Objektově orientované programování

Přednáška č. 5

Ing. Lukáš Slánský

Ústav elektrotechniky a informatiky, Univerzita Pardubice

2. 11. 2006

Obsah

Důsledky dědičnosti

- Potomek má všechny vlastnosti předka:

Důsledky dědičnosti

- Potomek má všechny vlastnosti předka:
  - má všechny atributy předka,
  - má všechny metody předka.

Důsledky dědičnosti

- Potomek má všechny vlastnosti předka:
  - má všechny atributy předka,
  - má všechny metody předka.
- Potomek je schopný zastupovat předka.

Důsledky dědičnosti

- Potomek má všechny vlastnosti předka:
  - má všechny atributy předka,
  - má všechny metody předka.
- Potomek je schopný zastupovat předka.
- Objekt potomka lze přiřadit do objektu předka

Dědičnost atributů v praxi

Dědičnost atributů v praxi

```
type TOsoba=object
       Jmeno:string;
       Prijmeni:string;
     end;
```

Dědičnost atributů v praxi

```
type TOsoba=object
       Jmeno:string;
       Prijmeni:string;
     end;
     TOsobaUPa=object(TOsoba)
       Fakulta:string;
     end;
```

Dědičnost atributů v praxi

```
type TOsoba=object
       Jmeno:string;
       Prijmeni:string;
     end;
     TOsobaUPa=object(TOsoba)
       Fakulta:string;
     end;
     TStudent=object(TOsobaUPa)
       rocnik:integer;
     end;
```

Dědičnost atributů v praxi

```
type TOsoba=object
       Jmeno:string;
       Prijmeni:string;
     end;
     TOsobaUPa=object(TOsoba)
       Fakulta:string;
     end;
     TStudent=object(TOsobaUPa)
       rocnik:integer;
     end;
     TZamestnanec=object(TOsobaUPa)
       kancelar:string;
     end;
```

Dědičnost atributů v praxi

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
```

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
begin
  WriteLn(Osoba.Fakulta);
end;
```

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
begin
  WriteLn(Osoba.Fakulta);
end;
...
var o:TOsoba;
    u:TOsobaUPa;
    s:TStudent;
    z:TZamestnanec;
```

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
begin
  WriteLn(Osoba.Fakulta);
end;
...
var o:TOsoba;
    u:TOsobaUPa;
    s:TStudent;
    z:TZamestnanec;
...
VypisFakultu(u);
```

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
begin
  WriteLn(Osoba.Fakulta);
end;
...
var o:TOsoba;
    u:TOsobaUPa;
    s:TStudent;
    z:TZamestnanec;
...
VypisFakultu(u);
VypisFakultu(s);
```

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
begin
  WriteLn(Osoba.Fakulta);
end;
...
var o:TOsoba;
    u:TOsobaUPa;
    s:TStudent;
    z:TZamestnanec;
...
VypisFakultu(u);
VypisFakultu(s);
VypisFakultu(z);
```

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
begin
  WriteLn(Osoba.Fakulta);
end;
...
var o:TOsoba;
    u:TOsobaUPa;
    s:TStudent;
    z:TZamestnanec;
...
VypisFakultu(u);
VypisFakultu(s);
VypisFakultu(z);
VypisFakultu(o);
```

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
begin
  WriteLn(Osoba.Fakulta);
end;
...
var o:TOsoba;
    u:TOsobaUPa;
    s:TStudent;
    z:TZamestnanec;
...
VypisFakultu(u);
VypisFakultu(s);
VypisFakultu(z);
VypisFakultu(o);    Nelze!!!
```

Dědičnost atributů v praxi

```
procedure VypisFakultu(Osoba:TOsobaUPa);
begin
  WriteLn(Osoba.Fakulta);
end;
...
var o:TOsoba;
    u:TOsobaUPa;
    s:TStudent;
    z:TZamestnanec;
...
VypisFakultu(u);
VypisFakultu(s);
VypisFakultu(z);
VypisFakultu(o);    Nelze!!!   (nezná atribut Fakulta)
```

Umístění atributů objektů v paměti

Umístění atributů v paměti

- Atributy jsou v objektu umístěny postupně dle zápisu ve zdrojovém kódu.

Umístění atributů objektů v paměti

Umístění atributů v paměti

- Atributy jsou v objektu umístěny postupně dle zápisu ve zdrojovém kódu.
- Metody nemají na polohu atributů žádný vliv.

Umístění atributů objektů v paměti

Umístění atributů v paměti

- Atributy jsou v objektu umístěny postupně dle zápisu ve zdrojovém kódu.
- Metody nemají na polohu atributů žádný vliv.
- Přidává-li zděděný objekt další atribut(y), jsou umístěny až za původní.

Změna metod při dědění

Metodu ve zděděné třídě lze předefinovat:

Změna metod při dědění

Metodu ve zděděné třídě lze předefinovat:

- Metoda musí mít stejný název.

Změna metod při dědění

Metodu ve zděděné třídě lze předefinovat:

- Metoda musí mít stejný název.
- Parametry i návratová hodnota se mohou lišit.

