Dědičnost (abstraktní třídy, rozhraní, kolekce objektů příbuzných tříd, polymorfismus)

Dědičnost

Dědičnost je vztah mezi objekty v OOP. Kde jsou objekty definované třídami, mohou třídy zdědit vlastnosti a metody od předem existujících tříd. Ty se obvykle nazývají rodičovské třídy. Výsledné třídy jsou pak nazývány jako třídy odvozené, podtřídy nebo potomci třídy. Koncept dědičnosti byl poprvé zaveden pro jazyk Simula v roce 1968.

Abstraktní třídy

Abstraktní třída je třída, která nemá definované některé metody. Předpokládá se, že z ní nějaká jiná třída bude dědit a tyto metody si doplní podle své potřeby. Obvykle již disponuje nějakými instantními proměnnými. Abstraktní třída také může obsahovat konkrétní metody. Všechny metody tedy nemusejí nutně mít definován algoritmus své činnosti. Potomek abstraktní třídy nemůže však dědit od nějaké další třídy (může dědit pouze od jedné).

Abstraktní by mohla být například třída `DopravniProstredek` například s proměnnými `spotreba`, `objemNadrze` či `SPZ` a metodami `jet()`, `natankovat()` či `vylozitNaklad()`. Každá třída, která by představovala nějaký dopravní prostředek, by pak dědila z třídy `DopravniProstredek` vlastnosti a metody, které by byly případně upraveny pro potřeby daného potomka (např. Auto by vykládalo náklad jinak než třeba Naklaďák).

Rozhraní

Rozhraní (interface) je na rozdíl od abstraktní třídy jakýsi **seznam metod**, které by měla nějaká třída podporovat, aby plnila nějaký účel. Hlásí-li se třída k tomu, že podporuje nějaký interface, musí definovat všechny metody, které interface předepisuje. Interface definuje pouze hlavičky metod, implementovat je musí třída samotná.

Na rozdíl od dědění z abstraktní třídy může třída podporovat neomezené množství rozhraní. Rozhraní neobsahuje žádné proměnné.

Kolekce objektů příbuzných tříd

Úzce související data (typicky třídy a příbuzné třídy) mohou být zpracována efektivněji právě při seskupení do kolekce. Namísto zapsání samostatného kódu pro zpracování každého jednotlivého objektu je tak možné použít stejný kód (obvykle cyklus) ke zpracování všech prvků kolekce.

Polymorfismus

V dědičnosti v OOP je polymorfismus vlastnost umožňující volat jednu metodu z různých tříd (často pomocí rozhraní) konajíc tutéž funkci v rámci třídy, z níž byla metoda volána.

Například `Pracka` i `Televize` by mohly z rozhraní `DomaciSpotrebice` implementovat metodu `prepnoutProgram()` a následně ji volat. V každé třídě by vykonávala stejnou činnost (přepínala program, ale v jiném kontextu – v každé třídě by metoda vykonávala jiné kroky).