# Session 14

Programming Recap and Materials  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Namespaces

Namespaces are tools that C# provides to separate and organise your program. Namespaces control what classes and methods another class can access.

There are different ways you can attach or add a namespace to a class, and it is different depending on what you would like to do.

If you want to let a class access a namespace you can use the ‘using’ directive as shown below:

**using UnityEngine;**

Anything after the ‘using’ is the package or namespace that you want to access or import.

If you would like to add a class to a namespace create a ‘namespace’ keyword, include the namespace you want to add it to then you make sure that the class is within the scope (inside the brackets) of the namespace. This can be seen below:

|  |
| --- |
| Example 1A |
| // Session14Namespace is the name of the namespace that we want to access namespace Session14Namespace  {  public class Example2 : MonoBehaviour  {  …  }  } |

## Classes

Classes are a type of reference. Usually, they are an object that holds data and manipulates it based on the algorithms it has been given. A class is always set out in a certain order:

1. Reference namespaces first
2. Declare namespace if required
3. Declare class and inherited classes
4. Public variables are always declared first
5. Private variables are always declared second
6. Then your main methods
7. Then your ‘helper’ methods (Methods that are not widely used in other classes but are used to support other methods)

A class can also have inheritance which means it will ‘inherit’ (or have the same) methods and variables as the class it inherits. Please refer to the example below.

|  |
| --- |
| Example 1B |
| // Implement required namespaces first  using System.Collections;  using System.Collections.Generic;  using UnityEngine;  // Session14Namespace is the name of the namespace that we want to access  namespace Session14Namespace  {  // Note that this class inherits the MonoBehaviour class  public class Example2 : MonoBehaviour  {  ////////////////////////////////////////////////////////////////////////  // Public variables  public int randomNumberOne = 2;  public int randomNumberTwo = 5;  ////////////////////////////////////////////////////////////////////////  // Private variables  private int \_randomNumberThree = 9;  ////////////////////////////////////////////////////////////////////////  // Main methods  void Start()  {  //...  }  // Update is called once per frame  void Update()  {  //...  }  ////////////////////////////////////////////////////////////////////////  // Helper methods  private void DoMath(int a, int b)  {  return a + b;  }  }  } |

## Methods

Methods (also called functions) are used to further organise your scripts and to allow for an easier time with object orientated programming (the method Unity and C# use). Methods can be called from anywhere depending on their access level. They can also have inputs and can return an output.

The construction of a method declaration is shown below:

**public void MethodName(argType argName) {}**

Like variables and classes, methods can have access levels (public and private) and they will always go first. Then the object return type identifier comes next, you declare the type of object you want to return here, if you are not returning anything type ‘void’. Then you type the name of the method, open brackets and put in all your arguments (which is another way to say ‘input’). This is done in the same way as you declare a variable but without the access level, separate the inputs with a comma. Finally, you have to ensure that you have curly brackets {} to declare the scope of the method (in other words have it around your segment of code).

When defining your return type, you just write what you write for the type of your variable.  
See some object types below

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Full name** | **Description** |
| int | Integer | You use this to store any number that is not a decimal number |
| float | Float | You should always use this to store a number that has a decimal number |
| double | Double | You can use this to store a number that has a decimal point if you cannot use a float |
| string | String | Use this to store words or sentences |
| GameObject | Game Object | This is the generic object for objects in the Unity scene |
| boolean | Boolean | This represents a true or false value |

If you want to return a value write ‘return’ and type the name of the variable you want to return.