Číslo a název šablony	III/2 Inovace a zkvalitnění výuky prostřednictvím ICT
Číslo didaktického materiálu	EU-OPVK-VT-III/2-ŠR-312
Druh didaktického materiálu	DUM
Autor	RNDr. Václava Šrůtková
Jazyk	čeština
Téma sady didaktických materiálů	Programování v C# v příkladech III
Téma didaktického materiálu	Metody objektů
Vyučovací předmět	Seminář z informatiky
Cílová skupina (ročník)	Žáci ve věku 17–18 let
Úroveň žáků	Středně pokročilí
Časový rozsah	1–2 vyučovací hodiny
Klíčová slova	Objekty, metody, třídy, konstruktor
Anotace	Studenti programují třídy grafických objektů a jejich metody
Použité zdroje	DRÓZD, Januš a Rudolf KRYL. *Začínáme s programováním*. 1.vyd. Praha: Grada, 1992, 306 s. ISBN 80-854-2441-X. ELLER, Frank. *C# - začínáme programovat: podrobný průvodce začínajícího uživatele*. 1. vyd. Praha: Grada, 2002, 240 s. ISBN 80-247-0324-6. OCHRANOVÁ, Renata a Michal KOZUBEK. *Objektově orientované programování v Turbo Pascalu*. 1. vyd. Brno: Masarykova univerzita, 1993, 117 s. ISBN 80-210-0659-5. TÖPFEROVÁ, Dana a Pavel TÖPFER. *Sbírka úloh z programování*. Vyd. 1. Praha: Grada, 1992, 98 s. Educa '99. ISBN 80-854-2499-1. VYSTAVĚL, Radek. *Moderní programování: sbírka úloh k učebnici pro středně pokročilé*. 1. vyd. Ondřejov: moderníProgramování, 2008-2009, 2 sv. ISBN 978-80-903951-3-8. VYSTAVĚL, Radek. *Moderní programování: učebnice pro pokročilé*. Ondřejov: moderníProgramování, 2011, 149 s. ISBN 978-80-903951-7-6. VYSTAVĚL, Radek. *Moderní programování: učebnice pro středně pokročilé*. Ondřejov: moderníProgramování s.r.o, 2008. ISBN 978-80-903951-2-1.
Typy k metodickému postupu učitele, doporučené výukové metody, způsob	Text je možno využít ke společné práci, samostatné přípravě studentů, domácímu studiu

hodnocení, typy k individualizované výuce apod.	apod. Při společné práci je vhodné nejprve obtížnější úlohy rozebrat, potom společně se studenty implementovat na počítači. (Rozbor nejlépe na tabuli, synchronní řešení s promítáním) Prezentace obsahuje stručné shrnutí poznatků potřebných pro řešení příkladů. V pracovním listu je zadání cvičení – většinou se jedná o úlohy, které by měli studenti naprogramovat samostatně. Není nutné, aby všichni zpracovali všechno, vhodné je diferencovat podle jejich zájmu a schopností. Obtížnější úlohy jsou označeny hvězdičkou. Součástí materiálu je zdrojový kód těchto příkladů. Návrh způsobu hodnocení: ohodnocení samostatné práce během hodiny např. podle volby a počtu úloh a elaborace řešení (efektivnost, komentáře…).

Metodický list k didaktickému materiálu

Prohlášení autora

Tento materiál je originálním autorským dílem. K vytvoření tohoto didaktického materiálu nebyly použity žádné externí zdroje s výjimkou zdrojů citovaných v metodickém listu.

Obrázky (schémata a snímky obrazovek) pocházejí od autora.

312. Objekty II – metody

Příklad s kolečky byl na psaní poněkud komplikovaný – i když se kolečko skládalo z několika proměnných, přesto jsme je kreslili obvyklým způsobem. Bude mnohem pohodlnější a přehlednější, naučíme-li kolečko, aby se nakreslilo samo – vytvoříme pro ně metodu kresli.