Změna metod při dědění

Metodu ve zděděné třídě lze předefinovat:

- Metoda musí mít stejný název.
- Parametry i návratová hodnota se mohou lišit.
- Pro další dědění je již používána předefinovaná metoda a stará je překryta.

Včasná a pozdní vazba

- Překladač má již při překladu jasno, jakou funkci (kterou metodu z které třídy) bude volat.

Včasná a pozdní vazba

- Překladač má již při překladu jasno, jakou funkci (kterou metodu z které třídy) bude volat.
  - *Včasná vazba* = již na začátku (při překladu) jsou spárovány volání metod s jejich umístěním v paměti.

Včasná a pozdní vazba

- Překladač má již při překladu jasno, jakou funkci (kterou metodu z které třídy) bude volat.
  - *Včasná vazba* = již na začátku (při překladu) jsou spárovány volání metod s jejich umístěním v paměti.
  - Používá se ve všech dosavadních případech použití volání metod.

Včasná a pozdní vazba

- Překladač má již při překladu jasno, jakou funkci (kterou metodu z které třídy) bude volat.
  - *Včasná vazba* = již na začátku (při překladu) jsou spárovány volání metod s jejich umístěním v paměti.
  - Používá se ve všech dosavadních případech použití volání metod.
- Program se rozhoduje kterou funkci volat až za běhu.

Včasná a pozdní vazba

- Překladač má již při překladu jasno, jakou funkci (kterou metodu z které třídy) bude volat.
  - *Včasná vazba* = již na začátku (při překladu) jsou spárovány volání metod s jejich umístěním v paměti.
  - Používá se ve všech dosavadních případech použití volání metod.
- Program se rozhoduje kterou funkci volat až za běhu.
  - *Pozdní vazba* = adresa metody se určuje až při volání.

Včasná a pozdní vazba

- Překladač má již při překladu jasno, jakou funkci (kterou metodu z které třídy) bude volat.
  - *Včasná vazba* = již na začátku (při překladu) jsou spárovány volání metod s jejich umístěním v paměti.
  - Používá se ve všech dosavadních případech použití volání metod.
- Program se rozhoduje kterou funkci volat až za běhu.
  - *Pozdní vazba* = adresa metody se určuje až při volání.
  - Budeme používat při použití některých předefinovaných metod.

Včasná a pozdní vazba

- Překladač má již při překladu jasno, jakou funkci (kterou metodu z které třídy) bude volat.
  - *Včasná vazba* = již na začátku (při překladu) jsou spárovány volání metod s jejich umístěním v paměti.
  - Používá se ve všech dosavadních případech použití volání metod.
- Program se rozhoduje kterou funkci volat až za běhu.
  - *Pozdní vazba* = adresa metody se určuje až při volání.
  - Budeme používat při použití některých předefinovaných metod.
  - Tento způsob nakládání s metodou je třeba explicitně přikázat.

Kniha pomocí včasné vazby

Kniha pomocí včasné vazby

```
type ETypObjektu=(obrazek, odstavec);
type CObjekt=object
       procedure vytiskniSe; {Abstraktní metoda}
       TypObjektu:ETypObjektu;
     end;
     CObrazek=object(CObjekt)
       procedure vytiskniSe;
     end;
     COdstavec=object(CObjekt)
       procedure vytiskniSe;
     end;
```

Kniha pomocí včasné vazby

```
type ETypObjektu=(obrazek, odstavec);
type CObjekt=object
       procedure vytiskniSe; {Abstraktní metoda}
       TypObjektu:ETypObjektu;
     end;
     CObrazek=object(CObjekt)
       procedure vytiskniSe;
     end;
     COdstavec=object(CObjekt)
       procedure vytiskniSe;
     end;
     CKniha=object
       Objekty:array[...] of CObjekt;
       procedure vytiskni;
     end;
```

Kniha pomocí včasné vazby

```
procedure CKniha.vytiskni;
var i:integer;
begin
  for i:=1 to PocetObjektu do
  begin
    case Objekty[i].TypObjektu of
      obrazek:  CObrazek(Objekty[i]).vytiskniSe;
      odstavec: COdstavec(Objekty[i]).vytiskniSe;
    end;
  end;
end;
```

Kniha pomocí pozdní vazby

```
procedure CKniha.vytiskni;
var i:integer;
begin
  for i:=1 to PocetObjektu do
    Objekty[i].vytiskniSe;
end;
```

Kniha pomocí pozdní vazby

```
type ETypObjektu=(obrazek, odstavec);
type CObjekt=object
       procedure vytiskniSe; {Abstraktní metoda}
       TypObjektu:ETypObjektu;
     end;
     CObrazek=object(CObjekt)
       procedure vytiskniSe;
     end;
     COdstavec=object(CObjekt)
       procedure vytiskniSe;
     end;
     CKniha=object
       Objekty:array[...] of CObjekt;
       procedure vytiskni;
     end;
```

