Object Oriented Programming

Prof. Ing. Loris Penserini

elpense@gmail.com

Modellare il Mondo Reale

Ciascun paradigma di programmazione fornisce allo sviluppatore un differente approccio concettuale per implementare il pensiero computazionale, cioè la definizione della strategia algoritmica utile per affrontare e risolvere un problema del mondo reale.

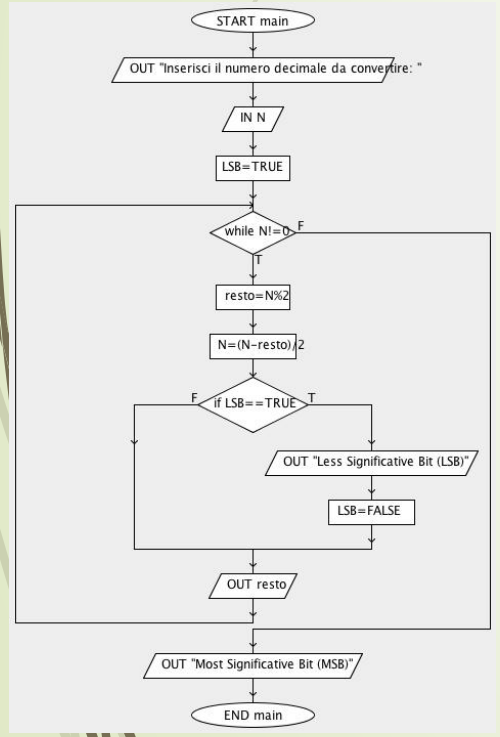
La OOP è attualmente il paradigma di sviluppo del SW più utilizzato, per questo molti linguaggi del Web che in passato nascevano non ad oggetti ora lo sono diventati, come il C → C++, il PHP dopo la ver. 5, mentre altri sono nati direttamente OO come JAVA (JSP e Servlet).

In ogni caso, per creare pagine Web dinamiche, il protocollo standard rimane sempre il **Common Gateway Interface (CGI)**, cioè, indipendentemente dal linguaggio di programmazione usato, si lascia aperta la possibilità di eseguire codice remoto (su un server) invocandolo da un client Web (browser).

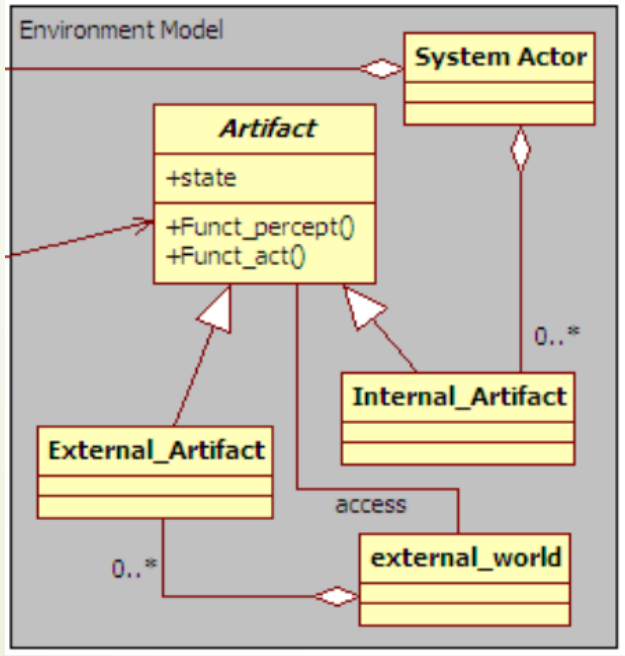
Paradigmi di programmazione…

Alcuni principali paradigmi di programmazione e livelli di astrazione del pensiero computazionale.

