

Principy SOLID

SOLID je akronym pro pět pravidel, která zformuloval Robert C. Martin (známý jako „Uncle Bob“). Tyto principy jsou považovány za zlatý standard pro psaní čistého a udržitelného kódu v [objektově orientovaném programování](#).

—

1. S - Single Responsibility Principle (SRP)

Princip jedné odpovědnosti Třída by měla mít pouze jednu odpovědnost a tedy pouze jeden důvod ke změně.

- **Špatně:** Třída `Zaměstnanec``, která obsahuje data o platu, logiku pro výpočet daní a metodu pro uložení do databáze.
- **Správně:** Rozdělení na `Zaměstnanec`` (data), `TaxCalculator`` (výpočty) a `EmployeeRepository`` (databáze).

2. O - Open/Closed Principle (OCP)

Princip otevřenosti/uzavřenosti Softwarové entity (třídy, moduly) by měly být **otevřené pro rozšiřování**, ale **uzavřené pro modifikaci**.

- **Význam:** Měli byste být schopni přidat novou funkci bez toho, abyste zasahovali do stávajícího, již otestovaného kódu. Toho se nejčastěji dosahuje pomocí rozhraní (Interfaces) nebo abstraktních tříd.

3. L - Liskov Substitution Principle (LSP)

Liskovové princip zastupitelnosti Objekty v programu by měly být nahraditelné svými podtřídami (potomky) bez toho, aby se změnila funkčnost nebo korektnost programu.

- **Klasický problém:** Třída `Čtverec`` dědící od `Obdélník``. Pokud metoda pro nastavení šířky obdélníku změnila u čtverce i výšku, porušuje tento princip, protože se čtverec nechová jako pravý obdélník.

4. I - Interface Segregation Principle (ISP)

Princip oddělení rozhraní Klienti by neměli být nuceni záviset na rozhraních, která nepoužívají. Lepší je mít více malých, specifických rozhraní než jedno velké „univerzální“.

- **Příklad:** Namísto jednoho rozhraní `IPracovník`` s metodami `pracuj()`` a `jez()``, je lepší mít `IPracující`` a `IStavující``. Robot totiž nepotřebuje metodu `jez()``.

5. D - Dependency Inversion Principle (DIP)

Princip obrácení závislostí Moduly vyšší úrovně by neměly záviset na modulech nižší úrovně. Oba by měly záviset na abstrakcích. Abstrakce by neměly záviset na detailech, detaily by měly záviset na abstrakcích.

- **V praxi:** Třída by neměla přímo vytvářet instanci konkrétní databáze (např. ``MySQLConnection``), ale měla by vyžadovat obecné rozhraní ``IDatabase``.

Proč SOLID používat?

Přínos	Popis
Udržitelnost	Snadnější hledání chyb a úprava stávajícího kódu.
Testovatelnost	Menší a jasně definované třídy se lépe testují (Unit testing).
Znovupoužitelnost	Moduly jsou méně svázané a lze je snadno přenést do jiných projektů.
Týmová práce	Jasná struktura umožňuje více vývojářům pracovat na různých částech bez konfliktů.

Související články:

- [Základy objektově orientovaného programování](#)
- [Návrhové vzory](#)
- [Refaktoring kódu](#)

Tagy: programming software-engineering solid clean-code oop architecture

From:

<https://www.serviceit.cz/> - IT ENCYKLOPEDIE

Permanent link:

<https://www.serviceit.cz/doku.php?id=it:sw:solid>

Last update: **2026/01/02 18:54**

