II.1102 – Algorithmique et Programmation
TP 1 : Introduction à Java

1 Objectifs du TP

- Se familiariser avec l'environnement de développement
- Premiers pas avec le langage Java
- Manipulations de quelques variables
- Premiers exercices avec les structures conditionnelles et itératives

2 Découverte de l'IDE

2.1 Installation d'un environnement de développement : IntellijIDEA Community Edition

- Lien de téléchargement : https://www.jetbrains.com/idea/
- Guide d'installation : https://www.jetbrains.com/help/idea/installation-guide.html

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.

Avec IntelliJ, la création du projet entraîne aussi la création d'un fichier (aussi appelé classe en Java) intitulé Main.java. Cependant, 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 ?

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'autocomplétions ;

Pour Eclipse, il faut saisir **sysout** puis appuyer sur la touche TAB ou ENTREE pour enclencher l'autocomplétions.

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().**

4. Ajoutez une instruction permettant de récupérer le prénom de l'utilisateur une fois qu'il a été saisi.
5. 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 deuxième 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.

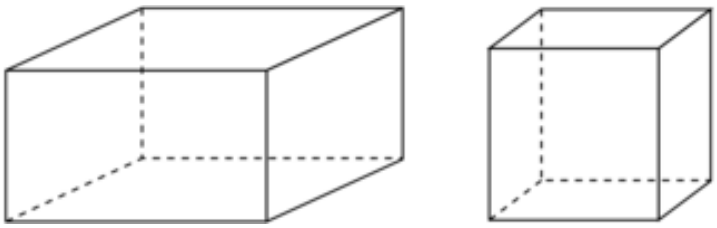

FIGURE 1 – Exemples de pavés droits.

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 fonc- tion, 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.

7 Les structures conditionnelles et itératives

7.1 Calcul du discriminant du second degré

Pour rappel, le discriminant est obtenu grâce à la formule suivante :

Soit P (x) = $ax^2 + bx + c$ avec (a, b, c) ∈ R^3, Δ = b2 – 4ac

Le programme suivant permet de calculer le discriminant d'un polynôme du second degré.

Puis, il est capable de traiter le cas Δ < 0. Cependant, les instructions ont été mélangées.

Question : Remettez les instructions dans l'ordre afin que le programme calcule effectivement le discriminant d'un polynome du second degré. On fera particulièrement attention à bien indenter le code une fois les instructions remises dans l'ordre.

```
public static void discriminant() {
     int delta = (int) (Math.pow(b, 2) - 4 * a * c);
     int c = scanner.nextInt();
     System.out.println("Quelle est la valeur de a ?");
    System.out.println("Ce polynome n'a pas de racine reelle");
    System.out.println("Quelle est la valeur de b ?");
    }
    int a = scanner.nextInt();
    int b = scanner.nextInt();
    if (delta < 0) {
    System.out.println("Quelle est la valeur de c ?");
    Scanner scanner = new Scanner(System.in);
}
```

Lorsque le discriminant est positif ou nul, le polynôme présente une racine double ou deux racines distinctes. Pour rappel :

- Si Δ = 0, on a une racine double $x_0 = \frac{-b}{a}$
- Δ > 0, on a deux racines $x_0 = \frac{-b+\sqrt{\Delta}}{2a}$ et $x_1 = \frac{-b-\sqrt{\Delta}}{2a}$

Questions :

- Modifiez le programme afin de traiter le cas **Δ = 0**. Note : On pourra utiliser **Math.sqrt()** afin de calculer la racine carrée d'un nombre.
- Modifiez le programme afin de traiter le cas Δ > 0.

Lorsque le discriminant est strictement négatif, on dit que le polynôme présente deux racines complexes

$$x_0 = \frac{-b + i\sqrt{-\Delta}}{2a}, \quad x_1 = \frac{-b - i\sqrt{-\Delta}}{2a}$$

Questions : Modifiez le programme afin de traiter le cas Δ < 0. On ne fera qu'afficher le résultat sous forme de chaînes de caractères (en concaténant variables et texte).

