Class Library Lab

Perform these labs on your own computer using Visual Studio 2022 to ensure you understand the lessons presented in the corresponding videos and lectures.

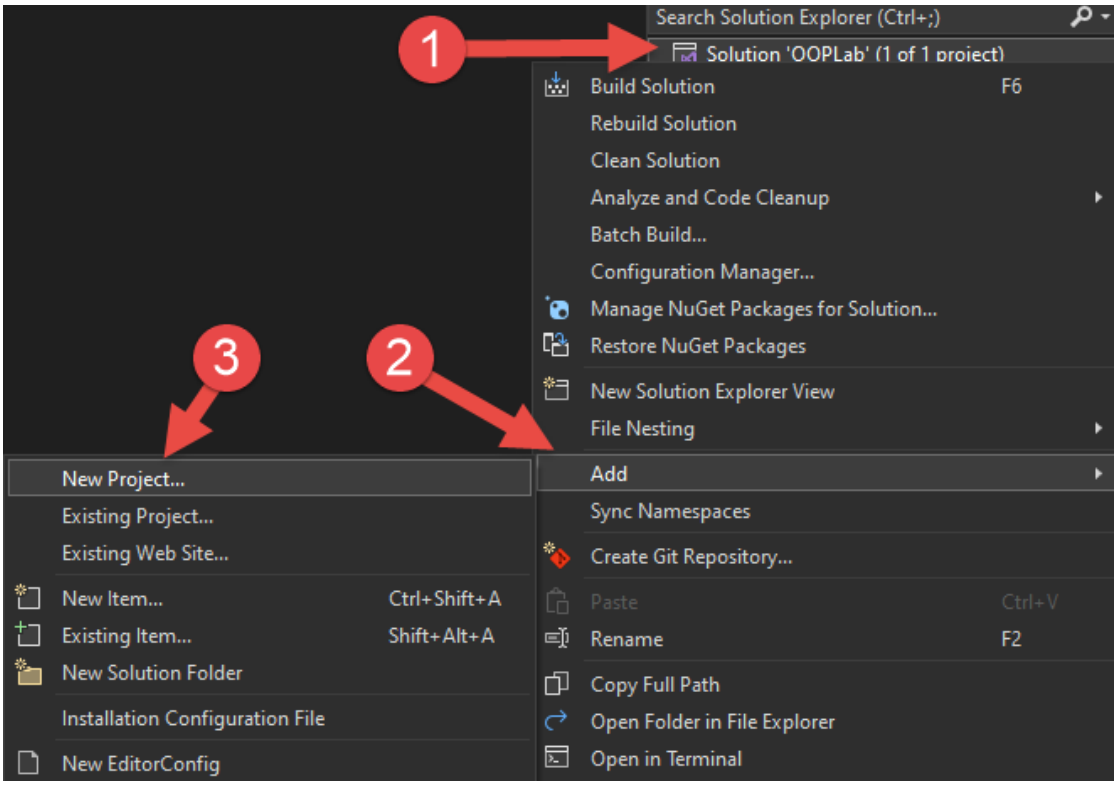
Lab 1: Create a Common Library

The first step in breaking up your application into reusable libraries is to identify the different categories of classes that you have. In this simple little OOPLab application that you have been building in this course, there are actually a few different categories of classes.

1. The FileHelper class and the IRepository interface can be used in any type of application.
2. The Customer, Person, CustomerEventArgs classes and the IPerson interface could probably be used in any type of application you would write for your company.
    a. Just like you kept these files in separate folders, you probably want to create a class library just for these "Entity" type classes.
3. The CustomerRepository and PersonRepository classes could probably be used in any type of application you would write for your company.
    a. These classes should be separated into their own class library.

