RICORSIONE

INTRODUZIONE

- Esempio di funzione ricorsiva: calcolare **fattoriale**
  - metodo che invoca se stesso

```
public static int factorial(int n) {
    if (n < 0) throw new IllegalArgumentException();
    if (n == 0) return 1;
    return n * factorial(n - 1);
}
```

PARADIGMA DI PROGRAMMAZIONE RICORSIVO

- Invocare un metodo mentre si esegue lo stesso metodo
  - sfrutta l'idea di suddividere un problema in sotto-problemi più semplici
- Cosa fa la JVM con la chiamata di un metodo ricorsivo?
  - sospende esecuzione metodo invocante
  - esegue il metodo invocato fino alla sua terminazione
  - riprende l'esecuzione del metodo invocante dal punto in cui era stato sospeso
- Non è mai **necessaria** perché può essere sempre tradotta in modo non ricorsivo
  - l'**interprete** si fa carico di eliminare la ricorsione in runtime

STRUTTURA ALGORITMO RICORSIVO

- Caso base: valore di partenza noto
  - possono esserci più casi base
- Passo ricorsivo: metodo che chiama se stesso avvicinandosi al caso base

> ⚠ **Ricorsione infinita**
>
> Se manca caso base o non si semplifica il problema ad ogni iterazione, il programma terminer
> con l'eccezione "StackOverflowError"

RICORSIONE SEMPLICE

- Invoca il metodo ricorsivo una volta sola

- È possibile tradurla con un ciclo while
  - ricorsione **meno efficiente** del ciclo while

RICORSIONE IN CODA (TAIL RECURSION)

- Il metodo invoca se stesso come ultima azione

RICORSIONE MULTIPLA

- Quando un metodo invoca se stesso più volte durante la propria esecuzione
- Molto lenta
- Esempio: calcolo dei numeri di Fibonacci
  - $F_n = F_{n-1} + F_{n-2}$
- Si può eliminare la ricorsione doppia facendo restituire un array contenente gli ultimi 2 eleme