


# TP 1 : Premiers pas en Java

## I. Découverte de l'invite de commandes *Windows*

Avant de commencer à programmer en *Java*, nous allons d'abord nous familiariser avec l'invite de commandes *Windows* et préparer notre espace de travail pour les travaux pratiques. Pour plus d'informations sur l'invite de commandes, consultez l'article  [Invite de commandes Windows](#).

1. Ouvrez l'invite de commandes *Windows* via le menu démarrer.
2. Si ce n'est pas déjà le cas, placez-vous dans votre répertoire de travail. L'ensemble des fichiers réalisés lors des travaux pratiques sera sauvegardé dans un répertoire `info0101`.
3. Créez le répertoire `info0101`.
4. Entrez dans le répertoire `info0101`.
5. Créez le répertoire `tp1`.

Votre répertoire de travail pour cette séance est maintenant prêt. N'oubliez pas de créer un répertoire à chaque début de séance.

## II. Utilisation de Notepad++

Pour écrire un programme *Java*, nous avons besoin d'utiliser un éditeur de texte. Pour `Info0101`, nous avons fait le choix d'utiliser *Notepad++*.

1. Cherchez *Notepad++* dans le menu démarrer de *Windows* et démarrez-le.

À l'ouverture de *Notepad++*, un document vierge est créé. Pour bénéficier de la coloration syntaxique, il faut sauvegarder le fichier avec l'extension `.java`.

2. Dans le menu *Fichier*, choisissez *Enregistrer sous...*. Placez-vous dans votre répertoire personnel, puis dans le répertoire `info0101`, puis dans le répertoire `tp1`. Spécifiez `Exemple.java` comme nom de fichier.

Nous allons maintenant écrire notre premier programme *Java*.

3. Dans le document vierge, copiez le code suivant :

```
class Exemple {  
  
}
```

4. Enregistrez les modifications, puis retournez sur l'invite de commandes *Windows*.
5. Pour vous assurer que vous vous trouvez bien dans le bon répertoire, listez les fichiers. Normalement, vous devriez voir le fichier `Exemple.java`.

Pour compiler ce programme *Java*, nous allons utiliser le compilateur nommé `javac`.

6. Tapez la commande suivante :

```
javac Exemple.java
```

Normalement, la compilation a réussi et le fichier `Exemple.class` est apparu dans le répertoire.

7. Vérifiez la présence du fichier `Exemple.class` en listant les fichiers du répertoire courant.

Le fichier `Exemple.class` correspond à du *bytecode*. Il n'est pas possible de l'exécuter directement. Pour exécuter un programme *Java*, il faut utiliser la commande *java*.

8. Tapez la commande suivante. Un message d'erreur sera ensuite normalement affiché à l'écran.

```
java Exemple
```



Alors qu'il faut spécifier l'extension `.java` lorsqu'on utilise le compilateur `javac`, il ne faut pas préciser l'extension `.class` lors de l'exécution avec `java`.

Pour qu'une classe *Java* puisse produire un programme exécutable, il faut ajouter la méthode principale (ou *main* en anglais).

9. Modifiez le code de `Exemple.java` par le code suivant :

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



N'oubliez pas de sauvegarder le fichier dans *Notepad++* à chaque modification (lorsque le fichier est modifié sans être sauvegardé, l'icône devant le nom est de couleur rouge et bleue sinon). Une fois le fichier sauvegardé, n'oubliez pas non plus de recompiler avec `javac` !

10. Recompilez et exécutez à nouveau. Votre programme s'exécutera normalement bien mais n'affichera rien.

Comme la méthode principale ne contient aucune instruction, rien ne se produit à l'écran. Nous allons modifier le programme en utilisant `System.out.println` pour afficher un message à l'écran.

11. Modifiez le programme comme suit :

```
class Exemple {  
  
    public static void main(String[] args) {  
        System.out.println("Bonjour tout le monde !");  
    }  
  
}
```

Vous venez d'écrire votre premier programme *Java*. Félicitations !

### III. Affichage en Java

La plupart des programmes *Java* que nous écrirons dans les différents TP ont besoin d'interagir avec l'utilisateur en affichant des informations à l'écran ou en demandant à l'utilisateur de saisir des données. Comme nous l'avons vu précédemment, l'affichage est réalisé à l'aide de `System.out.println`.