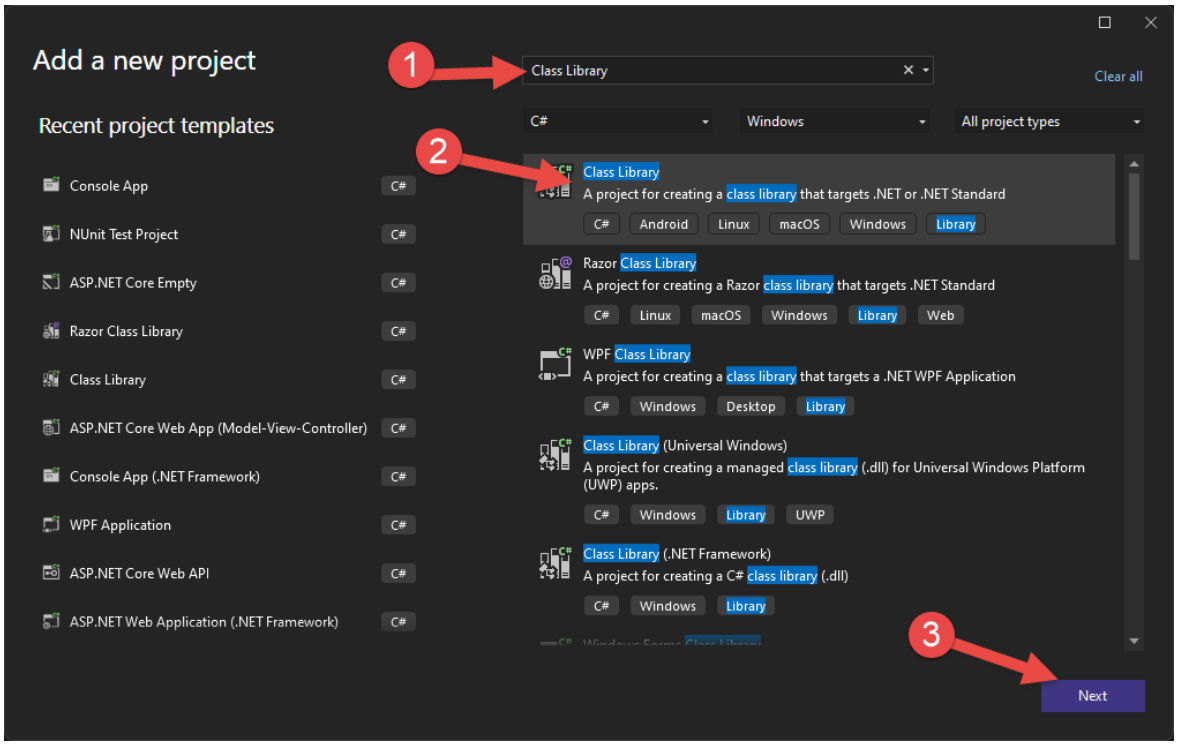
Create the Common.Library Project

Right mouse-click on the OOPLab solution and select **Add | New Project…**



Type in "Class Library" in the search box.

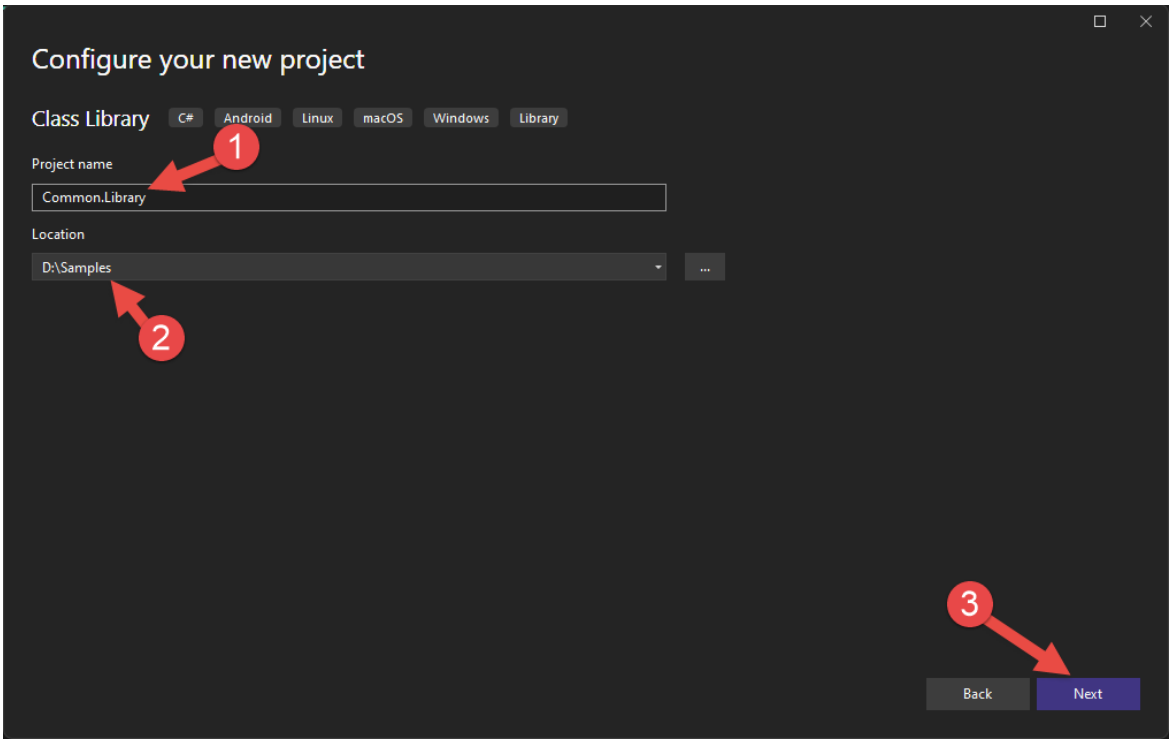
Locate the Class Library that has the description "**A project for creating a class library that targets .NET or .NET Standard**" and click the Next button



