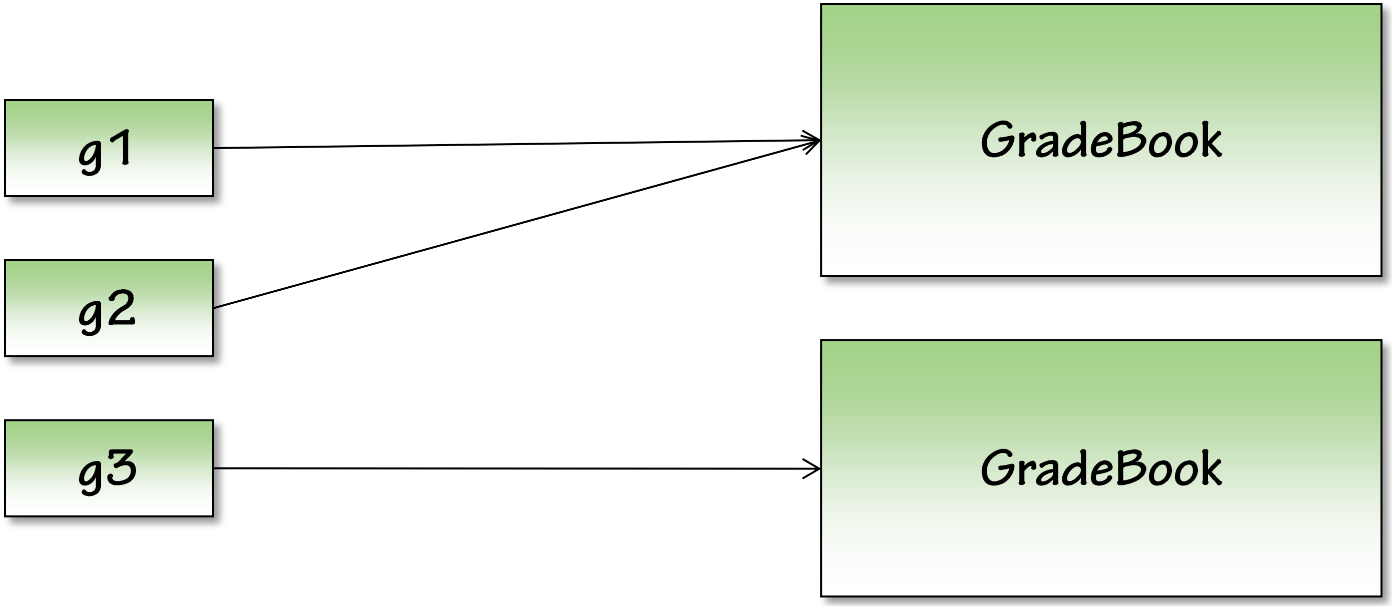
Types and Assemblies

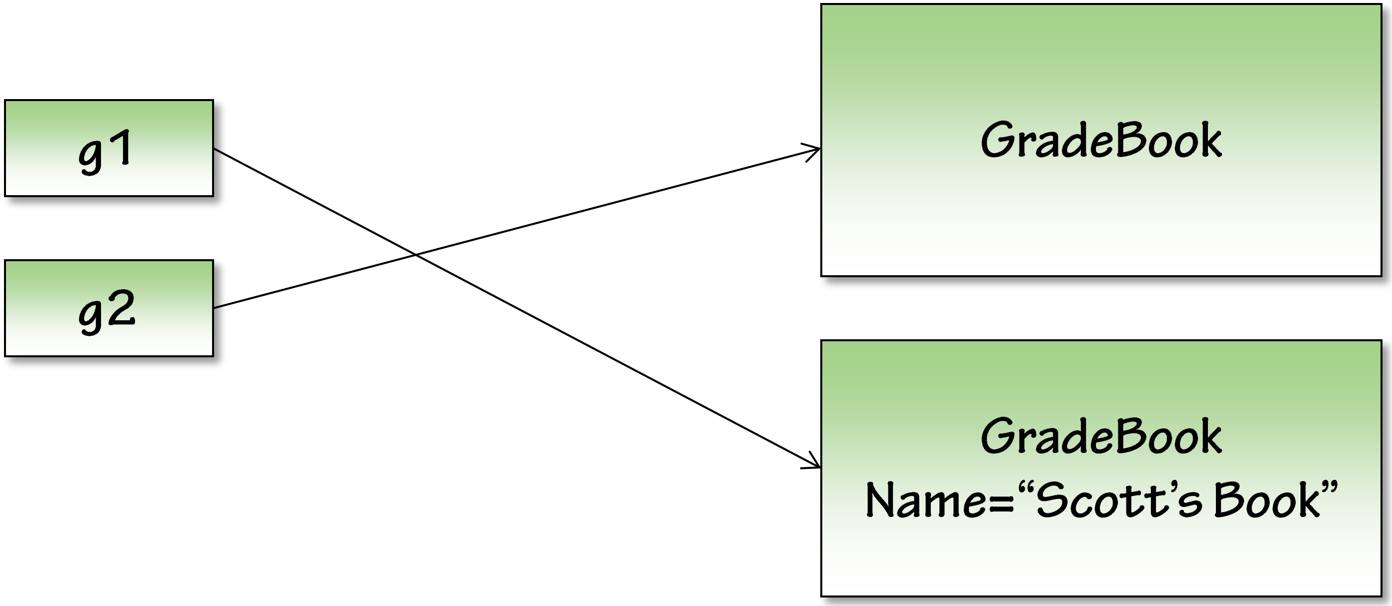
K. Scott Allen
odetocode.com
@OdeToCode

pluralsight
hardcore dev and IT training

Reference Types

- **Variables store a reference to an object**
  - Multiple variables can point to the same object
  - Single variable can point to multiple objects over it’s lifetime
  - Objects allocated into memory using new





