

Objekty - třídy a instance tříd

Objekty skutečného světa sdílejí dvě charakteristiky: všechny mají stav a všechny mají chování. Objekt musí mít proměnné, které indikují současný stav a má také metody.

Objekt je instance třídy objekt . Objekty mají společný stav a společné chování. V objektově orientovaném programu je možno mít mnoho objektů stejného typu, které sdílí nějaké charakteristiky. Při vytváření instance třídy se vytváří objekt, systém alokuje paměť pro proměnné deklarované ve třídě. Poté je možné vyvolávat metody objektu a provádět s objektem různé činnosti.

Vzájemným působením objektů získá program svoji funkčnost a složité chování. Programové objekty vzájemně působí na jiné objekty a komunikují s nimi pomocí zasílání zpráv. Když objekt A chce, aby objekt B provedl jednu ze svých metod, pak objekt A zasílá zprávu objektu B.

- Třída je deklarace typu objektu
- Objekt je konkrétní instance třídy
- Veřejná část třídy je veřejné rozhraní třídy
- Soukromá část třídy není přístupná z vnějšku třídy

Metody, atributy, konstruktory, destruktory

Objekt obsahuje metody objektu, což jsou procedury a funkce, které vykonávají nějakou činnost nad vnitřní pamětí objektu. Také *metody* objektu jsou z vnějšku neviditelné - nepřístupné, nelze tyto metody volat přímo.

Objekt obsahuje vnitřní paměť, tj. má vlastnost si něco pamatovat. Tato vnitřní paměť se nazývá *atributy* objektu. Vnitřní paměť objektu je z vnějšku objektu nepřístupná.

Konstruktory jsou metody, které vyvolají zrod objektu a následují bezprostředně po zrodu objektu. Do metody spojené se zrodem se většinou umísťují další zrody jiných, podřízených objektů, které je potřeba zrodit spolu s objektem, aby mohl fungovat. Objekt může mít několik konstruktorů a můžeme vyvolat podle potřeby ten, který odpovídá dané situaci, výsledkem je však vždy alokace paměti pro objekt a vyvolání metody daného konstrukturu.

Destruktor je metoda vyvolávající "zničení" objektu, jeho dealokaci a současně vyvolání metody zapsané v destrukturu. V destrukturu se většinou zavolají destruktory podřízených objektů.

Vlastnosti OOP - dědičnost, zapouzdření, polymorfismus

Objektově orientované systémy umožňují, aby třídy byly definovány pomocí jiných tříd. Každá podtřída *dí* stav (ve formě instancí proměnných) a metody od nadtřídy. Podtřída ale není omezena stavem a chováním poskytnutým jí její nadtřídou. Podtřída může k zděděným proměnným a metodám přidat své vlastní proměnné a metody. Podtřída také může přepisovat zděděné metody a poskytnout tak specializovanou implementaci těchto metod. Obecně, čím hlouběji v hierarchii tříd se třída vyskytuje, tím více specializované je

její chování. Podtřídy poskytují specializované chování na základě společných prvků poskytnutých nadtřídou. Programátoři mohou implementovat nadtřídy nazývané abstraktní třídy, které definují všeobecné chování. Abstraktní třídy mají částečnou implementaci chování, detaily jsou doplněny ve specializovaných podtřídách.

Proměnné objektu jsou umístěny uprostřed (v jádru) objektu. Metody obklopují a skrývají jádro objektu před ostatními objekty v programu. Zabalení proměnných objektu v jádru chráněném svými metodami se nazývá *zapouzdření*. Je obvykle použito k ukrytí implementačních detailů před jinými objekty. Nepotřebujeme znát, jak je třída implementována, potřebujeme pouze vědět, kterou metodu vyvolat.

Polymorfismus je vlastnost OOP, která umožňuje pojmenovat metodu jedním jménem a tato metoda může být společná pro různé objekty ve stromové hierarchii, i když pro každý objekt v této hierarchii se bude chovat různě. Výsledkem je, že pro každý objekt ve stromové hierarchii bude mít volání metody jinou odezvu, avšak pro každý objekt vždy tu správnou.

Statické a virtuální metody

Všechny metody, pokud nejsou virtual, jsou *statické metody*. Překladač jim vyhradí místo (adresu) v paměti již v době překladu a také řeší všechny vztahy mezi metodami a objekty již v době překladu. Těmto vazbám se říká *early binding* - brzká vazba. Statické metody jsou velmi rychlé při jejich vykonávání a jejich kód je velice efektivní.

Virtuální metody řeší pojem zvaný polymorfismus. Díky těmto metodám můžeme pojmenovat jedním jménem metodu, která je stejná pro všechny objekty ve stromové hierarchii, avšak pro každý objekt má tato metoda jinou implementaci. Virtuální metody umožňují vytvářet vazby mezi objekty a jejich metodami až v době výpočtu. Těmto vazbám se říká *late binding* - pozdní vazba. Tyto metody jsou pomalejší při vykonávání než metody statické. Virtuální metoda se označuje v objektu direktivou `virtual`. Adresa této metody je uložena v tabulce virtuálních metod. K naplnění tabulky VMT slouží speciální metoda, zvaná konstruktor. Konstruktor se musí vyskytovat v definici každého objektu, který využívá alespoň jednu virtuální metodu. Metoda, definovaná v objektu jako virtuální, musí být definovaná jako virtuální, musí být definovaná i ve všech potomcích tohoto objektu jako virtuální.

Vztahy mezi objekty, diagramy tříd

Pokud chceme vytvořit objektově orientovanou aplikaci, musí mezi sebou zúčastněné objekty nějak komunikovat. Tato komunikace představuje volání metod (někdy se také říká zasílání zpráv) mezi objekty, které jsou spolu svázány pomocí spojení. Spojení mezi objekty mohou být obousměrná, kdy oba objekty mají na sebe navzájem k dispozici odkaz, nebo jednosměrná, kdy odkaz má pouze jeden z této dvojice objektů, který pak druhý objekt řídí. Grafické znázornění objektů a jejich vztahů v určitém čase se nazývá *objektový diagram*.

Na úrovni tříd spojením odpovídají asociace, spojení jsou vlastně instancemi asociací. Každá *asociace* může být podrobněji popsána dalšími vlastnostmi jako název asociace, názvy rolí asociovaných tříd, násobnost a řiditelnost asociace. Název asociace vyjadřuje obvykle slovesnou frázi, kterou vykonává jeden objekt pomocí druhého objektu. Uvádíme buď pouze název asociace (případně se šipkou označující směr asociace), nebo pouze

Objektově orientované programování

názvy rolí, ne obojí současně. Násobnost asociace můžeme uvést také jako interval, např. 0..1, 1..*, 1..10 (1 firma, více zaměstnanců)

Grafické znázornění tříd a jejich asociací se nazývá třídní diagram. Tento diagram představuje statickou strukturu modelovaného systému, zachycuje jeho prvky a vztahy mezi nimi.

Agregace představuje volnou vazbu mezi celkem a součástí, kdy jeden objekt (celek) využívá služby dalších objektů (součástí). Agregace je formou asociace a v grafické podobě se odlišuje prázdným kosočtvercem na straně celku. (počítač 0..1 - tiskárna 0..*)

Silnější formou agregace je kompozice. Vztah mezi objekty je velmi těsný a neumožňuje samostatnou existenci součásti, aniž by byla připojena k nějakému celku. Navíc na rozdíl od agregace tato součást musí patřit pouze jen jedinému celku. Např. faktura a její položky