Hello CsvUtility

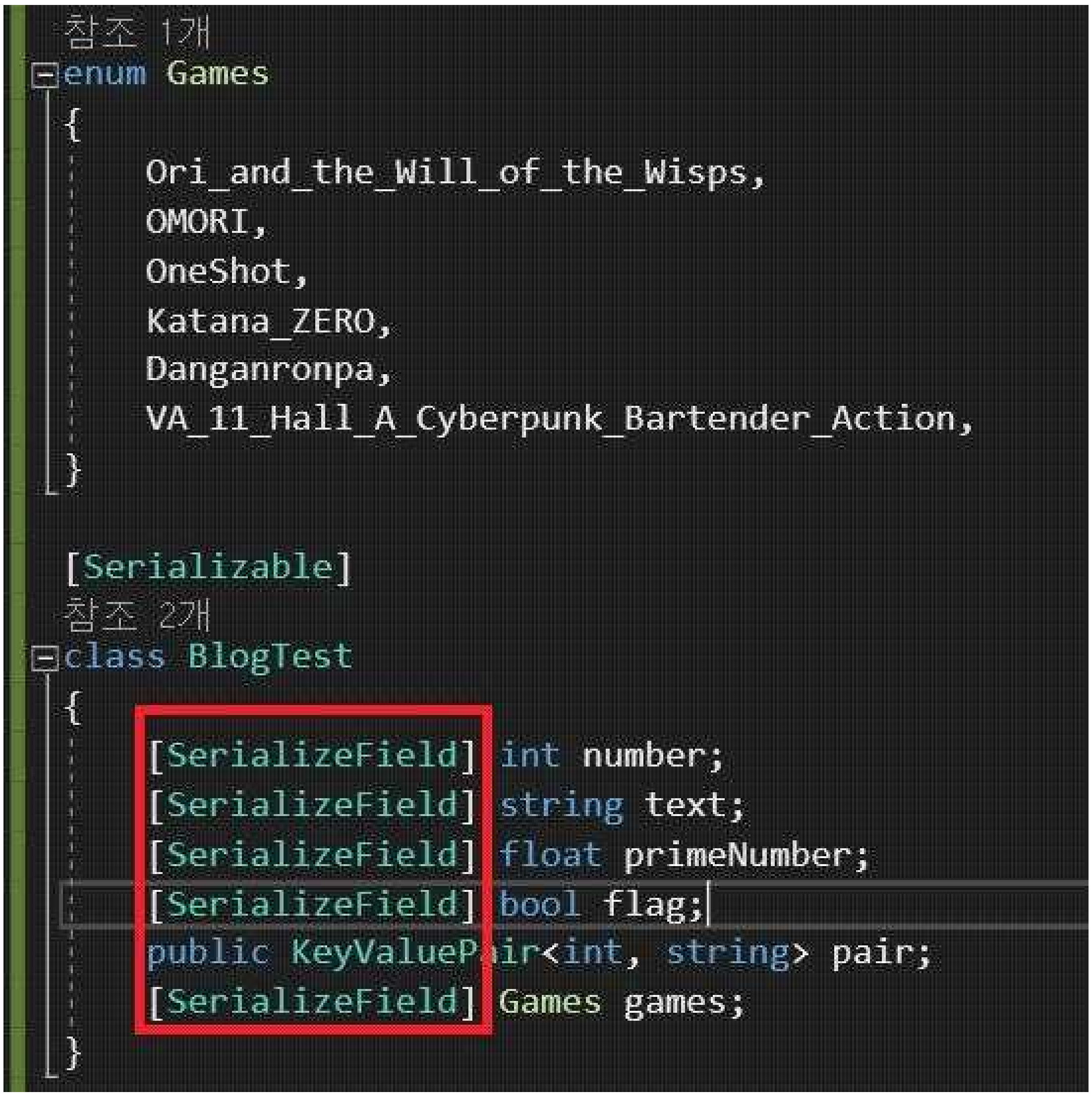
This file is the user manual for CsvUtility.

