

# Programmazione ad Oggetti

## Classi e Oggetti

---

A.A. 2022/2023

---

*Docente: Prof. Salvatore D'Angelo*  
*Email: [salvatore.dangelo@unicampania.it](mailto:salvatore.dangelo@unicampania.it)*



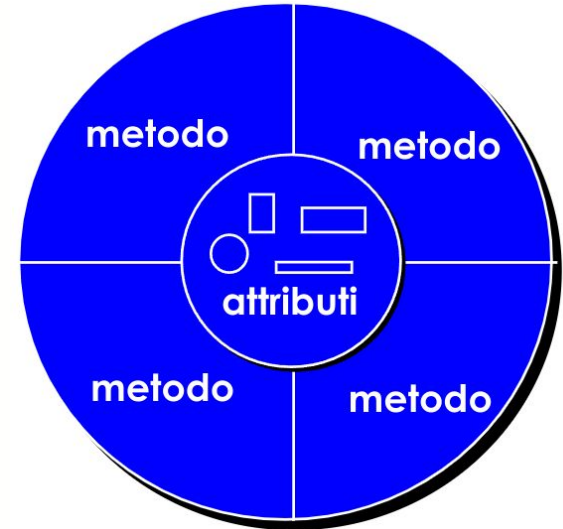
Università  
degli Studi  
della Campania  
*Luigi Vanvitelli*

*Dipartimento di Ingegneria*

# Tipo Dati Astratto

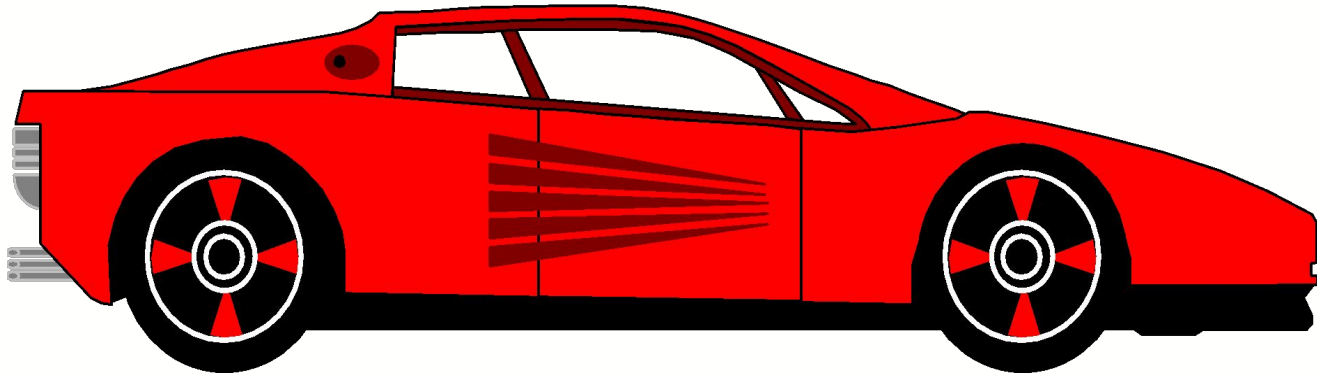
## ASTRAZIONE SUI DATI

Il tipo di dati astratto (ADT)



una classe ha un nome, e contiene due tipi di membri: attributi e metodi

# Esempio: Un'automobile



Funzioni      Dati:

- |            |                   |
|------------|-------------------|
| - Avviati  | - Targa           |
| - Fermati  | - Colore          |
| - Accelera | - Velocità        |
| - ...      | - Livello benzina |

# Le classi

- Java è un linguaggio di programmazione orientato agli oggetti (object oriented), che si basa sul concetto di classe
- La classe è identificata univocamente da un nome, che deve essere necessariamente uguale al nome del file che la contiene (affinché la classe possa essere eseguita)
- La classe ed è caratterizzata da una coppia di parentesi graffe { } che contiene il corpo della classe ossia le istruzioni e le dichiarazioni

Esempio: (File sorgente con nome Prova.java)

```
[modifiers] class Prova {  
    corpo della classe  
}
```

I nomi delle classi possono essere precedute da parole chiave [modifiers]: public, private, static, ...

