

L'objectif de ce TP est de mettre en œuvre les premiers concepts et conseils relatifs à la notion de classe en java et à l'intérêt d'une classe proposant des services utilisés par d'autres classes d'un même projet.



RAPPEL : le fichier `UTILISATION_IJ` accessible à partir de ce parcours, peut vous être utile...

Avant de commencer...

- ✓ Lisez entièrement ce sujet
- ✓ Ouvrez un terminal et placez-vous dans votre répertoire `R1.01`
- ✓ Lancez IJ avec la commande `idea`, puis créez un nouveau projet `TP3_A` (cf. fichier `UTILISATION_IJ`)

1. Classe Utilitaire

Dans ce TP, la classe `Utilitaire` rassemblera les fonctions ou procédures nécessaires aux contrôles de saisie ainsi qu'à l'initialisation de figures géométriques et à l'affichage de leurs caractéristiques.

Dans le projet `TP3_A`, créez une classe java `Utilitaire` où vous devrez :

- Coller le code de la fonction `arrondi2` qui vous est donné ci-dessous

```
public static float arrondi2(float unFloat) {
    // {} => {résultat = valeur de unFloat après arrondi à 2 décimales}
    return (float) Math.round(100*unFloat)/100.0f;
}
```

- Coder la fonction suivante :

```
public static float saisieFloatPos() {
    // {} => {résultat = un réel (float) strictement positif}
}
```

Partie A : Autour du rectangle

A1. Création de la Classe Rectangle

Dans ce TP, un rectangle sera défini par la longueur de ses côtés, quelle que soit sa position dans le plan.

- 1.1.** Dans le projet `TP3_A`, créez une classe java `Rectangle` (cf. fichier `UTILISATION_IJ`)

Dans la classe `Rectangle` :

- Déclarez les **attributs** `cote1` (longueur d'un côté) et `cote2` (longueur de l'autre côté) d'un rectangle (type requis : `float`)
- Codez le **constructeur** d'un rectangle

```
public Rectangle(float cote1, float cote2) {
    ...
}
```

- Codez les **méthodes** suivantes :

- ✓ Consultation de la longueur d'un côté : méthode `getCote1()`
- ✓ Consultation de la longueur de l'autre côté : méthode `getCote2()`
- ✓ Calcul du périmètre : méthode `getPerimetre()` – résultat de type `float`
- ✓ Calcul de la surface : méthode `getSurface()` – résultat de type `float`

- Vérifiez que votre code est correct (pas de rouge partout) et, si besoin, corrigez-le avant de poursuivre

N'OUBLIEZ PAS DE COMMENTER
VOTRE CODE !

A2. Fonctions et procédures de saisie et affichage d'un rectangle

2.1. Dans la classe Utilitaire, ajoutez les fonctions suivantes :

- ✓ Saisie contrôlée d'un rectangle - **NOTE** : cette fonction doit utiliser `saisieFloatPos()`

```
public static Rectangle saisirRectangle() {  
    // {} => {résultat = un Rectangle}
```

- ✓ Affichage des caractéristiques d'un rectangle

```
public static void afficherRectangle(Rectangle unRectangle) {  
    // {} => {les longueurs des côtés de unRectangle ont été affichées  
    //         ainsi que les valeurs arrondies à 2 décimales de son  
    //         périmètre et de sa surface}
```

2.2. Vérifiez votre code avant de poursuivre

A3. Classe Geometrie

Cette classe sera complétée au fur et à mesure des besoins : elle contiendra la procédure principale permettant de tester le code développé dans les autres classes.

3.1. Dans le projet TP3_A, ajoutez une classe Geometrie, puis insérez dans son enveloppe une procédure `main`

3.2. Ajoutez les instructions permettant :

- ✓ l'initialisation d'un Rectangle par saisie
- ✓ l'affichage des caractéristiques de ce Rectangle

3.3. **Testez** : Vous vérifierez que la saisie des côtés d'un rectangle est bien contrôlée et que les caractéristiques de ce rectangle ont les valeurs attendues

cote1	cote2	périmètre	surface
-3		impossible (réel positif attendu)	
2	-12,5	impossible (réel positif attendu)	
2	7	18.0	14f.0
10,5	5,8	32.6	60.9
15,2578	20	70.52	305.16
0,05	100	200.1	5.0

LES VALEURS EN ITALIQUE SONT
SAISIES PAR L'UTILISATEUR

3.4. Corrigez si besoin les erreurs d'exécution (elles peuvent provenir d'erreurs que vous n'aviez pas identifiées dans les classes Rectangle ou Utilitaire).