I hope you will use this asset to save time on cumbersome csv work. also hope to use that time in more creative ways to develop more fun games.

Lastly, I am using a translator because I am a Korean who does not speak English. Therefore, the quality of the description may be reduced. If you do not understand the explanation, please use the pictures and demo code.

1. Load Sample

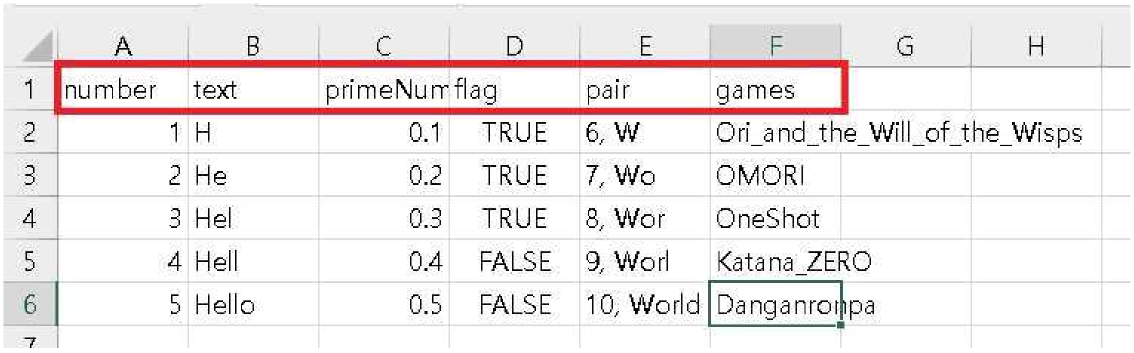
- sample class



```
참조 1개
enum Games
{
    Ori_and_the_Will_of_the_Wisps,
    OMORI,
    OneShot,
    Katana_ZERO,
    Danganronpa,
    VA_11_Hall_A_Cyberpunk_Bartender_Action,
}

[Serializable]
참조 2개
class BlogTest
{
    [SerializeField] int number;
    [SerializeField] string text;
    [SerializeField] float primeNumber;
    [SerializeField] bool flag;
    public KeyValuePair<int, string> pair;
    [SerializeField] Games games;
}
```

And in order to be applied to CsvUtility, it is unconditional!! It must be public or have the [SerializeField] property. Note that this rule is the same as for JsonUtility.

