Assembly Types & Execution

In this chapter we will discuss about different assembly
we have three different types of assemblies namely:

- ✓ **Private Assembly**
- ✓ **Shared Assembly**
- ✓ **Satellite Assembly**

Out of these, we are going to deep dive into first two & s
to them apart from normal basics. We are not going to se
this chapter

We know .Net based projects are typically going to get
format. In generic way we will call this output as an a
EXE/DLL - is not like any normal windows based binar
packaging structure is different. It is suitable to sup
language interoperability, memory management, GC by C

Assembly (this is what we will refer to EXE/DLL) has d
Let us say, that we have created a project using languag
have used some images like resources into the project. W
(integer, character, custom classes etc.) in the project
project, we get an output i.e. assembly.

What this assembly has?

Well, the output assembly has majorly four sections:

- ✓ **Assembly Metadata**
- ✓ **Resources**
- ✓ **Type Metadata**
- ✓ **MSIL**

Let us see these sections:

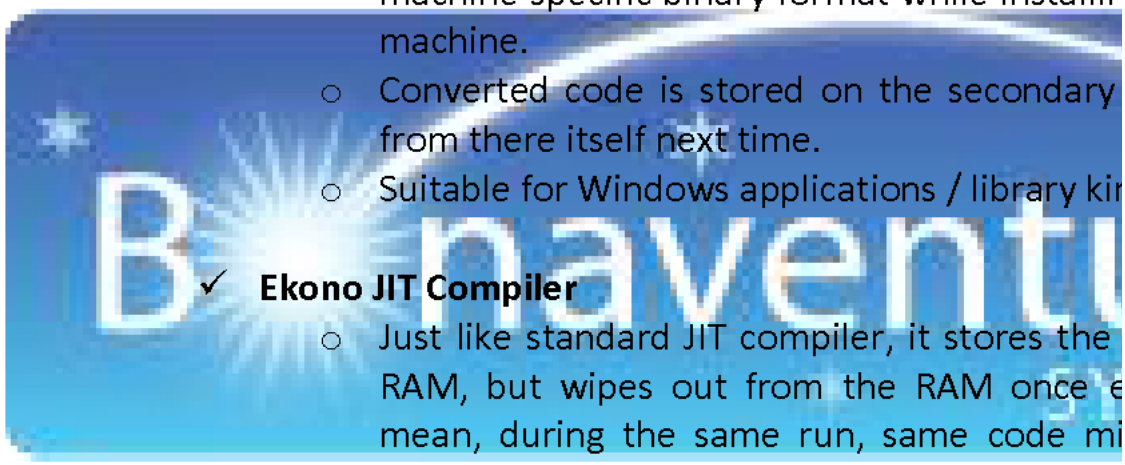
- ✓ **Assembly metadata:** This section contains info
  version of the assembly, optionally company nam
  which will be acting just like an index for rest of th
- ✓ **Resources:** This section will contain resources

compiler available with framework does the job of
code in to the machine specific code.

JIT in turn has three different **flavors**:

- ✓ **Standard JIT Compiler:**
  - Shipped by default
  - Used / Called by default by CLR
  - Compiles the code in to machine specific bina
  - Keeps the code in RAM
  - If application goes down/ machine's power
    gone!
  - Next time – it starts again
  - Suitable for web applications where the
    application keeps on running 24 X 7

- ✓ **Pre – JIT Compiler:**
  - Need to call specifically
  - Use **NGEN.EXE (N**ative Code **Gen**erator**)** to
    machine specific binary format while installi
    machine.
  - Converted code is stored on the secondary
    from there itself next time.
  - Suitable for Windows applications / library ki



- ✓ **Ekono JIT Compiler**
  - Just like standard JIT compiler, it stores the
    RAM, but wipes out from the RAM once
    mean, during the same run, same code mi
    cleaned from the RAM number of times.
  - It works as -> Compile MSIL on reference -> S
    > Wipe after execution -> Compile MSIL on re
  - Suitable for mobile based application, where
    need to be used effectively.

Let us discuss now, **how an assembly executes**:

Normal binary EXE (just like one you create using C++) has s
text section, data section, binary statements etc. This st

- ✓ Windows runtime realizes that it can execute t structure very well.
- ✓ It then, allocates memory for execution.
- ✓ Reads text section, data section & executes stater

Now, see what will happen around when you double click on th

Wait! Before that let us discuss one more term – **EXE & DLLs:**

One of the important point to note here is by default **DLLs can't**

They can't have their own memory for execution. **They are paras**

It is just like a Client – Server terminology. **How come?**

See,

What Client does? – Always requests for something to Server.

What Server does? – Always responds to the client's requests.

Same here, if you see DLLs are meant for reference. They just execute. EXEs can refer the DLLS. What happens when the DLL is

See here:

- ✓ EXE is running & has its own memory under which
- ✓ EXE gives a call to DLL
- ✓ As, we know now, DLLs can't have their own mer one
- ✓ DLL loads itself into the EXE memory area (just like
- ✓ See from the Client – Server terminology: EXE r input & DLL replies with output.
- ✓ So, DLL is active as a Server & EXE is acting as a Cli
- ✓ But something more interesting is: Server (DLL) ru Client (EXE) process.
- ✓ So, DLLs are also called as In-Process Server (tha Client). In short we refer these as In-Proc Server.

Now, let us see what will happen around when you double

- ✓ What this DLL is? Let us discuss about this first &
  execution part.

**What is MSCORLIB? (Alias MSCOREE – Microsoft Common
Environment)**

It is CLR (Common Language Runtime) – heart & soul of .NET

About Common Language Runtime:

CLR is a heart of .NET framework. It does lot of important task

- ✓ MSIL into native code conversion through JIT c
- ✓ Execution
- ✓ Memory allocation
- ✓ Garbage Collection
- ✓ Exception Handling
- ✓ Type Loader
- ✓ Security Check
- ✓ COM Marshalling
- ✓ Many more ....

Since, these are very core tasks, so is the CLR in terms of .NET.

If you have noticed above, it's a DLL. So what? - Obviously
execution. Where will CLR get the memory? We know from
don't have their own memory space & always execute insi
happens in this case? Will there be multiple CLRs when you
(EXEs) on your machine?

CLR is a special kind of DLL. It can have its own memory for e

If you remember, in VB 6.0, apart from conventional DLLs, w
concept which can have its own memory for execution. It
library which needs to be referred & can have its own memor

But, we don't have such ActiveX concept here.

Rather **in terms of out of process memory, similar concept he**

We will see shared assembly concept in more detail later in th

- ✓ After this, the CLR takes over the control of ex runtime & starts executing an assembly.
- ✓ So, what it does then?
- ✓ CLR starts reading the assembly top to bottom.
- ✓ First it reads the **assembly metadata** – with whic rest of the structure of an assembly
- ✓ It then, **loads the resources, types** required t program
- ✓ Then. It gives a call to **JIT compiler to convert native code**.
- ✓ Depending on what type of application (Web/W **JIT compiler (Standard/Pre/Ekono)** can be calle upon the case.

Who allocates the memory for application EXE or in this case ou

Obviously, CLR does this. When?

When it comes in picture for the first time, it gets some 'X' amc for management. Then onwards CLR will act as if a virtual c program (assembly running) under the CLR will ask to CLR for an

Any program will not directly talk with OS at any time. So, within the Root OS.

So, EXE gets the memory from CLR. We will discuss about the garbage collection algorithm in details later in the book.