Kniha pomocí pozdní vazby

```
type ETypObjektu=(obrazek, odstavec);
type CObjekt=object
       procedure vytiskniSe;virtual; {Abstraktní metoda}
       TypObjektu:ETypObjektu;
     end;
     CObrazek=object(CObjekt)
       procedure vytiskniSe;virtual;
     end;
     COdstavec=object(CObjekt)
       procedure vytiskniSe;virtual;
     end;
     CKniha=object
       Objekty:array[...] of CObjekt;
       procedure vytiskni;
     end;
```

Kniha pomocí pozdní vazby

```
type CObjekt=object
       procedure vytiskniSe;virtual; {Abstraktní metoda}

     end;
     CObrazek=object(CObjekt)
       procedure vytiskniSe;virtual;
     end;
     COdstavec=object(CObjekt)
       procedure vytiskniSe;virtual;
     end;
     CKniha=object
       Objekty:array[...] of CObjekt;
       procedure vytiskni;
     end;
```

Virtuální metody

- Jsou-li v objektu definovány virtuální metody, je jako jeden z atributů uložena i *tabulka virtuálních metod* (VMT – Virtual Methods Table)

Virtuální metody

- Jsou-li v objektu definovány virtuální metody, je jako jeden z atributů uložena i *tabulka virtuálních metod* (VMT – Virtual Methods Table)
  - Tabulka obsahuje ukazatele na metody, které se mají zavolat, pokud se program chystá provést některou z virtuálních metod.

Virtuální metody

- Jsou-li v objektu definovány virtuální metody, je jako jeden z atributů uložena i *tabulka virtuálních metod* (VMT – Virtual Methods Table)
  - Tabulka obsahuje ukazatele na metody, které se mají zavolat, pokud se program chystá provést některou z virtuálních metod.
  - Každý objekt si nese VMT s sebou a má ji unikátní stejně jako atributy.

Virtuální metody

- Jsou-li v objektu definovány virtuální metody, je jako jeden z atributů uložena i *tabulka virtuálních metod* (VMT – Virtual Methods Table)
  - Tabulka obsahuje ukazatele na metody, které se mají zavolat, pokud se program chystá provést některou z virtuálních metod.
  - Každý objekt si nese VMT s sebou a má ji unikátní stejně jako atributy.
- Při volání virtuální metody se program podívá do VMT, kde zjistí, kterou funkci má zavolat.

Používání virtuálních metod

Pro použití virtuálních metod musí být splněny následující podmínky:

Používání virtuálních metod

Pro použití virtuálních metod musí být splněny následující podmínky:

- Virtuální metoda musí mít stejnou deklaraci (tj. stejné atributy a návratová hodnota).

Používání virtuálních metod

Pro použití virtuálních metod musí být splněny následující podmínky:

- Virtuální metoda musí mít stejnou deklaraci (tj. stejné atributy a návratová hodnota).
- Třída musí obsahovat speciální metodu nazývanou *konstruktor*
  Např.: `constructor init;`

Používání virtuálních metod

Pro použití virtuálních metod musí být splněny následující podmínky:

- Virtuální metoda musí mít stejnou deklaraci (tj. stejné atributy a návratová hodnota).
- Třída musí obsahovat speciální metodu nazývanou *konstruktor*
  Např.: `constructor init;`
  - Konstruktor ještě před vykonáním svého kódu naplní VMT.

Používání virtuálních metod

Pro použití virtuálních metod musí být splněny následující podmínky:

- Virtuální metoda musí mít stejnou deklaraci (tj. stejné atributy a návratová hodnota).
- Třída musí obsahovat speciální metodu nazývanou *konstruktor*
  Např.: `constructor init;`
  - Konstruktor ještě před vykonáním svého kódu naplní VMT.
  - Konstruktor musí být volán před prvním použitím objektu.

Používání virtuálních metod

Pro použití virtuálních metod musí být splněny následující podmínky:

- Virtuální metoda musí mít stejnou deklaraci (tj. stejné atributy a návratová hodnota).
- Třída musí obsahovat speciální metodu nazývanou *konstruktor*
  Např.: `constructor init;`
  - Konstruktor ještě před vykonáním svého kódu naplní VMT.
  - Konstruktor musí být volán před prvním použitím objektu.
  - Použijete-li virtuální metodu před voláním konstruktoru, jsou následky nepredikovatelné (nejčastěji se program zhroutí i s DOSem).

Studentská přednáška

- Na poslední přednášku si mohou někteří připravit prezentaci tématu se vztahem k OOP.

Studentská přednáška

- Na poslední přednášku si mohou někteří připravit prezentaci tématu se vztahem k OOP.
- Rozsah cca 10-15 minut, pro 1-2 studenty.

Studentská přednáška

- Na poslední přednášku si mohou někteří připravit prezentaci tématu se vztahem k OOP.
- Rozsah cca 10-15 minut, pro 1-2 studenty.
- Bonus 10 bodů ke zkoušce.

Studentská přednáška

- Na poslední přednášku si mohou někteří připravit prezentaci tématu se vztahem k OOP.
- Rozsah cca 10-15 minut, pro 1-2 studenty.
- Bonus 10 bodů ke zkoušce.
- Témata:
  - Návrhové vzory (Design Patterns)
  - Agenty a multiagentní systémy
  - CORBA a jiné obdobné technologie