g1

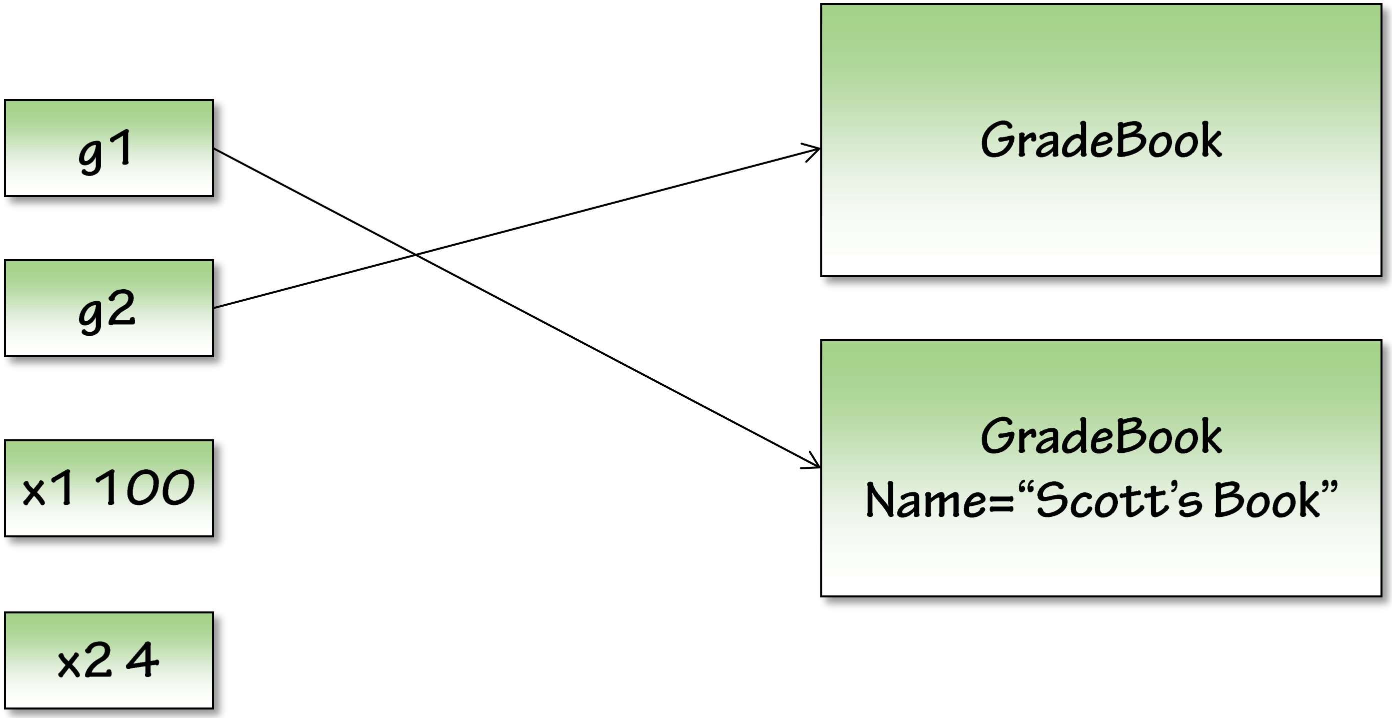
g2

GradeBook

GradeBook
Name="Scott's Book"

Value Types

- **Variables hold the value**
  - No pointers, no references
- **Many built-in primitives are value types**
  - Int32, DateTime, double, float



g1

g2

x1 100

x2 4

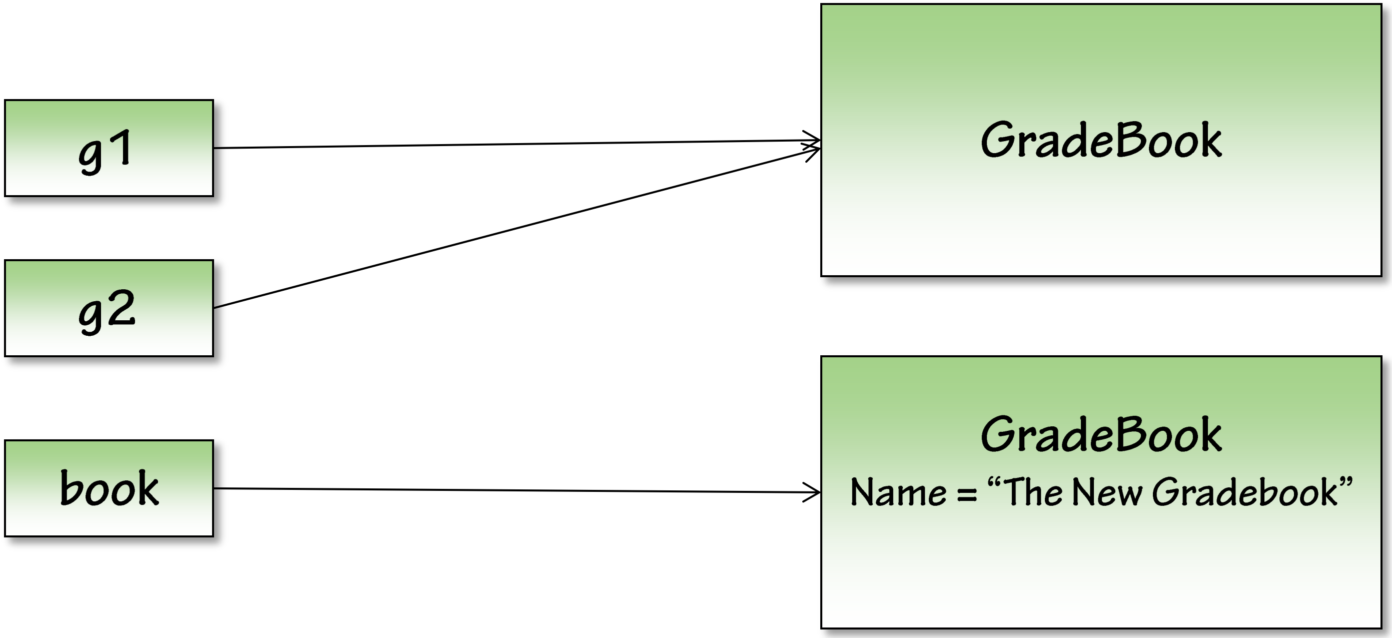
GradeBook

GradeBook
Name="Scott's Book"

Method Parameters

- **Parameters pass “by value”**
  - Reference types pass a copy of the reference
  - Value types pass a copy of the value

```
public void DoWork(GradeBook book)
{
    book.Name = “Grades”;
}
```


g1
g2
book
GradeBook
GradeBook
Name = “The New Gradebook”

Creating Value Types

- **struct definitions create value types**
  - Should represent a single value
  - Should be small

```
public struct DateTime
{
    // ...
}
```

