**Lab 1 – Hello World: Project Setup**

**Create your first AutoCAD managed application**

In this lab, we will use Visual Studio .NET and create a new Class Library project. This project will create a .NET dll that can be loaded into AutoCAD which will add a new command to AutoCAD named “HelloWorld”. When the user runs the command, the text “Hello World” will be printed on the command line.

* Launch Visual Studio 2019 and then select File> New> Project. In the New Project dialog select Visual C# Projects for the Project Type. Select “Class Library” template. Make the name “Lab1” and set the location where you want the project to be created. Select OK to create the project.
* If the solution explorer is not visible in visual studio, you may turn it on by going to “View” menu and selecting “Solution Explorer”. This view will allow us to browse through files in the project and add references to managed or COM Interop assemblies. Open Class1.cs that was added by the NET wizard by double-clicking on it in the solution explorer.

**Connect to the AutoCAD Managed API –AcMgd.dll , AcDbMgd.dll ,AccoreMgd.dll**

* In Class1.cs notice that a public class was automatically created. We will add our command to this class. To do this we need to use classes in the AutoCAD .NET managed wrappers. These wrappers are contained in two managed modules. To add references to these modules:

a. Right click on References in solution explorer and select “Add Reference”.

b. In the “Add Reference” dialog select “Browse”.

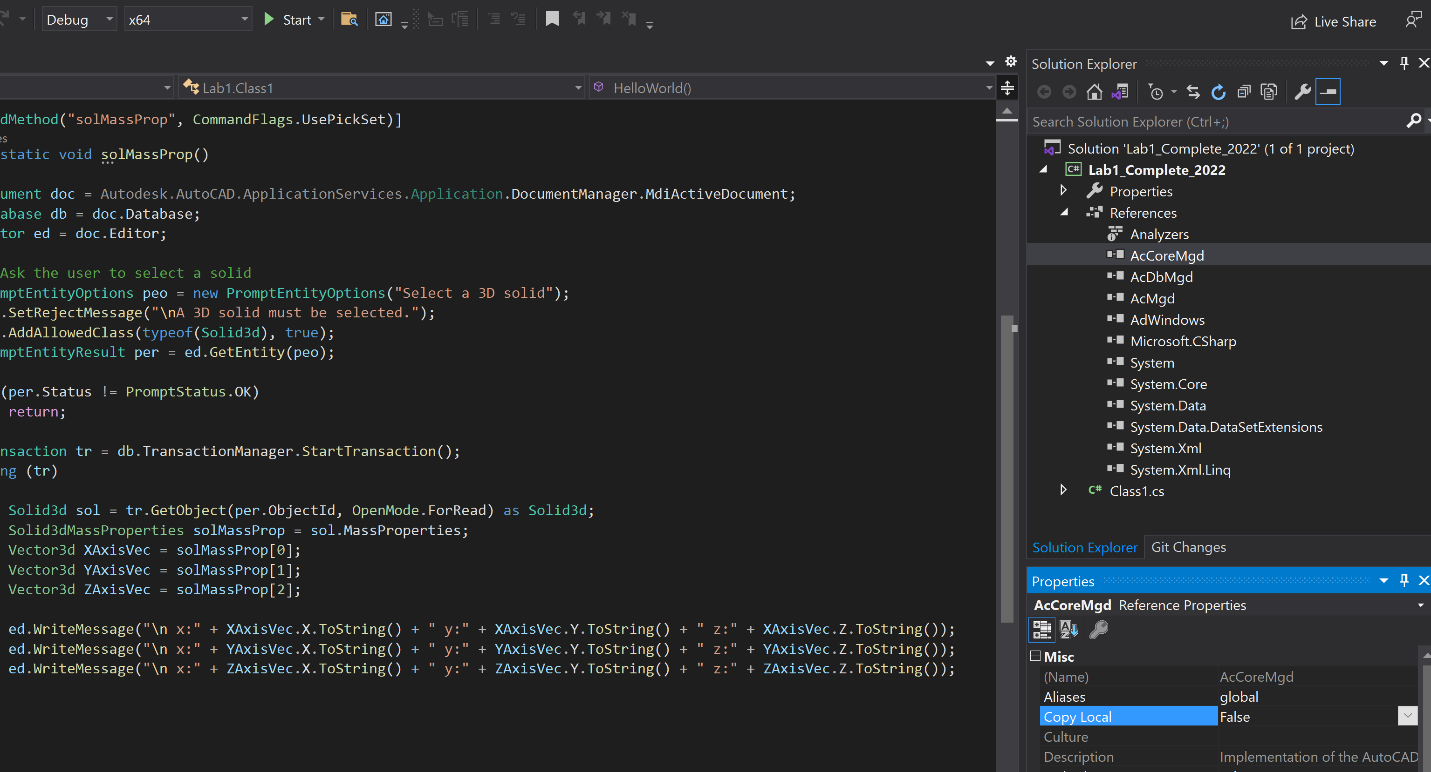
c. In the “Select Component” dialog navigate to the AutoCAD 2022 directory. (C:\Program Files\Autodesk\AutoCAD 2022) Find “acdbmgd.dll” and select open. Click “Browse” again then find and open “acmgd.dll” and “accoremgd.dll”. You can also type \*mgd.dll to filter for the required assemblies.