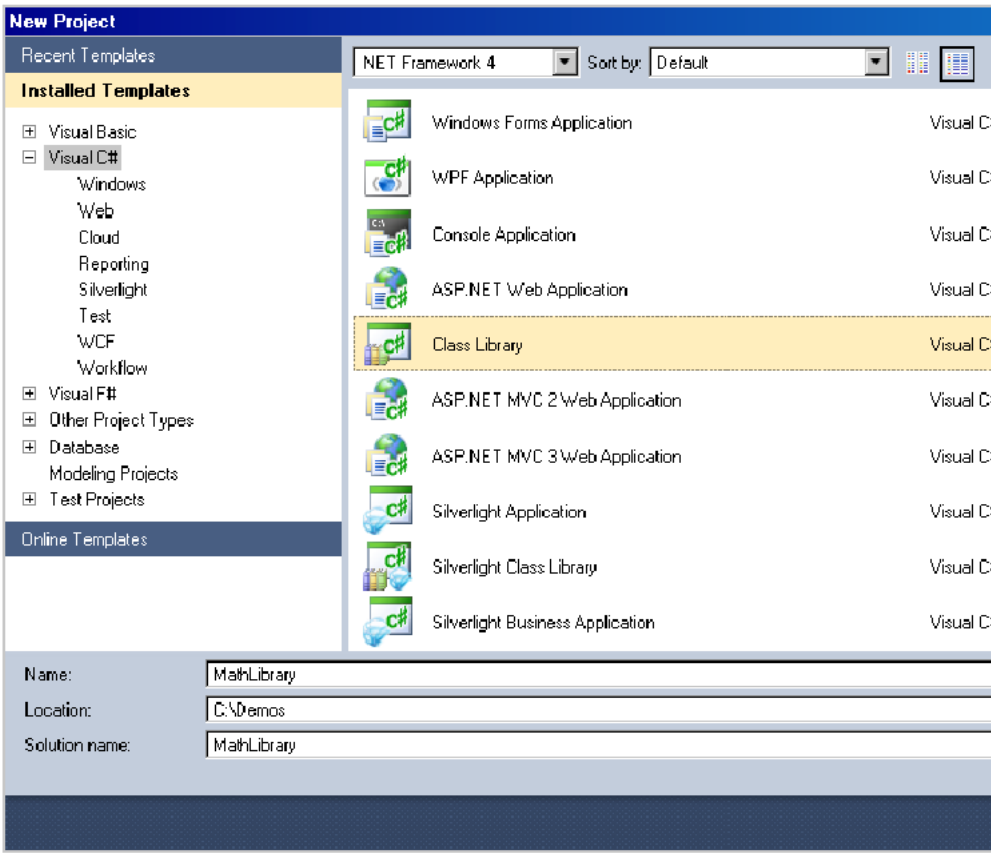
We talked about the EXE, DLL assemblies. But what are private &

By default every assembly is a private assembly.

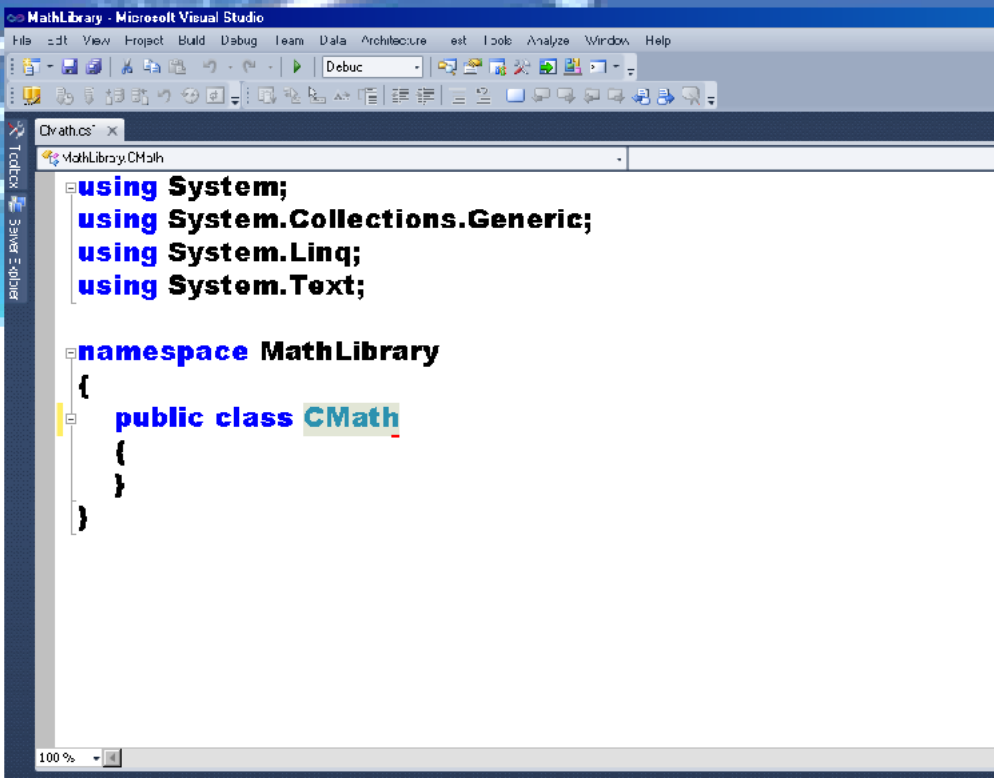
Let us create a program to understand private assembly concept

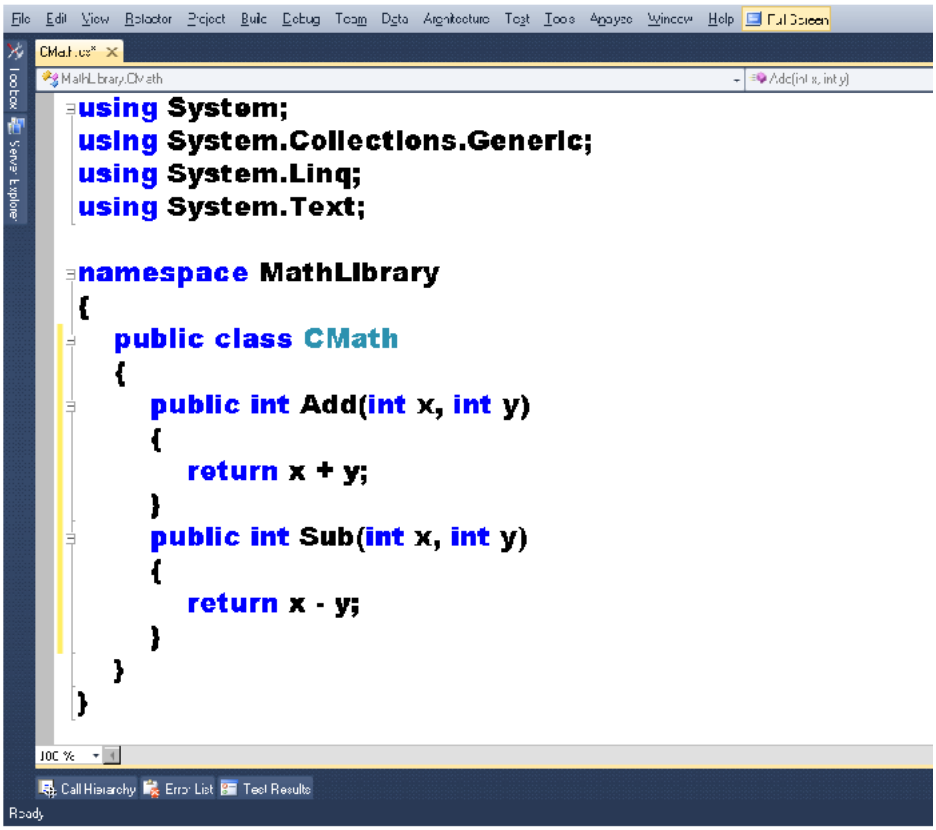
Steps:

