Shrnutí

Hromadné zpracování

Polymorfismus

OOP

Objekty v C#

- Jakákoliv entita
  - Tlačítko
  - Pero
  - Generátoro náhodných čísel
  - Ctverec
- …

Objekt obsahuje:

- **Proměnné** – základní charakteristiky objektu, které vždy obsahují určitou hodnotu.
- **Vlastnosti** – navenek se s nimi pracuje jako s proměnnými, ale ve skutečnosti jsou implementovány dvojicí metod: get – čtení vlastnosti, set – zápis do vlastnosti.
- **Proměnné** většinou programujeme jako soukromé private) a příslušné vlastnosti jako veřejné (public).
- **Metody –** jsou akce, které objekt umí provádět, vytváříme je jako podprogramy.
- **Události –** představují zprávy, vysílané objektem do okolí, typicky při nějaké změně jeho stavu.

Třída

- Datový typ (druh) objektu
- Jednotlivé objekty tříd – instance
- Např. tlačítka jsou instance třídy Button.

Dědičnost

- Sdílení kódu mezi podobnými třídami.
- Odvozená třída přebírá všechny složky bázové třídy (předka), další si může přidat a něco zděděného rovněž předefinovat.
- Zápis v definici třídy: class Potomek:Předek
- Konstruktory:
  Bezparametrický se dědí beze změny, u parametrického doplníme, co je navíc.
  public Kvadr(double a,double b, double c, double ro) :base(a,ro)

Hromadné zpracování dat

- Kdekoli je očekávána instance předka, lze použít instance potomka
- Do proměnné typu předek lze tedy uložit objekt typu potomek.
- Nelze tedy přímo provádět nic, co předek neumí nebo používat vlastnosti, které v ní nejdou nadefinované.

Hromadné zpracování dat

- Objekty odvozených tříd lze zpracovávat v cyklech – typicky: **foreach**
- Protože ovšem předek nemusí mít (a obvykle nemá) všechny vlastnosti a metody potomka, používáme přetypování pomocí operátorů **is** a **as**
- **is** zjistí typ
- **as** provede přetypování

Třída čtverec

```
class Ctverec
   {
      double a;
      public double A
      {
         get
         {
            return a;
         }
         set
         {
            a = value;
         }
      }
         public Ctverec()
         {
         }
         public Ctverec(double a)
         {
            this.A=a;
         }

         public double ObsahCtverce()
         {
            return A * A;
         }
         public string InfoCtverce()
         {
            return "Ctverec: a = " + a.ToString("F2") + " S = " + ObsahCtverce().ToString();
         }
       }
```

Třída obdélník

```
class Obdelnik:Ctverec
   {
      double b;
      public double B
      {
         get
         {
            return b;
         }
         set
         {
            b = value;
         }
      }
         public Obdelnik():base()
         {
         }
         public Obdelnik(double a, double b):base(a)
         {
            this.B=b;
         }

         public double ObsahObdelnika()
         {
            return A * B;
         }
         public string InfoObdelnika()
         {
            return "Obdelnik: a = " + A.ToString("F2")+" b = " + B.ToString("F2")+ " S = " + ObsahObdelnika().ToString();
         }
   }
```

Přetypování – příklad

```
foreach (Ctverec c in utvary)
                if (c is Obdelnik)
                textBoxVypis.Text += (c as
Obdelnik).InfoObdelnika() +
Environment.NewLine;
                else
                    textBoxVypis.Text +=
c.InfoCtverce() + Environment.NewLine;
```

Polymorfismus

- Při předefinování stejnojmenných metod předka většinou potřebujeme polymorfní chování:
  - Aby se používala metoda instance potomka, i když s ní pracujeme v typu předka
- Metodu bázové třídy, u které požadujeme toto chování označujeme slovem **virtual,** metodu odvozené třídy, která ji předefinovává pak slovem **override**.

Polymorfismus – příklad

```
class Ctverec
    {
        …
        public virtual double Obsah()
        {
            return A * A;
        }
        public virtual string Info()
        {
            return "Ctverec: a = " + a.ToString("F2") + " S = " + Obsah().ToString();
        }
    }
```

Polymorfismus – příklad

```
class Obdelnik:Ctverec
    {
            …
             public override double Obsah()
             {
                 return A * B;
             }
             public override string Info()
             {
                 return "Obdelnik: a = " + A.ToString("F2")+" b = " + B.ToString("F2")+ " S = " + Obsah().ToString();
             }
    }
```

Polymorfismus – příklad

```
private void button1_Click(object sender, EventArgs e)
        {
            Ctverec c1 = new Ctverec(1);
            Ctverec c2 = new Ctverec(10);
            Obdelnik o1 = new Obdelnik(3, 4);
            Obdelnik o2 = new Obdelnik(5, 6);
            List<Ctverec> utvary = new List<Ctverec>();
            utvary.Add(c1);
            utvary.Add(c2);
            utvary.Add(o1);
            utvary.Add(o2);

           foreach (Ctverec c in utvary)
             textBoxVypis.Text += c.Info() + Environment.NewLine;
           }
```