C# - OOP (object oriented programming)

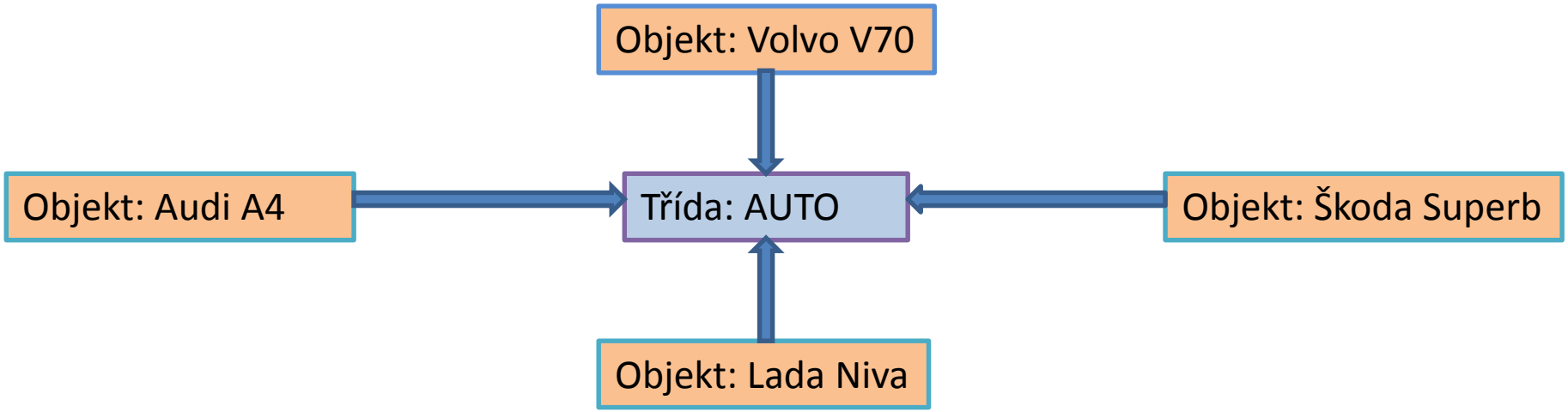
Centrum pro virtuální a moderní metody a formy vzdělávání na Obchodní akademii T.G. Masaryka, Kostelec nad Orlicí

OOP – proč?

- Velmi využívaný přístup k programování
- Využívá přirozené vnímání reálného světa
  - V OOP vytváříme objekty, které reprezentují entity reálného světa (auto, osoba, …)
- Výhody:
  - Přehlednost
  - Znovu-použitelnost kódu
  - Jednodušší správa aplikací

Třída a instance

- Každý objekt v programování může být vnímán jako abstrakce objektu reálného světa
- Třída je pak jakýsi popis (šablona) tohoto objektu
- Instance je pak každý konkrétní objekt (výskyt třídy)



Každá třída může obsahovat:

metody: - operace které může provádět

- např.: vypočítat průměrnou spotřebu, rychlost, nastartovat …

atributy: - představují stavy, vlastnosti třídy

- např.: barva, hmotnost, výkon, …

Zapouzdření objektů (encapsulating)

- Jedna ze základních vlatností objektově orientovaného přístupu
- Metody a atributy objektu mohou být před okolím objektu skryty a nebo slouží pro komunikaci s okolím pak se nazývají ***rozhraním*** objektu
- Zapouzdření je důležitou součástí bezpečnosti a stability tvořených aplikací
- Ten kdo využívá otestovaný a funkční objekt ve své aplikaci se nemusí starat o to, jak je funkčnost tohoto objektu implementována (Console.Writeline,….)

OOP prakticky – Třídy(Cíle)

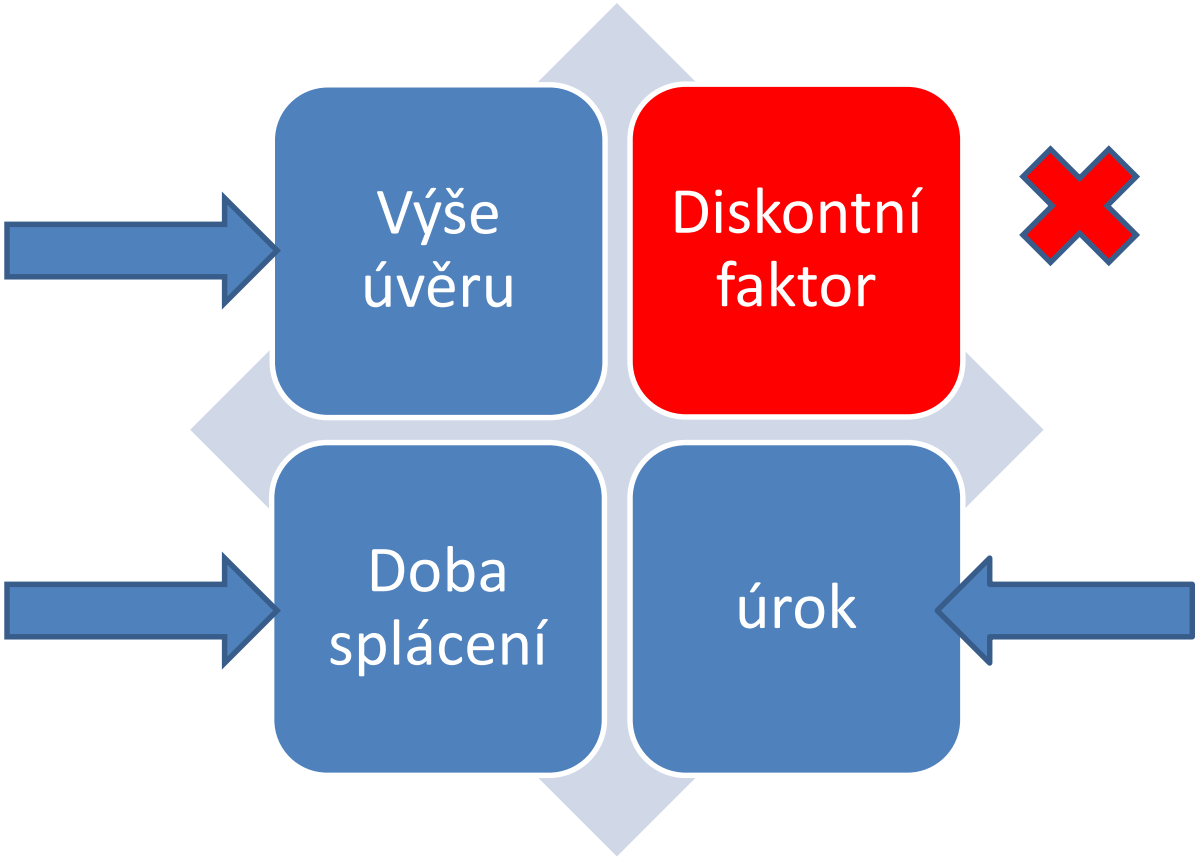
- Implementace konstruktorů
- Porozumění rozdílu mezi instancí a statickými členy
- Porozumění destruktorů
- Seznámení se s použitím členů tříd

OOP prakticky – Zapouzdření (Cíle)

- Porozumět principu zapouzdření OOP
- Porozumět dostupným modifikátorům datových členů
- Ochrana stavu objektu přes jeho vlastnosti
- Nastavení přístupu k metodám objektů
- Modifikace typů pro kompletaci zapouzdření

Co je zapouzdření ?

