JAVA 05 : PASSAGE DE LA PROGRAMMATION PROCÉDURALE À LA PROGRAMMATION ORIENTÉE OBJET (POO)

PROGRAMMATION PROCÉDURALE

Reprenons l'exemple des nombres amis. Deux nombres sont dits amis lorsque la somme des chiffres qui les composent est identique. Nous avons écrit le programme suivant :

```
import java.util.*;

public class NombreAmi {

	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		int somme1 = 0 ;
		int somme2 = 0 ;
		System.out.println("entrer un premier nombre : ");
		String nb1 = scanner.next();

		for(int i = 0 ; i < nb1.length() ; i++){
			somme1 +=  Character.getNumericValue(nb1.charAt(i));
		}
		System.out.println("entrer un deuxième nombre : ");
		String nb2 = scanner.next();

		for(int i = 0 ; i < nb2.length() ; i++){
			somme2 +=  Character.getNumericValue(nb2.charAt(i));
		}

		if ( somme1 == somme2)
			System.out.println("nombres amis");
		else
			System.out.println("pas amis");
	}
}
```

Nous avons placé tout notre code dans la méthode main(), ce qui n'est pas précisément orienté objet. De plus nous avons une redondance du code, nous avons écrit deux fois les mêmes instructions. Dans un langage procédural de type PHP, pour éviter cette redondance et pour éventuellement réutiliser ce code dans un autre programme , on pourrait écrire une fonction :

```
function sommeNombre ($nbre){
// traitement
return $somme;
}
```

Avec Java, vous avez également la possibilité d'écrire une fonction propre à la classe, en POO , on ne parle plus de fonctions mais de méthode, on va donc écrire une méthode de classe. **Une méthode de classe se définit grâce au mot clef : static**, elle est généralement **public,** c'est à dire accessible par d'autres classes ou alors private (privé) accessible uniquement dans la classe, nous verrons plus tard qu'il existe d'autres possibilités et que la distinction est plus subtile. Une méthode comme une fonction va retourner un résultat, et elle peut recevoir un ou des paramètres mais comme JAVA est un langage typé, il faut indiquer le type du retour et des paramètres. Ainsi la déclaration de votre méthode de classe, peut ressembler à :

```
public static int sommeNombre(String nombre) {
		int somme = 0 ;

		for(int i = 0 ; i < nombre.length() ; i++){
		somme +=  Character.getNumericValue(nombre.charAt(i));
		}
		return somme;
	}
```

Et notre programme :

```
public class NombreAmi {

	public static void main(String[] args) {

		Scanner scanner = new Scanner(System.in);

		System.out.println("entrer un premier nombre : ");
		String nb1 = scanner.next();

		System.out.println("entrer un deuxième nombre : ");
		String nb2 = scanner.next();

		if ( sommeNombre(nb1) == sommeNombre(nb2))
			System.out.println("nombres amis");
		else
			System.out.println("pas amis");
	}

	public static int sommeNombre(String nombre) {
		int somme = 0 ;

		for(int i = 0 ; i < nombre.length() ; i++){
		somme +=  Character.getNumericValue(nombre.charAt(i));
		}
		return somme;
	}
}
```

Les méthodes statiques sont des méthodes indépendantes de tout objet: elles n'ont pas besoin des valeurs liées à un objet pour fonctionner. Ce sont souvent des résidus de la programmation procédurale : fonctions liées aux données de types primitifs.
Par exemple : la classe Math ne contient que des méthodes statiques.

Class Math

Method Summary	
`static double`	**pow**(double a, double b) Returns the value of the first argument raised to the power of the second argument.

Une méthode statique ne s'applique pas sur un objet mais sur une classe.

Syntaxe
NomClasse.nomMethode(...,...) ;

Exemple
System.out.println (" 2 a la puissance 3 = " + Math.pow(2,3)) ;

Nous avons déclaré notre méthode public et static, c'est à dire qu'il est possible d'utiliser cette méthode dans une autre classe, sans avoir à créer un objet NombreAmi. Par exemple :

```
public class TestNombre {
   public static void main(String[] args) {
      int a = 5511;
      System.out.println(NombreAmi.sommeNombre(a));// va afficher 12
   }
}
```

PROGRAMMATION ORIENTÉE OBJET (POO)

Si la POO se limitait à définir des méthodes de classe à la place de fonctions son intérêt serait limité, en fait l'idéal pour résoudre ce problème de nombre ami aurait été de disposer d'une classe de ce type :

Nombre

Class Nombre
Object
  ↳Nombre
public class Nombre extends Object

Constructor Summary

Nombre()	Crée un nouvel objet "Nombre" avec une valeur par défaut à 0.
Nombre(int nb)	Crée un nouvel objet "Nombre" qui sera initialisé avec la valeur du paramètre "nb".

Method Summary

boolean	**estAmi**(int nb2) retourne vrai si la somme des chiffres (en base 10) qui composent le nombre est égale à la somme des chiffres (en base 10) qui composent le nombre nb2
int	**getNb()** Permet de connaître la valeur de l'état de l'objet représenté en interne par une variable de type int
void	**setNb(int nb)** Permet de modifier l'état de l'objet
int	**sommeChiffres()** calcul puis retourne la somme des chiffres (en base 10) qui composent le nombre

Écrire le programme :

NOTION DE CLASSE ET D'OBJET

La POO permet :
- la diminution du temps de développement grâce à la réutilisation de composants logiciels déjà définis (les objets)
- la réalisation d'IHM : interface homme machine : interface utilisateur graphique

La programmation orientée objet consiste donc à construire un programme composé d'objets interagissant les uns avec les autres. Chacun remplissant une tâche et réagissant en fonction de méthodes et de propriétés pré-définies.

