Méthode d'Analyse : Programmation orientée Objet

NB : certaines notions présentée ici seront revues dans d'autres cours.

I. Introduction :

De nombreux problèmes ne sont pas aisé à représenter sous la forme d'un arbre.
En particulier dès que l'on veut faire de la programmation Windows, on atteint très vite les limites de la programmation structurée.

Aussi les informaticiens ont-ils recherchés d'autres approches. Parmi celles-ci la programmation orientée Objet.

II. Exemple :

On veut concevoir un Chronomètre.

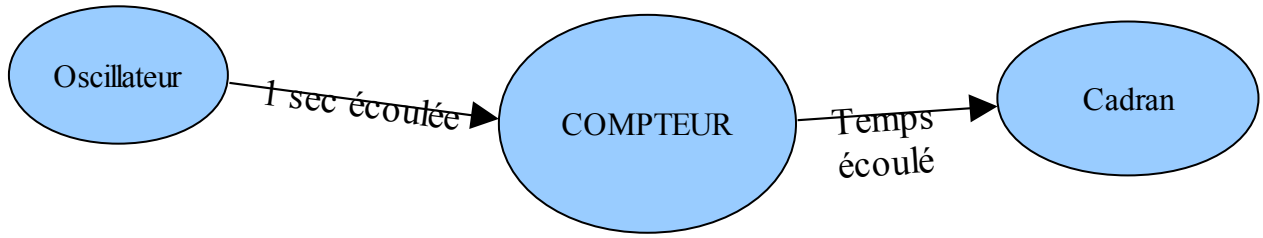
On peut décrire le problème sous la forme d'un graphe (hic...).

Essayons de décrire l'objet Chronomètre et ces composants :

Un Chronomètre comporte 3 composants :

- un oscillateur (mécanique ou à quartz) : produit 1 message par seconde.
- un compteur : compte et mémorise le nombre de secondes écoulées depuis sa création.
- un cadran : affiche un nombre sous forme sexagésimal (heures:minutes:secondes).

Ces composants sont assemblés de tel manière que : l'Oscillateur signale au compteur chaque fois qu'une seconde passe. (On dit qu'il envoie un **message** correspondant à l'**évènement** « fin de la période de 1 seconde ».)
De même Le Cadran qui veut afficher l'heure demande au compteur ce qu'il doit afficher (C'est un **message** de demande de la valeur du compteur).



Il ne reste plus qu'à définir plus précisément les différents composants.

L'**objet** Compteur est caractérisé par :

- son état : un nombre i qui représente le nombre de seconde écoulée.
- Son comportement :
    - Il renvoie par la méthode « Valeur » le nombre représentant son **état** .
    - Il s'incrémente à chaque appel de la méthode « Inc ».
    - Il s'initialise lors de sa création (au démarrage du programme).

Si on veut fabriquer plusieurs compteur, on va dessiner un plan de compteur décrivant tous les compteurs. On saura que tous les objets de la **classe** « Plan de compteur » auront le même comportement.

III. Définitions :

a) Un **objet** est caractérisé par :

- son état : ce sont les **propriétés** de l'objet à un instant donné.
- son comportement : c'est la manière qu'a l'objet de réagir aux messages qu'il reçoit.

Il est implémenté par les **méthodes** ou opérations de l'objet.

b) Une **classe** est la description de l'ensemble des objets d'un programme ayant les mêmes propriétés d'état et les mêmes comportements.

c) Une **instance** d'une classe est un **objet** que l'on a **construit** en suivant la description donnée dans la **classe**.

Ici, quand on construit un Compteur, on instancie la classe PlanDeCompteur**.**

IV. Description d'une classe : la classe Compteur :

(voir II)

Nom de la classe	Compteur
Propriétés	Privé Entier i // Variable interne permet de mémoriser l'état du compteur Public

Méthodes	Public
	Constructeur() /// Initialisation au début
	Valeur() // méthode appelée pour l'affichage Donne la valeur de i
	Inc() // méthode appelée pour faire avancer le compteur : Incrémente i

Comme dans l'analyse structurée, il y a une notion de portée de nom de fonctions ou de variable :

Seul ce qui est **public** est visible et accessible en dehors de la classe. Ce qui est **privé** n'est accessible que depuis l'intérieur de la classe.

Il existe des langages de représentations standards comme UML (Unified Modeling Language).

Représentation UML :

Compteur
i : entier
+ Compteur() +Valeur():entier +Inc()

« + » veut dire que la propriété ou la méthode est **public**.

Algorithme des différentes méthodes :

Act 2 : ajouter une méthode de remise à zéro, Stop /start

Act 3 : Traduire en C++ ou C# et implémenter (ensemble avec visual studio)
Act 4 : affichage sexagésimal.

V. Autres exemples et exos :

1. Les nombres complexes en mathématique sont définis par l'ensemble des nombres x tel que $x = a + b * i$ où a et b sont des réels et i la solution de l'équation $x^2 + 1 = 0$.