d. Click ok in the “Add Reference” dialog once these components are selected.

e. In the solution explorer right click on the acmgd.dll and select Properties. In the properties dialog change the Copy Local setting to false. (If this copy local is true then the dll does not load properly when debugging).

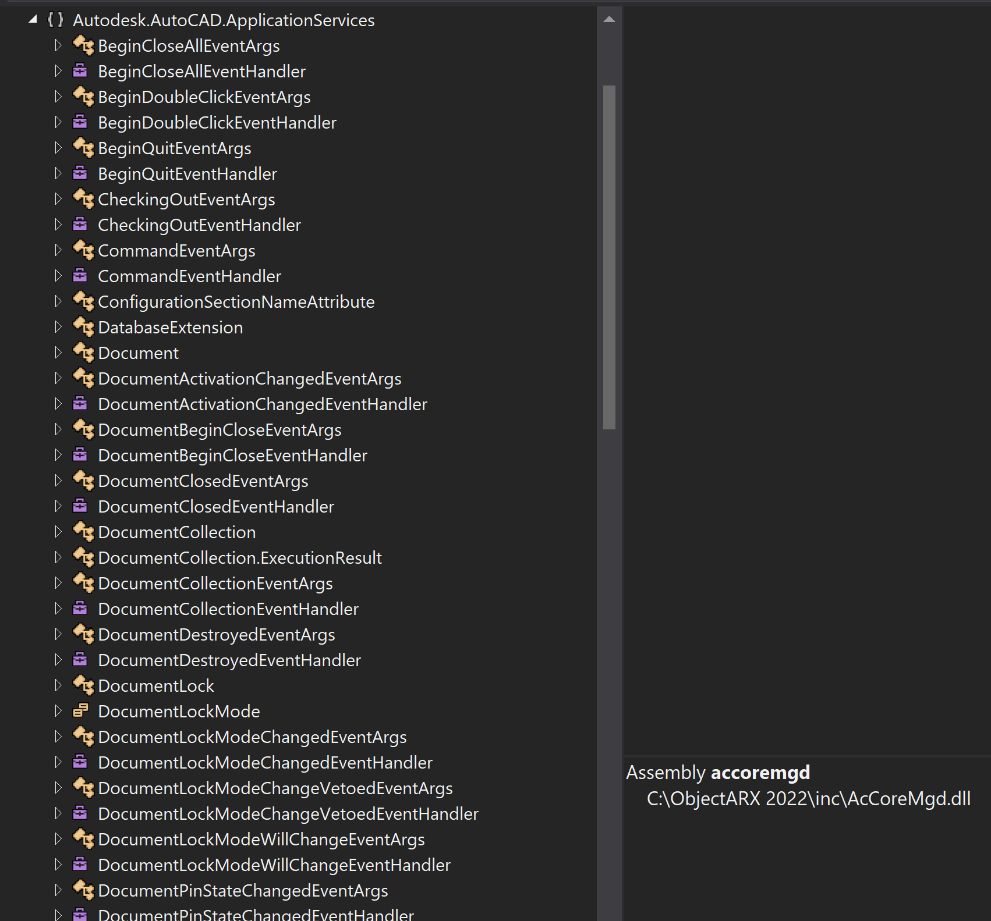
f. In the solution explorer right click on the accoremgd.dll and select Properties. In the properties dialog change the Copy Local setting to false. (If this copy local is true then the dll does not load properly when debugging).

