Coding Standards

File Formatting

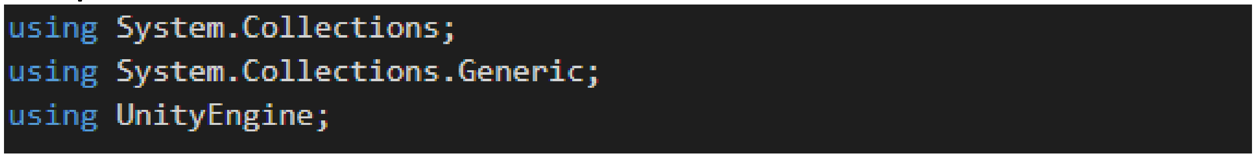
Indentation

Each new level of code signaled by the presence of a {on the previous line should be indented using 4 spaces and decremented by 4 spaces at the} character. Tabs should be avoided.

Libraries

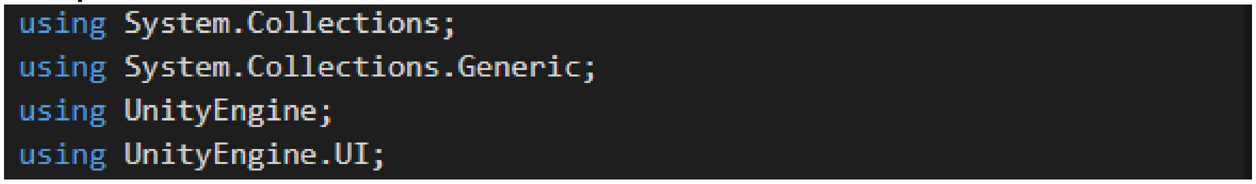
Libraries should be ordered alphabetically at the head of a file. If multiple sub libraries are being used the top-level library should be at the head of this list.

Example 1:



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

Example 2:



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
```

Whitespaces

Developers should follow the spacing guidelines below:

- All functions should be separated by at least 2 empty lines in code
- All logic/arithmetic operations done in code should have variables and operators separated by one space
- Place additional blank lines to show separation between unique sections of code

Naming Conventions

Folders

Folders should follow the PascalCasing format, where the first letter of every word is capitalized. No spaces, underscores, numbers, or special characters are used in the name.

Files

File names should always be similar to the class they have. If more than one class is present in a file, the more relevant class should be named similar to the file.
For any file that is not code should follow PascalCasing format, where the first letter of every word is capitalized. No spaces, underscores, numbers, or special characters are used in the name.

Classes

All classes or structures should follow the PascalCasing format, where the first letter of every word is capitalized. No spaces, underscores, numbers, or special characters are used in the name. The name of the class should match the name of the file it resides in.

Example:

```
public class CustomClass
{

}
```

Functions

All functions should follow the PascalCasing format. No spaces, underscores, numbers, or special characters are used in the name. The starting '{' should appear on the line after the function name.

Example:

