Maturitní témata 2020/21

Otázka č. 27 – Počítačové sítě a programování

OOP:

- specifické programovací paradigma (základní programovací styl, liší se v pojmech a abstrakcích), které ho odlišilo od původního imperativního (procedurálnímu), zdrojový kód je v OOP přidružen k datům (metody jsou zapouzdřeny v objektech), což umožňuje snadnější přenos kódu mezi různými projekty (abstrakce a zapouzdření), propojení umožnilo zavést dědičnost ale kvůli zjednodušení si vyžádalo zavedení polymorfismu

Konstruktor:

- speciální metoda třídy která se volá ve chvíli vytváření instance této třídy, konstruktor se podobá ostatním metodám třídy ale liší se od nich tím že nemá nikdy explicitní návratový typ, nedědí se a obvykle má jiná pravidla pro modifikátory přístupu

- ve většině programovacích jazyků může být konstruktor přetížen takže jedna třída má několik konstruktorů s odlišnými parametry a odlišnou funkcionalitou

Výchozí (defaultní):

- pokud programátor nedodá konstruktor pro instantní třídu, vloží kompilátor do našeho kódu výchozí konstruktor, chování konstruktoru závisí na jazyce, může inicializovat datové členy na nulu nebo jiné stejné hodnoty nebo nemusí dělat vůbec nic, konstruktor bez parametrů

Obecný:

- výchází z C

Implicitní:

- konstruktor který nemusí být výslovně definován programátorem a kompilátor jazyka ho umí vytvořit automaticky, např. v jazyce C++ je defaultní (výchozí) konstruktor implicitní

Výchozí parametry:

- při volání funkce můžeme některé parametry vynechat a překladač nám nevynadá, jen uvnitř funkce budou mít potom hodnotu undefined

- pokud bychom chtěli dosadit nějakou výchozí hodnotu parametru v případě že není zadán, stačí si opodmínkovat jeho hodnotu pro undefined a dosadit hodnotu výchozí

```
function f(a, b) {
    document.write("a=" + a);
    document.write("b=" + b);
}

f(5);
f(1, 2);
```

```
def lol(b, a=5):
  print(a,b)
```

Dynamické objekty:

- dynamické objekty zveřejňují členy jako jsou vlastnosti a metody v době běhu místo v době kompilace, to umožňuje vytvářet objekty pro práci se strukturami které neodpovídají statickému typu nebo formátu, např. můžeme použít dynamický objekt pro odkaz na model DOM, HTML (DOM) který může obsahovat libovolnou kombinaci platných elementů a atributů HTML značek

Metody:

- v OOP podobné funkcím, v podstatě je to funkce která může pracovat s daty třídy nebo objektu, z vnějšku jsou data neviditelná - nepřístupná jelikož jsou zapouzdřena v objektu a nelze je volat přímo, metody určené k tomu aby s daty objektu mohly pracovat i jiné objekty nazýváme rozhraním objektu

- rozlišujeme volání uvnitř třídy a volání metody určitého objektu, v prvním případě se běžně používá pouze její název, ve druhém případě je nejdříve uveden název objektu a pak název volané metody

- typy metod:

- Statické metody - jsou součástí třídy ale lze je použít aniž by byla vytvářena instance třídy, chceme-li označit danou metodu jako statickou, použijeme klíčové slovo static
- Konečné (finální) metody - metody se mohou v odvozených třídách překrývat ale existují případy kdy chceme mít jistotu že danou metodu nebude možné změnit, slouží k tomu konečné metody které nemohou být přepsány v žádné z odvozených tříd, chceme-li označit danou metodu jako konečnou → final
- Abstraktní (virtuální) metody - při návrhu se lze setkat s případy kdy nechceme nechat implementaci určitých metod až na potomky, např. třída Obrazec která definuje obecný geometrický útvar a u níž víme, že potomci budou mít stejné metody (např. obvod a obsah) ale jejich implementace bude různá, pokud tedy označíme metodu jako abstraktní říkáme tím že tuto metodu implementuje její potomek, tyto metody se označují slovem abstract nebo virtual
- Speciální metody - sem patří metody pro tzv. zapouzdření: konstruktor, destruktor a tzv. „gettery“ a „settery“
  - poslední dvě metody slouží k získávání a nastavování hodnot příslušných vlastností třídy, nebývá vhodné samotné vlastnosti či členy tříd definovat jako veřejné, mohlo by např. dojít k nechtěné změně jejich hodnot

Druhy dědičnosti:

- Soukromá dědičnost:
  - používá se velmi zřídka
  ```
  class Programator : private Zamestnanec // Soukromá dědičnost
  ```
- Chráněná dědičnost:
  - jako soukromá dědičnost se moc nepoužívá
  ```
  class Programator : protected Zamestnanec // Chráněná dědičnost
  ```
- Veřejná dědičnost:

- veřejné položky zůstanou stejné (veřejné)

- Vícenásobná dědičnost:
  - daná třída dědí z více tříd najednou, třída A bude současně dědit z třídy B a C

```
class A : public B, public C
```

Typ dědičnosti Typ atributu (metody)	Soukromá dědičnost	Chráněná dědičnost	Veřejná dědičnost
soukromý (private)	položky budou přístupné jen přes základní třídu	položky budou přístupné jen přes základní třídu	položky budou přístupné jen přes základní třídu
chráněný (protected)	soukromé položky	chráněné položky	chráněné položky
veřejný (public)	soukromé položky	chráněné položky	veřejné položky