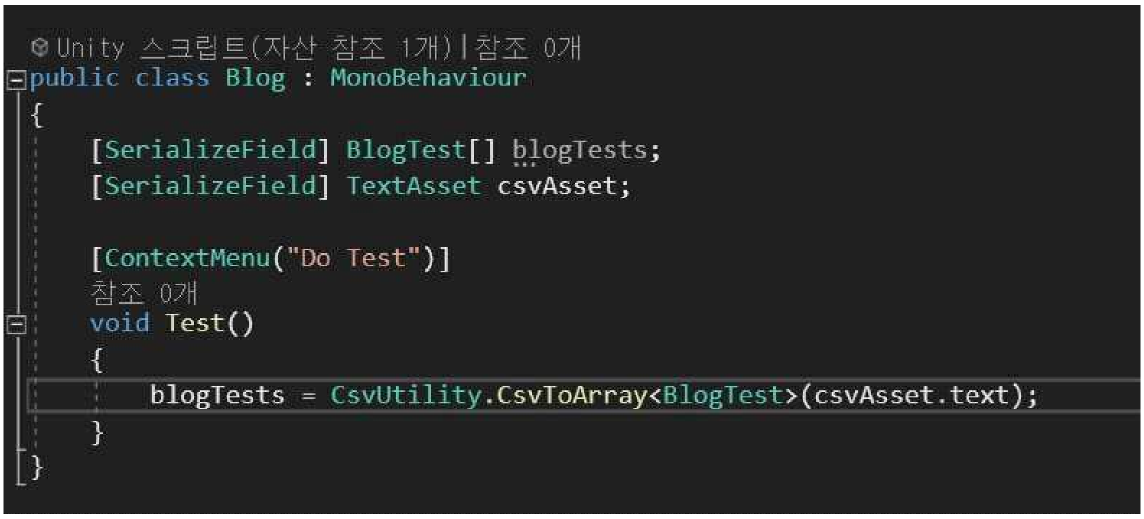
● sample csv file



	A	B	C	D	E	F	G	H
1	number	text	primeNum	flag	pair	games		
2	1	H	0.1	TRUE	6, W	Ori_and_the_Will_of_the_Wisps		
3	2	He	0.2	TRUE	7, Wo	OMORI		
4	3	Hel	0.3	TRUE	8, Wor	OneShot		
5	4	Hell	0.4	FALSE	9, Worl	Katana_ZERO		
6	5	Hello	0.5	FALSE	10, World	Danganronpa		

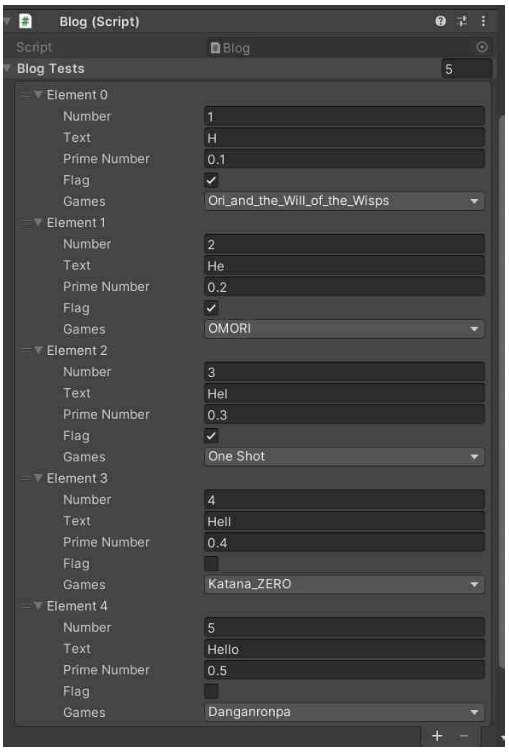
The first line is the variable name, followed by the data.

● code



```
Unity 스크립트(자산 참조 1개)|참조 0개
public class Blog : MonoBehaviour
{
    [SerializeField] BlogTest[] blogTests;
    [SerializeField] TextAsset csvAsset;

    [ContextMenu("Do Test")]
    참조 0개
    void Test()
    {
        blogTests = CsvUtility.CsvToArray<BlogTest>(csvAsset.text);
    }
}
```

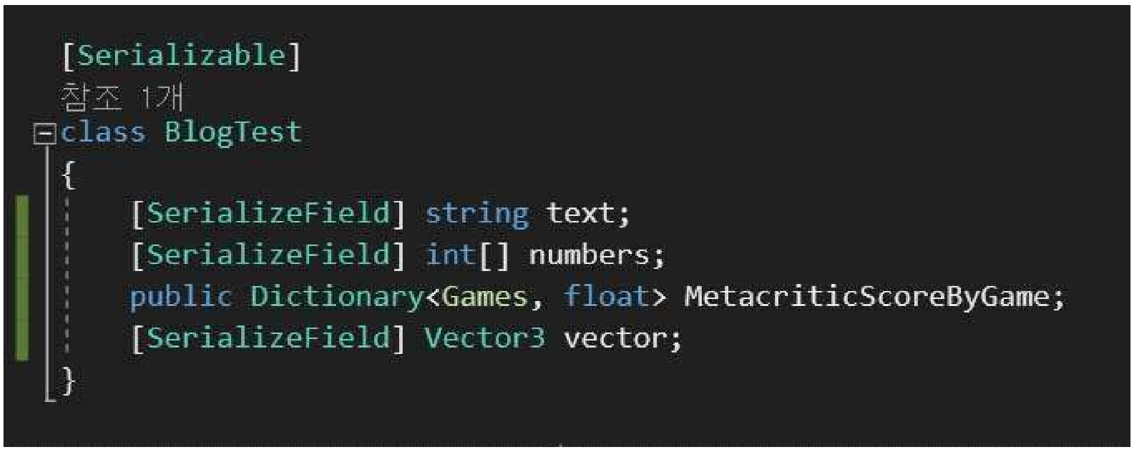
● result



```
Blog (Script)
Script          Blog
Blog Tests      5
  Element 0
    Number          1
    Text            H
    Prime Number    0.1
    Flag            ✓
    Games           Ori_and_the_Will_of_the_Wisps
  Element 1
    Number          2
    Text            He
    Prime Number    0.2
    Flag            ✓
    Games           OMORI
  Element 2
    Number          3
    Text            Hel
    Prime Number    0.3
    Flag            ✓
    Games           One Shot
  Element 3
    Number          4
    Text            Hell
    Prime Number    0.4
    Flag
    Games           Katana_ZERO
  Element 4
    Number          5
    Text            Hello
    Prime Number    0.5
    Flag
    Games           Danganronpa
```