7.2 Calcul de la parité d'un nombre

Questions :

1. Créez une fonction **parite();**
2. Complétez cette fonction pour demander à l'utilisateur de saisir un entier ;
3. Si cet entier est pair, affichez un message indiquant que ce chiffre est pair, tout en rappelant sa valeur ;
4. Si cet entier est impair, affichez un message indiquant que ce chiffre est impair, tout

en rappelant sa valeur ;

7.3 Calcul d'extremum Questions :

1. Créez une fonction **max();**
2. Complétez cette fonction pour demander à l'utilisateur de saisir deux entiers
3. Déterminez le maximum de ces deux entiers, puis affichez la valeur du maximum à l'utilisateur ;
4. Reprendre les étapes 1 – 3, mais cette fois-ci pour déterminer le minimum entre deux entiers.

7.4 Structures itératives

Le programme suivant permet de calculer la factorielle d'un entier, mais les instructions ont été mélangées. Pour rappel, la factorielle d'un entier est définie de la façon suivante :

$$\forall n \in \mathbb{N}^*, n! = 1 \times \ldots \times n \, et \, 0! = 1$$

```
public static void factorielle() {
    }
    factorielle *= i;
    for (int i = 0; i <= n; i++) { int n =
    scanner.nextInt();
    System.out.println(n + "! = " + factorielle); Scanner scanner = new
    Scanner(System.in);
    int factorielle = 1;
    System.out.println("Saisir un entier positif ou nul");
}
```

Questions :

1. Remettez les instructions dans l'ordre afin que le programme calcule effectivement la factorielle d'un entier.
2. Testez votre programme avec n = 1. Que se passe-t-il ? Corrigez le programme puis testez une nouvelle fois avec cette valeur.
3. Testez désormais votre programme avec des valeurs quelconques afin de vérifier qu'il est correct. Corrigez-le si ce n'est pas le cas.

7.5 Compte à rebours Questions :

1. Créez une fonction **countdown()** ;
2. Complétez cette fonction pour afficher un compte à rebours allant de 10 jusqu'à 0 ;
3. Une fois que le compte à rebours a atteint 0, affichez le message « BOOM ! »

7.6 Valeurs de carrés Questions :

1. Créez une fonction **carres ()** ;

2. Complétez cette fonction afin d'afficher côte à côte les valeurs de x et de x2. Pour cet exercice, nous privilégierons la multiplication de deux termes plutôt que la fonction **Math.pow()** et nous séparerons les valeurs par des tabulations.

7.7 Tables de multiplication

Pour cet exercice, nous souhaitons afficher la table de multiplication affichée en Figure 1.

```
1   2   3   4   5   6   7   8   9   10
2   4   6   8   10  12  14  16  18  20
3   6   9   12  15  18  21  24  27  30
4   8   12  16  20  24  28  32  36  40
5   10  15  20  25  30  35  40  45  50
6   12  18  24  30  36  42  48  54  60
7   14  21  28  35  42  49  56  63  70
8   16  24  32  40  48  56  64  72  80
9   18  27  36  45  54  63  72  81  90
10  20  30  40  50  60  70  80  90  100
```

FIGURE 1 – Table de multiplication à afficher

Pour cela, nous allons utiliser la fonction **System.out.print()** (et non pas println()) pour afficher du texte sans créer de nouvelle ligne à la fin. De plus, nous utiliserons des tabulations pour séparer les éléments sur chaque ligne.

Nous allons dans un premier temps nous concentrer sur l'affichage de la première ligne.

Questions :

1. Quel est le type de boucle le plus approprié pour afficher la première ligne ?
2. Créez une fonction tableMultiplication();
3. Complétez ce programme pour afficher la première ligne de la table de multiplication.

La table de multiplication est construite de sorte que :

4. Les éléments de la ligne 1 sont les éléments de la ligne 1 multipliés par 1 ;