Objektově orientované programování

Úvod (organizace, témata, úkoly)

2023/24

Co je nového?

```
class KeyValue
{
    private:
        int key;
        double value;

    public:
        KeyValue(int k, double v);
        int GetKey();
        double GetValue();
};
```

Proč potřebujeme OOP

OOP je více o návrhu než o programování.

Zaměříme se na ty techniky OOP, jejichž využití se promítá do dobře organizovaného kódu s měřitelnou kvalitou.

Tento předmět není o programování v C++.

Budeme využívat jen minimální syntaxi a konstrukce jazyka C++.

Osnova prezentace

- Informace k výuce
- O předmětu
- Přehled témat
- Příklad

Informace k výuce

Kontakt

- doc. Mgr. Miloš Kudělka, Ph.D.
- milos.kudelka@vsb.cz
- http://homel.vsb.cz/~kud007
- EA 439, +420 597 325 877

Studijní požadavky

- Návštěvy přednášek jsou doporučené.
- Cvičení jsou povinná.
- Účast na cvičení může být podmíněna domácí přípravou nebo vypracovaným domácím úkolem.
- Požadovanou výstupní dovedností je schopnost napsat malý program s objektovým návrhem.

Zápočet

- Požadavky na získání zápočtu:
  - Aktivní účast na cvičeních (9 bodovaných cvičení + projekt, celkem je potřeba získat nejméně 23 bodů ze 45 možných).
  - Písemná zkouška s otázkami z přednášek a jednoduchými programovými kódy v C++ (nejméně je potřeba získat 28 bodů z 55 možných).
  - Za jedno cvičení je možno získat body za účast, odpovědi na kontrolní otázky z přednášky, aktivitu při řešení úloh a zpracování zadaného domácího úkolu (celkem maximálně 4 body za jedno cvičení).
  - Na konci semestru bude hodnocen projekt dle zadání cvičícího (maximálně 9 bodů).

O předmětu

Cíle předmětu

- Co?
  - Co rozumíme objektovými principy a technikami? Které to jsou a co jimi rozumíme?
- Proč?
  - Čím je motivována potřeba objektového programování? Co je účelem objektových principů a k čemu jsou jednotlivé techniky?
- Kdy?
  - Kdy bychom měli použít jednotlivé techniky a kdy ne?
- Jak?
  - Jak správně rozumět objektovým principům? Jak správně použít jednotlivé techniky?

Zdroje

- Meyer, B. *Object-Oriented Software Construction*. Prentice Hall, 1997, ISBN: 978-0136291558.

- Eckel B. *Myslíme v jazyku C++*. Grada Publishing, 2000, ISBN 80-247-9009-2.

- Stroustrup, B. *C++ Programovací jazyk*. BEN-technická literatura, Praha 1997.

Témata

Programovací paradigmata

- Jak a proč se programovací jazyky vyvíjely a vyvíjejí?
- V čem se objektově orientované programování (OOP) liší od jiných paradigmat?
- Jaké jsou aspekty kvality software?

Paradigma *(z řeckého παράδειγμα parádeigma = vzor, příklad, model) je* ***obecně přijímané schéma, vzorec myšlení, či model****. [Wikipedia]*

Třídy a objekty

- Co jsou třídy a co objekty v OOP?
- Třída je statický popis.
- Objekt je run-time (běhová) reprezentace, která má:
  - stav (data)
  - chování (funkčnost)

Principy OOP

- Obecně
  - Skrývání informace
  - Skládání
  - Zasílání zpráv
  - Vztah obecný – speciální
- Technicky
  - Zapouzdření
  - Polymorfismus
  - Dědičnost

Životní cyklus objektů

- Jak objekty vznikají a zanikají?
- K čemu jsou konstruktory a destruktory?
- Jak to celé funguje v různých situacích?

Skrývání informace

- Co je veřejná a co soukromá část objektu?
- Proč je důležité informace skrývat?
- Kdy je návrh veřejné a soukromé části objektu správný?

Dědičnost

- Jednoduchá dědičnost a důvody pro její použití (polymorfismus).
- Abstraktní třídy.
- Typy skrývání informace (implementace).
- Vícenásobná a opakovaná dědičnost. Problémy.

A další...

- Návrhy objektových programů.
- Generické typy.
- Výjimky.
- Objektové knihovny.

Příklad

Deklarace třídy

```
#include <iostream>
using namespace std;

class KeyValue
{
    private:
        int key;
        double value;

    public:
        KeyValue(int k, double v);
        int GetKey();
        double GetValue();
};
```

Definice (implementace) třídy

```
KeyValue::KeyValue(int k, double v)
{
    this->key = k;
    this->value = v;
}

int KeyValue::GetKey()
{
    return this->key;
}

double KeyValue::GetValue()
{
    return this->value;
}
```

Použití třídy

```
int main()
{
    KeyValue kv1(1, 1.5);
    cout << kv1.GetValue() << endl;

    KeyValue *kv2 = new KeyValue(2, 2.5);
    cout << kv2->GetValue() << endl;
    delete kv2;

    getchar();
    return 0;
}
```

Úkoly na cvičení

- Navrhněte, deklarujte a definujte jednoduché třídy a napište kód, který pracuje s objekty této třídy.
  - E-mail
  - Osoba
  - Dokument