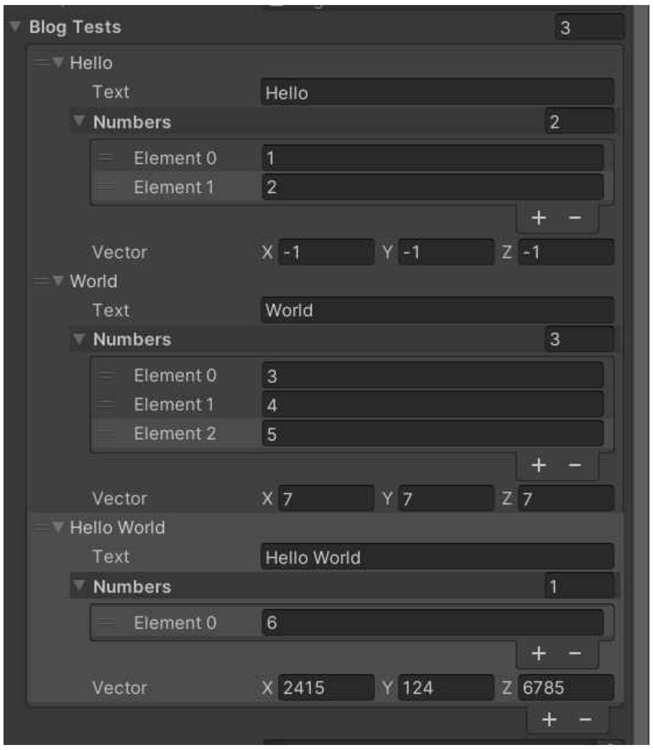
2. Save Sample

- sample class



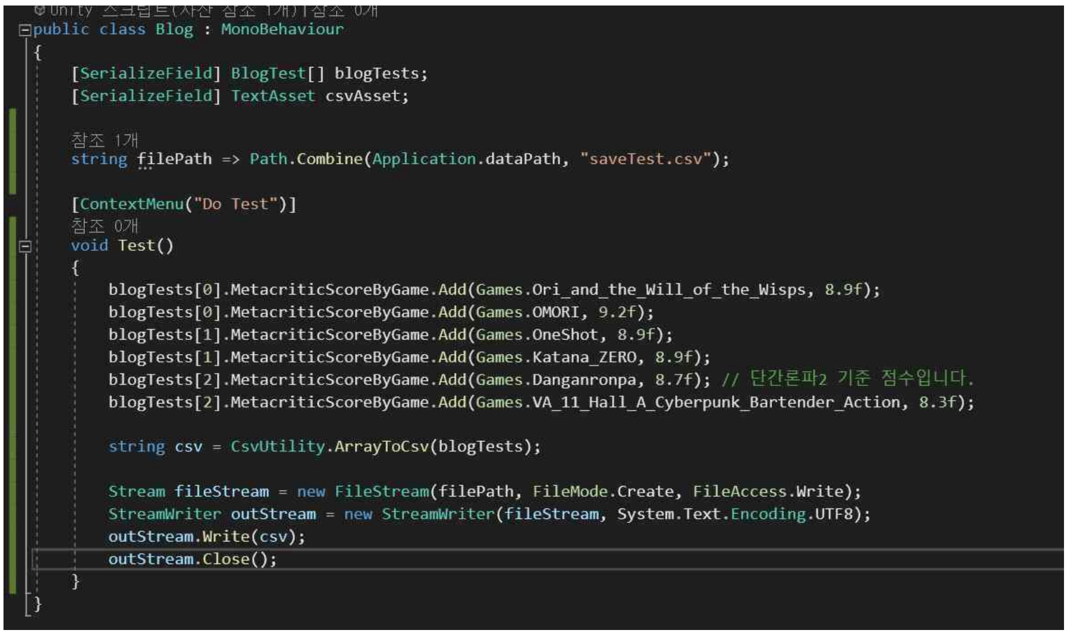
```
[Serializable]
참조 1개
class BlogTest
{
    [SerializeField] string text;
    [SerializeField] int[] numbers;
    public Dictionary<Games, float> MetacriticScoreByGame;
    [SerializeField] Vector3 vector;
}
```