1. Start Visual Studio 2010 by clicking on: **Start -> All Progr Studio 2010 ->Microsoft Visual Studio 2010 Icon**
2. Click on: **File -> New -> Project**
3. In the dialog: Choose C# as a language -> Choose project

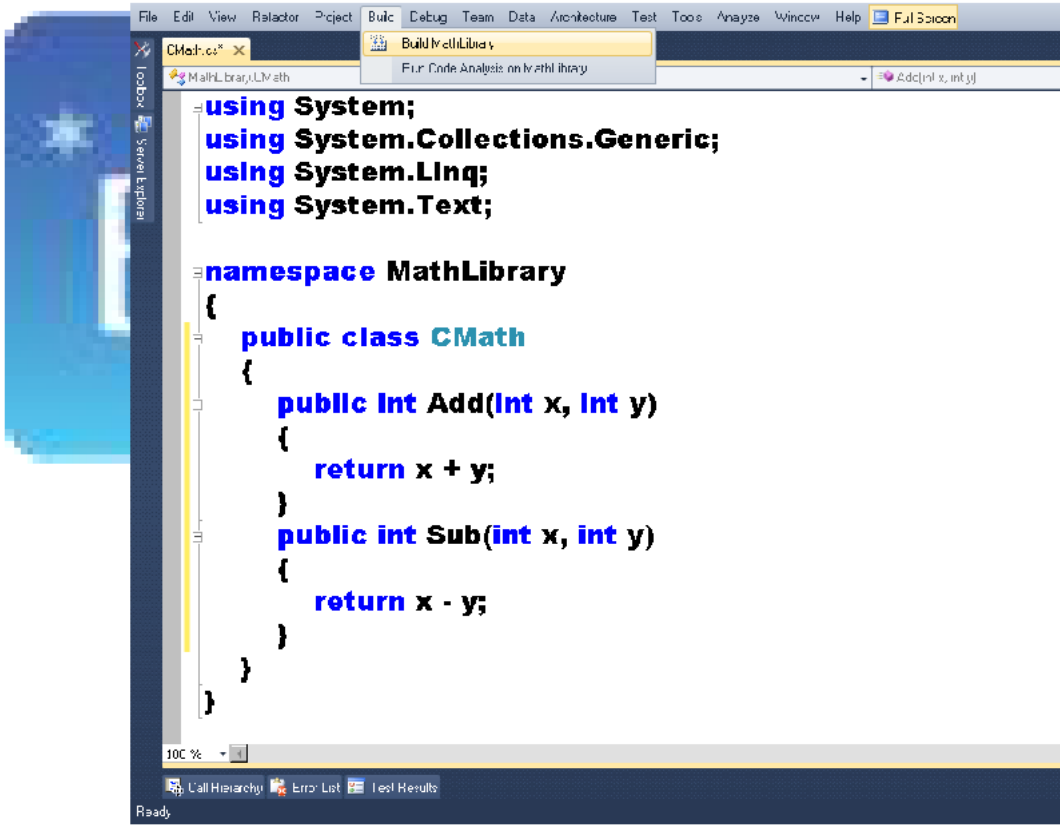


4. Click on Ok Button.
5. Name the class file as CMath.cs in the solution explorer as w file as shown:

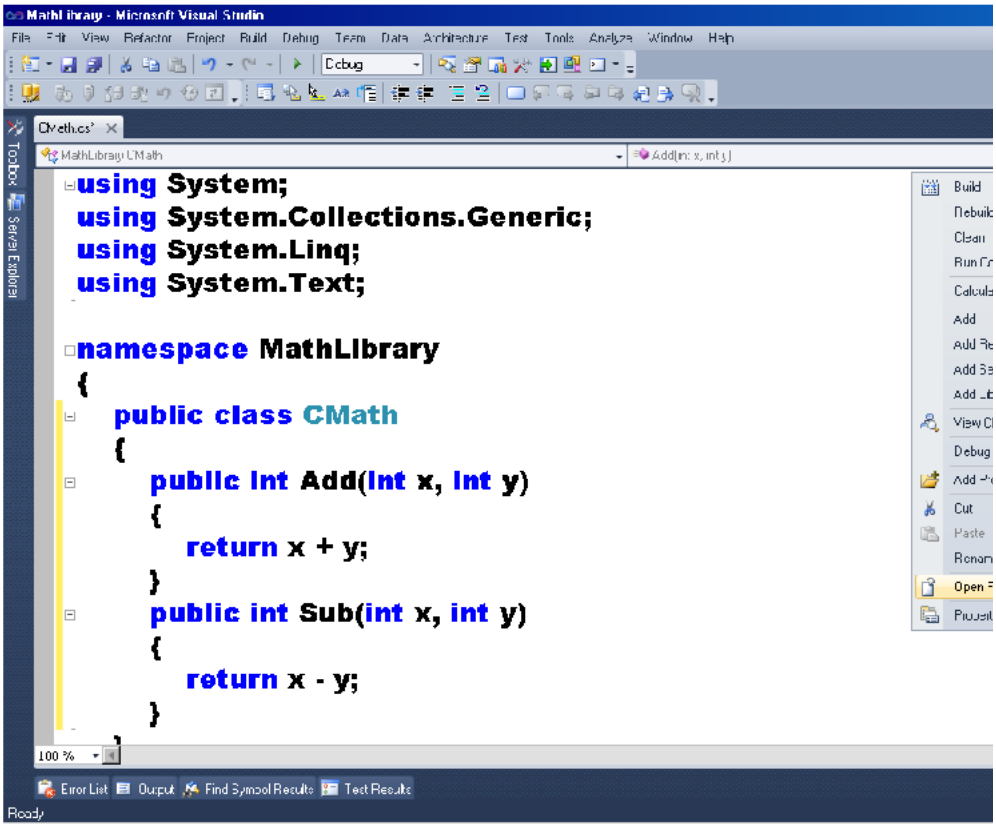




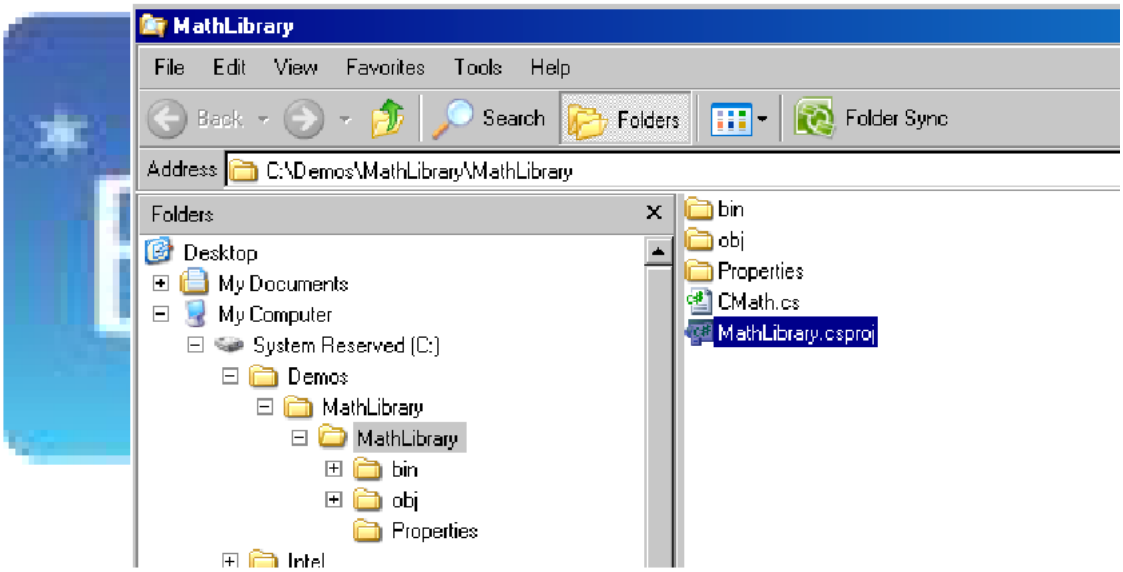
7. Compile the code:



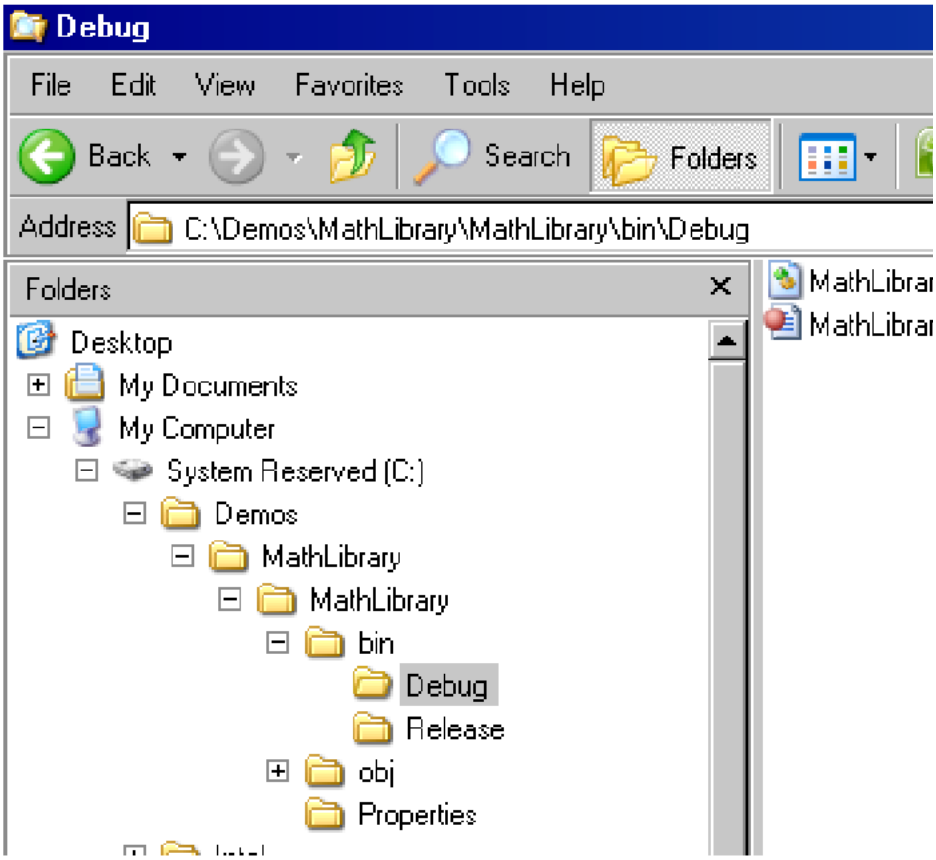
8. Right click on the solution explorer & click on Open Folde



