Dědičnost (Inheritance)

Dědičnost definuje vztah typu **„je“ (is-a)**. Pokud třída `Pes` dědí od třídy `Zvíře`, pak platí, že „Pes **je** Zvíře“. Tento mechanismus umožňuje programátorům psát kód jen jednou a následně jej rozšiřovat, místo aby jej znovu kopírovali.

Základní terminologie

- **Rodičovská třída (Superclass / Base class):** Obecná třída, ze které se dědí (např. `Vozidlo`).
- **Potomkovská třída (Subclass / Derived class):** Specializovaná třída, která přejímá prvky od rodiče a doplňuje je o vlastní (např. `Auto`, `Motorka`).

Hlavní výhody dědičnosti

- **Znovupoužitelnost kódu:** Obecné funkce (např. `pohniSe()`) napíšete jednou do rodičovské třídy a všechny tisíce potomků je budou umět automaticky také.
- **Udržovatelnost:** Pokud najdete chybu v obecné logice, opravíte ji na jednom místě (u rodiče) a oprava se projeví u všech potomků.
- **Hierarchická struktura:** Pomáhá logicky organizovat složité systémy (např. v grafických rozhraních, kde `Tlacitko` i `TextovePole` dědí od obecného `Prvku`).

Jak dědičnost funguje v praxi

Představme si hierarchii zvířat. Třída `Zvíře` definuje atribut `jmeno` a metodu `spi()`.

Třída	Má k dispozici	Vlastní unikátní prvek
Zvíře (Rodič)	jmeno, spi()	(není)
Pes (Potomek)	jmeno, spi()	metodu stekej()
Pták (Potomek)	jmeno, spi()	metodu let()

Přepsání metody (Overriding)

Potomek může metodu, kterou zdědil, upravit tak, aby mu lépe vyhovovala. Tomu se říká **Overriding**. Například obě zvířata dědí metodu `vydejZvuk()`, ale Pes ji přepíše na „Haf“, zatímco Kočka na „Mňau“.

Omezení a pravidla

- **Jednoduchá vs. vícenásobná dědičnost:** Většina moderních jazyků (Java, C#, Python) preferuje jednoduchou dědičnost – třída může mít pouze **jednoho** přímého rodiče. Vyhýbají se tak „problému diamantu“ (nejasnost, od koho dědit, pokud mají dva rodiče metodu se stejným názvem).
- **Klíčové slovo 'super':** Potomek se může odkazovat na originální kód rodiče pomocí klíčového slova `super` (nebo `base`). Často se používá v konstruktorech.
- **Finální třídy:** Některé třídy lze označit jako „final“ (v Javě) nebo „sealed“ (v C#), což znamená, že z nich již nikdo dál dědit nesmí.

Příklad (Java)

```
class Vozidlo {
    void trub() {
        System.out.println("Túúúút!");
    }
}

class Auto extends Vozidlo {
    int pocetDveri = 5;
}

// Použití
Auto mojeAuto = new Auto();
mojeAuto.trub(); // Zděděno od Vozidla
System.out.println(mojeAuto.pocetDveri); // Vlastní atribut Auta
```

Související pojmy: OOP, Class, Instance, Polymorfismus, Konstruktor, Abstrakce, Přetěžování.