

# Vlastnosti

## 8. cvičení

Jiří Zacpal

KMI/ZP3CS – Základy programování 3 (C#)

# Vlastnosti

- vlastnost je kříženec datové složky a metody
- syntaxe:

```
modifikátor Typ Nazev
{
    get
    {
        //kód pro čtení vlastnosti
    }
    set
    {
        //kód pro zápis vlastnosti
    }
}
```

# Použití vlastností

- příklad:

```
class Bod
{
    private int x,y;
    public int X
    {
        get{return this.x;}
        set{this.x=value;}
    }
}
Bod b=new Bod();
b.X=10;
```

# Vlastnosti jen pro čtení

- obsahuje pouze přístupovou metodu get
- syntaxe:

```
modifikátor Typ Nazev
{
    get
    {
        //kód pro čtení vlastnosti
    }
}
```

# Vlastnosti jen pro zápis

- obsahuje pouze přístupovou metodu set
- syntaxe:

```
modifikátor Typ Nazev
{
    set
    {
        //kód pro zápis vlastnosti
    }
}
```

# Přístupnost vlastností

- přístupnost vlastností (public, private, protected) je možné definovat pro jednotlivé metody get a set
- platí tato pravidla:
  - přístupnost lze změnit jen u jedné ze dvou přístupových metod
  - modifikátor nesmí dát přístupové metodě větší přístupnost, než jakou má celá vlastnost

# Omezení vlastností

- hodnotu lze přiřadit až po vytvoření objektu
- vlastnost nelze použít jako parametr s modifikátorem `ref` a `out`
- metody `set` a `get` nemají parametry
- u vlastností nelze použít modifikátor `const`

# Deklarace vlastností v rozhraní

- v rozhraní lze také definovat vlastnosti
- tělo metod get a set je ale nahrazeno středníkem
- všechny třídy, které implementují toto rozhraní, musí implementovat i vlastnosti
- příklad:

```
interface IPozice
{
    int X {get; set;}
    int Y {get; set;}
}
```



# Generování automatických vlastností

- kompilátor umožňuje generovat vlastnosti automaticky
- příklad:

```
class Kruh
{
    public int Polomer {get; set;}
}
```

- kompilátor převede třídu takto:

