Lekce 4
(Teoreticko-praktická část)

Objektově orientované programování (OOP) #1
(Úvod do objektů a tříd)

Prerekvizity

- V tuto chvíli bychom měli být seznámeni s těmito poznatky:
    - (Lekce 1) Deklarativní a imperativní (krokovátko) paradigma
    - (Lekce 1) Podstata imperativního programovacího jazyka, povědomí o existenci knihoven
    - (Lekce 2) Idea datových typů a základní datové typy (int, bool, string, objekt)
    - (Lekce 2) Proměnné a jejich odkazování na hodnoty, manipulace s jejich hodnotami
    - (Lekce 2) Větvení kódu podmínkou IF
    - (Lekce 2) Cykly jako FOR či WHILE, (FOREACH)
    - (Lekce 3) Funkce, metody a procedury
    - (Lekce 3) Princip definice funkcí a hodnot na jednom místě
    - (Lekce 3) Dokumentace
- Pokud některá z výše uvedených věcí není jasná, doporučuji se vrátit před pokračováním.

Pojmy objekt a třída a jejich vztah

- Anglicky či všeobecně v programovacích jazycích:
    - Objekt => object
    - Třída => class
- Pojmy spolu přímo souvisí
- Objekt dává smysl v imperativním kontextu, třída v deklarativním
- **Třída je obecná definice (deklarace), objekt je konkrétní instance této definice**
    - Například “čokoládový dort Sacher” je určitá **obecná** definice dána základní recepturou (~ třída), ale konkrétní produkt, který někdo na tom či onom místě v tom či onom čase vytvoří, je konkrétní instancí tohoto obecného předpisu, je **konkrétním** objektem (~ objekt).
    - Stejně tak třeba Škoda Octavia je nějaký obecný model (~ třída) automobilu, ale konkrétní produkt ten či onen je konkrétní instancí tohoto modelu (~ objekt).

- Do třetice: hra v šachy je čistě abstraktní idea (~ třída). Konkrétní partie (~ objekt) na určitém místě a s určitými figurkami je instancí onoho obecného předpisu, kterým je definována hra v šachy.

Deklarace třídy, instanciace nového objektu třídy (klíčové slovo “new”)

- Základní deklarace nějaké třídy (např. v php) vypadá asi takto:
  - class DortSacher {
    // zatím nic zde
    }
  - Běžnou praxí je, že každá třída je deklarovaná v samostatném souboru. V jazyce php tedy například soubor DortSacher.php
- Takto máme deklarovanou třídu, která však hraje roli pouze v deklarativním kontextu, nemáme žádnou její konkrétní instanci, objekt této třídy. Deklarace jedné třídy nijak nepředchází deklaraci jiné třídy či dokonce výkonu nějakého konkrétního příkladu - deklarace prostě jsou, současně (tzv. *staticky*).
  - Konkrétní objekt dané třídy vytvoříme pomocí klíčového slova “**new**” a názvu třídy (asi v drtivé většině objektových programovacích jazyků).
  - Například tedy: **new DortSacher();**
    - *[Závorky nyní zůstanou prázdné. Jejich význam se týká tzv. konstruktoru, o kterém bude řeč dále]*
- Aby bylo možné s objektem cokoli provádět, je třeba jej přiřadit nějaké proměnné, která od této chvíle bude **odkazovat** na vytvořený objekt. Provedeme to jednoduše takto:
  - $mojePromenna = new DortSacher();
  - Název proměnné většinou volíme dle třídy objektu, na který chceme odkazovat. Tedy lépe: $dortSacher = new DortSacher();
  - [Ale pozor, viz dřívější lekce: proměnná není tím objektem, pouze na objekt odkazuje a ten existuje v paměti stroje. Objekt totiž docela dobře může existovat i tehdy, když už na něj žádná proměnná neodkazuje. Tehdy nám bude zbytečně plnit operační paměť výpočetního stroje, což bývá obecně problematické.]

Obsah třídy #1 - pole

- Výše deklarovaná třída DortSacher je zatím bezcenná, protože s ní nelze vůbec nic dělat, nemá totiž žádný obsah a nic neumí, není k ničemu použitelná.
- Jedním z obsahů, které třída/objekt může mít, jsou tzv. “pole” (nikoli array!), neboli určité proměnné, které lze vnímat jako vlastnosti objektu a je zajištěno, že každý objekt určité třídy je bude mít shodné.