# Le classi in JAVA

Definizione di una classe vuota:

```
public class Auto {  
    /* Class body goes here */  
}
```

Creazione di un oggetto:

```
Auto panda = new Auto();
```

*Non posso fare niente con una classe vuota!!!!  
Occorre aggiungere attributi e metodi !!!!*

# Gli attributi

***[modificatore]*** ***Tipo*** ***nomeAttributo*** =  
***[Valore]***

Il ***modificatore*** può essere definito: public, privato, ..... (per default è public)

# Esempio classe auto

```
Public class Auto {  
    public int  cilindri=4;  
    public int  speed;           // inizializzazione  
    public String targa;  
}
```

```
Auto panda=new Auto();
```

Ho una classe costituita solo di attributi !!!!!

Unica cosa che possiamo fare è accedere agli attributi per modificarli o utilizzarli:

```
panda.speed =10;  
System.out.println ( panda.cilindri );
```

# I metodi

Ogni classe può contenere uno o più metodi

I metodi sono caratterizzati da un nome e una coppia di graffe {} che contiene la specifica procedura

## Syntax

```
[modifiers]return_type method_identifier([arguments])  
{ method_code_block }
```

I nomi dei metodi possono essere precedute da parole chiave [modifiers]: *public*, *private*, *static*, ..

```
public class Auto {  
    public int    cilindri=4;  
    public int    speed=0;  
  
    public String targa;  
    public void  accelera () { speed++; }  
}
```



# \*I metodi

Il termine *metodo* in Java viene preferito al termine *funzione* (ovvero un sottoprogramma con un valore di ritorno)

```
[modifiers] returnType  methodName ( /*  
Argument list */ ) { /*  
    Method body  
    */ }
```

*I metodi in Java possono essere definiti solo come parti di una classe*

# I parametri di ritorno

```
// questo metodo ritorna il numero di byte della  
stringa s
```

```
public int storage (int s) {  
    return s = 2;  
}
```

Oggetti e tipi semplici vengono passati allo stesso modo  
Il parametro si utilizza nel metodo come una qualunque  
variabile

**return** ha due funzioni:

- Terminazione
- Restituisce il valore di ritorno

# Esempi di terminazione

```
public boolean flag() { return true; }
```

```
public float naturalLogBase() { return 2.718f; }
```

```
public void nothing() { return; }
```

```
public void nothing2() {}
```

# Esempio classe auto

```
class Auto{  
public int   cilindri=4;  
public int   speed=0;  
public String targa;  
  
public int getSpeed(){return speed;};  
public void setSpeed(int s){speed= s;};  
public int getCilindri(){return cilindri};  
    // Tre alternative ???  
    public void setTarga(){targa = " XF345PF"; }  
    public void setTarga(String s){ targa = s;}  
}
```

# Chiamata di un metodo

Esempio:

- Supponiamo di avere un metodo `f()` che non ha parametri e ritorna in tipo `int`. □ `public int f()`
- Supponendo di aver un oggetto `a` per il quale è possibile chiamare `f()` □ `int x = a.f();`

```
panda.setSpeed(100);  
int s = panda.getSpeed();
```

Il tipo ritornato da `f` deve essere compatibile con il tipo di `x`

# **VARIABILI LOCALI DI UN METODO**

# VARIABILI LOCALI DI UN METODO

- Le variabili locali a un metodo:
  - sono visibili solo dal corpo del metodo
  - vengono allocate (nello stack di run-time) alla chiamata e deallocate all'uscita del metodo
  - non vengono inizializzate automaticamente (diversamente dai campi di una classe)
- Non si può accedere a una variabile a cui non si sia prima assegnato un valore (e viene segnalato in compilazione !)