Set the Project name to **Common.Library**

Set the Location to a folder next to your OOPLab. In the first lab I created my OOPLab in the D:\Samples\OOPLab folder, so I am setting the Location to **D:\Samples**.

Click the Next button



On the next screen ensure the Framework version is .NET 6.0 (Long term support) and click the Create button.

In the new class library you created, delete the **Class1.cs** file.

Move Files and Folders Around

You can now drag and drop the following folders from the OOPLab project into the new Common.Library project.

- HelperClasses
- Interfaces

In the OOPLab project, delete the **HelperClasses** folder.

In the OOPLab project, delete the **IRepository.cs** file from the **Interfaces** folder.

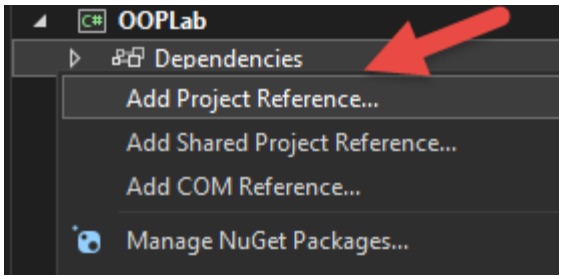
In the Common.Library project delete the **IPerson.cs** file from the **Interfaces** folder.

You now have classes in the Common.Library that are completely generic and can be used from any type of application.