Enumerations

- **An enum creates a value type**
  - A set of named constants
  - Underlying data type is int by default

```
public enum PayrollType
{
    Contractor = 1,
    Salaried,
    Executive,
    Hourly
}
```

```
if(employee.Role == PayrollType.Hourly)
{
    // ...
}
```

Immutability

- **Value types are usually immutable**
  - Can not change the value of 4
  - Can not change the value of August 9th, 2002

```
DateTime date = new DateTime(2002, 9, 11);
date.AddDays(1)

string name = " Scott ";
name.Trim();
```

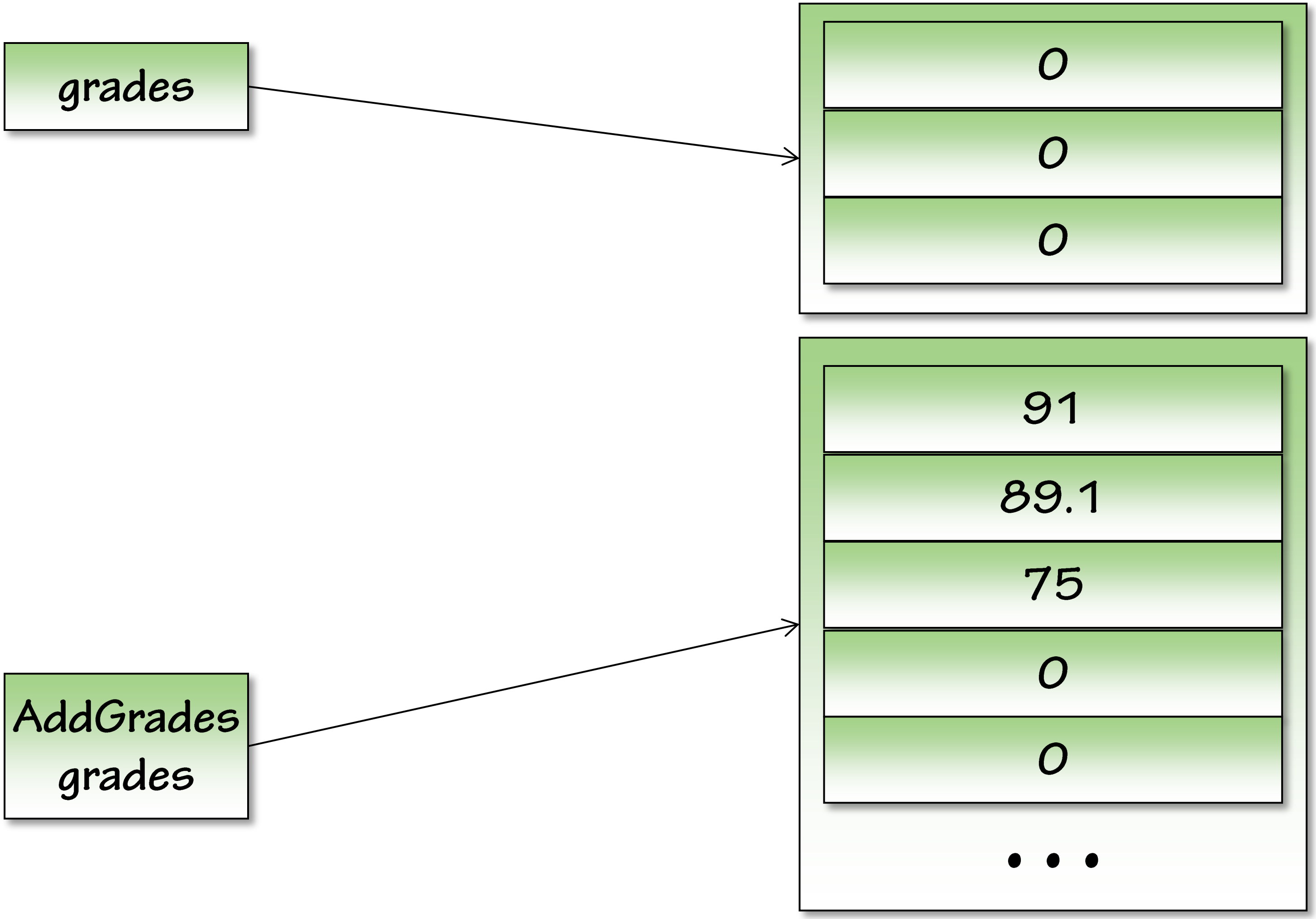
Arrays

- **Manage a collection of variables**
  - Fixed size
  - Always 0 indexed

```
const int numberOfStudents = 4;
int[] scores = new int[numberOfStudents];

int totalScore = 0;
foreach(int score in scores)
{
    totalScore += score;
}

double averageScore = (double)totalScore / scores.Length;
```



