OOP

Secondo il paradigma a oggetti, un sistema software è un insieme di oggetti *cooperanti*, caratteriazzati da uno *stato* (attributi) e delle *funzionalità* (metodi), e che cooperano scambiandosi *messaggi* (invio: chiamata di metodo, risposta: valore restituito).

Incapsulamento: idealmente lo stato di un oggetto non è accessibile dagli altri oggetti.

Approccio object-based

Gli oggetti sono strutture simili a record con campi che possono essere associati a funzioni, le quali hanno a disposizione i campi dell'oggetto stesso tramite il riferimento `this`.

Permette una maggiore flessibilità: si possono creare varianti di un oggetto senza dover definire una classe per ciascuno. Questo al costo di maggiori difficoltà nel prevedere con precisione il tipo di un oggetto, soprattutto se la sua struttura può cambiare a runtime (ostacola i controlli statici).

Caratterizzato da structural typing e prototype-based inheritance.

Approccio class-based

Introduzione del concetto di *classe*: una classe definisce i membri degli oggetti di un certo tipo; questi vengono creati come *istanze* della classe a cui appartengono.

Richiede maggiore disciplina dell'approccio object-based, ma consente controlli statici aggiuntivi mediante *nominal typing*.

Caratterizzato da nominal typing e class-based inheritance.

Esempi

JavaScript e OCaml supportano sia classi che oggetti indipendenti.