```
void CustomFunction()
{

}
```

Variables

All variables should follow the camelCasing format, where the first word of the variable is lower case, and all other words capitalized. No spaces, underscores, numbers, or special characters are used in the name.

Example:

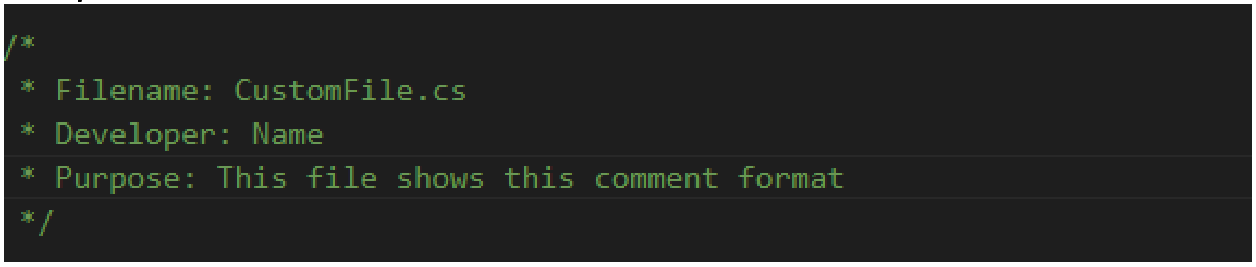
```
public GameObject itemOne;
int i = 0;
Player player = new Player();
float newXPosition = 0f;
```

Documentation

Files

At the beginning of a file before any code or libraries, a comment with the following format should be written as shown below.

Example:

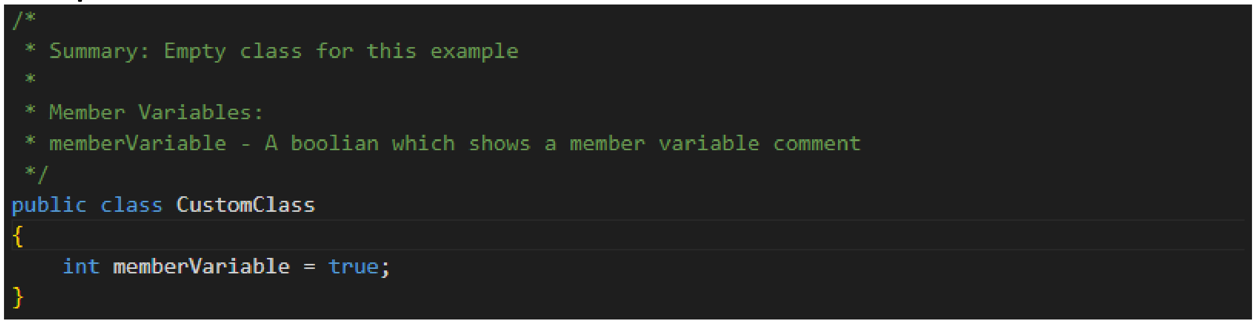


```
/*
 * Filename: CustomFile.cs
 * Developer: Name
 * Purpose: This file shows this comment format
 */
```

Classes

Class comments should be put right above the class declaration. As shown below, the first line should contain a summary of the class. Member variables should also be included with a space between the summary and itself as shown below.

Example:

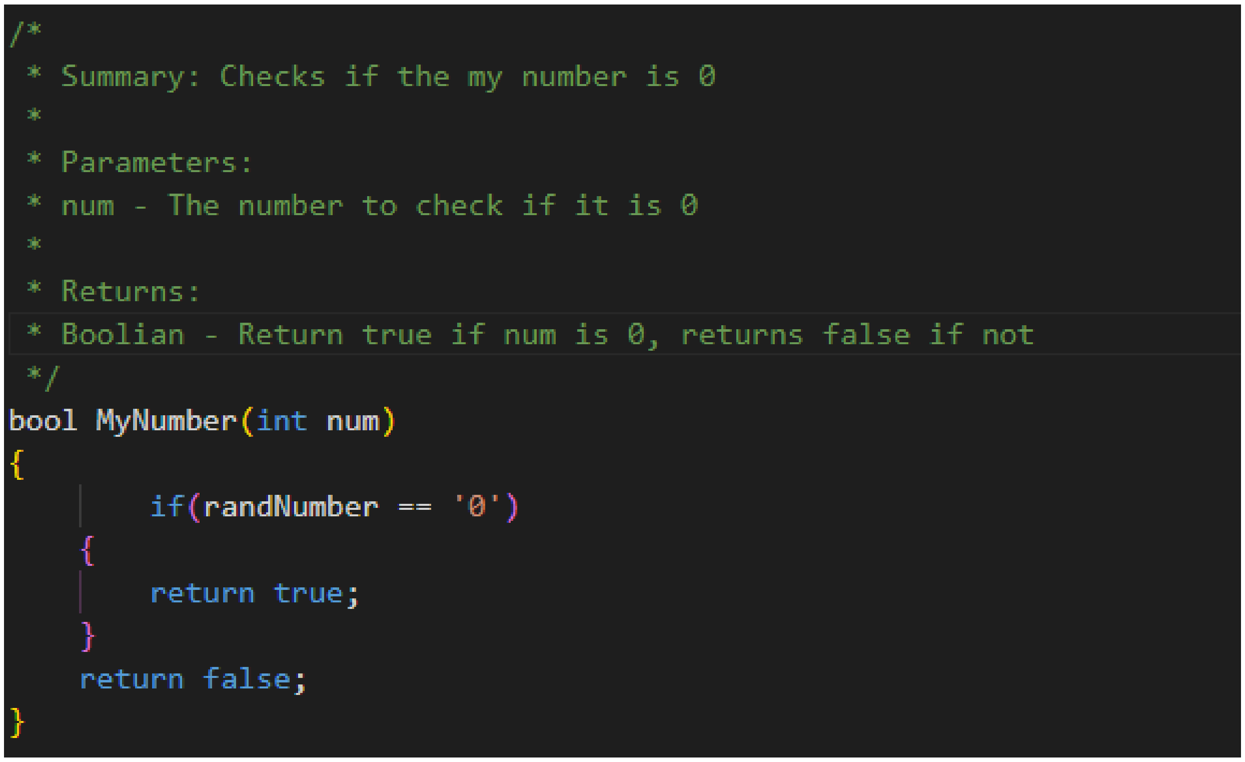


