



SAPIENZA  
Università di Roma

# Fondamenti di Informatica

**Luca Dan Serbanati**

# Cosa è un computer

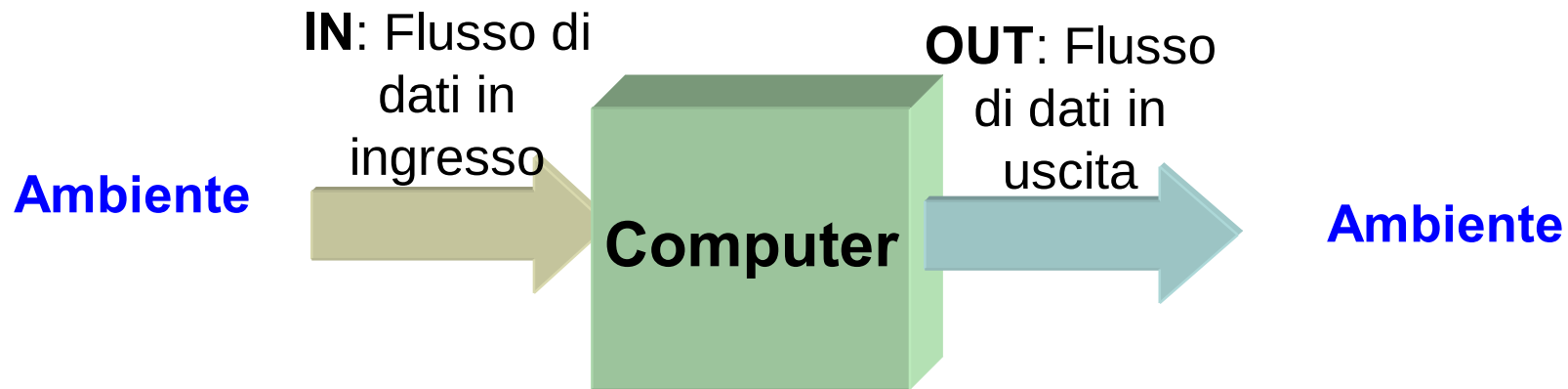
*Computer* è un dispositivo che:

- immagazzina informazioni rappresentate come *dati*;
- interagisce con l'ambiente attraverso di dispositivi esterni;
- esegue programmi che elaborano i dati immagazzinati in memoria.

I computer possono:

- eseguire in pochi secondi numerose operazioni ripetitive;
- controllare la temperatura in un appartamento o
- collegarsi con fonti di informazioni remote.

# Schema di un computer



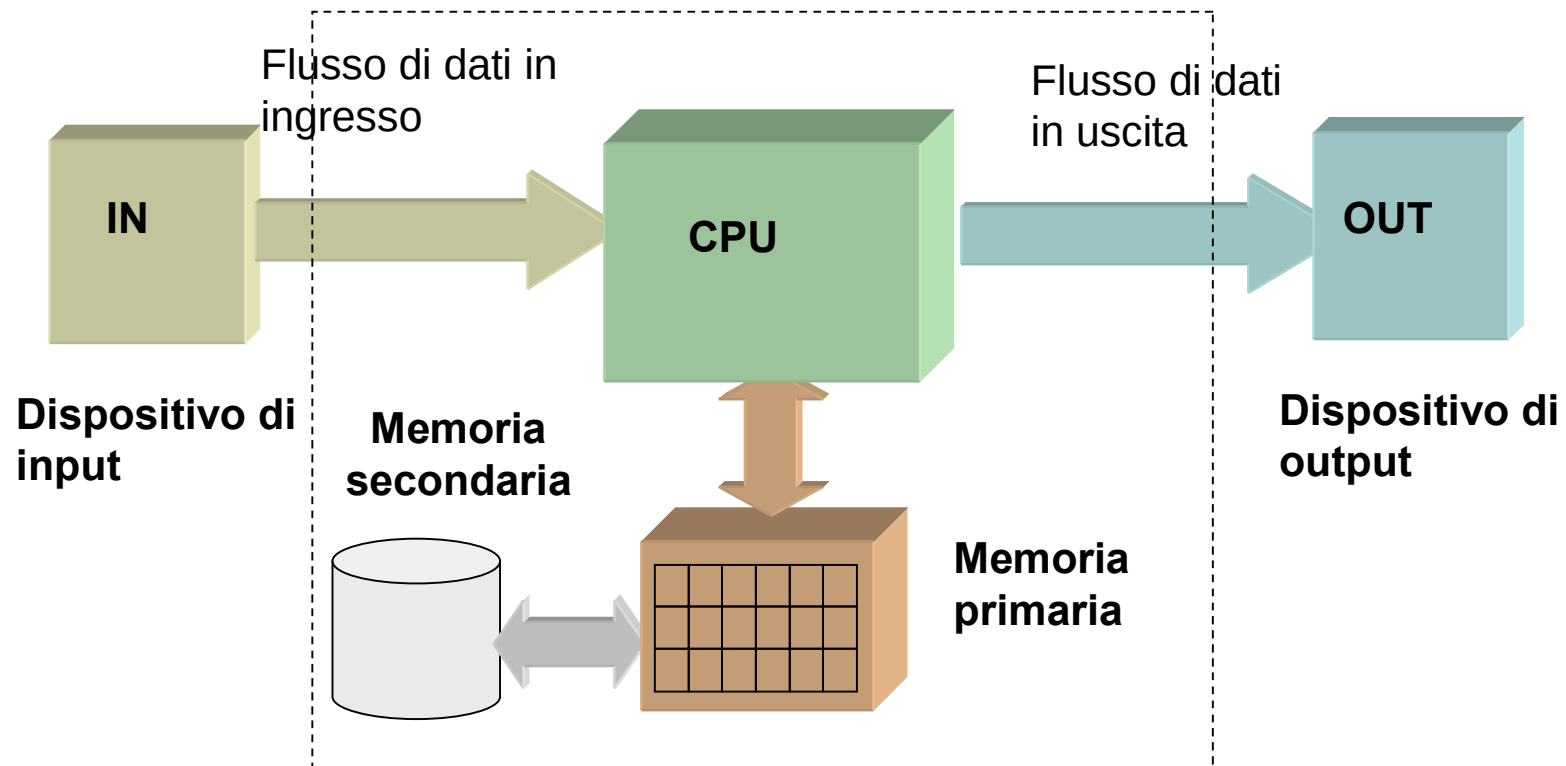
- Un computer riceve dati dall'ambiente, li elabora attraverso un *calcolo* e li fornisce così elaborati all'ambiente.
- *Programmi* sono sequenze di istruzioni che il computer esegue per svolgere un'attività di calcolo.

# Istruzioni

Comandi semplici rivolte al computer:

- “sommare questi due numeri !”
  - “inviare la lettera M alla stampante !”
  - “estrarre il numero immagazzinato in questa posizione nella memoria!”
  - “visualizzare un punto rosso in questa posizione dello schermo!”
  - “se questo valore è positivo, continuare il programma da questa data istruzione!”
- Programmazione
  - Programmatore

# Hardware = CPU + Memoria + I/O

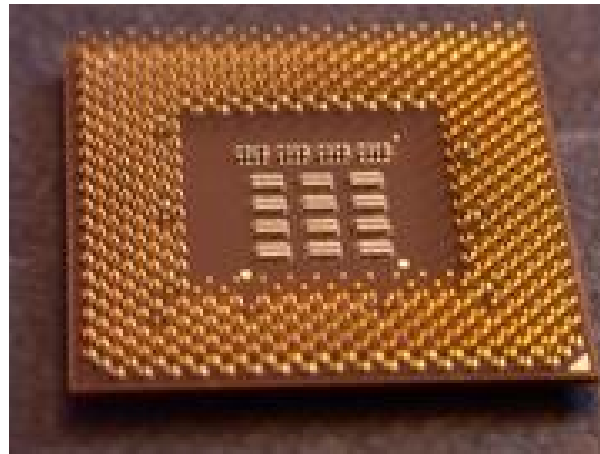


# CPU

Funzioni della CPU:

- estrae una dopo l'altra le istruzioni dal programma da eseguire,
- interpreta ed esegue istruzioni: operazioni di calcolo, operazioni logiche o di altri tipi dopo che prima ha acquisito i dati necessari dalla memoria,
- comanda gli altri componenti del computer, controlla il modo in cui il computer realizza l'elaborazione.

Architettura CPU = una famiglia di CPU.



# Memoria

Posto nel computer dove si possono accogliere i dati e il programma.

Due tipi di memoria:

Memoria primaria = chip di *memoria RAM (random access memory)* volatile e di *memoria ROM (read only memory)* persistente.

Memoria secondaria = dispositivi interni o esterni: unità di dischi (hard, floppy, CD), flash memory, unità di nastro etc.

La *memoria primaria* è più rapida che la memoria secondaria nelle operazioni di *lettura e scrittura* dei dati.

*Capacità di memorizzazione* = numero complessivo di celle nella memoria misurato in kilo (1024 celle), mega (1024 kilo celle) o giga (1024 mega).

Nella memoria secondaria i dati sono organizzati in *file system*.

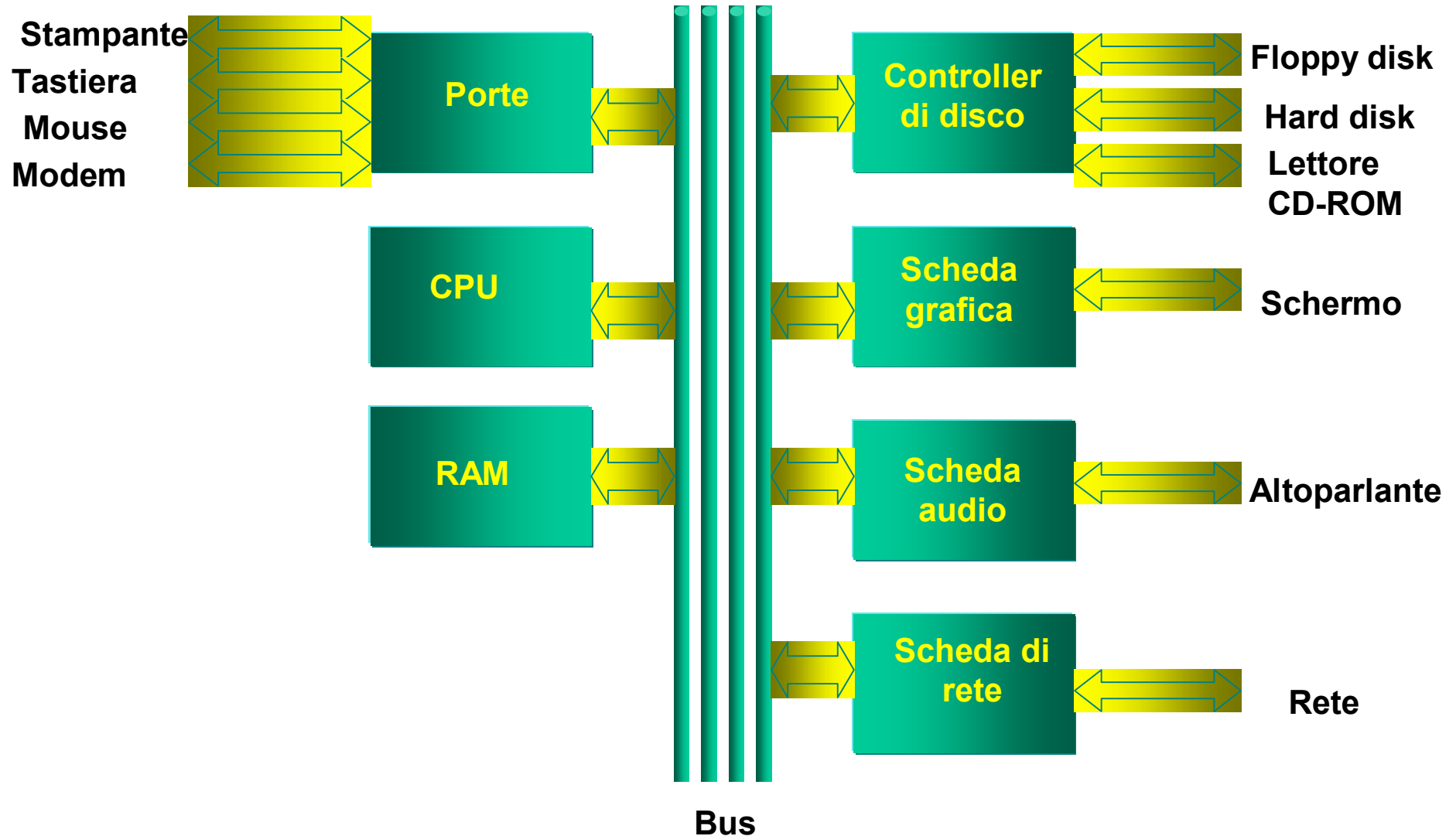
*File system* = struttura ad albero di contenitori chiamati *cartelle* (directory) in cui si memorizzano i file contenenti dati. Salvo i file una tale cartella può contenere altre cartelle.

*File* = sequenza di celle simile con le celle della memoria primaria.

L'indirizzo di una cella relativo all'inizio del file permette un accesso casuale, ma di solito accesso ai file è sequenziale.

Un file possono essere accedute in lettura (read-only), in scrittura (write-only) o in lettura/scrittura (read/write).

# Componenti hardware



# Algoritmo

*Procedimento* che consente di *ottenere* un dato *risultato* eseguendo, in un determinato ordine, un insieme di *passi semplici* corrispondenti ad azioni scelte solitamente da un insieme finito di azioni.

L'algoritmo necessita la descrizione di una serie di passaggi (passi) che non implichi congetture (cioè presupposti o ragionamenti basati su probabilità).

Un algoritmo:

- **non deve essere ambiguo:** i passi debbono essere evidenti all'*esecutore* dell'algoritmo. Nell'algoritmo si deve indicare che si deve fare per ciascun passaggio e quando è finito un passaggio dove si deve andare ;
- **deve essere eseguibile** cioè ciascun passaggio si deve poter eseguire concretamente;
- **deve arrivare a una conclusione.** Non è un algoritmo la descrizione di passaggi che si seguono all'infinito.

# Algoritmo

Depositare 10000 euro in un conto bancario che produce il 2% di interessi all'anno. Gli interessi vengono capitalizzati annualmente. Quanti anni occorrono affinché il saldo del conto raddoppi la cifra iniziale?

| Anno  | Saldo     |
|-------|-----------|
| 0     | 10.000,00 |
| 1     | 10.200,00 |
| 2     | 10.404,00 |
| 3     | 10.612,08 |
| 4     | 10.824,32 |
| ..... |           |
| 35    | 19.998,90 |
| 36    | 20.398,87 |

# Descrizione costruzione tabella

## **Passo 1:** Iniziare la tabella

| Anno | Saldo    |
|------|----------|
| 0    | 10000,00 |

## **Passo 2 :** Ripetere i passi da 2.1 a 2.3 fin tanto che il saldo è inferiore a 20000.

2.1. Aggiungere una riga nuova alla tabella

2.2. Nella colonna 2 della nuova riga, inserire il valore dell'anno precedente incrementato di 1

2.3. Nella colonna 2 della nuova riga inserire il valore del soldo precedente moltiplicato per 1,02

## **Passo 3 :** Leggere l'ultima cifra nella colonna dell'anno e riportare il numero di anni necessari per raddoppiare l'investimento.

