INSTRUCTION D'ECRITURE DANS LA CONSOLE

Maintenant que nous avons déjà installé les outils de développement, et que nous avons vu comment on utilise éclipse, on va se lancer dans notre premier programme en java. L'idéale à savoir pour la suite c'est que tous les programmes Java sont composés d'au moins une **classe**. Elle doit contenir **une méthode** appelée **main** : ce sera le point de démarrage de notre programme. On parle e point de démarrage car il peut exister plusieurs autres classes, et plusieurs autres méthodes. Du coup, on peut se poser la question: ou va exactement démarrer notre programme. La réponse est simple, par la méthode **main**, méthode qui peut être inscrit dans n'importe qu'elle classe. Ce qui signifie qu'un programme n'est qu'une multitude de classe (constituer d'une multitude de méthode), qui s'inter utilise.

Une méthode est une suite d'instructions à exécuter. C'est un morceau de logique de notre programme. Une méthode contient :

- Un en-tête : celui-ci va être en quelque sorte la carte d'identité de la méthode ;
- Un corps : le contenu de la méthode, délimité par des accolades ;
- Une valeur de retour : le résultat que la méthode va retourner.

Je vous avais demandé de créer un projet Java ; ouvrez-le ! Vous voyez la fameuse classe dont je vous parlais ? Ici, elle s'appelle « First », comme à la figure suivante. Vous pouvez voir que le mot **class** est précédé du mot **public**, dont nous verrons la signification lorsque nous programmerons des objets.

First.java

```
package projet1;

public class First {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Pour le moment, ce que vous devez retenir, c'est que votre classe est définie par un mot clé (**class**), qu'elle a un nom (ici, « **First** ») et que son contenu est délimité par des accolades ({ }). Nous écrirons nos codes sources entre les accolades de la méthode main. La syntaxe de cette méthode est toujours la même :

```
public static void main(String[] args) {
	// Contenu de la page

}
```

Donc, voilà la méthode dans laquelle on va écrire nos codes pour le moment. On pourra mieux expliquer sa structure plus tard, dans la partie sur les sous-algorithme. Mais pour le moment, d'idéale est de savoir qu'une méthode est constitué : d'un entête, d'un corps, et d'une valeur de retour.

Les Commentaires

Tout a l'heure, vous avez remarqué probablement la ligne ou étais écrit :

```
// Contenu de la page
```

On n'a pas écrit tout simplement : `Contenu de la page`

Car, il y'a des instructions qu'on écrit dans le code, mais qu'on ne voudrait pas que le compilateur le compile en byte code… bref, le considère. C'est ce qu'on appelle des **commentaires**.

Deux syntaxes sont disponibles pour commenter son code :

1. Les commentaires unilignes : introduits par les symboles « // », ils mettent tout ce qui les suit en commentaire, du moment que le texte se trouve sur la même ligne ;
2. Les commentaires multilignes : ils sont introduits par les symboles « /* » et se terminent par les symboles « */ ».

```
public static void main(String[] args) {
	//Un commentaire
	//Un autre
	//Encore un autre

