SOLID (Principy čistého návrhu)

SOLID je mnemotechnická pomůcka pro pět pravidel, která pomáhají programátorům vyhnout se „hnilobě softwaru“ (situaci, kdy je kód tak složitý, že každá změna způsobí chybu na jiném místě).

---

1. S – Single Responsibility Principle (SRP)

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

- **Špatně:** Třída `Report`, která generuje data, formátuje je do PDF a zároveň je odesílá e-mailem.
- **Správně:** Třída `Report` pro data, `PdfGenerator` pro formát a `EmailSender` pro odeslání.

---

2. O – Open/Closed Principle (OCP)

Princip otevřenosti/uzavřenosti: Softwarové entity by měly být otevřené pro rozšiřování, ale uzavřené pro modifikaci.

- **V praxi:** Pokud chcete přidat novou funkci, neměli byste přepisovat stávající kód, ale přidat kód nový (např. pomocí dědičnosti nebo rozhraní). Tím neriskujete rozbití již fungujících částí.

---

3. L – Liskov Substitution Principle (LSP)

Liskovové princip zastupitelnosti: Objekty v programu by měly být nahraditelné svými potomky, aniž by se změnila správnost programu.

- **Příklad:** Pokud třída `Pstros` dědí od třídy `Ptak`, ale metoda `let()` u pštrosa vyhodí chybu (protože pštros nelétá), je tento princip porušen. Potomek nesmí omezovat schopnosti rodiče.

---

4. I – Interface Segregation Principle (ISP)

Princip oddělení rozhraní: Klienti by neměli být nuceni záviset na metodách, které nepoužívají.

- **V praxi:** Místo jednoho obřího rozhraní (např. `IMultifunkce`) je lepší vytvořit více malých

(`ITiskarna`, `ISkener`, `IFax`). Program, který jen tiskne, by neměl být nucen implementovat metodu pro faxování.

---

5. D – Dependency Inversion Principle (DIP)

Princip obrácení závislostí: Moduly na vyšší úrovni by neměly záviset na modulech na nižší úrovni. Oba by měly záviset na abstrakcích (rozhraních).

- **Jednoduše:** Nekódovat „na tvrdo“ konkrétní třídy, ale používat rozhraní.
- **Příklad:** Třída `Eshop` by neměla přímo vytvářet objekt `PayPal`, ale měla by vyžadovat jakoukoliv `IPlatebniMetoda`. To umožní kdykoliv přidat jinou platební metodu bez zásahu do kódu e-shopu.

Proč SOLID používat?

Výhoda	Popis
Udržovatelnost	Snadnější hledání a oprava chyb v izolovaných třídách.
Testovatelnost	Malé třídy s jasnou odpovědností se mnohem lépe testují.
Znovupoužitelnost	Kód lze snadněji přenášet do jiných projektů.
Týmová práce	Jasně definované hranice mezi moduly umožňují souběžný vývoj.

Související pojmy: OOP, Class, Dědičnost, Polymorfismus, Interface, Abstraktní třída, DRY princip.