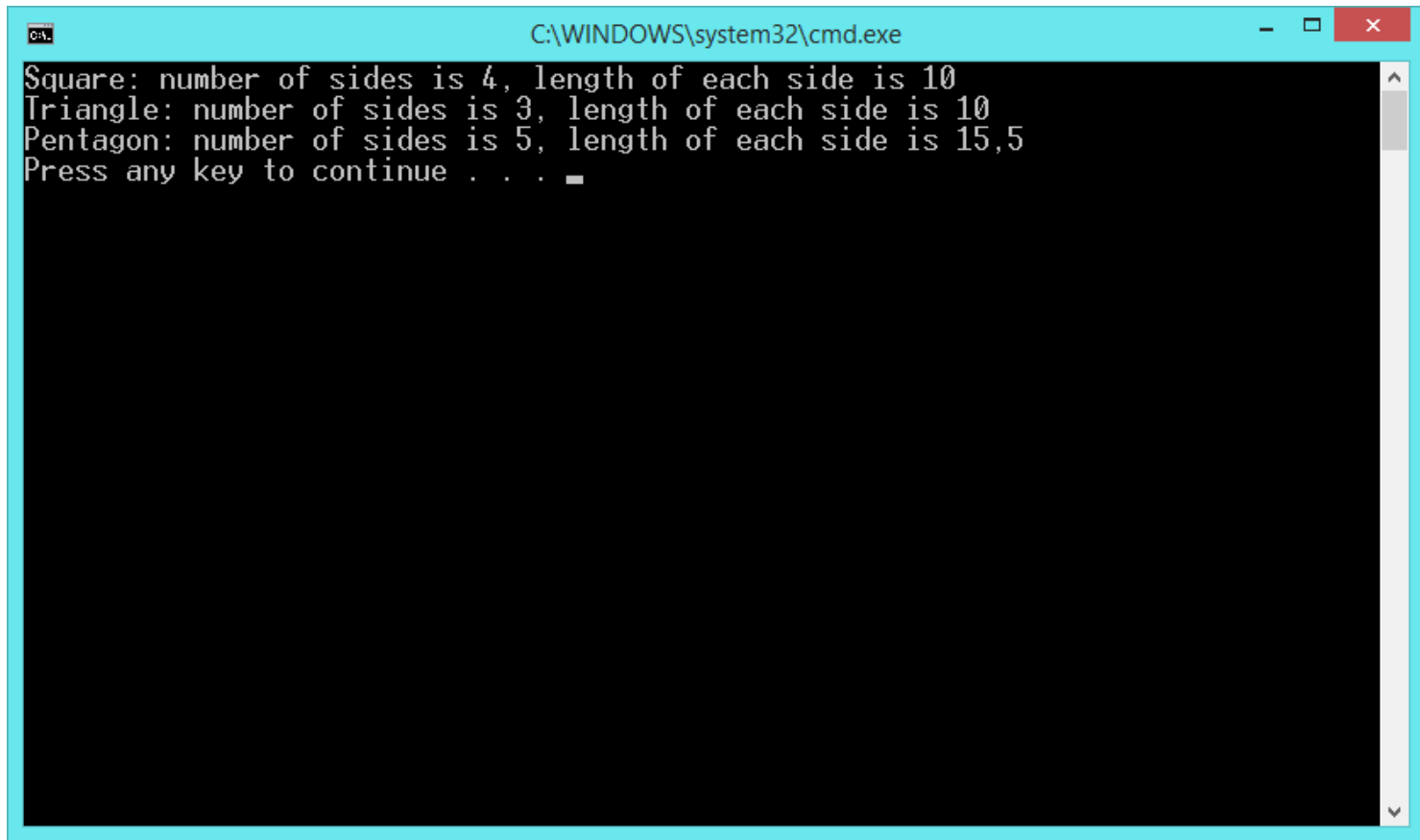
```
class Kruh
{
    private int _polomer;
    public int Polomer
    {
        get{return this._polomer;}
        set{this._polomer=value;}
    }
}
```

# Objektový inicializátor

- při vytvoření objektu umožní inicializovat datové složky pomocí vlastností
- není potřeba psát tolik konstruktorů
- příklad:

```
class Trojuhelnik
{
    private string nazev;
    private int stranaA=10;
    private int stranaB=10;
    private int stranaC=10;
    public Trojuhelnik(string n){this.nazev=n;}
    public int DelkaA{set{this.stranaA=value;}}
    public int DelkaB{set{this.stranaB=value;}}
    public int DelkaC{set{this.stranaC=value;}}
}
Trojuhelnik t1=new Trojuhelnik{DelkaC=15};
Trojuhelnik t2=new Trojuhelnik{DelkaA=15, DelkaC=20};
Trojuhelnik t3=new Trojuhelnik{DelkaB=12, DelkaC=17};
Trojuhelnik t4=new Trojuhelnik(„Rovnostranný“){DelkaA=9, DelkaB=25, DelkaC=30};
```

# Příklad 1



A screenshot of a Windows command prompt window. The title bar is light blue and contains the text "C:\WINDOWS\system32\cmd.exe" along with standard window controls (minimize, maximize, close). The command prompt itself has a black background with white text. The text displayed is as follows:

```
Square: number of sides is 4, length of each side is 10  
Triangle: number of sides is 3, length of each side is 10  
Pentagon: number of sides is 5, length of each side is 15,5  
Press any key to continue . . .
```

The text is left-aligned and uses a monospaced font. A small cursor is visible at the end of the last line.

# Indexery

- je to „chytré“ pole
- podobně jako vlastnost zapouzdřuje skupinu hodnot (pole kolekce)
- syntaxe:

```
public typ this [typ index]
{
    get{}
    set{}
}
```

# Vlastnosti indexerů

- indexer není metoda
- indexer používá klíčové slovo `this`
- indexer obsahuje přístupové metody `get` a `set`

# Srovnání indexerů a polí

- indexer mohou používat nečíselné indexy
- indexer lze přetěžovat
- indexer nelze použít jako parametry s modifikátory ref a out, zatímco prvky pole ano

# Příklad 2

Phone Book

Name: Pavel

Phone Number: 847845847

Add

Search by Name

Search by Phone