```
/*
 * Summary: Empty class for this example
 *
 * Member Variables:
 * memberVariable - A boolian which shows a member variable comment
 */
public class CustomClass
{
    int memberVariable = true;
}
```

Functions

Function comments should be put right above the function declaration with the following format. Comments should include a summary, parameters, and returns. If the function returns void, then the returns section is not needed.

Example:



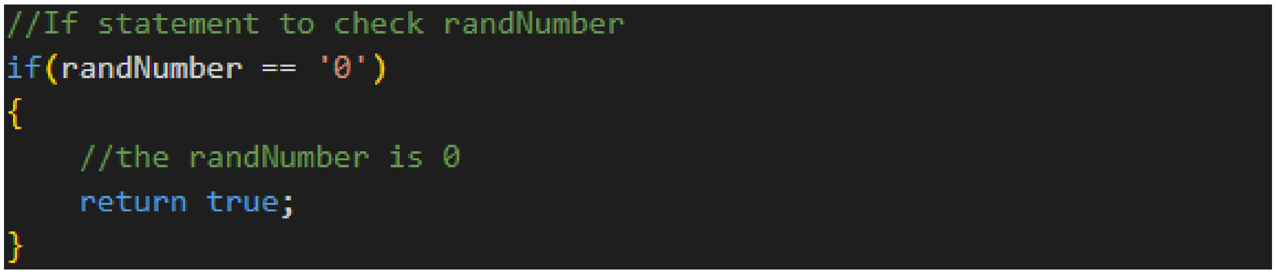
```
/*
 * Summary: Checks if the my number is 0
 *
 * Parameters:
 * num - The number to check if it is 0
 *
 * Returns:
 * Boolian - Return true if num is 0, returns false if not
 */
bool MyNumber(int num)
{
        if(randNumber == '0')
    {
        return true;
    }
    return false;
}
```

Prefabs

Comment Prefabs
Name: Semi generic
Summary: include keywords
Description: more detail, troubleshooting guide, integration instructions, ect. videos (to make BC happy) (will be graded)

Other Comments

Any additional comments should be placed above the line of code as shown below in the example. Comments should answer the question likely to be asked and are there to increase the readability of the code.



```
//If statement to check randNumber
if(randNumber == '0')
{
    //the randNumber is 0
    return true;
}
```