**Risposta :** 35 anni

# Algoritmo in pseudo-codice

**start**

**assign** anno = 0

**assign** saldo = 10000

**while** saldo < 20000 **do** {

**assign** anno = anno + 1

**assign** saldo = saldo \* 1,02

}

**print** anno

**stop**

# Descrizione in Java

```
public class Raddoppio {  
    public static void main(String [] args) {  
        int anno = 0;  
        double saldo = 10000;  
        while (saldo < 20000) {  
            anno = anno + 1;  
            saldo = saldo * 1.02;  
        }  
        System.out.println("Anno del raddoppio: "+ anno);  
    }  
}
```

# Installazione JDK

1. Scaricare Java SE Development Kit dal indirizzo:

`http://java.sun.com/javase/downloads/index.jsp`

Il file scaricato si chiama file `jdk-6-windows-i586.exe` e rappresenta il programma d'installazione del Java SE Development Kit. Il file verrà memorizzato in una directory a propria scelta sul disco rigido.

2. Installare JSDK. La procedura d'installazione per il sistema Windows si trova all'indirizzo:

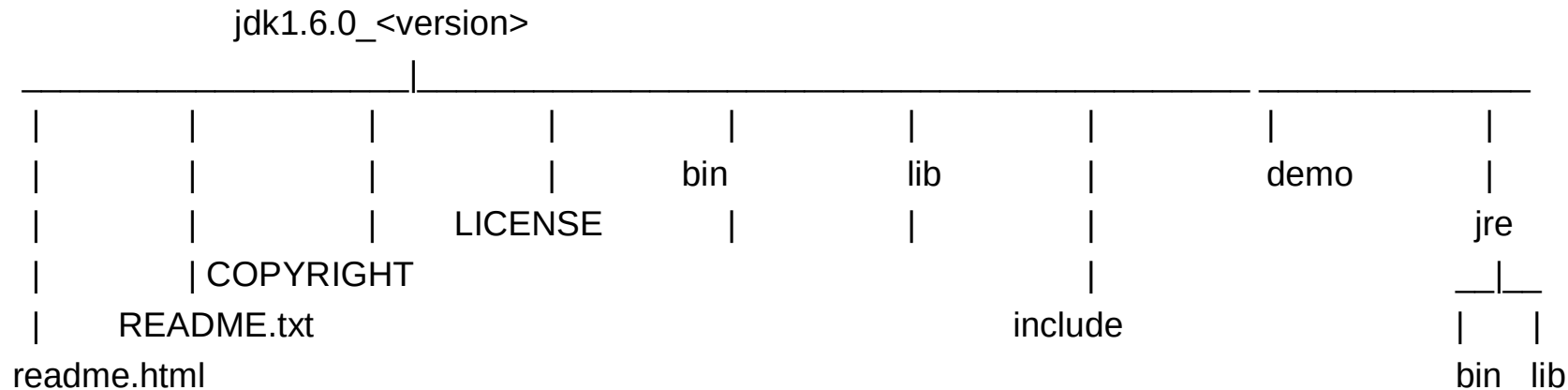
<http://java.sun.com/javase/6/webnotes/install/jdk/install-windows.html>

2.1 Eseguire il programma d'installazione clickando due volte sul file che installa JDK memorizzato sul disco:

`jdk-1_6_0_2_<version>-windows-i586-i.exe`

Durante l'esecuzione seguire le indicazioni visualizzate dal programma.

# Struttura di file sul disco



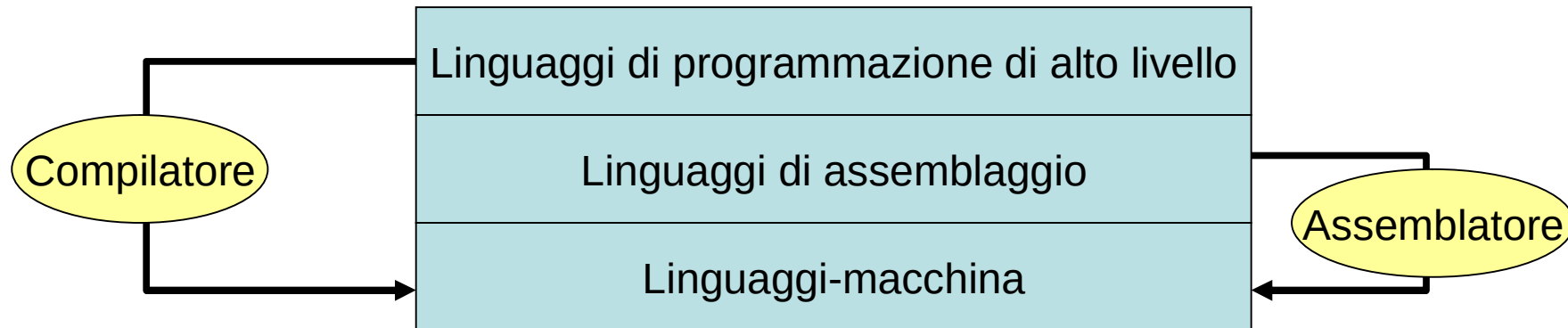
Avviare un programma (javac) sul disco con:

**C:>.\jdk1.6.0\_<version>\bin\javac MyClass.java**

## 2.2 Modificare la variabile PATH

- Scegliete Avvio, Pannello di controllo e cliccate due volte Sistema.
- Selezionate il tab Avanzate e poi Variabili d'ambiente.
- Ricercate la variabile "Path" tra Variabili di sistema.
- Selezionate la variabile "Path", cliccare su Modifica e aggiungete alla fine la stringa `C:\Program Files\Java\jdk1.6.0_<version>\bin`
- Verificate che la stringa sia preceduta da ";".
- Cliccate "OK" e di nuovo "OK" o "Imposta".

# Linguaggi di programmazione



**Linguaggio di programmazione di alto livello** = linguaggio dotato di una sintassi definita in modo matematico e utilizzato per scrivere programmi che “implementano” algoritmi.

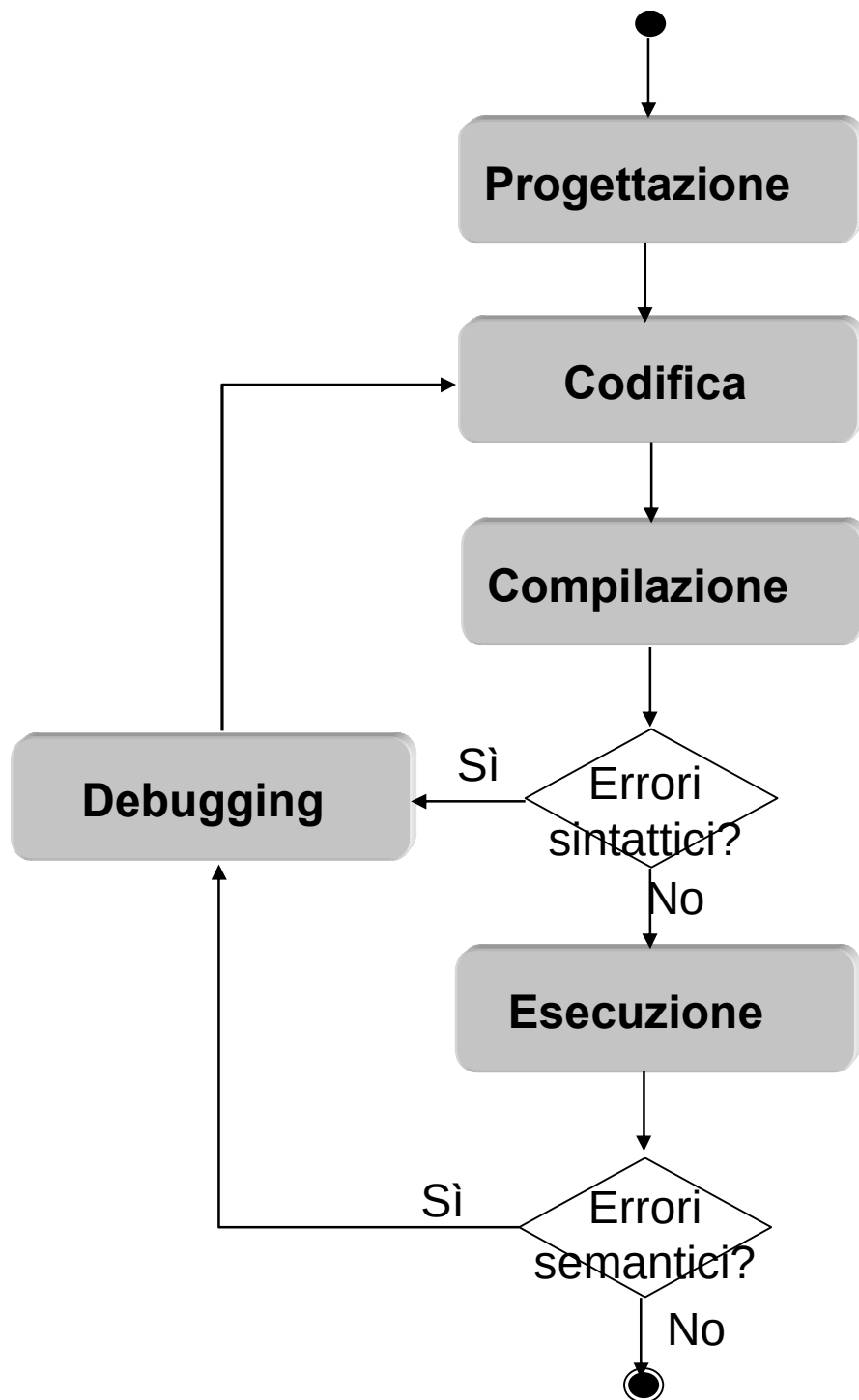
Esempi: Pascal, Java, C, C++.

**Programma** = descrizione di un algoritmo in un linguaggio di programmazione.

**Compilazione** = traduzione del programma in un linguaggio di alto livello in un linguaggio-macchina.

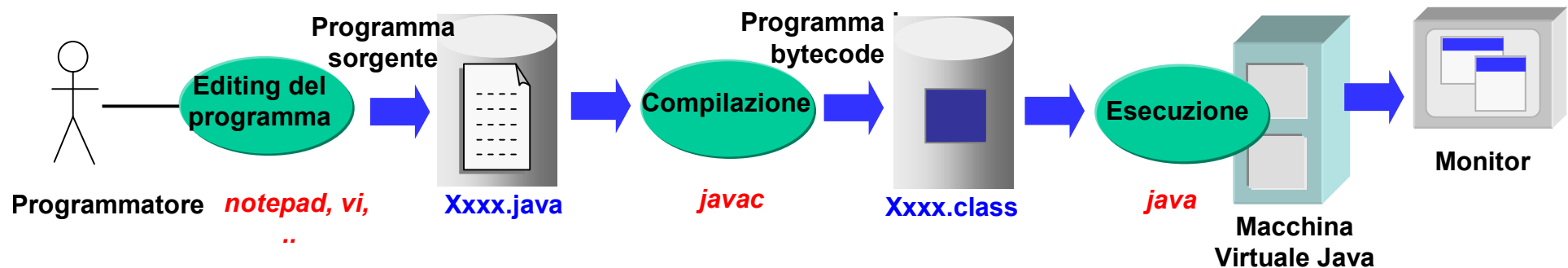
**Caso Java: Macchina Virtuale Java**

# Sviluppo di un programma in un linguaggio di livello alto

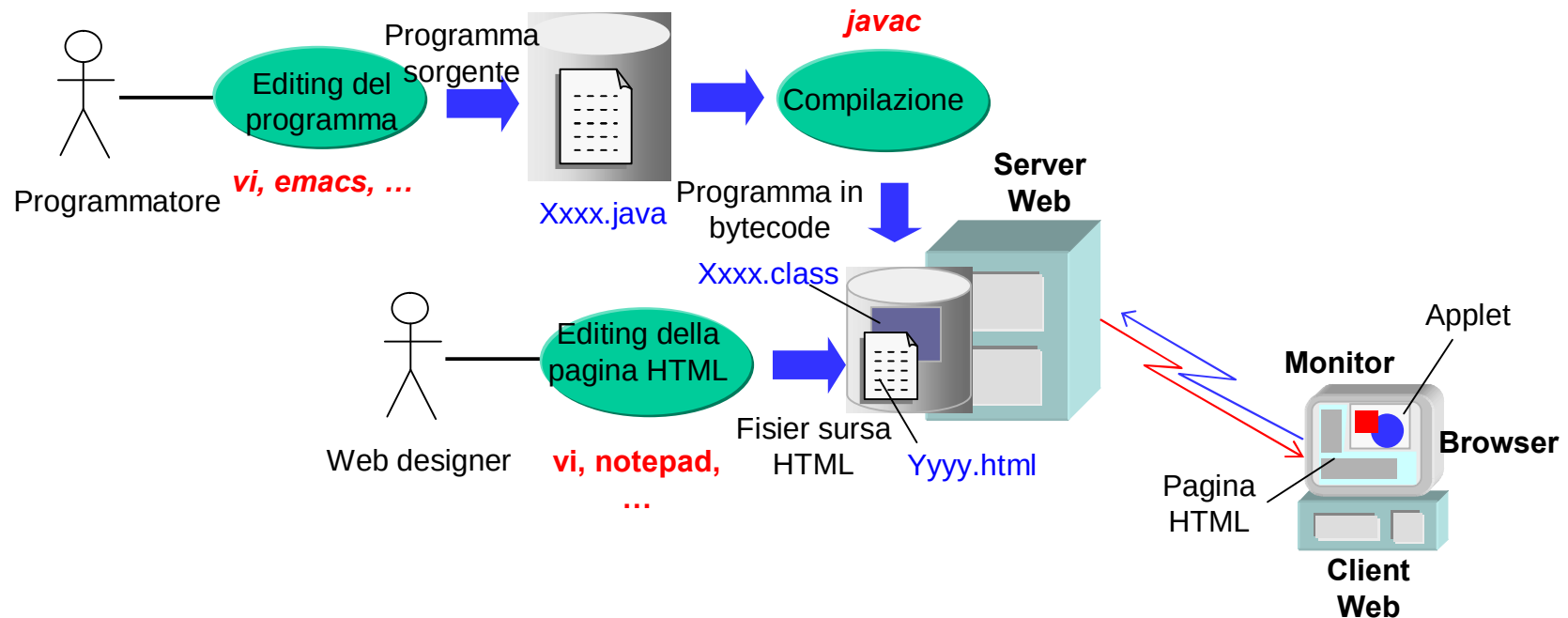


# Sviluppare programmi Java

## Sviluppare applicazioni Java

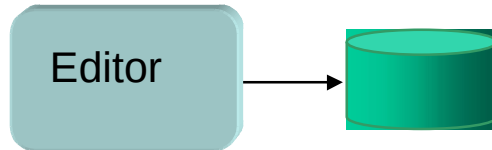


## Sviluppare applet Java



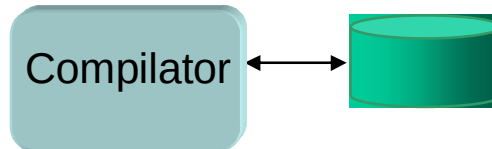
# Sviluppo ed esecuzione di un programma Java