- sample data



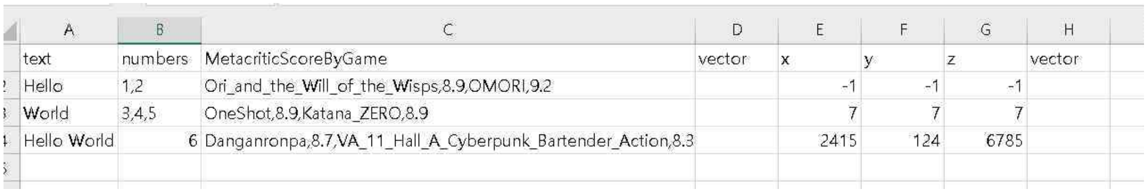
```
Blog Tests                          3
  Hello
    Text          Hello
    Numbers                         2
      Element 0   1
      Element 1   2
    Vector        X -1    Y -1    Z -1
  World
    Text          World
    Numbers                         3
      Element 0   3
      Element 1   4
      Element 2   5
    Vector        X 7     Y 7     Z 7
  Hello World
    Text          Hello World
    Numbers                         1
      Element 0   6
    Vector        X 2415  Y 124   Z 6785
```

- code



```
 Unity 스크립트(자산 참조 1개)|참조 0개
public class Blog : MonoBehaviour
{
    [SerializeField] BlogTest[] blogTests;
    [SerializeField] TextAsset csvAsset;

    참조 1개
    string filePath => Path.Combine(Application.dataPath, "saveTest.csv");

    [ContextMenu("Do Test")]
    참조 0개
    void Test()
    {
        blogTests[0].MetacriticScoreByGame.Add(Games.Ori_and_the_Will_of_the_Wisps, 8.9f);
        blogTests[0].MetacriticScoreByGame.Add(Games.OMORI, 9.2f);
        blogTests[1].MetacriticScoreByGame.Add(Games.OneShot, 8.9f);
        blogTests[1].MetacriticScoreByGame.Add(Games.Katana_ZERO, 8.9f);
        blogTests[2].MetacriticScoreByGame.Add(Games.Danganronpa, 8.7f); // 단간론파2 기준 점수입니다.
        blogTests[2].MetacriticScoreByGame.Add(Games.VA_11_Hall_A_Cyberpunk_Bartender_Action, 8.3f);

        string csv = CsvUtility.ArrayToCsv(blogTests);

        Stream fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);
        StreamWriter outStream = new StreamWriter(fileStream, System.Text.Encoding.UTF8);
        outStream.Write(csv);
        outStream.Close();
    }
}
```

- result



	A	B	C	D	E	F	G	H
	text	numbers	MetacriticScoreByGame	vector	x	y	z	vector
2	Hello	1,2	Ori_and_the_Will_of_the_Wisps,8.9,OMORI,9.2		-1	-1	-1	
3	World	3,4,5	OneShot,8.9,Katana_ZERO,8.9		7	7	7	
4	Hello World	6	Danganronpa,8.7,VA_11_Hall_A_Cyberpunk_Bartender_Action,8.3		2415	124	6785	
5								

