Java - úvod do OOP (3.díl)

30.09.2007 Filip Koval

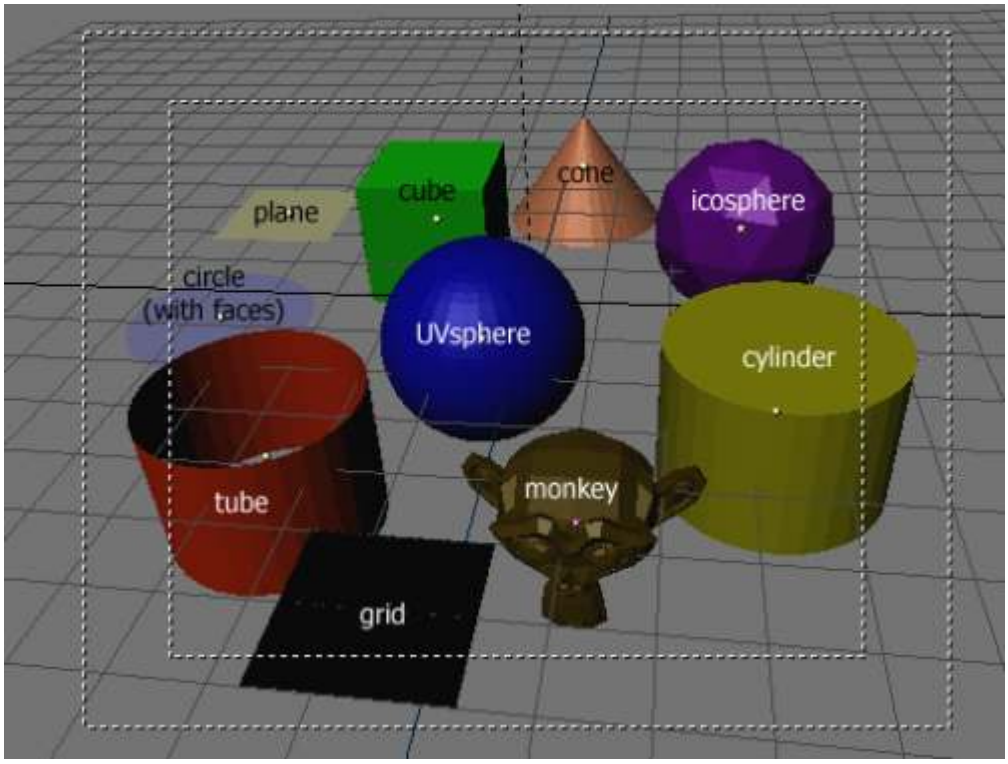
Úvod do objektově orientovaného programování v Javě

Java je založena na jazyce C++, je však více objektově orientovaným programovacím jazykem než C++. Lze prohlásit, že se jedná o čistokrevný objektově orientovaný programovací jazyk. V Javě je (prakticky) vše objektem, tento jednotný přístup umožňuje jednodušší osvojení základů jazyka a jeho používání. Programátor nemusí přemýšlet, zda pracuje či nepracuje s objektem, využívá tedy jednotnou syntaxi.

Úvod do objektově orientovaného programování

Objektově orientované programování (OOP) představuje metodický přístup k popisu a řešení problému. Na rozdíl od procedurálního programování vázaného na algoritmus umožňuje OOP popis problému realizovat komplexněji a efektivněji. OOP poskytuje model využívající obecné nástroje pro optimální řešení zadaného problému. S objektově orientovaným přístupem k řešení problému souvisí dva pojmy: objekt a metoda.

Objekty



Objekty nás obklopují i v reálném světě. V OOP představují komponenty nehmotné. Mohou spolu vzájemně komunikovat zasíláním požadavků (tj. zpráv), liší se svým vzájemným vnitřním stavem, který se mění v průběhu vykonávání programu. Objekt má tedy jako v jeho hmotná obdoba svůj vlastní vnitřní stav. Každý z objektů je tvořen daty ovlivňujícími jeho vlastnosti a metodami určujícími jeho chování. Proměnné objektu nazýváme

proměnné instance, metody objektu metody instance.

Každý program je tvořen skupinou různých typů objektů. Objekt si lze představit jako ovoce s jádrem uprostřed (např. švestku). Jádro představuje data, slupku tvoří metody. Metody umožňují objektu komunikovat se svým okolím, představují jeho rozhraní. Proměnné instance nejsou z vnějšku přímo viditelné, nemůžeme s nimi ani přímo manipulovat. K tomuto účelu používáme pouze metody. Popsaný princip nazýváme zapouzdřením.

