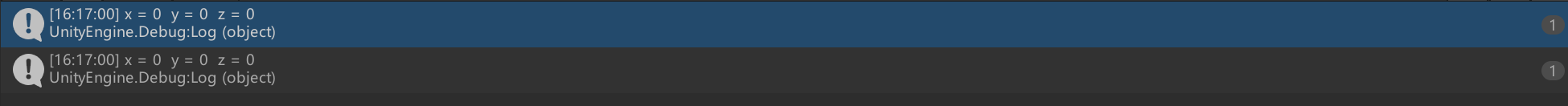
3.1

(1) The using statement defines a scope at the end of which an object is disposed. It also directive creates an alias for a namespace or imports types defined in other namespaces.

(2) The class ‘MonoBehaviour’ inherit from “Behaviour”. The new/overridden members (attributes and functions) defined are the Start() and Update() methods

3.2

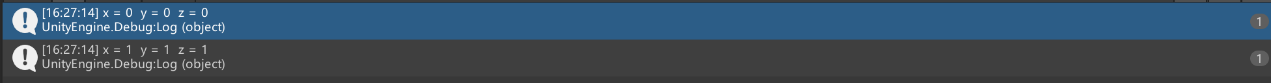
(1) It printout the following picture

The sphere does not change its position

(2) It is a value type. Therefore the “transform.position” will not change. Because v1 is just a copy of transform.position

3.3

It printout the following picture



The sphere change its position. Because this time it is a reference type.

When we amend “trans” the ‘transform.position’ will change as well

3.4

the accessibility of the attribute ‘name’ is private

3.5

void createGameObject2D () {  
  
 *// create 3 instances of the class ‘GameObject2D’  
 // and their names are Peter, David and John* GameObject2D Peter = new GameObject2D();  
 Peter.setName("Peter");  
 GameObject2D David = new GameObject2D();  
 David.setName("David");  
 GameObject2D John = new GameObject2D();  
 John.setName("John");  
 *// define a List (C# data structure, as introduced in the lecture)  
 // which holds 3 instances  
 // note that, if ‘List’ is not recognized by Visual Studio, you need to   
 // include another namespace, i.e. ‘using …’.* List<GameObject2D> list = new List<GameObject2D>();  
 list.Add(Peter);  
 list.Add(David);  
 list.Add(John);  
 *// using the loop structure, foreach, to print names of the 3 instances  
 // created earlier* foreach (GameObject2D obj in list)  
 {  
 Debug.Log(obj.getName());  
 }  
}

3.6

class Rectange : GameObject2D  
{  
 int width;  
 int height;  
 public Rectange(string name, int width, int height)  
 {  
 setName(name);  
 this.width = width;  
 this.height = height;  
 }  
}

3.7

It printout the following picture 