Faza 1



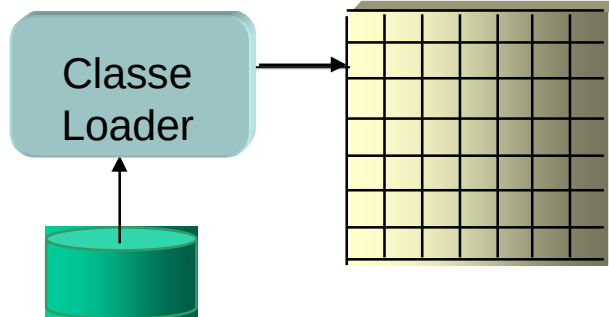
Il programma è creato col editor e memorizzato sul disco

Faza 2



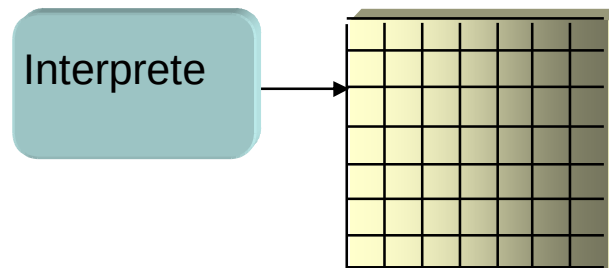
Il compilatore crea il bytecode e lo memorizza sul disco

Faza 3



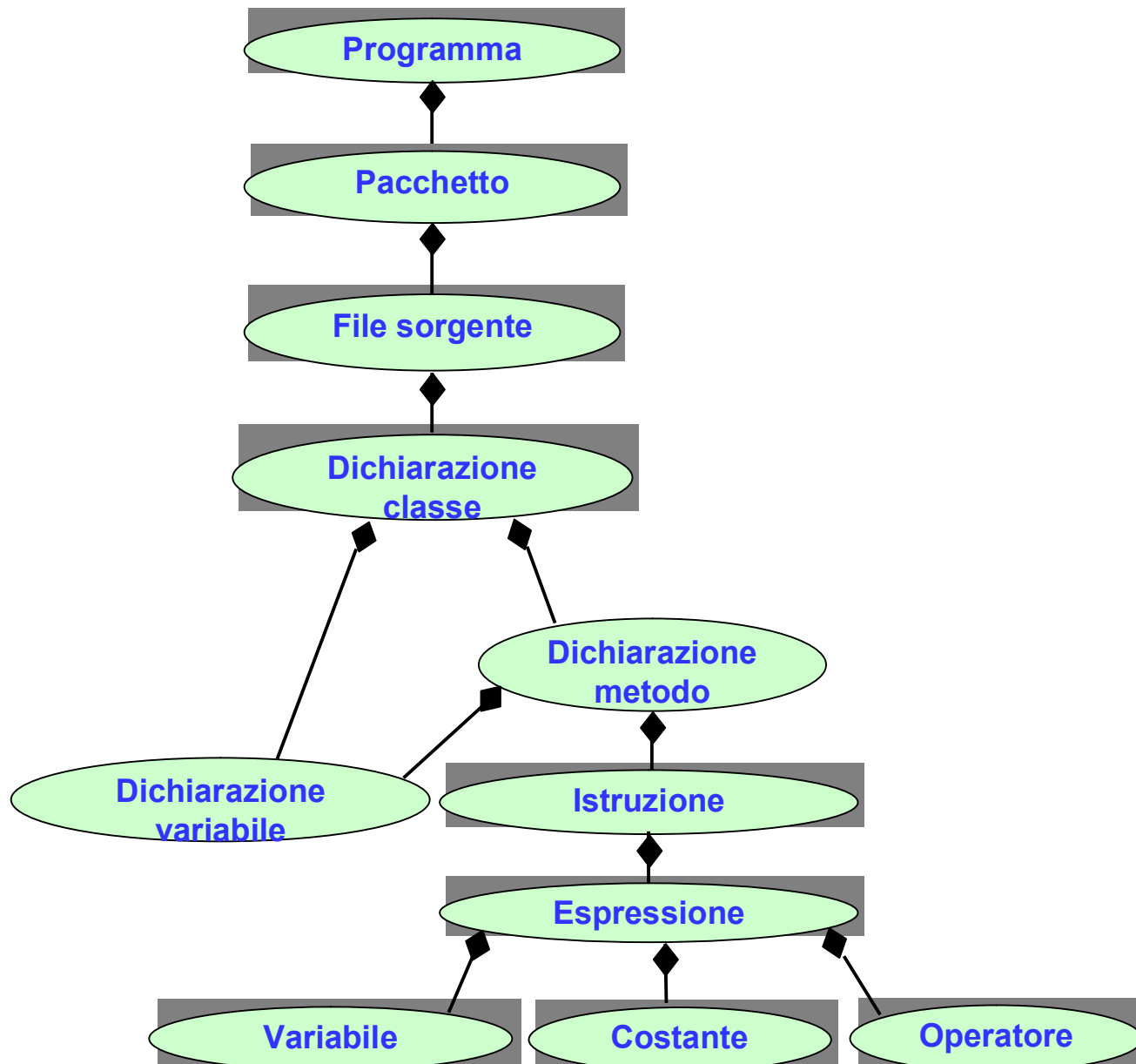
La classe Loader carica il bytecode in memoria

Faza 4



L'interprete legge il bytecode, lo traduce nel linguaggio-macchina ed esegue il risultato della traduzione.

# Struttura gerarchica di un linguaggio



# Variabili

*Variabile* = identifica una locazione di memoria destinata a contenere dei dati *che possono essere modificati* nel corso dell'esecuzione del programma.

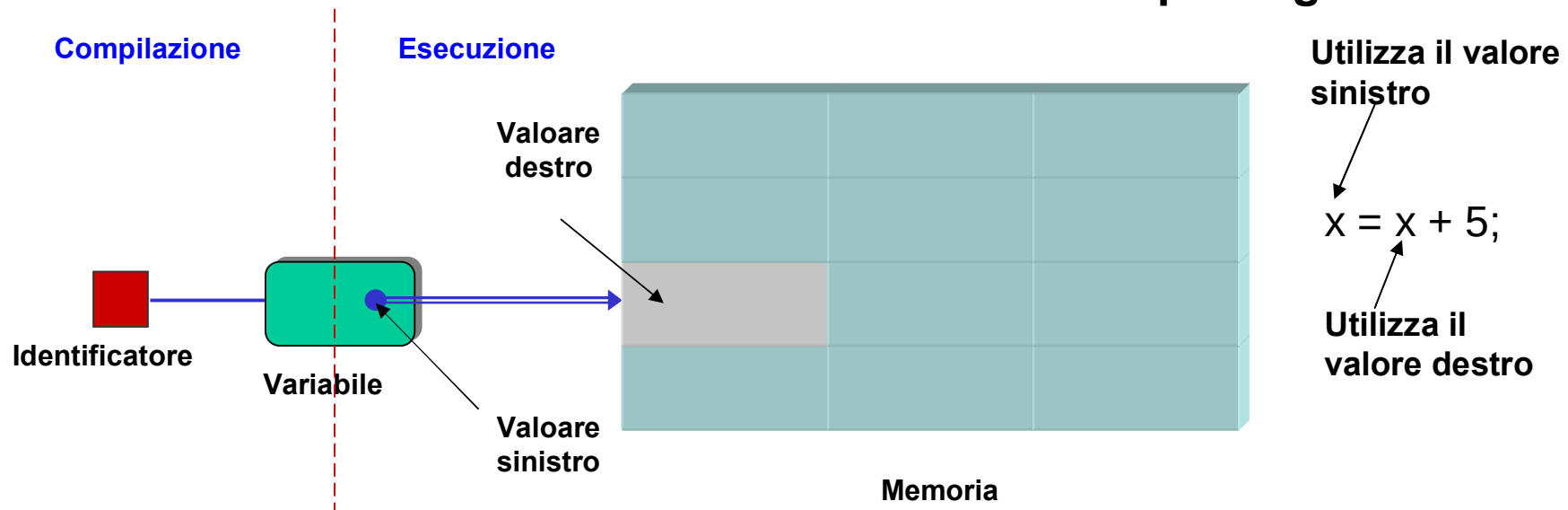
Una variabile è spesso rappresentata nel programma come un *identificatore*.

*Identificatore* = nome nel programma che identifica un'entità del programma: variabile, metodo, classe, pacchetto etc. E' formato da una o più lettere e cifre, con il primo carattere lettera.

Esempio: **x saldo anno2 x123 alfa3bis**

Non sono identificatori: **Mr.John C++ clasa-1 3abc**

Identificatori riservati: **boolean class do interface package**



# Lavorare con variabili

1. Prima di essere utilizzate le variabili devono essere **dichiarate**.
2. La dichiarazione di una variabile specifica il suo tipo di dato e riserva una definita estensione della memoria per la variabile.
3. Le operazioni basilari con una variabile:
  - **lettura** del valore in essa contenuto. Il valore non si modifica
  - **scrittura** di un nuovo valore(**assegnamento**). Il valore si modifica.

## Esempi:

```
saldo = 25834.68;  
somma = 84;  
numero1 = 14;  
numero2 = -85;  
somma = numero1 + numero2;
```

# Programma Variabile1

```
public class Variabile1 {  
    public static void main (String[] args){
```

```
        int x;    //dichiarazione di variabile x
```

x 

```
        x=10;    //assegnamento di una valore alla variabile
```

x 

```
        System.out.println(x); //visualizza il valore di x
```

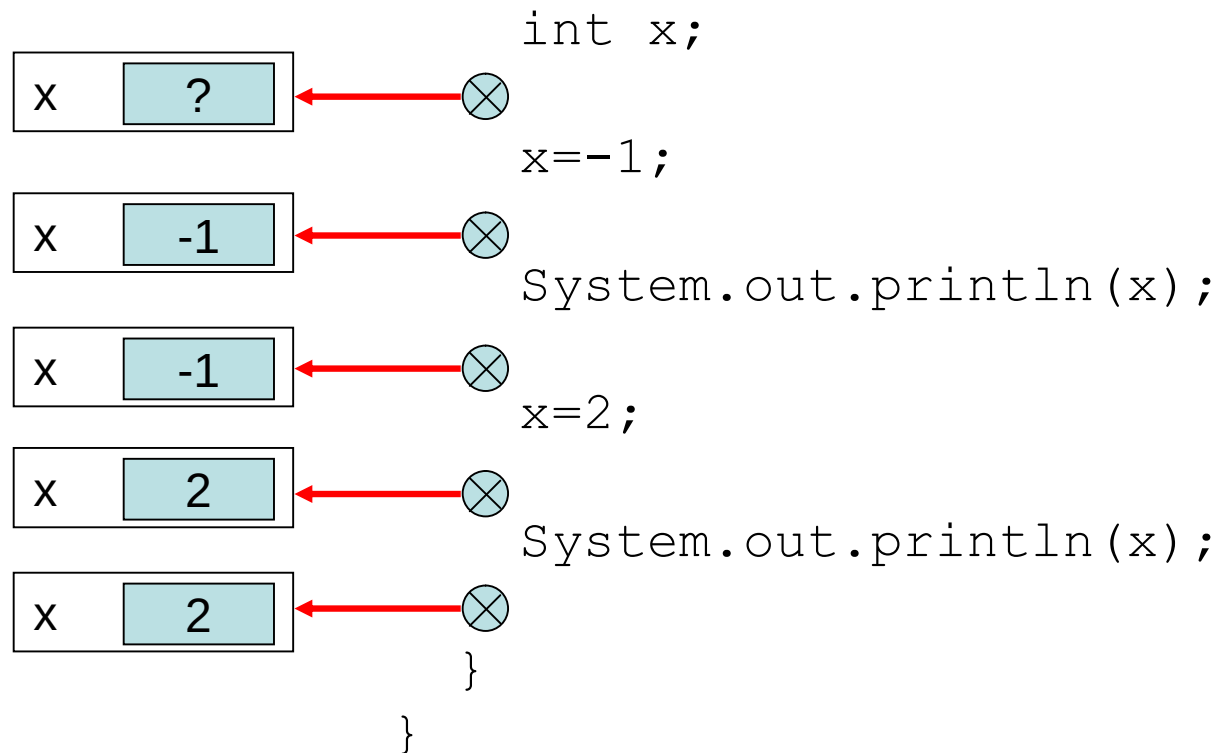
x 

```
    }
```

```
}
```

# Programma Variabile2

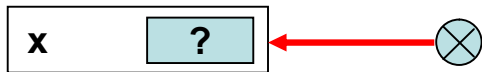
```
public class Variabile2 {  
    public static void main (String[] args) {
```



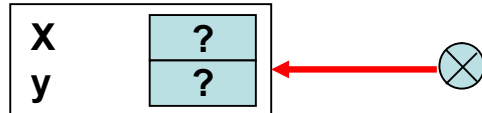
# Copiare dati tra variabili

```
public class DueVariabili1 {  
    public static void main(String[] args){
```

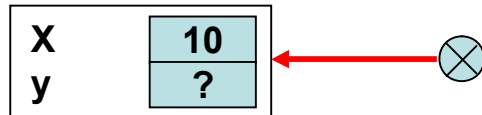
```
        int x;
```



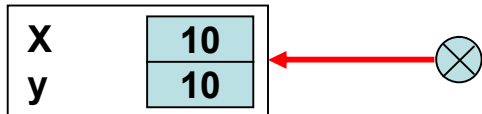
```
        int y;
```



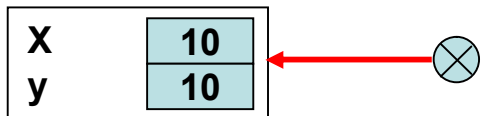
```
        x=10;
```



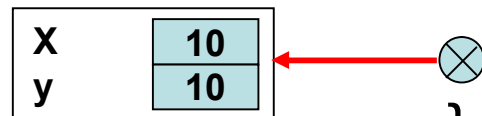
```
        y=x;
```



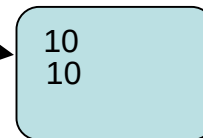
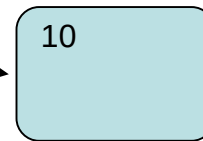
```
        System.out.println(x);
```



