Class Libraries:

A class library is a collection of prewritten code stored in a separate file – usually a DLL (Dynamic Link Library) file. Using class libraries simplifies maintenance (patching software), reduces project size, and promotes code reuse.

Creating a Class Library Project:

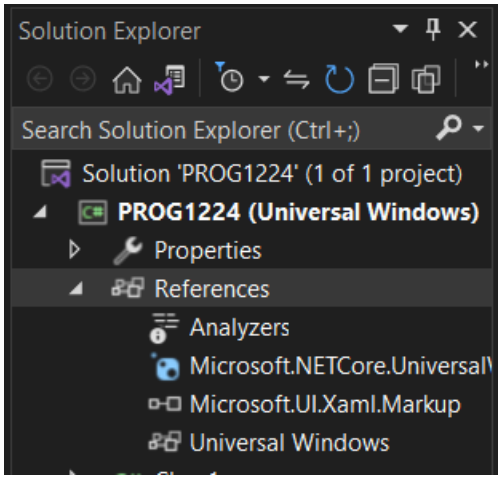
A) Create a new project → Filter C# and Library → C# → Choose Library to support the product you're creating such as **Class Library (Universal Windows)** supports **Blank App (Universal Windows).**

Note: Some Libraries support multiple project types and others are specific to one project.

B) The class library project can either be added to a current solution or you can start a new solution just for the class library. Separate assembly for Libraries supports multiple projects and makes it easier to patch software.

C) Name the Class Library using camel notation and follow best practices for naming conventions. An industry standard is to use a company's domain name in the Library Project/Namespace.

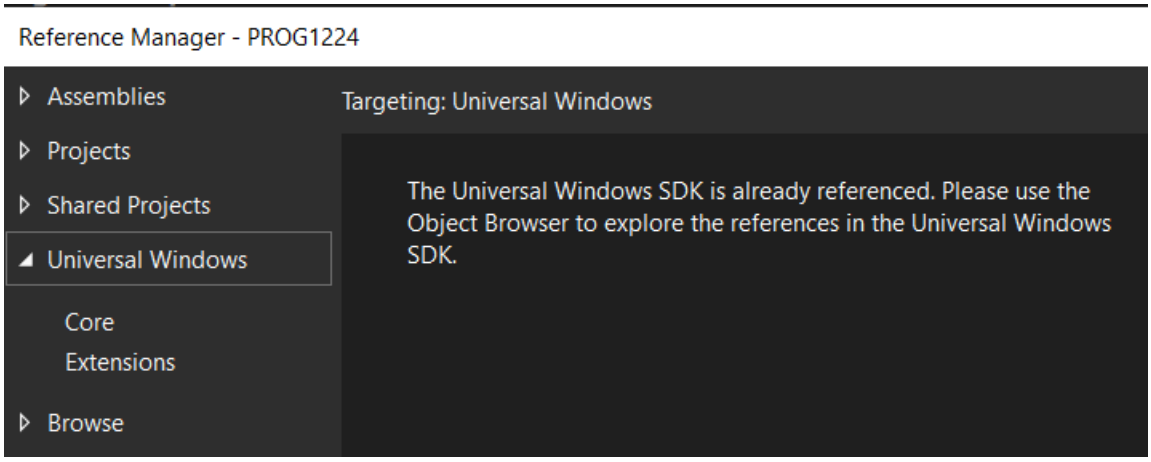
Adding a Microsoft Library to a project:



If any of your classes are to work with UWP, ensure you have a reference to the Universal Windows DLL file…

A) Right-click on References in the Solution Explorer window and select Add Reference.

B) In the Reference manager under Assemblies or Universal Windows Extensions you will notice many Libraries available that can be added to a project. If you add a library add a using statement at the top of each source file that requires access to library member.



C) Right-click on a Library in Solution explore and select the option to View in Object Browser to discover contents of the library.

Adding Code to a Class Library:

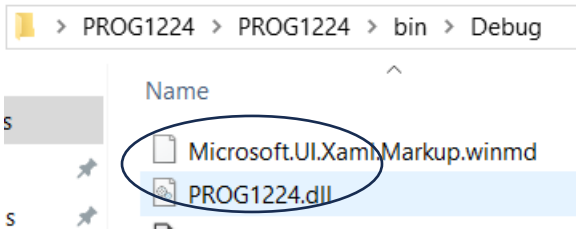
A) A default Library project includes a class. Rename this class and source file. Add folders and organize/move classes. Add additional classes to the library project as needed. Project → Add Class **OR** right-click on the project and select New Item or class from the menu.

B) Write the code in the library to support application (business rules) logic. Build the project. Debug and fix errors. Compile to obtain the **DLL** output file.

Note: You will not be able to run a library project.

C) In another project that is compatible with your Library add a reference to the library – the DLL file. In UWP projects add under the **References** in Solution Explorer. In other MS Core projects add under Dependencies in Solution Explorer.

D) Using the **Reference Manager** select the **Browse** button to locate your Library assembly. Copy the DLL file from the Library Project Folder within the bin\Debug folder. For example, in a Library project named PROG1224:



E) Add a using statement at the top of each source file that will need access to the code in the library. For example:

using PROG1224;

F) Write code in your application to test/use the library.

Note: When both the Library and Client Application projects are opened Visual Studio copies new versions of the compiled Library over to the Client as changes are made. If both projects are not in the same assembly, you can open multiple instances of visual studio having both the Library and the Client application opened at the same time on your machine.

If you make changes to your library and they do not reflect in the client application, make sure you Rebuild the library to compile a new version of the DLL. Check in solution explorer that you do not see a warning symbol! beside the library. You can remove and add a library back into a project to refresh changes you have made.