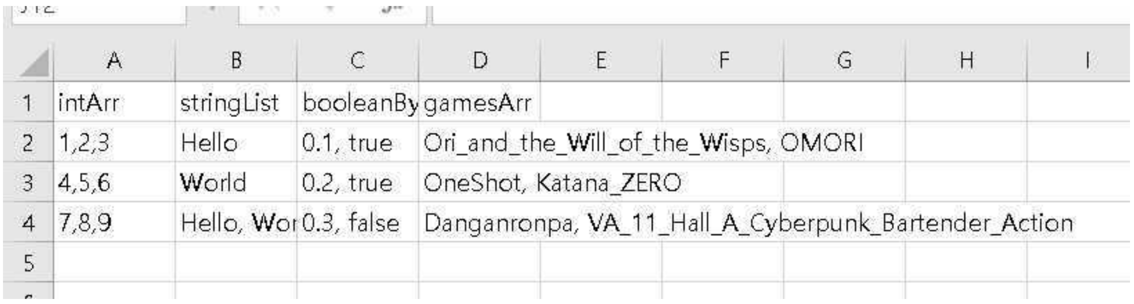
3. Rules

3-1. Array, List, Dictionary

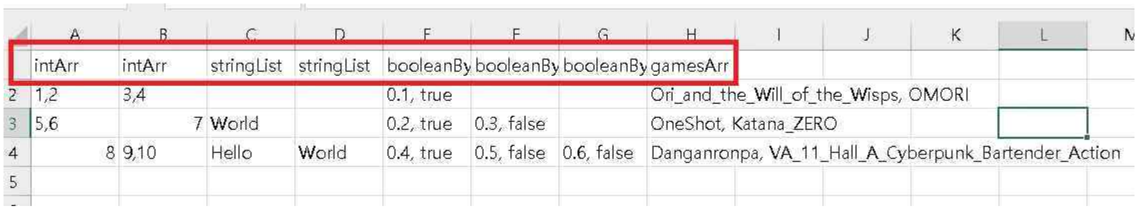
Arrays, lists, and dictionaries separate values with commas(,). At this time, it can be inconvenient to put all values in one cell.

To this end, we made it possible to put data into two or more cells consecutively.

The two pictures below both load the same values



	A	B	C	D	E	F	G	H	I
1	intArr	stringList	booleanBy	gamesArr					
2	1,2,3	Hello	0.1, true	Ori_and_the_Will_of_the_Wisps, OMORI					
3	4,5,6	World	0.2, true	OneShot, Katana_ZERO					
4	7,8,9	Hello, Wor	0.3, false	Danganronpa, VA_11_Hall_A_Cyberpunk_Bartender_Action					
5									

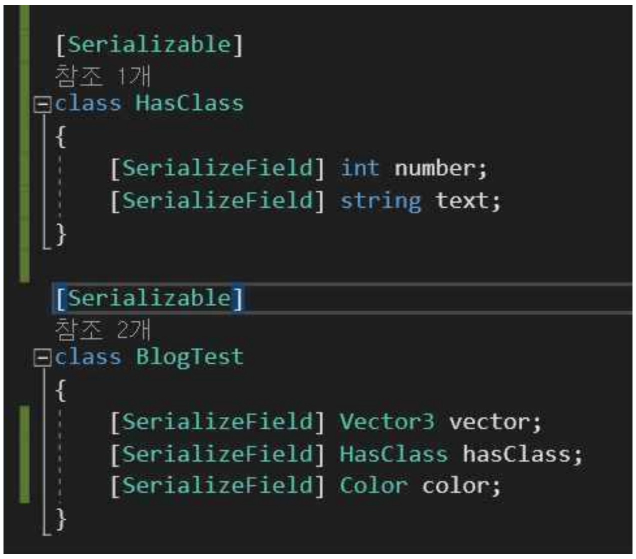


	A	B	C	D	E	F	G	H	I	J	K	L
	intArr	intArr	stringList	stringList	booleanBy	booleanBy	booleanBy	gamesArr				
2	1,2	3,4			0.1, true			Ori_and_the_Will_of_the_Wisps, OMORI				
3	5,6	7	World		0.2, true	0.3, false		OneShot, Katana_ZERO				
4	8	9,10	Hello	World	0.4, true	0.5, false	0.6, false	Danganronpa, VA_11_Hall_A_Cyberpunk_Bartender_Action				
5												