9. It opens:



10. Go to bin-> debug folder

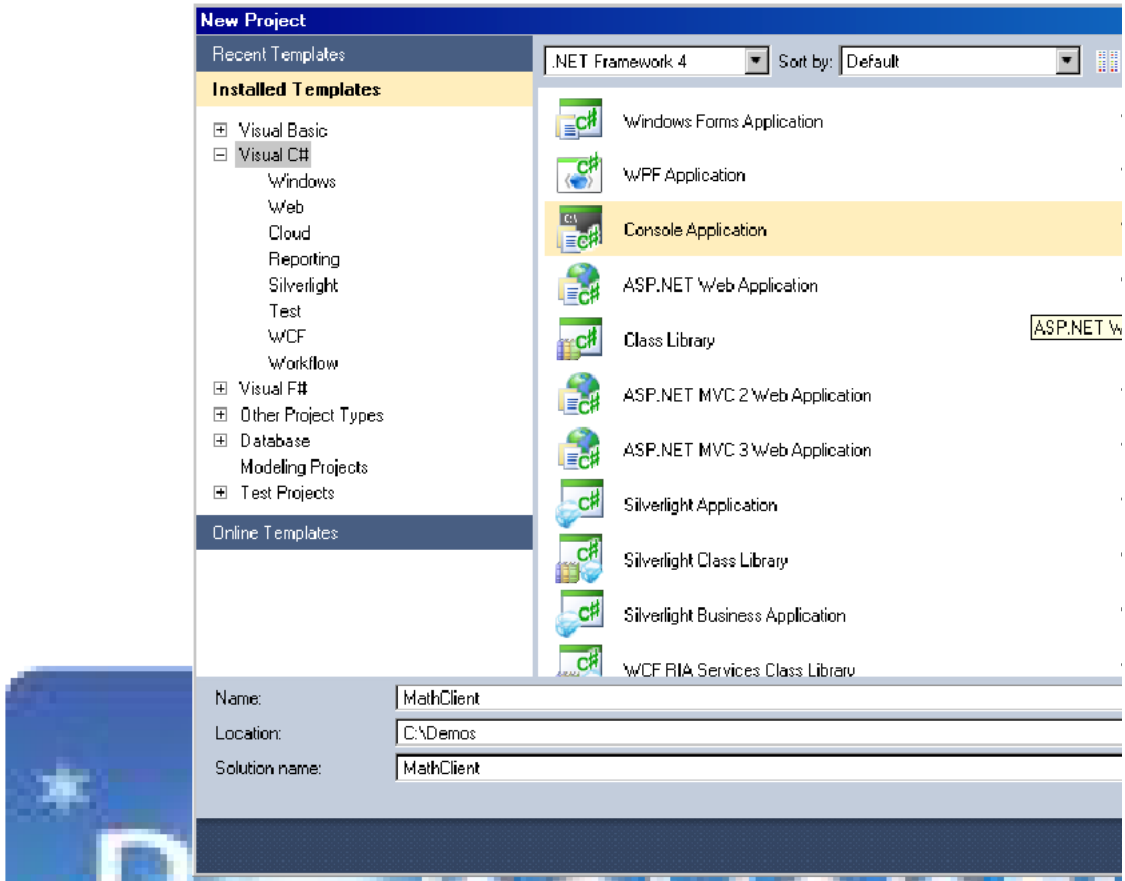


11. You will an output assembly (DLL) created called MathL
click on the DLL, it will not execute.

12. This is server (DLL) program, Now let us create a client (E

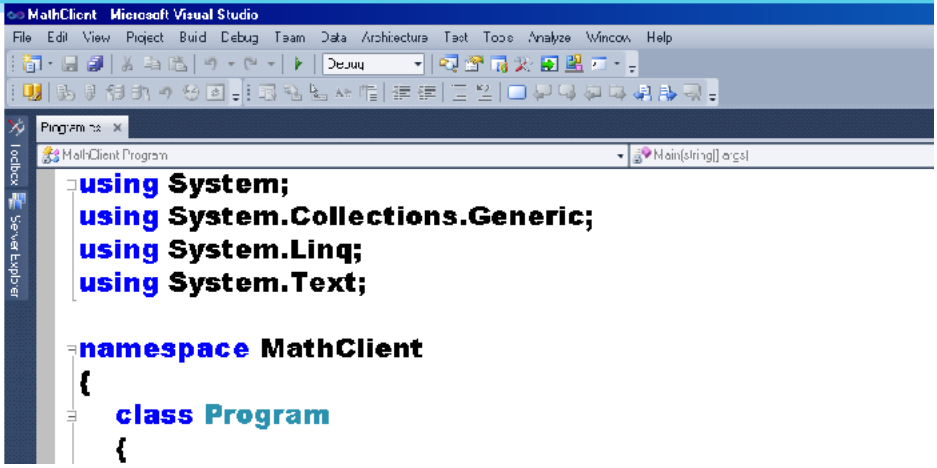
13. See on Next Page:

14. Start one more time Visual Studio 2010 by clicking on: **St**
    > Microsoft Visual Studio 2010 ->Microsoft Visual Studic
15. Click on: **File -> New -> Project**
16. In the dialog: Choose C# as a language → Choose pro
    Application → Name the project as MathClient as shown

