Programování

Objektově orientované programování

Objektově orientovaný přístup

- Přístup ve kterém jednotlivé problémy dělíme na menší části
- Těmto částem říkáme objekty
- Zmíněné objekty spolu na příslušné úrovní interagují (komunikují)
- Objekty jsou modelovány jako předlohy skutečných věcí
- Objekty mají své specifické vlastnosti a funkce
- Objekty poskytují služby a jejich funkcionalitu dalším objektům
- V závislosti na míře abstrakce určujeme, jak je třeba mít modelované objekty přesné
- Žádný model není úplně přesný – potřebujeme vyvážit přesnost modelu mezi abstrakcí (model je velmi nepřesný) a absolutní kopií (model je zbytečně přesný)

Objektové modelování

- **Modelování** = vytváření abstrakce skutečného objektu
- Principy modelování:
  - **Abstrakce** – umožnit identifikovat podobnosti potřebné v modelu a odstínit se od nedůležitých rozdílů
  - **Formalizace** – umožnit efektivní komunikaci během vývoje nejen v týmu, ale i se zákazníkem (přesné vyjadřování)
  - **Jednoznačnost** – díky formalizaci můžeme přesně popsat jednotlivé části vytvářeného systému (data, funkce, vlastnosti, …)
  - **Zamezení redundancím** – modelování by mělo zamezit výskytu dvojího, vzájemně se vyvracejícímu tvrzení, případně duplicit

Objekt

- Celý svět vnímáme jako objekty
- Objekty mají svůj **stav**, **identitu**, příslušné **chování** a **vztahy** ke svému okolí
- V programování každý objekt ví v jakém stavu se nachází a na základě vnitřního mechanizmu ví, jak se má v konkrétních situacích zachovat
- Každý objekt je jedinečný s vlastní identitou
- Vlastnosti objektu nám umožňují specifikovat identitu objektu
- **Vlastnost** = pojmenování vztahu objekt : hodnota (abstrakce stavu objektu)
  - Příklad: 21 let -> abstrakcí získáváme vlastnost věk
- Specifikace vlastnosti objektu nazýváme **atribut**
- Chování objektu lze popsat jako posloupnost změn stavu vyvolaného impulzem – co konkrétní objekt dělá, na které impulzy umí reagovat
- Operace nám popisuje konkrétní chování objektu
- Konkrétní způsob provedení operace nazýváme **metoda**
- **Objekt je instancí třídy**

Třída (class)

- Třída je předpisem (šablonou) pro objekt
- Specifikuje veškeré metody a atributy, které bude objekt mít
- Objekty, které jsou vytvořeny na základě konkrétní třídy mají stejné vlastnosti a chování
- Jedná se o abstraktní datový typ popisující vnitřní strukturu objektu

- Definuje množinu objektů na základě:
  - Jaké jsou (atributy)
  - Co umí (metody)
  - Jak to, co umí provádí (implementace metod)

Objekt vs třída

Třída – popis deklarace

- Seznam atributů
  - Jméno
  - Výška
  - Váha
  - barva

Konkrétní objekt

- Konkrétní hodnoty
  - Čenda
  - 30 cm
  - 12 kg
  - Hnědo bíla

Koncepce OOP

- Způsob programování s důrazem na znovu použitelnost a snadnou rozšiřitelnost
- Objekty – základní stavební prvek OOP
- Abstrakce – zjednodušení skutečného objektu (zůstávají pouze potřebná data)
- Kompozice – objekt může být součástí jiného objektu
- Tři hlavní pilíře OOP:
  - **Dědičnost**
  - **Zapouzdření**
  - **Polymorfismus**

Dědičnost v OOP

- Dědičnost v OOP nám umožňuje vytvářet hierarchii mezi jednotlivými třídami
- Třídy mohou z předcházející , již existující, třídy dědit atributy a metody
- Vznikají nám tak podtřídy a super třídy
- **Podtřída** – třída, která je odvozena z některé třídy
- **Super třída** – třída, ze které je odvozováno, může vytvářet rozhraní pro odvozené třídy
- Odvozené metody z nadřazené třídy mohou být reimplementovány
  - Pro objekt vytvořený z této třídy se pak funkce chová odlišně
- C# umožňuje některé metody nebo atributy označit jako nedědítelné a odovozené třídy tak tyto části neobsahují

Dědičnost v OOP

- Mějme třídu člověk – každý člověk má nějaké jméno, příjmení a datum narození
- Odvozenými třídami z třídy člověk pak mohou být třídy, které jsou konkrétnější – student, zaměstnanec, učitel
- Třída student je odvozena z třídy člověk – zdědí její atributy a přidáme ji vlastní atributy, třída, ročník, obor
- Třída zaměstnanec je také odvozena z třídy člověk – zdědí atributy a přidáme vlastní atributy plat, pozice
- Student, zaměstnanec jsou tedy třídy odvozené a třída člověk je pro třídy student a zaměstnanec super třídou
- V C# je super třídou pro všechny ostatní odvozené třída **class Object**

Zapouzdření v OOP

- Zapouzdření chápeme jako zabalení vnitřní struktury třídy před okolím
- Třída se tak jeví jako černá skříňka, do které nevidíme
- Zapouzdření nám díky skrytí atributů a dat před ostatními třídami umožňuje snadno modifikovat konkrétní třídu aniž bychom ovlivnili chování jiné třídy
- Veškeré atributy třídy jsou uvedeny z privátní viditelností – přímo z atributy tak může pracovat pouze třída, které atributy náleží
- Výhodou je, že tím zabráníme neoprávněné změně atributů jiným než dovoleným způsobem
  - Nechceme například někomu umožnit změnit plat zaměstnanců
- Stejným způsobem můžeme před ostatními třídami skrýt i metody třídy

Polymorfismus v OOP

- Možnost objektu volat metodu s různými vstupními parametry
  - Již jsme se setkali – přetížení funkcí
- Volaní vlastní implementace funkce třídy, která danou metodu zdědila
  - Nadřazená třída nám může vytvářet rozhraní pro třídy odvozené
- Na třídy, které jsou odvozené ze stejné super třídy můžeme nahlížet jako na stejný typ objektu
- Třídy auto, autobus, motocykl jsou odvozené z třídy vozidlo – pokud například vytváříme pole vozidel, tak toto pole může obsahovat objekt třídy vozidlo, ale stejně tak objekty ze tříd auto, autobus a motocykl