Une classe définit la structure d'un objet, elle permet de déclarer l'ensemble des propriétés qui composent un objet. Une classe est une sorte de modèle pour les objets. Une classe est composée de deux parties:
- Les attributs : cette partie définit les attributs communs aux futurs objets de la classe.
- Les méthodes : cette partie définit les opérations applicables aux objets

REPRÉSENTATION UML D'UNE CLASSE

UML (Unified Modeling Language) : méthodologie d'analyse pour la conception d'application orientée objet.

EXEMPLE :

– int nb
+ getNb() : int + setNb(int nombre) : void + sommeChiffres() : int +estAmi (int nombre2) : boolean

SYNTAXE :

– attribut1 – attribut2 – etc....
+ methode1 + methode2

L' API JAVA D'UNE CLASSE

Nombre

Class Nombre
Object
  └Nombre
public class Nombre extends Object

Constructor Summary

Nombre()	Crée un nouvel objet "Nombre" avec une valeur par défaut à 0.
Nombre(int nb)	Crée un nouvel objet "Nombre" qui sera initialisé avec la valeur du paramètre "nb".

Method Summary

`boolean`	**`estAmi`**`(int nb2)` retourne vrai si la somme des chiffres (en base 10) qui composent le nombre est égale à la somme des chiffres (en base 10) qui composent le nombre nb2
int	**getNb()** Permet de connaître la valeur de l'état de l'objet représenté en interne par une variable de type int
void	**setNb(int nb)** Permet de modifier l'état de l'objet
int	**sommeChiffres()** calcul puis retourne la somme des chiffres (en base 10) qui composent le nombre

LA NOTION D'OBJET

DÉFINITION

Un objet est donc "issu" d'une classe, c'est le produit qui sort d'un moule. En réalité on dit qu'un objet est une instance d'une classe, c'est la raison pour laquelle on pourra parler indifféremment d'objet ou d'instance. Un objet est une variable caractérisée par des propriétés. Ces propriétés peuvent être des :

- Variables d'instances (attributs): données caractérisant l'objet.
- Opérations (méthodes) : actions que l'objet est à même de réaliser. Ces opérations sont étroitement liées aux attributs, car leurs actions peuvent dépendre des valeurs des attributs, ou bien les modifier

CRÉATION D'UN OBJET

On crée un objet à l'aide du mot clef **new** suivi de l'appel d'un des constructeurs de la classe.

Syntaxe

```
Classe nomObjet = new Constructeur( param1, … ) ;
```

Exemple

```
Nombre nb1 = new Nombre( ) ;
Nombre nb2 = new Nombre(123) ;
```

REGLES :

- Une classe possède forcément un ou plusieurs constructeurs.
- Un constructeur est une méthode servant à créer une instance d'une classe donc un objet. Plus précisément, il permet l'initialisation des différents attributs d'un objet.
- Un constructeur porte le même nom que la classe.
- Toute classe possède un constructeur par défaut, celui-ci initialise les attributs à l'aide des valeurs définies par défaut pour chaque type, il n'attend donc aucun paramètre.

MANIPULER UN OBJET

Manipuler un objet, c'est manipuler ses propriétés (attributs et méthodes) !

DEFINITION D'UN ATTRIBUT

Un attribut est une variable d'instance (variable à l'intérieur d'une autre variable). Les attributs peuvent être :

- publiques
- privés

MANIPULATION D'UN ATTRIBUT PUBLIQUE

Un attribut publique peut être directement atteint, il suffit de préciser le nom de l'objet puis le nom de l'attribut. Il n'y a donc aucun contrôle.

Syntaxe

```
momObjet.nomAttribut
```

MANIPULATION D'UN ATTRIBUT PRIVE

En POO, les attributs d'un objet sont généralement privés. Ainsi, on ne peut accéder aux attributs de l'objet que par l'intermédiaire de méthodes, appelées accesseurs. Celles-ci peuvent contrôler l'état de l'objet et garantir l'intégrité des données contenues dans l'objet. Etant donné que ces méthodes sont internes à l'objet, on dit que l'objet est responsable de lui même.=> C'est le principe d'**encapsulation**

Reprenons l'exemple de la classe Nombre :

Remarque :
• un **accesseur** en modification commence par set (appelé setter)
• un **accesseur** en lecture commence par get (appelé getter)
Un attribut privé peut être atteint à condition que des accesseurs aient été définis.

Syntaxe

```
variable = momObjet.get....( ) ;
momObjet.set....( param ) ;
```

Exemple :

```
Nombre nb2 = new Nombre( 123) ;
System.out.println(nb2.getNombre());// affiche 123
nb1.setNombre(55);// maintenant nb2 vaut 55
```

UTILISER LES MÉTHODES D'UN OBJET

Les méthodes sont généralement publiques. Pour les utiliser, il suffit de les appliquer à un objet car c'est l'objet qui exécute des opérations. Puis, il faut faire attention :

- aux paramètres attendus (respectez leur nombre et leur type)
- au résultat retourné (pensez à l'utiliser en vérifiant son type)

Syntaxe

```
nomObjet.nomMethode( [ param ,...] )
```

Exemple

```
Nombre nb2 = new Nombre( 123) ;
System.out.println(nb2.getNombre());// affiche 123
nb1.setNombre(55);// maintenant nb2 vaut 55
int somme = nb2.sommeChiffres() ; // somme vaut ?
nb2.estAmi(95) retourne ?
nb2.estAmi(64) retourne ?
```

La classe Nombre n'existe pas nativement dans JAVA, nous allons donc l'écrire