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

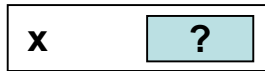


```
    }  
}
```

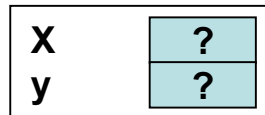


# Copiare dati tra variabili

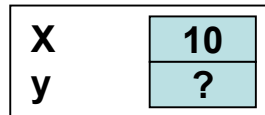
```
public class DueVariabili2 {  
    public static void main(String[] args){  
        int x;
```



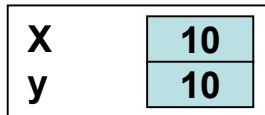
```
        int y;
```



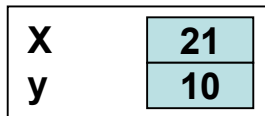
```
        x=10;
```



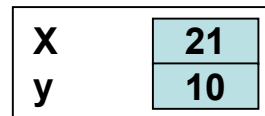
```
        y=x;
```



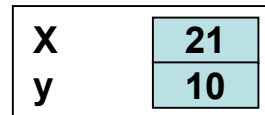
```
        x=21;
```



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

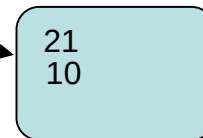
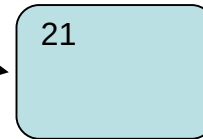


```
        System.out.println(y);
```



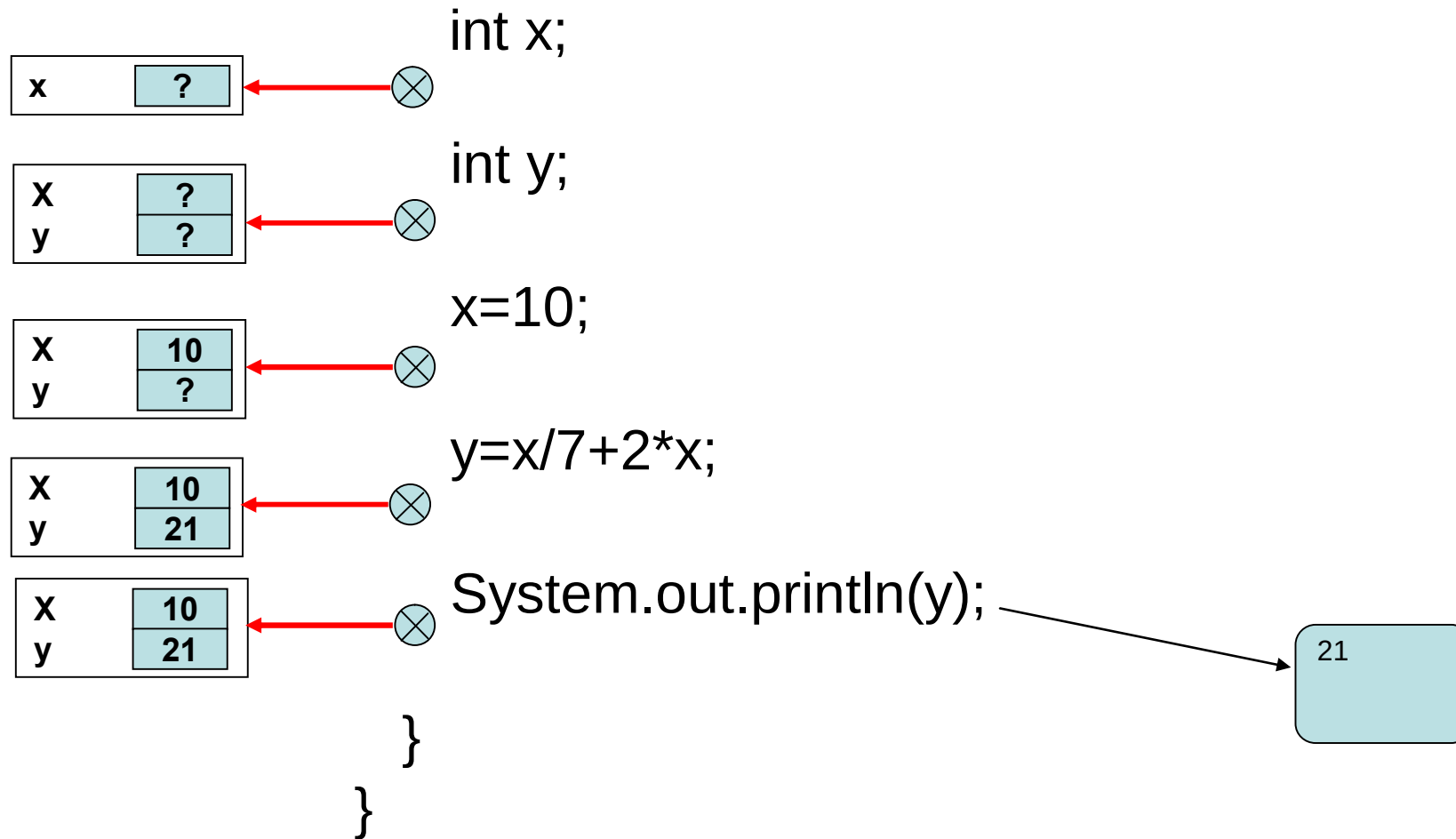
```
    }
```

```
}
```



# Copiare dati tra variabili

```
public class DueVariabili3 {  
    public static void main(String[] args){
```



# Lavorare con variabili reali

1.

```
public class VariabileReale{  
    public static void main(String[] args){  
        double x;    // dichiarazione di variabile reale  
        x=10.5/4.2; //divisione e assegnazioni di numeri reali  
        System.out.println(x); // viene visualizzato 2.5  
    }  
}
```

2.

```
public class VariabileRealeValoreIntero{  
    public static void main(String[] args){  
        double x;  
        x=5/2; // risultato intero 2 viene convertito a 2.0  
        System.out.println(x); // visualizza 2.0  
    }  
}
```

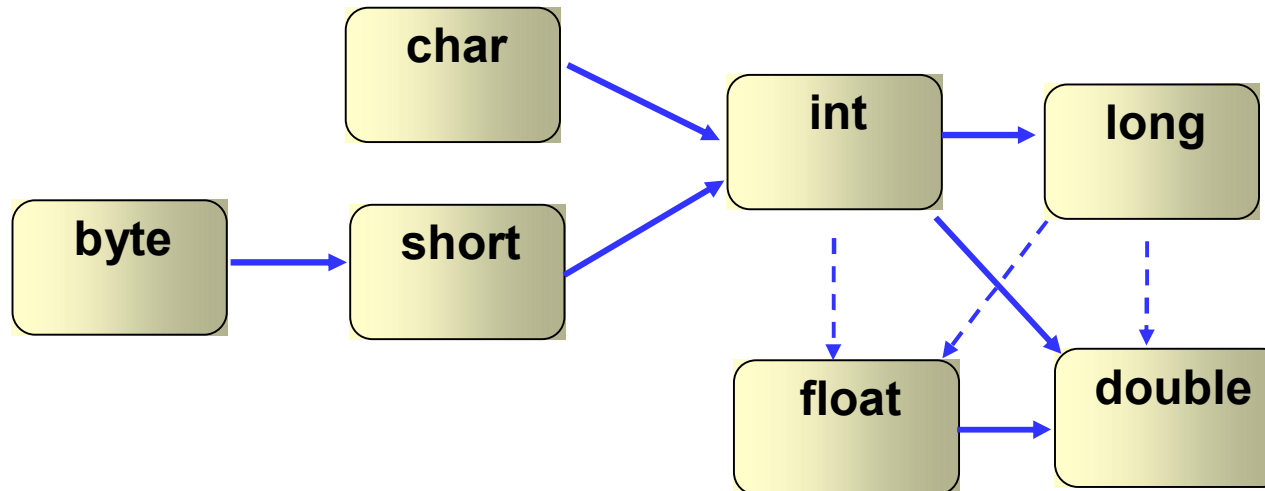
# Media Aritmetica

```
public class MediaAritmetica{  
    public static void main(String[] args){  
        int x1=5;  
        int x2=6;  
        int x3=3;  
        double media=(x1+x2+x3)/3; //Risultato : 4.0  
        System.out.print("Valore medio e' ");  
        System.out.println(media);  
    }  
}
```

# Tipi primitivi di dato in Java

| Tipo    | Rappresentazione in memoria | Dominio                                                                   |
|---------|-----------------------------|---------------------------------------------------------------------------|
| byte    | 1 byte = 8 bits             | da -128 a 127                                                             |
| short   | 2 bytes = 16 bits           | da -32.768 a 32.767                                                       |
| int     | 4 bytes = 32 bits           | da -2.147.483.648 a 2.147.483. 647                                        |
| long    | 8 bytes = 64 bits           | da -9.223.372.036.854.775.808L a 9.223.372.036.854.775.807L               |
| float   | 4 bytes = 32 bits           | appross. $\pm 3.40282347E+38F$ (6–7 cifre decimali significative)         |
| double  | 8 bytes = 64 bits           | appross. $\pm 1.79769313486231570E+308$ (15 cifre decimali significative) |
| char    | 2 bytes = 16 bits           | da 0 a 65.535                                                             |
| boolean | 1 bit                       | due valori                                                                |

# Conversioni di tipi primitivi



# Tipi di dato costruiti in Java

- La dichiarazione di classe introduce un tipo costruito da vari altri dati avente il nome della classe:

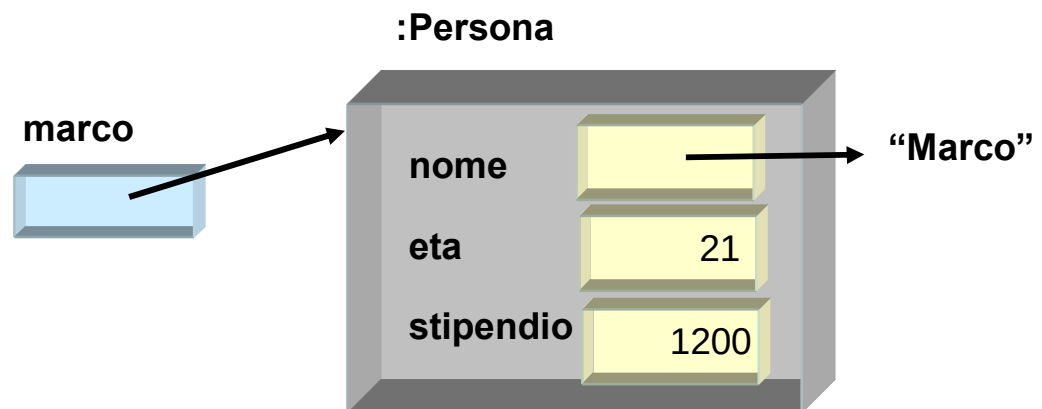
```
class Persona {  
    String nome;  
    int eta;  
    double stipendio;  
}
```

Con un tipo costruito possiamo creare quanti valori (istanze) ci occorrono nel nostro programma. Questi valori si devono memorizzare in variabili dichiarate di tipo della classe.

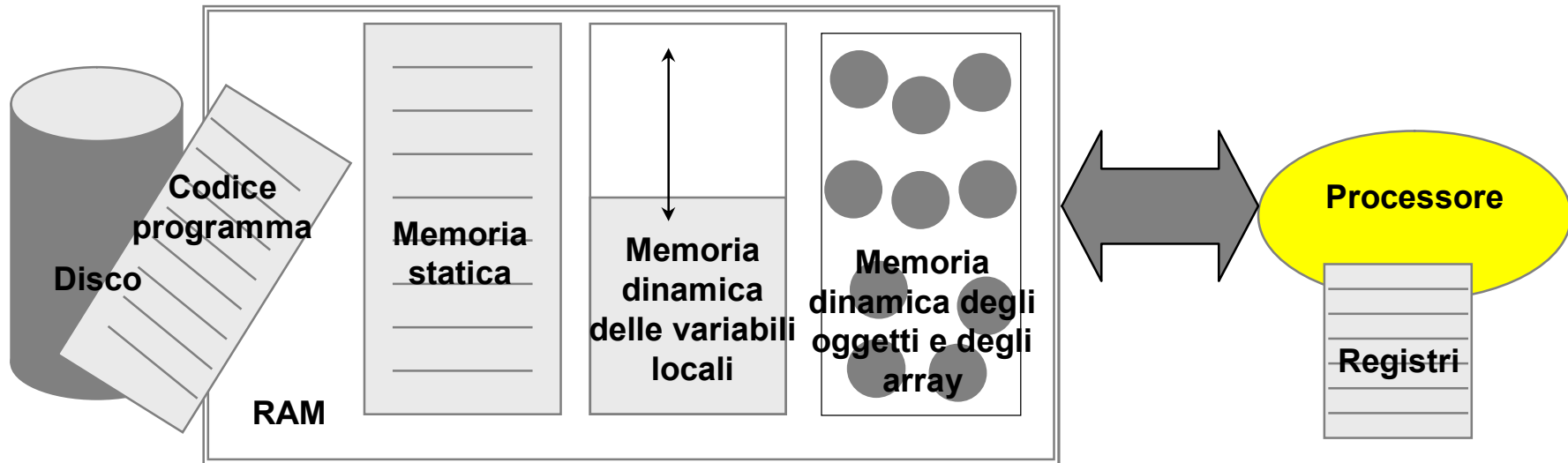
```
Persona marco = new Persona();
```

Le variabili sono utilizzate per accedere ai componenti del tipo di dato.

```
marco.nome = "Marco";  
marco.eta = 21;  
marco.stipendio = 1200;
```



# Struttura della memoria della JVM



# Operatori in Java

| Operatori aritmetici |                      |
|----------------------|----------------------|
| *                    | moltiplicazione      |
| /                    | divisione            |
| %                    | modulo               |
| +                    | somma/concatenazione |
| -                    | sottrazione          |

| Operatori di confronto |                   |
|------------------------|-------------------|
| <                      | minore            |
| <=                     | minore o uguale   |
| >                      | maggiore          |
| >=                     | maggiore o uguale |
| ==                     | uguale            |
| !=                     | diverso           |

| Operatori logici |     |
|------------------|-----|
| &&               | e   |
|                  | o   |
| !                | non |