Metody jsou součástí definice třídy, mohou a nemusejí mít parametry, mohou a nemusejí mít návratovou hodnotu. Speciálním typem metod jsou konstruktory.

Konstruktor je metoda, která se jmenuje stejně jako třída a volá se při vytvoření její nové instance. Protože jsou nově vytvářené třídy automaticky vybavené bezparametrickým konstruktorem, bylo možné psát v našem příkladu:

Kolo k = new Kolo()

– podobně jako Random nahoda = new Random(), aniž bychom něco programovali. Vytvoření parametrického konstruktoru si ukážeme dále.

Připomeňme si definici třídy kolo a začneme nejjednodušší metodou rostu, která zvětší poloměr kola o konstantní hodnotu.

```
class Kolo
    {
        public int x, y, r, d;
        public Color barva;
        //x,y - střed, r - poloměr, d - tloušťka pera
public void rostu()
        {
            r += 5;
        }
…dále popisované metody
}
```

Poloměr můžeme také zvětšit o nějakou konkrétní hodnotu – pak použijeme metodu s parametrem.

```
public void rostuo(int delta)
        {
            r += delta;
        }
```

Metoda pro kreslení bude mít jako parametr objekt typu Graphics – tedy typ kreslicí plochy, jinak bychom nevěděli, kam se kolečko bude kreslit a jak.

```
public void kresli(Graphics p)
        {
            Pen pero = new Pen(barva, d);
            p.DrawEllipse(pero,x - r, y - r, 2 * r, 2 * r);
        }
```

Třída může mít více konstruktorů – doplňme parametrický konstruktor, který bude inicializovat její proměnné.

Obecně:

public Jmeno_Tridy(parametry)

{

Příkazy inicializující novou instanci

}

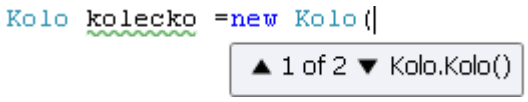
```
public Kolo(int x, int y, int r, Color b)
```

```
        {
            this.x=x;
            this.y=y;
            this.r = r;
            barva = b;
        }
```

Klíčové slovo **this** označuje odkaz objektu sám na sebe – this.r je tedy poloměr kolečka, do kterého dosazujeme parametr konstruktoru, který se jmenuje také r. U barvy nemůže vzniknout omyl – slovo this není nutné uvádět.

Použití třídy kolo a jejích metod v programu

Teď by ovšem systém hlásil chybu – postrádal by parametry b konstruktoru. Proto doplníme ještě sami bezparametrický konstruktor, podle užitých parametrů při volání bude jasné, který je třeba použít. (Různé varianty se v parametrech musí lišit).




```
public Kolo()
        {
        }
namespace Objekty1
{
    public partial class Form1 : Form
    {
        Kolo k1, k2, k3,pk;//deklarace koleček
        public Form1()

        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
              k1=new Kolo();
```

```
            k1.r = 100;
            k1.x = 200;
            k1.y = 100;
            k1.barva = Color.Brown;
            k1.d = 5;
            k2 = new Kolo() { r = 200, x = 300, y = 200, d = 2, barva =
Color.Blue };
            k3 = new Kolo() { r = 50, x = 100, y = 200, d = 1, barva =
Color.Green };
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics kp=e.Graphics;
            k1.kresli(kp);
            k2.kresli(kp);
            k3.kresli(kp);
            Kolo kolecko = new Kolo(200, 200, 20, Color.Black);
            //ukázka parametrického konstruktoru
            kolecko.kresli(kp);
         }

        private void radioButtonK1_CheckedChanged(object sender, EventArgs e)
        {
            if (radioButtonK1.Checked)
                pk = k1;
            if (radioButtonK2.Checked)
                pk=k2;
            if (radioButton3.Checked)
                pk=k3;;
         }
```

```
        private void buttonrostu_Click(object sender, EventArgs e)
        {
            int x = Convert.ToInt32(textBox1.Text);
            pk.rostuo(x);
            Refresh();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            pk.rostu();
            Refresh();
        }
    }
}
```

Důležité

Metody jsou součástí definice třídy, mohou a nemusejí mít parametry, mohou a nemusejí mít návratovou hodnotu.

