COURS JAVA N°2 : STRUCTURES RÉPÉTITIVES, CHAÎNES DE CARACTÈRES, MÉTHODE *RANDOM()*

I. Structures répétitives en Java : while

Syntaxe :

```
while (condition)
{
        // TRAITEMENT
        // ne pas oublier de faire evoluer les variables qui interviennent
        // dans la condition au cours du traitement !!!!
}
```

II. Les chaînes de caractères : les classes *String* et *StringBuffer*.

II.a. La classe *String*.

La classe String permet de représenter chaque chaîne de caractères sous la forme d'un objet de cette classe disposant d'un certain nombre de méthodes (environ une cinquantaine) permettant de manipuler aisément une chaîne de caractères.
En voici quelques unes souvent utilisées :

Soit ch, une variable de type *String*.

String ch;
ch = "Java est un langage objet";

ch.length()	retourne la longueur de la chaîne de caractères associée à ch.
ch.toString()	retourne la chaîne de caractères associée à ch, c'est à dire "Java est un langage objet".
ch.indexOf('u')	retourne la position du caractère 'u' dans la chaîne de caractères ch, c'est à dire 9, car le premier caractère est à la position 0.
ch.substring(0,4)	retourne l'extrait de la chaîne de caractères commençant à la position 0 et finissant à la position 3, c'est à dire "Java". ch.substring(5) retourne l'extrait de la chaîne de caractères commençant à la position 5, c'est à dire "est un langage objet".
ch.startsWith("Java")	retourne *true* car la chaîne de caractère commence par "Java".
ch.charAt(2)	retourne le caractère 'v' car sa position est 2.
ch.toUpperCase()	retourne la chaîne de caractères mise en majuscules "JAVA EST UN LANGAGE OBJET".
ch.toLowerCase()	retourne la chaîne de caractères mise en minuscules "java est un langage objet".
ch.isEmpty()	retourne *true* si la longueur de ch vaut 0 et retourne *false* dans le cas

	contraire.

La concaténation :

Il est possible de concaténer une chaîne avec une autre en utilisant la méthode *concat(...)*.

Exemple :
String ch2 = new String("bonne chance !");
ch2 = ch2.concat(" à bientôt !");

Ceci donne à ch2 la valeur "bonne chance ! à bientôt !".

On peut aussi concaténer en introduisant le signe + :
ch2 = ch2 + " à bientôt";

II.b. La classe *StringBuffer*

Alors que la classe String considère les chaînes de caractères comme des sortes de « constantes », StringBuffer a été conçue pour développer des traitements sur des chaînes de caractères variables.
Parmi toutes les méthodes que cette classe propose, notons la méthode *append(...)* qui permet de concaténer de manière efficace une chaîne de caractères avec une autre, et la méthode *insert(...)* qui permet d'insérer une chaîne.

Exemple :

StringBuffer ch3 = new StringBuffer("il fait beau");
ch3.append(" aujourd'hui");
// La valeur de ch3 devient "il fait très beau aujourd'hui".

ch3.insert(8, "très ");
// on insère à la position 8 la chaîne "très ", la valeur de ch3 devient "il fait très beau aujourd'hui"

III. La méthode *random()* de la classe *Math* en détails.

Math.random() donne une valeur décimale comprise entre 0 et1, les deux bornes étant exclues. Pour obtenir des nombres entiers aléatoires, il faut multiplier le nombre par un coefficient, puis prendre la partie entière du résultat.

Exemple :

Soit int x = (int)(Math.random()*10). Si on multiplie la valeur aléatoire par 10, elle est alors comprise entre 0 et 10, borne exclues. Puis en plaçant (int) devant (Math.random()*10), seule la partie entière de l'opération est retenue. Le résultat est un nombre entier compris entre 0 et 9, bornes incluses.