**TP2 : Types simples et instructions de bases Java**

**Objectifs:** Le but de ce TD est de vous faire expérimenter les constructions de base du langage JAVA (types simples, instructions de contrôle, itérations) qui sont très proches syntaxiquement de celles utilisées par le langage C. Les programmes que vous aurez à écrire ne feront donc que très peu appel aux notions d'objets et classes, mais avant d'expérimenter celles-ci, il paraît préférable de vous assurer que vous maîtrisez correctement ces constructions élémentaires. De plus, cela vous permettra de parfaire votre maitrise des outils de base (compilateur javac, interpréteur java) du JDK vus lors du [TP1](https://lig-membres.imag.fr/genoud/teaching/PL2AI/tds/PL2/sujets/tp01_jdk/tp01_jdk.html).

**Ressource utiles**

* Les types simples (entiers, réels, ...) et les instructions de base du langage JAVA:
  + les [slides de présentation du cours](https://lig-membres.imag.fr/genoud/teaching/PL2AI/cours/pdf/PL2/PL2_02_TypesSimples.pdf).
* Pour effectuer des lectures au clavier:
  + utilisez la classe **Scanner** (javadoc : [JDK 17](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Scanner.html) ou bien [JDK 8](https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html)) du package standard **java.util** (dans le module **java.base** pour versions du JDK supérieures à 9). Dans le fichier [ScannerDemo.java](https://lig-membres.imag.fr/genoud/teaching/PL2AI/tds/PL2/sujets/tp02_typesSimples/src/ScannerDemo.java) fourni avec ce TP vous trouverez un exemple d'utilisation de cette classe pour lire des données au clavier.

Les exercices proposés dans ce TD sont indépendants et sont organisés par thème (expressions, instructions conditionnelles, itérations. Les thèmes sont présentés selon un ordre de difficulté croissant.

Chaque thème comporte plusieurs exercices. Traitez ces thèmes selon leur ordre de présentation en effectuant au moins un des exercices proposés. Lorsque vous aurez abordé tous les thèmes, vous pourrez revenir en arrière pour réaliser les exercices que vous n'avez pas encore traités.

**Rappel:** tout programme source JAVA doit avoir comme suffixe .java. La version compilée d'un programme JAVA a le même nom que le programme source correspondant mais son suffixe est .class.

Par exemple le programme source MonTP2.java, se compilera en un fichier bytecode MonTP2.class (obtenu par la commande javac MonTP2.java) , et sera exécuté par la commande java MonTP2.

**Style de codage:** Adopter un style de codage uniforme et cohérent est un principe de base dans l'activité de programmation : cela facilite la lecture et la compréhension du code.

La communauté des développeurs Java a l'habitude de respecter un certain nombre de règles de base, parmi lesquelles:

1. Les noms de classes sont définis par un identificateur débutant par une majuscule.
2. Les noms de variables locales sont définis par un identificateur débutant par une minuscule.
3. Les programmes doivent être correctement indentés.

Prenez dès maintenant de bonnes habitudes de codage Java. **Il vous est demandé de respecter impérativement les 3 règles ci-dessus**.

Pour en savoir plus sur les bonnes pratiques, vous pouvez vous référer au document de Google: [Google Java Style](https://google.github.io/styleguide/javaguide.html) (https://google.github.io/styleguide/javaguide.html)

**Exercice 0 : Organiser votre espace de travail**

Avant tout chose organisez votre espace disque afin de pouvoir par la suite retrouver facilement le travail effectué lors de cette séance de TP. Pour cela, dans le répertoire **TPS/Java** créé lors du TP1, ajoutez un sous répertoire **TP02** dans lequel vous rangerez le travail effectué au cours ce second TP.

**Thème 1 : Expressions**

**Exercice 1 : Conversion de températures**

En utilisant la formule \(tempC = (5/9)(tempF-32)\) écrire en langage JAVA un programme Degres.java qui lit une température exprimée en degrés Fahrenheit et affiche sa valeur en degrés centigrades ou degrés Celsius.

exemples d'exécution du programme :

C:\JAVA\TP2>java Degres

donnez une temperature en Fahrenheit : O.O

cette temperature equivaut a -17.8 degres Celsius

C:\JAVA\TP2>java Degres

donnez une temperature en Fahrenheit : 60.0

cette temperature equivaut a 15.6 degre Celsius

C:\JAVA\TP2>

**Exercice 2 : Conversion de durées**

Ecrire un programme Hjms.java en langage JAVA qui pour un nombre de secondes donné calcule et affiche son équivalent en nombre de jours, d'heures, de minutes et de secondes.

exemples d'exécution du programme :