g. Change the Copy Local setting for acdbmgd.dll to false also.



Note: acdbmgd.dll contains ObjectDBX managed types (everything to do with manipulating drawing files). acmgd.dll contains the AutoCAD’s managed types classes which only work in AutoCAD. Accoremgd.dll contains the AutoCAD’s core business logic.

* Use the Object Browser to explore the classes available in these managed modules. (View > Object Browser. Expand the “AutoCAD .NET Managed Wrapper” (acmgd) object. Throughout the labs we will be using these classes. In this lab an instance of “Autodesk.AutoCAD.EditorInput.Editor” will be used to display text on the AutoCAD command line. Expand the “ObjectDBX .NET Managed Wrapper” (acdbmgd) object. The classes in this object will be used to access and edit entities in the AutoCAD drawing.



* Now that we have the classes referenced we can import them. At the top of Class1.vb above the declaration of Class1 import the ApplicationServices, EditorInput and Runtime namespaces.

using Autodesk.AutoCAD.ApplicationServices;

using Autodesk.AutoCAD.EditorInput;

using Autodesk.AutoCAD.Runtime;

**Define your first command**

* We will now add our command to Class1. To add a command that can be called in AutoCAD use the “CommandMethod” attribute. This attribute is provided by the Runtime namespace. Add the following attribute and function to Class1.

[CommandMethod("HelloWorld")]

public void HelloWorld()

{

}

* When the “HelloWorld” command is run in AutoCAD, the HelloWorld function will be called. In this function we will get the instance of the editor class which has methods for accessing the AutoCAD command line. (as well as selecting objects and other important features) . The editor for the active document in AutoCAD can be returned using the Application class. After the editor is created, use the WriteMessage method to display “Hello World” on the command line. Add the following to the function HelloWorld:

Editor ed =

Application.DocumentManager.MdiActiveDocument.Editor;

ed.WriteMessage("Hello World");

**Test in AutoCAD**

* To test this in AutoCAD we can have Visual Studio start a session of AutoCAD in debug mode:

a.Right click on “Lab1” in Solution Explorer and select “Properties”.

b.In the Lab1 Property Page dialog select “Debug”.

c.In the Start Action area, select “Start external program:”

d**.** Next use the ellipses button and browse to acad.exe.

e.Select acad.exe and press “OK”.

f.After changing this setting hit F5 or select Debug>Start from the menu to launch a session of AutoCAD. This will build your application and start AutoCAD automatically or stop after building if there are any errors. Try fixing any build errors you may have.

* The “NETLOAD” command is used to load the managed application you just built. Type NETLOAD on the AutoCAD command line to open the “Choose .NET Assembly” dialog. Browse to the location of “lab1.dll” (..\lab1\bin\debug) select it and then hit open.
* Enter “HelloWorld” on the command line. If all went well, the text “Hello World” should appear. Switch to Visual Studio and add a break point at the line: ed.WriteMessage(“Hello World”). Run the HelloWorld command in AutoCAD again and notice that you can step through code. The “Debug” menu in Visual Studio has several options to step through the flow of execution in your program.

A point to note for future reference is that if you do get problems loading your application, use the fuslogvw.exe to diagnose.

Back in Visual Studio try Exploring the CommandMethod attribute in the ObjectBrowser. Notice that it has seven different flavors. We used the simplest one that only takes one parameter, the name of the command. You can use the other parameters to control how the command will work. For example, you can specify command group name, global and local names, command flag (for the context in which the command will run), and more.

End of Lab1.