

II.1102 – Algorithmique et Programmation

TP 1 : Introduction à Java

Patrick Wang

02 Octobre 2020

1 Objectifs du TP

- Se familiariser avec l’environnement de développement
- Premiers pas avec le langage Java
- Manipulations de quelques variables

2 Découverte de l’IDE

2.1 Installation de Java

Si vous utilisez les ordinateurs de l’ISEP, Java ainsi qu’Eclipse sont déjà installés. Vous n’avez donc rien à faire de particulier. Si vous souhaitez utiliser vos propres machines, commencez par vérifier si Java est installé dessus en suivant les étapes ci-dessous :

1. Ouvrir votre terminal ou ligne de commandes ;
2. Saisir l’instruction `java -version`. Si Java est installé sur votre machine, cette instruction va afficher la version de Java présente. Sinon, une erreur va s’afficher.

Pour télécharger le kit de développement Java, il faut vous rendre à l’adresse suivante : <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html> et télécharger la version correspondante à votre système d’exploitation. **Attention, il est nécessaire de télécharger Java 8.**

Une fois le téléchargement terminé, installez le kit de développement en suivant les consignes à l’écran.

2.2 Installation d’un environnement de développement

L’environnement de développement est le logiciel utilisé pour gérer et écrire des programmes en Java. À l’ISEP, l’outil utilisé est Eclipse. Ce logiciel est gratuit, est vous pourrez le télécharger ici : <https://www.eclipse.org/downloads/packages/release/2019-06/r/eclipse-ide-java-developers>.

Un autre environnement de développement populaire est IntelliJ. Si vous préférez utiliser cet outil, sachez que vous pouvez avoir accès à la version *Ultimate* en demandant une licence étudiante. Vous pouvez télécharger IntelliJ ici : <https://www.jetbrains.com/idea/download/#section=windows>.

Lorsque votre environnement de développement préféré est installé, lancez-le pour démarrer ce TP.

3 Création d'un nouveau projet

Quel que soit l'environnement de développement utilisé, celui-ci doit pouvoir vous aider à créer un nouveau projet. Suivez les instructions affichées à l'écran, puis créez un nouveau projet intitulé `TP1_GroupeAPP_Noms`. Par exemple, si vous êtes du Groupe 1 et que vous travaillez seul, votre projet pourra s'appeler `TP1_G1_Dupont`.

Si vous utilisez IntelliJ, la création du projet entraîne aussi la création d'un fichier (aussi appelé *classe* en Java) intitulé `Main.java`. Par contre, si vous utilisez Eclipse, il faudra le créer vous-même en suivant les instructions décrites dans le cours, ou rappelées ici : <https://youtu.be/SZwT5ePz2Wg>.

4 Interaction avec l'utilisateur

Un grand nombre de programmes sont interactifs. Cela signifie que l'utilisateur peut saisir des données qui seront ensuite utilisées par le programme ou lire des informations fournies par le programme. En Java, ces deux types d'interactions sont possibles.

Question : Analysez le code suivant, et prévoyez son résultat.

```
public class Main {
    public void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int unEntier = scanner.nextInt();
        float unReel = scanner.nextFloat();

        System.out.println("J'ai recupere un entier: " + unEntier);
        System.out.println("J'ai aussi recupere un reel: " + unReel);
    }
}
```

Question :

- Sans recopier le programme ci-dessus ni l'exécuter, prévoyez son résultat ;
- Selon vous, et toujours sans exécuter le programme, que font les instructions `scanner.nextInt()` et `scanner.nextFloat()` ?
- Recopiez le contenu de la fonction `main()` dans votre classe, puis lancez le programme. Que se passe-t-il ?

4.1 Affichage de messages en console

L'instruction `System.out.println()` permet d'afficher des informations en console. Plus précisément, cette fonction va afficher le message en console, puis effectuer un saut de ligne. Nous allons dans cette partie nous familiariser avec cette instruction que nous allons souvent utiliser. D'ailleurs, cette instruction est tellement utilisée qu'il existe un raccourci sur Eclipse et sur IntelliJ pour taper plus rapidement `System.out.println()` :

- Pour IntelliJ, il faut saisir `sout` puis appuyer sur la touche `TAB` ou `ENTREE` pour enclencher l'autocomplétion ;
- Pour Eclipse, il faut saisir `sysout` puis appuyer sur la touche `TAB` ou `ENTREE` pour enclencher l'autocomplétion.

Questions :

1. Commencez par commenter le contenu de la fonction `main()`.

2. Ajoutez une instruction permettant d'afficher le message suivant à l'utilisateur : « Bonjour, quel est votre prénom ? ».
3. Vérifiez que votre programme affiche bien ce message.

La classe `Scanner` permet de lire des informations saisies en console. Le programme précédent montre deux exemples d'utilisation de cette classe avec les instructions `nextInt()` et `nextFloat()`. Pour lire une chaîne de caractères en console, on pourra utiliser l'instruction `nextLine()`.

Questions :

1. Ajoutez une instruction permettant de récupérer le prénom de l'utilisateur une fois qu'il a été saisi.
2. Affichez à la fin de ce programme le message de bienvenue suivant : « Bonjour, *prénom* », où *prénom* est le prénom de l'utilisateur saisi un peu plus tôt.