- V OOP vytváříme objekty, které mají stavy a chování



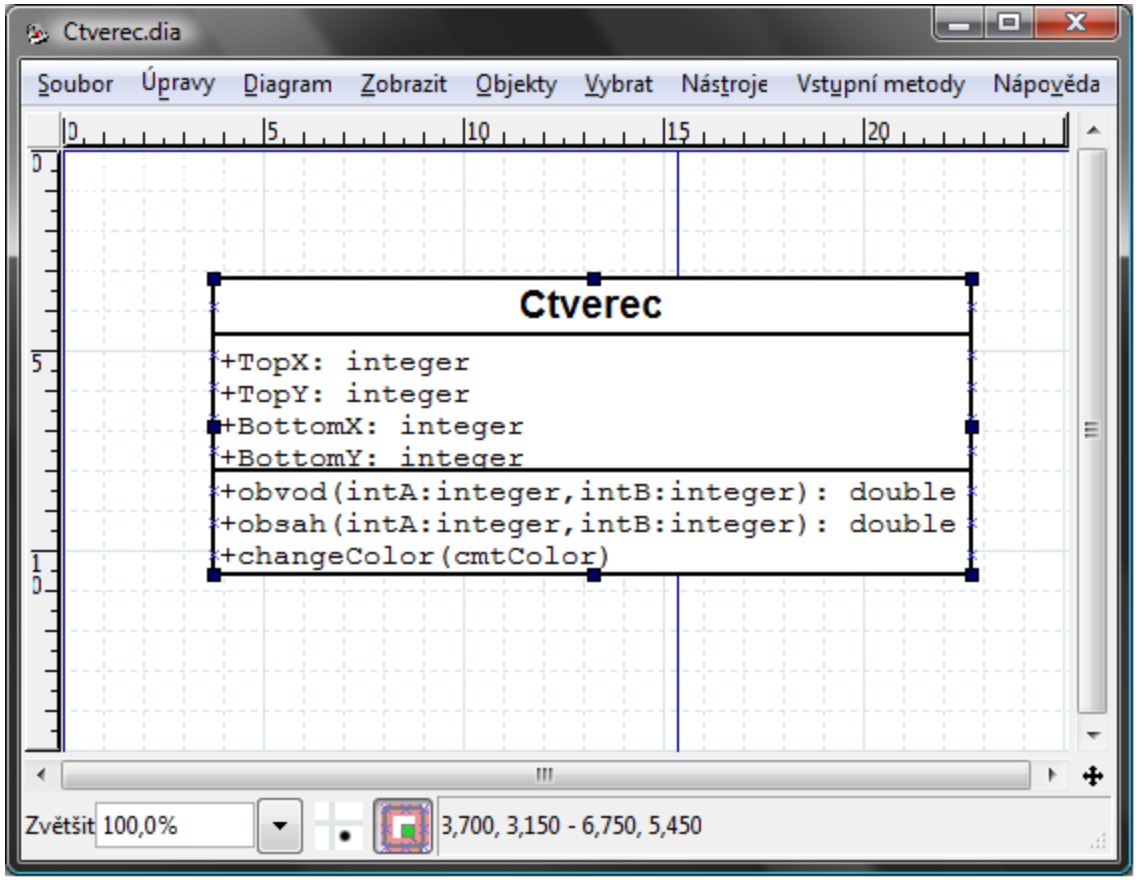
Specifikátory přístupu

- K definici viditelnosti atributů a metod vně objektů a tedy k definici rozhraní objektů se používají specifikátory:
  - ***Private*** – člen je přístupný pouze uvnitř třídy
  - ***Public*** – člen je viditelný z jiných tříd
  - ***Protected*** – člen je přístupný pouze z potomků třídy

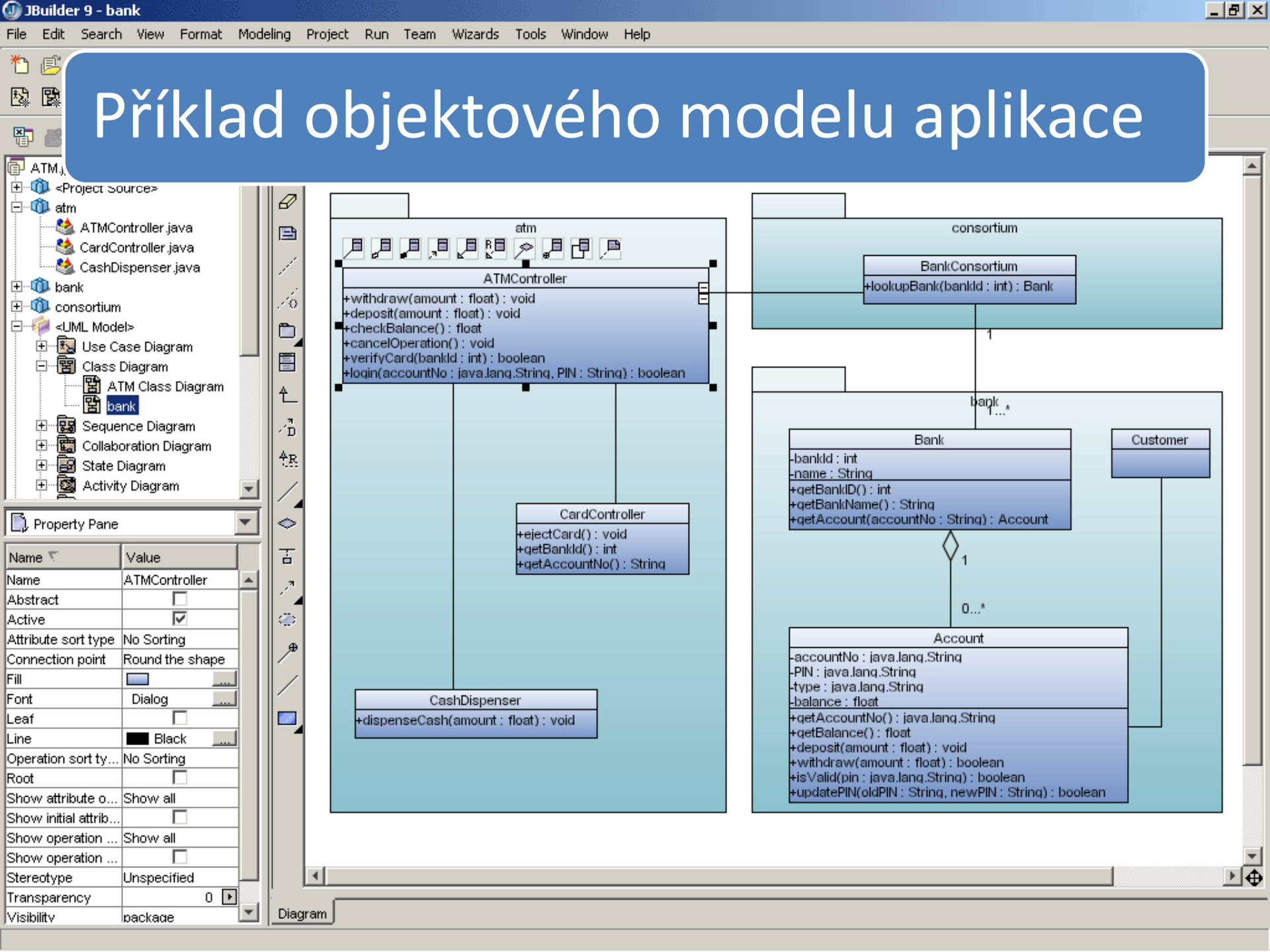
- class access video presentation

UML

- Návrh v UML (unified modelling language) – jazyk pro vytváření objektových konceptů



Příklad objektového modelu aplikace



Step by step vytvoření třídy OOP

- Třída je definována klíčovým slovem class, např.:
  - public class Person { } //public je přístupový modifikátor , který indikuje, že nejsou žádná omezení k přístupu k této třídě
- Třída je zatím prázdná, přidáme k ní nějaké atributy (name, age, gender)

```
public class Person {
  private string name;
  private int age;
  private string gender;
}
```

- Atributy uchovávají data jednotlivých objektů, ale je tu jeden problém - klíčová slova private u jednotlivých atributů znamenají, že k těmto proměnným nemůžeme přistupovat přímo tzn. takto: `age=32;`

- Proč jsou tyto atributy private ? Jak k nim můžeme přistupovat a využít je k uložení dat ?
  - Můžeme je udělat public, ale z principu OOP je vhodné mít atributy objektů private
  - K řešení vede využití tzv. Properties
  - (pravidlem pro použití názvů atributů a properties je použití malých písmen u atributů a velkých prvních písmen pro názvy odpovídajících properties)

```
public class Person {
private string name;
private int age;
private string gender;

public string Name {
get
{
return name;
}
set
{
name=value;
}
}
```

- Property „*Name*“ je public a tedy k ní může být přistupováno z naší aplikace
- *Set* blok je použit k uložení hodnoty do atributu
- *Get* blok vrací hodnotu uloženou v atributu

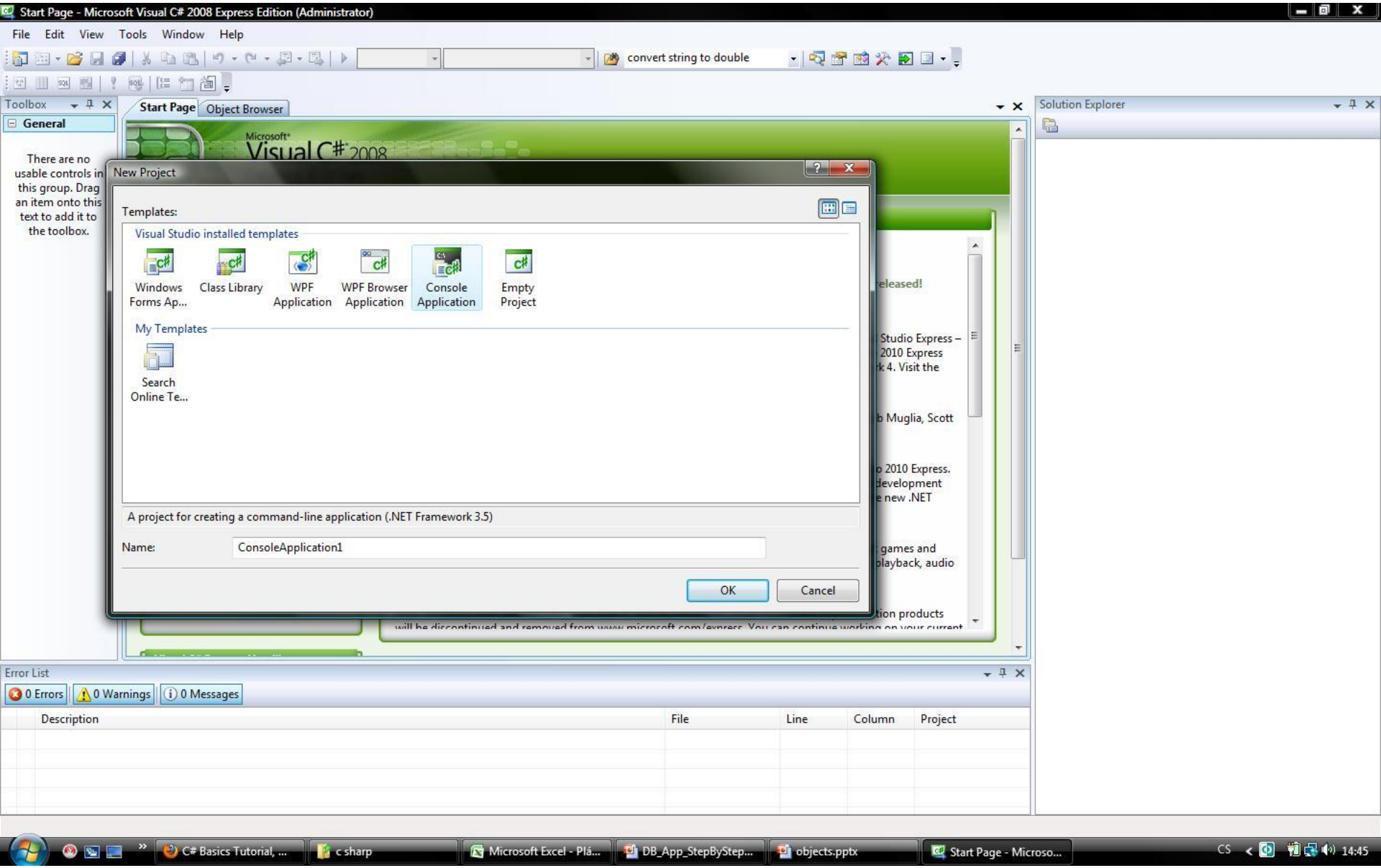
```
public class Person {
private string name;
 private int age;
 private string gender;

public string Name {
get { return name; }
set { name=value; }
}
public string Gender {
get { return gender; }
set { gender=value; }
}
public int Age {
get { return age; }
set { age=value; }
}
}
```