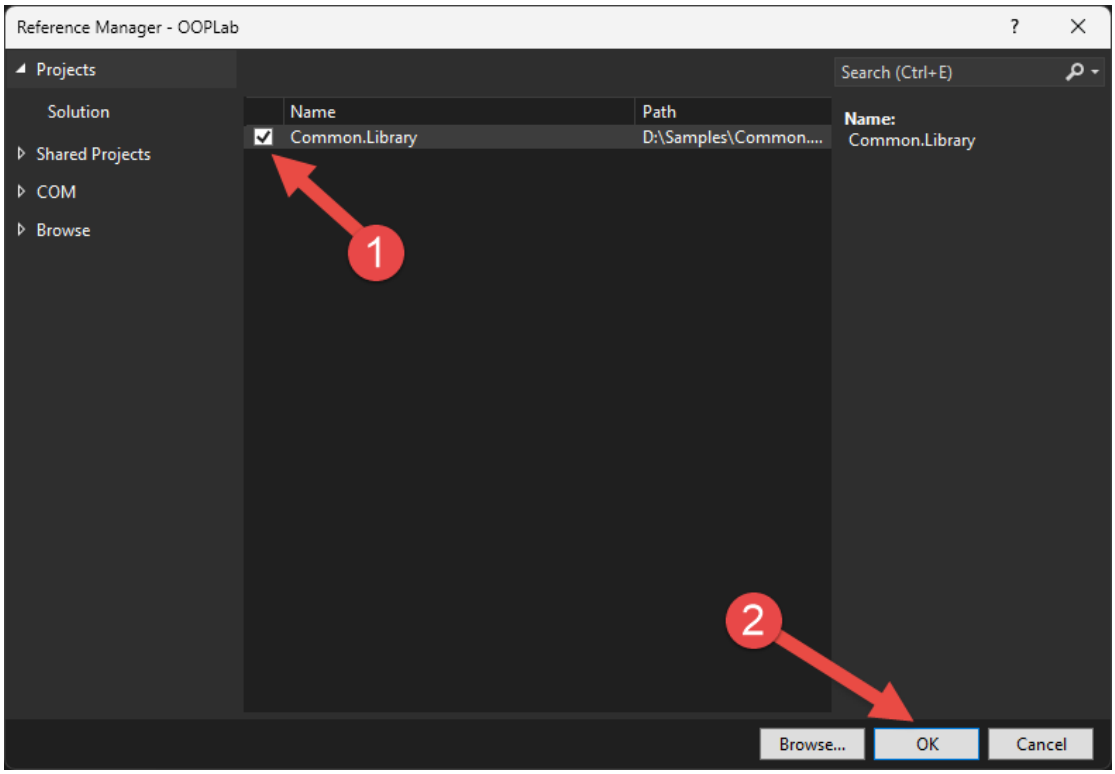
Set a Dependency

Since you now removed the classes from the OOPLab folder and put them into a different assembly, you need to set a reference to the new Common.Library.

Right mouse-click on the Dependencies folder of the OOPLab project and select **Add Project Reference…**



Check the box next to the Common.Library and click the OK button.



Try it Out

Run the application and ensure everything still works.

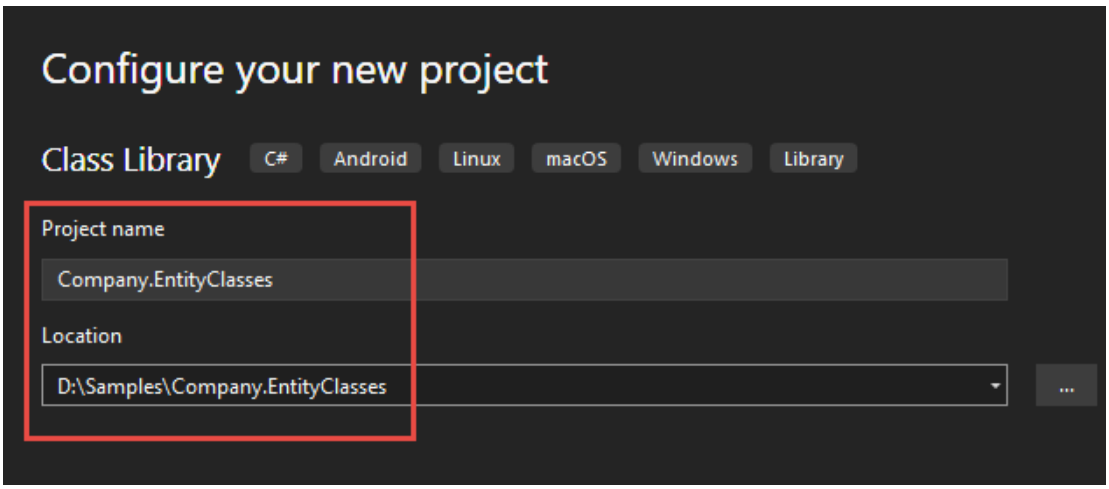
Lab 2: Create an Entity Classes Library

Right mouse-click on the OOPLab solution and select **Add | New Project…**

Choose the Class Library project again.

Set the Project name to **Company.EntityClasses**. Normally, I would use my URL for my company for any company-wide class libraries, so I would use

PDSA.EntityClasses. I would recommend you do the same. I am just keeping the name generic for purposes of this course.



Set the Framework version to .NET 6.0 (Long term support) and click the Create button.

In the new Company.EntityClasses library delete the **Class1.cs** file.