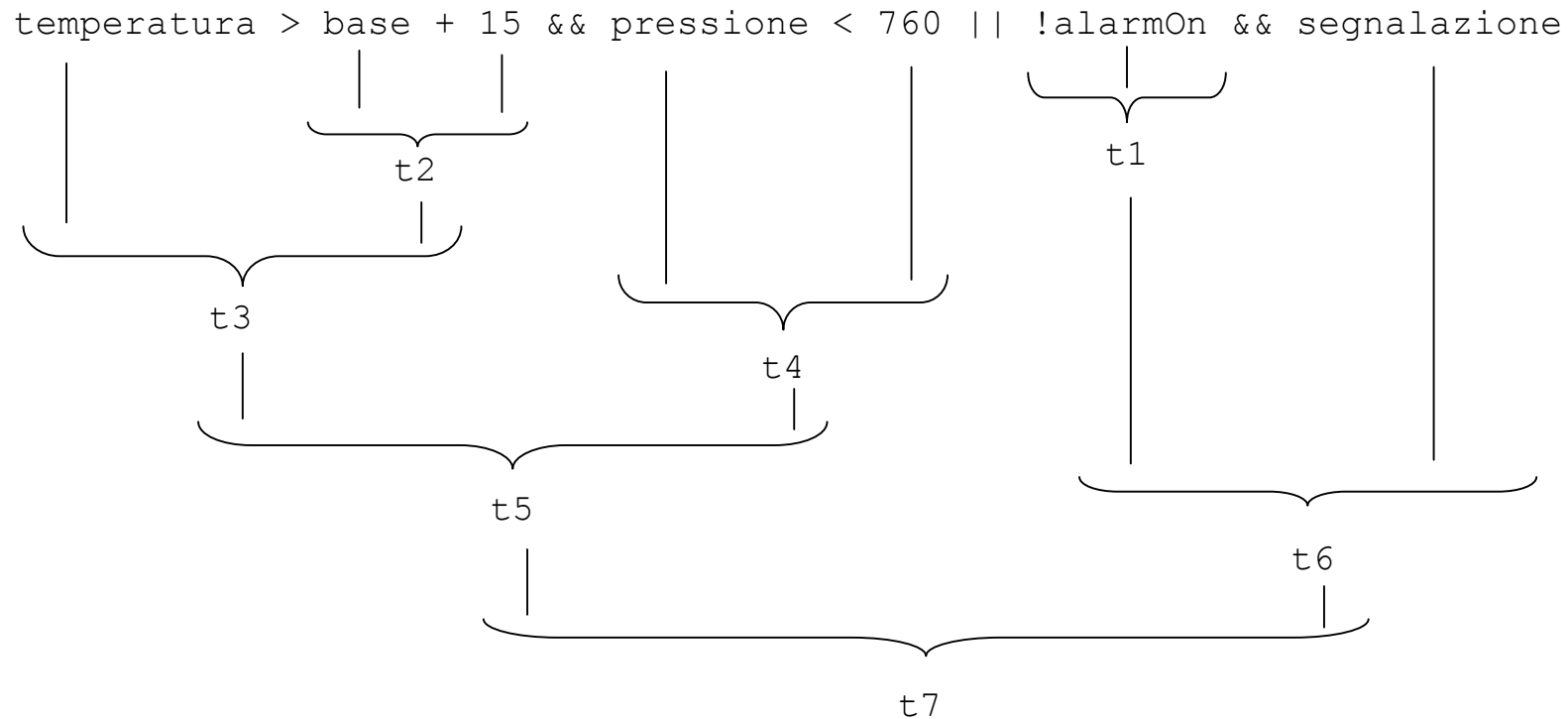
| Operatori multifunzionali |                              |
|---------------------------|------------------------------|
| ++                        | incremento                   |
| --                        | decremento                   |
| =                         | assegnamento                 |
| +=                        | assegnamento con somma       |
| -=                        | assegnamento con sottrazione |
| *=                        | diverso                      |

| Altri operatori   |                 |
|-------------------|-----------------|
| <b>instanceof</b> | verifica tipo   |
| <b>new</b>        | crea un oggetto |
| ?:                | condizionale    |
| (tipo)            | cast            |

# Precedenza degli operatori

| Tipo operatore        | Operatori                              | Associatività |
|-----------------------|----------------------------------------|---------------|
| operatori postfissati | [] . (parametri) expr++ expr--         | destra        |
| operatori unari       | ++expr --expr +expr -expr ~ !          | destra        |
| creazione e cast      | new (tip)expr                          | destra        |
| molteplicativi        | * / %                                  | sinistra      |
| addittivi             | + -                                    | sinistra      |
| spostamento           | << >> >>>                              | sinistra      |
| relazionali           | < > <= >= instanceof                   | sinistra      |
| uguaglianza           | == !=                                  | sinistra      |
| AND per bits          | &                                      | sinistra      |
| XOR per bits          | ^                                      | sinistra      |
| OR per bits           |                                        | sinistra      |
| AND logico            | &&                                     | sinistra      |
| OR logico             |                                        | sinistra      |
| condizionale          | ?:                                     | sinistra      |
| assegnazioni          | = += -= *= /= %= &= ^=  = <<= >>= >>>= | destra        |
| sequenza              | ,                                      | sinistra      |

# Ordine di valutazione delle espressioni



| Espressione                                                                            | Significato                                                                                       |
|----------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| <i>Espressioni numeriche</i>                                                           |                                                                                                   |
| $(x+7) * y / 2$                                                                        | $\frac{(x + 7) \times y}{2}$                                                                      |
| $2 * a * 3.14$                                                                         | $2 \times a \times 3,14$                                                                          |
| $(x - 'a') + 'A'$                                                                      | una minuscola trasformata in una maiuscola                                                        |
| <i>Espressioni boolean</i>                                                             |                                                                                                   |
| $x > 3$                                                                                | <b>x</b> maggiore di 3                                                                            |
| $x == z$                                                                               | <b>x</b> uguale a <b>z</b>                                                                        |
| $x != y$                                                                               | <b>x</b> diverso da <b>y</b>                                                                      |
| $b$                                                                                    | <b>b</b> è true                                                                                   |
| <i>Espressioni logiche</i>                                                             |                                                                                                   |
| $x > 3 \ \&\& \ x == z$                                                                | <b>x&gt;3</b> e <b>x=z</b>                                                                        |
| $a + d < c \    \ b$                                                                   | <b>a + d &lt; c</b> o <b>b</b>                                                                    |
| <i>Chiamate di metodi</i>                                                              |                                                                                                   |
| <code>getValue()*77.4</code>                                                           | Il valore ottenuto dall'esecuzione del metodo <code>getValue()</code> è moltiplicato per 77.4     |
| <code>mioAmico.getNome()</code>                                                        | apelează metoda instanță <code>getPrenume()</code> a variabilei instanță <code>amiculMeu</code>   |
| <i>Creazione di oggetti con la memorizzazione delle istanze in variabili reference</i> |                                                                                                   |
| <code>Mamifero m=new Mamifero()</code>                                                 | creazione di un'istanza della classe Mamifero e la sua memorizzazione in una variabile            |
| <i>Espressioni con operatori aritmetici e assegnazioni</i>                             |                                                                                                   |
| <code>x = y = z = 82;</code>                                                           | <code>z=82; y=z; x=y</code>                                                                       |
| <code>s = "Ciao";</code>                                                               | creazione di un oggetto di tipo String con il valore "Ciao" e memorizzazione in una variabile s   |
| <code>x = 123.5 / 8.75f * y;</code>                                                    | nuova valore della variabile x è il valore dell'espressione: $123,5 / 8,75 \times$ il valore di y |
| <i>Espressioni con l'operatore di concatenazione delle stringhe</i>                    |                                                                                                   |
| <code>s=s+" a tutti!"</code>                                                           | se il valore di s è un reference a "Ciao", il nuovo valore di s è un reference a "Ciao a tutti!"  |

# Istruzioni

## 1. Istruzioni-espressione

- Espressioni di assegnamento
- Utilizzo di ++ o --
- Chiamate di metodi
- Espressioni di creazione oggetti

Esempi:

```
aValue = 8933.234;           // istruzione-espressione
aValue++;                    // istruzione-espressione: incremento
System.out.println(aValue); // istruzione-espressione: chiamata
Integer integerObject = new Integer(4); // creazione oggetto
```

## 2. Istruzione dichiarazione.

```
double aValue = 8933.234;           // Istruzione dichiarazione
```

## 3. Istruzioni di controllo del flusso dell'esecuzione.

- Istruzione condizionale o di decisione (if)
- Istruzione di selezione (switch)
- Istruzione iterativa, ciclo (while, do-until, for)

# Istruzioni condizionali

## Istruzione

## Sintassi

1. Istruzione **if-else** con una sola alternativa

```
if (Condizione)  
    Istruzione
```

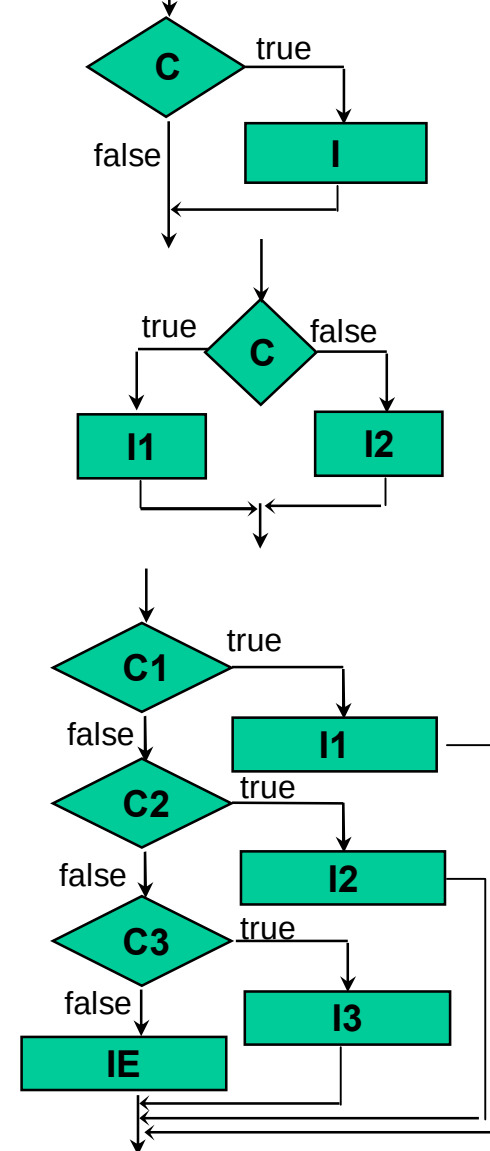
2. Istruzione **if-else** con due alternative

```
if (Condizione)  
    Istruzione1  
else Istruzione2
```

3. Istruzione **if-else** multipla

```
if (Condizione1)  
    Istruzione1  
else if (Condizione2)  
    Istruzione2  
else if (Condizione3)  
    Istruzione3  
else IstruzioneElse
```

## Semantica



# Utilizzare istruzioni if

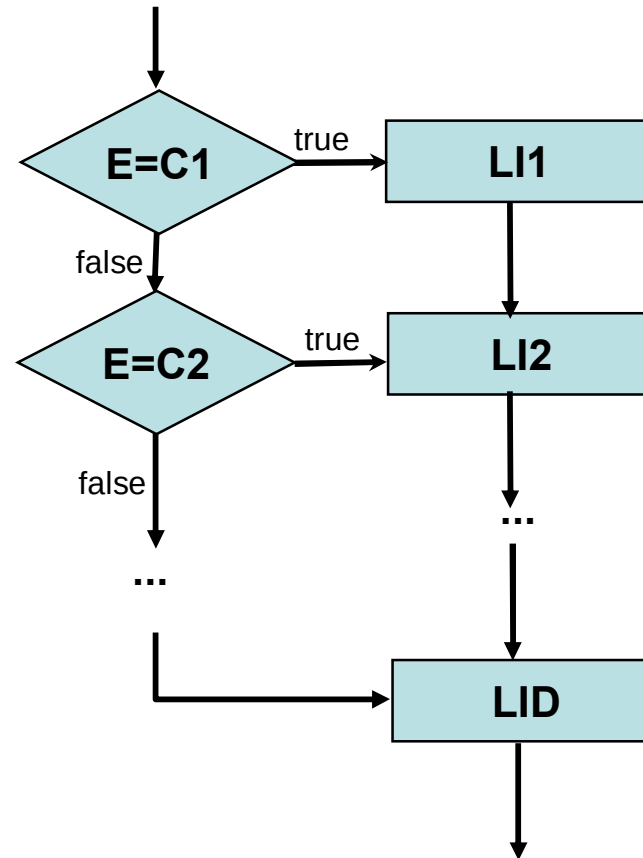
1.

```
public class Equazione {  
    public static void main (String args[]) {  
        double a=10.2, b=-5.7, c=2.1;  
        if ((b*b - 4*a*c)<0)  
            System.out.println("Soluzioni complesse");  
        else if (b*b == 4*a*c)  
            System.out.println("Soluzioni reali, uguali");  
        else  
            System.out.println("Soluzioni reali");  
    }  
}
```

2.

# Istruzione di selezione

```
switch (E) {  
  case C1:  
    ListaIstruzioni1  
  case C2:  
    ListaIstruzioni2  
  ....  
  default:  
    ListaIstruzioniDef  
}
```



# Utilizzare l'istruzione switch

```
public class SwitchDemo {  
    public static void main(String[] args) {  
        int mese = 8;  
        switch (mese) {  
            case 1: System.out.println("January"); break;  
            case 2: System.out.println("February"); break;  
            case 3: System.out.println("March"); break;  
            case 4: System.out.println("April"); break;  
            case 5: System.out.println("May"); break;  
            case 6: System.out.println("June"); break;  
            case 7: System.out.println("July"); break;  
            case 8: System.out.println("August"); break;  
            case 9: System.out.println("September"); break;  
            case 10: System.out.println("October"); break;  
            case 11: System.out.println("November"); break;  
            case 12: System.out.println("December"); break;  
            default: System.out.println("Invalid month.");break;  
        }  
    }  
}
```

# Istruzioni iterative

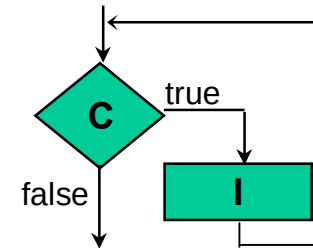
## Istruzione

Istruzione **while**

## Sintassi

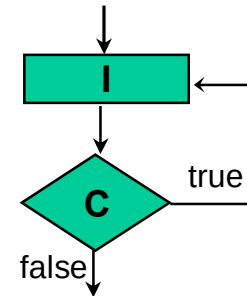
```
while (Condizione)  
    Istruzione
```

## Semantica



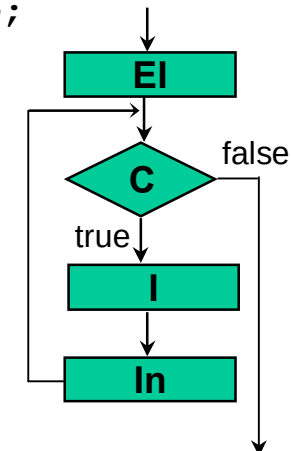
Istruzione **do-while**

```
do  
    Istruzione  
while (Condizione)
```



Istruzione **for**

```
for (EsprIniziale; Condizione;  
      EsprIncrement)  
    Istruzione
```



Interruzione iterazione corrente

```
continue;
```

Interruzione ciclo

```
break;
```

# Blocco di istruzioni

Un **blocco** di istruzioni = gruppo di zero o più istruzioni racchiuse tra {}.

```
public class Demo {  
    public static void main(String[] args)  
    {  
        int somma = 0;  
        for (int cursore = 1; cursore <= 10; cursore ++)  
        {  
            somma += cursore;  
        }  
        System.out.println("Somma = " + somma);  
    }  
}
```

Risultato: Somma = 55

# Metodi

Metodo = Algoritmo che viene eseguito nell'ambito di un oggetto o di una classe.