## Esempio:

```
int i;  
if ( cond ) { i = 55; ... }  
i++;    /* compile-time error */
```

# VARIABILI LOCALI DI UN METODO

```
public void multiply (int num1, int num2) {  
    int result = num1* num2;  
}  
public void foo ( ) {  
    int x = 5, y = 12;  
    multiply (x, y);  
}
```

foo()

Stack Memory





# VARIABILI LOCALI DI UN METODO

```
public void multiply (int num1, int num2) {  
    int result = num1* num2;  
}  
public void foo ( ) {  
    int x = 5, y = 12;  
    multiply (x, y);  
}
```

foo()

Stack Memory



# VARIABILI LOCALI DI UN METODO

```
public void multiply (int num1, int num2) {  
    int result = num1* num2;  
}  
public void foo ( ) {  
    int x = 5, y = 12;  
    multiply (x, y);  
}
```

foo()

Stack Memory

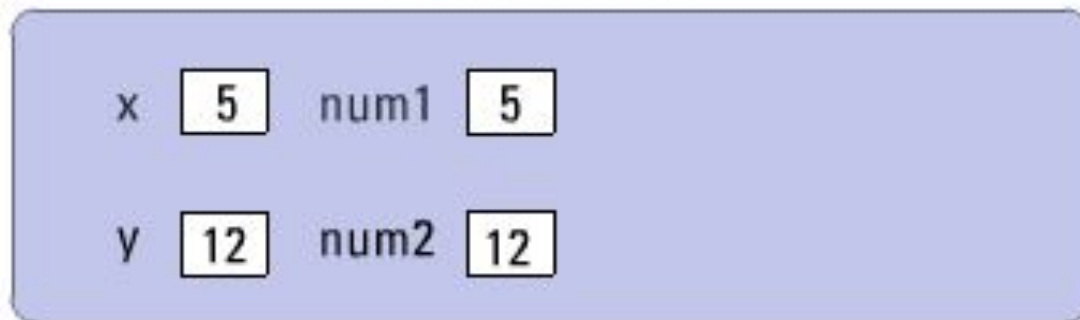


# VARIABILI LOCALI DI UN METODO

```
public void multiply (int num1, int num2) {  
    int result = num1 * num2;  
}  
public void foo ( ) {  
    int x = 5, y = 12;  
    multiply (x, y);  
}
```

foo()

Stack Memory

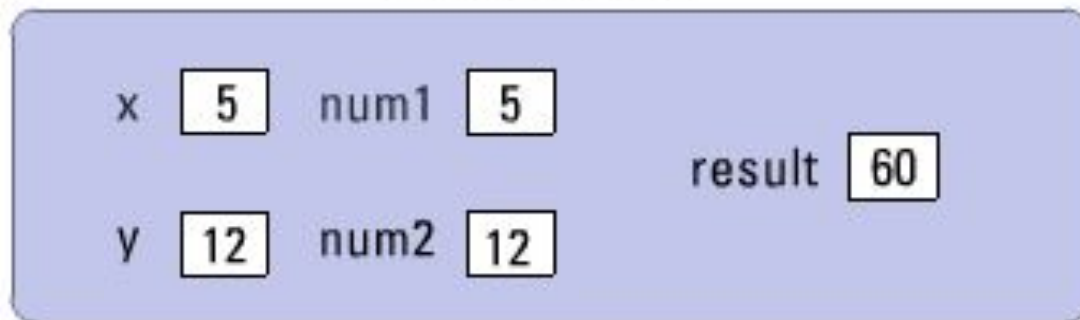


# VARIABILI LOCALI DI UN METODO

```
public void multiply (int num1, int num2) {  
    int result = num1 * num2;  
}  
  
public void foo ( ) {  
    int x = 5, y = 12;  
    multiply (x, y);  
}
```

foo()

Stack Memory



# VARIABILI LOCALI DI UN METODO

```
public void multiply (int num1, int num2) {  
    int result = num1* num2;  
}  
public void foo ( ) {  
    int x = 5, y = 12;  
    multiply (x, y);  
}
```

foo()

Stack Memory



# SCAMBIO DI PARAMETRI

# Tutto viene passato per valore

La lista dei parametri deve specificare il tipo e l'ordine  
Il compilatore dà errore se il tipo del parametro passato è diverso da quello definito nel prototipo.

Dei tipi semplici viene passato il valore  
Degli oggetti viene passato il riferimento (praticamente il puntatore)

Come si fa a passare un parametro di uscita ?????

