Utilize MyClass without assuming any prior knowledge

```
/*
C#: The Complete Reference
by Herbert Schildt

Publisher: Osborne/McGraw-Hill (March 8, 2002)
ISBN: 0072134852
*/

// Utilize MyClass without assuming any prior knowledge.

using System;
using System.Reflection;

public class ReflectAssemblyDemo1 {
public static void Main() {
int val;
Assembly asm = Assembly.LoadFrom(“MyClasses.exe”);

Type[] alltypes = asm.GetTypes();

Type t = alltypes[0]; // use first class found

Console.WriteLine(“Using: ” + t.Name);

ConstructorInfo[] ci = t.GetConstructors();

// Use first constructor found.
ParameterInfo[] cpi = ci[0].GetParameters();
object reflectOb;

if(cpi.Length > 0) {
object[] consargs = new object[cpi.Length];

// initialize args
for(int n=0; n < cpi.Length; n++) consargs[n] = 10 + n * 20;
```

```
// construct the object reflectOb = ci[0].Invoke(consargs); } else reflectOb = ci[0].Invoke(null); Console.WriteLine(" Invoking methods on reflectOb."); Console.WriteLine(); // Ignore inherited methods. MethodInfo[] mi = t.GetMethods(BindingFlags.DeclaredOnly | BindingFlags.Instance | BindingFlags.Public) ; // Invoke each method. foreach(MethodInfo m in mi) { Console.WriteLine("Calling {0} ", m.Name); // Get the parameters ParameterInfo[] pi = m.GetParameters(); // Execute methods. switch(pi.Length) { case 0: // no args if(m.ReturnType == typeof(int)) { val = (int) m.Invoke(reflectOb, null); Console.WriteLine("Result is " + val); } else if(m.ReturnType == typeof(void)) { m.Invoke(reflectOb, null); } break; case 1: // one arg if(pi[0].ParameterType == typeof(int)) { object[] args = new object[1]; args[0] = 14; if((bool) m.Invoke(reflectOb, args)) Console.WriteLine("14 is between x and y"); else Console.WriteLine("14 is not between x and y"); } break; case 2: // two args if((pi[0].ParameterType == typeof(int)) && (pi[1].ParameterType == typeof(int))) { object[] args = new object[2]; args[0] = 9; args[1] = 18; m.Invoke(reflectOb, args); } else if((pi[0].ParameterType == typeof(double)) && (pi[1].ParameterType == typeof(double))) { object[] args = new object[2]; args[0] = 1.12; args[1] = 23.4; m.Invoke(reflectOb, args); } break; } Console.WriteLine(); } } } //=========================================================== /* C#: The Complete Reference by Herbert Schildt Publisher: Osborne/McGraw-Hill (March 8, 2002) ISBN: 0072134852 */ // A file that contains three classes. Call this file MyClasses.cs. using System; class MyClass { int x; int y; public MyClass(int i) { Console.WriteLine("Constructing MyClass(int). "); x = y = i; show(); } public MyClass(int i, int j) { Console.WriteLine("Constructing MyClass(int, int). "); x = i; y = j; show(); } public int sum() { return x+y; } public bool isBetween(int i) { if((x < i) && (i < y)) return true; else return false; } public void set(int a, int b) { Console.Write("Inside set(int, int). "); x = a; y = b; show(); } // Overload set. public void set(double a, double b) {
```

```
Console.Write("Inside set(double, double). "); x = (int) a; y = (int) b; show(); } public void show() { Console.WriteLine("Values are x: {0}, y: {1}", x, y); } } class AnotherClass { string remark; public AnotherClass(string str) { remark = str; } public void show() { Console.WriteLine(remark); } } public class Demo12 { public static void Main() { Console.WriteLine("This is a placeholder."); } } [/csharp]
```