L03A\_SURCHARGE DE MÉTHODES

Table des matières

[1 Exercice 1 – Calcul d’aire géométrique 2](#_Toc97549079)

[2 Exercice 2 – Types de clients 3](#_Toc97549080)

# Exercice 1 – Calcul d’aire géométrique

On fait une méthode qui permet de calculer le périmètre pour : un carré, un rectangle ou un triangle. La méthode porte le même nom, mais accepte des paramètres différents et applique la bonne formule en fonction du nombre de paramètres.

Dans un projet **l03a-surcharge-1**, créer un programme en Java qui respecte les spécifications suivantes :

1 package **mathematique**

1 classe **Perimetre** qui contient

* 1 attribut **perimetreCalcule** de type **float**
* 1 méthode appelée **calculPerimetre** qui :
  + Accepte 1 paramètre de type **float** qui représente la longueur d’un côté d’un carré
  + Calcul le périmètre d’un carré (4 \* 1 côté)
  + Enregistre la valeur du périmètre dans le champ **perimetreCalcule**
  + Affiche à la console le message : Le périmètre du carré est **x**
    - Où x est le périmètre calculé
* 1 méthode appelée **calculPerimetre** (même nom) qui :
  + Accepte 2 paramètres de type **float** qui représentent les longueurs du grand côté et du petit côté d’un rectangle
  + Calcul le périmètre d’un rectangle (2 \* (grand côté + petit côté))
  + Enregistre la valeur du périmètre dans le champ perimetreCalcule
  + Affiche à la console le message : Le périmètre du rectangle est **x**
    - Où x est le périmètre calculé
* 1 méthode appelée **calculPerimetre** (même nom) qui :
  + Accepte 3 paramètres de type **float** qui représentent la longueur de chaque côté d’un triangle
  + Calcul le périmètre d’un triangle (côté 1 + côté 2 + côté 3)
  + Enregistre la valeur du périmètre dans le champ perimetreCalcule
  + Affiche à la console le message : Le périmètre du triangle est **x**
    - Où x est le périmètre calculé

1 classe appelée **Geometrie** qui contient la méthode **main**, qui doit inclure:

* Création d’un objet basé sur la classe **Perimetre**
* Appel de chaque surcharge de la méthode avec des valeurs appropriées pour en tester le fonctionnement.

Gérez les modificateurs d’accès selon les meilleures pratiques.

# Exercice 2 – Types de clients

La classe de cet exercice permet de gérer les clients d’une application, en les classant en 3 types, présentée ci-dessous :

* Un **individu** n’a pas de numéro de taxes ou de numéro de compagnie
* Un **organisme sans but lucratif** n’a pas de numéro de compagnie, mais a un numéro de taxes
* Une **compagnie** a un numéro de compagnie **et** un numéro de taxes

Dans un projet **l03a-surcharge-2**, créer un programme en Java qui respecte les spécifications suivantes :

1 package **gestionnaireclient**

1 classe **Client** qui contient

* 1 attribut **typeClient** de type **String**
* 1 attribut **noTaxe** de type **String**
* 1 attribut **noCompagnie** de type **String**
* 1 méthode appelée **ajoutIndividu** qui :
  + N’accepte aucun paramètre
  + Enregistre la valeur **“Individu”** dans le champ **typeClient**
  + Enregistre **“”** (chaine vide) pour **noTaxe**
  + Enregistre **“”** (chaine vide) pour **noCompagnie**
  + Affiche à la console le message : Le client est un individu et n’a pas de numéro de taxes ou de numéro de compagnie.
* 1 méthode appelée **ajoutIndividu** qui :
  + Accepte 1 paramètre (String)
  + Enregistre la valeur **“OSBL”** dans le champ **typeClient**
  + Enregistre **la valeur du paramètre** pour **noTaxe**
  + Enregistre **“”** (chaine vide) pour **noCompagnie**
  + Affiche à la console le message : Le client est un OSBL et son numéro de taxes est **x**
    - Où x est la valeur du champ **noTaxe**
* 1 méthode appelée **ajoutIndividu** qui :
  + Accepte 2 paramètres (String)
  + Enregistre la valeur **“Compagnie”** dans le champ **typeClient**
  + Enregistre la valeur du premier paramètre pour **noTaxe**
  + Enregistre la valeur du deuxième paramètre pour **noCompagnie**
  + Affiche à la console le message : Le client est une compagnie, son numéro de taxes est **x** et son numéro de compagnie est **y**
    - Où **x** est la valeur du champ **noTaxe** et **y** est la valeur du champ **noCompagnie**

1 classe appelée **GestionClient** qui contient la méthode main, qui doit inclure:

* Création d’un objet basé sur la classe **Client**
* Appel de chaque surcharge de la méthode avec des valeurs appropriées pour en tester le fonctionnement.

Gérez les modificateurs d’accès selon les meilleures pratiques

**Question :**

La troisième version de la méthode prévoit la réception du noTax comme premier paramètre et le noCompagnie comme deuxième paramètre. Qu’arrivera-t-il si les valeurs de ces paramètres sont inversées dans l’appel de la méthode du programme principal (le noCompagnie est passé en premier)?

|  |
| --- |
| Rien, parce que le type passer en parametre est le même. |