# Polymorfismus (Mnohotvárnost)

**Polymorfismus** umožňuje programátorovi pracovat s různými objekty, jako by to byl jeden obecný typ, přičemž každý z těchto objektů si zachovává své specifické chování. Je to mechanismus, který dělá kód flexibilním, rozšiřitelným a snadno udržitelným.

---

# Typy polymorfismu

V programování rozlišujeme dva hlavní druhy polymorfismu podle toho, kdy se rozhoduje o tom, která metoda se spustí:

## 1. Statický polymorfismus (Ad-hoc)

Dochází k němu v době překladu (**Compile-time**). Typickým příkladem je **přetěžování metod (Overloading)**. V jedné třídě existuje více metod se stejným názvem, které se liší pouze počtem nebo typem parametrů.

- **Příklad:** Metoda `vypocti(int a)` vs. `vypocti(double a)`.

## 2. Dynamický polymorfismus

Dochází k němu až za běhu programu (**Runtime**). Je úzce spjat s dědičností a využívá **překrývání metod (Overriding)**. Program zavolá metodu na obecném objektu, ale díky polymorfismu se spustí verze definovaná v konkrétním potomkovi.

---

# Praktický příklad: Grafické tvary

Představte si, že vyvíjíte grafický editor. Máte obecnou třídu `Tvar` a z ní zděděné třídy `Kruh`, `Ctverec` a `Trojuhelnik`.

- Všechny tyto třídy mají metodu `vykresli()`.
- Vy máte seznam (pole) objektů typu `Tvar`.
- Program projde seznam a u každého prvku zavolá `tvar.vykresli()`.

**Výsledek:** Programátor nemusí řešit, co je zrovna na řadě. Kruh se vykreslí jako kruh a čtverec jako čtverec, i když se k nim v kódu přistupuje jako k obecnému „tvaru“.

# Proč je polymorfismus důležitý?

- **Flexibilita:** Pokud do programu přidáte nový druh tvaru (např. `Hvezda`), nemusíte měnit kód, který tvary vykresluje. Stačí, aby `Hvezda` implementovala metodu `vykresli()`.
- **Zjednodušení kódu:** Odstraňuje potřebu složitých konstrukcí typu `if (objekt je Kruh) kresliKruh() else if (objekt je Ctverec)…`.
- **Rozšiřitelnost:** Umožňuje vytvářet systémy zásuvných modulů (pluginů), kde hlavní program volá metody, aniž by předem znal jejich konkrétní implementaci.

# Polymorfismus a Rozhraní (Interface)

Kromě dědičnosti lze polymorfismus realizovat pomocí **rozhraní**. Rozhraní říká, „co“ musí objekt umět, ale neříká „jak“. * **Příklad:** Rozhraní `IPlatebniMetoda` má metodu `zaplat()`. Tuto metodu implementuje `KreditniKarta` (odesláním čísla karty) i `PayPal` (přihlášením k účtu). Programu je jedno, čím platíte, prostě jen zavolá `zaplat()`.

*Související pojmy: OOP, Dědičnost, Class, Instance, Overriding, Overloading, Interface, Abstraktní třída.*