Il metodo deve essere dichiarato nel suo ambito.

## Ambito di un oggetto

```
public class Persona {  
    String nome;  
    int eta;  
    double stipendio;  
    public String getNome() {  
        return nome;  
    }  
    public void setNome(String n) {  
        nome = n;  
    }  
    public int getEta() {  
        return eta;  
    }  
    public int getStipendio() {  
        return stipendio;  
    }  
}
```

nome, eta, stipendio sono variabili-istanza dell'oggetto. Sono variabili globali per i metodi.

n è il parametro del metodo `setNome()`. È anche una variabile locale del metodo.

## Ambito di una classe

```
public class NumeriQuadrati {  
    public static void main(String  
                                args[]) {  
        for(int i=0;i<=100;i++)  
            System.out.println("quadrato(" +  
                                i + ")=" + quad(i));  
    }  
    private static int quad (int x) {  
        return x * x;  
    }  
}
```

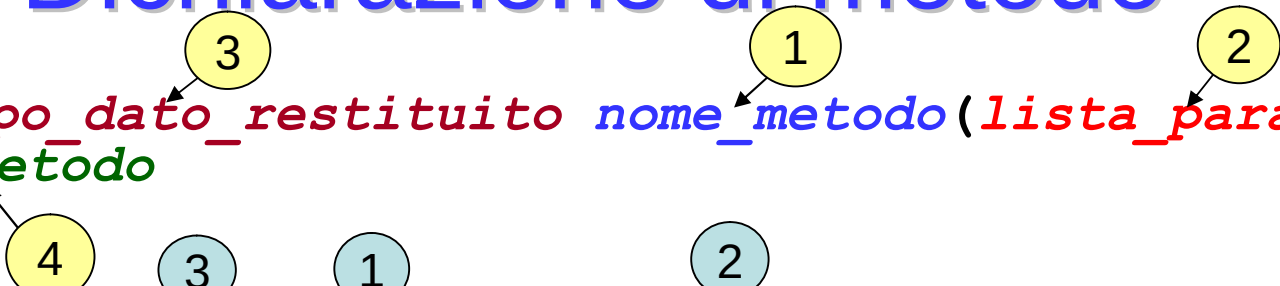
args è il parametro del metodo `main()`. E' anche una variabile locale del metodo.

x è il parametro del metodo `quad()`. E' anche una variabile locale del metodo

# Dichiarazione di metodo

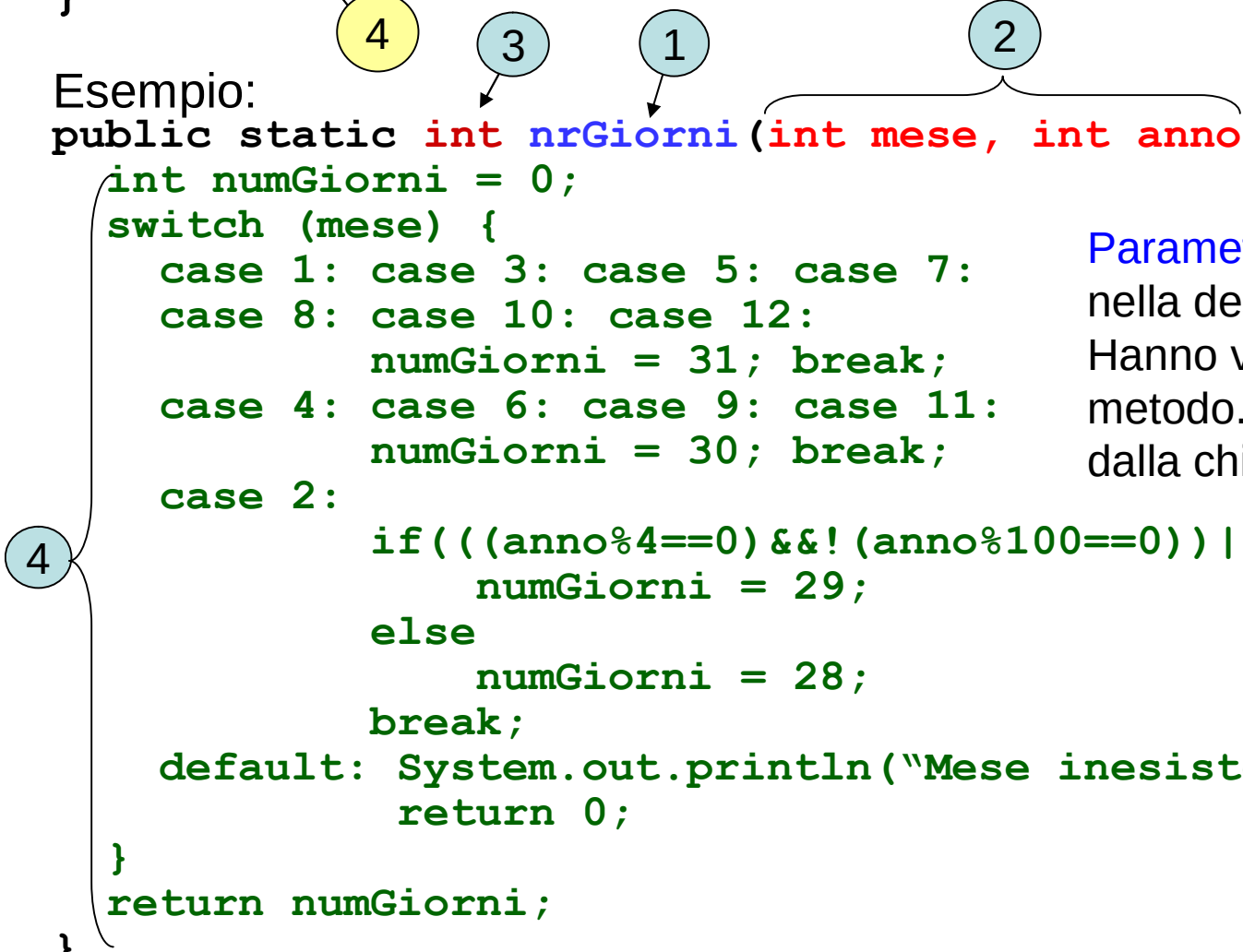
Sintassi:

```
public tipo_dato_restituito nome_metodo(lista_parametri) {  
    corpo_metodo  
}
```



Esempio:

```
public static int nrGiorni(int mese, int anno) {  
    int numGiorni = 0;  
    switch (mese) {  
        case 1: case 3: case 5: case 7:  
        case 8: case 10: case 12:  
            numGiorni = 31; break;  
        case 4: case 6: case 9: case 11:  
            numGiorni = 30; break;  
        case 2:  
            if(((anno%4==0)&&!(anno%100==0)) || (anno%400==0) )  
                numGiorni = 29;  
            else  
                numGiorni = 28;  
            break;  
        default: System.out.println("Mese inesistente");  
                return 0;  
    }  
    return numGiorni;  
}
```



**Parametri** = Variabili utilizzate nella descrizione dell'algoritmo. Hanno visibilità locale nel metodo. I loro valori sono definiti dalla chiamata del metodo.

# Chiamata di metodo

Un metodo viene chiamato secondo la sua dichiarazione:

```
nomeMetodo(lista_argomenti)
```

dove `lista_argomenti` è una lista di espressioni di cui valori vengono passati come parametri della chiamata.

Se il metodo restituisce un valore, questo valore deve essere salvato in una variabile o utilizzato in una espressione.

```
double totale = ammontare * Math.pow(1.0 + tasso, anno);
```

## Metodi-istanza e metodi della classe

- quando il metodo non è dichiarato con `static`, con una variabile reference che riferisce un oggetto della classe,. Esempio:

```
Persona persona = new Persona();
```

```
. . . .
```

```
int e = persona.getEta();    // variabile.nomeMetodo(...)
```

- quando il metodo è dichiarato `static` (metodo “statico”) con il nome della classe. Esempio:

```
Math.pow(x*z, y+2)
```

I metodi appartenenti ad uno stesso oggetto o ad una stessa classe si possono chiamare tra loro senza precisare l'oggetto o il nome della classe.

# Come si esegue una chiamata di metodo

Un metodo può essere “chiamato” da un altro metodo (*metodo chiamante*).

Esecuzione di una chiamata:

1. Inizia con la valutazione degli argomenti.
2. I valori degli argomenti inizializza i parametri formali del metodo chiamato.
3. Si avvia l'esecuzione del metodo chiamato con i nuovi valori dei parametri.
4. Il risultato (se c'è) sostituisce la chiamata nel corpo del metodo chiamante.

I parametri formali accolgono i valori passati come **argomenti** dal metodo chiamante al metodo chiamato. Questo meccanismo si chiama **passaggio per valore dei parametri** (parameter passing by value).

Con questo meccanismo i parametri sono simili a delle variabili locali che sono inizializzate con i valori degli argomenti prima di eseguire il metodo chiamato.

Esempio:

```
public double confrontoNumeri(double d1, double d2){  
    return d1-d2;  
}  
.  
.  
.  
if (confrontoNumeri(-7.5, 10)>0){  
    .  
    .  
    .
```

Chiamata con passaggio di parametri

```
confrontoNumeri(double d1, double d2){  
    d1 = -7.5;  
    d2 = (double)10;  
    return d1-d2; // return -17.5;  
}
```

Restituzione del risultato

-17.5

# Come specificare un metodo

Nella documentazione del linguaggio Java un metodo viene specificato solo con la sua intestazione, senza il corpo del metodo.

Esempio di un metodo della classe `String`:

```
public int compareTo(String anotherString)
```

Per chiamarlo si deve utilizzare un oggetto:

```
String s="JAVA";
```

```
if (s.compareTo("java")>0)....
```

Esempio di metodo statico della classe `Integer`:

```
public static int parseInt(String s)throws NumberFormatException
```

Per chiamarlo si deve utilizzare il nome della classe `Integer`:

```
int nr=Integer.parseInt(JOptionPane.ShowDialog("Numero:"));
```

# Metodi nella classe Math

| Metodo Java | Valore restituito                                  | Metodo Java | Valore restituito                          |
|-------------|----------------------------------------------------|-------------|--------------------------------------------|
| abs (x)     | Valore assoluto                                    | min (x, y)  | Minimo tra x e y                           |
| exp (x)     | $e^x$                                              | pow (x, y)  | Elevamento a potenza, $x^y$                |
| log (x)     | ln x                                               | sqrt (x)    |                                            |
| ceil (x)    | L'intero $\geq x$ minimo, convertito in double     | cos (x)     | $\sqrt{x}$                                 |
| floor (x)   | L'intero $\leq x$ massimo, convertito in double    | sin (x)     |                                            |
| round (x)   | Il valore arrotondato di x convertito ad un intero | tan (x)     |                                            |
| max (x, y)  | Massimo tra x e y                                  | random ()   | Numero casuale tra 0.0 e 1.0 (escluso 1.0) |

```

public class Interessi {
    public static void main (String args[]) {
        double ammontare = 10000.0, tasso = 0.05, totale=0.0;
        System.out.println( "Anno\t\tTotale ");
        for (int anno = 1; anno <= 10; anno++) {
            totale = (int) (ammontare * Math.pow(1.0 + tasso, anno));
            System.out.println( anno + "\t\t"+ totale);
        }
    }
}

```

# Stringhe di caratteri

- In Java le stringhe, cioè le sequenze di caratteri, sono memorizzate nelle istanze della classe **String**. Le stringhe **String** sono immutabili.
- `+` è l'operatore di concatenazione delle stringhe.
- Tutti gli oggetti in Java hanno un metodo `toString()` che traduce oggetti in stringhe di caratteri. Per un oggetto `String`, `toString()` riporta la stringa memorizzata nell'oggetto.
- In contesto di stringhe, Java converte tutti i tipi di dati a stringhe, automaticamente chiamando `toString()`  
`System.out.println("sum = " + sum);` //intero sum è convertito a `String`
- Con `new String(. . .)` si possono creare stringhe da altre stringhe:  
`String s = new String(s1);`
- Oppure si può preferire l'utilizzo di una soluzione più diretta con letterali `String`:  
`String s = "a = ";` // `"a ="` è un letterale `String`.  
// L'oggetto viene creato dal compiler

# Metodi della classe String

```
int charAt(int index)
int compareTo(String altroStringa)
boolean endsWith(String suffisso)
boolean equals(Object altroOggetto)
boolean equalsIgnoreCase(String altroStringa)
int indexOf(String str)
int indexOf(String str, int dalIndice)
int lastIndexOf(String str)
int lastIndexOf(String str, int dalIndice)
int length()
String replace(char carVecchio, char carNuovo)
boolean startsWith(String prefisso)
String substring(int indiceInizio)
String substring(int indiceIniz,int indiceFine)
String toLowerCase()
String toUpperCase()
String trim()
```

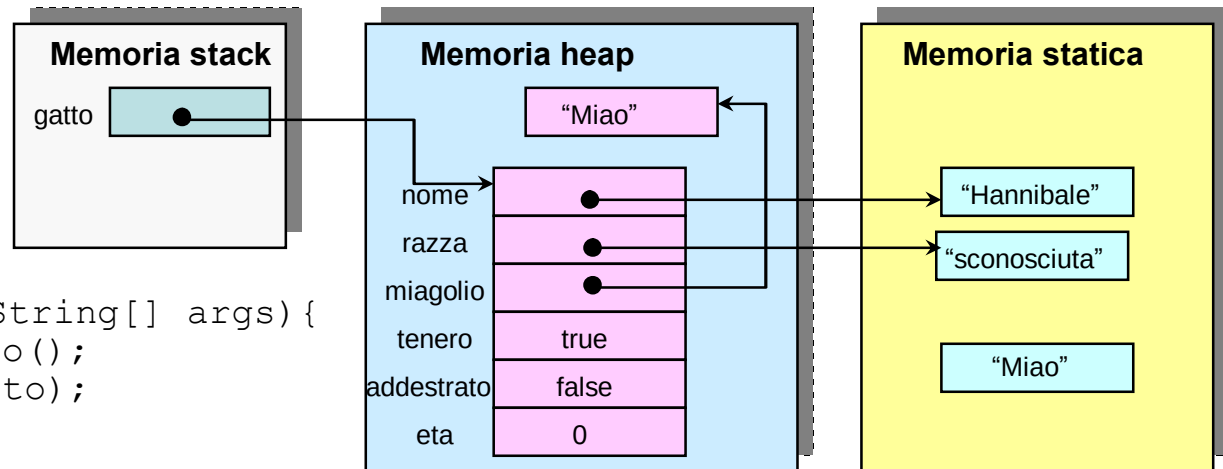
# Come utilizzare le stringhe

Spesso, in una classe definita dal programmatore conviene definire un metodo `toString()` secondo le necessità dell'applicazione.

```
class Gatto {  
    String nome = "Hannibale";    // memorizza il reference di un oggetto statico  
    String razza = "sconosciuta"; // memorizza il reference di un oggetto statico  
    String miagolio = new String("Miao"); // crea un nuovo oggetto String  
    boolean tenero = true;  
    boolean addestrato = false;  
    int eta = 0;  
    . . . . .  
    public String toString() {  
        return "[nome = " + nome + ", miagolo = " + miagolio + ", razza ="  
            + razza + ", eta\' = " + eta + "]" ;  
    }  
}
```

