Classe :

- Une classe est la définition que nous décidons d'associer à un objet. Dans notre cas, chaque components pouvant être attachée à un GameObject est une classe. Par exemple, le rigidbody est une classe. Nous n'en avons pas accès, puisque celle-ci est directement caché dans le « source code » de l'engine de unity. (DLL : Direct link library)

-Le Monobehaviour est une classe. Nous faisons de l'Héritage de cette classe dans chacun de nos scripts qui seront présent dans la scène, on dit donc que l'on Hérite de la classe MonoBehaviour;

Déclaration de classe
public Class maClasse

Une classe n'hérite pas nécessairement de MonoBehaviour dans Unity. On peut facilement déclarer une classe pour entreposer du data ou simplement effectuer des calculs séparé de notre principal intéressé.

Exemple :
public Class Mammal

Déclaration de classe avec Polymorphie / Polymorphism
public Class maClasse : MonoBehaviour

public Class Biped : Mammal
public Class Human : Biped

De cette façon nous avons accès à tout les méthodes publiques qui sont dans la classe de base MonoBehaviour et exceptionnellement pour cette classe en particulier, Unity nous dit que nous pouvons recevoir les messages suivant par l'engine directement.

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

Monobehaviour Messages
-Awake
-Start
-Update (Voir Time.deltaTime)
-FixedUpdate (Voir Time.FixedDeltaTime)
-LateUpdate

GameObjects Actions
-AddComponent
-Destroy

Classes Connus et que nous auront Besoins
- Vector3 : Classe contenu dans le namespace UnityEngine. (x,y,z)
https://docs.unity3d.com/ScriptReference/Vector3.html

- String : Classe contenu dans le namespace System. (Liste de character pour arriver à des mots)
https://docs.unity3d.com/ScriptReference/String.html

Variables :

- Une variable est un espace mémoire que nous demandons à l'ordinateur/console/système d'exploitation, afin d'y mettre de l'information que nous auront besoins dans la code.

Binary :

1 byte = 8 bit

0000 = 0
0001 = 1
0010 = 2
0100 = 4
1000 = 8
1010 = 10

On Compte en base 2, donc pour savoir la valeur du 3 ième index on fait 2 exposant 3 = 8.
On compte les index à partir de 0.

Déclaration de la variable :

private bool m_CanJump;

Types de variables de bases :

- bool (1bytes, 8 bit)
- char (1bytes, 8bit) 0; 255
    * Voir Table Ascii pour les 256 characters disponibles
- int (4 bytes, 32 bit) -2,147,483,648; 2,147,483,647
- uint (4 bytes, 32 bit), on sauve 1 bit pour le sign, toujours positif
- float (4 bytes, 32 bit), 1 bit sign, 23 pour la valeur et 8 pour pour calculer le chiffre à virgul
- double (8 bytes, 64bit), 1 bit sign, 52 pour la valeur et 11 pour calculer le chiffre à virgul

Float Points : https://www.mikeash.com/pyblog/friday-qa-2011-01-04-practical-floating-point.html

Accessibilité des Variables :

- Variables membres (De classe)

* Cette variables est déclaré au début de la classe est a pour but d'avoir de l'information nécessaire à plusieurs éléments de la classe. Son espace mémoire est libéré lorsque sa classe est détruite.

- Variables de fonctions

* Cette variable est créé à l'intérieur d'une fonction, elle est accessible par n'importe qui dans la fonction et son espace mémoire est libéré à la fin de la fonction.

-Variables de statement

* Cette variable est créé à l'intérieur de statement (exemple : if), elle est accessible à l'intérieur du statement seulement et son espace mémoire est libéré à la fin du statement.

Fonctions :

-Une fonction est un regroupement de ligne de code à l'intérieur de la classe qui permet de la réutilisation, de la propreté de code et une bien meilleure pratique.

```
private void Jump(int i_JumpForce)
{
	//Faire un jump qui utilise le i_JumpForce
}
```

-void est la valeur de retour de la fonction, si on s'attend à une valeur de retour on peut indiquer de quel type on voudrait que la fonction nous retourne. Void indique que la fonction ne retourne aucune valeur. On utilise la commande return afin de retourner la valeur voulu par la fonction.

```
private int Addition(int i_Valeur1, int i_Valeur2)
{
	return i_Valeur1 + i_Valeur2;
}
```

- De plus, entre les parenthèse nous pouvons indiquer à la fonction si elle reçoit des valeurs afin d'avoir un comportement différents selon la valeur reçu.

Accesseurs :

-Private, La classe, la fonction, la variable sera exclusive à la classe à laquelle elle a été déclaré. Donc personne d'autre que la classe elle même ne pourra être utilisée.

-Public, La classe, la fonction, la variable est disponible à quiquonque a une référence à cette classe.

Références :

-Dans unity il y a deux façon facile d'obtenir la référence à un autre gameObject ou à un autre component que nous voudrions utiliser.

*Par la référence public et le drag and drop.
*Par l'utilisation de la fonction GetComponent<>()

If Statement :

- Le if statement est la facon que nous utilisons avec la programmation de faire des choix selon des conditions définis.

*On l'écrit de cette façon.

```
if(condition)
{
	//Faire quelque chose
}
```

*On peut utiliser le else pour déterminer une action si la première condition n'est pas respecter.

```
if(condition)
{
	//Faire quelque chose
}
else
{
	//Faire quelque chose d'autre
}
```

*On peut utiliser le else if pour déterminer une condition différente dans le même statement.

```
if(condition1)
{
	//Faire quelque chose
}
else if(condition2)
{
	//Faire quelque chose d'autre
}
else
{
	//Faire quelque chose d'autre
}
```

-On peut utiliser les conditions suivantes.

```
int a = 5;
int b = 5;
bool d = true;
```

* d == true // si la valeur de d est true
* a == b (true) // si la valeur de a est égale à b
* a > b (false) // si la valeur de a est plus grande que b
* a < b (false) // si la valeur de a est plus petite que b
* a >= b (false) // si la valeur de a est plus grande ou égale à b
* a <= b (false) // si la valeur de a est plus petite ou égale à b
* a != b (false) // si la valeur de a n'est pas égale à b

- On utilise la commande && pour rajouter une condition AND à notre if.
if(condition1 && condition2), à ce moment les 2 conditions doivent être vraie pour accédé au statement.
-On utilise la commande || pour rajouter une condition OR à notre if.
if(condition1 || condition2), à ce moment seulement l'une des 2 conditions doit être vraie pour accédé au statement.

Cool Bonus :

-On utilise la string pour utiliser une chaine de character. Donc pour arriver à des mots / phrase.

*string maString = "une phrase quelquonque";

*string maString1 = "Hello";
*string maString2 = "World";
*string maString3 = mString1 + maString2; // maString3 = "HelloWorld"

La fonction Debug.Log utilise une string.
On peut aussi l'utiliser avec une string déclaré directement : Debug.Log("HelloWorld");