Partie B : Autour du cercle

B.1 Classe Cercle

Dans ce TP, un cercle sera défini par la longueur de son rayon.

1.1. Dans le projet TP3_A, ajoutez une classe Cercle

1.2. Dans la classe Cercle :

- Déclarez une constante PI de type double et de valeur 3.14
- Déclarez l'attribut rayon (longueur du rayon) d'un cercle, de type float
- Codez le constructeur d'un cercle
- Codez les méthodes suivantes :
 - ✓ Consultation de la longueur du rayon : méthode getRayon ()
 - ✓ Consultation du périmètre : méthode getPerimetre() – résultat de type float
 - ✓ Consultation de la surface : méthode getSurface() – résultat de type float
- Vérifiez que votre code est correct et, si besoin, corrigez-le avant de poursuivre

N'OUBLIEZ PAS DE COMMENTER
VOTRE CODE !

B.2 Classes Utilitaire et Geometrie

2.1. Dans la classe Utilitaire, codez les fonctions suivantes :

a) Saisie contrôlée d'un cercle - **NOTE** : cette fonction doit utiliser saisieFloatPos()

```
public static Cercle saisirCercle() {  
    // {} => {résultat = un Cercle}}
```

b) Affichage des caractéristiques d'un cercle

```
public static void afficherCercle(Cercle unCercle) {  
    // {} => {le rayon du cercle unCercle a été affiché, ainsi que  
    //         les valeurs arrondies à 2 décimales de son périmètre  
    //         et de sa surface}}
```

2.2. Dans la procédure main de la classe Geometrie, ajoutez les instructions permettant la saisie d'un Cercle puis l'affichage de ses caractéristiques

2.3. Testez

rayon	périmètre	surface
-2	impossible (réel positif attendu)	
0	impossible (réel positif attendu)	
1	6.28	3.14
10	62.8	314.0
10,5	65.94	346.19
10,99	69.02	379.25

LES VALEURS EN ITALIQUE SONT
SAISIES PAR L'UTILISATEUR

Partie C : Autour du triangle

C1. Classe Triangle

Dans ce TP, un triangle sera défini par la longueur de chacun de ses côtés.


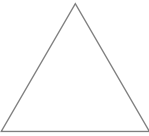
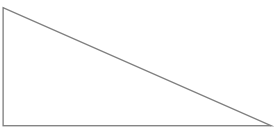
PROPRIÉTÉS D'UN TRIANGLE

- SURFACE D'UN TRIANGLE, CONNAISSANT LA LONGUEUR DE SES TROIS CÔTÉS

La surface S d'un triangle est calculable à partir des longueurs a , b et c de ses côtés, en appliquant la **formule de Héron** :

$$S = \sqrt{p(p-a)(p-b)(p-c)} \text{ avec } p = (a+b+c) / 2$$

- TRIANGLES PARTICULIERS

	TRIANGLE ISOCÈLE	TRIANGLE ÉQUILATÉRAL	TRIANGLE RECTANGLE
Exemple			
Caractérisation	Au moins deux côtés de même longueur	Les trois côtés ont la même longueur	La somme des carrés des longueurs de deux des côtés est égale au carré de la longueur du troisième côté

- INÉGALITÉ TRIANGULAIRE

Dans un triangle, la longueur d'un côté est inférieure à la somme des longueurs des autres côtés