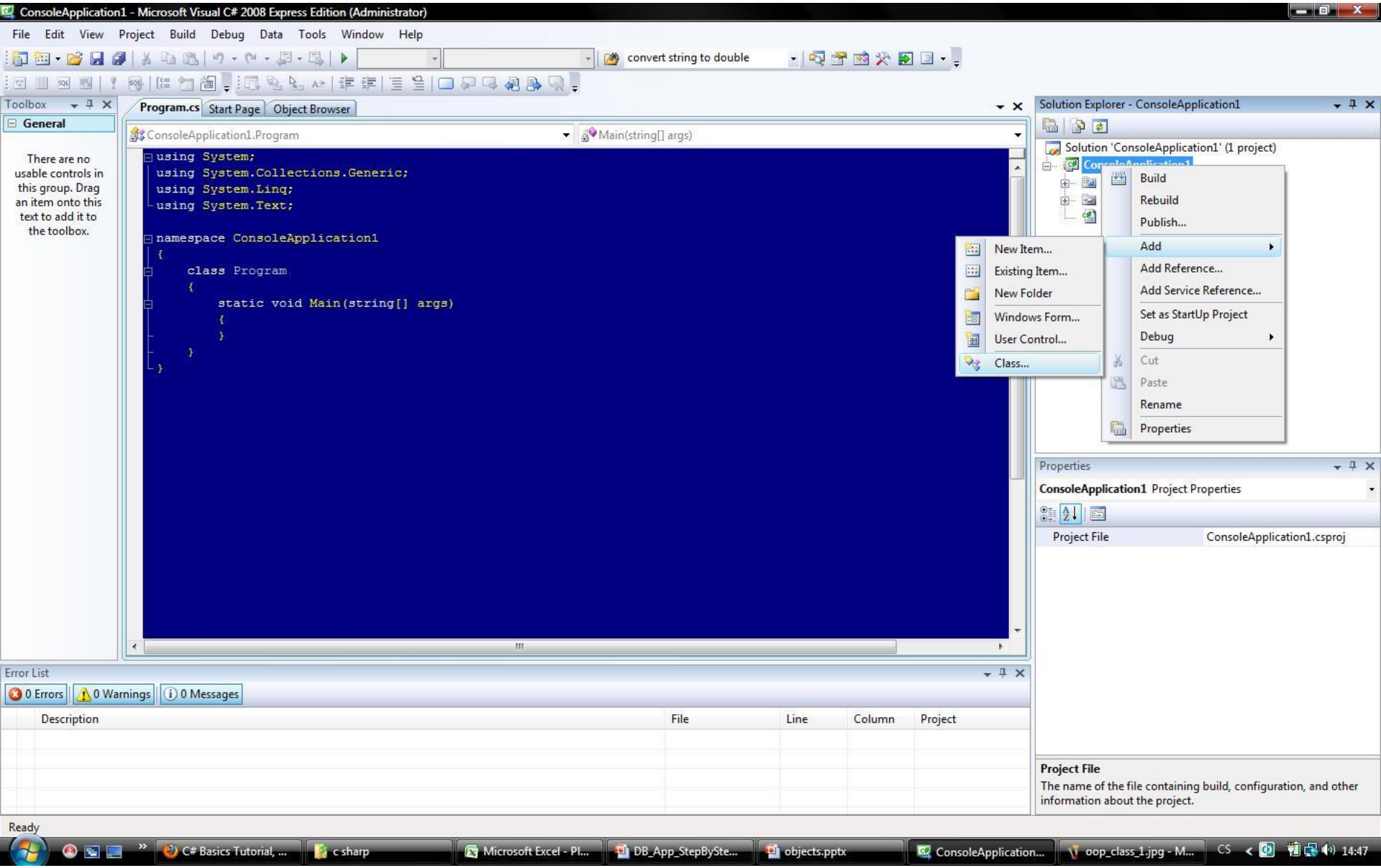
- Nyní má naše třída tři atributy a tři properties umožňující přístup k atributům třídy

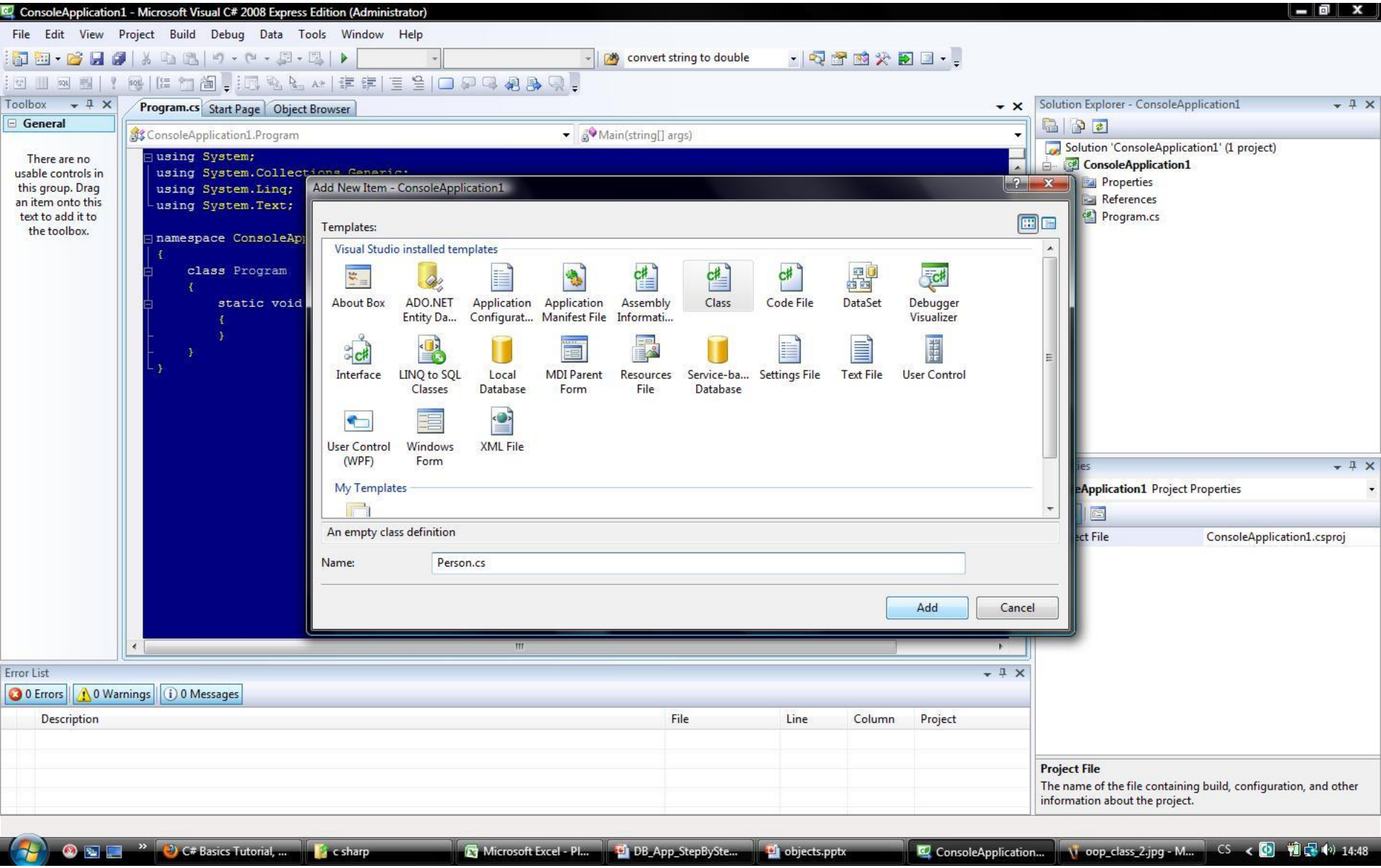
Využití vytvořené třídy v aplikaci – spusťte C# Express edt.

