UNIVERSITÉ TOULOUSE CAPITOLE

FACULTÉ D'INFORMATIQUE

Algorithmique et Programmation

Niveau débutant

Nous vous proposons une remise à niveau sous forme de défis à relever.

Les exercices sont regroupés par compétences requises pour les réaliser. Cela peut vous aider en vous donnant des indications sur ce qui est attendu de vous.

Les paragraphes en bleu sont là pour vous aider à réaliser certains éléments. Ils sont présents avant les exercices qui en ont besoin. Les parties qui concernent du code sont en italique.

Tri :
https://interstices.info/les-algorithmes-de-tri/

Exercices faciles :
https://www.tresfacile.net/exercices-corriges-en-java-les-bases-java-pour-debutant/

Copiez-collez le code qui vous sera donné dans un IDE online comme https://www.jdoodle.com/online-java-compiler . Exécutez le, vous devriez voir apparaitre le résultat dans la console. Vous ferez les exercices dans la fonction main.
Sauvegardez régulièrement votre travail dans un fichier texte a côté : jdoodle ne garde pas trace de votre travail.

Manipulation de variable et entrée utilisateur

De la même manière qu'en mathématique avec les variables x,y, etc, les programmes stockent des valeurs dans des conteneurs que l'on appelle **variable**. On peut comparer une variable à une boîte, dans laquelle on choisit de stocker une valeur en particulier. (le chiffre 4, le mot "bonjour", etc).
En java, il faut préciser le type de variable que l'on va utiliser. (la forme de la boite, qui ne pourra contenir que des éléments de la même forme - du même type)
La syntaxe est la suivante

Type_de_variable nom_de_variable = valeur_de_variable;

par exemple, si on veut travailler avec un entier qui va symboliser l'âge de quelqu'un, et qui aura une valeur de 23, on écrira :

int age = 23;

Pour être précis, deux étapes ont été réalisées en même temps. La **déclaration** de l'existence de la variable

int age;

puis une **affectation** d'une valeur pour cette variable

age = 23;

Ces étapes peuvent être faites à des moments différents. Il est donc possible de modifier la valeur contenue dans une variable plus tard dans le programme. On change la chose qu'on avait mise dans la boite. Le programme se souvient que l'on a déclaré qu'on utiliserait que des entiers dans la variable**age**.

Les types (de boite) les plus utilisées sont les suivants

Type de variable	Nom en francais	Exemple
int	Entier	4, -18, 1000 …
double	réel	0.21, 5, -10000, 45.70, …
String	chaine de caractères	Bonjour, au revoir, etc, …

Affichage dans console

Dans les environnements de programmation, la console permet de communiquer vers le programme et depuis le programme (périphérique d'entrée et périphérique de sortie). Il s'agit d'une zone de texte

généralement située en bas de l'écran de l'IDE.

pour afficher des éléments dans la console, il faut faire appel à une fonctionnalité qui s'écrit en java de la manière suivante

```
System.out.println( "mon texte");
```

le ***system.out.println()*** peut être utilisé de deux manières :

- Afficher directement une chaîne de caractère ou un chiffre
- Afficher la valeur contenue dans une variable (ce qui est contenu dans la boite)

```
System.out.println(12);
System.out.println(3+5);
String toto = "super" ;
System.out.println(toto);
```

Qu'affiche les lignes suivantes dans la console?

Lecture depuis la console

De manière équivalente, il est possible de récupérer une valeur d'un utilisateur pour l'utiliser dans un programme. Nous allons pour le moment éluder les détails du fonctionnement des scanners.
(Pour aller plus loin sur les scanners, https://www.w3schools.com/java/java_user_input.asp)
Pour le moment, trois fonctions sont disponibles pour utilisation dans les sources du programme disponible sur moodle.

Pourquoi faire la différence entre un entier et un double puisque les entiers sont compris dans l'ensemble des doubles? Pour des raisons d'arrondis,
int i = 100/10;
et
double i = 100.0/10.0;
n'auront pas nécessairement la même valeur.

