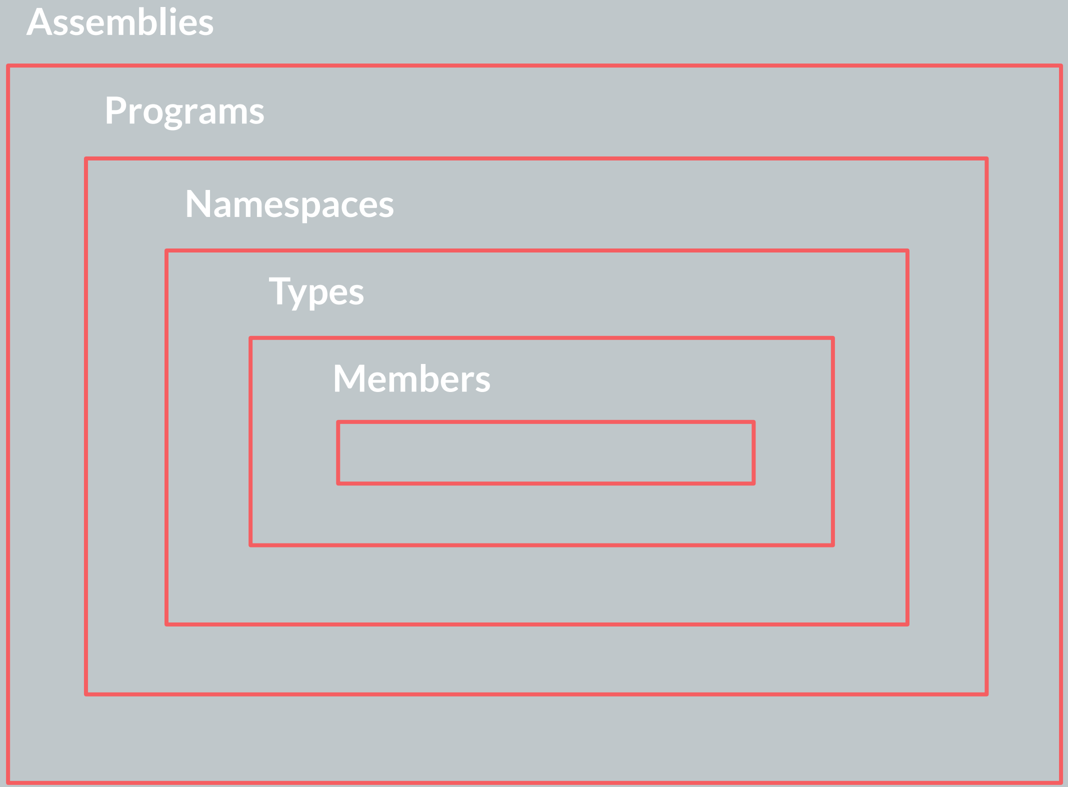
Structure of a C# Program

Christie Thammavong

Key Organizational Concepts

- Assemblies
    - Applications, .exe
    - Libraries, .dll
- Programs
- Namespaces
- Types
- Members

