Atelier Programmation Orientée Objet Avancée –JAVA-

TP2 - Généralité

ISET Zaghouan

Enseignant

Boukchim_mossaab@yahoo.fr

Préface 1 – Nos Souvenirs –

1. Le concept de classe

Une classe est le support de l'encapsulation : c'est un ensemble de données et de fonctions regroupées dans une même entité. Une classe est une description abstraite d'un objet. Les fonctions qui opèrent sur les données sont appelées des méthodes. Instancier une classe consiste à créer un objet sur son modèle. Entre classe et objet il y a, en quelque sorte, le même rapport qu'entre type et variable.

Dans Java : tout appartient à une classe sauf les variables de types primitifs (int, float…). Pour accéder à une classe il faut en déclarer une instance de cette classe (ou un objet). Une classe comporte sa déclaration, des variables et la définition de ses méthodes. Une classe se compose de deux parties : un en-tête et un corps. Le corps peut être divisé en 2 sections : la déclaration des données et des constantes et la définition des méthodes. Les méthodes et les données sont pourvues d'attributs de visibilité qui gère leur accessibilité par les composants hors de la classe.

La syntaxe de déclaration d'une classe

```
modificateurs nom_classe [extends classe_mere] [implements interface]
{
// insérer ici les champs et les méthodes
}
```

2. Les objets

Les objets contiennent des attributs et des méthodes. Les attributs sont des variables ou des objets nécessaires au fonctionnement de l'objet. En java, une application est un objet. La classe est la description d'un objet. Un objet est une instance d'une classe. Pour chaque instance d'une classe, le code est le même, seul les données sont différentes à chaque objet.

La création d'un objet : instancier une classe

Il est nécessaire de définir la déclaration d'une variable ayant le type de l'objet désiré. La déclaration est de la forme **classe nom_variable.**

L'opérateur **new** se charge de créer une instance de la classe et de l'associer à la variable.

```
MaClasse m = new MaClasse();
```

3. La variable this

Cette variable sert à référencer dans une méthode l'instance de l'objet en cours d'utilisation. This est un objet qui est égale à l'instance de l'objet dans lequel il est utilisé.

4. Les propriétés ou attributs

Les données d'une classe sont contenues dans les propriétés ou attributs. Ce sont des variables qui peuvent être des variables d'instances, des variables de classes ou des constantes.

Les variables d'instances

Une variable d'instance nécessite simplement une déclaration de la variable dans le corps de la classe.

Exemple :

```
public class MaClasse {
public int valeur1 ;
int valeur2 ;
protected int valeur3 ;
private int valeur4 ;
}
```

Chaque instance de la classe a accès à sa propre occurrence de la variable.

5. Les méthodes

Les méthodes sont des fonctions qui implémentent les traitements de la classe.

La syntaxe de la déclaration

La syntaxe de la déclaration d'une méthode est :

```
modificateurs type_retourné nom_méthode ( arg1, ... ) {...
// définition des variables locales et du bloc d'instructions
}
```

Le type retourné peut être élémentaire ou correspondre à un objet. Si la méthode ne retourne rien, alors on utilise **void.**

6. Déclaration d'une méthode main() :

```
public static void main (String args[]) { ... }
```

Si la méthode retourne un tableau alors les [] peuvent être préciser après le type de retour ou après la liste des paramètres.

7. L'émission de messages

Un message est émis lorsqu'on demande à un objet d'exécuter l'une de ses méthodes.

La syntaxe d'appel d'une méthode est :

```
nom_objet.nom_méthode(parametre, … ) ;
```

Si la méthode appelée ne contient aucun paramètre, il faut laisser les parenthèses vides.

Préface 2 – La syntaxe et les éléments de bases de java –

1. **Règles de base**

- Les blocs de code sont encadrés par des accolades
- Chaque instruction se termine par un ";"
- Une instruction peut tenir sur plusieurs lignes.
- L'indentation (la tabulation) est ignorée du compilateur mais elle permet une meilleure compréhension du code par le programmeur.

2. **Les identificateurs**

Chaque objet, classe, programme ou variable est associé à un nom : l'identificateur qui peut se composer de tous les caractères alphanumériques et des caractères "_" et "$". Le premier caractère doit être une lettre, le caractère de soulignement ou le signe "$".

3. **Les commentaires**

Ils ne sont pas pris en compte par le compilateur donc ils ne sont pas inclus dans le pseudocode.

Ils ne se terminent pas par un ";".

Il existe trois types de commentaires en Java :

Type de commentaires	Exemple
Commentaire abrégé	`// commentaire sur une seule ligne` `int N=1; // déclaration du compteur`
Commentaire multiligne	`/* commentaires ligne 1` `commentaires ligne 2 */`
Commentaire de documentation automatique	`/** commentaire */`

4. La déclaration et l'utilisation de variables

Une variable possède un nom, un type et une valeur. La déclaration d'une variable doit donc contenir deux choses : un nom et le type de données qu'elle puisse contenir. Une variable est utilisable dans le bloc où elle est définie.

La déclaration d'une variable permet de réserver la mémoire pour en stocker la valeur.