**Solo oggetti possono essere parametri di uscita**

Perché ? In che modo ?

**L'esempio più semplice**  
**HELLO WORLD**



# Esempio - Hello World!

Affinché un applicazione ad oggetti “parta”, devo avere una classe con un metodo statico e pubblico di nome main

Hello.java

```
class Hello {  
    public static void main (String args []){  
        System.out.println("Hello World!");  
    }  
}
```

obbligatorio in questa forma

# Esempio - Hello World!

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

- **void** indica che *main* non ritorna nulla, il che è necessario per superare il type-checking del compilatore
- **args[]** sono gli argomenti passati a *main* dalla shell quando si digita: **java Hello arg1 arg2 ... argn**
- **String** dice che gli argomenti sono di classe **String**
- **public** rende il metodo *main* visibile alle altre classi - e al comando **java** (interprete)
- **static** associa *main* alla classe *Hello*, e non alle sue istanze

**System.out.println("HelloWorld!")**

- invoca il metodo *println* dell'oggetto *out* della classe *System*, che stampa la stringa sul file *stdout*

# Passare argomenti all'applicazione da linea di comando

Per eseguire un'applicazione java e passargli un argomento, basta far seguire al nome dell'applicazione il valore desiderato: verrà interpretato come una stringa e memorizzato nell'array di stringhe `args[]` definito in `main`

Esempio: **`java Shirt verde`**

```
// Determina se passata una stringa sulla riga di comando

public class Shirt {
    public static void main(String args[]) {
        System.out.println("Stringa digitata: " + args[0]);
    }
}
```

# La classe String

# Stringhe e classi String

- Una stringa è una sequenza di caratteri Unicode racchiusa tra virgolette "Massimo"
- Non esiste il tipo primitivo stringa, ma esiste una classe predefinita -> le stringhe sono oggetti in Java
- Le operazioni su stringhe sono realizzate mediante metodi della classe String

# Dichiarazione di una variabile stringa

- Per dichiarare una variabile stringa:
  - `String nome; // stringa con valore nullo`
  - `String nome = "Massimo";`
- Il secondo caso equivale alla creazione e inizializzazione dell'oggetto nome:
  - `String nome = new String("Massimo");`

# Alcuni metodi della classe String

- Il metodo `length()` restituisce la lunghezza della stringa:

```
String nome = "Massimo";  
int i = nome.length() -> vale 7  
"Massimo".length() -> vale 7  
"".length() -> vale 0
```

- Gli elementi della stringa vanno dalla posizione 0 a `length()-1`

# Alcuni metodi della classe String

- Metodo `charAt(int p)` restituisce il carattere che occupa la posizione `p`.

```
nome.charAt(0) -> carattere 'M'  
nome.charAt(3) -> carattere 's'
```

- Il metodo `concat()`, a partire da due stringhe, restituisce una stringa il cui valore è dato dalla sequenza di caratteri della prima stringa seguito dalla sequenza di caratteri della seconda stringa □ `String concat(String x)`

```
String s1, s2, s3;  
s1 = "Massimo";  
s2 = "Ficco";  
s3= s1.concat(s2) -> la stringa s3 varrà  
"MassimoFicco";
```



# Alcuni metodi della classe String

Il metodo `String substring(int inizio)` restituisce un nuovo oggetto `String` che consiste una sottostringa della stringa su cui è invocato

```
String s1, s2;  
s1 = "pianoforte";
```

```
s2= s1.substring(5) -> restituisce la  
stringa "forte"
```

```
s2= s1.substring(5,8) -> restituisce la  
stringa "for"
```

# Alcuni metodi della classe String

Il metodo `int indexOf(char c)` verifica se la stringa contiene il carattere `c` e restituisce la sua posizione:

```
s1 = "Massimo";  
s1.indexOf('a') -> valore 1  
s1.indexOf('p') -> valore -1  
s1.indexOf('ssi') -> valore 2
```

Il metodo boolean `equals(string s)` permette di comparare due stringhe (non è corretto usare `==`).

```
s1 = "Massimo";  
s2 = "Massimo";  
System.out.println(s1.equals(s2)); ->  
Stampa true
```