Exercices d'application sur les entrées, sorties, déclaration de manipulation de variables

1) Déclarez une variable de type **String** et de nom ***unMot***, placez y le mot "**bonjour**". L'afficher. Remplacer la valeur contenue par la variable ***unMot*** par au "**revoir"**, l'afficher.

```
import java.util.Scanner;

public class MyClass
{
    /* Fonction qui retourne un entier saisi par l'utilisateur */
    public static int entreeUtilisateurEntier ()
        {
        Scanner sc = new Scanner(System.in);
        int nb = sc.nextInt();
        return nb;
        }

    /* Programme de démarrage */
    public static void main(String args[])
        {
        int n = entreeUtilisateurEntier();

        System.out.println("-> "+ n);
        }
}
```

2) Demandez à l'utilisateur son année de naissance en utilisant la fonction ***entreeUtilisateurEntier***() fourni dans le fichier source sur votre espace de travail. Affichez en résultat son année de naissance et son âge. Modifiez votre programme pour afficher la phrase suivante :

   "Vous avez [x] années"

   x étant l'âge calculé.

Nous verrons plus en détail l'appel de fonction ultérieurement. Pour l'instant, nous nous contenterons d'écrire la ligne suivante

```
int anneeNaissance = entreeUtilisateurEntier();
```

```
import java.util.Scanner;

public class MyClass
{
    /* Retourne un entier saisi par l'utilisateur */


    /* Retourne la somme de 2 entiers */
    public static int sommeInt (int a, int b)
        {
        return a + b;
        }


    /* Programme de démarrage (un seul par fichier) */
    public static void main (String args[])
        {
        // Saisir 2 entiers.
        int nb1 = saisirEntier("Saisir un entier nb1 : ");
        System.out.println("nb1 -> " + nb1);

        int nb2 = saisirEntier("Saisir un entier nb2 : ");
        System.out.println("nb2 -> " + nb2);

        // Faire la somme.
        int somme = sommeInt (nb1, nb2);

        // Afficher le résultat.
        System.out.println("Somme de " + nb1 + " + " + nb2 + " = " + somme);
        }

} // Fin de la classe.
```

Séquence conditionnelle et opérateur logique

Les séquences conditionnelles permettent de réaliser des instructions en fonction d'une condition. Une condition peut être composée de plusieurs parties, connectées entre elles par des connecteurs logiques, tels que ET(&&), OU(||) ou la négation (!).

Séquence conditionnelle

IF

```
int i = 4;
if(i > 5)
{
   System.out.println("I est bien supérieur a 5");
   String top = "ok";
}
```

```
System.out.println("au revoir");
```

L'exemple ci-dessus montre la syntaxe d'une condition. Si l'élément entre parenthèses est vrai, la série d'instruction entre accolade sera réalisée. On dit que la valeur de vérité de la condition est évaluée. Autrement dit, le programme vérifie si la condition est VRAI ou FAUSSE.
Dans l'exemple au dessus, quel sera le résultat dans la console si on affecte la valeur 6 à la variable i?

3) Demandez à l'utilisateur de rentrer un chiffre. Si ce chiffre est supérieur à 10, afficher la phrase de votre choix. Si le chiffre est inférieur à 10, affichez "non".

IF ELSE

Vous avez possiblement utilisé deux "if" pour l'exercice précédent. La possibilité d'associer des instructions dans le cas ou la condition est fausse a été prévue :

```
int i = 4;
if(i > 5)
{
   System.out.println("I est bien supérieur a 5");
}
else
{
   String reponse = "le chiffre n'est pas supérieur a 5";
   System.out.println(reponse);
}
```

la clause ***else*** permet d'associer des instructions au cas où la condition n'est pas vraie.

4) Si ce n'est pas déjà le cas, réécrivez l'exercice 3 avec une séquence **if, else**.

IF, ELSE IF, ELSE

```
if(i > 5)
{
   System.out.println("I est bien supérieur a 5");
}
else if(j>10)
{
   String reponse = "le chiffre n'est pas supérieur a 5";
   System.out.println(reponse);
}
else
{
     System.out.println("i  n'est  pas  supérieur  a  5  et  j  n'est  pas  supérieur  a
10");
}
```

if (i>5)

Ici, on regarde si i est supérieur à 5 avec le .
else if (j>10)
si ce n'est pas le cas, on va regarder si le j est supérieur à 10.
else
et si cette condition n'est pas vrai non plus, on rentre dans le dernier else.
Il est possible d'enchaîner autant de **else** et **else if** qu'on le souhaite en partant du même **if**.