Move Files and Folders Around

You can now drag and drop the following folders from the OOPLab project into the new Company.EntityClasses project.

- EntityClasses
- EventArgsClasses
- Interfaces

In the OOPLab project, delete the three folders you copied to the Company.EntityClasses project.

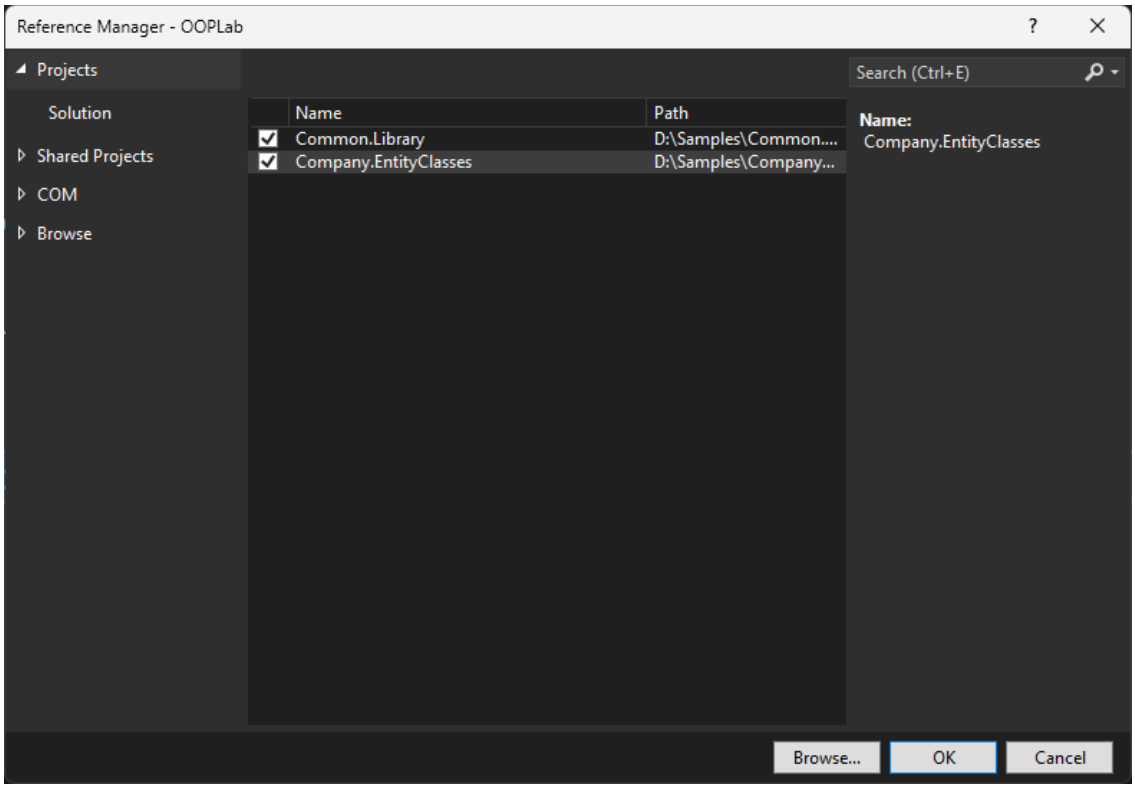
You now have classes in the Company.EntityClasses project that can be used from any type of application you are developing within your company.

Set a Dependency

Since you now removed the classes from the OOPLab folder and put them into a different assembly, you need to set a reference to the new Company.EntityClasses.

Right mouse-click on the Dependencies folder of the OOPLab project and select **Add Project Reference…**

Select the new Common.EntityClasses library from the Reference Manager window and click the OK button.



Try it Out

Run the application and ensure everything still works.