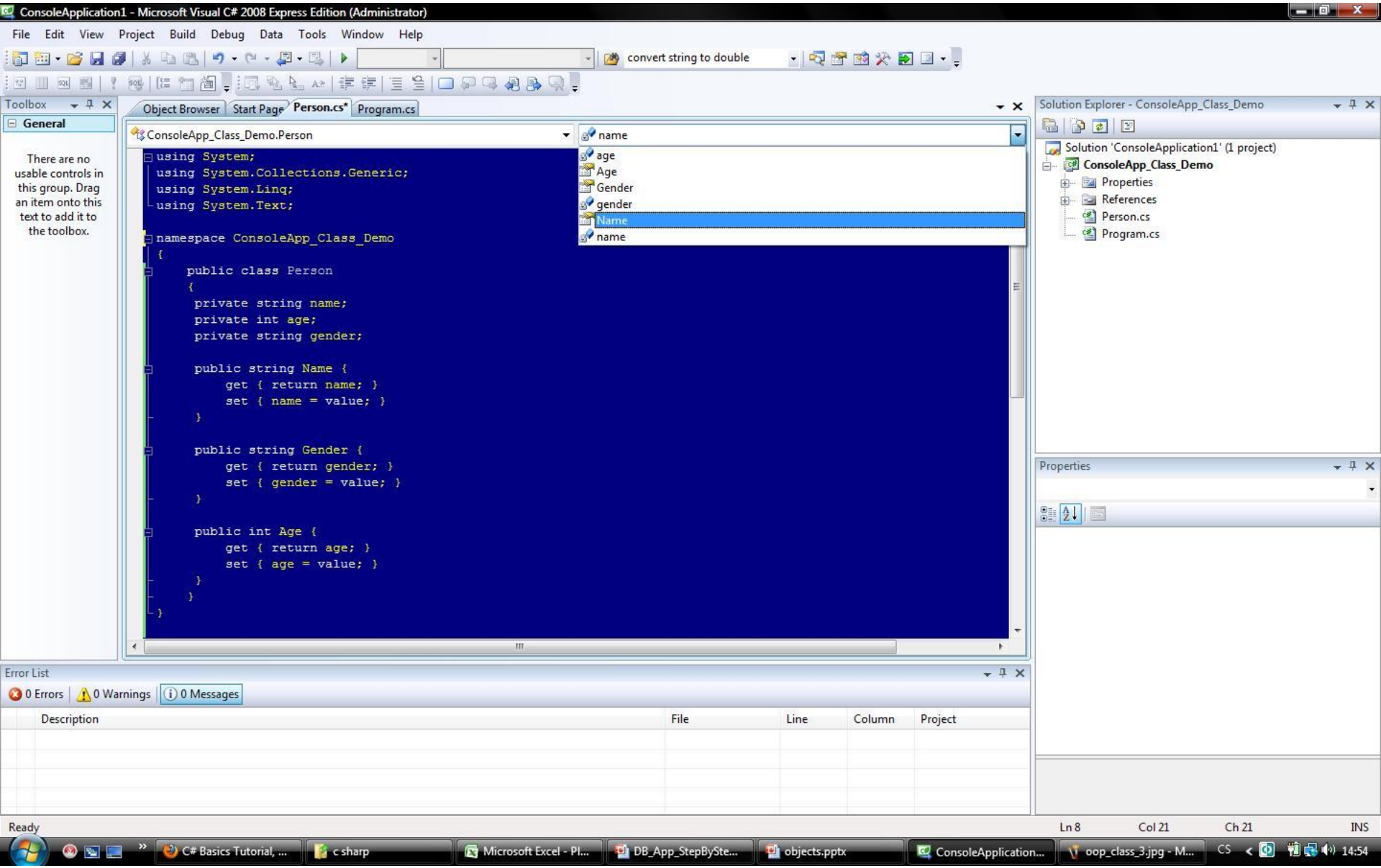
Jiný zápis zabezpečení argumentů
tento zápis není doporučen pro 100% zapouzdření a zabezpečení properties objektu (public)

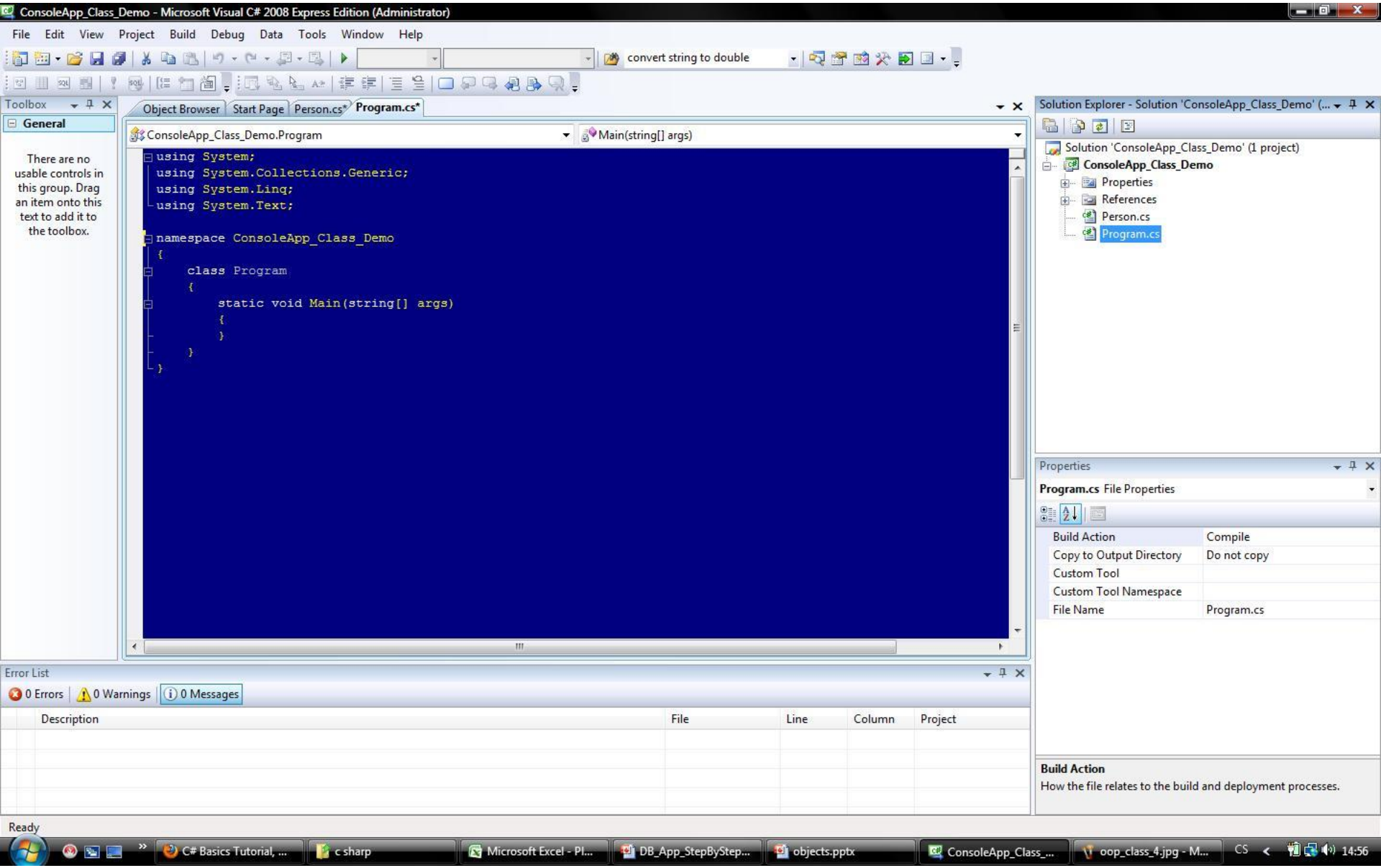
```
public class Person
{
    public string name {get;private set;} //argument můžeme z vnějšku pouze číst
    public int age {get; set;} //argument je možno z vnějšku číst i nastavit
    public string gender {private get;set;} //argument je možno z vnějšku pouze nastavit

    public Person(string _name, int _age, string _gender)
    {
        this.name = _name;
        this.age = _age;
        this.gender = _gender;
    }

}                    //private set, private get se mohou ze zápisu vynechat

class Program
{
    static void Main(string[] args)
    {
        Person osoba = new Person("Pepa",15,"muž");
        Console.WriteLine(osoba.name); //OK
        osoba.name = "Jirka"; // nelze
        Console.WriteLine(osoba.age); //OK
        osoba.age = 10; // OK
        Console.WriteLine(osoba.gender); //nelze
        osoba.gender = "hermafrodit"; // OK
    }
}
```