5) Depuis l'exemple ci-dessus, donnez les différentes valeurs de i et de j qui permettent d'avoir les trois cas d'affichages différents.

6) Demandez à l'utilisateur sa note pour la matière anglais, et sa note pour la matière francais littéraire.
    a) Si les deux notes sont supérieures à 10, affichez "Année réussie".
    b) Si l'une des notes est entre 8 et 10, mais que la moyenne est supérieure à 10, affichez "Année réussie avec compensation".
    c) Et enfin, si les deux notes sont inférieures à 10, affichez "redoublement".

Séquence conditionnelle imbriquée

Il est tout à fait possible d'utiliser un if à l'intérieur d'un autre if. On dit alors qu'ils sont imbriqués.

```
if(i > 5)
{
   System.out.println("I est bien supérieur a 5");
   if(j>10)
   {
       String reponse = "le chiffre n'est pas supérieur a 5";
       System.out.println(reponse);
   }
}
```

Attention, les résultats ne sont pas les mêmes qu'avant, ici les deux sorties peuvent être obtenues dans la console.

7) réécrire l'exercice précédent en faisant usage d'imbrication.

Connecteurs logiques

Il est possible de complexifier les conditions qui sont entre parenthèses, en faisant l'usage de connecteurs logiques. Les plus simples sont le ET et le OU.

Il pleut	Il y a des nuages	Il pleut ET il y a des Nuages	Il pleut OU il y a des N.
VRAI	VRAI		
VRAI	FAUX		
FAUX	VRAI		
FAUX	FAUX		

On peut mettre autant de connecteurs logiques que l'on souhaite a la suite pour composer une meme condition. Ces connecteurs obéissent à des règles de priorités, comparable à celle des mathématiques, ou la multiplication est à effectuer avant l'addition. En informatique, le connecteur ET est prioritaire au connecteur OU.

Faux ET faux OU vrai

aura comme valeur de vérité "vrai".

Si vous souhaitez que vos conditions soient évaluées dans un ordre différent, vous pouvez utiliser des parenthèses pour rendre l'évaluation entre parenthèses prioritaire.

Faux ET (faux OU vrai)

aura comme valeur de vérité "faux".

8) Définissez une variable ***nombre*** à laquelle vous donnez la valeur de votre choix.
   a) Affichez "pair" si le nombre est pair, "impair" sinon.
   b) Améliorez votre programme pour afficher "multiple de 4" si c'est le cas, et multiple de 6 sinon.
   c) SI ce n'est pas déjà fait, faites en sorte que votre programme ne cherche pas à tester si le nombre est multiple de 4, si vous avez déjà trouvé qu'il n'était pas multiple de 2.

9) Demandez à l'utilisateur de rentrer un nombre, puis sans utiliser de fonction fourni par Java,
   a) Calculez sa valeur absolu
   b) Arrondissez ce nombre à une valeur entière

Approfondissements

Nous allons voir plus en détails certains éléments. En ce qui concerne les opérateurs de comparaison, il faut noter la différence entre strictement **supérieur** (>) ou supérieur ou égal (>=).

10)

Il existe un opérateur particulier, et très utile, le modulo (%). Pour faire simple, le modulo permet de compter par paquet de taille de votre choix. x%4 signifie que vous comptez x par paquet de 4, et qu'une fois un paquet complet vous recommencez à zéro.
1%4 sera égal a 1. 5%4 aussi. Dans le 5, pour pouvez retirer un paquet de 4, il restera 1.
10%4, vous pouvez enlever 2 paquet de 4, il restera 2.
Attention, en informatique on commence a compter à 0.

L'avantage du modulo est de pouvoir faire quelque chose de cyclique depuis un entier qui augmente de façon linéaire. Vous faites augmenter x de 1 en 1, mais pouvez subdiviser. Nous reverrons cet usage ultérieurement.
Pour l'instant, le modulo va nous servir à savoir si un nombre est pair ou impair.

11) quelle condition écrire avec le modulo pour s'assurer qu'un nombre est pair?
12) Utiliser le modulo pour convertir un nombre d'heures en nombre de jours et heures restantes. Par exemple, 26h devra renvoyer 1 jour et 2h, 49 h devra renvoyer 2 jours et 1h, etc.