Konstruktor je metoda, která se jmenuje stejně jako třída a volá se při vytvoření její nové instance.

Nově vytvářené třídy jsou automaticky vybavené bezparametrickým konstruktorem.

Třída může mít více konstruktorů.

public Jmeno_Tridy(parametry)

{

Příkazy inicializující novou instanci

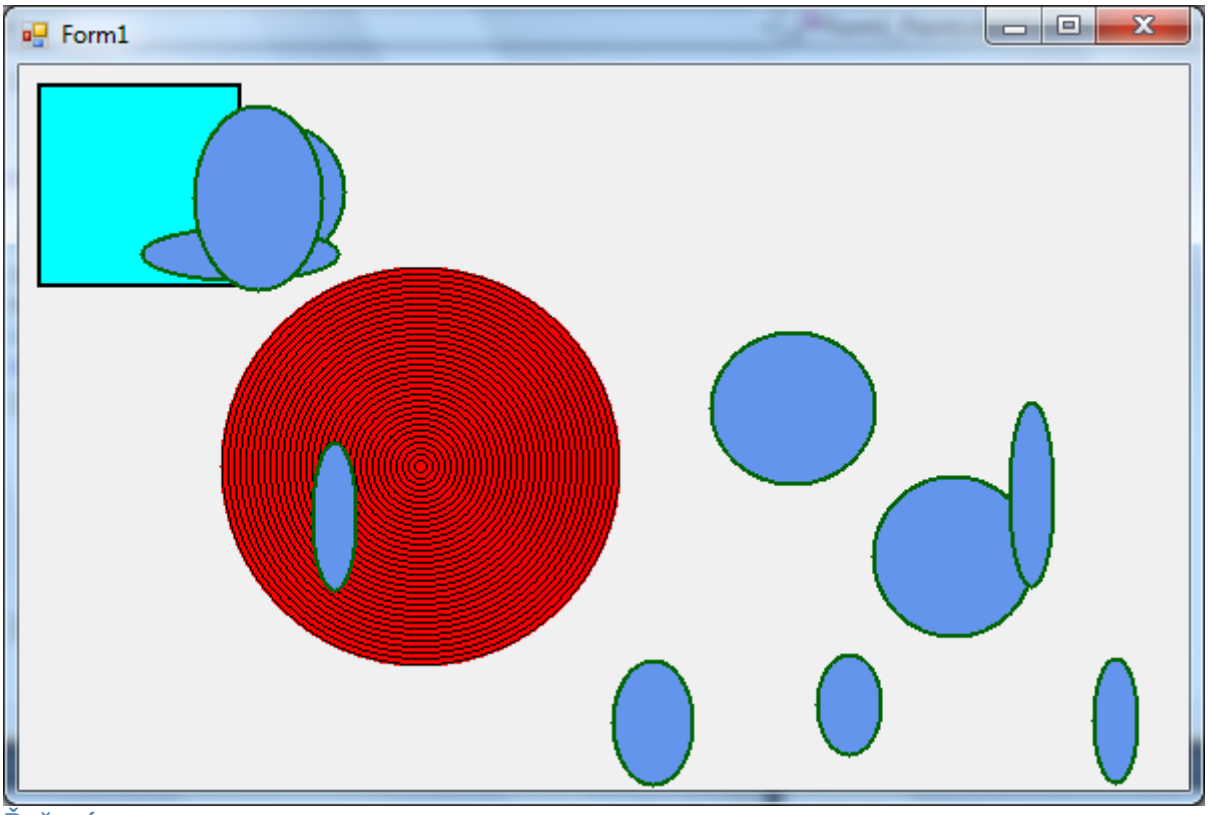
}

Klíčové slovo **this** označuje odkaz objektu sám na sebe

Pracovní list

Cvičení

Definujte další grafické třídy – čtverec, elipsu, obdélník, několik soustředných kružnic apod. Přidejte jim barvu výplně, definujte metody na kreslení, konstruktory, případně další metody. Rozhodněte sami, jakými parametry je budete reprezentovat.