C:\JAVA\TP2>**java Hjms**

donnez une durée en secondes : **3621**

cette durée équivaut à 0 jours 1 heures 0 minutes 21 secondes

C:\JAVA\TP2>**java Hjms**

donnez une durée en secondes : **567231**

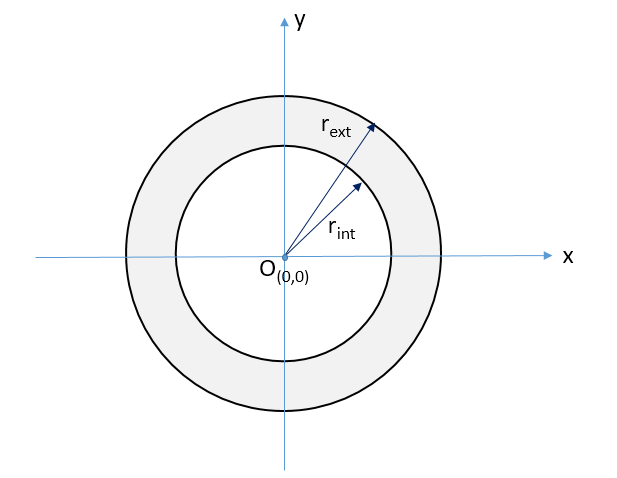
cette durée équivaut à 6 jours 13 heures 33 minutes et 51 secondes

C:\JAVA\TP2>

**Thème 2 : Instructions conditionnelles**

**Exercice 1 : Déterminer de l'appartenance d'un point à une couronne**

Ecrire un programme Couronne.java en langage JAVA qui pour un pour un point P du plan détermine si ce point se trouve ou non à l'intérieur de la couronne de centre l'origine et définie par la donnée de son rayon extérieur rext et son rayon intérieur rint



C:\JAVA\TP2>**java Couronne**

rayon extérieur : **14**

rayon intérieur : **10**

donnez un point

x : **12**

y : **0**

ce point est dans la couronne

C:\JAVA\TP2>

**Exercice 2 : Classer 3 nombres**

Ecrire un programme TroisNombres.java qui lit 3 nombres au clavier, les classes dans l'ordre croissant et les affiche du plus petit au plus grand sur la console.

C:\JAVA\TP2>**java TroisNombres**

1er nombre : **14**

2ème nombre : **10**

3ème nombre : **17**

les nombres dans l'ordre croissant : 10 14 17

C:\JAVA\TP2>

**Thème 3 : itérations**

**Exercice 1 : Moyenne olympique**

Ecrire un programme qui lit au clavier une suite de nombres réels positifs ou nuls (correspondant à des notes), terminée par la valeur -1, et calcule la moyenne olympique de ces valeurs, c'est à dire la moyenne des notes sans prendre en compte la note la plus élevée ni la note la moins élevée.

Exemple de trace d'exécution (en vert les valeurs introduites par l'utilisateur):

C:\JAVA\TP3java>**MoyenneOlympique**

donnez une note ( >=0 ou -1 pour arrêter): **9.6**

donnez une note ( >=0 ou -1 pour arrêter): **9.7**

donnez une note ( >=0 ou -1 pour arrêter): **10.0**

donnez une note ( >=0 ou -1 pour arrêter): **9.8**

donnez une note ( >=0 ou -1 pour arrêter): **9.2**

donnez une note ( >=0 ou -1 pour arrêter): **9.9**

donnez une note ( >=0 ou -1 pour arrêter): **-1**

La note la plus élevée (10.0)et la note plus basse (9.2) ont été retirées

La moyenne olympique est : 9.75

C:\JAVA\TP3>

**Exercice 2 : Affichage de motifs - escaliers**

Ecrire un programme affiche un motif triangulaire dont la taille est fixée par une valeur lue au clavier.

Exemple de trace d'exécution (en vert les valeurs introduites par l'utilisateur):

C:\JAVA\TP3>**java Triangle1**

donnez taille du motif : **7**

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*\*

C:\JAVA\TP3>

a) écrire un programme Triangle1.java affichant ce motif en utilisant uniquement des instructions *tant que* (*while()*).

b) écrire un programme Triangle2.java affichant ce motif en utilisant uniquement des instructions *pour* (*for*).

**Exercice 3 : Affichage de motifs - pyramides**

Même exercice que le précédent mais le motif affiché n'est plus un triangle mais une pyramide (voir ci-dessous) et le choix des instructions pour le réaliser est laissé à votre jugement.

Exemple de trace d'exécution (en vert les valeurs introduites par l'utilisateur):

C:\JAVA\TP3>**java Pyramide**

donnez taille du motif : **7**

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*

C:\JAVA\TP3>

**Exercice 4 : Suite de Fibonacci**

La suite de Fibonacci est définie par la formule de récurrence suivante: u0 = 0 u1 = 1

un = un-1 + un-2 (n >= 3).