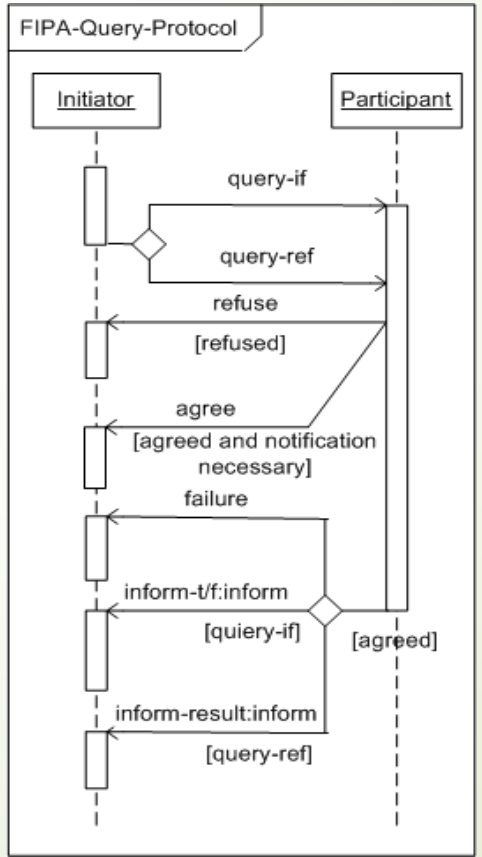
Flow-Chart (Structured Prog.)



UML (Object Oriented Prog.)



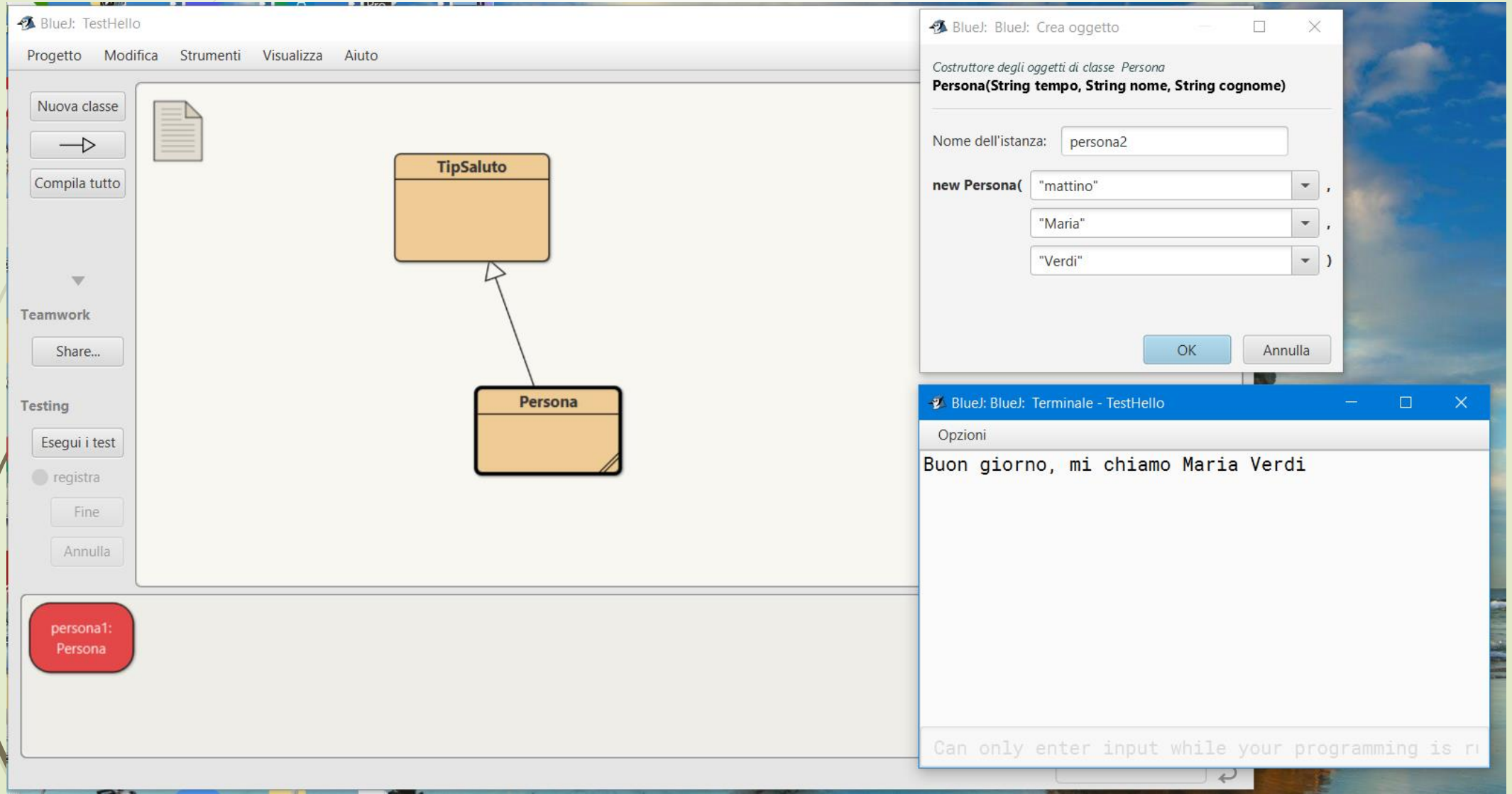
Agent-UML (Agent Oriented Prog.)



