# ORALE DI PROGRAMMAZIONE

## 1 METODI

**Intestazione** definire nome, valore di ritorno e parametri di un metodo

**Variabili locali** usate e dichiarate internamente al metodo

↳ aventi lo stesso nome, ma dichiarate in metodi diversi, sono variabili distinte

**Parametro attuale** detto anche **argomento**, è il valore effettivo

**Parametro formale** quello definito nell'intestazione, rappresenta il valore che sarà disponibile all'invocazione del metodo

↳ all'invocazione, ogni parametro viene inizializzato al valore dell'argomento (**chiamata per valore**).

↳ Il valore dell'argomento non può essere modificato (solo con tipi primitivi)

↳ Il tipo deve essere lo stesso, ma vale il casting implicito

**Math** fornisce metodi che definiscono delle operazioni matematiche

**Record di attivazione** contiene tutte le informazioni necessarie per gestire correttamente l'esecuzione di un metodo invocato

↳ struttura dati che contiene:

- Parametri (con gli argomenti attuali)
- Variabili locali
- Indirizzo di rientro per ritornare al chiamante
- Risultato, il valore di ritorno

↳ Viene creato dinamicamente nel momento del richiamo:

1. viene posto in cima allo **stack** (area di memoria statica)
2. rimane qui per tutto il tempo dell'esecuzione
3. viene rimosso al termine dell'esecuzione

↳ Sono gestiti con politica L.I.F.O.

**Driver** serve per verificare la correttezza di un metodo (che è bene collaudare individualmente)

**Stub** un prototipo, una versione semplificata del metodo che può essere usata per testare senza avere la versione collaudata del metodo

## 2 ARRAY

**Avvay** è una sequenza di variabili di tipo omogeneo, distinguibili l'una dall'altra in base alla loro posizione nella sequenza (**indice**)

↳ è un particolare tipo di oggetto

**Dichiarazione** avviene mediante:

- l'operatore **new**
- in fase di dichiarazione con **{}**

**Length** proprietà che definisce la lunghezza dell'array

↳ è la capacità, non il numero di elementi usati

**Operatori** una variabile di tipo array contiene l'indirizzo in cui l'array è memorizzato in memoria (**reference**), in particolare:

- **=** copia l'indirizzo
- **==** verifica se due array sono memorizzati nella stessa area di memoria

**Multidimensionali** array con due indici [riga][colonna], quindi un array di array.

↳ le righe non devono avere per forza la stessa lunghezza

## 3 RICORSIONE

**Definizione** quando una parte di algoritmo contiene una versione ridotta dell'algoritmo completo

**Iterazione** ogni algoritmo ricorsivo ha una versione iterativa

↳ il metodo ricorsivo utilizza più memoria ed è un'esecuzione più lenta

**Memoria** ogni richiamo ricorsivo comporta la creazione di un nuovo record di attivazione, che si posiziona in cima allo stack (L.I.F.O.)

↳ c'è sempre un limite alle dimensioni dello stack (se si estende oltre i limiti, va in **stack overflow**, come ad esempio quando le chiamate ricorsive non hanno fine)

## 4 PASSAGGIO DI PARAMETRI

**Tipo primitivo** vengono allocati nello stack come copia, modificati e poi deallocati

**Non primitivo** come gli array, nello stack c'è un riferimento all'area di memoria nello heap che contiene i valori. Il parametro non è altro che un nuovo riferimento

OGNI MODIFICA INFLUISCE SUI VALORI ORIGINALI, NON SU UNA COPIA