1.1. Dans le projet TP3_A, créez une classe Triangle

1.2. Dans la classe Triangle :

- Déclarez les **attributs** `cote1` (longueur d'un côté), `cote2` (longueur d'un autre côté) et `cote3` (longueur du troisième côté) d'un triangle (type requis pour chacun des côtés : `float`)
 - Codez le **constructeur** d'un triangle
 - Codez les **méthodes** suivantes :
 - ✓ Consultation de la longueur de chaque côté : méthodes `getCote1()`, `getCote2()` et `getCote3()`
 - ✓ Consultation du périmètre : méthode `getPerimetre()` – résultat de type `float`
 - ✓ Consultation de la surface : méthode `getSurface()` – résultat de type `float`
 - ✓ Consultation de la nature (`équilateral`, `isocèle et rectangle`, `rectangle`, `isocèle` ou `quelconque`) : méthode `getNature()` – résultat de type `String`
- NOTE** : un triangle équilatéral ne peut pas être rectangle, alors qu'un triangle isocèle non équilatéral peut l'être

N'OUBLIEZ PAS DE
COMMENTER VOTRE CODE

- Vérifiez que votre code est correct et, si besoin, corrigez-le avant de poursuivre, puis testez

C2. Classes Utilitaire et Geometrie

2.1. Dans la classe Utilitaire, ajoutez les fonctions suivantes :

- a) Saisie contrôlée du troisième côté d'un triangle - **NOTE** : cette fonction doit utiliser `saisieFloatPos()`

```
public static float verifInegaliteTriangulaire(float cote1, float cote2) {  
    // {cote1 et cote2 > 0} => {le résultat est un float strictement positif,  
    //                               tel qu'un triangle dont les côtés seraient ce  
    //                               ce float, cote1 et cote2, puisse être construit}
```

- b) Saisie d'un Triangle - **NOTE** : cette fonction doit utiliser `saisieFloatPos()` pour les deux premiers côtés et `verifInegaliteTriangulaire()` pour le troisième côté

```
public static Triangle saisirTriangle() {
    // {} => {résultat = un Triangle}
```

- c) Affichage des caractéristiques d'un triangle

```
public static void afficherTriangle(Triangle unTriangle) {
    // {} => {les longueurs des côtés de unTriangle, sa nature
    //         et les valeurs arrondies à 2 décimales de son
    //         périmètre et de sa surface, ont été affichées}
```

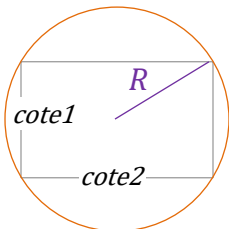
- 2.2. Dans la procédure `main` de la classe `Geometrie`, ajoutez les instructions permettant la saisie d'un Triangle puis l'affichage de ses caractéristiques

2.3. Testez

cote1	cote2	cote3	périmètre	surface	équilateral	isocèle	rectangle	quelconque
2	0		impossible (réel positif attendu)					
2	10	5	impossible (plus grand côté supérieur à la somme de 2 autres)					
2	10	11	23.0	9.05	✓			
3	4	5	12.0	6.0			✓	
5	4	4	13.0	7.81		✓		
5	5	5	15.0	10.83	✓			

Partie D : Compléments Rectangle : cercle inscrit et cercle circonscrit

CERCLE CIRCONSCRIT À UN RECTANGLE



$$\text{Rayon du cercle circonscrit : } R = \frac{\sqrt{cote1^2 + cote2^2}}{2}$$

1. Ajoutez dans la classe `Utilitaire` les fonctions suivantes :

- a) Cercle inscrit dans un rectangle – **NOTE** : un peu de bon sens suffit 😊

```
public static Cercle cercleInscrit(Rectangle unRectangle) {
    // {} => {résultat = le Cercle inscrit dans unRectangle}
```

- b) Cercle circonscrit à un rectangle

```
public static Cercle cercleCirconscrit(Rectangle unRectangle) {
    // {} => {résultat = le Cercle circonscrit à unRectangle}
```

2. Dans la procédure `main` de la classe `Geometrie`, ajoutez en fin de programme, les instructions permettant d'afficher les caractéristiques du cercle *inscrit* dans le rectangle (précédemment saisi par l'utilisateur) et celles du cercle *circonscrit* à ce rectangle.

3. Testez