Scratch/Blockly (Block based Prog.)

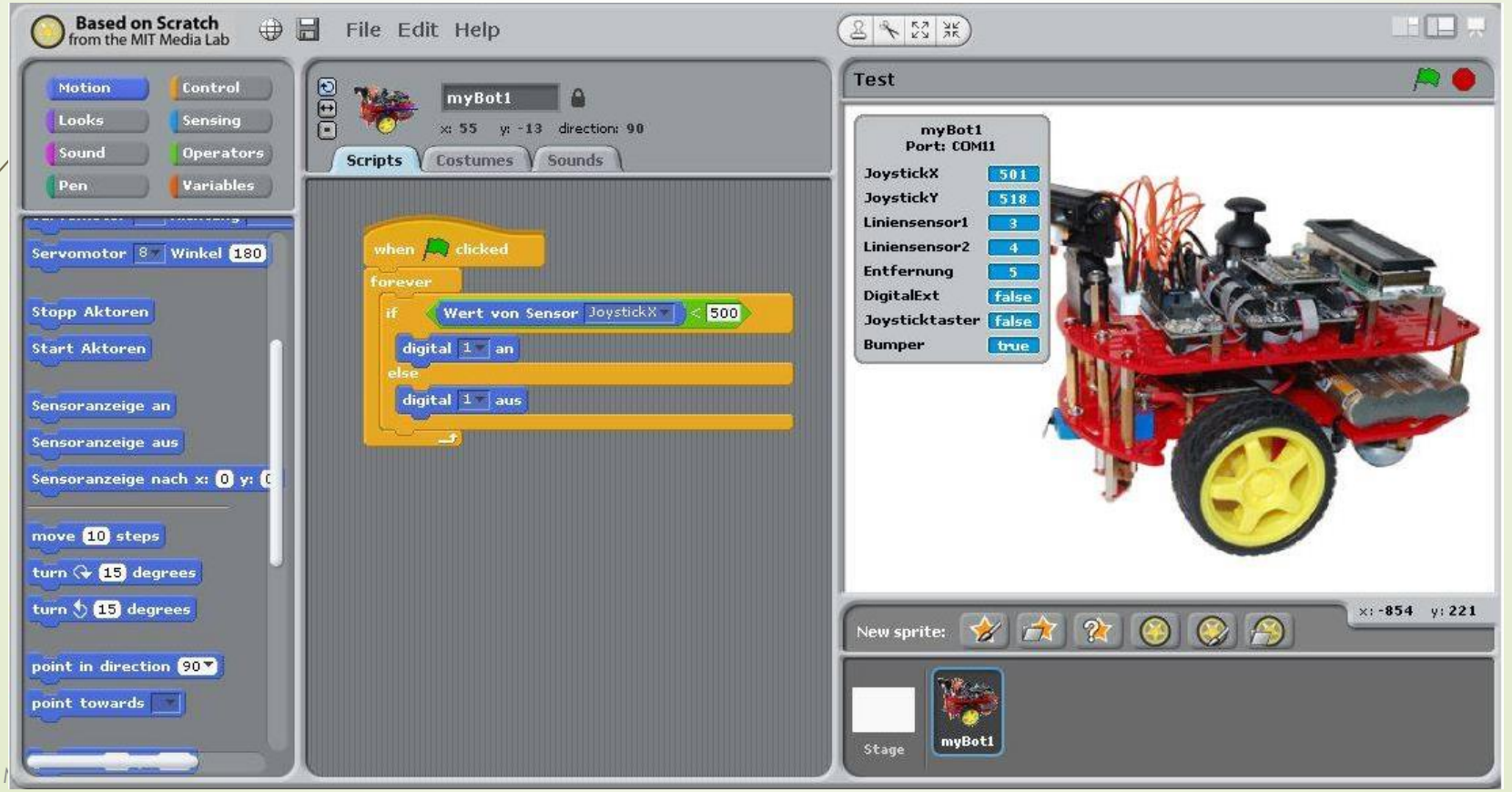
Ambienti visuali per OOP…

BlueJ è uno dei primi ambienti visuali per OOP…



Nella Robotica

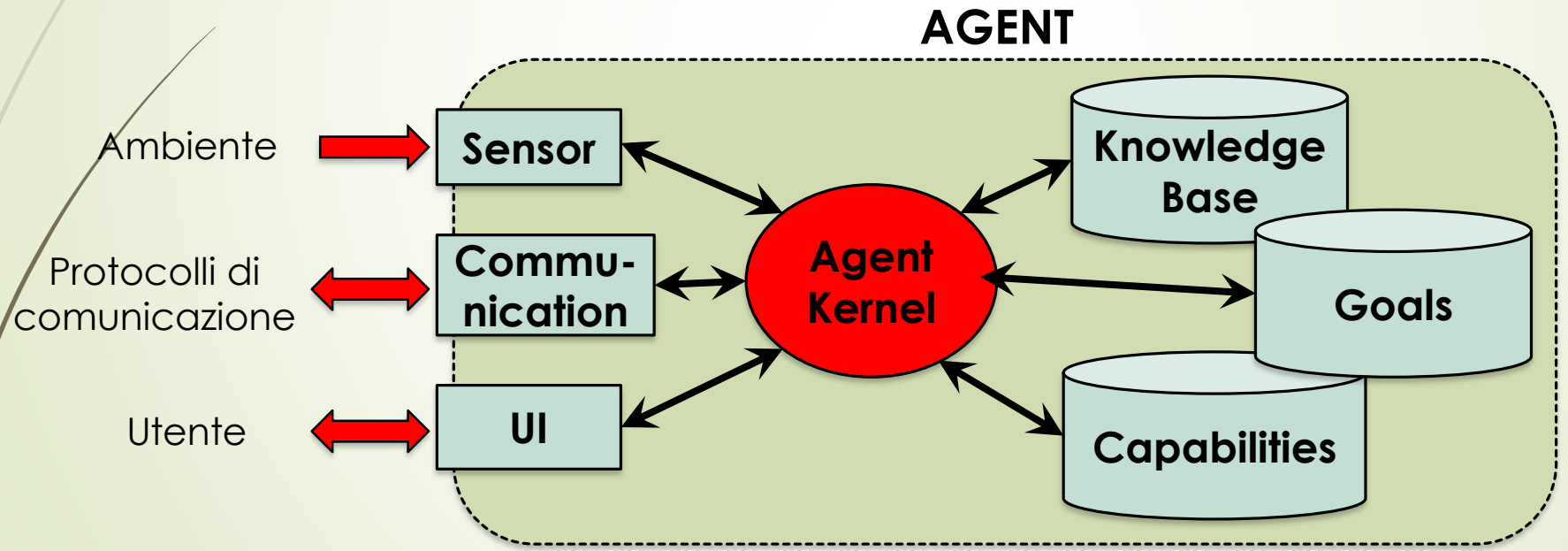
Semplificare l'integrazione di sensori e microcontrollori di un robot attraverso l'utilizzo di «moduli» preconfezionati che si possono incastrare come puzzle per realizzare algoritmi efficienti.



In IA: programmare sistemi autonomi

Esempio di uno «smart agent» con architettura BDI (*Beliefs – Desires – Intentions*).

