

# Objektově orientované programování (OOP)

**Objektově orientované programování** je programovací paradigma založené na konceptu „objektů“, které mohou obsahovat data (ve formě polí či atributů) a kód (ve formě procedur neboli metod). OOP pomáhá zpřehlednit vývoj velkých systémů, usnadňuje znovupoužitelnost kódu a jeho údržbu.

## 1. Třída vs. Objekt

Pro pochopení OOP je zásadní rozdíl mezi těmito dvěma pojmy:

- **Třída (Class):** Abstraktní plán nebo šablona (např. „Auto“). Definuje, jaké vlastnosti a schopnosti budou mít všechny instance této třídy.
- **Objekt (Instance):** Konkrétní realizace třídy (např. „moje červená Škoda Octavia“). Má konkrétní hodnoty svých vlastností.

## 2. Čtyři pilíře OOP

Aby byl jazyk považován za skutečně objektový, musí podporovat tyto čtyři principy:

### A. Zapouzdření (Encapsulation)

Skrývání vnitřního stavu objektu před vnějším světem. Přístup k datům je umožněn pouze přes veřejné metody (getter a setter).

- **Výhoda:** Zamezuje nechtěné modifikaci dat a usnadňuje změnu vnitřní implementace bez dopadu na zbytek programu.

### B. Abstrakce (Abstraction)

Zjednodušení komplexní reality tím, že modelujeme pouze ty aspekty, které jsou pro náš program důležité.

- **Příklad:** Pro e-shop s auty nás zajímá barva a cena, ale nepotřebujeme modelovat chemické složení brzdové kapaliny.

### C. Dědičnost (Inheritance)

Umožňuje vytvářet nové třídy na základě již existujících. Potomek přebírá (dědí) vlastnosti a metody rodiče a může přidávat vlastní.

- **Příklad:** Třída Pes a Kočka dědí od třídy Savec.
- **Výhoda:** Drastické snížení duplicity kódu.

## D. Polymorfismus (Mnohotvárnost)

Schopnost různých objektů reagovat na stejné volání metody různým způsobem.

- **Příklad:** Máme metodu `vydej_zvuk()`. U objektu `Pes` zaštěká, u objektu `Kočka` zamňouká. Programátorovi stačí vědět, že volá metodu na „zvířeti“, a objekt se postará o zbytek.

## 3. Další klíčové pojmy

- **Konstruktor:** Speciální metoda, která se zavolá při vytváření nového objektu (nastavuje počáteční hodnoty).
- **Rozhraní (Interface):** Předpis, který říká, co třída musí umět, ale neříká jak. Je to „smlouva“ o chování.
- **Statické členy:** Patří samotné třídě, nikoliv konkrétnímu objektu (např. počítadlo všech vytvořených aut).

## 4. Výhody a nevýhody OOP

| Výhody                                                                         | Nevýhody                                                                         |
|--------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| <b>Modularita:</b> Snazší řešení chyb v izolovaných objektech.                 | <b>Komplexita:</b> Pro velmi malé programy může být OOP zbytečně složitě.        |
| <b>Znovupoužitelnost:</b> Jednou napsanou třídu lze použít v mnoha projektech. | <b>Výkon:</b> Režie spojená s objekty může mírně zpomalit běh (v porovnání s C). |
| <b>Rozšiřitelnost:</b> Snadné přidávání nových funkcí pomocí dědičnosti.       | <b>Paměťová náročnost:</b> Objekty zabírají v paměti více místa než prostá data. |

## 5. Jazyky podporující OOP

Dnes téměř každý moderní jazyk podporuje OOP, i když v různé míře:

- **Čistě objektové:** Smalltalk, Ruby.
- **Hybridní (nejčastější):** [Java](#), [C++](#), [Python](#), [C#](#), [PHP](#).

Související články:

- [Programovací jazyky a jejich syntaxe](#)
- [Datové struktury a třídy](#)
- [Návrhové vzory \(Design Patterns\)](#)

Tagy: *dev oop programming software\_engineering inheritance polymorphism encapsulation*

From:

<http://serviceit.cz/> - **IT ENCYKLOPEDIE**

Permanent link:

<http://serviceit.cz/doku.php?id=it:dev:oop>

Last update: **2026/01/02 13:40**

