**APOO : semaine 1**

Un objet se caractérise grâce à un type, un état et un comportement.

Ce type de programmation (orienté objet) a pour but de représenté des objets concrets ou conceptuels, permet de créer de nouveaux types de données. Les objets interagissent et collaborent ensuite entre eux.

*// on utilise ce type de programmation orienté objet c’est-à-dire que l’on créée un objet que l’on va ensuite réutiliser cela permet de diminuer le risque d’erreurs, plus de lisibilité ainsi que plus de maintenabilité.*

**Classe**

Une classe va décrire un type d’objet ainsi que ses caractéristiques et son comportement.

Elle possède :

* Un nom (commence toujours par UNE MAJ)
* Un état (attribut et valeur)
* Un comportement (action)

Une instance de la classe est un objet ayant un comportement et un état

|  |
| --- |
| Cercle |
| rayon : double |
| calculerAire (): double |

En UML on écrit en inverse D’abords le nom ensuite le type   
*c’est notre classe avec notre objet sur UML*

Quand on utilise une méthode comme ici dessus on met des () lorsqu’il y a un paramètre et on indique le type de retour (double dans le cas présent)

|  |
| --- |
| cercle1 : Cercle |
| rayon : 10.3 |

Voici l’instance de la classe sur UML = objets

On peut revoir que ‘Cercle’ porte une majuscule car c’est le type d’objets qu’est cercle1 et 2

|  |
| --- |
| cercle2 : Cercle |
| rayon : 10.3 |

**Les variables**

Ils existent 2 types de variables :

Les variables primitifs et les variables de références

**Variables primitifs = type simple**

*int ou long : nombre entier  
double ou float : nombre a virgule  
char : pour un caractère unique (on mets toujours le caractères entre ‘ ‘)  
boolean : pour représenter un booléan ( vrai ou faux)  
string : pour une chaine de caractères ( toujours « blablabla »)*

Il est possible de réutiliser une variable. Juste mettre le nom de la variable

nombre

On peut aussi assigner une valeur a une valeur pour le type primitif par ex  
 int nombre = 45 ;

EX : int nombre ;

**Variables de références = une classe**

EX : Cercle uncercle = new Cercle (32.3) ; ici on invoque le constructeur

Dans l’exemple au-dessus on a invoqué le constructeur

Le constructeur porte le même nom que la classe (l’objet) et a une majuscule. Il est suivi d’un paramètre entre parenthèses. Dans la parenthèse est mentionné le type et le nom.

Ex ; Cercle (double nRayon){  
 rayon = nRayon ;

}

Il permettra de construire de nouveau objets. Le constructeur est invoqué en unissant **new**