Nel 1996 nasce in Svizzera la Foundation for Intelligent Physical Agents (FIPA) che nel 2005 entrò a far parte degli standard della IEEE Computer Society.



Perché OOP per lo sviluppo di SW

Ecco alcuni vantaggi di pensare ad un algoritmo in termini di Oggetti:

- Astrarre dalla complessità del codice per concentrarsi maggiormente sulle similitudini tra gli oggetti software e gli oggetti del mondo reale che si vogliono modellare.
- Pensare al comportamento (behavior) di un oggetto software come al suo analogo reale: causalità e relativi effetti.
- Notevole aumento della possibilità di riuso del codice, con conseguente aumento della produttività di qualità:
  - Maggiore stabilità delle applicazioni;
  - Facilità nell'aggiornare mantenere il codice

La «classe»

Nella OOP, la **classe** è il modulo autocontenuto che definisce il progetto (parziale o intero) dell'algoritmo che si vuole realizzare. E' caratterizzata da:

- Proprietà o variabili (in PHP la tipizzazione del dato non è obbligatoria)
- Funzioni o metodi, che contengono la logica dell'algoritmo
- Il nome della classe deve coincidere con quella del file
- I blocchi delimitati dalle parentesi graffe determinano lo scope o campo d'azione delle proprietà
- Lo stato di una classe dipende dai valori di inizializzazione delle sue proprietà nel momento in cui viene caricata in memoria

La classe assomiglia al progetto della casa, ma non è la casa!

Esempio concettuale di «classe» in OOP

//classe
Persona
- nome
- cognome
- età
- Interessi
- pagina di saluto

Buongiorno sono «nome» + «cognome»

specializzare

generalizzare

//sottoclassi
Studente
- nome
- cognome
- età
- interessi
- indirizzo_studi
- pagina di saluto_stu

Buongiorno sono «nome» + «cognome», e frequento «indirizzo_studi»

Insegnante
- nome
- cognome
- età
- interessi
- materia
- pagina di saluto_ins

Buongiorno sono «nome» + «cognome» e insegno «materia»

Cosa è un «oggetto»?

Nella OOP il concetto di «oggetto» è :

- Un processo in esecuzione in memoria
- Una applicazione che fa uso di funzioni/metodi appartenenti a classi diverse (librerie diverse)
- Una applicazione alla quale si può dare uno stato iniziale che ne determina il comportamento (behavior) iniziale.
- Una applicazione che interagisce con il mondo esterno determinando il comportamento che più si adatta per quello scenario

Differenza tra «classe» e «oggetto»

DESIGN time

//classe: è un file

Persona

- nome
- cognome
- età
- Interessi
- pagina di benvenuto

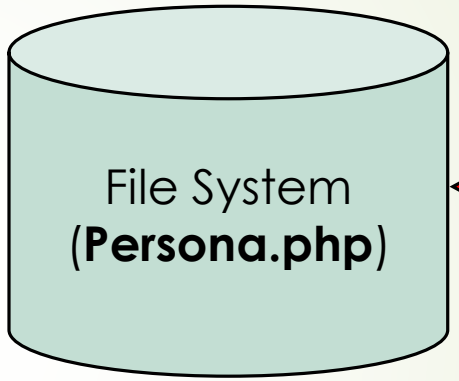
//oggetto: è un processo

Persona1

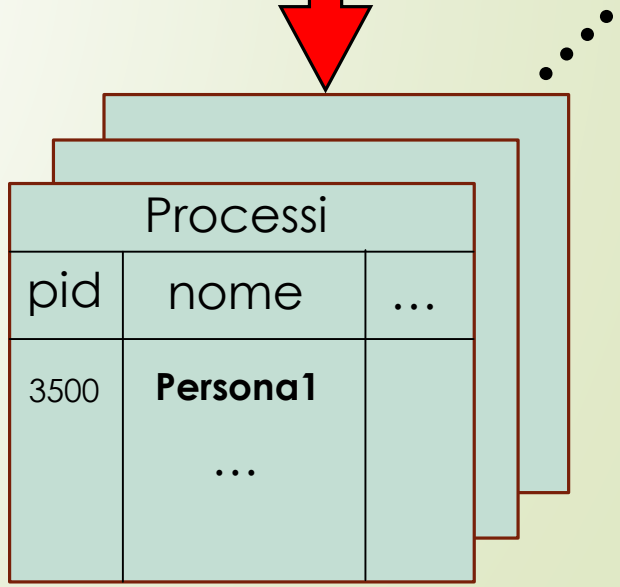
- nome → Mario
- Cognome → Rossi
- Età → 50
- Interessi → robotica
- pagina di benvenuto → Ciao Mario Rossi

