SOFTWARE GUILD

Overview of C# Applications

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

Lesson Goals

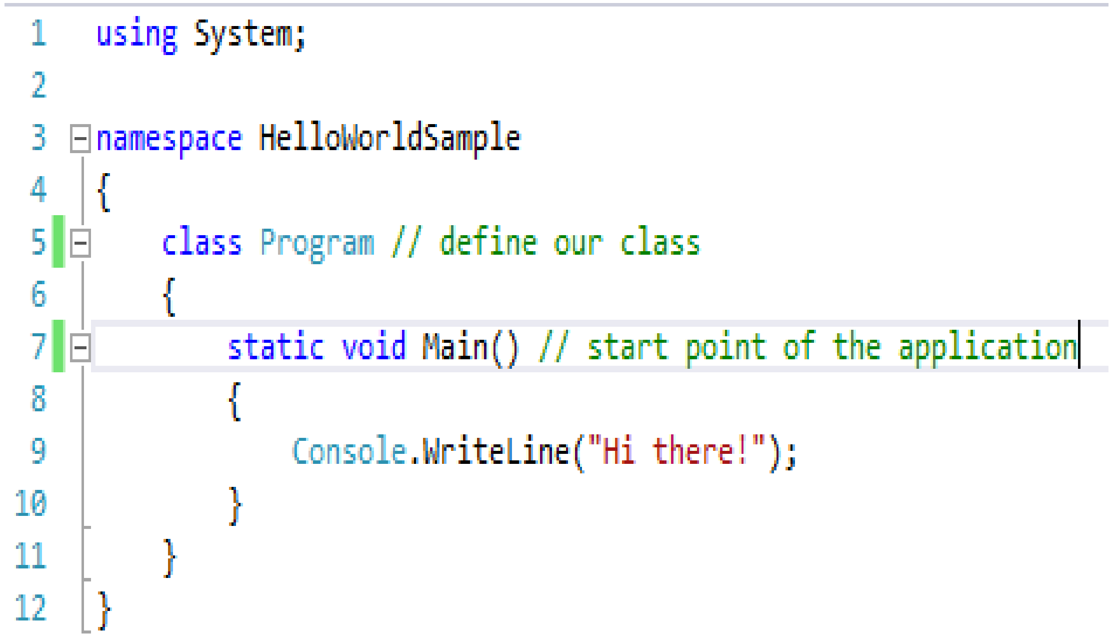
- Demonstrate the basic components of a “Hello World” application
- Demonstrate Namespaces, Comments, and Console Output
- Get Started with Visual Studio

A programming ritual

DEMO – HELLO WORLD

Disassembling the Program

- Line 1: Tells the compiler we are using the System namespace
- Line 3: Declares a new namespace called HelloWorldSample
    - Anything placed between the braces of line 4 and 12 belong to the namespace. (Curly braces in C# are used to group things)
- Line 5: Declares a new Class Type called Program
    - Anything placed between 6 and 11 belong to Program
- Line 7: Declares a Main method, which is a member of Program
    - Main() is special; it is the entry point for a console application
- Line 9: A statement
    - Tells the Console type of the System namespace to write a line of text out. Statements end with semi-colons in C#.



```
1   using System;
2
3   namespace HelloWorldSample
4   {
5       class Program // define our class
6       {
7           static void Main() // start point of the application
8           {
9               Console.WriteLine("Hi there!");
10          }
11      }
12  }
```

Key Points About the Program

- A C# program consists of one to many type declarations. The way you organize and call type members is what programming in C# is all about.
- A namespace is a way for C# to organize types so the compiler knows where to look for the type definition you call in code. Thus, it is possible to have two types with the same name in different namespaces.
  - Example: You could have a Connection class in the Oracle Namespace and Connection in the SqlServer Namespace.
  - If you don’t have a using statement, you must “fully qualify” your call. So instead of Console.WriteLine(“”) we could say System.Console.WriteLine(“”).

