Triangle orienté objet [ge03] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 22 mai 2018

Table des matières

1 Points du plan **2**
1.1 Classe Point 2
1.2 Test de la classe Point 4

2 Triangle orienté objet **5**
2.1 Classe Triangle 5
2.2 Test de la classe Triangle 6
2.3 Géométrie d'un triangle 6

3 Références générales **10**

Java - Triangle orienté objet (Solution)

Mots-Clés Géométrie ■
Requis Structures de base, Structures conditionnelles, Algorithmes paramétrés, Classes ■
Difficulté • ○ ○ (1 h 30) ■

Objectif
Cet exercice conçoit un programme orienté objet relatif à la géométrie d'un triangle.

...(énoncé page suivante)...

1 Points du plan

1.1 Classe Point

Ce problème réalise une classe `Point` modélisant des points du plan de $\mathbb{R}^2$.

Analyse

On notera que :

- Un objet de type `Point` a typiquement les coordonnées `x` et `y`.
- Les coordonnées des points peuvent être spécifiées lors de leur construction. Néanmoins il est utile de définir un constructeur par défaut car les points sont des éléments de base de la géométrie et l'on souhaitera souvent en instancier puis les saisir. Ceci signifie que la classe doit être muni de deux constructeurs :
  - Le constructeur par défaut qui crée le point de coordonnées $(0.0, 0.0)$.
  - Un constructeur à deux paramètres, à savoir l'abscisse et l'ordonnée.
- Une méthode `distance` peut être utile pour calculer la distance entre deux points. Si cette méthode se trouve dans la classe `Point`, le premier point est accessible sous forme de l'objet utilisé pour appeler la méthode, mais il faut envoyer le deuxième point en paramètre.

Écrivez une classe `Point` ayant pour attributs `x` (réel) et `y` (réel).

Écrivez un constructeur par défaut qui fixe le point en $(0.0, 0.0)$.

Écrivez un constructeur à deux paramètres de réels `x` et `y`.

Écrivez des accesseurs `getX à de l'abscisse et \lstinlinegetY` à de l'ordonnée du point.

Écrivez un mutateur `assign(x,y)` qui fixe les nouvelles coordonnées `(x,y)`.

Écrivez une méthode `afficher` qui affiche le point sous le format `(x,y)`.

Écrivez une méthode `distance(p)` qui calcule et renvoie la distance entre l'objet courant et un `Point p`. Rappel : La distance d entre deux points (x_1, y_1) et (x_2, y_2) est :

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Outil Java

L'opération $\sqrt{x}$ s'écrit `Math.sqrt(x)`.

Que peut-on dire de la méthode `distance` ?

Solution simple

Elle est symétrique :

```
p1.distance(p2) <==> p2.distance(p1)
```

Validez votre classe avec la solution.

Solution Java @[Point.java]

```
import java.lang.Math;
public class Point {
private double m_x, m_y;

public Point(){
  assign(0.0, 0.0);
}

public Point(double x, double y){
  assign(x, y);
}

public double getX(){
  return m_x;
}

public double getY(){
  return m_y;
}

public void assign(double x, double y){
  m_x = x;
  m_y = y;
}

public void afficher(){
  System.out.print("("+getX()+","+getY()+")");
}

public String str(){
  return "("+getX()+", "+getY()+")";
}

public double distance(Point p2){
  double dx = (getX()-p2.getX());
  double dy = (getY()-p2.getY());
  return Math.sqrt(dx*dx + dy*dy);
}

public void translater(double dx, double dy){
  m_x += dx;
  m_y += dy;
}
}
```

1.2 Test de la classe Point

Écrivez une procédure `saisirPoint(txt,p)| qui affiche une chaîne de caractères \lstinline`txt@ puis saisit un `Point` dans `p` sous la forme d'un couplet de réels `(x,y)`.

Déduisez une procédure `saisirPoints(p1,p2,p3)` qui saisit trois `Point`s dans `p1`, `p2` et `p3` en affichant le texte adéquat :

```
Premier point (x,y)?
Deuxième point (x,y)?
Troisième point (x,y)?
```

Écrivez une procédure `afficherPoints(p1,p2,p3)` qui affiche trois `Point| \lstinline`p1@, `p2` et `p3` (où `[x]` désigne le contenu de `x`) :

```
P1 = [p1], P2 = [p2], P3 = [p3]
```

Validez vos procédures avec la solution.

Écrivez une procédure `test_Points` qui instancie et saisit trois `Point`s puis qui les affichent.

Testez. Exemple d'exécution :

```
Premier point (x,y)? (-3.5,0)
Deuxieme point (x,y)? (3.5,0)
Troisieme point (x,y)? (0,4)
  P1 = (-3.5,0), P2 = (3.5,0), P3 = (0,4)
```

Validez votre procédure avec la solution.