Zapouzdření umožňuje před ostatními objekty zatajit vnitřní stav objektu. Při náhodných operacích nemohou jiné objekty měnit stav objektu přímo a zanést do něj nechtěné chyby. Taková nechtěná změna by mohla ovlivnit funkčnost jiné části programu. Výhody zapouzdření lze shrnout ve dvou bodech:

- zatajování informací: Objekt má své rozhraní, které mu umožňuje komunikovat s ostatními objekty. Proměnné instance nejsou jinému objektu přístupné z důvodu možného zavlečení chyby.

- modularita: každý objekt lze udržovat a spravovat nezávisle na jiném objektu, aniž by to nějak ovlivnilo celkovou funkčnost programu. Zveřejňování rozhraní není příliš častým případem, jedná se spíše o výjimku používanou ve speciálních případech. Vede k popření zásad a přístupu objektově orientovaného programování. Objekty mezi sebou komunikují, jak již bylo výše uvedeno, prostřednictvím zpráv. Pokud nějaký objekt A vyžaduje po objektu B, aby vykonal nějakou činnost, zašle mu zprávu v následujícím tvaru:

~ Objekt, na kterém se má akce provést

~ Činnost, která bude vykonána pomocí metody

~ Seznam parametrů předávaných metodě

Třída

Třída představuje šablonu pro tvorbu objektů. Určuje, jak bude objekt vypadat, a jak se bude chovat. Třída je tvořena komponentami nazývanými členy třídy, představují je:

- **Datové položky**

- **Metody**

Z jedné třídy můžeme vytvořit libovolný počet objektů, každý z nich má k dispozici svojí vlastní sadu proměnných instance, které byly iniciovány při jeho vytvoření. Proměnné jedné instance jsou nezávislé na proměnných jiné

instance, které byly iniciovány při jeho vytvoření. Proměnné jedné instance jsou nezávislé na proměnných jiné instance. Existují i tzv. proměnné třídy, které jsou společné všem instancím vytvořeným z jedné třídy. Proměnné třídy představují statické proměnné, vznikají při první inicializaci objektu a existují, dokud není program dokončen. Statické proměnné se používají pro sledování stavu instancí, např. zjištění jejich počtu.
Pro ovlivnění přístupu k členům třídy používá java tři následující klíčová slova: **public, private a protected**. Klíčové slovo **public** znamená, že přístup ke komponentám třídy má každý, tj jsou veřejně přístupné. Klíčové slovo **private** znamená, že z vnějšku třídy nebude mít ke komponentám přístup nikdo, jsou označovány jako soukromé. Klíčové slovo **protected** se pro rodičovskou třídu chová stejně jako private, odvozené třídy však mají ke komponentám **public** přístup. Základní vlastnosti objektově orientovaného programování lze shrnout do několika následujících
bodů:

- **Abstrakce**

Abstrakce umožňuje programátorovi abstrahovat se od detailů, jakým způsobem vnitřně pracují jednotlivé objekty. Pro práci s objektem mu postačuje znalost rozhraní, kterým objekt komunikuje se svým okolím.

- **Opětovné použití implementace**

Opětovné používání kódu je jedním ze základních pilířů OOP, hovoříme o tzv. Znovupoužitelném kódu. Existuje několik variant opětovného použití kódu: použití objektu jedné třídy v definici jiné třídy. Takovou třídu lze složit z libovolného počtu a typu různých objektů, tomuto přístupu říkáme skládání.

- **Polymorfismus**

Polymorfismus je vlastnost objektu. Určuje chování objektu v závislosti na jeho typu. V případě, že několik objektů poskytuje stejné rozhraní, pracujeme s nimi stejným způsobem, ale jejich konkrétní chování se liší.

- **Dědičnost**

Dědičnost představuje opětovné použití rozhraní. Umožňuje používat při návrhu tříd hierarchický přístup. Třídy na nižší úrovni mohou dědit vlastnosti a chování tříd nacházejících se na vyšší úrovni. Novou třídu můžeme odvodit z podobné existující třídy přidáním požadovaných funkcí a vlastností. V užším slova smyslu se pro spojení dědičnosti a polymorfismu používá termín objektově orientované programování.