllo del flusso del codice**: utilizzate per controllare l'esecuzione del programma (ad esempio if e while).

## Esempio di istruzioni

```
/**  
 * Esempio di programma con varie istruzioni  
 */  
public class Istruzioni {  
    public static void main(String[] args) {  
        int valore = 0, j = 0;  
        while (j < 30) {  
            if (j == 9) {  
                valore++;  
                System.out.println("10");  
            } else if (j == 19) {  
                valore++;  
                System.out.println("20");  
            }  
            j++;  
        }  
        System.out.println("Valore finale: " + valore);  
    }  
}
```

## Istruzioni per l'output

Per l'**output di testo su console** bisogna utilizzare l'istruzione:

```
System.out.print(testo);
```

oppure:

```
System.out.println(testo);
```

La seconda va a capo automaticamente alla fine dell'output.

testo deve essere una variabile di tipo String, un letterale di tipo String (fra virgolette, ad esempio "Hello world"), oppure una variabile o un letterale di qualsiasi altro tipo di dato convertibile in tipo String.

## Istruzioni per l'input

Per l'**input di testo o di valori numerici da console** si utilizza:

```
import java.util.Scanner;
// ...
Scanner input = new Scanner(System.in);
String testo = input.nextLine();
int numero = input.nextInt();
double numero2 = input.nextDouble();
// ...
input.close();
```