	/*  Un commentaire
	    Un autre
	    Encore un autre */
	Ceci n'est pas un commentaire !
}
```

Ainsi, les commentaires vont vous permettre de commenter votre code. Pour vous-même, mais surtout pour tout lecteur potentiel qui ne sera pas vous.

Exemple :

- Au début de la définition d'une classe, vous pouvez dire en commentaire a quoi la classe va exactement servir…
- Pareillement lors de la définition d'une méthode…

La console

Durant toute cette partie consacrée à l'étude du java général, nous allons essentiellement travailler avec une console. C'est une sorte d'écran d'affichage prédéfini, qui a travers ces différents méthodes, va nous permettre d'afficher les résultats de nos traitements.

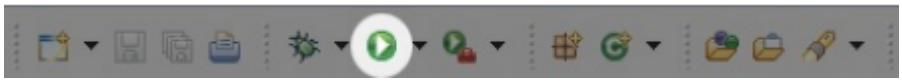
Ainsi, pour afficher un message a l'écran, on peut utiliser deux méthodes :

1. System.out.print("message") : qui va afficher le message texte en paramètre dans la console
2. System.out.println("message") : qui fait la même chose que la méthode précédente, mais après le message, provoque un retour à la ligne.

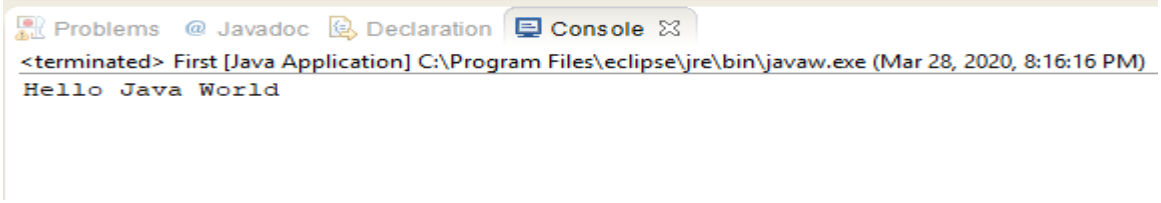
```
Public static void main(String[] args) {
        System.out.print("Hello Java World");
}
```

- *N'oublier pas, à la fin d'une instruction en programmation java, il y'a toujours un point-virgule.*

Une fois que vous avez saisi cette ligne de code dans votre méthode main, il vous faut lancer le programme. Si vous vous souvenez bien de la présentation faite précédemment, vous devez cliquer sur la flèche blanche dans un rond vert, comme à la figure suivante.



Si vous regardez dans votre console, dans la fenêtre du bas sous Eclipse, vous devriez voir quelque chose ressemblant à la figure suivante.



Expliquons un peu cette ligne de code. Littéralement, elle signifie « la méthode **print()** va écrire « Hello World ! » en utilisant l'objet **out** de la classe **System** ». Quelques précisions :

- **System** : ceci correspond à l'appel d'une classe qui se nomme « System» . C'est une classe utilitaire qui permet surtout d'utiliser l'entrée et la sortie standard, c'est-à-dire la saisie clavier et l'affichage à l'écran.
- **out** : objet de la classe System qui gère la sortie standard.
- **print** : méthode qui écrit dans la console le texte passé en paramètre (entre les parenthèses).

Pareillement pour la méthode **println()**

```
System.out.print("Hello World !");
System.out.print("My name is");
System.out.print("Mané");
```

Java code

```
Hello World !My name isMané
```

Console code

Pas mal déjà. Mais j's sûr qu'il y en a qui se disent, mais ça serai plus cool si "Hello World !" étais sur sa ligne tout seule, et Mané aussi, mais en plus Mané tabuler. Alors, essayons ça. D'abord on sait que pour aller à la ligne a après l'écriture du message, on doit utiliser println(). Cependant, avec des caractères spéciaux, on peut tout autant utiliser print(), et aller à la ligne, et tabuler. On a :

- « \n » : caractère spécial pour un retour à la ligne
- « \t » : caractère spécial pour une tabulation

Soit :

```
System.out.println("Hello World !");
System.out.print("My name is\n");
System.out.print("\tMané");
```

Java code

```
Hello World !
My name is
      Mané
```

Console code

- *Vous avez sûrement remarqué que la chaîne de caractères que l'on affiche est entourée par des « " ». En Java, les guillemets doubles sont des délimiteurs de chaînes de caractères ! Si vous voulez afficher un guillemet double dans la sortie standard, vous devrez « l'échapper » avec un « \ », ce qui donnerait : "Coucou mon \"chou\" !". Il n'est pas rare de croiser le terme anglais quote pour désigner les guillemets droits. Cela fait en quelque sorte partie du jargon du programmeur.*