Programmazione ad Oggetti

prof. Francesco Ranzato

Moodle

https://elearning.unipd.it/math/course/view.php?id=531

Prof non ha scritto nulla sull'esame, scrivo ciò che ha detto a voce

ESAME

-scritto (una volta passato il voto rimane valido per tutto l'anno)
-progetto (2/3persone deciderà maggiori informazioni a novembre) consegnabile solo dopo aver passato lo scritto
non si sa la % di peso nel voto

Paradigmi di Programmazione

Un programma è costituito da:

- **Algoritmi**
- **Dati** su cui operano gli algoritmi

Quando l'accento è posto sugli algoritmi si parla di *programmazione procedurale*.
Quando l'accento è posto sui dati (o meglio sui tipi di dato) si parla di *programmazione ad oggetti*.

Programmazione orientata agli oggetti

Da Wikipedia, l'enciclopedia libera.

La **programmazione orientata agli oggetti** (**OOP**, **O**bject **O**riented **P**rogramming) è 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). Un ambito che più di altri riesce a sfruttare i vantaggi della programmazione ad oggetti è quello delle interfacce grafiche.

Tra gli altri vantaggi della programmazione orientata agli oggetti:

- fornisce un supporto naturale alla modellazione software degli oggetti del mondo reale o del modello astratto da riprodurre
- permette una più facile gestione e manutenzione di progetti di grandi dimensioni
- l'organizzazione del codice sotto forma di classi favorisce la modularità e il riuso di codice

Object-oriented programming

From Wikipedia, the free encyclopedia

> *"Object-oriented" redirects here. For other meanings of object-oriented, see Object-orientation.*
> *"Object-oriented programming language" redirects here. For a list of object-oriented programming programming languages.*

Object-oriented programming (**OOP**) is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as *attributes* or *properties*), and code, in the form of procedures (often known as *methods*). A feature of objects is an object's procedures that can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another.[1][2] OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

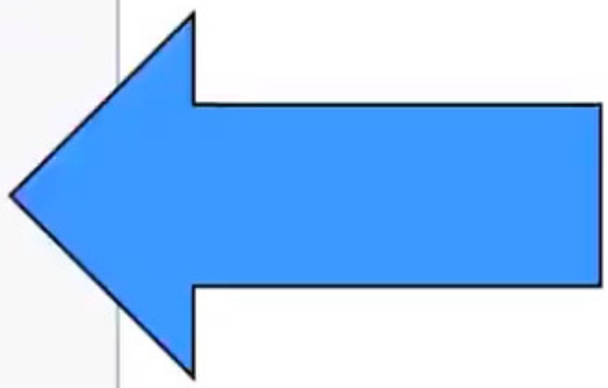
Many of the most widely used programming languages (such as C++, Java, Python, etc.) are multi-paradigm and they support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, MATLAB, and Smalltalk.

Qualità del software

Da Wikipedia, l'enciclopedia libera.

Per **qualità del software** si intende la misura in cui un prodotto software soddisfa un certo numero di aspettative rispetto sia al suo funzionamento sia alla sua struttura interna. Gran parte della ricerca nel campo dell'ingegneria del software è dedicata, direttamente o indirettamente, al tema della qualità.

Indice [nascondi]



Un linguaggio di programmazione è definito *ad oggetti* quando permette di implementare tre meccanismi usando la sintassi nativa del linguaggio:[1]

- incapsulamento
- ereditarietà
- polimorfismo

L'incapsulamento consiste nella separazione della cosiddetta *interfaccia* di una classe dalla corrispondente *implementazione*, in modo che i *client* di un oggetto di quella classe possano utilizzare la prima, ma non la seconda.

L'ereditarietà permette essenzialmente di definire delle classi a partire da altre già definite.

Il polimorfismo permette di scrivere un *client* che può servirsi di oggetti di classi diverse, ma dotati di una stessa interfaccia comune; a tempo di esecuzione, quel client attiverà comportamenti diversi senza conoscere a priori il tipo specifico dell'oggetto che gli viene passato.

Le classi permettono di realizzare il concetto generale di *abstract data type (ADT)*

C++

Da Wikipedia, l'enciclopedia libera.

Il **C++** è un linguaggio di programmazione orientato agli oggetti, con tipizzazione statica. È stato sviluppato (in origine col nome di "C con classi") da Bjarne Stroustrup ai Bell Labs nel 1983 come un miglioramento del linguaggio C. Tra i miglioramenti principali troviamo: l'introduzione del paradigma di programmazione a oggetti, funzioni virtuali, overloading degli operatori, ereditarietà multipla, template e gestione delle eccezioni.

Il C++ fornisce inoltre strumenti che permettono di integrare ed estendere i due paradigmi procedurale e ad oggetti:

- I *template* di funzione
- I *template* di classe

} Programmazione generica

- *Gestione delle eccezioni*

Principali caratteristiche del linguaggio C++

- Compilato
- Tipizzazione forte statica (strongly typed)
- No garbage collector
- Standardizzato (ultimo standard ANSI C++17, C++20 in progress)
- General-purpose e molto diffuso (Adobe SW, Mozilla SW, MySQL, Microsoft SW, Google Chromium, Games...)
- Efficienza
- Operatori e loro overloading
- Librerie

C++20

From Wikipedia, the free encyclopedia

C++20 is the name for the revision of the ISO/IEC standard for the C++ programming language following C++17.[1] The standard became technically finalized[2] by WG21 at the meeting in Prague in February 2020.[3] C++20 is currently undergoing final editorial work, after a draft was approved on 4th September 2020.[4][5]

C++ Language Revisions

C++98 · C++03 · C++11 · C++14 · C++17 · **C++20** · C++23

C++20 adds more new major features than C++14 or C++17.[6] Below is a partial list of changes that have been accepted into or have been discussed for inclusion into C++20.[7]

Programmazione ad Oggetti in

Principles

There were five primary goals in the creation of the Java language

1. It must be "simple, object-oriented, and familiar".
2. It must be "robust and secure".
3. It must be "architecture-neutral and portable".
4. It must execute with "high performance".
5. It must be "interpreted, threaded, and dynamic".