**Note** Now that you better understand the role of class constructors, here is a nice short cut. The Visual Studio IDE provides the ctor code snippet. When you type “ctor” and press the Tab key twice, the IDE will automatically define a custom default constructor! You can then add custom parameters and implementation logic. Give it a try

C# supplies a this keyword that provides access to the current class instance. One possible use of the

this keyword is to resolve scope ambiguity, which can arise when an incoming parameter is named

identically to a data field of the class. Of course, ideally you would simply adopt a naming convention

that does not result in such ambiguity

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace constructor\_ex

{

class myclass

{

public int a;

public void mymeth(int a)

{

a = a; //after trying this example add --this.a

Console.WriteLine("{0}", a);

Console.WriteLine("{0}", this.a);

}

}

class Program

{

static void Main(string[] args)

{

myclass m = new myclass();

m.mymeth (5);

Console.WriteLine("{0}", m.a);

Console.ReadKey();

}

}

}

Although this code will compile just fine, Visual Studio will display a warning message informing

you that you have assigned a variable back to itself! and then print out the value of the name field. You might be surprised to find that the value of the name

field is 0