# Esempio classe auto

```
class Auto{
    public int    cilindri=4;
    public int    speed=0;
    public String targa;

    public int getSpeed(){return speed;};
    public void setSpeed(int s){speed= s;};
    public int getCilindri(){return cilindri};

    // Tre alternative ???
    public void setTarga(){targa = "XF345PF"; }
    public void setTarga(String s){ targa = s;}
    public void setTarga(String s){ targa = new
String(s); }
}
```

# 1° Esempio- Scambio dei parametri (per valore)

```
public class P-Valore{
    static void  cambiaString(String s)
    {
        s=s.concat(" casa");
        System.out.println("metodo: "+s); □ metodo: ciao casa
    }

    public static void  main(String args[])
    {
        String s="ciao"; □ stringa costante
        System.out.println("main1: "+s); □ main1: ciao
        cambiaString(s); □ passaggio per valore
        System.out.println("main2: "+s); □ main2: ciao
    }
}
```

# 1° Esempio- Scambio dei parametri (per riferimento)

```
public class P-Valore{
    static void  cambiaString(String s)
    {
        s=s.concat(" casa");
        System.out.println("metodo: "+s); □ metodo: ciao casa
    }

    public static void  main(String args[])
    {
        String s= new String("ciao"); □ Riferimento
        System.out.println("main1: "+s); □ main1: ciao
        cambiaString(s); □ passaggio per riferimento
        System.out.println("main2: "+s); □ main2: ciao casa
    }
}
```

## 2° Esempio- Scambio dei parametri (per riferimento)\*

```
class Auto{  
    public int    cilindri=4;  
    public int    speed=0;  
    public String targa;  
  
    public int getSpeed(){return speed;};  
    public void setSpeed(int s){speed= s;};  
    public int getCilindri(){return cilindri;};  
        public void setTarga(String s){targa = s;}  
}
```

## 2° Esempio- Scambio dei parametri (per riferimento)\*

```
public class Prova{
    static void accelera(Auto b)
    {
        b.setSpeed(b.speed+1);
        System.out.println("metodo: "+b.speed); □ metodo: 1
    }

    public static void main(String args[])
    {
        Auto a =new Auto();
        System.out.println("main1: "+a.speed); □ main1: 0
        accelera(a); □ passaggio per riferimento
        System.out.println("main2: "+a.speed); □ main2: 1
    }
}
```

# 3° Esempio preliminare

```
class Contatore {  
    int cont=0;  
    public void incr() { cont++; }  
  
    public static void main(String args[]) {  
        Contatore a=new Contatore();  
        Contatore b=new Contatore();  
        a.incr();  
        System.out.println(a.cont);           output ?  
        System.out.println(b.cont);           output ?  
        b=a; // copia i riferimenti  
        System.out.println(a.cont);           output ?  
        System.out.println(b.cont);           output ?  
        b.incr();  
        System.out.println(a.cont);           output ?  
        System.out.println(b.cont);           output ?  
    }  
}
```



# Class String

La classe String:

```
String str = new String();  
String str2 = new String("stringa_di_esempio");  
String str3 = "abc"; // Stringa costante. Non può essere più  
cambiata  
System.out.println(str2);  
System.out.println(str3);
```

I metodi:

```
String str = str2.substring(1, 2);  
System.out.println(str);  
int i = str.length();  
str.replace(char oldChar, char newChar); // Sostituisce il  
carattere oldChar con NewChar  
int i = str.hashCode();  
str.concat(String anotherString);  
str.compareTo(String anotherString);
```

# Passare argomenti all'applicazione da linea di comando

Per eseguire un'applicazione java e passargli un argomento, basta far seguire al nome dell'applicazione il valore desiderato: verrà interpretato come una stringa e memorizzato nell'array di stringhe `args[]` definito in `main`

Esempio: **java Shirt verde**

// Determina se è passata una stringa sulla riga di comando

```
public class Shirt {  
    public static void main(String args[]) {  
        if (args.length < 1) {  
            System.out.println("Nessun argomento");  
        }  
        else {  
            System.out.println("Stringa digitata: " + args[0]);  
        }  
    }  
}
```