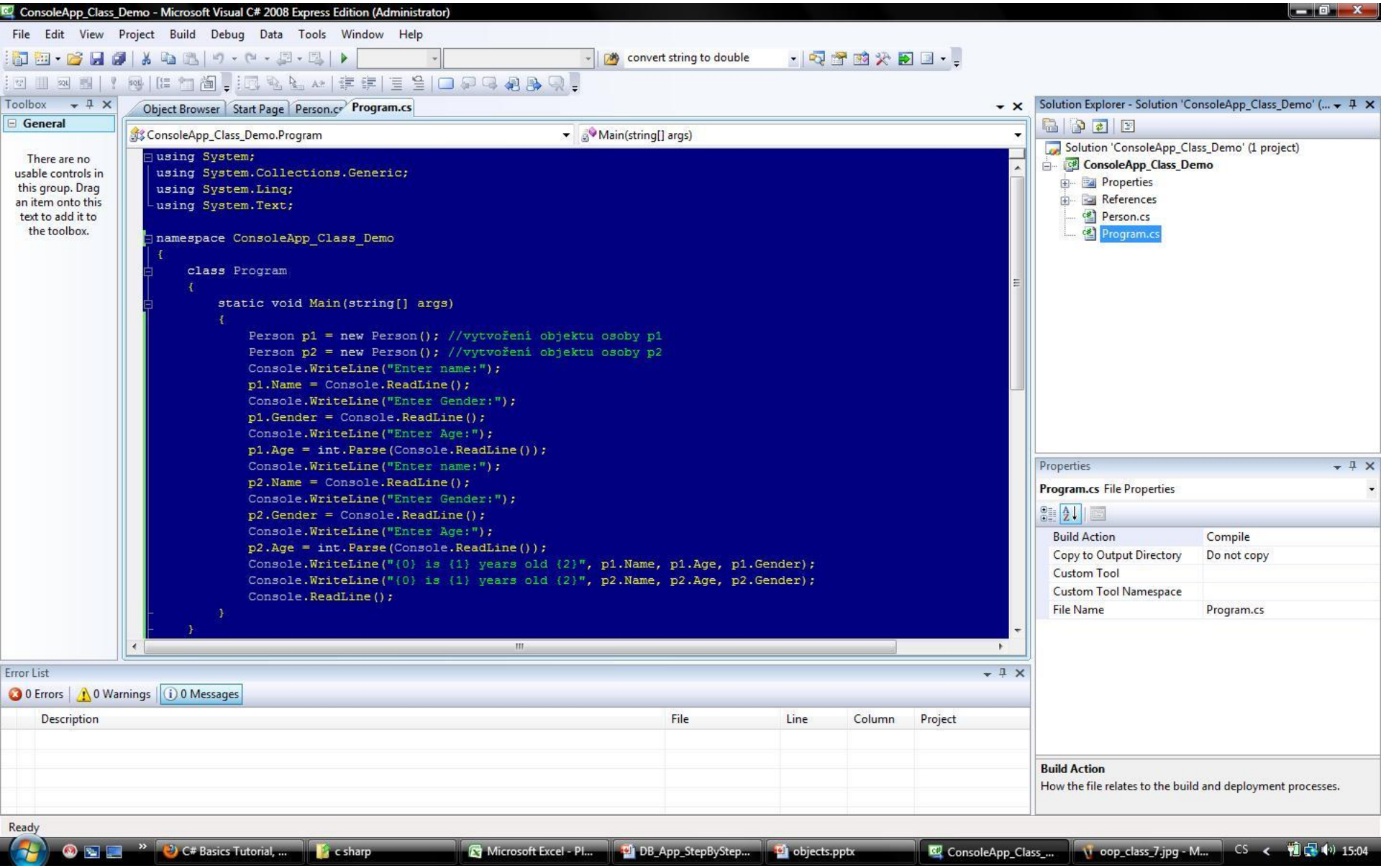

Start Page - Microsoft Visual C# 2008 Express Edition (Administrator)
File Edit View Tools Window Help
convert string to double
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Start Page
Object Browser
Microsoft Visual C# 2008
New Project
Templates:
Visual Studio installed templates
Windows Forms Ap...
Class Library
WPF Application
WPF Browser Application
Console Application
Empty Project
My Templates
Search Online Te...
A project for creating a command-line application (.NET Framework 3.5)
Name:
ConsoleApplication1
OK
Cancel
Solution Explorer
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
C# Basics Tutorial, ...
c sharp
Microsoft Excel - Plá...
DB_App_StepByStep...
objects.pptx
Start Page - Microso...
CS
14:45

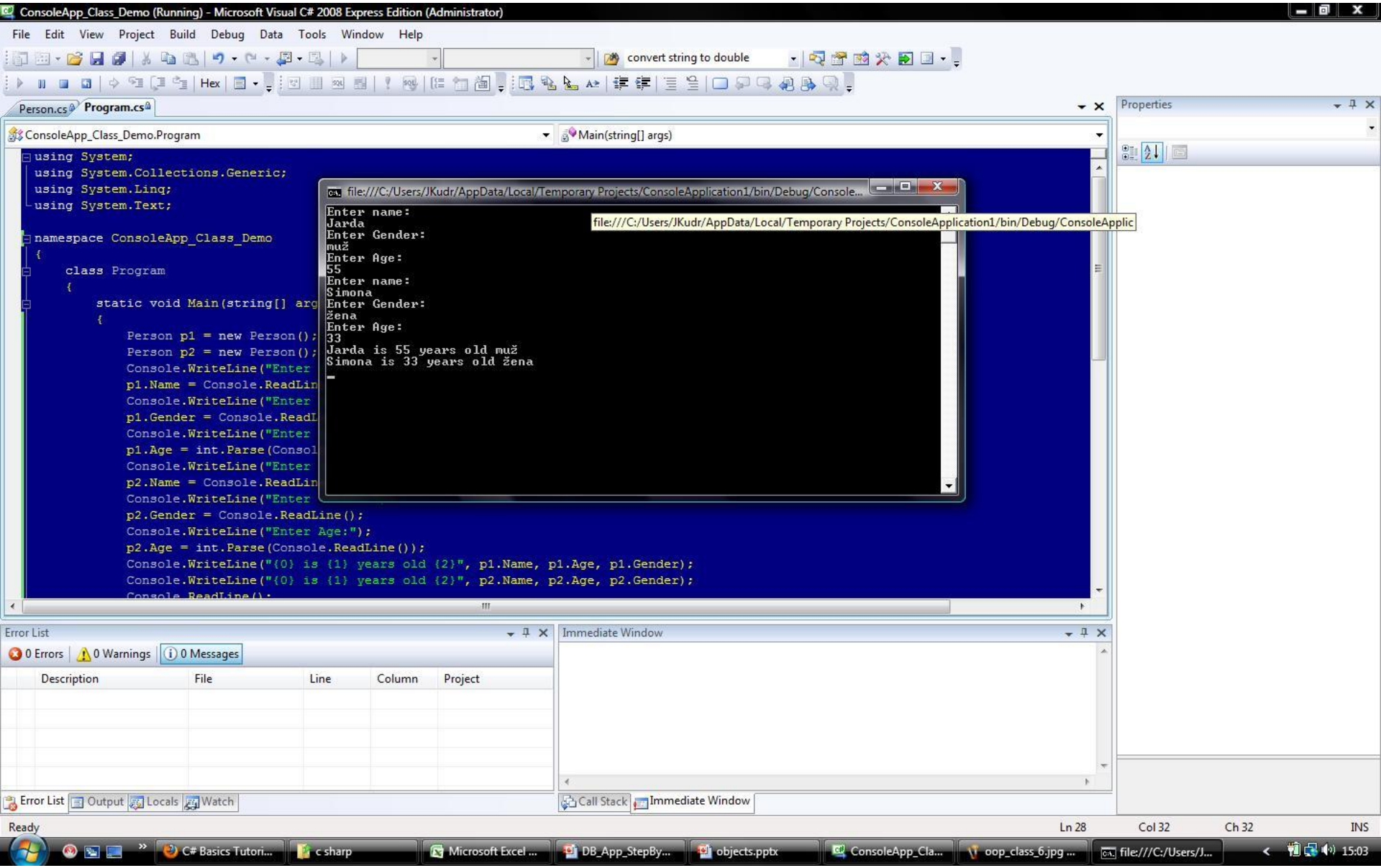

ConsoleApplication1 - Microsoft Visual C# 2008 Express Edition (Administrator)
File Edit View Project Build Debug Data Tools Window Help
convert string to double
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Program.cs Start Page Object Browser
ConsoleApplication1.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
}
}
}
Solution Explorer - ConsoleApplication1
Solution 'ConsoleApplication1' (1 project)
ConsoleApplication1
Build
Rebuild
Publish...
Add
Add Reference...
Add Service Reference...
Set as StartUp Project
Debug
Cut
Paste
Rename
Properties
New Item...
Existing Item...
New Folder
Windows Form...
User Control...
Class...
Properties
ConsoleApplication1 Project Properties
Project File
ConsoleApplication1.csproj
Project File
The name of the file containing build, configuration, and other information about the project.
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Ready
C# Basics Tutorial, ...
c sharp
Microsoft Excel - Pl...
DB_App_StepBySte...
objects.pptx
ConsoleApplication...
oop_class_1.jpg - M...
CS
14:47


ConsoleApplication1 - Microsoft Visual C# 2008 Express Edition (Administrator)
File Edit View Project Build Debug Data Tools Window Help
convert string to double
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Program.cs Start Page Object Browser
ConsoleApplication1.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApp
{
class Program
{
static void
{
}
}
}
Add New Item - ConsoleApplication1
Templates:
Visual Studio installed templates
About Box
ADO.NET Entity Da...
Application Configurat...
Application Manifest File
Assembly Informati...
Class
Code File
DataSet
Debugger Visualizer
Interface
LINQ to SQL Classes
Local Database
MDI Parent Form
Resources File
Service-ba... Database
Settings File
Text File
User Control
User Control (WPF)
Windows Form
XML File
My Templates
An empty class definition
Name:
Person.cs
Add
Cancel
Solution Explorer - ConsoleApplication1
Solution 'ConsoleApplication1' (1 project)
ConsoleApplication1
Properties
References
Program.cs
ConsoleApplication1 Project Properties
ConsoleApplication1.csproj
Project File
The name of the file containing build, configuration, and other information about the project.
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
C# Basics Tutorial, ...
c sharp
Microsoft Excel - Pl...
DB_App_StepByStep...
objects.pptx
ConsoleApplication...
oop_class_2.jpg - M...
CS
14:48