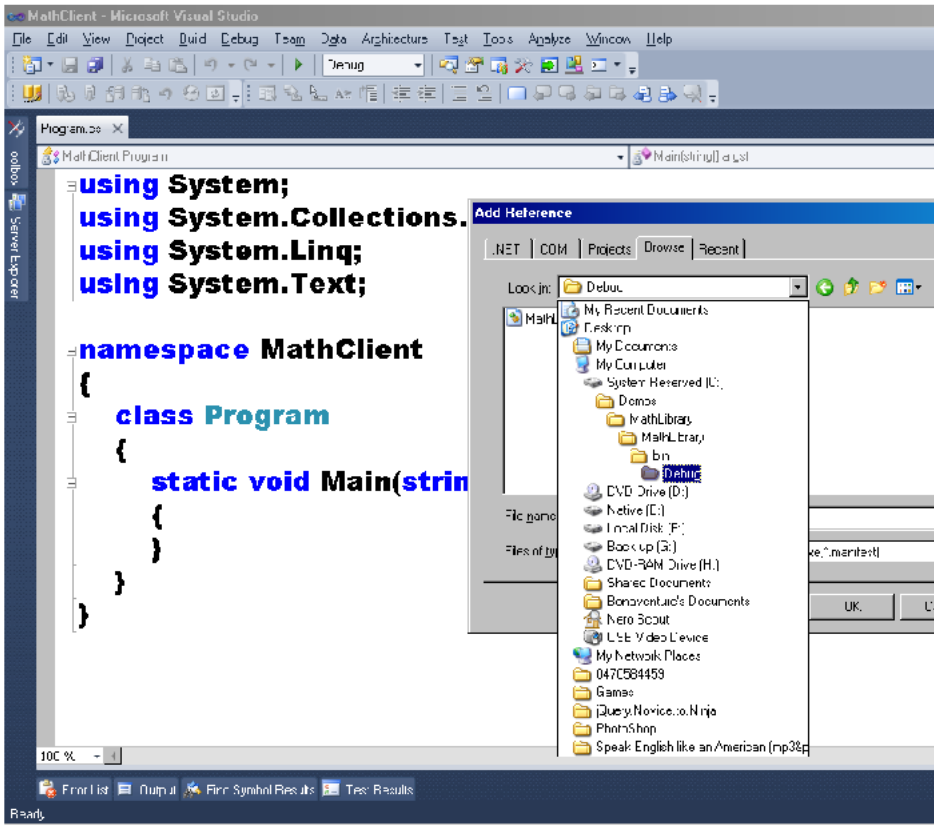


17. Click on Ok

18. In the solution explorer, right click on references & click

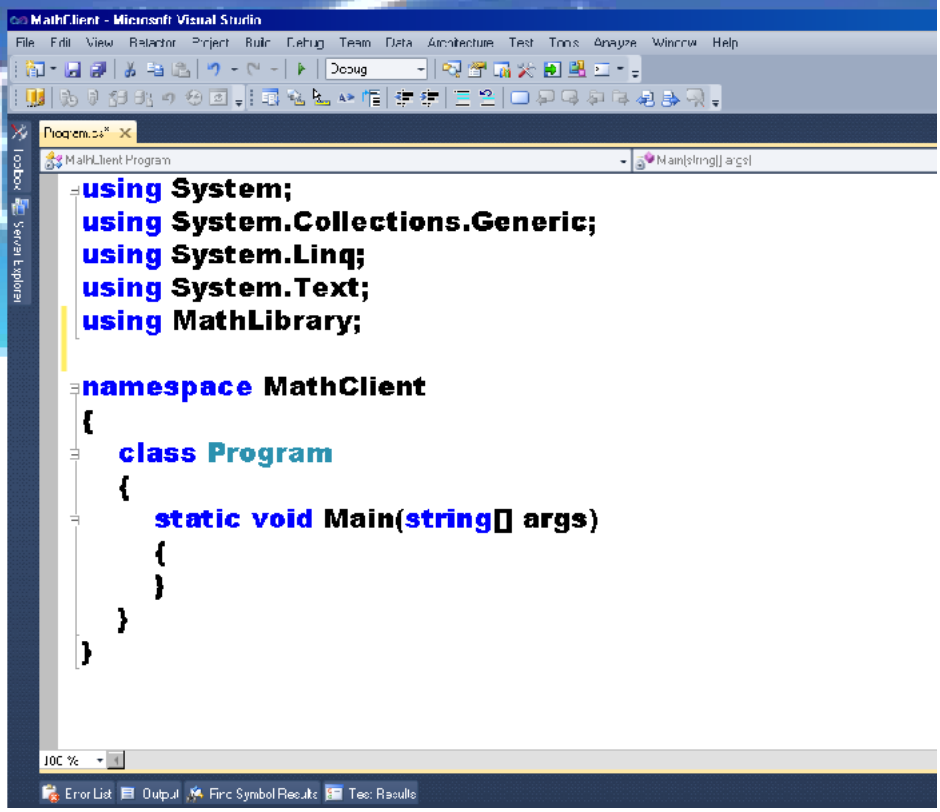


20. Navigate to previously created DLL.



21. Click on Ok.

22. In Program.CS file add namespace by writing: Using Math

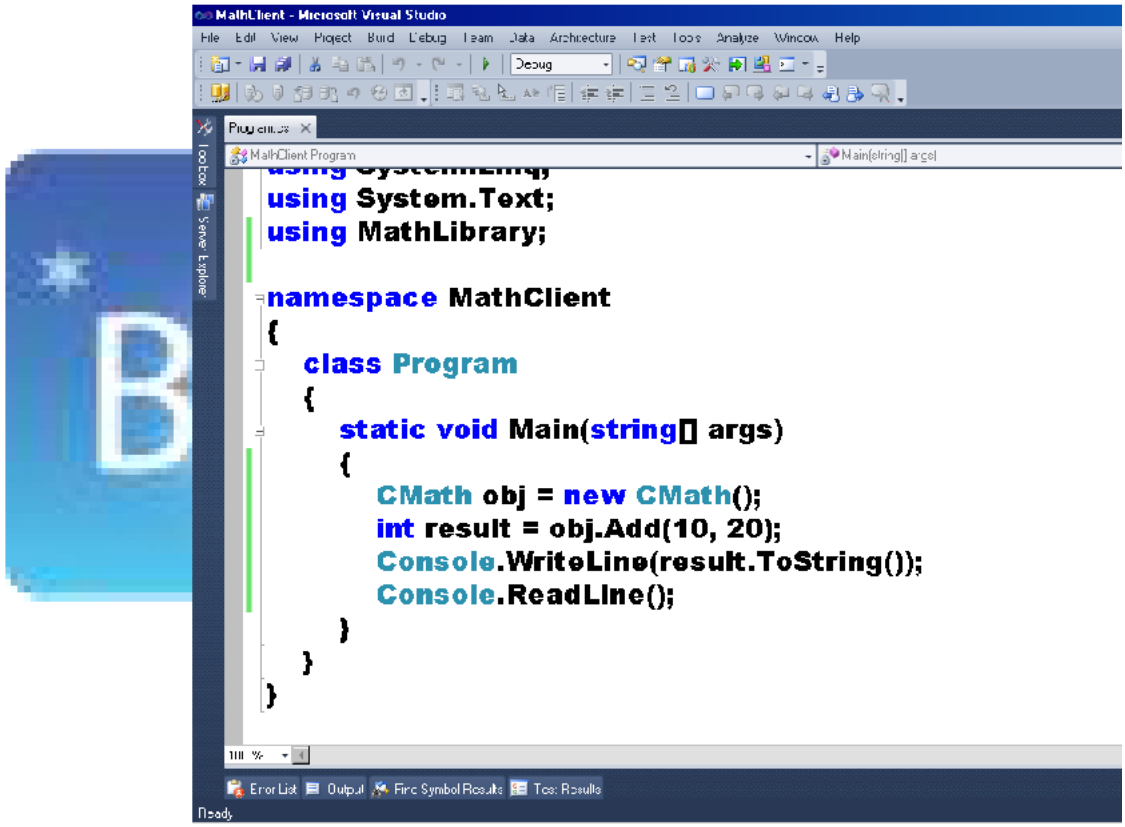


24. Press F5 to run the program. You should see the output a

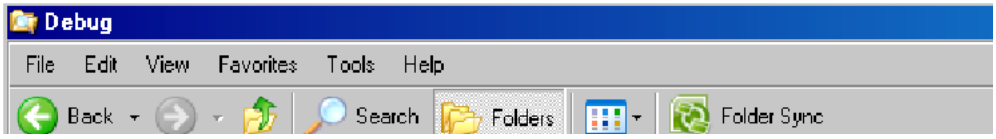
25. So, the code is fine. Client is calling Server & we are gettir

26. All the assembly execution that we discussed above is co

27. Now, right click on solution explorer. Click on Open Explorer.

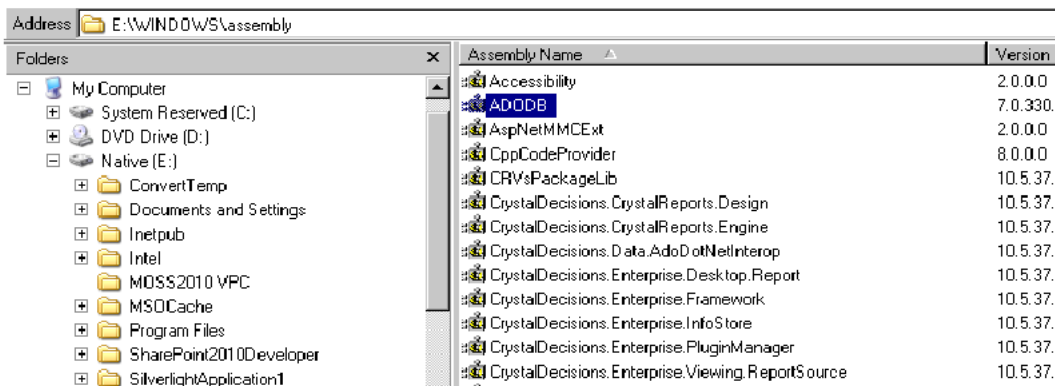


28. Observe now, what the folder bin-> debug has:



29. If you see, there is a copy of MathLibrary.DLL created in t
30. It's literally copy – paste. Now what will happen if we MathLibrary.DLL? Will the change reflect in client prograr
31. Answer is **No.**
32. It will not unless & until the Client EXE program is comp DLL again.
33. This current copy of the DLL is called as Private copy or Pr
34. If we build another Client program then that client will al the same DLL.
35. If we run both the clients at the same time on same mach be referring to its own copy.
36. If one of the client's private DLL gets corrupt then that wi other client running.
37. Which means with private copy helps in running independently but at the same time, it also generates a p
38. These DLLs run inside the memory of client EXEs.
39. The problem is when all clients' needs to be referring ne DLL then everyone should be either recompiled or old c replaced with the new one!