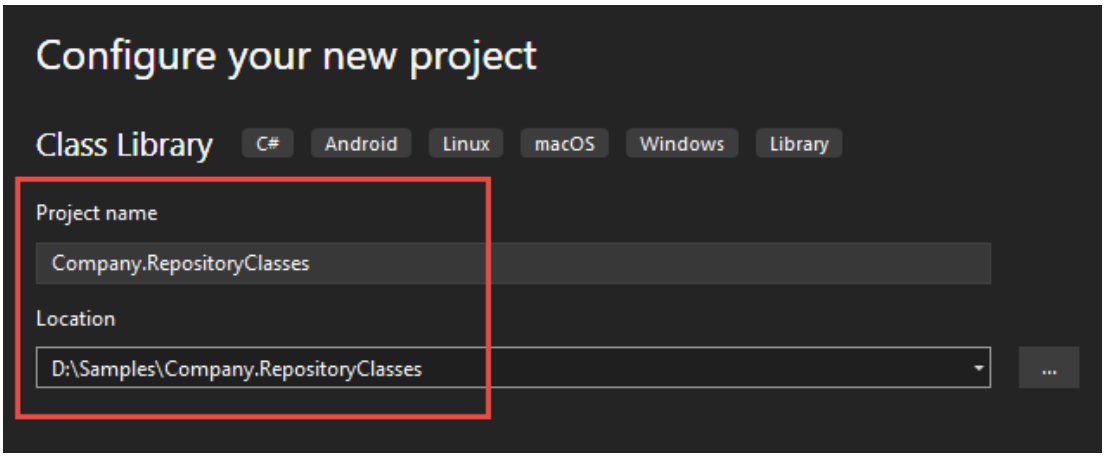
Lab 3: Create a Repository Classes Library

The only folder you have left in your OOPLab project is the RepositoryClasses folder. The two repository classes in this folder should also be broken out into their own class library.

Right mouse-click on the OOPLab solution and select **Add | New Project…**

Choose the Class Library project again.

Set the Project name to **Company.RepositoryClasses**. Normally, I would use my URL for my company for any company-wide class libraries, so I would use PDSA.RepositoryClasses. I would recommend you do the same. I am just keeping the name generic for purposes of this course.



Set the Framework version to .NET 6.0 (Long term support) and click the Create button.

In the new Company.RepositoryClasses library delete the **Class1.cs** file.

Move Files and Folders Around

You can now drag and drop the following folders from the OOPLab project into the new Company.RepositoryClasses project.

- RepositoryClasses

In the OOPLab project, delete the folder you copied to the Company.RepositoryClasses project.

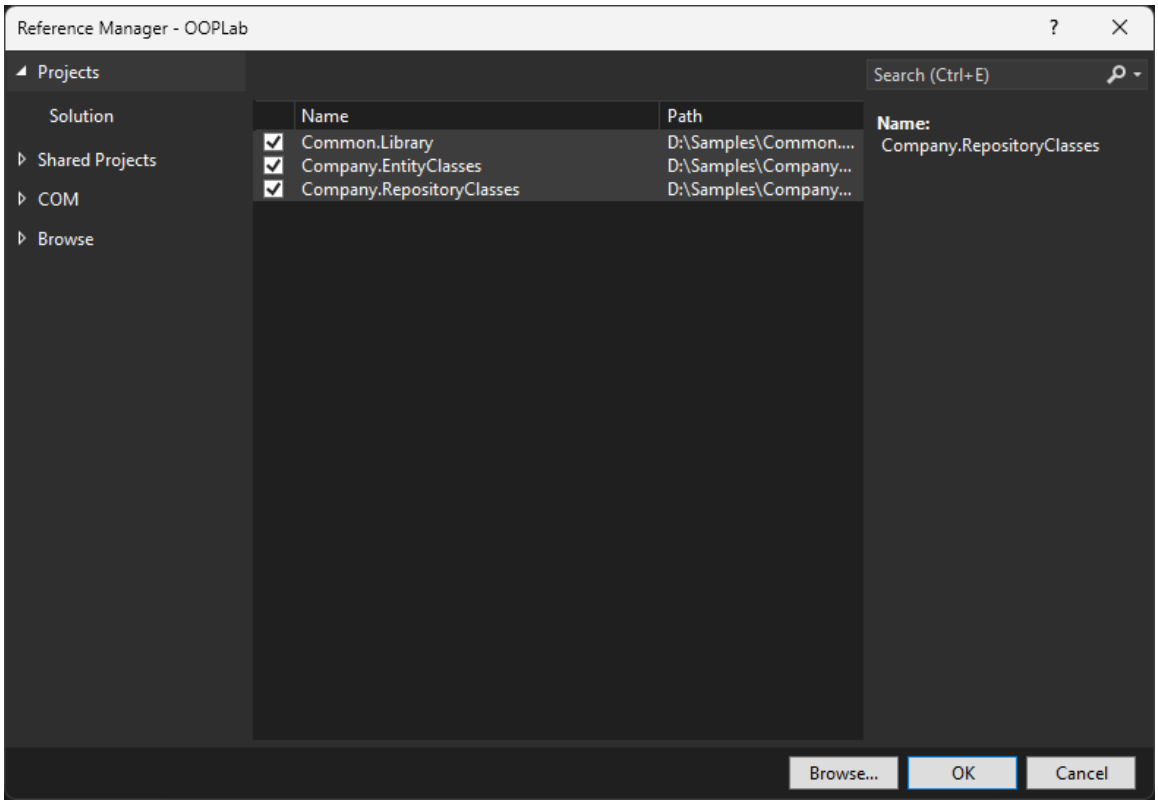
You now have classes in the Company.RepositoryClasses project that can be used from any type of application you are developing within your company.