grades

0

0

0

91

89.1

75

0

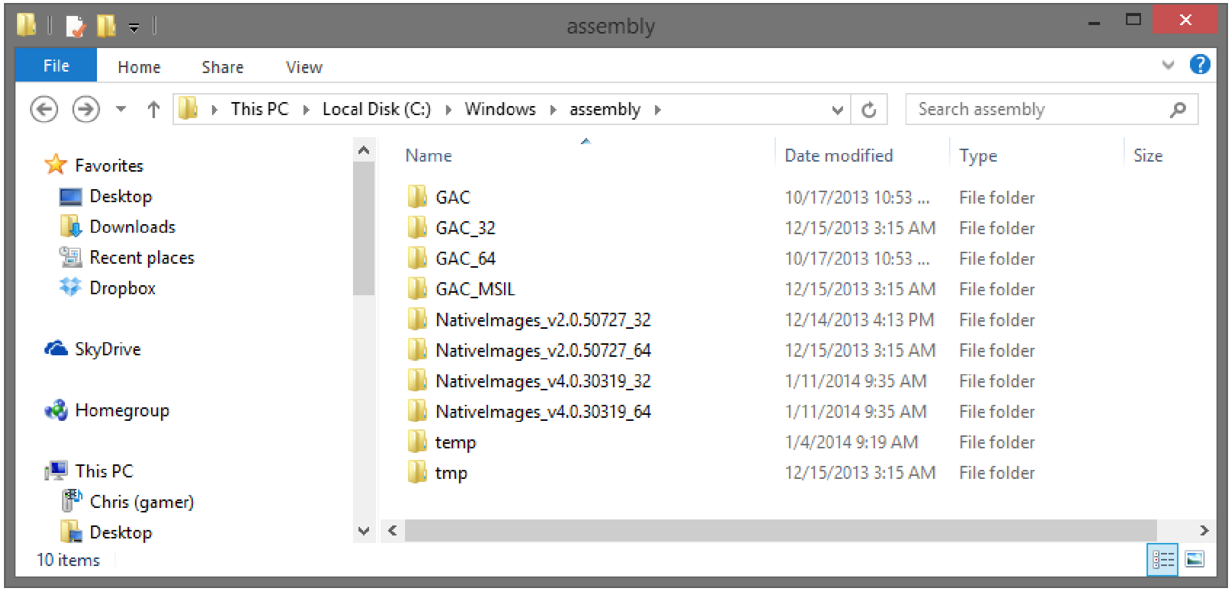
0

• • •

AddGrades
grades

Assemblies

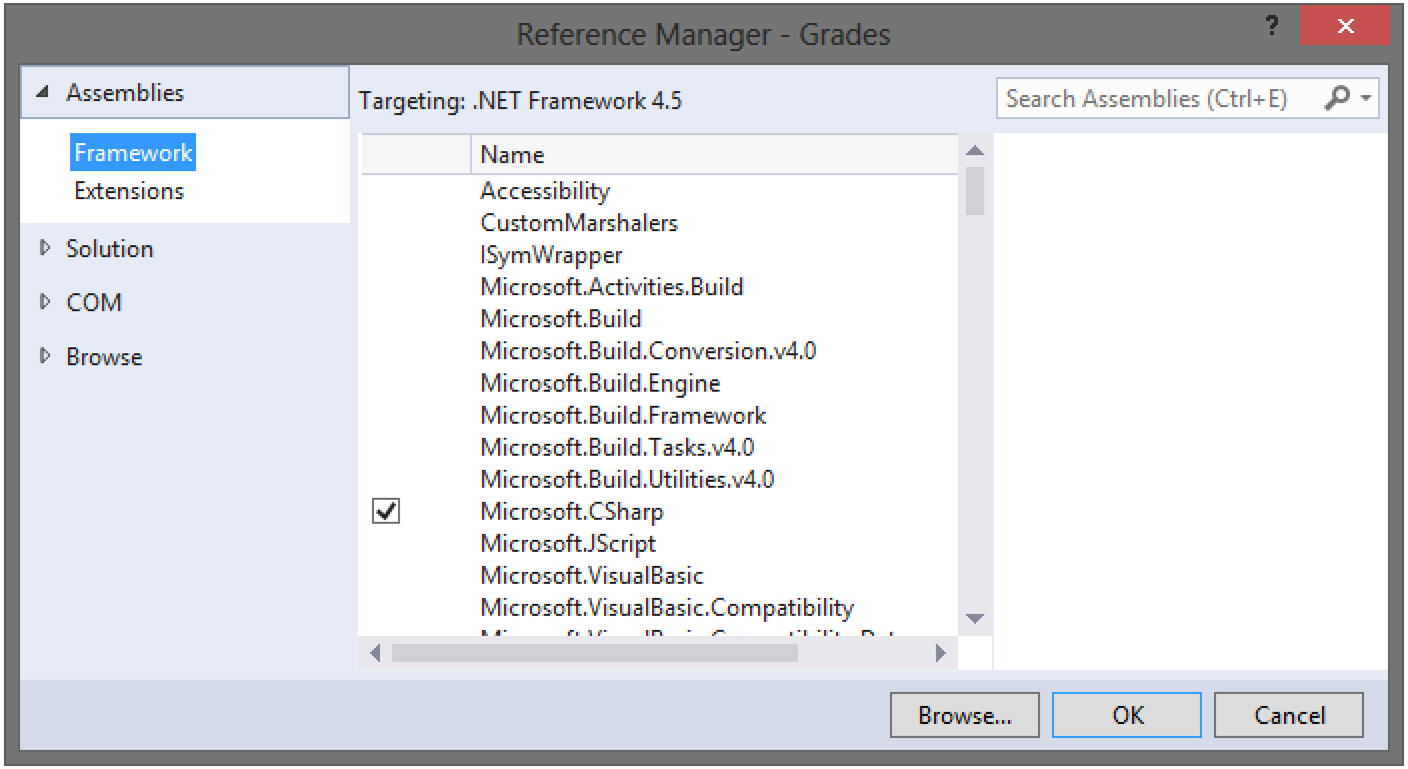
- **Assemblies are .exe or .dll files**
  - Contain metadata about all types inside
- **Global Assembly Cache**
  - A central location to store assemblies for a machine



Name	Date modified	Type	Size
GAC	10/17/2013 10:53 ...	File folder	
GAC_32	12/15/2013 3:15 AM	File folder	
GAC_64	10/17/2013 10:53 ...	File folder	
GAC_MSIL	12/15/2013 3:15 AM	File folder	
NativeImages_v2.0.50727_32	12/14/2013 4:13 PM	File folder	
NativeImages_v2.0.50727_64	12/15/2013 3:15 AM	File folder	
NativeImages_v4.0.30319_32	1/11/2014 9:35 AM	File folder	
NativeImages_v4.0.30319_64	1/11/2014 9:35 AM	File folder	
temp	1/4/2014 9:19 AM	File folder	
tmp	12/15/2013 3:15 AM	File folder	

References

- **Must load assembly into memory before using types inside**
  - Easy approach – reference the assembly in Visual Studio



Targeting: .NET Framework 4.5

	Name
	Accessibility
	CustomMarshalers
	ISymWrapper
	Microsoft.Activities.Build
	Microsoft.Build
	Microsoft.Build.Conversion.v4.0
	Microsoft.Build.Engine
	Microsoft.Build.Framework
	Microsoft.Build.Tasks.v4.0
	Microsoft.Build.Utilities.v4.0
☑	Microsoft.CSharp
	Microsoft.JScript
	Microsoft.VisualBasic
	Microsoft.VisualBasic.Compatibility

Summary

- **Every type is a value type or reference type**
  - Use struct to create a value type
  - Use class to create a reference type
- **Arrays and strings are reference types**
  - Strings behave like a value type
- **Types live in assemblies**