Le type d'une variable peut être un type élémentaire ou un objet :

```
type_élémentaire variable ;
class variable ;
```

Type	Désignation	Longueur	Valeurs	Commentaires
boolean	valeur logique : true ou false	8 bits	True ou false	pas de conversion possible vers un autre type
byte	Octet signé	8 bits	-128 à 127	
short	Entier court signé	16 bits	-32768 à 32767	
char	caractère Unicode	16 bits	\u0000 à \uFFFF	entouré de cotes simples dans un programme Java
int	Entier signé	32 bits	-2×10^{9} à 2×10^{9}	
Float	virgule flottante simple précision (IEEE754)	32 bits	1.401e-045 à 3.40282e+038	
Double	virgule flottante double précision (IEEE754)	64 bits	2.22507e-308 à 1.79769e+308	
Long	Entier long	64 bits	-9×10^{18} à 9×10^{18}	

a. L'affectation

Le signe "=" est l'opérateur d'affectation, il est utilisé avec une expression de la forme :

variable = expression.

Opérateur	Exemple	Signification
=	a=10	équivalent à : a = 10
+=	a+=10	équivalent à : a = a + 10
-=	a-=	équivalent à : a = a - 10
*=	a*=	équivalent à : a = a * 10
/=	a/=10	équivalent à : a = a / 10
%=	a%=10	reste de la division
^=	a^=10	équivalent à : a = a ^ 10
<<=	a<<=10	équivalent à : a = a << 10 a est complété par des zéros à droite
>>=	a>>=10	équivalent à : a = a >> 10 a est complété par des zéros à gauche
>>>=	a>>>=10	équivalent à : a = a >>> 10 décalage à gauche non signé

b. Les comparaisons

Java propose des opérateurs pour toutes les comparaisons :

Opérateur	Exemple	Signification
>	a > 10	strictement supérieur
<	a < 10	strictement inférieur
>=	a >= 10	supérieur ou égal
<=	a <= 10	inférieur ou égal
==	a == 10	Egalité
!=	a != 10	différent de
&	a & b	ET binaire
^	a ^ b	OU exclusif binaire
|	a | b	OU binaire
&&	a && b	ET logique (pour expressions booléennes) : l'évaluation de l'expression cesse dès qu'elle devient fausse
||	a || b	OU logique (pour expressions booléennes) : l'évaluation de l'expression cesse dès qu'elle devient vraie
? :	a ? b : c	opérateur conditionnel : renvoie la valeur b ou c selon l'évaluation de l'expression a (si a alors b sinon c) : b et c doivent retourner le même type

5. Les opérations arithmétiques

Les opérateurs arithmétiques se notent **+** (addition), **-** (soustraction), ***** (multiplication), / (division) et % (reste de la division). Ils peuvent se combiner à l'opérateur d'affectation.

6. La priorité des opérateurs

Java définit les priorités dans les opérateurs comme suit (du plus prioritaire au moins prioritaire).

les parenthèses	( )
les opérateurs d'incrémentation	++ et --
les opérateurs de multiplication, division, et modulo	* / et %
les opérateurs d'addition et soustraction	+ et -
les opérateurs de décalage	<< et >>
les opérateurs de comparaison	< > <= et >=
les opérateurs d'égalité	== et !=
l'opérateur OU exclusif	^
l'opérateur ET	&
l'opérateur OU	|
l'opérateur ET logique	&&
l'opérateur OU logique	||
les opérateurs d'assignement	= += et -=

7. Les structures de contrôles

a. Les boucles

- **La boucle while**

```
while ( boolean )
{
... // code à exécuter dans la boucle
}
```

Le code est exécuté tant que le booléen est vrai. Si avant l'instruction while, le booléen est faux,

alors le code de la boucle ne sera jamais exécuté .

Ne pas mettre de ";" après la condition sinon le corps de la boucle ne sera jamais exécuté

- **La boucle do .. while**

```
do {
...
} while ( boolean )
```

Cette boucle est au moins exécuté une fois quelque soit la valeur du booléen.

- **La boucle for**

```
for ( initialisation; condition; modification) {
...
}
```

b. Les branchements conditionnels

```
if (boolean) {...
} else if (boolean) {...
} else {...
}
```

```
swith (expression) {
case constante1 :
instr11;
instr12;
break;
case constante2 :...
default :...
}
```

– Les exercices –

Exercice 1

Ecrire un programme qui calcule la somme des 100 premiers entiers et indique à l'écran le résultat.

Exercice 2

Il s'agit d'écrire un programme (classe nommée addition) qui calcule la somme des deux entiers et indique à l'écran le résultat. Les nombres n1 et n2 doivent être lu sur la ligne de commande.

Exercice 3

Il s'agit d'écrire un programme (classe nommée soustraction) qui calcule la soustraction des deux entiers et indique à l'écran le résultat. Les nombres n1 et n2 doivent être lu sur la ligne de commande.

Exercice 4

Il s'agit d'écrire un programme (classe nommée calcul) qui calcule la soustraction ou l'addition des deux entiers et indique à l'écran le résultat. Le choix de l'opération doit être lu sur la ligne de commande.

Astuce: Utilisez les deux précédentes classes.

Exercice 5

Il s'agit d'écrire un programme (classe nommée factorielle) qui calcule la factorielle des n premiers entiers et indique à l'écran le résultat. Le nombre n doit être lu sur la ligne de commande.