Objektově orientované programování

Přednáška č. 1

Ing. Lukáš Slánský

Ústav elektrotechniky a informatiky, Univerzita Pardubice

5. 10. 2006

Obsah

Literatura

1. Ilja Kraval – Základy objektově orientovaného programování (ve STAGu)

Literatura

1. Ilja Kraval – Základy objektově orientovaného programování (ve STAGu)
2. Joseph Schmuller – Myslíme v jazyku UML

Literatura

1. Ilja Kraval – Základy objektově orientovaného programování (ve STAGu)
2. Joseph Schmuller – Myslíme v jazyku UML
3. Fowler Martin – Refaktoring – Zlepšení existujícího kódu

Historie programovacích stylů

Historie programovacích stylů

- Strojový kód

Historie programovacích stylů

- Strojový kód
  - pouze instrukce a skoky

Historie programovacích stylů

- Strojový kód
  - pouze instrukce a skoky
- Strukturované programování

Historie programovacích stylů

- Strojový kód
  - pouze instrukce a skoky
- Strukturované programování
  - přehlednější a čitelnější programovací styl

Historie programovacích stylů

- Strojový kód
  - pouze instrukce a skoky
- Strukturované programování
  - přehlednější a čitelnější programovací styl
- Modulární programování

Historie programovacích stylů

- Strojový kód
  - pouze instrukce a skoky
- Strukturované programování
  - přehlednější a čitelnější programovací styl
- Modulární programování
  - možnost spolupráce více programátorů na stejném projektu

Historie programovacích stylů

- Strojový kód
  - pouze instrukce a skoky
- Strukturované programování
  - přehlednější a čitelnější programovací styl
- Modulární programování
  - možnost spolupráce více programátorů na stejném projektu
- Objektově orientované programování

Historie programovacích stylů

- Strojový kód
  - pouze instrukce a skoky
- Strukturované programování
  - přehlednější a čitelnější programovací styl
- Modulární programování
  - možnost spolupráce více programátorů na stejném projektu
- Objektově orientované programování
  - spojení dat a práce s nimi do jednoho balíku, znovupoužitelnost

Historie programovacích stylů

- Strojový kód
  - pouze instrukce a skoky
- Strukturované programování
  - přehlednější a čitelnější programovací styl
- Modulární programování
  - možnost spolupráce více programátorů na stejném projektu
- Objektově orientované programování
  - spojení dat a práce s nimi do jednoho balíku, znovupoužitelnost
- Komponentové programování

Základní pojmy OOP

- Objekt (angl. object)

Základní pojmy OOP

- Objekt (angl. object)
- Zapouzdření (angl. encapsulation)

Základní pojmy OOP

- Objekt (angl. object)
- Zapouzdření (angl. encapsulation)
- Zpráva (angl. message)

Základní pojmy OOP

- Objekt (angl. object)
- Zapouzdření (angl. encapsulation)
- Zpráva (angl. message)
- Třída (angl. class)

Objekt

Objekt

- Má stav a chování

Objekt

- Má stav a chování
- Stav je uložen v proměnných – *atributech*

Objekt

- Má stav a chování
- Stav je uložen v proměnných – *atributech*
- Chování implementují funkce – *metody*

Objekt

- Má stav a chování
- Stav je uložen v proměnných – *atributech*
- Chování implementují funkce – *metody*
- *Žádný atribut ani metoda nejsou vidět z vnějšku*

Objekt

- Má stav a chování
- Stav je uložen v proměnných – *atributech*
- Chování implementují funkce – *metody*
- *Žádný atribut ani metoda nejsou vidět z vnějšku*
  - Tato vlastnost se nazývá zapouzdření – *encapsulation*

Zpráva

- Jak se dostat přes "slupku" zapouzdření?

Zpráva

- Jak se dostat přes "slupku" zapouzdření?
- Skrz slupku objektu vytvořím vstupně/výstupní kanál, přes který pošlu objektu zprávu a příjmu odpověď na ni.

Zpráva

- Jak se dostat přes "slupku" zapouzdření?
- Skrz slupku objektu vytvořím vstupně/výstupní kanál, přes který pošlu objektu zprávu a příjmu odpověď na ni.
- Tento mechanizmus je ve většině OO jazyků implementován jako volání metody.