Séquences de répétition

Les deux séquences de répétition les plus communes sont la boucle FOR et la boucle WHILE. Globalement, on utilise la boucle FOR lorsque l'on sait à l'avance le nombre de répétitions que l'on va réaliser, et la boucle WHILE lorsque la condition d'arrêt doit être réévaluée à chaque fois.
Pour aller plus loin, vous pouvez regarder la différence entre la boucle WHILE et la boucle DO…WHILE.

13) Affichez l'ensemble des multiples de 3 compris entre 1 et 1000
14) Calculez la partie entière de la racine carrée d'un nombre. Par exemple, la partie entière de racine de 10 est 3.
15) En utilisant la séquence de répétition WHILE, cherchez dans un tableau d'entier une valeur précise. Le while doit avoir deux conditions d'arrêt.
    a) Soit vous avez trouvé la valeur que vous cherchiez
    b) Soit vous avez parcouru l'ensemble du tableau, et que la valeur de s'y trouve pas.

    Essayez votre code avec un tableau contenant 10 valeurs.
    Pour déclarer un tableau en y mettant directement des valeurs, vous pouvez par exemple ecrire

int[] tab = new int[]{2,4,5,0};

16) Écrire un programme qui demande un nombre à l'utilisateur, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer : 1 + 2 + 3 + 4 + 5 = 15. Réalisez ce programme en n'affichant que le résultat final, puis une seconde version qui affiche la décomposition du calcul.

https://codegym.cc/fr/quests/lectures/fr.questsyntax.level10.lecture03

Réunir plusieurs données en java dans une seule variable

Plusieurs structures de données existent en Java, en fonction de l'utilisation que l'on souhaite en faire. Pour le moment, nous nous contenterons d'utiliser des tableaux, qui permettent de réunir ensemble des valeurs du même type.
Comme vous l'avez (re)vu hier pendant le test, la taille d'un tableau doit être connue avant son utilisation. Il faut être vigilant à différencier les valeurs contenues par le tableau, qu'il ne faut pas confondre avec l'index de ces valeurs, c'est-à dire l'indice ou se trouve la variable dans le tableau. L'indexation commence à la place 0. La taille d'un tableau tab s'obtient par

tab.length

son dernier indice utilisable sera donc

tab.length - 1

Soit un tableau de 4 entier, à savoir 4, 50, -4, 120, alors leur indices seront :

Indice	0	1	2	3
Valeur	4	50	-4	120

tab.length retournera 4.

17) Ecrire un algorithme qui permet de :

- Demander 5 fois à l'utilisateur de rentrer une note
- Mettre chaque note dans une case d'un tableau (de ?)
- Afficher :
  - La note minimum et la note maximum,
  - La moyenne,
  - Le nombre de note au-dessus de 10,
  - Le tableau de notes.
- Trier le tableau par ordre croissant (en utilisant l'algorithme tri-bulle) et l'afficher.

  Le principe de l'algorithme tri-bulle est d'échanger de proche en proche les valeurs du tableau pour obtenir un tableau trié.

  § Version 1 : parcourir tout le tableau et échanger les valeurs du tableau aux indices i et i+1 si et seulement la valeur à i est plus grande que celle à i+1. Faire ceci jusqu'à ce qu'aucune permutation n'est été faite durant tout un parcours du tableau.

  Version 2 : On constate qu'après un premier passage le dernier élément est le plus grand du tableau. Après un second passage, l'avant-dernier est le second plus grand… Donc on peut améliorer l'algorithme précédent en ne parcourant pas tout le tableau : au premier passage on va jusqu'au dernier, au second passage jusqu'à l'avant dernier …

18) Définissez un tableau de 10 entiers au début de votre programme.
    a) Triez ce tableau en ordre croissant,
    b) Améliorez votre programme pour pouvoir facilement trier en ordre décroissant.

19) Trouvez l'ensemble des nombres premiers compris entre 0 et 1000. Vous pouvez vous inspirer du crible d'Eratosthène.

20) Un des plus anciens systèmes de cryptographie (aisément déchiffrable) consiste à décaler les lettres d'un message pour le rendre illisible. Ainsi, les A deviennent des B, les B des C, etc.