**a)** Ecrire un programme Fibo1.java qui permet de calculer le nième terme de la suite de Fibonacci, n étant fixé par l'utilisateur.

**b)** Ecrire un programme Fibo2.java qui permet d'obtenir la valeur et le rang du premier terme de cette suite supérieure à une valeur donnée par l'utilisateur.

**Exercice 5 : Tester si un nombre est premier**

Un nombre est n premier si il a seulement deux diviseurs : 1 et n.

Ecrire un programme Premier.java qui permet de tester si un nombre introduit par l'utilisateur est premier ou non.

Exemple de trace d'exécution (en vert les valeurs introduites par l'utilisateur):

C:\JAVA\TP2>**java Premier**

donnez un entier positif : **7**

7 est un nombre premier

Voulez-vous essayer un autre nombre O/N ? **o**

donnez un entier positif : **25**

25 n'est pas un nombre premier, il est divisible par 5

Voulez-vous essayer un autre nombre O/N ? **n**

Au revoir

C:\JAVA\TP2>

**Exercice 6 : Prix d'un lot**

**a)** Le jeu consiste à découvrir par essais successifs le prix d'un lot . Pour chaque essai, le joueur reçoit un message : "Trop grand", "Trop petit" ou "BRAVO ! Vous avez trouvé en *K* essais". Le jeu est fini quand le joueur a trouvé le prix du lot.

On propose d'écrire un programme JAVA PrixLot1.java qui joue le rôle de meneur de jeu ; l'exécution de ce programme vous fera tenir le rôle du joueur. Le programme Prixlot1doit définir le prix du lot en tirant un entier aléatoire entre 1 et 1000 et dialoguer avec le joueur pendant le jeu.

Pour choisir un nombre au hasard on utilisera la méthode random de la classe Math qui retourne un réel (double) tiré au hasard et de manière uniforme dans l'intervalle [0 1].

exemple :

double x; x = Math.random();

**exemple d'exécution du programme** PrixLot1

C:\JAVA\TP3>**java PrixLot1**

Le but est de chercher un prix entre 0 et 1000

Tapez un prix : **566**

Trop grand

Tapez un prix : **400**

Trop grand

Tapez un prix : **150**

Trop petit

Tapez un prix : **200**

Trop petit

Tapez un prix : **214**

BRAVO ! Vous avez gagné en 5 essais

C:\JAVA\TP3>

**b)** Copier dans le fichier PrixLot2.java le fichier PrixLot1.java, et modifier PrixLot2.java de manière à pouvoir :

* enchaîner plusieurs jeux consécutifs lors d'une même exécution du programme (à la fin de chaque jeu, il est demandé au joueur de préciser s'il veut s'arrêter ou rejouer)
* limiter le nombre d'essais du joueur lors d'une partie (au début de chaque partie, le programme demandera le nombre maximum d'essais autorisés).

Exemple d'exécution du porgramme PrixLot2

C:\JAVA\TP3>**java PrixLot2**

Le but est de chercher un prix entre 0 et 1000

1ère partie. Nombre maximum d'essais : 4

Tapez un prix : **678**

Trop petit

Tapez un prix : **920**

Trop grand

Tapez un prix : **860**

Trop petit

Tapez un prix : **910**

Trop petit

PERDU ! Vous avez épuisé le nombre d'essais autorisés

Le prix était : 917

Voulez-vous rejouer O/N ? **o**

2ème partie. Nombre maximm d'essais : **8**

Tapez un prix : **678**

Trop grand

Tapez un prix : **333**

Trop grand

Tapez un prix : **300**

Trop petit

Tapez un prix : **320**

Trop petit

Tapez un prix : **324**

BRAVO !

Vous avez gagné en 5 essais

Voulez-vous rejouer O/N ? **n**

C:\JAVA\TP3>

**c)** En jouant avec le programme précédent, essayer de trouver une stratégie systématique et efficace. Ecrire un programme PrixLot3.java qui fait jouer l'ordinateur à votre place suivant la tactique que vous venez de définir ; les deux rôles de meneur de jeu et de joueur sont donc à la charge du programme, le joueur "ignorant" évidemment le prix connu du meneur de jeu, mais "sachant" si son essai est plus grand ou plus petit que le prix.

Le programme PrixLot3 effectue 20 parties ; il affiche pour chaque partie le prix tiré au sort et le nombre d'essais successifs qui ont été nécessaires pour le trouver avec la tactique programmée.

Pouvez-vous prévoir, pour la tactique que vous avez choisie, l'ordre de grandeur des nombres d'essais ? Compléter le programme PrixLot3 en lui faisant calculer la moyenne des 20 nombres d'essais obtenus pour les 20 parties.

