Locate an assembly, determine types, and create an object using reflection

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

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

/* Locate an assembly, determine types, and create
an object using reflection. */

using System;
using System.Reflection;

public class ReflectAssemblyDemo {
public static void Main() {
int val;

// Load the MyClasses.exe assembly.
Assembly asm = Assembly.LoadFrom("MyClasses.exe");

// Discover what types MyClasses.exe contains.
Type[] alltypes = asm.GetTypes();
foreach(Type temp in alltypes)
Console.WriteLine("Found: " + temp.Name);

Console.WriteLine();

// Use the first type, which is MyClass in this case.
Type t = alltypes[0]; // use first class found
Console.WriteLine("Using: " + t.Name);

// Obtain constructor info.
```

```
ConstructorInfo[] ci = t.GetConstructors();

Console.WriteLine(“Available constructors: “);
foreach(ConstructorInfo c in ci) {
// Display return type and name.
Console.Write(” ” + t.Name + “(“);

// Display parameters.
ParameterInfo[] pi = c.GetParameters();

for(int i=0; i < pi.Length; i++) { Console.Write(pi[i].ParameterType.Name + " " + pi[i].Name); if(i+1 < pi.Length) Console.Write(", "); } Console.WriteLine(")"); } Console.WriteLine(); // Find matching constructor. int x; for(x=0; x < ci.Length; x++) { ParameterInfo[] pi = ci[x].GetParameters(); if(pi.Length == 2) break; } if(x == ci.Length) { Console.WriteLine("No matching constructor found."); return; } else Console.WriteLine("Two-parameter constructor found. "); // Construct the object. object[] consargs = new object[2]; consargs[0] = 10; consargs[1] = 20; object reflectOb = ci[x].Invoke(consargs); Console.WriteLine(" Invoking methods on reflectOb."); Console.WriteLine(); MethodInfo[] mi = t.GetMethods(); // Invoke each method. foreach(MethodInfo m in mi) { // Get the parameters ParameterInfo[] pi = m.GetParameters(); if(m.Name.CompareTo("set")==0 && pi[0].ParameterType == typeof(int)) { // This is set(int, int). object[] args = new object[2]; args[0] = 9; args[1] = 18; m.Invoke(reflectOb, args); } else if(m.Name.CompareTo("set")==0 && pi[0].ParameterType == typeof(double)) { // This is set(double, double). object[] args = new object[2]; args[0] = 1.12; args[1] = 23.4; m.Invoke(reflectOb, args); } else if(m.Name.CompareTo("sum")==0) { val = (int) m.Invoke(reflectOb, null); Console.WriteLine("sum is " + val); } else if(m.Name.CompareTo("isBetween")==0) { object[] args = new object[1]; args[0] = 14; if((bool) m.Invoke(reflectOb, args)) Console.WriteLine("14 is between x and y"); } else if(m.Name.CompareTo("show")==0) { m.Invoke(reflectOb, null); }
```

```
}                                   }                                   }
//=================================================================
==== /* 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]
```