I LINGUAGGI DI PROGRAMMAZIONE AD OGGETI

PRINCIPALI PARADIGMI DEI LINGUAGGI DI PROG.

Dichiarativi:

- Funzionale: usato in applicazioni matematiche e scientifiche, spesso usato in progetti di intelligenza artificiale (LISP)

- Logica: usato in Intelligenza Artificiale nell'apprendimento automatico (Machine Learning) si occupa della realizzazione di sistemi e algoritmi che si basano su osservazioni come dati per la sintesi di nuova conoscenza (PROLOG)

Anche CSS e SQL sono esempi di linguaggi dichiarativi!

PRINCIPALI PARADIGMI DEI LINGUAGGI DI PROG.

Imperativo:

- Procedurale: consiste nel creare dei blocchi di codice sorgente, identificati da un nome e racchiusi da dei delimitatori, detti anche sottoprogrammi (in inglese subroutine) procedure o funzioni.

Altri:

Concorrente, Orientata agli eventi, A vincoli, ...

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

E' un paradigma di programmazione che permette di definire oggetti software in grado di interagire gli uni con gli altri attraverso lo scambio di messaggi. È particolarmente adatta nei contesti in cui si possono definire delle relazioni di interdipendenza tra i concetti da modellare (contenimento, uso, specializzazione).

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

La programmazione ad oggetti prevede di raggruppare in una zona circoscritta del codice sorgente (chiamata classe) la dichiarazione delle strutture dati e delle procedure che operano su di esse. Le classi costituiscono dei modelli astratti, che a tempo di esecuzione vengono invocate per instanziare o creare oggetti software relativi alla classe invocata. Questi ultimi sono dotati di attributi (dati) e metodi (procedure) secondo quanto definito/dichiarato dalle rispettive classi.

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

Un linguaggio di programmazione è definito ad oggetti quando permette di implementare tre meccanismi:
- incapsulamento: separare interfaccia da implementazione in modo che i client usino solo la prima senza conoscere l'impl.
- ereditarietà: definire classi a partire da altre già definite (tramite l'overriding creando sottotipi)
- polimorfismo: lo stesso codice può essere utilizzato con istanze di classi diverse, aventi una superclasse comune

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

Classi:

Le classi definiscono dei tipi di dato e permettono la creazione degli oggetti secondo le caratteristiche definite nella classe stessa. Grazie alle relazioni di ereditarietà, è possibile creare nuove classi a partire da quelle esistenti, estendendole con caratteristiche aggiuntive.

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

La classe è composta da:
- attributi: cioè delle variabili e/o costanti che definiscono le caratteristiche o proprietà degli oggetti instanziabili invocando la classe; i valori inizializzati degli attributi sono ottenuti attraverso il cosiddetto costruttore;
- metodi: procedure che operano sugli attributi

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

Oggetti:

Un oggetto è una istanza di una classe. Esso è dotato di tutti gli attributi e i metodi definiti dalla classe, ed agisce come un fornitore di "messaggi" (i metodi) che il codice eseguibile del programma (procedure o altri oggetti) può attivare su richiesta.

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

Inviare un messaggio ad un oggetto si dice invocare un metodo su quell'oggetto. Il metodo riceve come parametro (spesso implicito) l'oggetto su cui è stato invocato, che può essere referenziato tramite una parola-chiave o una sintassi apposita, anche se è passato come parametro implicito; per esempio, in C++, in Java, e in C# si usa la parola-chiave this ($this in PHP).

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

Offre vantaggi di:

- modularità: le classi sono i moduli del sistema software

- coesione dei moduli: una classe è un componente software ben coeso in quanto rappresentazione di una unica entità

- disaccoppiamento dei moduli: gli oggetti hanno un alto grado di disaccoppiamento in quanto i metodi operano sulla struttura dati interna ad un oggetto; il sistema complessivo viene costruito componendo operazioni sugli oggetti

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

- information hiding: sia le strutture dati che gli algoritmi possono essere nascosti alla visibilità dall'esterno di un ogget.
- riuso: l'ereditarietà consente di riusare la definizione di una classe nel definire nuove (sotto)classi; inoltre è possibile costruire librerie raggruppate per tipologia di applicazioni
- estensibilità: il polimorfismo agevola l'aggiunta di nuove funzionalità, minimizzando le modifiche necessarie al sistema esistente quando si vuole estenderlo.

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

```
class Employee {
    public static $legs = 2;
    public static $basic_salary = 1000;
    private $age;
    public function getAge() { echo $age; }
    public function setAge($age) { $this- >age = $age; }
    public function getSalary()
}
```

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

```
class Manager extends Employee {
    public function getSalary() {
        return parent::$basic_salary * 10;
    }
}
class SoftwareDeveloper extends Employee {
    public function getSalary() { ... }
}
```

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

```
Employee $employee = new Manager();
$employee- >setAge(40);
echo 'Age: ' . $employee- >getAge();
echo 'Salary: ' . $employee- >getSalary();
$softwareDeveloper = new SoftwareDeveloper();
$employee = $softwareDeveloper;
echo 'Software Developer Salary: ' . $employee- >getSalary();
```

GRAZIE

per l'attenzione!