**Importante:** è necessario inizializzare e finalizzare lo Scanner all'inizio e alla fine del programma.

## Esempio di input e di output

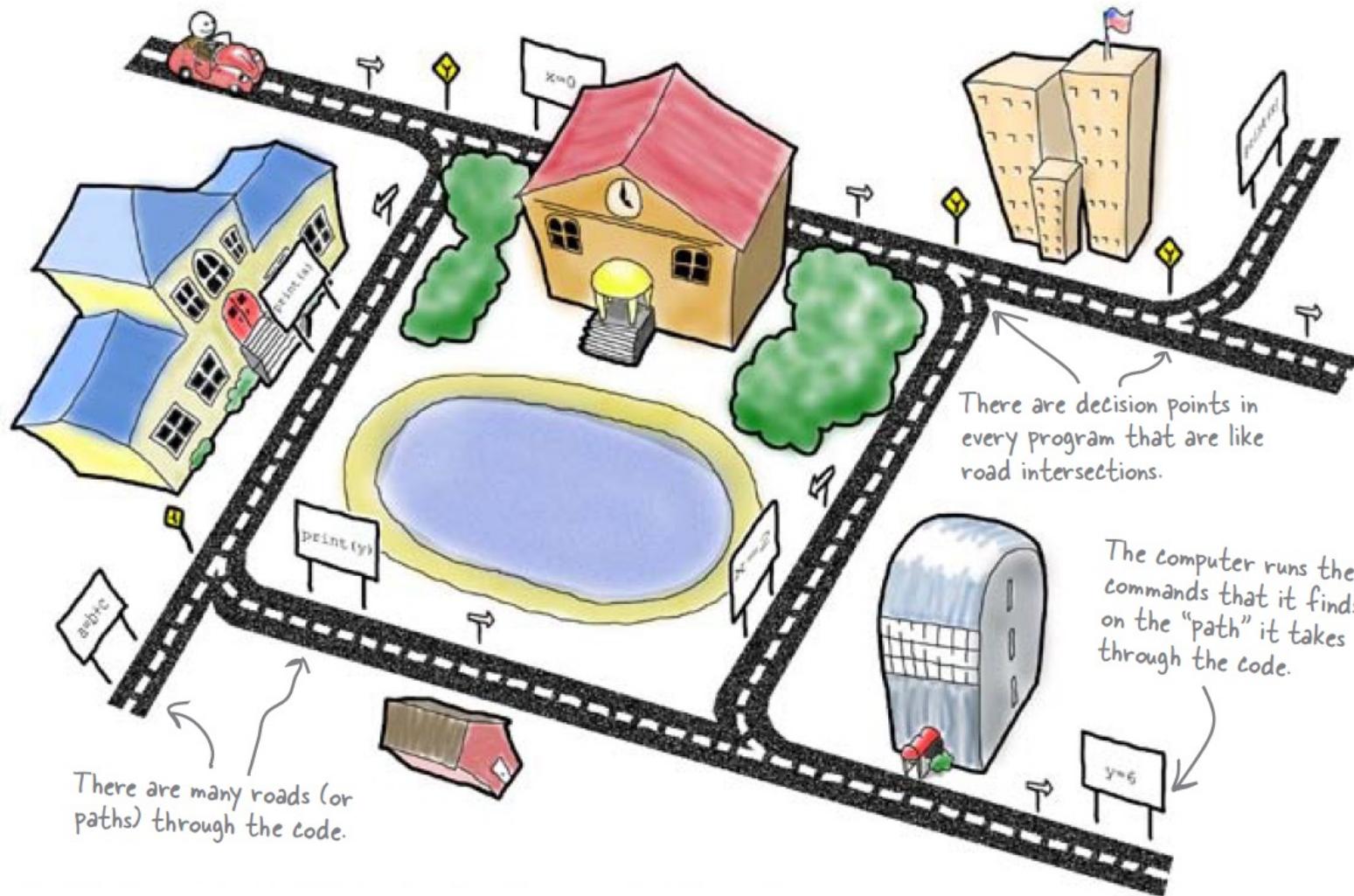
```
import java.util.Scanner;

/**
 * Programma di esempio per le istruzioni di input e di output
 */
public class InputOutput {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Inserire una stringa ed "
                + "un numero: ");
        String s = input.nextLine();
        int x = input.nextInt();

        System.out.println("La stringa è " + s);
        System.out.println("Il numero è " + x);

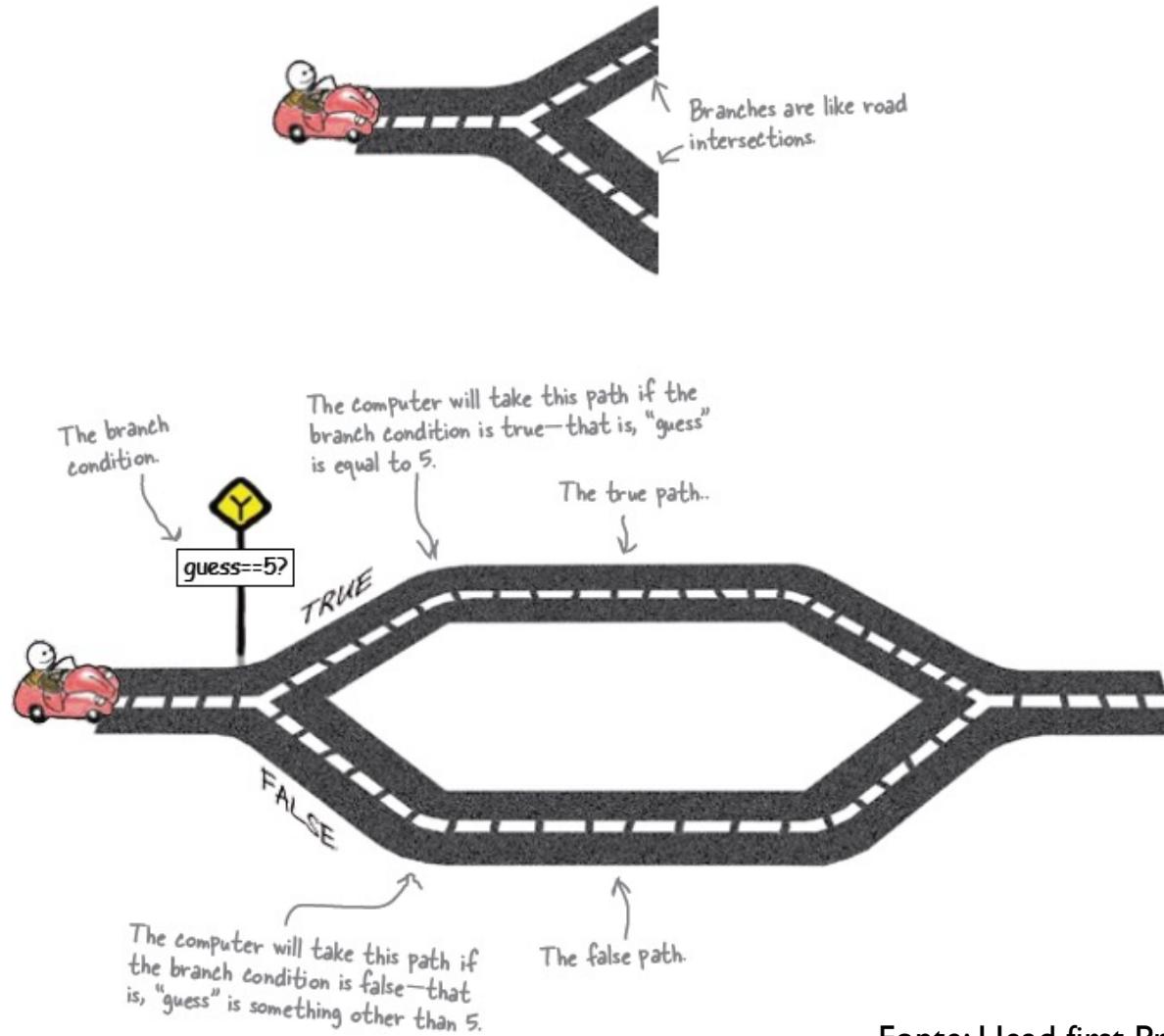
        input.close();
    }
}
```