ConsoleApplication1 - Microsoft Visual C# 2008 Express Edition (Administrator)
File Edit View Project Build Debug Data Tools Window Help
convert string to double
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Object Browser
Start Page
Person.cs*
Program.cs
ConsoleApp_Class_Demo.Person
name
age
Age
Gender
gender
Name
name
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApp_Class_Demo
{
public class Person
{
private string name;
private int age;
private string gender;
public string Name {
get { return name; }
set { name = value; }
}
public string Gender {
get { return gender; }
set { gender = value; }
}
public int Age {
get { return age; }
set { age = value; }
}
}
}
Solution Explorer - ConsoleApp_Class_Demo
Solution 'ConsoleApplication1' (1 project)
ConsoleApp_Class_Demo
Properties
References
Person.cs
Program.cs
Properties
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Ready
Ln 8
Col 21
Ch 21
INS
C# Basics Tutorial, ...
c sharp
Microsoft Excel - Pl...
DB_App_StepBySte...
objects.pptx
ConsoleApplication...
oop_class_3.jpg - M...
CS
14:54


ConsoleApp_Class_Demo - Microsoft Visual C# 2008 Express Edition (Administrator)
File Edit View Project Build Debug Data Tools Window Help
convert string to double
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Object Browser
Start Page
Person.cs*
Program.cs*
ConsoleApp_Class_Demo.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApp_Class_Demo
{
class Program
{
static void Main(string[] args)
{
}
}
}
Solution Explorer - Solution 'ConsoleApp_Class_Demo' (...
Solution 'ConsoleApp_Class_Demo' (1 project)
ConsoleApp_Class_Demo
Properties
References
Person.cs
Program.cs
Properties
Program.cs File Properties
Build Action
Compile
Copy to Output Directory
Do not copy
Custom Tool
Custom Tool Namespace
File Name
Program.cs
Build Action
How the file relates to the build and deployment processes.
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Ready
C# Basics Tutorial, ...
c sharp
Microsoft Excel - Pl...
DB_App_StepByStep...
objects.pptx
ConsoleApp_Class_...
oop_class_4.jpg - M...
CS
14:56


ConsoleApp_Class_Demo - Microsoft Visual C# 2008 Express Edition (Administrator)
File Edit View Project Build Debug Data Tools Window Help
convert string to double
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Object Browser
Start Page
Person.cs
Program.cs
ConsoleApp_Class_Demo.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApp_Class_Demo
{
class Program
{
static void Main(string[] args)
{
Person p1 = new Person(); //vytvoření objektu osoby p1
Person p2 = new Person(); //vytvoření objektu osoby p2
Console.WriteLine("Enter name:");
p1.Name = Console.ReadLine();
Console.WriteLine("Enter Gender:");
p1.Gender = Console.ReadLine();
Console.WriteLine("Enter Age:");
p1.Age = int.Parse(Console.ReadLine());
Console.WriteLine("Enter name:");
p2.Name = Console.ReadLine();
Console.WriteLine("Enter Gender:");
p2.Gender = Console.ReadLine();
Console.WriteLine("Enter Age:");
p2.Age = int.Parse(Console.ReadLine());
Console.WriteLine("{0} is {1} years old {2}", p1.Name, p1.Age, p1.Gender);
Console.WriteLine("{0} is {1} years old {2}", p2.Name, p2.Age, p2.Gender);
Console.ReadLine();
}
}
Solution Explorer - Solution 'ConsoleApp_Class_Demo' (...
Solution 'ConsoleApp_Class_Demo' (1 project)
ConsoleApp_Class_Demo
Properties
References
Person.cs
Program.cs
Properties
Program.cs File Properties
Build Action Compile
Copy to Output Directory Do not copy
Custom Tool
Custom Tool Namespace
File Name Program.cs
Build Action
How the file relates to the build and deployment processes.
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Ready
C# Basics Tutorial, ...
c sharp
Microsoft Excel - Pl...
DB_App_StepByStep...
objects.pptx
ConsoleApp_Class_...
oop_class_7.jpg - M...
CS
15:04


ConsoleApp_Class_Demo (Running) - Microsoft Visual C# 2008 Express Edition (Administrator)
File Edit View Project Build Debug Data Tools Window Help
convert string to double
Person.cs Program.cs
ConsoleApp_Class_Demo.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApp_Class_Demo
{
class Program
{
static void Main(string[] arg
{
Person p1 = new Person();
Person p2 = new Person();
Console.WriteLine("Enter
p1.Name = Console.ReadLin
Console.WriteLine("Enter
p1.Gender = Console.ReadL
Console.WriteLine("Enter
p1.Age = int.Parse(Consol
Console.WriteLine("Enter
p2.Name = Console.ReadLin
Console.WriteLine("Enter
p2.Gender = Console.ReadLine();
Console.WriteLine("Enter Age:");
p2.Age = int.Parse(Console.ReadLine());
Console.WriteLine("{0} is {1} years old {2}", p1.Name, p1.Age, p1.Gender);
Console.WriteLine("{0} is {1} years old {2}", p2.Name, p2.Age, p2.Gender);
Console.ReadLine();
file:///C:/Users/JKudr/AppData/Local/Temporary Projects/ConsoleApplication1/bin/Debug/Console...
Enter name:
Jarda
Enter Gender:
muž
Enter Age:
55
Enter name:
Simona
Enter Gender:
žena
Enter Age:
33
Jarda is 55 years old muž
Simona is 33 years old žena
file:///C:/Users/JKudr/AppData/Local/Temporary Projects/ConsoleApplication1/bin/Debug/ConsoleApplic
Properties
Error List
0 Errors 0 Warnings 0 Messages
Description File Line Column Project
Immediate Window
Error List Output Locals Watch
Call Stack Immediate Window
Ready
Ln 28 Col 32 Ch 32 INS
C# Basics Tutori... c sharp Microsoft Excel ... DB_App_StepBy... objects.pptx ConsoleApp_Cla... oop_class_6.jpg ... file:///C:/Users/J...
15:03

Ukázka vytvoření třídy a využití metod třídy

```
class Calculate {
    //deklarace atributů
    //definice properties
    //methody
    public int CalculateSquare(int number) {
        int s;
        s = number*number;
        return s;
    }
}
```

```
int square;  //použití v aplikaci
Calculate c = new Calculate();
square= c.CalculateSquare(4);
```

Úkol

- Vytvořte objekt CalculateCube, který bude umět
  - Spočítat obsah krychle
  - Spočítat objem krychle
  - Vrátit poloměr opsané kulové plochy
  - Vrátit poloměr vepsané kulové plochy
  - Daný objekt využijte v konzolové aplikaci
  - Strana krychle bude privátní atribut, přístup k němu bude přes veřejnou property viz. Minulý příklad

Implementace tříd v C#

- ***Třídy*** jsou v C# deklarovány pomocí klíčového slova ***class***
- ***Konstruktor***
  - slouží k počáteční inicializaci atributů objektu
  - je volán automaticky pokaždé, když je vytvořena instance objektu
  - Nemá návratovou hodnotu
  - Má vždy stejný název jako třída
- ***Destruktor***
  - slouží k operacím souvisejícím s koncem existence objektu
  - Nemá návratovou hodnotu ani parametry
  - Normálně jsou volány v okamžiku, kdy ***Garbage Collector*** odstraňuje objekt y paměti
- Pokud není konstruktor a detruktor definován je použit implicitní
- ***Atributy a metody***:

  Dva specifikátory přístupu
  - ***Get*** – slouží k přístupu a čtení hodnoty atributu objektu
  - ***Set*** – slouží k nastavení atributu objektu

```
// Namespace deklarace
using System;

// deklaruji třídu Output která má vnitřní členy proměnou string a metodu printString
class OutputClass
{
   string myString;

   // Constructor naší třídy – naplní vnitřní proměnou myString hodnotou parametru
   public OutputClass(string inputString)
   {
      myString = inputString;
   }

   // Instance Method vypíše na konzoli obsah vnitřní proměnné
   public void printString()
   {
      Console.WriteLine("{0}", myString);
   }

   // Destructor
   ~OutputClass()
   {
  //kód související s ukončením činnosti objektu
   }
}

// Program start class
class ExampleClass
{
   // Main – funkce kde začíná program
   public static void Main()
   {
      // vytvoříme nový objekt (instanci) třídy OutputClass
      OutputClass outCl = new OutputClass(„Toto píše objekt třídy OutputClass.");

      // Zavolání metody námi vytvořeného objektu
      outCl.printString();
   }
}
```

Static a non-static metody

- *Dva typy metod:*
- ***Non static:***
  - Patří konkrétní instanci objektu
  - Příklad:

```
OutputClass oc1 = new OutputClass("OutputClass1");
OutputClass oc2 = new OutputClass("OutputClass2");
```

    Obě tyto instance třídy **OutputClass** jsou oddělené a samostatné, každý z těchto objektů má vlastní metodu **printString**
  - Volání této metody **–** přes instanci konkrétního objektu:

```
oc2.printString ();
oc1.printString();
```

- **Statické metody:**
  - Na rozdíl od nestatických nejsou spjaty s instancí objektu ale s třídou
  - Počet výskytů těchto členů nebo metod nezávisí na počtu výskytů vytvořených objektů, ale je vždy pouze jeden
  - Přistupujeme k nim přes název třídy ne instance objektu
  - Předpokládejme v našem příkladu následující definici:

```
public static void staticPrinter()
   {
      Console.WriteLine("There is only one of me.");
   }
```

  - Volání této metody

```
OutputClass.staticPrinter();
```

  - **Statické konstruktory** – volaný ještě před vytvořením instance objektu, před použitím statického členu třídy a před statickým konstruktorem dědice třídy (jsou volány pouze jednou)