Set a Dependency

Since you now removed the classes from the OOPLab folder and put them into a different assembly, you need to set a reference to the new Company.RepositoryClasses.

Right mouse-click on the Dependencies folder of the OOPLab project and select **Add Project Reference…**

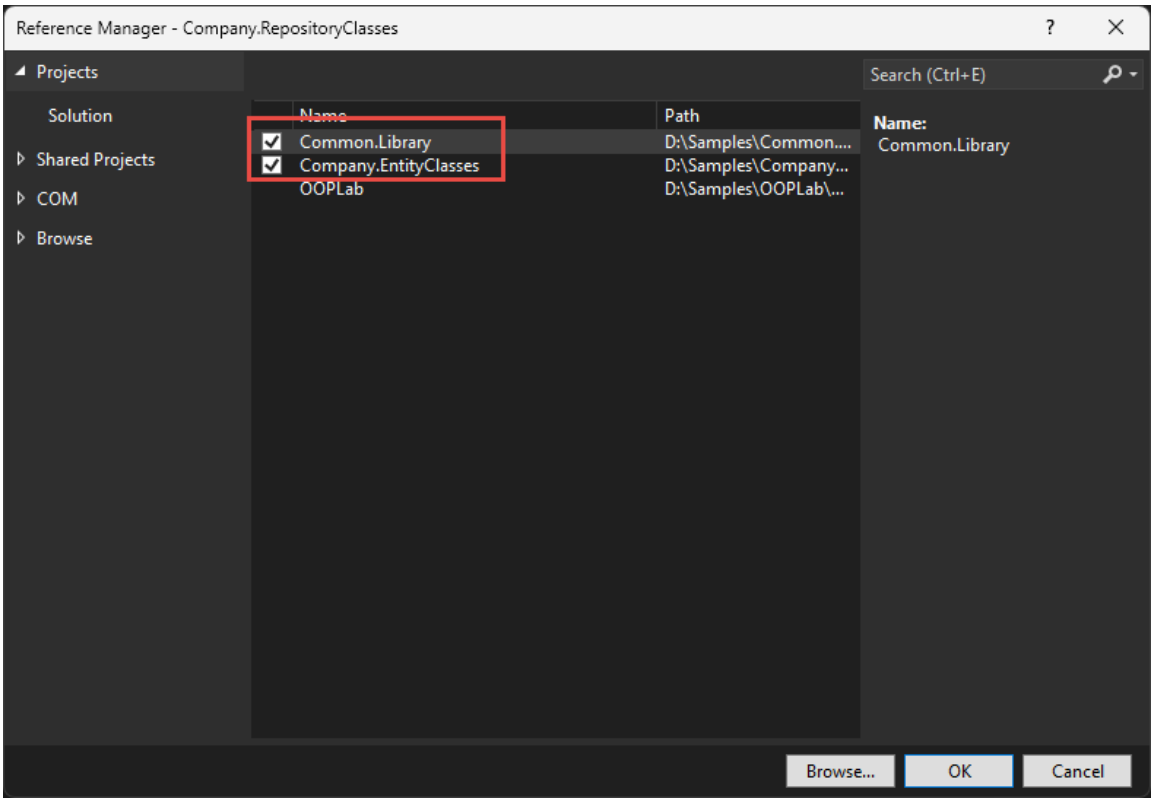
Select the new Common.RepositoryClasses library from the Reference Manager window and click the OK button.