Řešení

Může být individuální, spíš pro inspiraci.

```
class Elipsa
    {
        public int a; //hlavní poloosa
        public int b; //vedlejší poloosa
        public int x, y;//souřadnice středu
        public Color obrys;
        public int d; //tloušťka obrysu
        public Color vypln; //barva výplně
        public Elipsa()
        {
            // bezparametrický konstruktor
        }
        public Elipsa(int a,int b, int x, int y, int d, Color vypln, Color obrys)
        {
            // parametrický konstruktor
            this.a = a;
            this.b = b;
            this.x = x;
            this.y = y;
            this.d = d;
            this.vypln = vypln;
```

```
            this.obrys = obrys;
        }
        public void KresliElipsu(Graphics p)
        {
            Pen pero = new Pen(obrys, d);
            Brush stetec = new SolidBrush(vypln);
            p.FillEllipse(stetec, x-a, y-b, 2*a, 2*b);
            p.DrawEllipse(pero, x - a, y - b, 2 * a, 2 * b);
        }
    }
class Terc
    {   public int r; //poloměr terče
        public int x, y;//souřadnice středu
        public Color obrys;
        public Color vypln;
        public Terc()
        {
            // bezparametrický konstruktor
        }
        public Terc(int r, int x, int y,  Color vypln, Color obrys)
        {
            // parametrický konstruktor
            this.r = r;
            this.x = x;
            this.y = y;
            this.vypln = vypln;
            this.obrys = obrys;
        }
        public void Kresliterc(Graphics p)
        {
            Pen pero = new Pen(obrys, 1);
            Brush stetec = new SolidBrush(vypln);
            p.FillEllipse(stetec, x-r, y-r, 2*r, 2*r);
            for (int i=1;i<r;i++)
                if (i %3==0)
                  p.DrawEllipse(pero, x-i, y-i, 2*i, 2*i);
        }
        }
class Ctverec
    {
        public int a; //strana
        public int x, y;//souřadnice levého horního rohu
```

```
        public Color obrys;
        public int d; //tloušťka obrysu
        public Color vypln; //barva výplně
        public Ctverec()
        {
            // bezparametrický konstruktor
        }
        public Ctverec(int a, int x, int y, int d, Color vypln, Color obrys)
        {
            // parametrický konstruktor
            this.a = a;
            this.x = x;
            this.y = y;
            this.d = d;
            this.vypln = vypln;
            this.obrys = obrys;
        }
        public void KresliCtverec(Graphics p)
        {
            Pen pero = new Pen(obrys, d);
            Brush stetec = new SolidBrush(vypln);
            p.FillRectangle(stetec, x, y, a, a);
            p.DrawRectangle(pero, x, y, a, a);
        }
    }
namespace _312_Cvičení
{
    public partial class Form1 : Form
    {
        Ctverec c1 = new Ctverec(100, 10, 10, 2, Color.Aqua, Color.Black);
        Terc t = new Terc(100, 200, 200, Color.Red, Color.Black);
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics kp = e.Graphics;
            c1.KresliCtverec(kp);
            t.Kresliterc(kp);
            Random nahoda = new Random();
```

```
            for (int i = 0; i < 10; i++)
            {
                int x=nahoda.Next(500)+50;
                int y=nahoda.Next(300)+50;
                int a=nahoda.Next(40)+10;
                int b=nahoda.Next(40)+10;
                Elipsa e1=new Elipsa(a,b,x,y,2,Color.CornflowerBlue,Color.DarkGreen);
                e1.KresliElipsu(kp);
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```