# metodo

- Termine caratteristico dei linguaggi OOP
- Un **insieme di istruzioni con un nome**
- Uno strumento per risolvere gradualmente i problemi scomponendoli in **sottoproblemi**
- Uno strumento per **strutturare** il codice
- Uno strumento per **ri-utilizzare** il lavoro già svolto
- Uno strumento per rendere il **programma più chiaro** e leggibile

# quando e perché usare i metodi

1. Quando il programma da realizzare è articolato diventa conveniente identificare **sottoproblemi** che possono essere risolti individualmente
2. scrivere **sottoprogrammi** che risolvono i sottoproblemi richiamare i **sottoprogrammi** dal programma principale (main)
3. Questo approccio prende il nome di **programmazione procedurale** (o astrazione funzionale)
4. In Java i **sottoprogrammi** si realizzano tramite **metodi ausiliari**
5. Sinonimi usati in altri linguaggi di programmazione: **funzioni**, **procedure** e (sub)**routines**

- **metodi statici**: dichiarati `static`
- richiamabili attraverso nome della classe
- p.es: `Math.sqrt()`

```
public class ProvaMetodi
{
        public static void main(String[] args) {
                stampaUno();
                stampaUno();
                stampaDue();
        }

        public static void stampaUno() {
                System.out.stampaln("Hello World");
        }

        public static void stampaDue() {
                stampaUno();
                stampaUno();
        }
}
```

# Metodi non static

- I metodi **non static** rappresentano operazioni effettuabili su singoli oggetti
- La documentazione indica per ogni metodo il tipo ritornato e la lista degli argomenti formali che rappresentano i dati che il metodo deve ricevere in ingresso da chi lo invoca
- Per ogni argomento formale sono specificati:
  - un tipo (primitivo o reference)
  - un nome (identificatore che segue le regole di naming)

# Invocazione di metodi non static

- L’invocazione di un metodo non static su un oggetto istanza della classe in cui il metodo è definito si effettua con la sintassi:
- Ogni volta che si invoca un metodo si deve specificare una lista di argomenti attuali
- Gli argomenti attuali e formali sono in corrispondenza posizionale
- Gli argomenti attuali possono essere delle variabili o delle espressioni
- Gli argomenti attuali devono rispettare il tipo attribuito agli argomenti formali
- La documentazione di ogni classe ( istanziabile o no) contiene l’elenco dei metodi disponibili
- La classe **Math** non è istanziabile
- La classe **String** è "istanziabile ibrida"
- La classe **StringBuilder** è "istanziabile pura"

## Metodi predicativi

Un metodo che restituisce un tipo primitivo `boolean` si definisce **predicativo** e può essere utilizzato direttamente in una condizione.
In inglese sono spesso introdotti da `is` oppure `has`: isMale(), hasNext().

# Esempi sui metodi ausiliari