RUN time

Memoria di massa



RAM



Proprietà fondamentali in OOP

Tutti i linguaggi OOP forniscono sempre queste tre proprietà:

- **Ereditarietà**
- **Incapsulamento**
- **Polimorfismo**

EREDITARIETA'

Ereditarietà

In PHP (come in Java) **non** è prevista l'ereditarietà multipla come invece è possibile nel C++, per cui in PHP una classe può al più ereditare da una sola altra classe.

In PHP, dalla versione 5.4, sono state aggiunte delle funzioni per ovviare a questa limitazione (i trait) che vedremo più avanti**.**

Tuttavia, per bravi programmatori OOP, l'ereditarietà singola non è considerata una limitazione ma un vantaggio per progettare SW efficiente e modulare.

La classe Persona

```php
class Persona {
    //Proprietà
    public $nome = "";
    public $cognome = "";
    public $eta = "";
    public $interessi = "";
    public $saluto = "";

    //costruttore
    public function __construct($nome,$cognome,$eta,$interessi) {
        //inizializzazione
        $this->nome = $nome;
        $this->cognome = $cognome;
        $this->eta = $eta;
        $this->interessi = $interessi;
        $this->saluto = "Buongiorno sono ".$nome." ".$cognome;
    }

    //metodo che restituisce la pagina di saluto
    public function getPagBenvenuto() {
        $this->saluto = "Buongiorno sono ".$this->nome." ". $this->cognome;
        return $this->saluto;
    }
}
```

La classe Studente

L'operatore «extends»

```php
class Studente extends Persona {
    public $ind_studio = "";

    //metodo per inserire info specifiche per lo studente
    public function setIndStudio($ind_studio) {
        $this->ind_studio = $ind_studio;
    }

    public function getPagBenvenutoStud() {
        $saluto_persona = $this->getPagBenvenuto();
        return $saluto_persona.", e frequento ".$this->ind_studio;
    }
}
```

Le Istanze di classi sono Oggetti

```php
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        include 'Persona.php';
        //require_once 'Persona.php';
        include 'Studente.php';

        $nome = "Loris";
        $cognome = "Penserini";
        $eta = "50";
        $interessi = "DRONI";

        //CREO L'OGGETTO "Personal"
        $Personal = new Persona($nome, $cognome, $eta, $interessi);
        $Studente1 = new Studente($nome, $cognome, $eta, $interessi);
        $Studente1->setIndStudio("Sistemi Informativi Aziendali");

        echo "SALUTO DI PERSONA_1: <br>".$Personal->getPagBenvenuto();
        echo "<br><br>SALUTO DI STUDENTE_1: <br>".$Studente1->getPagBenvenutoStud();

        ?>
    </body>
</html>
```

Costruttori di classe

Ogni classe dovrebbe avere il metodo costruttore, poiché definisce lo stato iniziale dell'oggetto associato alla classe. Cioè il metodo costruttore crea un punto di partenza nell'esecuzione del codice dell'oggetto.

Allora nella classe Studente da quale blocco di codice si inizia?

Project work

Riutilizzate il codice del progetto appena presentato. E con modifiche minimali, aggiungere la classe «Dirigente» (utilizzando come guida la classe Studente), poi dalla pagina index.php lanciare un'istanza e accodare a video il relativo saluto:

output

```
SALUTO DI PERSONA_1:
Buongiorno sono Loris Penserini

SALUTO DI STUDENTE_1:
Buongiorno sono Mario Rossi, e frequento Sistemi Informativi Aziendali

SALUTO DI DIRIGENTE_1:
Buongiorno sono Giovanna Rossini, e dirigo la scuola IIS POLO3 FANO
```