40. Can't there be any location where in if we replace the machine will get it from the same location.
41. This is where shared assembly concept comes in picture.

All the shared assemblies are registered with .NET registry (.N registered with Windows registry) & can be seen at [Native Drive]:\\Windows\Assembly folder.



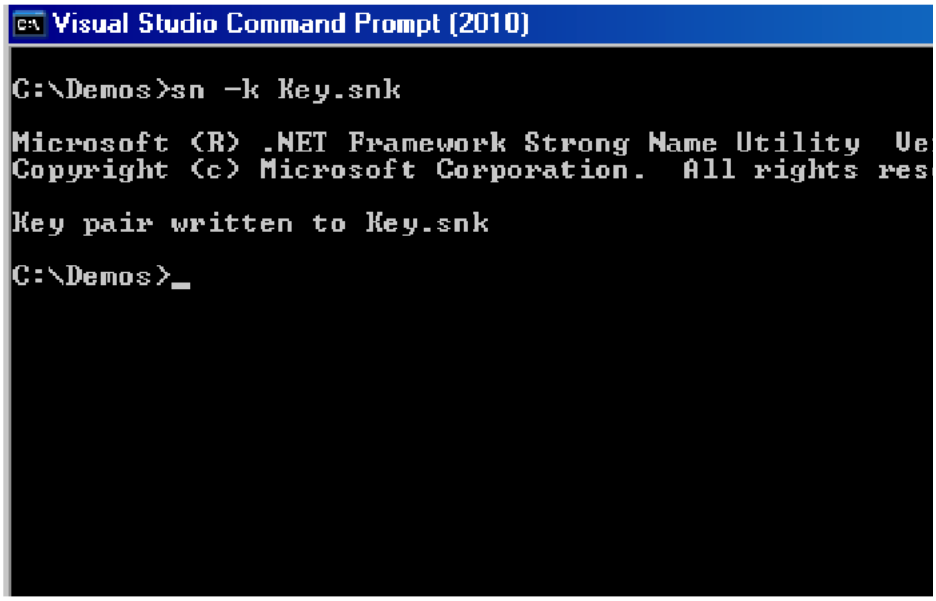
- ✓ This DLL is private assembly. This is just a simple coded DL done apart from compilation.
- ✓ When CLR gives a call while executing this DLL then it simply the EXEs memory space without even checking whether execute. Somebody might have written /tampered the DLL w
- ✓ To enable this security checking, the above mentioned key private key & public key.
- ✓ These are special prime numbers (one prime other co - pr the help of an algorithm called RSA (stands for three peopl algorithm – **R**ivest, **S**hammir, **A**dleman).
- ✓ One number is called private key & other is public key.
- ✓ Later on after generating these keys, hashing algorithm assembly. The contents of assembly are hashed (encryp algorithm with private key as a second operand. This hash c assembly metadata.
- ✓ The encrypted output can't be decrypted. Then what's the us
- ✓ This hash code acts just like a checksum bits. Later on, wh loaded by CLR, it simply does the same task of hashing ag check old hash (from metadata) with new hash. If the com then there is no problem. But is the comparison fails then th comes.
- ✓ Public key is used for unique identification of the assembly.