**Exercice 7 : Calcul de xn**

**a)** Ecrire un programme JAVA Puissance1.java permettant de calculer et d'afficher la valeur de xn, où x et n sont respectivement un réel et un entier introduits au clavier.

Pour effectuer ce calcul on ne fera pas appel aux méthodes de la classe Math (pow ou exp et log), mais on utilisera des instructions itératives (for, while ou do while). Le programme devra permettre de traiter tous les cas possibles : n = 0, n > 0 et n < 0 avec X réel négatif, positif ou nul. Rappelons que :

* X0 = 1 pour X > 0,
* 00 = 1 par convention,
* X0 n'est pas défini pour X < 0,
* 0n = 0 pour n > 0,
* 0n n'est pas défini pour n < 0.

On pourra organiser le calcul de la façon suivante :

**b)** à partir du programme Puissance1.java précédent écrire un programme Puissance2.java qui après chaque calcul de puissance demande à l'utilisateur si il veut faire un autre calcul ou non.

exemple d'exécution :

C:\JAVA\TP2>java Puissance2

x = **2.0**

n = **3.0**

xn = 8.0

voulez-vous faire un autre calcul O/N : **o**

x = **3.5**

n = **-3**

xn = 0.0233236151

voulez-vous faire un autre calcul O/N : **n**

C:\JAVA\TP2>

**Exercice 8 : Valeur approchée de la racine carrée d'un nombre réel positif**

On considère un nombre réel positif A ; on sait que la suite (un) n=0,1,2,... définie par la donnée d'un réel u0 positif et par la relation de récurrence un = (un-1 + A / un-1) \* 0,5 (pour n > 0) converge vers la racine carrée de A . On suppose le nombre A compris entre 1 et 100, et on prend u0 = A / 2.

Pour obtenir une valeur approchée de racine carrée de A, on cherche le premier terme un tel que | un2 - A | < E-5 . Le nombre trouvé est une valeur approchée de racine carrée de A ( en effet | un2 - A | < E-5 implique que | un- rac(A) | < E-5 / ( un+ rac(A) où rac(A) correspond à la racine carrée de A).

**a)** Ecrire un programme JAVA Raca1.java qui permet

1. de lire le nombre A,
2. de calculer et d'afficher les approximations intermédiaires et la valeur approchée de la racine carrée de A définie ci-dessus.

Exemple de l'état de l'écran obtenu par exécution du programme Raca1 :

Entrer un nombre A entre 1 et 100: **19.23**

Approximations successives :

u0 = 9.615

u1 = 5.8075

u2 = 4.559367735686612

u3 = 4.388528900180239

u4 = 4.385203650605606

Valeur approchée de la racine carrée  = 4.385202389856321

***Indications pour écrire le programme :***

* définir une constante : static final double EPS = 1E-5;
* utiliser une boucle *tant que* :
* while (. . . . . . . >= EPS) {

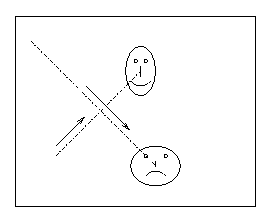
}

**b)** à partir du programme Raca1.java écrire un programme Raca2.java qui vérifie que le nombre introduit au clavier est bien un nombre positif supérieur à 1, et dans le cas contraire affiche un message d'erreur et redemande une nouvelle valeur à l'utilisateur.

**Thème 4 : instanciation d'objets - envois de messages**

Pour en savoir plus sur [comment créer puis utiliser des objets en java](https://lig-membres.imag.fr/genoud/teaching/PL2AI/tds/PL2/sujets/tp01_jdk/babajava.html)

1. En vous inspirant de l'application [AppliVisage1](https://lig-membres.imag.fr/genoud/teaching/PL2AI/tds/PL2/sujets/tp01_jdk/Visages/src/AppliVisage2.java) vue lors du TD n°1, écrivez une seconde application permettant d'animer simultanément deux visages. Le premier ayant le même mouvement que celui défini dans l'application Applivisage1, le deuxième partant du centre de l'écran, ayant une taille différente du premier et se déplaçant sur la seconde diagonale de la fenêtre.



1. Ecrivez une troisième application donnant aux visages qui se déplacent dans la fenêtre graphique une trajectoire moins répétitive. Au lieu de se déplacer uniquement selon une diagonale, les visages se déplacent  selon les quatre directions Nord-Ouest -> Sud-Est, Sud-Ouest -> Nord-Est, Sud-Est -> Nord-Ouest et Nord-Est -> Sud-Ouest. La direction du déplacement d'un visage sera modifiée à chaque rencontre de ce dernier  avec un bord de la fenêtre. Si vous avez le temps vous pouvez aussi animer  les visages en les dotant d'un changement d'expression à chaque changement de direction.