Identifiers

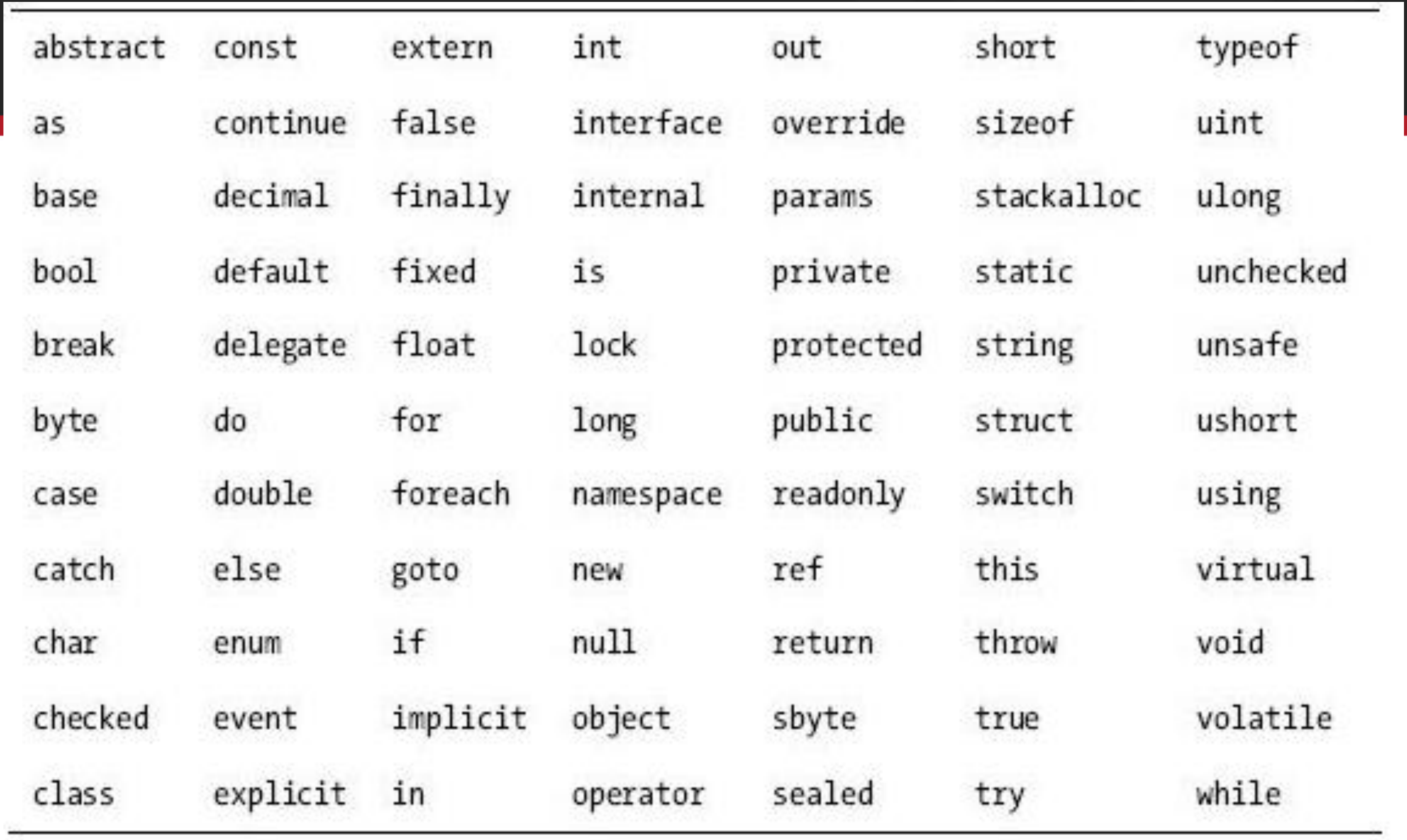
- Identifiers are names we create to name things like types, variables, parameters, etc.
- Identifiers should be descriptive! (ex: FirstName, LastName, StreetAddressLine1, Player1Score)
- Identifiers must start with an [a-z] character and they may include numbers in anything but the first position.
- No special characters! (only: a-z, 0-9, _ (underscore))
- You can use @ as the first character if you want to make a variable name the same as a reserved word, but this is generally frowned upon.

Identifiers in C# Are Case Sensitive

myInteger MyInteger myinteger

- These are all different identifiers in C#, and you could create all 3 in the same method.
- Pro tip: Don’t do this or the developer who maintains the code after you will hunt you down. It makes debugging very difficult.

