**OOP**

**Úvod:**

* **object-oriented programming**

**(objektově orientované programování)**

* vhodné pro větší a složitější programy
	+ potřeba dopředu hlubokého
	promyšlení celé funkčnosti
* X procedurální programování
(funkce a proměnné u sebe v jednom)
* základ – objekt (třída = **CLASS**)
* třída = šablona pro objekty
	+ metody (funkce), proměnné

**Tvorba:**

* vhodné pro všechny druhy aplikací, ale hlavně pro tvorbu složitější aplikací (delší začátek X snadnější růst)
* hlavně vhodné pro udržované/updatované programy
	+ práce jde lépe rozdělit do více skupin

**Třídy:**

* třída = datový objekt
* = samostatná funkční jednotka
* schopnost komunikovat s dalšími třídami
* obsahuje potřebnou funkcionalitu
a její proměnné
* funkce tříd = metody
 +
proměnné pro danou entitu

**Hlavní vlastnosti OOP:**

* **4 hlavní potřebné vlastnosti:**
	+ encapsulation (zapouzdření)
		- jednotlivé objekty (třídy) nemají možnost zasahovat do fungování jiných tříd (př. brát proměnné)
	+ abstraction (abstrakce)
		- objekty neukazují všechny metody, ale jen ty svému okolí potřebné
		- ulehčení práce

>>> dir(list)
['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_delitem\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_iadd\_\_', '\_\_imul\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_reversed\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_setitem\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

* + inheritance (dědičnost)
		- nové třídy mohou dědit funkcionalitu starších tříd
		- možnost použít vícekrát podobnou funkcionalitu bez potřeby dlouhého přepisování
	+ polymorphism (polymorfismus)
		- instance objektů (stejných tříd), nebývají podobu, podle toho, jakou funkcionalitu používáme

**Jazyky:**

* jako první OOP jazyk označován **Simula** (simulation language) – 1960
* tvorba dalších známých OOP jazyků:
	+ Java
	+ Javascript
	+ Python
	+ C++
	+ PHP
	+ …

**Kritika:**

* přílišné zaměření na „objektovost“ a neklade se důraz na využití výpočetních algoritmů