Set Dependency in Repository Classes Library

Right mouse-click on the Dependencies folder of the **Company.RepositoryClasses** project and select **Add Project Reference…**

Make sure both the Common.Library and Company.EntityClasses libraries are selected in the Reference Manager window and click the OK button.

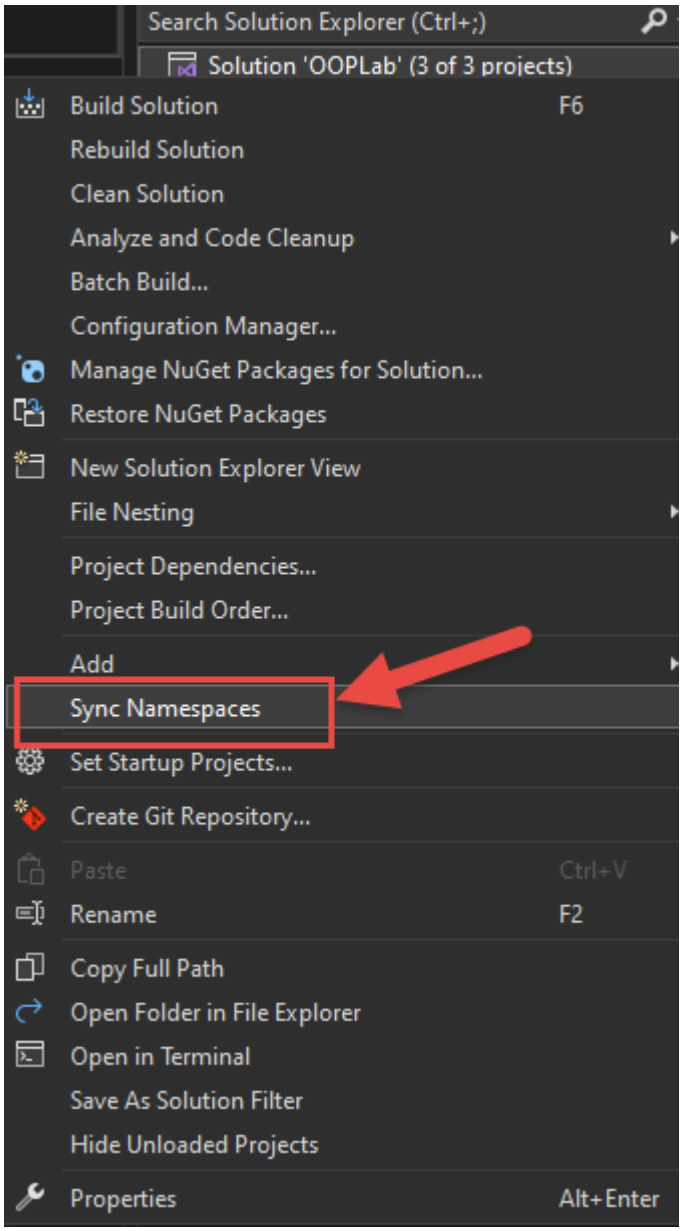


Try it Out

Run the application and ensure everything still works.

Lab 4: Sync Namespaces

When you create a class using Visual Studio it uses the assembly name plus the name of the folder under which you are creating the class for the default namespace. When you first created all the classes in the OOPLab project, the namespace for each class was simply OOPLab. However, now that you have moved the classes to different assemblies, it would be best to change the namespaces on each of those classes to use the assembly and folder in which they are located. Don't worry, you do not need to go through class and set each one individually. Instead, right mouse-click on the Solution and choose the **Sync Namespaces** menu as shown in the following screen shot.



If you now go look at each class, you will see that each namespace has been updated.

Try it Out

Run the application and ensure everything still works.