**Lesson 2 Notes**

**Assembly –** a deployment unit. Represents one file on disk. If you have 4 projects in your solution, there will be 4 assemblys.

**Namespace**

You can have same types within different assemblies. Problem is when you try to use two different assemblies with the same type. To limit the possibility of this, you use namespaces. It’s sole purpose is for scoping.

A type is unique as far as it’s fully qualified name is unique. Fully qualified name is namespace + type name. The compiler always works with fully qualified namespaces.

Namespaces are always pascal cased. They follow identifier rules.

The defacto standard is: company name.product[.subnamespaces]

Company types are included in company namespace. Product types are in the product namespace.

Generally, the namespace name follows the assembly name.

The compiler always required the fully qualified type name. The compiler has a shortcut so you don’t have to use the FQTN every time.

Using name;

Now you can just use ‘product’ in your code.

Example. Int32 is not the FQTN. It is actually System.Int32. So you would put ‘using System’ a the top of the page. They are USING DECLARATIONS.

Using Keyword followed by Namespace name;

Namespace Nile.Host (contains two namespaces… Nile… and sub namespace Host).

To create a namespace, you create a namespace statement.

Namespace ‘name of namespace’

{

}

You can also nest namespaces; but, it isn’t advisable.

Projects talk to each other through assemblies

Declaring a class:

Class T

{

}

This declares the existence of this type with this name. Instead of the curly braces are the members of the class.

Class

Instance/Object

A class is a template (a blueprint). It describes functionalities.

An Instance is the actual creation of that class (type).

Rectangle

Length

Width

A Rectangle class would declare that there is a length and a width. You could also have a function to calculate the area given the length and width.

You could have a function to draw the rectangle.

The new keyboard takes a type and creates and instance of it in memory. So your variable is storing an instance of the class.

Classes are all about grouping together the functionality and properties of something.

Classes are almost always nouns. Are always pascal case. The fields are camel cased.

Class definition

[Access] class T

Types can only be one of two accessibility types. **Public or Internal.**

**Accessibility Types:**

**Public –** anyone can use it.

**Internal –** anything in the assembly can use it.

**Private –** only the type (class) can use it.

Accessibility determines who has access at compile time.

**Variables that are a part of a class are referred to as fields**

**(Class Members)**

**Fields**

**Properties  
Methods**