Premiers pas en Algo - B

Objectifs du module

L'objectif de ce module est l'apprentissage des concepts communs à tous les langages informatiques.

Compétences

- valeur de retour d'une fonction
- conditions et opérateurs logiques
- les branchements et les boucles

Démarche pédagogique

Tous les exercices doivent être réalisés seul(e), en autonomie. Les livrables doivent être validés par le formateur avant de passer à l'exercice suivant.

Exercice B.1.1 - Billets SVP

Énoncé

Imaginez que vous deviez développer un logiciel de billetterie qui doit permettre, quand un client se présente à la caisse, de calculer le prix à payer en fonction du nombre de billets achetés.

C'est VOUS qui avez été chargé(e) d'écrire la fonction qui va faire le calcul.

Pour commencer, écrivez une **fonction** `calculePrix` qui prend un **paramètre** `nombreDeBillets` et qui affiche avec `println` le résultat du calcul.

Le prix d'un billet à l'unité est de 10,50 euros.

Livrables

- Une fonction `calculePrix` avec les bons **paramètres**
- Un programme qui affiche le prix à payer pour 4 personnes et pour 9 personnes (appelez deux fois la fonction)

Compétences

- Programmer avec des variables
- Programmer une fonction et l'appeler
- Passez des paramètres à une fonction

Note

Non, il n'y a pas de faute à `calculePrix`, car c'est le verbe et non un nom. Une fonction étant une action, il est conseillé de les faire commencer par un verbe pour exprimer cette action.

Ressources

- https://fr.flossmanuals.net/processing/les-methodes/

Exercice B.1.2 - Billets SVP

Énoncé

Dans l'exercice précédent, nous affichons directement le résultat dans la console mais comment faire si nous souhaitons, par exemple, l'utiliser dans calcul ?

Il faudrait que l'on puisse récupérer la valeur en dehors de la fonction...

C'est possible, en la modifiant de manière à **retourner** le prix calculé. Il faut pour cela remplacer le type de retour, précédemment `void`, qui veut dire "rien" par `float` qui veut dire "nombre à virgule" et utiliser entre les accolades de la fonction le mot clé `return` suivi de l'expression/la valeur à retourner (dans notre cas le résultat du calcul).

Cela permet de récupérer la valeur au moment de l'**appel** de la **fonction** (c'est-à-dire au moment où on écrit le nom de la fonction suivi des parenthèses) pour l'utiliser tout de suite ou plus tard.

```
// ici, définir la fonction calculePrix qui retourne un float
// ...

float total = calculePrix(4)+calculePrix(9);
println(total)
```

Livrables

- La fonction `calculePrix` modifiée pour retourner le résultat
- un programme qui affiche le total encaissé si un premier client achète 4 billets puis le suivant 9 billets.

Compétences

- Programmer avec des variables
- Programmer une fonction et l'appeler
- Passez des paramètres à une fonction

Exercice B.2 - Tarif groupe

Énoncé

Une nouvelle politique de prix entre en vigueur et vous devez faire évoluer la fonction `calculePrix`.
Le prix est toujours de 10,50 euros jusqu'à 5 personnes, mais à partir de 6 personnes, un tarif de groupe s'applique et le prix passe à 9 euros par personne (pour 6 personnes il faut donc payer 54 euros).
Pour calculer le bon prix, nous devons donc avoir deux formules et choisir celle à utiliser en fonction du nombre de personnes.

Comment faire ? Nous allons utiliser une **condition**, c'est-à-dire une **expression** qui peut prendre la valeur **vrai** ou **faux**.
Dans notre cas, l'expression que vous devez trouver doit correspondre à "est-ce que `nombreDeBillets` est plus grand que 5". Sachant que l'on utilise la même syntaxe qu'en math c'est à dire `>`, à vous de l'écrire en Java.

Livrables

- Le même programme que dans l'exercice précédent mais qui tient compte du prix de groupe

Ressources

- https://fr.flossmanuals.net/processing/les-conditions/
- https://www.tutorialspoint.com/java/if_else_statement_in_java.htm
- https://www.w3schools.com/java/java_booleans.asp
- https://www.w3schools.com/java/java_conditions.asp

Compétences

- Programmer avec des variables
- Programmer une fonction et l'appeler
- Passez des paramètres à une fonction
- **Programmer une condition**

Exercice B.3.1 - Conditions

Énoncé

Remplissez les case vides par vrai ou faux en évaluant les **expressions** suivantes avec les différentes valeurs de x et y :

en français	**en code**	x=3 , y=2	x=3 , y=3	x=2 , y=4
x supérieur à y	x > y			
x égal à y	x == y			
x supérieur ou égal à y	x >= y			
x inférieur à y	x < y			
x inférieur ou égal à y	x <= y			
x différent de y	x != y			
x moins y égal 0	x - y == 0			
x égal à y plus un	x == y + 1			
x moins y supérieur à 0	x - y > 0			
x fois deux égal y	x * 2 == y			
x plus x égal y	x + x == y			
x plus y égal à y plus x	x + y == y + x			
x supérieur à moins y	x > -y			

Exercice B.3.2 - Opérateurs logiques

Énoncé

Souvent, vous aurez besoin de combiner les valeurs de plusieurs conditions, par exemple “i > 10” **et** “i < 20”. Remplissez le tableau ci dessous :

`!` (pas / **not**) - inverse le résultat)i	
! true	
! false	

`||` (ou / **or**) - l’un des deux côté doit-être vrai	
true || false	
true || false	
false || false	

`&&` (et / **and**) - les deux côtés doivent être vrais	
true && true	
true && false	
false && false	

`!=` (different / **not equal**) - les deux côtés doivent être différents	
true != true	
true != false	
false != fase	

`==` (égale / **equal**) - les deux côtés doivent être égaux	
true == true	
true == false	
false == false	

Exercice B.4 - Go, Toto go!

Énoncé

Revenons à notre ami Toto et améliorons le programme avec ce que nous venons d'apprendre :

Écrire une fonction `isNearWater` qui retourne une valeur de **type** `boolean`. Elle doit retourner `true` si Toto est devant une flaque d'eau (en comparant `totoPosition` et avec la position des flaques) et sinon elle doit retourner `false`.

Écrire une fonction `go` qui appelle `isNearWater` et, en fonction de la valeur retournée, appelle ensuite soit `walk`, soit `jump`.

Livrables

- Fonctions `isNearWater` et `go`
- Un qui fait arriver Toto au bout du chemin en appelant la fonction `go` plusieurs fois.

Compétences

- ...
- Programmer une condition

Exercice B.5 - Tant que…

Énoncé

Il existe une variante du `if` appelée `while`, qui exécute le code entre les accolades en **boucle** tant que la **condition** est vraie.

Modifiez votre programme, pour faire avancer Toto automatiquement en appelant la fonction `go` tant que `totoPosition` est inférieur à 7.

Livrables

- Un programme qui permet à Toto d'avancer jusqu'au bout du chemin et s'arrêter, mais l'appel à la méthode `go` se fait automatiquement dans une boucle `while`.

Compétences

- ...
- Programmer une condition
- **Programmer une boucle**

Ressources

- https://processing.org/reference/while.html
- https://www.tutorialspoint.com/java/java_while_loop.htm
- https://www.w3schools.com/java/java_while_loop.asp