. . . . .

```
public static void main(String[] args){  
    Gatto gatto = new Gatto();  
    System.out.println(gatto);  
}
```

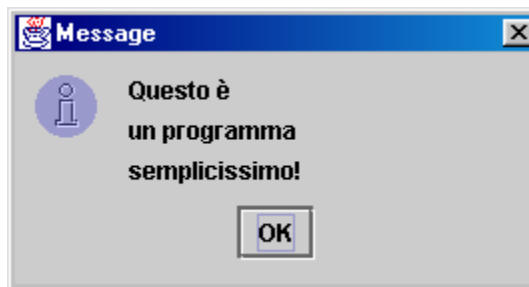


Output:

```
[nome = Hannibale, miagolio = Miao, razza = sconosciuta, eta' = 2]
```

# Comunicare con l'utente: Finestra di messaggio

```
import javax.swing.JOptionPane; // importa la classe JOptionPane
public class Messaggio1 {
    // Il metodo main inizia l'esecuzione di un applicazione Java
    public static void main( String args[] ){
        JOptionPane.showMessageDialog(
            null, "Questo e'\nun programma\nsemplicissimo!" ,
            "Message", JOptionPane.INFORMATION_MESSAGE );
        System.exit( 0 ); // termina l'applicazione
    } // end metodo main
} // end class Messaggio1
```



# Tipi di finestre di dialogo

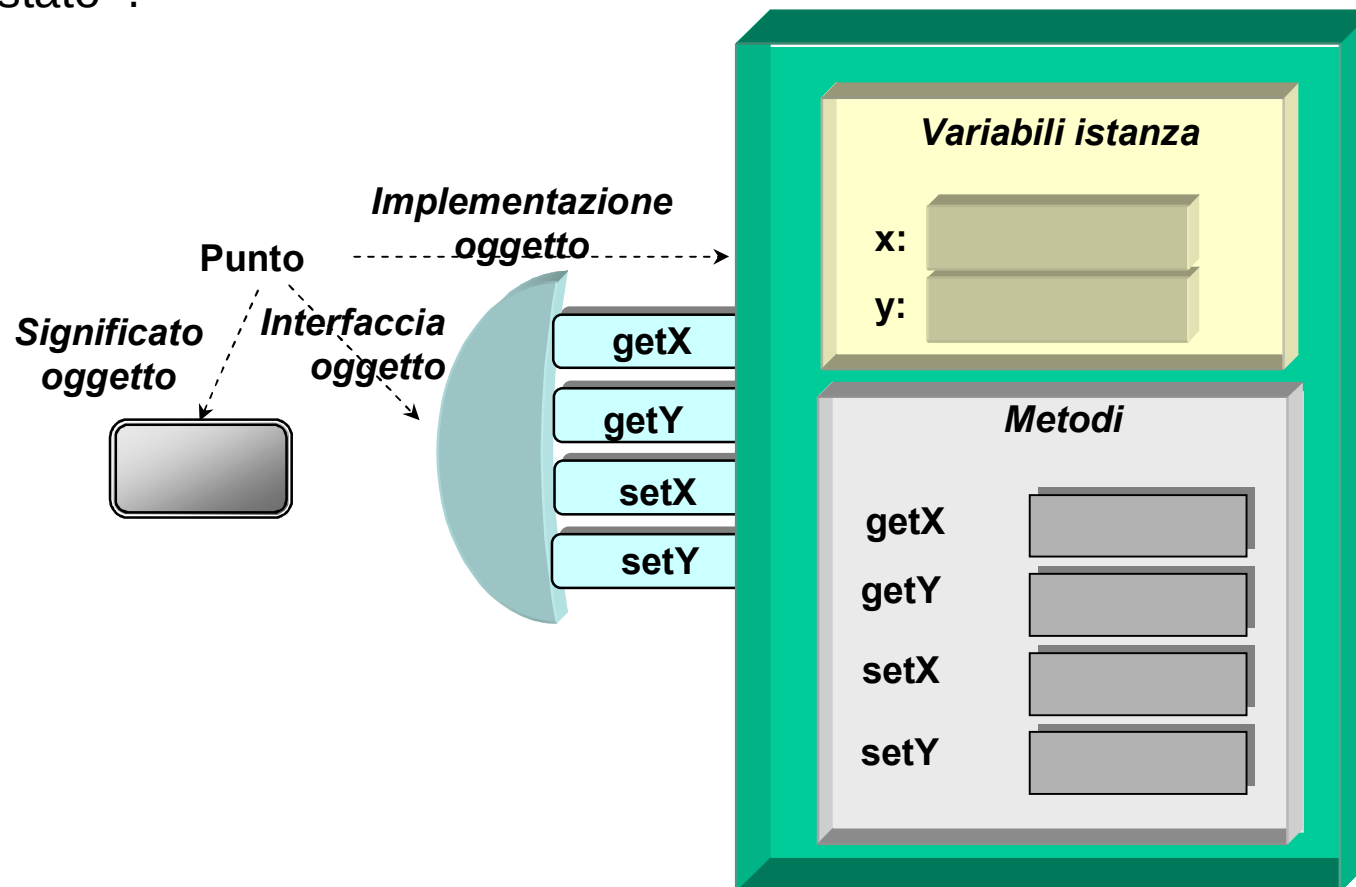
| Tipo messaggio                               | Icona                                                                                | Descrizione                                                                                                                                                |
|----------------------------------------------|--------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>JOptionPane.ERROR_MESSAGE</code>       |   | Visualizza una finestra di dialogo che informa l'utente di un errore                                                                                       |
| <code>JOptionPane.INFORMATION_MESSAGE</code> |   | Visualizza una finestra di dialogo che include un messaggio per l'utente. L'utente può chiudere la finestra                                                |
| <code>JOptionPane.WARNING_MESSAGE</code>     |   | Visualizza una finestra di dialogo che avverte l'utente su un possibile problema                                                                           |
| <code>JOptionPane.QUESTION_MESSAGE</code>    |  | Visualizza una finestra di dialogo che sottopone un quesito all'utente. Il quesito richiede una risposta di tipo Sì o No, effettuata con appositi bottoni. |
| <code>JOptionPane.PLAIN_MESSAGE</code>       | senza icona                                                                          | Visualizza una finestra di dialogo che contiene un messaggio senza icona.                                                                                  |

# Comunicare con l'utente: Finestra di input

```
import javax.swing.JOptionPane; // importa la classe JOptionPane
public class Somma {
    public static void main( String args[] ) {
        String primo;    // la prima stringa inserita dall'utente
        String secondo;  // la seconda stringa inserita dall'utente
        int numero1;      // il primo numero
        int numero2;      // il secondo numero
        int somma;        // somma di numero1 e numero2
        // legge il primo numero come stringa
        primo = JOptionPane.showInputDialog("Inserire il primo numero");
        // legge il secondo numero come stringa
        secondo = JOptionPane.showInputDialog("Inserire il secondo numero");
        // converte i numeri dal tipo String al tipo int
        numero1 = Integer.parseInt( primo );
        numero2 = Integer.parseInt( secondo );
        // somma i due numeri
        somma = numero1 + numero2;
        // visualizza i risultati
        JOptionPane.showMessageDialog(null, "La somma è " + somma,
                                     "Resultati", JOptionPane.PLAIN_MESSAGE );
        System.exit( 0 ); // terminate application
    } // end metodo main
} // end classe Somma
```

# Oggetti

- Oggetto = modo semplificato di identificare una cosa o un fatto nel mondo reale. Questa è un'istanza di un concetto.
- Un oggetto è una combinazione di:
  1. *Informazione di stato* fornita da un insieme di variabili istanza che rappresentano le proprietà dell'oggetto
  2. *Funzionalità* rappresentata da un insieme di metodi che accedono all'informazione di stato .



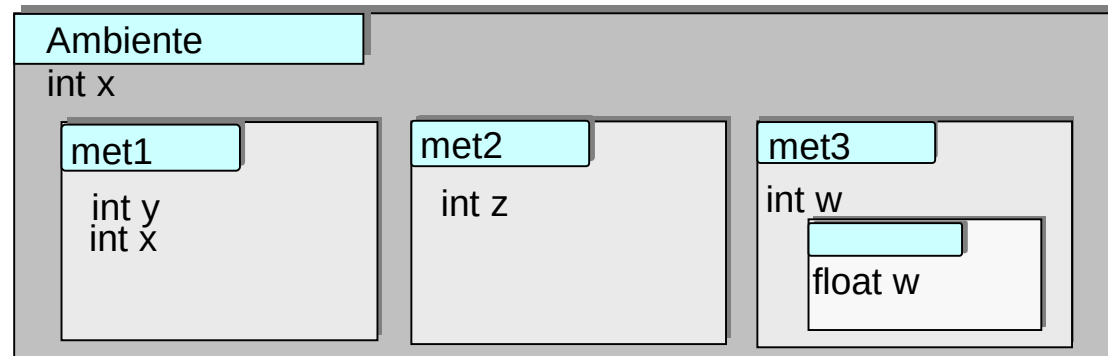
# Dichiarazione di una classe

```
public class Persona {    // Dichiarazione nome classe
    private String nome;    // Dato membro dell'istanza. Variabile di tipo String
    private int eta=21;    // Dato membro dell'istanza. Variabile di tipo int
    private double stipendio=1100; //Dato membro dell'istanza. Variabile di tipo double
    public int getNome() {    // Metodo membro dell'istanza
        return nome;
    }
    public int getEta() {    // Metodo membro dell'istanza
        return eta;
    }
    public void setNome(String stNome) {    // Metodo membro dell'istanza
        nome=stNome;
    }
    public void setEta(int e) { // Metodo membro dell'istanza
        eta=e;
    }
    public void toString() {    // Metodo membro dell'istanza
        return "Nome:"+nome+" Eta:"+eta+" Stipendio:"+stipendio;
    }
    public static void main(String args[]) {    // Metodo membro della classe
        Persona mioAmico = new Persona(); //Creazione di un'istanza della classe
        mioAmico.setNome("Paolo"); // Valorizzare la variabile nome per mioAmico
        Persona tuaAmica = new Persona(); //Creazione di un'istanza della classe
        tuaAmica.setNome("Maria"); // Valorizzare la variabile nome per tuaAmica
        System.out.println(mioAmico.toString());
        System.out.println(tuaAmica);
    }
}
```

# Domini di visibilità

```
public class Ambiente {  
    int x=0;    // variabile istanza, globale nell'oggetto  
    void met1() {  
        int y, x=5; // dichiarazione di variabile locale  
        y=x;        // y==5 e non y==0!  
    }  
    void met2() {  
        int z=1;    // dichiarazione di variabile locale  
        x=y+z;      // errore! y non è un identificatore conosciuto qui  
    }  
    void met3() {  
        met1();    // chiamata di un altro metodo dell'oggetto  
        int w=x*x;  
        {        // inizia un dominio di visibilità incluso in met3  
            x+=2;  
            float w=1.2*x; // variabile locale  
            x=(int) (w-2.2);  
        }        // qui finisce il dominio di visibilità incluso in met3  
        x*=w;  
    }  
}
```

Diagramma  
di contorno



# Ambito di visibilità ed ambito di esistenza delle variabili

1. **Ambito di visibilità (scope)** = Regione nel programma in cui i nomi appartengono allo stesso spazio di nomi. In questo spazio un nome è unico e denomina una certa entità del programma.
2. Le variabili dal punto di vista dell'ambito di visibilità possono essere:
  - **Globali (variabili istanza e variabili della classe)**
  - **Locali (variabili locali)**
3. **Ambito di esistenza (lifetime)** = periodo durante l'esecuzione del programma in cui una variabile esiste.

Variabili della classe: esistono per l'intera esecuzione del programma.

Variabile istanza: esistono per la durata di esistenza dell'oggetto in cui si trovano.

Variabili locali: esistono per la durata di esecuzione del blocco in cui sono dichiarate.

4. **Notazione puntata**. Si utilizza per chiamare un metodo-membro di un oggetto:

```
personale[i].aumentoStipendio(5)
```

o per identificare le variabili-membro di un oggetto:

```
rettangolo.origin.y
```

# Costruzione di oggetti

**Costruttore** di una classe = Un algoritmo che descrive come si deve inizializzare le variabili-istanza del nuovo oggetto. Un costruttore ha lo stesso nome con la classe. È molto simile ad un metodo-istanza della classe però non ha un tipo di ritorno.

```
public Persona() { //Costruttore(overloaded)
}
public Persona(String nome,int eta, double stipendio){ // overloaded)
    this.nome=new String(nome);
    this.eta=eta;
    this.stipendio=stipendio
}
```

• . . .

```
Persona tuoAmico = new Persona();
Persona mioAmico = new Persona("Pippo",25,1800.50);
```

- Il costruttore non si chiama come qualunque metodo. Viene impegnato dall'operatore **new** per creare un nuovo oggetto:

```
new NomeClasse(Parametri);
           costruttore
```

- Un costruttore può avere parametri, *ma non restituisce risultati*.
- Una classe può avere più costruttori (overloading dei costruttori).
- **this** = Parola chiave che si utilizza all'interno del corpo di un metodo di un'istanza e che produce un reference all'oggetto per cui è stato richiamato il metodo.

# Costruttori

```
public class Persona {  
    private String nome;           // variabile istanza private  
    private int eta;               // variabile istanza private  
    private double stipendio;      // variabile istanza private  
    public Persona() {}           //Costruttore predefinito (overloaded)  
    public Persona(String n, int e, double s){ //Costruttore(overloaded)  
        nome=new String(n);  
        eta=e;  
        stipendio=s;  
    }  
    public int getNome() {         // Metodo  
        return nome;  
    }  
    public int getEta() {         // Metodo  
        return eta;  
    }  
    public double getStipendio(){ // Metodo  
        return stipendio;  
    }  
    public void setNome(String stNome) { // Metodo  
        nome=stNome;  
    }  
    public void setEta(int e) {    // Metodo  
        eta= e;  
    }  
    public String toString () {    // Metodo  
        return "Nome:"+nome+" Eta:"+eta+" Stipendio:"+stipendio;  
    }  
    public static void main(String args[]) { //Da qui parte l'esecuzione  
        Persona mioAmico = new Persona("Pippo Rossi", 21, 1200.0);  
        System.out.println( mioAmico.toString());  
        Persona tuoAmico = new Persona();  
        tuoAmico.setNome("Marco Valenti");  
        tuoAmico.setEta(28);  
        tuoAmico.setStipendio(1850.5);  
        System.out.println( tuoAmico);  
    }  
}
```

# Modificatori

- modificatori di accesso (visibilità)
  - modificatore non specificato
  - `public`
  - `private`
  - `protected`.
- `static`.
- `final`
  - per variabili: trasforma una variabile in una costante  
`final String ragioneSociale = "Datamat S.p.A.";`
  - per metodi: non si possono più ridefinire.
  - per classi: non possono più essere estese.
- `synchronized`.
- `native`.
- `abstract`.