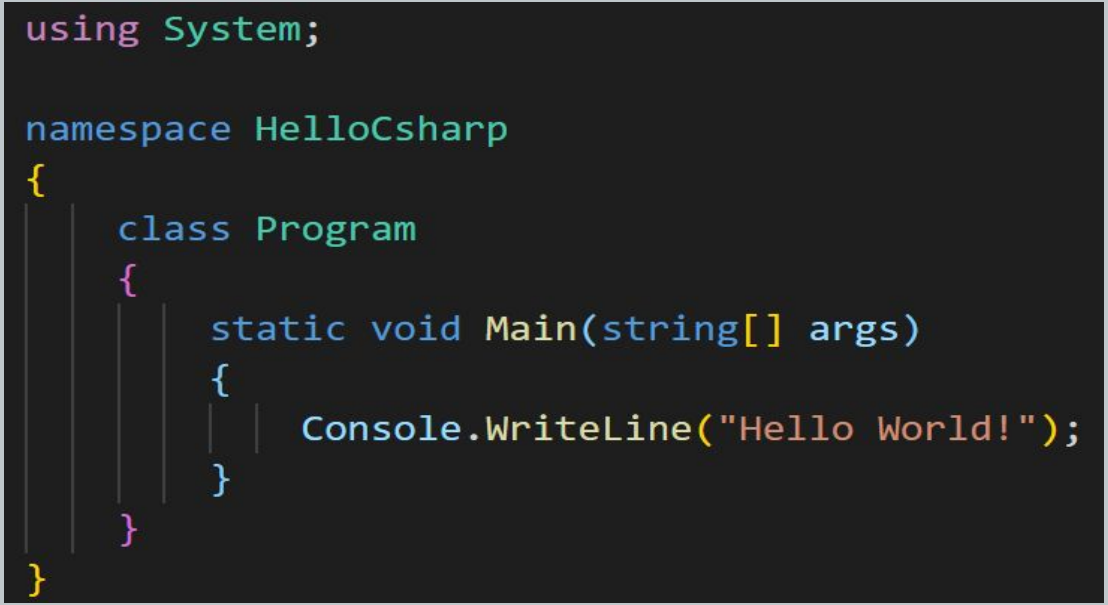


Assemblies

- Assemblies form the fundamental units of deployment, version control, reuse, activation scoping, and security permissions for .NET-based applications
- An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality
- An assembly is essentially a project that has been compiled (.exe, .dll)
- Example: different layers (UI, business, context, models)

Programs

- C# programs consist of one or more files. Each file contains zero or more namespaces



```csharp
using System;

namespace HelloCsharp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Namespaces

- .NET uses namespaces to organize classes
    - Example: System.Console.WriteLine(“Hello World!”);
    - System is a namespace, Console is a class in that namespace
- Declaring your own namespaces can help you control the scope of class and method names in larger programming projects

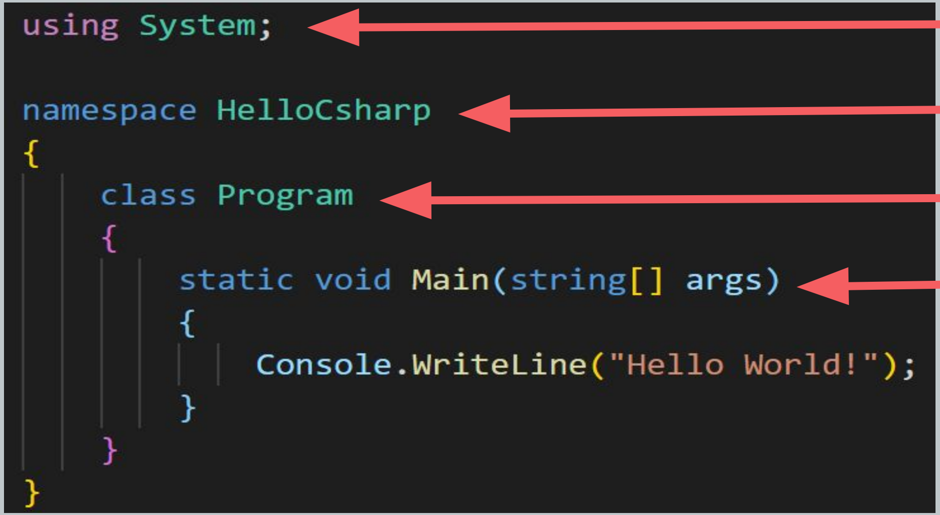
Types

- Programs declare types
- Types can be organized into namespaces
- Examples: classes, structs, interfaces
    - These types are custom types
- Built-in types include integers, booleans, strings, objects

Members

- Types contain members that represent their data and behavior
- Examples: fields, methods, properties, events

Structure Overview



```
using System;

namespace HelloCsharp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

References to .NET Namespace

Our namespace

Class (type)

Method (member)

The Main method is the entrypoint to a C# program

We could declare a field called ‘greeting’ and set an initial value of “Hello World!” and pass the field into WriteLine(). This field would be another member of the class

public string greeting = “Hello World!”;
Console.WriteLine(greeting);

Best Practices

- Good layout
    - Write only one statement per line
    - Write only one declaration per line
    - Add at least one blank line between method definitions and property definitions
    - Use parentheses to make clauses in an expression apparent
- Architecture Design Level
    - Implement loosely coupled architecture using interfaces and abstract class
    - Use of generics would help you to make reusable classes and functions
    - Separate your application into multiple assemblies (UI, Business, Data Access)
    - Design Patterns
        - Creational, Structural, Behavioral
            - Singleton Design Pattern: the class has only one instance and provides a global point of access to it
        - Repository Pattern: encapsulate the logic required to access data sources

References

C# docs - get started, tutorials, reference.

C# Coding Guidelines And Best Practices v1.0