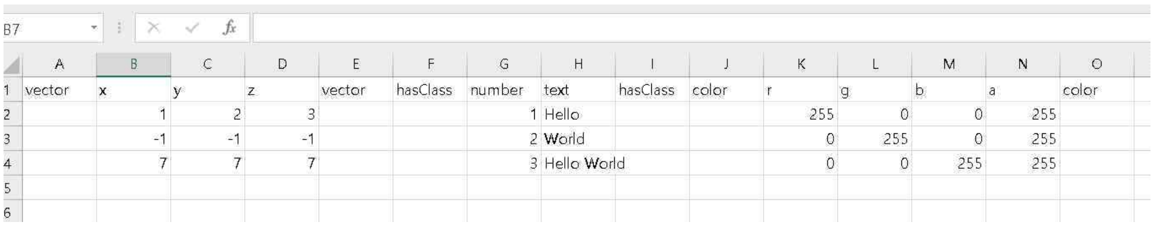
3-2. Nested Class

Nested classes put field names at the beginning and end. And in the meantime fill out the fields inside the class.

In case of a Vector3 type variable, first input the variable name, vector. And write the fields x, y, z of Vector3. And finally, write the variable name again.



```
[Serializable]
참조 1개
class HasClass
{
    [SerializeField] int number;
    [SerializeField] string text;
}

[Serializable]
참조 2개
class BlogTest
{
    [SerializeField] Vector3 vector;
    [SerializeField] HasClass hasClass;
    [SerializeField] Color color;
}
```

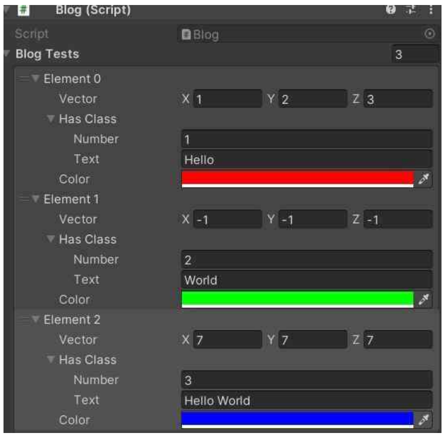


B7

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	vector	x	y	z	vector	hasClass	number	text	hasClass	color	r	g	b	a	color
2		1	2	3			1	Hello			255	0	0	255	
3		-1	-1	-1			2	World			0	255	0	255	
4		7	7	7			3	Hello World			0	0	255	255	
5															
6															

The pictures below show examples of some classes.

load result



```
Blog (Script)
Script            Blog
Blog Tests        3
  Element 0
    Vector        X 1    Y 2    Z 3
    Has Class
      Number      1
      Text        Hello
    Color         🟥
  Element 1
    Vector        X -1   Y -1   Z -1
    Has Class
      Number      2
      Text        World
    Color         🟩
  Element 2
    Vector        X 7    Y 7    Z 7
    Has Class
      Number      3
      Text        Hello World
    Color         🟦
```

4. Precautions

4-1. Unsupported type

Unsupported types are all types except supported types.
CsvUtility supports byte, int, long, string, float, double, bool, enum and Array, List, Dictionary, class, and struct using them.
Also, data structures such as two-dimensional arrays are not supported.

4-2. When using a dictionary

If you use a dictionary, it is unconditional!! You have to assign it with new.

public Dictionary<T1, T2> dictionary = new Dictionary<T1, T2>();
Like this. In other words, it is the same as saying that you cannot use a dictionary in a struct.