Table des matières

# Table des matières

1	Préambule		
2	$\mathbf{Les}$	bases	2
	2.1	Commentaires	2
	2.2	Types de variables	3
		Entrées/Sorties	
		2.3.1 Entrées	
	2.4	Boucles	4
		2.4.1 Boucle if	4
		2.4.2 Boucle Switch	5

## 1 Préambule

Java est un language orienté objet, il n'est pas possible d'y couper. Dans toute la suite, il sera supposé que vous connaissez déjà le principe de la Programmation Orienté Objet (POO).

En Java, l'exécution d'un programme va lancer la méthode main() de l'objet associé. Le code cidessous est le code minimum d'un programme Java qui défini la classe sdz1. Ce code ne fait rien.

```
public class sdz1 {
  /**
  * @param args
  public static void main(String[] args) {
  // TODO Auto-generated method stub
}
  Le même code, affichant le sempiternel "Hello World" :
import java util Scanner;
public class sdz1 {
    * @param args
  public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.out.println("Hello_World!");
  }
}
  Quelques informations de base:

    Le programme commence par le lancement de la méthode .main() de la classe du programme
```

### - Les lignes doivent se terminer par ";"

- Il faut déclarer les variables avant de les utiliser (voir [§ 2.2, p3])
- Il faut compiler le programme avant de pouvoir l'utiliser (via une plateforme java, ce n'est pas un binaire mais un bytecode utilisable uniquement par un environnement Java)

## 2 Les bases

principal.

#### 2.1 Commentaires

Il existe deux types de commentaires :

- les commentaires unilignes : introduits par les symboles //, ils mettent tous ce qui les suit en commentaires, du moment que le texte se trouve sur la même ligne que les //.

```
public static void main(String[] args){
    //Un commentaire
    //un autre
    //Encore un autre
    Ceci n'est_pas_un_commentaire_!_!_!_!
}
```

les commentaires multilignes : ils sont introduits par les symboles /\* et se terminent par les symboles \*/.

3 2 Les bases

```
public static void main(String[] args){
    /*
    Un commentaire
    Un autre
    Encore un autre
    */
    Ceci n'est_pas_un_commentaire_!_!
}
```

## 2.2 Types de variables

En Java, nous avons deux type de variables :

- des variables de type simple ou "primitif",
- des variables de type complexe ou encore des objets.

Ce qu'on appelle des types simples, ou types primitifs, en Java ce sont tout bonnement des nombres entiers, des nombres réels, des booléens ou encore des caractères. Mais vous allez voir qu'il y a plusieurs façons de déclarer certains de ces types.

Commençons par les variables de type numérique

```
- Le type byte (1 octet) peut contenir les entiers entre -128 et +127.
```

```
byte temperature;
temperature = 64;
```

- Le type short (2 octets) contient les entiers compris entre -32768 et +32767.

```
short vitesseMax;
vitesseMax = 32000;
```

- Le type int (4 octets) va de -2\*109 à 2\*109 (2 et 9 zéros derrière... ce qui fait déjà un joli nombre).

```
int temperatureSoleil;
temperatureSoleil = 15600000;
```

C'est en kelvins...

- Le type long (8 octets) de -9\*1018 à 9\*1018 (encore plus gros...).

```
long anneeLumiere;
anneeLumiere = 9460700000000000;
```

- Le type float (4 octets) correspond à des nombres avec virgule flottante.

```
float pi;
pi = 3.141592653f;
ou encore
float nombre;
nombre = 2.0f;
```

Vous remarquerez que nous ne mettons pas de virgule mais un point! Et vous remarquerez aussi que même si le nombre en question est rond, on met tout de même .0 derrière celui-ci!

- Le type double (8 octets) est identique à float, si ce n'est qu'il contient un nombre plus grand derrière la virgule.

Nous avons aussi des variables stockant du caractère

Le type char contient UN caractère stocké entre de simples quotes ' ' comme ceci...

```
char caractere;
caractere = 'A';
```

Nous avons aussi le type booléen

Le type **boolean** qui lui contient **true** (vrai) ou **false** (faux).

2.3 Entrées/Sorties 4

```
boolean question;
question = true;
```

Et aussi le type *String*. Celle-ci correspond à de la chaîne de caractères. Ici, il ne s'agit pas d'une variable mais d'un objet qui instancie une classe qui existe dans Java; nous pouvons l'initialiser en utilisant l'opérateur unaire new() dont on se sert pour réserver un emplacement mémoire à un objet (mais nous reparlerons de tout ceci dans la partie deux, lorsque nous verrons les classes), ou alors lui affecter directement la chaîne de caractères.

Vous verrez que celle-ci s'utilise très facilement et se déclare comme ceci :

```
String phrase;
phrase = "Titi_et_gros_minet";
//Deuxieme methode de declaration de type String
String str = new String();
str = "Une_autre_chaine_de_caracteres";
//La troisieme
String string = "Une_autre_chaine";
//Et une quatrieme pour la route
String chaine = new String("Et_une_de_plus_!_");
```

On peut très bien compacter la phase de déclaration et d'initialisation en une seule phase! Comme ceci :

```
int entier = 32;
float pi = 3.1416f;
char carac = 'z';
String mot = new String("Coucou");
```

Et lorsque nous avons plusieurs variables d'un même type, nous pouvons compacter tout ceci en une déclaration comme ceci :

```
int nbre1 = 2, nbre2 = 3, nbre3 = 0;
```

## 2.3 Entrées/Sorties

#### 2.3.1 Entrées

Afin de récupérer ce qu'on tape au clavier, il faut importer une nouvelle classe

```
import java.util.Scanner;
```

Voici l'instruction pour permettre à Java de récupérer ce que vous avez saisi et ensuite de l'afficher:

```
Scanner sc = new Scanner(System.in);

System.out.println("Veuillez_saisir_un_mot_:");

String str = sc.nextLine();

System.out.println("Vous_avez_saisi_:_" + str);
```

#### 2.4 Boucles

#### 2.4.1 Boucle if

```
if (//condition)
{
   // execution des instructions si la condition est remplie
}
else
{
   // execution des instructions si la condition n'est pas remplie
```

5 Les bases

```
}
Exemple:
int i = 10;
if (i < 0)
   System.out.println("Le_nombre_est_negatif");
else
   System.out.println("Le_nombre_est_positif");
Remarque: On n'est pas obligés de mettre les accolades quand il n'y a qu'une seule ligne d'instruction
dans la boucle.
  On peut aussi mettre des tests multiples :
int i = 0;
if (i < 0)
  System.out.println("Ce_nombre_est_negatif_!");
else if(i > 0)
  System.out.println("Ce_nombre_est_positif_!!");
else
  System.out.println("Ce_nombre_est_nul_!!");
  Si on souhaite faire beaucoup de tests, on peut souhaiter utiliser la structure switch à la place.
2.4.2 Boucle Switch
int nbre = 5;
switch (nbre)
  case 1:
    System.out.println("Ce_nombre_est_tout_petit");
    break;
  case 2:
    System.out.println("Ce_nombre_est_tout_petit");
    break;
    System.out.println("Ce_nombre_est_un_peu_plus_grand");
    break;
```

System.out.println("Ce\_nombre\_est\_un\_peu\_plus\_grand");

System.out.println("Ce\_nombre\_est\_la\_moyenne");

break;

break;

case 5:

case 6:

2.4 Boucles 6

```
System.out.println("Ce_nombre_est_tout_de_meme_grand");
break;

case 7:
    System.out.println("Ce_nombre_est_grand");
    break;

default:
    System.out.println("Ce_nombre_est_compris_entre_8_et_10");
}
```