- Naši třídu můžeme obohatit o pole například takto (php):
  [Měníme původní předpis, nejedná se o posloupnost nějakých příkazů, nejde o imperativní kontext]
  - class DortSacher {
    // o klíčových slovech public/private později
    public $mnozstviCukru;
    public $velikostDortu;
    // zde definujeme poli výchozí hodnotu false
    public $maSlehacku = true;
    }
- Pole naplněná různými hodnotami dělají z objektu určitý balíček, kontejner, který shromažďuje data, která spolu souvisí. Protože s objekty se vždy zachází jako s celky, jednotlivá data budeme mít vždy pospolu v jednom balíčku.

Přístup k obsahům objektu

- S výjimkou statických obsahů (o tom později) se přistupuje k obsahům objektu pomocí nějaké značky aplikované na proměnnou odkazující na daný objekt. V php jde o značku šipky: “->”
- Příklad:
  $mujDortSacher = new DortSacher(); // inicializace objektu dortu
  echo $mujDortSacher->maSlehacku; // vypíše “1”
- Pomocí přístupové značky zpřístupňujeme proměnnou pole, tedy můžeme její obsah také měnit. Například:
  $mujDortSacher = new DortSacher(); // inicializace objektu dortu
  $mujDortSacher->mnozstviCukru = “300g”; // přiřadí hodnotu danému poli
  echo $mujDortSacher->mnozstviCukru; // vypíše “300g”

Jednotlivé instance (také stejných tříd) se od sebe odlišují

- Můžeme například napsat:
  $mujDortSacher1 = new DortSacher();
  $mujDortSacher2 = new DortSacher();
  $mujDortSacher1->mnozstviCukru = “150g”;
  $mujDortSacher2->mnozstviCukru = “300g”;
  echo $mujDortSacher1->mnozstviCukru; // vypíše 150g
  echo $mujDortSacher2->mnozstviCukru; // vypíše 300g
  $jsouStejne = $mujDortSacher1 === $mujDortSacher2; // má hodnotu false

Obsah třídy #2 - metody

- Druhý nejdůležitější obsah tříd/objektů tvoří metody (= či funkce).
- Stejně jako jsme deklarovaly metody v dřívějších lekcích, deklarujeme je zde uvnitř tříd:

```
class DortSacher() {
        public $mnozstviCukru;
        public $velikostDortu;
        public $maSlehacku = true;

        public function vratNazevDortu() {
                $nazevDortu = “Čokoládový dort Sacher”;
                return $nazevDortu;
        }
}
```

- K metodám, stejně jako polím, mohu přistupovat přes přístupovou značku:
    - $mujDort = new DortSacher();
      $nazevDortu = $mujDort->vratNazevDortu();
      echo $nazevDortu; // Vypíše Čokoládový dort Sacher

Konstruktor

- Objekty obecně mívají některé vyhrazené, již existující, speciální metody (záleží na programovacím jazyku či použitém frameworku). Jedna z nich je tzv. **konstruktor**. Jedná se o nejdůležitější speciální metodu, která se s odlišným zápisem nachází snad ve všech programovacích jazycích, které jsou objektově orientované.

Volání konstruktoru

- Konstruktor je metoda, která zpravidla nevrací žádnou hodnotu, může mít libovolný počet parametrů a především: volá se právě jedenkrát v životním cyklu objektu a to hned při jeho vzniku jakožto reakce na příkaz “new” (viz výše), kterým je objekt inicializován.

Parametry konstruktoru

- Má-li konstruktor nějaké parametry, pak jsou tyto vyžadovány při inicializaci objektu uvnitř závorek.
- Např.: new DortSacher($velikost, $mnozstviCukru);
- Metoda konstruktoru slouží ke vhodné inicializaci objektu. Může se jednat o víceméně libovolné úkony, ale většinou jde o uchování odkazů na jiné objekty či k definici nějaké hodnoty, která pak bude významněji určovat chování objektu.

Deklarace v php:

- class DortSacher() {
    public $mnozstviCukru;
    public $velikostDortu;
    public $maSlehacku = true;

    public function __constructor($velikost, $mnozstviCukru) {
        $this->mnozstviCukru = $mnozstviCukru;
        $this->velikostDortu = $velikost;
    }

    public function vratNazevDortu() {
        $nazevDortu = “Čokoládový dort Sacher”;
        return $nazevDortu;
    }
  }
- Pokud třída nedeklaruje žádný konstruktor (což je v pořádku), programovací jazyky tuto situaci chápou jako implicitní existenci prázdného konstruktoru, tedy konstruktoru, který nemá žádné parametry a nevykonává žádnou logiku.