Since the shared assembly has to be loaded in shared memory area outside the client EXE, CLR must be careful while loading any assem tampered one. It checks the same using above discussed proces discussed process, the assembly must have private – public key pair

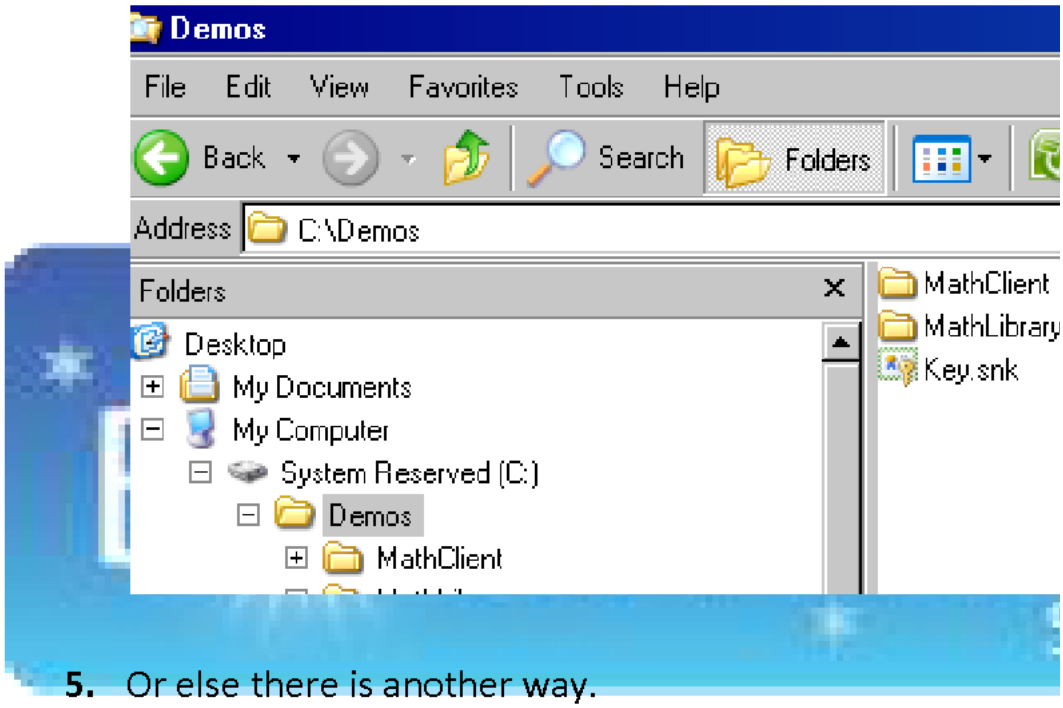
So, to create a shared assembly one should have key pair assigned t

Let us create one pair & attach with MathLibrary.DLL.