    • Écrivez un algorithme qui demande une phrase à l'utilisateur et qui la code selon ce principe (transformez la chaîne récupérée en tableau de char et transformez ce tableau avant de l'afficher).

    • Une amélioration (relative) du principe précédent consiste à opérer avec un décalage non de 1, mais d'un nombre quelconque de lettres. Ainsi, par exemple, si l'on choisit un décalage de 12, les A deviennent des M, les B des N, etc. Réalisez un algorithme sur le même principe que le précédent, mais qui demande en plus quel est le décalage à utiliser. Le décalage sera effectué dans une fonction qui prendra en paramètre le tableau de char et la clé de décalage et renverra le nouveau tableau de char.

    NB : l'alphabet pourra être représenté comme un tableau de char que vous pourrez utiliser pour le codage.

21) Un système de cryptographie beaucoup plus difficile à briser que les précédents fut inventé au XVIe siècle par le français Vigenère. Il consistait en une combinaison de différents chiffres de César.
    On peut en effet écrire 25 alphabets décalés par rapport à l'alphabet normal :
    • l'alphabet qui commence par B et finit par …YZA
    • l'alphabet qui commence par C et finit par …ZAB
    • etc.
    Le codage va s'effectuer sur le principe du chiffre de César : on remplace la lettre d'origine par la lettre occupant la même place dans l'alphabet décalé. Mais à la différence du chiffre de César, un même message va utiliser non un, mais plusieurs alphabets décalés. Pour savoir quels alphabets doivent être utilisés, et dans quel ordre, on utilise une clé. Si cette clé est "VIGENERE" et le message "Il faut coder cette phrase", on procèdera comme suit :
    La première lettre du message, I, est la 9e lettre de l'alphabet normal. Elle doit être codée en utilisant l'alphabet commençant par la première lettre de la clé, V. Dans cet alphabet, la 9e lettre est le D. I devient donc D. La deuxième lettre du message, L, est la 12e lettre de l'alphabet normal. Elle doit être codée en utilisant l'alphabet commençant par la deuxième lettre de la clé, I. Dans cet alphabet, la 12e lettre est le S. L devient donc S, etc. Quand on arrive à la dernière lettre de la clé, on recommence à la première.
    • Ecrire l'algorithme qui effectue un cryptage de Vigenère, en demandant bien sûr au départ la clé à l'utilisateur.

Les fonctions

Les fonctions sont des éléments de votre programme qui permettront de réaliser des instructions si vous y faites appel.
Une fonction est définie par un nom, des paramètres d'entrées, et un type de retour. Pour l'instant, nous allons considérer que la seule manière de communiquer des valeurs avec la fonction se fait uniquement par le passage de paramètre.
Lorsque vous spécifiez des variables en tant que paramètres de la fonction, c'est la valeur de ces variables qui est en réalité donnée à la fonction.

22) Utilisation de fonction

Nous allons créer des fonctions permettant la manipulation de tableau. Nous utiliserons ces fonctions par la suite.

a) Dans votre programme principal (main), déclarez un tableau d'entier de la taille de votre choix, avec des valeurs numériques dedans. C'est depuis ce programme qui vous testerez les fonctions que vous allez écrire.

b) Créer une fonction **affichageTableau**, qui prend en paramètre un tableau d'entier, et qui ne renvoie pas de variable en retour. (Type de retour : **void**). Cette fonction va afficher tous les éléments d'un tableau. Faites en sorte que tous les éléments soient écrit sur une meme ligne. Testez votre fonction depuis le main.

c) Recopier la fonction de recherche que vous aviez faites hier, qui renvoie l'existence ou non d'un entier dans un tableau (Type de retour : boolean). Testez votre fonction depuis le main.

d) Créez une fonction d'insertion de valeur dans un tableau. Cette fonction prend en paramètre un tableau ainsi qu'une valeur a y ajouter. Le type de retour de cette fonction est un tableau d'entiers. Dans un premier temps, la fonction va regarder si la valeur existe déjà dans le tableau en paramètre. Pour ce faire, utilisez la fonction créée en b).

   i) Si la valeur est déjà dans le tableau, la fonction renvoie le tableau sans modification.

   ii) Si la valeur n'est pas présente dans le tableau en paramètre, il va falloir :