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy
  - Alík, Punťa i Rex jsou Psi

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy
  - Alík, Punťa i Rex jsou Psi
  - mají stejné metody

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy
  - Alík, Punťa i Rex jsou Psi
  - mají stejné metody – umí štěkat, běhat, žrát, spát, ...

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy
  - Alík, Punťa i Rex jsou Psi
  - mají stejné metody – umí štěkat, běhat, žrát, spát, ...
  - mají stejné atributy

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy
  - Alík, Punťa i Rex jsou Psi
  - mají stejné metody – umí štěkat, běhat, žrát, spát, ...
  - mají stejné atributy – jméno, srst, oči, výšku, ...

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy
  - Alík, Punťa i Rex jsou Psi
  - mají stejné metody – umí štěkat, běhat, žrát, spát, ...
  - mají stejné atributy – jméno, srst, oči, výšku, ...
- Zavedeme novou (vyšší) abstrakci

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy
  - Alík, Punťa i Rex jsou Psi
  - mají stejné metody – umí štěkat, běhat, žrát, spát, ...
  - mají stejné atributy – jméno, srst, oči, výšku, ...
- Zavedeme novou (vyšší) abstrakci – třídů objektů Pes

Třída

- Více objektů má často stejné vzorce chování (metody) a stejné atributy
  - Alík, Punťa i Rex jsou Psi
  - mají stejné metody – umí štěkat, běhat, žrát, spát, ...
  - mají stejné atributy – jméno, srst, oči, výšku, ...
- Zavedeme novou (vyšší) abstrakci – třídů objektů Pes
  - Alík, Punťa, Rex i další psi jsou objekty ze třídy Pes

Příklad třídy v Pascalu – I.

```
type TPes=object
  public
    procedure JmenujesSe(noveJmeno:string);
    function JakSeJmenujes:string;
    procedure Stekej;
  private
    jmeno: string;
end;
```

Příklad třídy v Pascalu – II.

```
function TPes.JakSeJmenujes:string;
begin
  JakSeJmenujes:=jmeno;
end;
```

Příklad třídy v Pascalu – II.

```
function TPes.JakSeJmenujes:string;
begin
  JakSeJmenujes:=jmeno;
end;

procedure TPes.JmenujesSe(noveJmeno:string);
begin
  jmeno:=noveJmeno;
end;
```

Příklad třídy v Pascalu – II.

```
function TPes.JakSeJmenujes:string;
begin
  JakSeJmenujes:=jmeno;
end;

procedure TPes.JmenujesSe(noveJmeno:string);
begin
  jmeno:=noveJmeno;
end;

procedure TPes.Stekej;
begin
  WriteLn('Haf haf');
end;
```

Příklad třídy v Pascalu – III.

```
var Alik:TPes;
...
```

Příklad třídy v Pascalu – III.

```
var Alik:TPes;
...
Alik.JmenujesSe('Alíček');
```

Příklad třídy v Pascalu – III.

```
var Alik:TPes;
...
Alik.JmenujesSe('Alíček');
Alik.Stekej;
```

Příklad třídy v Pascalu – III.

```
var Alik:TPes;
...
Alik.JmenujesSe('Alíček');
Alik.Stekej;
WriteLn('Jmenuji se: ', Alik.JakSeJmenujes);
```

Zapouzdření v Pascalu

- Objektové rozšíření Pascalu (Object Pascal) je velmi jednoduché

Zapouzdření v Pascalu

- Objektové rozšíření Pascalu (Object Pascal) je velmi jednoduché
- Veškeré atributy a metody jsou viditelné v celém modulu, kde jsou deklarovány (mají přidělené zprávy přístupné v modulu)

Zapouzdření v Pascalu

- Objektové rozšíření Pascalu (Object Pascal) je velmi jednoduché
- Veškeré atributy a metody jsou viditelné v celém modulu, kde jsou deklarovány (mají přidělené zprávy přístupné v modulu)
- Mimo modul (tj. je-li třída definována v samostatném modulu) se pro řízení přístupu používají klíčová slova `public` a `private`

Příklad zapouzdření v Pascalu

```
type TPes=object
  public
    procedure JmenujesSe(noveJmeno:string);
    function JakSeJmenujes:string;
    procedure Stekej;
  private
    jmeno: string;
end;
```