# Istruzioni di controllo



Fonte: Head first Programming, O'Reilly

## L'istruzione di selezione



## L'istruzione di selezione if

```
if (condizione1) {  
    sequenzaIstruzioni1;  
} else if (condizione2) {  
    sequenzaIstruzioni2;  
} else {  
    sequenzaIstruzioni3;  
}
```

Opzionale

Opzionale, possono  
essercene più di una

## Esempio istruzione if

```
import java.util.Scanner;

public class IstruzioneIf {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Inserire un numero "
            + "minore di 1000: ");
        int x = input.nextInt();

        if (x > 1000) {
            System.out.println("Errore: il numero "
                + "inserito è maggiore di 1000");
            x = 1000;
        }

        System.out.println("Il numero è: " + x);
        input.close();
    }
}
```

## Esempio istruzione if ... else

```
import java.util.Scanner;

public class Divisione {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Inserire due numeri: ");
        int a = input.nextInt();
        int b = input.nextInt();

        if (a % b == 0) {
            System.out.println(a + " è divisibile per " + b);
        } else {
            System.out.println(a + " non è divisibile "
                + "per " + b);
        }
        input.close();
    }
}
```

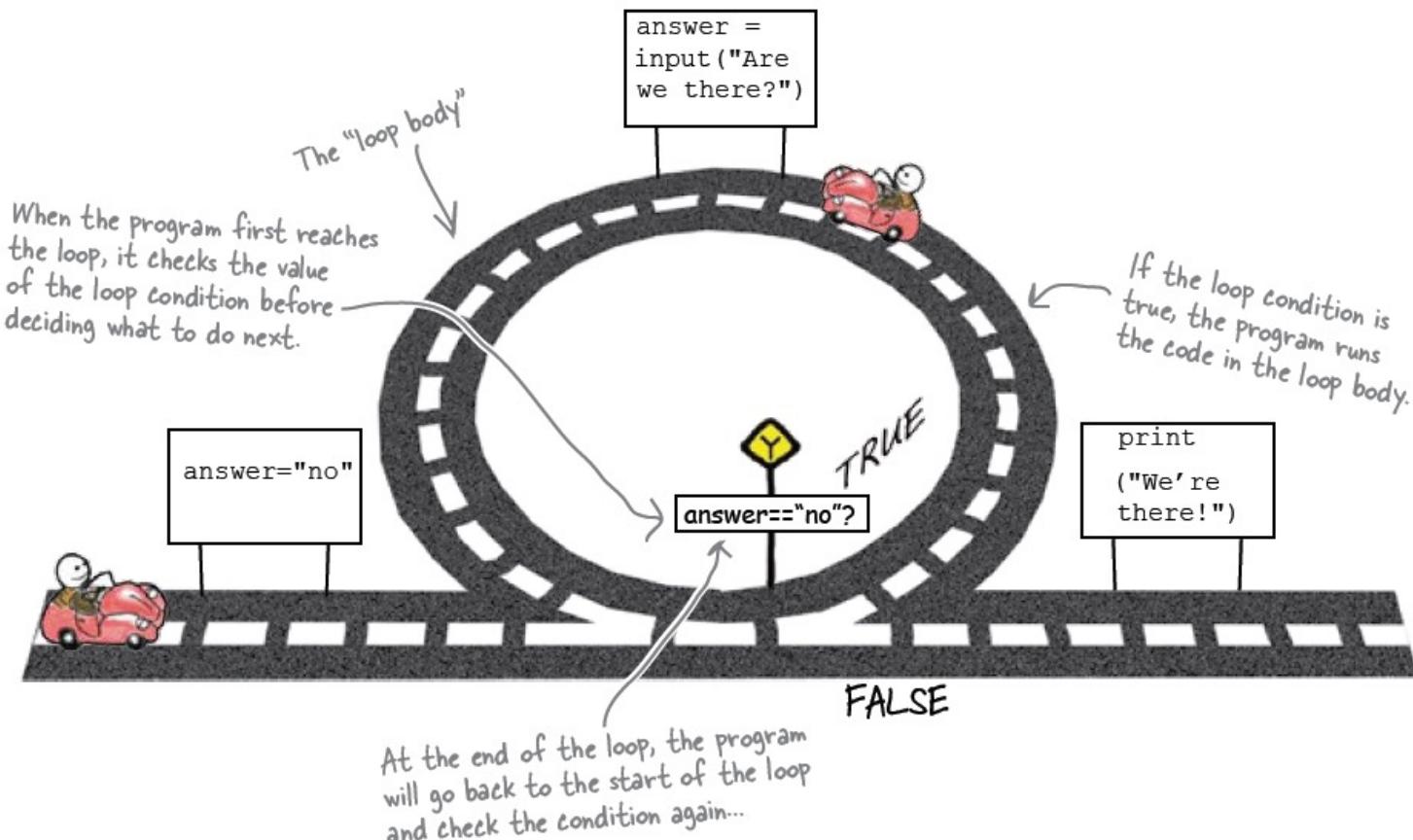
## Esempio istruzione if ... else if ... else

```
import java.util.Scanner;

public class IstruzioneIfElseIf {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Inserire un numero: ");
        int x = input.nextInt();

        if (x < 10) {
            System.out.print("Il numero è minore di 10");
        } else if (x < 20) {
            System.out.print("Il numero è minore di 20");
        } else if (x < 30) {
            System.out.print("Il numero è minore di 30");
        } else {
            System.out.print("Il numero è maggiore di 30");
        }
        input.close();
    }
}
```

# L'istruzione di ripetizione



## L'istruzione di ripetizione while

```
while (condizione1) {  
    sequenzaIstruzioni;  
}
```

## Esempio istruzione while

```
public class SommaNumeri {  
    public static void main(String[] args) {  
        // Stabilire il limite  
        int limite = 50;  
  
        System.out.println("Calcolo della somma dei numeri");  
        int somma = 0;  
        int x = 0;  
        while (x <= limite) {  
            somma = somma + x;  
            x++;  
        }  
  
        System.out.print("Risultato: " + somma);  
    }  
}
```

## Esempio istruzioni while e if

```
import java.util.Scanner;

public class IstruzioneWhileEIf {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Inserire un numero: ");
        int x = input.nextInt();
        int y = 1;
        while (y < 10) {
            x = x * y;
            if (x > 1000) {
                System.out.println("Il risultato parziale è > 1000");
                x = 1;
            }
            y++;
        }
        System.out.println("Risultato: " + x);
        input.close();
    }
}
```

# Riepilogo

- Compilazione ed esecuzione
- Struttura di un programma
- Java keywords
- Commenti
- Procedura `main`
- Blocchi di codice
- Variabili, costanti e identificatori
- Tipi di dato (`int`, `double` e `String`)
- Operatori e precedenze
- Letterali ed espressioni
- Istruzioni per l'input e l'output
- Istruzione di selezione (`if`)
- Istruzione di ripetizione (`while`)