# Memberi statici di una classe

1. I metodi e le variabili non dichiarate con “static” sono metodi e variabili delle istanze.
2. I metodi e le variabili dichiarate “static” sono metodi e variabili della classe.

## Dati statici

### Definizione

```
class Ordine {  
    String ragioneSociale;  
    String p_IVA;  
    int nrOrdine;  
    Data data;  
    static int totaleOrdini=0;  
    . . .  
}
```

**Accesso:** utilizzare un oggetto o la classe

1. Ordine nuovo = new Ordine (. . .);  
 . . .  
 nuovo.totaleOrdini++;
2. Ordine.totaleOrdini++;

## Metodi statici

### Definizione

```
class Ordine {  
    String ragioneSociale;  
    String p_IVA;  
    int nrOrdine;  
    Data data;  
    static int totaleOrdini = 0;  
    static void cancella() {  
        totaleOrdini = 0;  
    }  
}
```

**Accesso:** utilizzare un oggetto o la classe

1. Ordine nuovo = new Ordine (. . .);  
 . . .  
 nuovo.cancella();
2. Ordine.cancella();

# Il modificatore "static"

```
class Esempio1 {
    static String text0="Un programma semplice!";
    public static void main(String[] args){
        System.out.println(text0);//OK. Var. statica in un metodo statico della stessa classe
    }
}

class Esempio2 {
    String text="Un altro programma semplice!";
    String getText(){
        return "Text: " + text;//OK. Var. istanza in un metodo istanza della stessa classe
    }
    public static void main(String[] args){
        Esempio2 istanza = new Esempio2();
        String unText = istanza.getText(); // OK. Chiamata di metodo istanza con oggetto
        System.out.println(Esempio1.text0 + "\n" + istanza.text + "\n" + unText); // OK
        System.out.println(text);// errore: Var. istanza senza oggetto in metodo statico
    }
}

class Esempio3 {
    String text="Un programma semplice!";
    static String text2="Un programma semplice!";
    String getText(){
        return text; // OK. Var. statica in un metodo istanza della stessa classe
    }
    String getStaticText(){
        return text2; // OK. Var. statica in un metodo istanza della stessa classe
    }
    static String getTextInStatic(){
        return text2; // OK. Var. statica in un metodo statico della stessa classe
    }
    public static void main(String[] args){
        Esempio3 istanza = new Esempio3();
        String unText = istanza.getText();// Un oggetto chiamato con un suo metodo istanza
        String unTextStatic = istanza.getTextInStatic();// Un oggetto chiamato con un
                                                    // metodo statico della sua classe
        System.out.println(istanza.text); //Utilizzo di una variabile istanza dell'oggetto
        System.out.println(getTextInStatic()); // Chiamata di metodo statico in un
                                                    // metodo statico della stessa classe
    }
}
```

# Struttura di un programma Java

```
import java.util.*;
```

*Classi da importare*

```
public class TestDipendente {
```

```
    public static void main(String[] args) {  
        Dipendente[] personale = new Dipendente[3];  
        personale[0] = new Dipendente("Paolo Rossi", 3500.00);  
        personale[1] = new Dipendente("Carlo Cacciari", 4500.00);  
        personale[2] = new Dipendente("Tonino Teste", 3800.00);  
        int i;  
        for (i = 0; i < 3; i++) personale[i].aumentoStipendio(5);  
        for (i = 0; i < 3; i++) personale[i].print();  
    }
```

*Metodo **main()** della classe*

```
}
```

```
class Dipendente {
```

```
    private String nome;  
    private double stipendio;
```

*Variabili dell'oggetto*

```
    Dipendente(String n, double s) {  
        nome = n;  
        stipendio = s;  
    }
```

*Costruttore dell'oggetto*

```
    public String getNome() {  
        return nome;  
    }
```

*Metodo dell'oggetto*

```
    public void aumentoStipendio(double percent) {  
        stipendio *= 1 + percent / 100;  
    }
```

*Metodo dell'oggetto*

```
    public String toString() {  
        return "Nome: " + nome + ", Stipendio: " + stipendio;  
    }
```

*Metodo dell'oggetto*

```
}
```

# Array

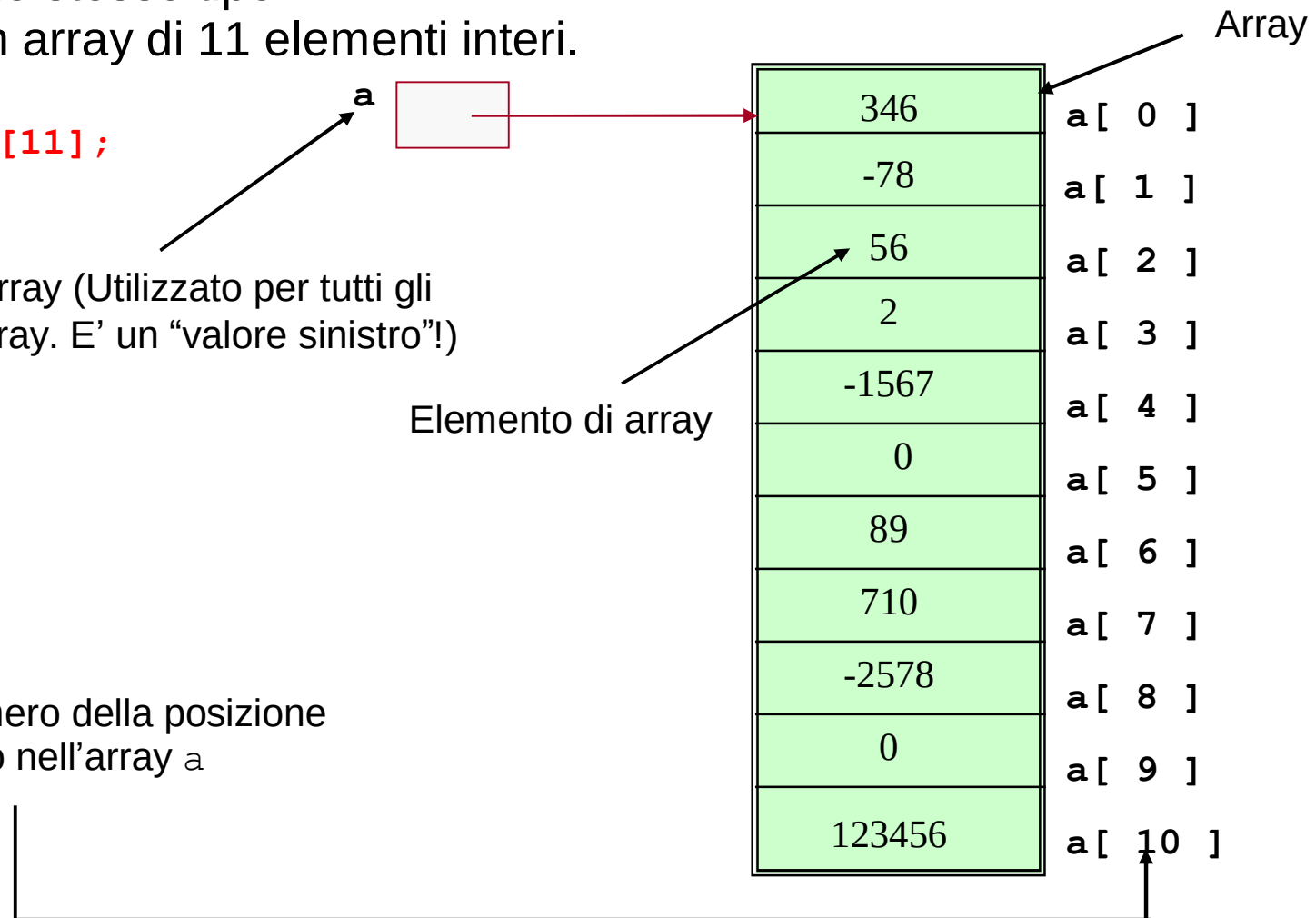
**Array** = Una collezione lineare di elementi in cui ciascun elemento è accessibile attraverso un indice.

- Un costrutto array fornisce spazio per valori di tipi primitivi o di reference ad oggetti di uno stesso tipo.
- **Esempio:** un array di 11 elementi interi.

```
int a[];  
a = new int[11];
```

a - Nome dell'array (Utilizzato per tutti gli elementi dell'array. E' un "valore sinistro"!)

indice = Numero della posizione dell'elemento nell'array a



# Array

- Un array si deve memorizzare in una apposita variabile. La dichiarazione di una variabile array contiene “[ ]”.

```
int[] numeri;  
Persona personale[];
```

- Un array, per esistere nel programma, deve essere creato (allocato) con **new()**:

```
numeri = new int[5];  
personale = new Persona[1000];
```

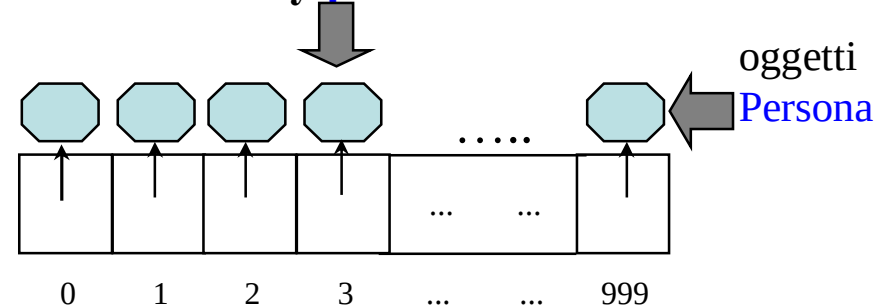
- Dopo la creazione si possono riempire con valori gli elementi dell'array.
- L'indici di un'array iniziano da 0. Un'array di 10 elementi avrà indici da 0 a 9.

array **numeri**

|     |     |     |     |   |
|-----|-----|-----|-----|---|
| 101 | 837 | -27 | -99 | 5 |
| 0   | 1   | 2   | 3   | 4 |

array unidimensionale di 5 interi  
**Persona**

array **personale**



array unidimensionale di 1000 oggetti

- Lunghezza di un'array = Numero di elemento dell'array.
- Per ogni array esiste una variabile **length** che dà la lunghezza:  
`personale.length`
- Un array può avere più dimensioni, cioè più indici attraverso cui gli elementi si ordinano tra loro.

# Inizializzare array

- Al momento della creazione di un array, i suoi elementi sono assegnati sia a zero quando sono tipi primitivi) o a **null** quando sono tipi reference.

```
char[] testo;    // variabile-reference ad un array di caratteri
int[] valori = {11, 7, 89};    // array creato e inizializzato
int numeri[] = {101, 837, -27, -99, 5}; // array creato e inizializzato
String giorno[] = {"Lunedì", "Martedì", "Mercoledì", "Giovedì", "Venerdì",
                   "Sabato", "Domenica"}; // array inizializzato con stringhe
String mese[] = { "Gennaio", "Febbraio", "Marzo", "Aprile", "Maggio", "Giugno",
                  "Luglio", "Agosto", "Settembre", "Ottobre", "Novembre",
                  "Dicembre"}; // array inizializzato con stringhe
long lista[][]; // variabile-array con due dimensioni
float[][] coordinate = new float[10][15]; // allocazione dell'array

int[] arrayDiInt = new int[100]; // creazione della variabile e allocazione array
for (int i=0; i<arrayDiInt.length; i++)
    arrayDiInt[i]=i;              // riempimento dell'array con interi

Integer[] arrayDiInteger = new Integer[100];
for (int i=0; i < arrayDiInteger.length; i++)
    arrayDiInteger[i] = new Integer(i); // riempimento dell'array con oggetti
```

- Metodo che permette di copiare un numero di elementi consecutivi da un array in un altro:

```
System.arraycopy(arraySorg, posSorg, arrayDest, posDest, nrVoci);
```

# Lavorare con array

```
import java.util.*;
public class ArrayDiInteri {
    public static void main(String[] args) {
        int[] a1 = { 1, 2, 3, 4, 5 };
        int[] a2;
        a2 = a1;
        for(int i = 0; i < a2.length; i++)
            a2[i]++;
        for(int i = 0; i < a1.length; i++)
            prt("a1[" + i + "] = " + a1[i]);
    }
    static void prt(String s) {
        System.out.println(s);
    }
}
```

```
import java.util.*;
public class ArrayDiOggetti {
    static Random rand = new Random();
    static int pRand(int mod) {
        return Math.abs(rand.nextInt()) % mod;
    }
    public static void main(String[] args) {
        Integer[] a = new Integer[pRand(20)];
        stampa("length of a = " + a.length);
        for(int i = 0; i < a.length; i++) {
            a[i] = new Integer(pRand(500));
            stampa("a[" + i + "] = " + a[i]);
        }
    }
    static void stampa(String s) {
        System.out.println(s);
    }
}
```

# Utilizzo di array nei metodi

```
class Gatto {
    String nome = "nessuno";
    String razza = "sconosciuta";
    String miagolio = new String("Miao");
    boolean tenero = true;
    boolean addestrato = false;
    int eta = 0;
    String[] dietaDelGiorno = null;
    . . .
    void setDieta(String[] dieta) {    // array come parametro di metodo
        dietaDelGiorno = new String[dieta.length];
        for (int i = 0; i < dieta.length; i++)
            dietaDelGiorno[i] = dieta[i];
    }
    String[] getDieta() {    // array come risultato di un metodo
        return dietaDelGiorno;
    }
    void stampaDieta() {
        if (dietaDelGiorno == null)
            System.out.println("Manca la dieta per il gatto " + getNome());
        else {
            System.out.println("Dieta per il gatto " + getNome() );
            for (int i = 0; i < dietaDelGiorno.length; i++)
                System.out.println(dietaDelGiorno[i]);
        }
    }
}
```

# Array multidimensionali

Dichiarazione di una variabile di un'array multidimensionali:

*tipo identificatore*[*dim1*][*dim2*]. . . [*dimN*];

Esempi:

1.

```
int [][] matrix;        // dichiarazione variabile-array di 2 dimensioni
matrix = new int[4][6]; // creazione array e inizializzazione variabile
```

2.

```
import java.util.Calendar;
public class TestDietaGatto {
    public static void main(String[] args) {
        Gatto ettore = new Gatto ("Ettore");
        String[][] dietaSettimanale = {
            {
                "50 grammi carne",
                "50 grammi pesce",
                "100 grammi acqua"
            },
            {
                "100 grammi cibo in scatola",
                "50 grammi biscotti",
                "150 grammi acqua",
                "50 grammi carne"
            },
            . . .
        };
        Calendar c = Calendar.getInstance();
        ettore.setDieta(dietaSettimanale[c.get(Calendar.DAY_OF_WEEK)]);
        ettore.stampaDieta();
    }
}
```