      1) Créer un nouveau tableau avec une taille supérieur à 1 par rapport au précédent
      2) Recopier les valeurs du tableau en paramètre
      3) Y ajouter la nouvelle valeur
      4) retourner ce tableau

   Tester votre fonction, en utilisant votre fonction d'affichage pour vérifier le résultat.

e) Ecrire une fonction tri, qui prend en paramètre un tableau d'entier, et qui retourne un nouveau tableau, trié en ordre croissant. Référez vous au site https://interstices.info/les-algorithmes-de-tri/ pour les différentes méthodes de tri.

```
package remise2024;

public class Remise2024
{
    // Sous-programme qui affiche un tableau (procédure).
    public static void afficheTableau1 (int[] t)
        {
        System.out.print("{ ");
        for (int i=0; i<t.length; i++)
            {
            System.out.print(t[i] + " ");
            }
        System.out.print("}");
        }

    public static void afficheTableau2 (int[] t)
        {
        String ch = "{ ";
        for (int i=0; i<t.length; i++)
```

```
        {
        ch = ch + t[i] + " ";
        }
    ch = ch + "}";

    System.out.println(ch);
    }

// Fonction qui retourne vrai si le nombre nb est présent dans le tableau.
public static boolean estInclus (int nb, int[] tab)
    {
    int indice = 0;
    while (indice < tab.length && tab[indice] != nb)
        indice++ ;

    return indice != tab.length ;
    }

// Fonction qui ajoute un nombre dans un tableau s'il n'est pas présent
public static int[] ajouteNombre (int nb, int[] tab)
    {
    if (estInclus(nb,tab))
        {
        // Si nb est inclus dans le tableau alors on retourne le tableau.
        return tab;
        }
    else
        {
        // Nb n'est pas inclus dans le tableau.
        // Création d'un nouveau tableau.
        int[] nouveau = new int[tab.length+1];

        // Copie des nombres/
        for (int i=0; i<tab.length; i++)
            nouveau[i] = tab[i];

        // Ajout du nombre dans la dernière case du tableau.
        nouveau[tab.length] = nb;

        return nouveau;
        }

    }


/**
 * Programme principal.
 */
public static void main(String[] args)
    {
    // Déclaration du tableau et une initialisation.
    int[] tab = {2, -1, 6, 7};

    afficheTableau1(tab);
    afficheTableau2(tab);

    System.out.println("Est ce que 3 appartient au tableau ? " + estInclus(3, tab));

    int a = -1;
   System.out.println("Est ce que " + a + " inclus dans tableau ? " + estInclus(a, tab));

    int b = 7;
    boolean resultat = estInclus(b, tab);
    System.out.println("Est ce que " + b + " inclus dans tableau ? " + resultat);

    System.out.println("-------");
    afficheTableau2(tab);

    int nbnew = 3;
    System.out.println("On ajoute " + nbnew);
    int[] tabnew = ajouteNombre(nbnew, tab);
    System.out.println("->");
```

```
        afficheTableau2(tab);
        afficheTableau2(tabnew);

        nbnew = 6;
        System.out.println("On ajoute " + nbnew);
        tabnew = ajouteNombre(nbnew, tab);
        afficheTableau2(tabnew);

        nbnew = 8;
        System.out.println("On ajoute " + nbnew);
        afficheTableau2(ajouteNombre(nbnew, tab));

        System.out.println("---->");
        afficheTableau2(tab);
        tab = ajouteNombre(10, tab);
        afficheTableau2(tab);

        System.out.println("========");

        int nb1  = 11;
        if (estInclus(nb1, tab))
            {
            System.out.println(nb1 + " est inclus dans le tableau");
            }
        else
            {
            System.out.println(nb1 + " n'est pas inclus dans le tableau");
            }

        System.out.println("- - - - - -");
        int nbb = 1;
        for (int i=0; i<5; i++)
            {
            nbb = i*5;
            }
        System.out.println(nbb);

        System.out.println("--- valeur absolu");
        int nb3 = 5;
        if (nb3 > 0)
                System.out.println(nb3);
        else
                System.out.println(-nb3);

        System.out.println("- - - - - -");
        int nb4 = 10;

        System.out.println("Racine de " + nb4 + " est " + Math.sqrt(nb4));

        int pe = (int)Math.sqrt(nb4);
        System.out.println(pe);
        }
}
```