1. The Basics: Creating and Using a Namespace

Goal: Define a class inside a custom namespace and access it from the Program class.

- Create a file where a class named Calculator is wrapped inside a namespace called MathUtilities.
- Inside Calculator, add a method Add(int a, int b).
- In your Main method, call the Add method **without** using a using directive (use the fully qualified name).

2. The using Directive

Goal: Simplify code by importing namespaces.

- Take the MathUtilities namespace from Exercise 1.
- In your Program.cs, add the appropriate using statement at the top of the file.
- Rewrite the Main method to instantiate the Calculator without typing out the full namespace path.

3. Nested Namespaces

Goal: Understand hierarchical organization.

- Create a structure for a business application:
    - Outer namespace: Corporate
    - Inner namespace: HR
- Inside HR, create a class Employee with a property Name.
- In a separate namespace (Corporate.Accounting), try to instantiate an Employee.
- **Bonus:** Try using the "file-scoped namespace" syntax (e.g., namespace Corporate.HR;) instead of curly braces.

4. Resolving Naming Conflicts (Aliases)

Goal: Use two different classes that share the exact same name.

- Imagine you are using two different UI libraries.
- Create namespace LegacyUI with a class Button.
- Create namespace ModernUI with a class Button.
- In your Main method, use **namespace aliasing** (e.g., using MUI = ModernUI;) to create one button from each namespace without the compiler getting confused.

5. Global Using Directives

Goal: Use C# 10+ features to clean up multiple files.

- Create a project with three different files.
- Create a GlobalUsings.cs file.
- Use the global using keyword to import System.Collections.Generic so that every file in your project can use List<T> without needing a using statement at the top of every single file.