   Pour pouvoir manipuler des nombres complexes dans un programme, on va définir une Classe d'Objet que l'on nommera Complexe.

   Diagramme de la classe :

Complexe
-a : double -b : double
+Complexe(double, double) +Affiche() +Complexe(Complexe) +operator+(const Complexe&, const Complexe&) : Complexe

Implémentation en java :

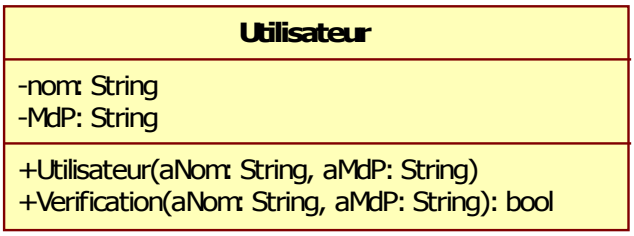
```
class Complexe {
double a;
double b;

/**
 * Method Complexe
 *
 *
 * @param a
 * @param b
 *
 */
 Complexe(double aa, double ab) {
            // TODO: Add your code here
            a = aa;
            b = ab;
}

void Affiche() {
            System.out.println( "(" + a + " + " + b + " i )");
}

Complexe Carre() {
            return new Complexe( a*a - b*b,  2*a*b);
}

Complexe Add (Complexe c1, c2) {
            return new Complexe( c1.a + c2.a, c1.b + c2.b);
}

}
```

implémentation en C++ :

```
class Complexe {
double a;
double b;

public:
Complexe(double aa = 0.0, double ab = 0.0) {
  // TODO: Add your code here
  a = aa;
  b = ab;
 };

void Affiche() {
  printf( "(%f + %f i)", a, b);
 };

Complexe Carre() {
  return Complexe( a*a - b*b,  2*a*b);
 };

friend Complexe operator + ( Complexe &c1, Complexe &c2) {
     return Complexe( c1.a + c2.a, c1.b + c2.b);
    };

};
```

2. Authentification :

a) utilisateur : décrire une classe Utilisateur d'un système informatique.



b) Administrateur :

3. Eléments géométriques :

a) Point : un point géométrique 2 dimensions est caractérisé par deux coordonnées x et y et un certains nombres d'opérations (+, *, …).

décrire la classe Point.

b) Décrire une classe Segment :

4. Une maison appartient à une propriétaire qui est une personne.
Pour les impôts une maison est caractérisée par une surface habitable, une surface brut,...
Décrire les classes Personne et Maison.
Comment décrire un hameau ?

5. Répertoire d'adresse : reprendre le TP répertoire en utilisant la démarche Objet.

a) Définir les classes Entrée et Répertoire.

b) Algorithme des méthodes Raz (dans la classe Répertoire), Insertion(dans la classe Entrée), Affichage(dans la classe Répertoire et Entrée).

c) Implémentation en mode Console avec programme de test : 3 insertions + Affichage de la liste.

d) Implémentation en mode Windows avec formulaire de test : 3 insertions + Affichage dans une Liste.

VI. Héritage :

On veut créer une nouvelle classe de compteurPause :

Elle renvoie par la méthode « Valeur » le nombre de seconde écouler depuis le démarrage.
Elle est initialisé au démarrage du programme.
Elle arrête de compter si on lui demande une Pause.

On voit que cette nouvelle classe à pratiquement le même comportement que la classe Compteur.
En programmation orientée objet, il existe un mécanisme permettant de conserver l'acquis que nous avons accumulé en définissant la classe Compteur. Ce mécanisme s'appelle **Héritage.**

On peut donc définir la classe CompteurPause comme suit :

CompteurPause : Compteur		CompteurPause à pour classe de base Compteur
bActif : booléen		Si faux le compte s'arrete si vrai le compte s'effectue
+ Pause() : rien		effectue la négation de bActif

On dit que la classe CompteurPause est **dérivée** de la classe de **base** Compteur.
On dit que la classe CompteurPause **hérite** du comportement de la classe Compteur.

Définition :

Une classe **est dérivée** ou **hérite** d'une classe de **base** si elle a :
• au moins le comportement de cette classe de **base** • mais en plus un comportement ajouté.

NB : ce mécanisme doit conduire à une structure d'arbre. (dans les langages « propres ».)

NB : dans beaucoup de langage toutes les classes sont dérivée d'un même classe ayant le comportement minimal d'un objet. Pe : en C# la classe « Object ».

Exemple des formes géométriques, moyens de transport.

1. Programme complet :

On crée un objet de chacune des trois classes comme propriété d'une classe Chronos (cette classe est **dérivé** d'une autre classe « Exécutable ») dont le comportement est de « fonctionner avec notre OS ».

Chronos	Dérivée de la classe exécutable
Tictac: Oscillateur; Seconde : Compteur; Cadran : Afficheur de Texte	
Constructeur();	