Konstruktor (Inicializace objektu)

Konstruktor je mechanismus, který zajišťuje, že objekt „vstoupí do života“ v platném a předvídatelném stavu. Bez konstruktoru by atributy objektu mohly obsahovat náhodná data nebo prázdné hodnoty (null), což by vedlo k pádům programu.

Klíčové vlastnosti konstruktoru

- **Stejné jméno:** Ve většině jazyků (Java, C++, C#) se konstruktor jmenuje úplně stejně jako samotná třída. (Výjimkou je např. Python, který používá `init`, nebo PHP s `__construct`).
- **Žádná návratová hodnota:** Konstruktor nevrací žádný typ (ani `void`), protože jeho účelem není vracet výsledek, ale vytvořit objekt.
- **Automatické volání:** Spouští se klíčovým slovem **new**.

Typy konstruktorů

1. Implicitní (Výchozí) konstruktor

Pokud programátor ve třídě žádný konstruktor nenapíše, kompilátor jej vytvoří sám. Tento konstruktor nemá žádné parametry a nastaví atributy na základní hodnoty (nuly, prázdné řetězce).

2. Parametrický konstruktor

Umožňuje předat data přímo při vytváření objektu. Tím zajistíte, že objekt bude mít hned od začátku konkrétní vlastnosti. * *Příklad:* `new Auto(„Červená“, „Škoda“)`

3. Kopírovací konstruktor

Slouží k vytvoření nového objektu jako přesné kopie jiného, již existujícího objektu téže třídy.

Přetěžování konstruktorů (Overloading)

V jedné třídě může existovat více konstruktorů najednou, pokud se liší svými parametry. To umožňuje vytvářet objekty různými způsoby.

Konstruktor	Příklad volání	Výsledek
Bez parametrů	`new Uzivatel()`	Vytvoří anonymního hosta.
S jedním parametrem	`new Uzivatel(„Petr“)`	Vytvoří uživatele se jménem.
Se všemi parametry	`new Uzivatel(„Petr“, 25)`	Vytvoří uživatele s plným profilem.

Příklad v kódu (Java)

```
class Kniha {
    String nazev;
    int rokVydani;

    // Konstruktor
    Kniha(String nazev, int rokVydani) {
        this.nazev = nazev;        // 'this' odkazuje na atribut třídy
        this.rokVydani = rokVydani;
    }
}

// Použití konstruktoru
Kniha mojeKniha = new Kniha("Zaklínač", 1993);
```

Destruktor – Protiklad konstruktoru

Zatímco konstruktor objekt staví, **destruktor** jej bourá. Volá se v okamžiku, kdy objekt zaniká. V jazycích s automatickou správou paměti (Java, Python, C#) se destruktory používají zřídka, protože úklid řeší Garbage Collector. V jazyce C++ je však destruktor klíčový pro ruční uvolnění paměti, aby nedocházelo k jejímu úniku (Memory Leak).

Související pojmy: Class, Instance, OOP, Atribut, Metoda, Garbage Collector, Memory Leak.