5 Exercices

Pour chacun de ces exercices, nous allons créer une nouvelle *fonction*. Nous verrons plus tard l'utilité des fonctions, et nous nous concentrons sur le moment uniquement sur leur syntaxe et leur utilisation.

Pour créer une nouvelle fonction, nous allons **pour le moment** écrire les quelques lignes suivantes sous la fonction `main()`. La classe `Main` ressemble alors à ça :

```
public class Main {
    public static void main(String[] args) {
        // Des choses ici
    }

    // Attention a bien recopier 'public static void'
    public static void maFonction() {
        // Contenu de la fonction
    }
}
```

Pour tester votre fonction, il faudra *l'appeler* dans la fonction `main()`. Si l'on souhaite appeler la fonction `maFonction()`, le contenu de la classe `Main` deviendrait alors :

```
public class Main {
    public static void main(String[] args) {
        maFonction();
    }

    public static void maFonction() {
        // Contenu de la fonction
    }
}
```

5.1 Somme de deux entiers

Le programme suivant permet de calculer la somme de deux entiers, mais les instructions ont été mélangées.

```
public static void somme() {  
    int deuxiemeEntier = scanner.nextInt();  
    System.out.println("La somme de " + premierEntier + " avec " + deuxiemeEntier +  
        "est egale a " + somme);  
    int somme = premierEntier + deuxiemeEntier;  
    System.out.println("Veuillez saisir le premier entier");  
    Scanner scanner = new Scanner(System.in);  
    int premierEntier = scanner.nextInt();  
    System.out.println("Veuillez saisir le deuxieme entier");  
}
```

Question :

1. Remettez les instructions dans l'ordre afin que le programme calcule effectivement la somme de deux entiers.
2. Testez votre programme pour vous assurer de son bon fonctionnement.

5.2 Division entre deux entiers

Questions :

1. Créez une fonction intitulée `division()` en vous aidant de ce qui a été fait avec la fonction `somme()` ;
2. Complétez cette fonction pour pouvoir calculer la division de deux entiers saisis par l'utilisateur ;
3. Testez votre programme afin de calculer la valeur de $5/3$;
4. Si votre programme affiche une erreur, prenez le temps de bien la lire et de tenter de la corriger par vous-même.

Remarque : L'utilisateur peut très bien demander à votre programme de calculer $0/0$, ce qui est évidemment impossible. Nous verrons plus tard comment empêcher l'utilisateur de faire ce genre d'actions malveillantes, en sécurisant notre code.

5.3 Calcul du volume d'un pavé droit

On souhaite écrire un programme permettant de calculer le volume d'un pavé droit (exemple en Figure 1). Suivez les étapes décrites ci-après pour concevoir ce programme.

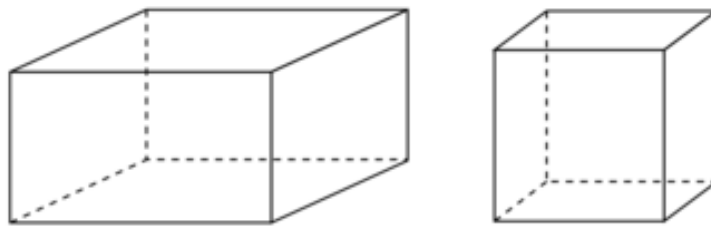


FIGURE 1 – Exemples de pavés droits.

Questions :

1. De combien de variables avons-nous besoin pour faire ce calcul ?
2. Pour chacune de ces variables, quel est son type ?
3. Comment pouvons-nous obtenir les valeurs de ces variables ?
4. Quelle est la formule à utiliser pour calculer le volume d'un pavé droit ?
5. Que doit-on faire du résultat ?

Une fois que vous avez répondu à ces questions, implémentez votre programme dans une nouvelle fonction appelée `volume()`, puis testez votre programme avec différentes valeurs.

Questions : Quels sont les problèmes présentés par ce programme ?

6 Remarques

La classe `Scanner` va être très souvent utilisée lors de ce semestre. Les différentes instructions présentées dans ce TP démontrent son utilité. Il y a pourtant quelques subtilités à connaître.

Les différentes instructions que l'on a vu dans ce TP permettent de récupérer les saisies des utilisateurs puis de les stocker dans des variables de types spécifiques. Cette action se fait au moment où la touche « Entrée » est pressée. Or, cette touche « Entrée » ajoute un caractère invisible à la chaîne de caractères saisie : le caractère `newline` ou `\n`.

Pour le `Scanner`, la subtilité est que les instructions `nextInt()`, `nextFloat()`, etc. ne vont pas lire ce caractère de retour à la ligne. Cela peut entraîner des comportements parfois difficiles à déboguer.

Par exemple, considérons le code suivant (n'hésitez pas à le recopier dans une nouvelle fonction, puis d'appeler celle-ci dans votre fonction `main()`) :

```
System.out.println("Saisir un entier");
// On saisit '11'
int entier = scanner.nextInt();
System.out.println("Saisir une operation");
// Parce que le caractere newline n'a pas ete lu, c'est lui qui va se
// retrouver dans la variable operation
// Il va aussi etre impossible de saisir une autre valeur pour operation
String operation = scanner.nextLine();
```

Il convient donc de faire très attention lorsque l'on demande à l'utilisateur de saisir plusieurs valeurs de types différents. L'utilisation de l'instruction `nextLine()` peut vous induire en erreur.