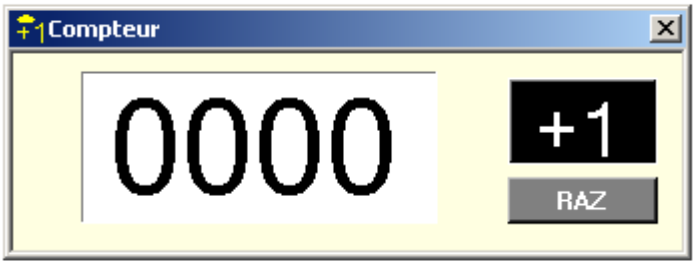
Introduction à la **programmation orientée objet**.

I. Objet et classe :

Pour compter le nombre de participant à une manifestation, on utilise un petit objet appelé « un compteur ».

Son équivalent sous Windows pourrait avoir l'apparence suivante :



Comment utilise-t-on un compteur ?

> Au départ, on remet le compteur à Zéro.
> A chaque passage, on appuie sur le bouton +1.
> A la fin, on lit le nombre.

Quel sont les éléments apparents ?

> Un affichage pour le nombre résultat.
> Un bouton de RAZ.
> Un bouton +1 de comptage.

L'utilisateur n'a pas besoin de savoir ce qui se passe à l'intérieur.

Que lui suffit-il de connaître pour pouvoir utiliser cet **objet** ?

> Le Nom de cette catégorie d'objet : c'est un Compteur.
> Le comportement de cette catégorie d'objet.

Une catégorie rassemblant tout les objets ayant ce même comportement s'appelle une **classe**. On pourrait lui donner le nom Compteur.

II. Objet et classe :

La personne qui conçoit cette catégorie d'objet, cette classe, doit déterminer ce qui se passe à l'intérieur. Pour cela il se pose quelques questions :

Qu'est ce qui détermine l'état de mon objet ?

C'est un nombre entier qui représentera le résultat du comptage.
Je pourrai lui donner le nom de compte.

Les *variables* qui mémorisent l'état d'un objet s'appellent des attributs.

Quelles sont les *fonctions* qui permettront de modifier l'état de mon objet ?

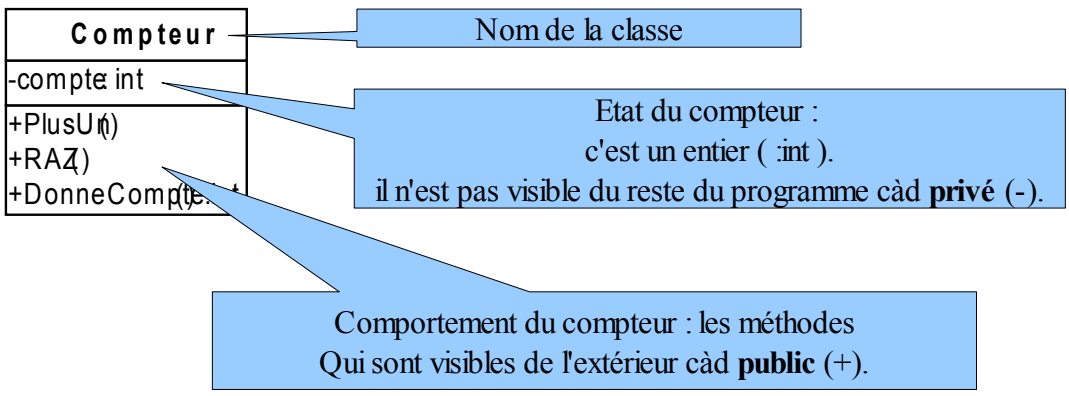
La fonction qui ajoute un au compte que je pourrai appeler PlusUn.
La fonction qui met à zéro le compte que j'appellerai RAZ.

Quelles sont les *fonctions* qui permettront de connaître l'état du compteur ?

La fonction qui renvoie le compte que j'appellerai DonneCompte.

Ces fonctions qui travaillent sur les attributs de l'objet s'appellent des **méthodes**.

Désormais je peux décrire la **classe** Compteur sous forme d'un diagramme :



III. Essayons d'implémenter la classe compteur.

Quel *langage* choisir ?

> Il existe de nombreux langages conçus pour la programmation orienté objet : C++, C#, Java, etc...

Ici, j'avais choisi C# :

Je traduis mon diagramme :

```
public class Compteur
    {
        int compte;
        public Compteur()
        {
            RAZ();
        }
        public void RAZ()
        {
            compte = 0;
        }
        public void PlusUn()
        {
            compte++;
        }
        public int DonneCompte()
        {
            return compte;
        }

    }
```

Simple nom ?

L'application Windows construite autour de cette classe Compteur se trouve ici.

IV. Définitions

IV.1 Objet

Désigne tout objet manipulé par un programme.

Un **objet** est caractérisé par son état (ensemble des valeurs qu'il contient) et par un comportement.

IV.2 Classe

Une classe décrit le comportement d'un ou de plusieurs objets.

Une **classe** possède 3 sortes de membres :

a) **attributs** : représentent l'état de l'objet. Ce sont des objets.
b) **Méthode** ou opération : c'est une fonction décrivant un comportement de l'objet. La liste des instructions d'une méthode s'appelle le **corps** de la méthode.
c) **Propriété** : nom décrivant des caractéristiques de l'objet définis via deux méthodes appelés accesseurs (affecter et récupéré, set et get).

IV.3 Visibilité

Suivant sa visibilité (ou portée), attribut et méthode peuvent être :

- privé : accessible seulement depuis la classe où il est défini.
- Protégé : accessible aussi par toutes les classes dérivées.
- Public : accessible par toutes les classes.

IV.4 Constructeur

Pour créer un objet décrite par la classe TA , il faut :

- Réserver la place en mémoire correspondant à la classe TA.
- Appeler la méthode **constructeur** qui initialisera les attributs.

Cela est réalisé par l'opérateur **new** (nouveau) du langage**.**

NB : Le nom de la méthode constructeur est le même que le nom de la classe.

IV.5 La vie d'un objet :

	OS	Avec C#
Naissance	Réserver l'espace mémoire Mettre l'objet dans l'état initial.	new Constructeur TA()
Vie	Appel des méthodes, opération sur les attributs publics, affectation, Test d'égalité, manipulation avec les opérateurs	
Mort	Libérer les ressources Détruire les champs utilisés libérer l'espace mémoire	**destructeur** ~TA()