1. Créez un nouveau fichier *Java* nommé `Saisie.java` et dont le contenu est le suivant :

```
class Saisie {  
  
    public static void main(String[] args) {  
        System.out.println("Bonjour");  
        System.out.println("tout le monde !");  
    }  
  
}
```

2. Compilez ce programme dans l'invite de commandes puis exécutez-le.

Avec `System.out.println`, les informations sont affichées à l'écran et un saut de ligne est ajouté.

3. À la place du premier `System.out.println`, écrivez `System.out.print` (sans le `ln`). Sauvegardez, recompilez et exécutez votre programme.



Les chaînes sont accolées car aucun espace n'est ajouté par défaut. Vous pouvez essayer d'ajouter un espace soit après `Bonjour` (avant le guillemet qui ferme la chaîne), soit avant `tout le monde` (après le guillemet qui ouvre la chaîne).

Si vous avez besoin d'afficher plusieurs informations sur une seule ligne, il est donc maintenant possible d'utiliser plusieurs `System.out.print`, en lieu et place d'un seul `System.out.println`.

## IV. Saisies en Java

Les programmes ont aussi besoin de demander à l'utilisateur de saisir des données. Dans un premier temps, nous allons demander à l'utilisateur de saisir un entier au clavier. En programmation, ce n'est pas généralement une chose simple : il faut en effet être capable de traduire les caractères correspondant à un entier (ceux tapés sur le clavier par l'utilisateur) en un entier. Pour nous faciliter la tâche, l'API *Java* nous propose un ensemble d'outils.

1. Modifiez le programme précédent comme suit puis compilez-le. Un message d'erreur sera normalement affiché à la compilation.

```
class Saisie {  
  
    public static void main(String[] args) {  
        System.out.print("Bonjour ");  
        System.out.println("tout le monde !");  
  
        Scanner clavier = new Scanner(System.in);  
  
        int a;  
        a = clavier.nextInt();  
    }  
}
```

L'API *Java* est très riche et contient de très nombreux outils (ou classes) qui peuvent avoir le même nom. Il est donc nécessaire d'indiquer au compilateur *Java* où trouver les différents outils utilisés dans le programme.

2. Avant la déclaration de la classe, ajoutez la ligne suivante puis compilez et exécutez le programme. Le programme sera normalement suspendu pendant son exécution.

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

Comme votre programme n'indique pas à l'utilisateur qu'il doit saisir un entier, il est nécessaire d'afficher des indications à l'utilisateur en lui demandant, par exemple, de saisir un entier.

3. Modifiez le programme en conséquence.
4. Modifiez maintenant le programme pour afficher le contenu de la variable qui contient la valeur saisie par l'utilisateur.

Vous savez maintenant afficher des informations à l'écran et demander à l'utilisateur de saisir des entiers. Nous reviendrons sur les subtilités des saisies un peu plus tard.

## Exercice 1 - La somme

Écrivez un programme *Java* qui demande à l'utilisateur de saisir deux entiers puis qui affiche la somme de ces deux entiers. Voici un exemple d'affichage que votre programme doit produire :

```
Valeur 1 : 12  
Valeur 2 : 14  
La somme de 12 + 14 = 26
```

## Exercice 2 - Le signe d'un entier

Écrivez un programme qui affiche le signe d'un nombre entier saisi au clavier par l'utilisateur. Si l'utilisateur saisit 0, le programme doit afficher **Vous avez saisi 0**.

## Exercice 3 - La calculatrice

1. Écrivez un programme qui réalise différentes opérations de base. L'utilisateur saisit 2 entiers et choisit un opérateur parmi '+', '-' ou '\*'. Le programme affiche ensuite le résultat correspondant.



Pour cette première version, utilisez une ou plusieurs conditionnelles.

2. Nous souhaitons maintenant prendre en compte la division. Modifiez votre programme pour prendre en compte la division en portant attention à gérer le cas d'erreur possible. En cas d'erreur, un message doit être affiché à l'écran.
3. Modifiez votre programme en utilisant un **cas parmi** en lieu et place de conditionnelles.



À la fin du TP, n'oubliez pas de créer une archive .zip avec votre répertoire tp1 et de déposer cette archive dans le dépôt du TP !