1. Create a strong name key pair (private key + public key) usi

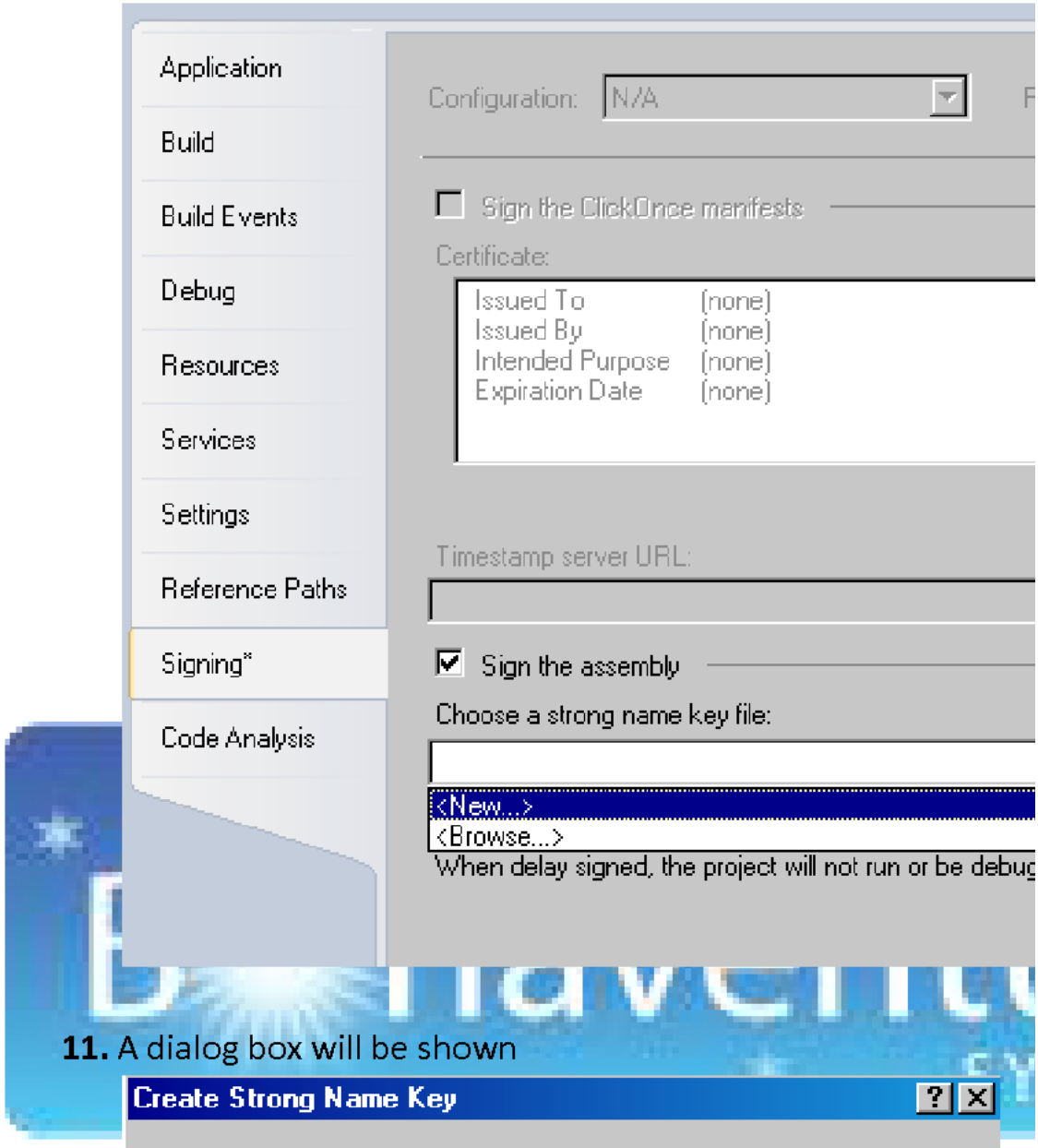


4. On C drive, see the file created.

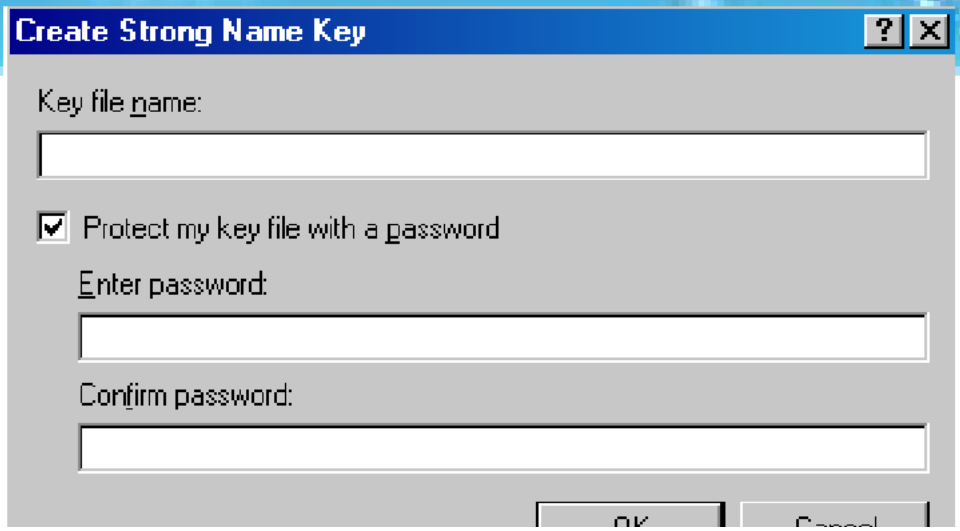


5. Or else there is another way.
6. Right click on the MathLibrary project under solution exp
7. Click on Properties menu item.
8. Click on the Signing Tab

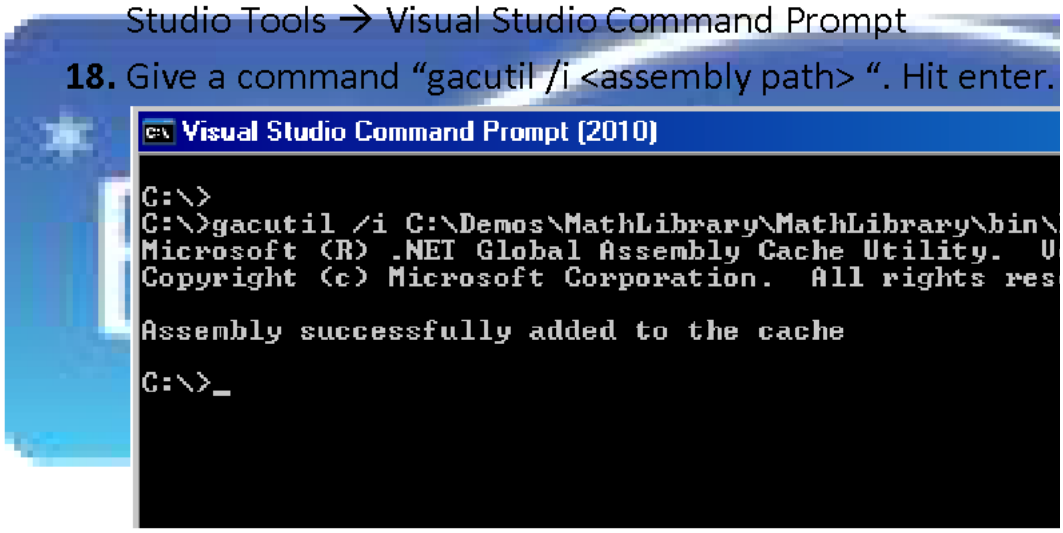
9. If you choose browse then you can navigate to the file that v
   Key.snk & compile the project.
10. Else click on sign the assembly then choose <New> from the



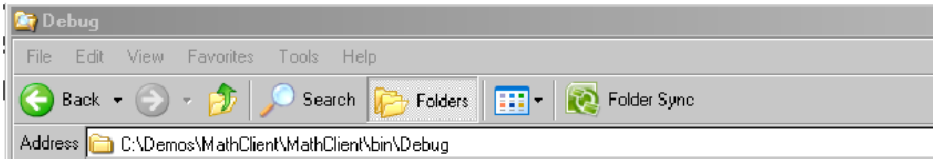
11. A dialog box will be shown



14. After this click on Ok & compile the project.
15. You will see that the file exists in the solution explorer.
16. Now, we have key pair added to the DLL or assembly.
17. To install this assembly, into the global assembly cache (G area, Click on start → All Programs → Microsoft Visua Studio Tools → Visual Studio Command Prompt
18. Give a command "gacutil /i <assembly path> ". Hit enter.



19. Now open the <Native Drive>:\\Windows\\Assembly Fold
20. You should see the MathLibrary.dll installed.
21. Now, recompile the MathClient EXE client program.



22. See, if it runs. It works.
23. In the output folder you will not see private copy of the Matl shared now just like CLR DLL!
24. We have successfully created shared DLL.

Generally, while executing EXE program whenever CLR sees any DLL **DLL is first searched in GAC area** by CLR. If not found in the GAC a local copy. If not found there as well then it throws an exception.

What will happen if there are two copies one at the shared area private copy with EXE & their versions are different?

Generally, a DLL is referred with which reference, the EXE is built. S with 1.0 version then CLR will always look for 1.0 versioned DLL. If i referred as a shared assembly else private copy is searched.

But to make sure that specific version is always referred one must DLL.

What will happen if there are two versions of the same DLL inside happen & which DLL will the CLR refer?

Same answer. It will refer the DLL with which reference, the EXE is b

What if one wants to redirect reference to other versioned DLL EXE program?

If you want to target some other version of the DLL for the EXE bu version of the same DLL then you will have to tell CLR to refer oth

Probing Element in Application (EXE) Configuration file:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-co
      <dependentAssembly>
        <assemblyIdentity name="<DLL Assembly Name>"
          publicKeyToken="<public key of the DLL Ass
          culture="en-us" />
           <bindingRedirect oldVersion="1.0.0.0" new
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

This is where we will end our chapter.

Do see following list of issues that you may face while doing prac

Known Issues while practicing:

1. **I can't create strong name key pair using command prompt have permission. What do I do?**
   Do see that you are working as a local administrator while Studio Command Prompt or else try to use visual studio wi click on the project under solution explorer & then choosir from vertical tabs.

2. **How do I uninstall the assembly from GAC?**
   Use command ***GACUTIL /u "<Assembly to be uninstalled*** Command Prompt.

3. **Can I install the assembly using drag & drop?**
   Yes, but you need to make sure that assembly is completes such as – assembly should have strong name key pair, etc.

4. **Can I copy & paste?**
   No, you won't be able to copy & paste the assembly but you assembly.

5. **How do I install the assembly on client machine? Since I v the Setup by my own. Is it possible to install the DLL assem installing the EXE project set up on the client end?**
   Yes, Right click on the EXE project under solution explorer -> under prebuilt command option you need to write the cor ***GACUTIL /i "<Assembly to be uninstalled>"*** which will be ex while client installs the EXE project on the machine. You nee DLL is also packaged in the setup project which you will hand

Do write an email in case of other queries to:
mahesh@bonaventuresystems.com