Úkol

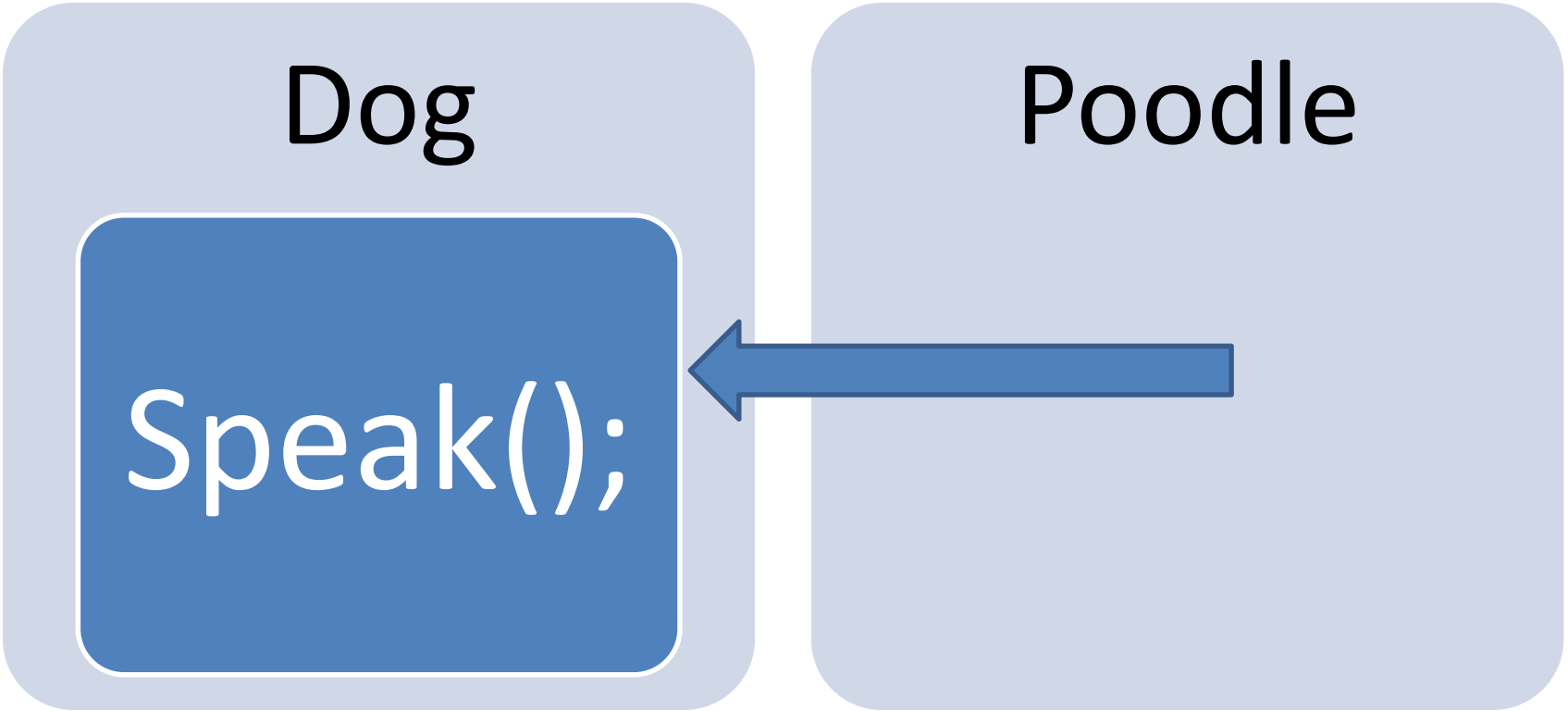
- Vytvořte vlastní třídu podle vzoru v prezentaci. V programu vytvořte 3 různé instance deklarované třídy a použijte jejich nestatické metody. Pak příklad upravte a metodu pro výpis změňte na statickou. Upravte funkčnost metody Main a sledujte změnu ve výstupu programu.

Inheritance – dědění v OOP

- Často se stane, že Vaše aplikace využívá více objektů, které jsou velice podobné a liší se pouze malým počtem ***properties*** nebo metod.

- **Dědičnost** (**inheritance**) třídy C1 z třídy C2 je nástroj, s jehož pomocí třída C2 implicitně definuje (tzv. **dědí**) všechny datové a funkční členy třídy C1, jako kdyby byly definovány přímo ve třídě C2.
  Třídu C1 nazýváme **základní** (**nadřízená**, **nadřazená**, **base class**) třída, třídu C2 **odvozená** (**podřízená**, **podřazená**, **derivovaná**, **derived class**) třída.

Příklad Dog



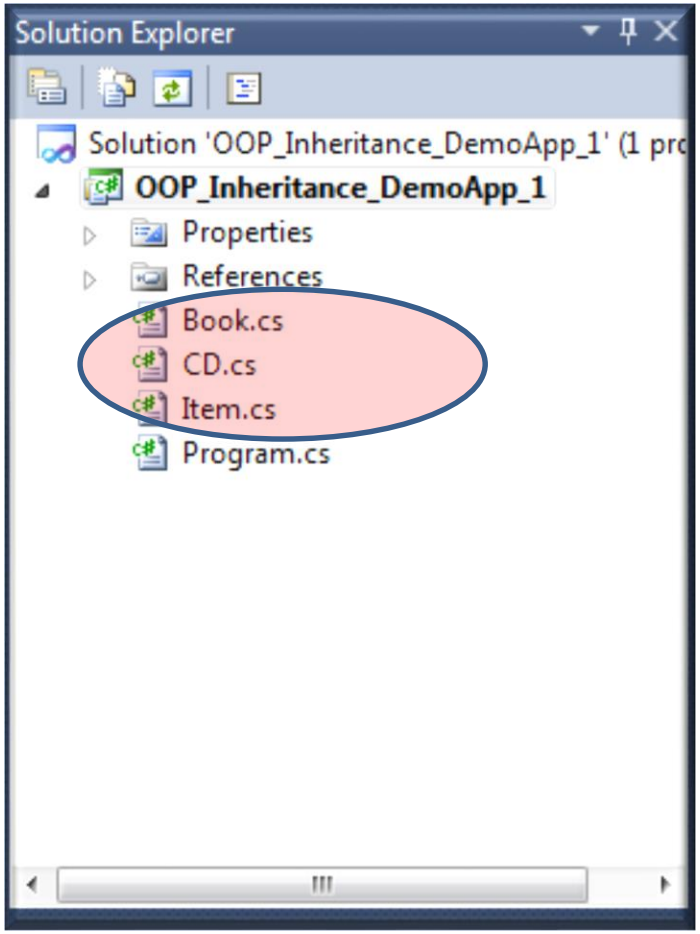
- inheritance video explanation

Step by Step Inheritance Application

- Vytvořte aplikaci, která bude obsahovat tři druhy objektů s následujícími properties:
- **Item:**
  - *Name*
  - *Price*
- **Book :**
  - *Name*
  - *ISBN*
  - *Price*
- **CD:**
  - *Name*
  - *Artist*
  - *Price*
- Definujte objekty Book a CD tak aby dědili po objektu Item.

Step 1

- New Project: OOP_Inheritance_DemoApp_1
- Type: Console Application
- Add 3 classes: Item, Book, CD



Step 2

- Přidejte properties objektu Item:
  - Name
  - Price
- Definujte konstruktor a prázdný konstruktor ke korektnímu fungování dědičnosti

CD.cs | Book.cs | Item.cs | Program.cs

ConsoleApplication1.Item | Item(string _name, double _price)

```
{
    class Item
    {
        //protected znamená, že mohou být přístupné v objektech, které dědí po objektu Item
        protected string name; //jméno
        protected double price; //cena

        //tento konstruktor bez parametrů je nutný kvůli dědičnosti !!!!
        public Item()
        {}

        //konstruktor
        public Item(string _name, double _price)
        {
            this.name = _name;
            this.price = _price;
        }

        //předefinuji si metodu ToString na objektu Item tak, že vrátí název "knihy : cena"
        public override string ToString()
        {
            return name + " : " + price.ToString ();
        }
    }
}
```

100 %

Step 3

- Napište kód objektu Book, který bude dědit po objektu Item

CD.cs | Book.cs* | Item.cs | Program.cs

ConsoleApplication1.Book — Book(string _name, double _price, string _ISBN)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    //touto konstrukcí říkám, že objekt Book dědí od nadřazeného objektu Item
    class Book : Item
    {
        //property name a price objekt dědí, nemusím je definovat
        private string ISBN;

        //v konstruktoru mohu nastavovat i děděné property
        public Book(string _name, double _price, string _ISBN)
        {
            this.name = _name;
            this.price = _price;
            this.ISBN = _ISBN;
        }
    }
}
```

Step 4

- Napište kód objektu CD, který bude dědit po objektu Item

CD.cs | Book.cs* | Item.cs | Program.cs

ConsoleApplication1.CD — artist

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    //objekt CD dědí od objektu Item
    class CD :Item
    {
        //property name a price objekt dědí, nemusím je definovat
        private string artist;

        //v konstruktoru mohu nastavovat i děděné property
        public CD(string _name, double _price, string _artist)
        {
            this.name  = _name;
            this.price = _price;
            this.artist = _artist;
        }

    }
}
```