C# Keywords



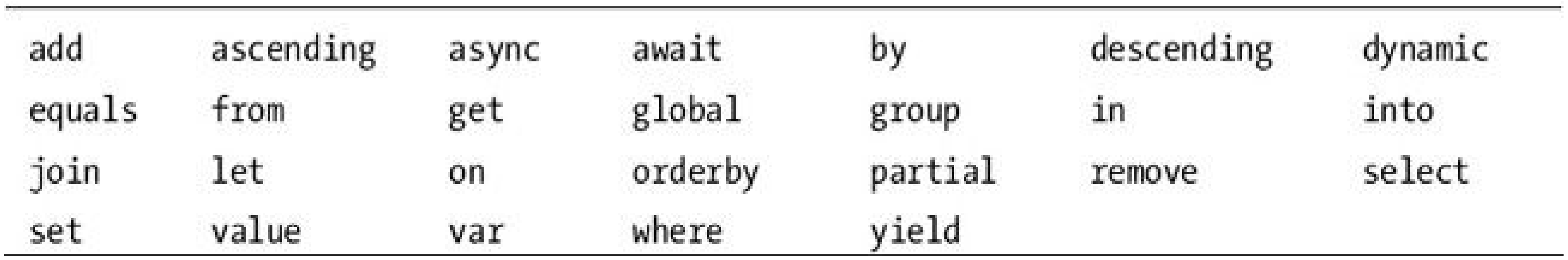
abstract	const	extern	int	out	short	typeof
as	continue	false	interface	override	sizeof	uint
base	decimal	finally	internal	params	stackalloc	ulong
bool	default	fixed	is	private	static	unchecked
break	delegate	float	lock	protected	string	unsafe
byte	do	for	long	public	struct	ushort
case	double	foreach	namespace	readonly	switch	using
catch	else	goto	new	ref	this	virtual
char	enum	if	null	return	throw	void
checked	event	implicit	object	sbyte	true	volatile
class	explicit	in	operator	sealed	try	while

Keywords Continued

- Cannot be used as variable names unless prefaced with @
- Always lowercase, colored dark blue by default in Visual Studio

Contextual Keywords

- These only become keywords in certain contexts.
- They may be used as variable names (but typically, we avoid doing that).



add	ascending	async	await	by	descending	dynamic
equals	from	get	global	group	in	into
join	let	on	orderby	partial	remove	select
set	value	var	where	yield		

Whitespace

- Spaces, tabs, carriage returns, etc. constitute whitespace
- Some languages (like Python) care very much about whitespace
- C# does not care about whitespace, so use it liberally to improve the readability of your code.
- Strive for consistency in your codebase.

Statements

- Statements in C# must be terminated with a semi-colon ;
- A general way to think of a statement is a command to be executed (things like type definitions are not terminated with ;).
- Because whitespace doesn’t matter, you can put several statements on the same line, but for readability this is frowned upon.

Blocks

- As mentioned earlier, curly braces { } define blocks of code.
- Blocks are used for organization purposes: namespaces, class definitions, loop/conditional logic containers, etc.
- Blocks show that code is related. Visual Studio will typically indent code inside a block to give a visual representation of the block.

Code Comments

- Comments are non-executable
- // for a single line comment
- /* for a multi line comment, close with */
- /// near a method or property will create an XML definition that can be used to generate documentation

Visual Studio

- Microsoft IDE (Integrated Development Environment)
- Assists developers and teams with debugging, tracing, compilation, deployment, source control, and project management
- You can actually work out of text files and compile using csc.exe, but that is masochistic so we won’t be doing that

Visual Studio Creates Many Types of Projects

- A project is a collection of types that compile into a specified output (.dll, .exe, website, .svc, etc.).
- Multiple projects make up a solution, and typically we separate types into projects for the purposes of reuse and physical boundaries.
- For example: we may have a solution that includes a data access layer project (.dll), a business logic project (.dll), a service project (.svc) and a User Interface (.exe).
- Applications that exist in multiple layers are often referred to as nTier applications.

Demo: Let’s Play With Output

- Console.Write
    - Outputs information to the same line the cursor is currently on.
- Console.WriteLine
    - Appends a carriage return after the output text.
- We can also use \n or Environment.NewLine to write new lines.

SOFTWARE GUILD

Demo: Formatting Strings

- Some output mechanisms in .NET can parse format strings. A format string defines *substitution markers* that will be replaced by a list of parameters in the statement.
- Substitution markers denote a parameter position in curly braces and must start with 0.

Demo: Alignment and Format Codes

- Numeric parameters can be specially formatted using the following syntax:

  {index, alignment :format}

- For example: {0:C} displays a number in currency format

Alignment

- The alignment number is the minimum length of characters to display. A negative number is left aligned and a positive number is right aligned.
- If an input is bigger than the alignment, it will overflow the space (but not cause an error).

Standard Numeric Formatters

- Currency (C, c)
- Decimal (D, d)
- Fixed Point (F, f)
- General (G, g)
- Hexadecimal (X, x)
- Number (N, n)
- Percent (P, p)
- Round-Trip (R, r)
- Scientific (E, e)

Fun with strings

DEMO

Other Useful String Methods

- Contains
- StartsWith/EndsWith
- Format
- IndexOf
- IsNullOrEmpty
- IsNullOrWhiteSpace
- LastIndexOf
- Remove
- Replace
- Split
- Substring
- ToLower/ToUpper
- Trim/TrimStart/TrimEnd