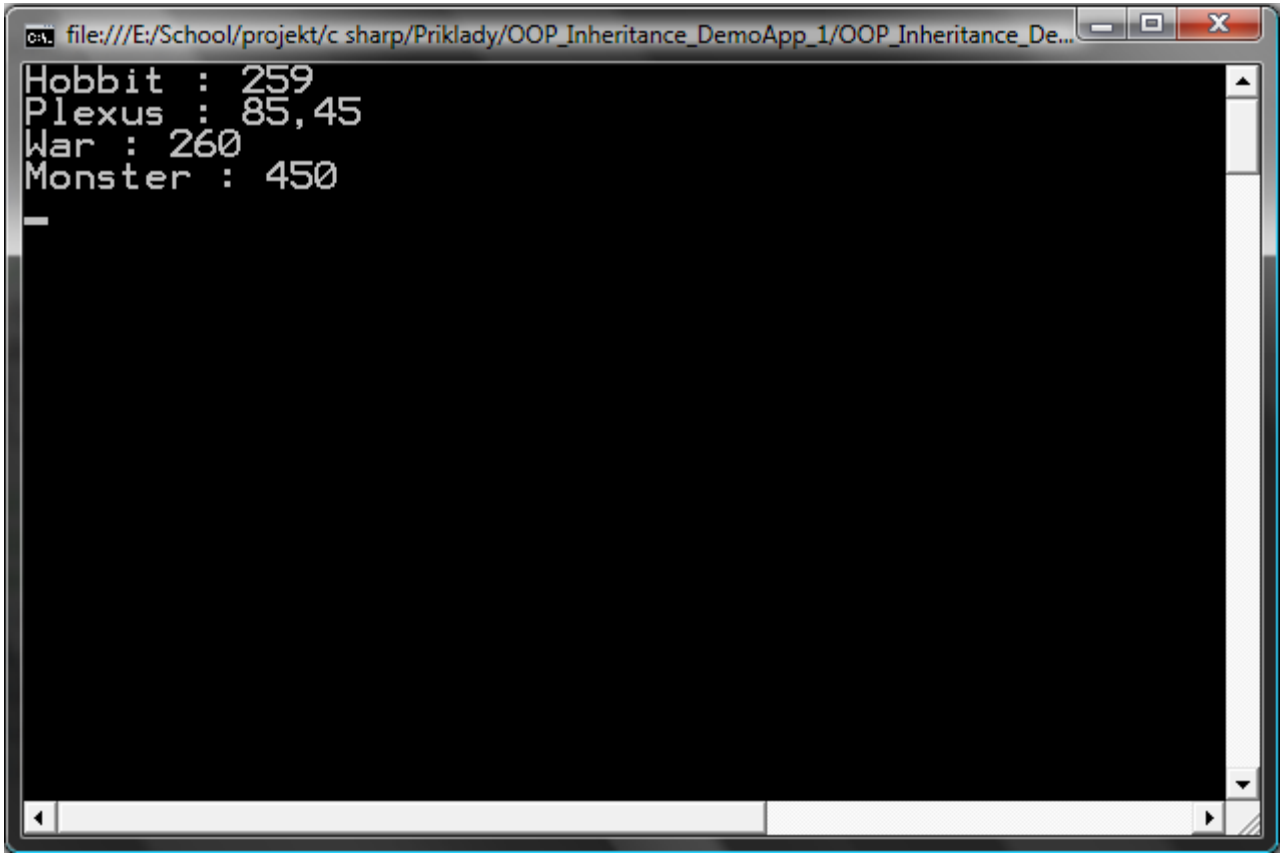
Step 5

- Nadefinujte konkrétní instance objektů Book a CD a vypište údaje o jejich ceně na konzoli (využijte přetíženou poděděnou metodu ToString objektu Item)

CD.cs | Book.cs | Item.cs | Program.cs

ConsoleApplication1.Program | Main(string[] args)

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            //vytvořím instance objektů Book a CD
            Book myBook_1 = new Book("Hobbit", 259, "EAI14552");
            Book myBook_2 = new Book("Plexus", 85.45, "DSA77898");
            CD myCD_1 = new CD("War", 260, "U2");
            CD myCD_2 = new CD("Monster",450,"REM");

            //vypíši údaje jednotlivých instancí objektů na konzoli
            //využiji přetížené metody ToString, kterou podědili od
            //nadřazeného objektu Item
            Console.WriteLine(myBook_1.ToString ());
            Console.WriteLine(myBook_2.ToString());
            Console.WriteLine(myCD_1.ToString());
            Console.WriteLine(myCD_2.ToString());
            Console.Read();
        }
    }
}
```


file:///E:/School/projekt/c sharp/Priklady/OOP_Inheritance_DemoApp_1/OOP_Inheritance_De...
Hobbit : 259
Plexus : 85,45
War : 260
Monster : 450

Závěr

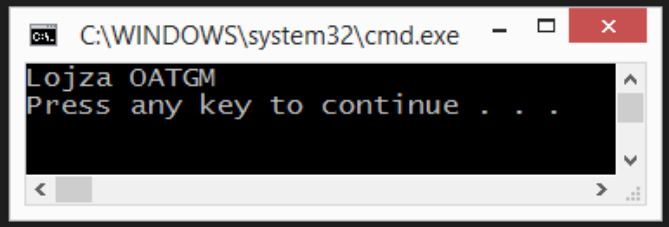
- Protrasujte pomocím ***breakpointů*** chod programu !!!
- Pokuste se změnit ***protected properties*** objektu ***Item*** na ***private*** a ověřte tak funkci zapouzdření objektu
- Zamyslete se nad výhodami dědění objektů v programování, co Vám umožňují ?

Virtuální metody override, new

- Pokud v děděných objektech ***chceme předefinovat kód*** (funkčnost) metody, kterou objekty dědí je nutné nadefinovat v base třídě tuto metodu jako ***virtuální***.
- Pokud předefinovaná metoda zachovává návratový typ a má stejný specifikátor přístupu jako metoda v base třídě jde o tzv. překrytí metody a tuto metodu označíme klíčovým slovem ***override***
- Pokud předefinovaná metoda má pouze stejný název jako metoda v base třídě, ale má jinou návratovou hodnotu nebo jiný specifikátor přístupu použijeme v metodě potomka klíčové slovo ***new***

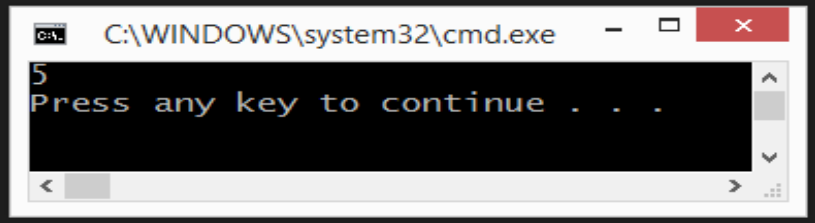
OVERRIDE

```
public class Person
{
    public string name;

    public Person()
    {
    }
    //tato metoda bude v dědicích přepsána
    public virtual string vypis()
    { return this.name; }

}

public class Student : Person
{
    public string school;

    public Student(string _name, string _school)
    {
        this.name  = _name;
        this.school = _school;
    }
    //přepisovaná metoda má stejný specifikátor přístupu a stejný návratový typ
    //jde o překrytí
    public override string vypis()
    {
        return this.name + " " + this.school;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Student lojza = new Student("Lojza", "OATGM");
        Console.WriteLine (lojza.vypis());
    }
}
```



new

```
public class Person
{
    public string name;

    public Person()
    {
    }
    //tato metoda bude v dědicích přepsána
    public virtual string vypis()
    { return this.name; }

}

public class Student : Person
{
    public string school;

    public Student(string _name, string _school)
    {
        this.name   = _name;
        this.school = _school;
    }
    //přepisovaná metoda má stejný specifikátor přístupu ale jiný návratový typ
    //jde o "novou" metodu
    new public int vypis()
    {
        return this.name.Length;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Student lojza = new Student("Lojza", "OATGM");
        Console.WriteLine (lojza.vypis().ToString ());
    }
}
```



A na co je to jako dobré ?

Vraťme se k našemu příkladu s knihami a CD.

- Otevřete v Moodle projekt

  Books_and_CD
- Postupujte dle pokynů učitele a vyzkoušejte obě varianty aplikace.
- Prodiskutujte výsledky aplikace a zamyslete se nad reálným použitím virtuálních metod v praxi
- Pro zájemce článek o virtuálních metodách s podrobným vysvětlením („co je za tím“) v jazyce C++ naleznete na tomto odkazu:

  Virtuální metody v C#

Úkol 1

- Pod vedením vyučujícího nasimulujte všechny možnosti virtuálních metod a volání jak z objektu vytvořeného přes bázovou třídu, tak i objektů vytvořených z potomků tříd.
- Všechny možnosti důkladně protrasujte a uvědomte si význam použití
- Pozn.: Tyto znalosti budou ověřeny v testu

Úkol 2

- Vytvořte vlastní aplikaci nákupní košík (pole deseti položek)
- **base class:**
  - ***Item*** *(barcode, price; Vypis())*
- ***derived classes:***
  - ***MP3*** *(memory, brand; Vypis())*
  - ***Mobile*** *(size, brand, OS; Vypis())*
- Po vytvoření košíku a naplnění ho konkrétními položkami jeho obsah vypište a napište celkovou sumu všech položek v košíku

- **Hlavní výhody dědičnosti**
- **Omezení duplicity kódu**
- společné rysy skupiny třídy sdílejí definici [a implementaci]
  - base class definuje společné rysy na jednom místě
  - odvozené třídy definují své specifické rysy
- **Stejné zacházení s více třídami**
- pro některé operace stačí rysy, které vykazuje společná nadtřída
- mechanizmus pro realizaci polymorfizmu

- pdf materiál o dědičnosti

Použité materiály:

Kniha: Programujeme profesionálně, nakladatelství WROX, autor: Jay Glynn,…
www.wikipedia.com

Seriály o programování v jazyce C# :

www.živě.cz
www.java2s.com
www.functionx.com
www.csharp-station.com
www.msdn.com
www.bytes.com
www.c-sharpcorner.com