2 Triangle orienté objet

2.1 Classe Triangle

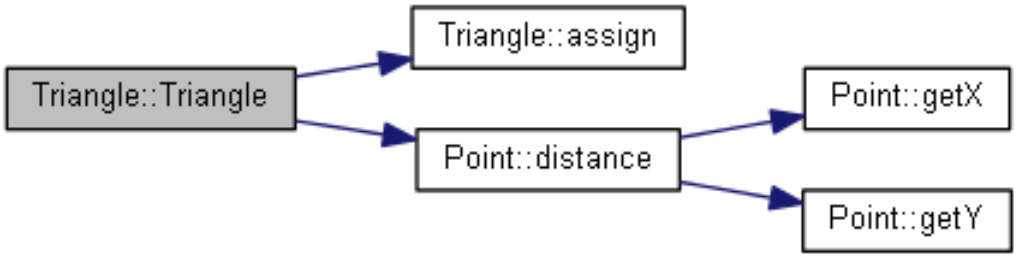
Propriété
Un **triangle** a trois côtés de longueurs $\ell 1$, $\ell 2$ et $\ell 2$.

Écrivez une classe `Triangle` munie des longueurs des trois côtés `l1`, `l2`, `l3` (réels).

Écrivez un mutateur `assign(a,b,c)` qui fixe les nouvelles longueurs `(a,b,c)` des côtés du triangle. Cette méthode doit fixer la longueur à 0 si le paramètre correspondant est négatif.

Écrivez un constructeur à trois paramètres réels `a`, `b`, `c`.

Écrivez un constructeur issu de trois `Point p1`, `p2`, `p3`.



Validez vos méthodes avec la solution.

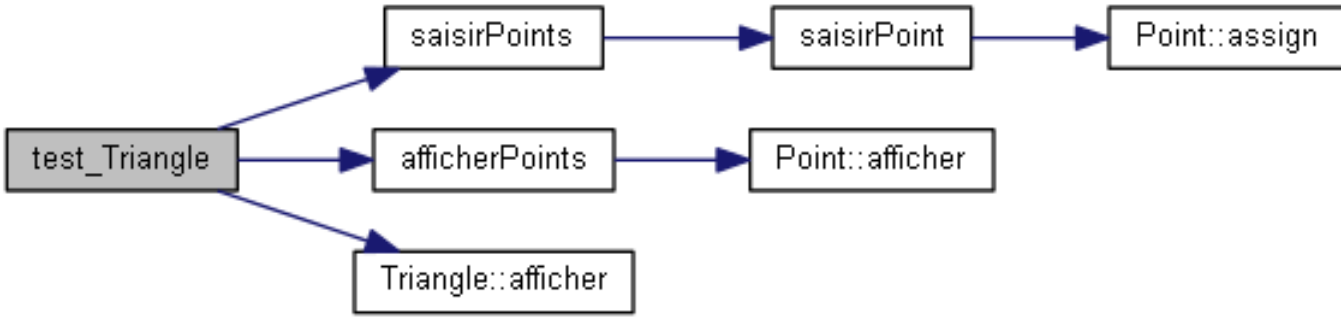
Écrivez des accesseurs `getL1`, `getL2` et `getL3` de la longueur de chacun des côtés.

Écrivez une méthode `afficher` qui affiche les longueurs des trois côtés du triangle sous le format `<l1,l2,l3>`.

Validez vos méthodes avec la solution.

2.2 Test de la classe Triangle

Ce problème utilise les procédures `saisirPoints` et `afficherPoints`.



Écrivez une procédure `test_Triangle` qui instancie et saisit trois `Point`s, les affichent, puis instancie un `Triangle` défini par ces trois points et l'affiche.

Testez. Exemple d'exécution :

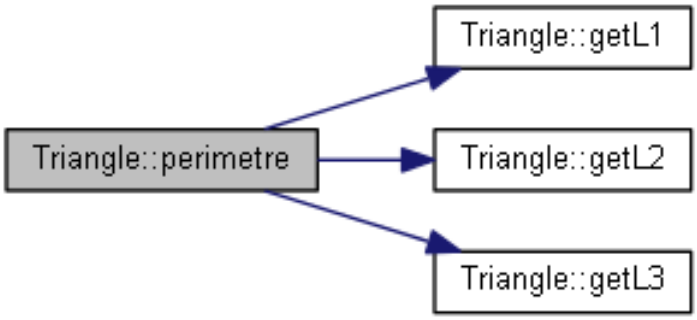
```
Premier point (x,y)? (-3.5,0)
Deuxieme point (x,y)? (3.5,0)
Troisieme point (x,y)? (0,4)
  P1 = (-3.5,0), P2 = (3.5,0), P3 = (0,4)
  Longueurs = <7,5.31507,5.31507>
```

Validez votre procédure avec la solution.

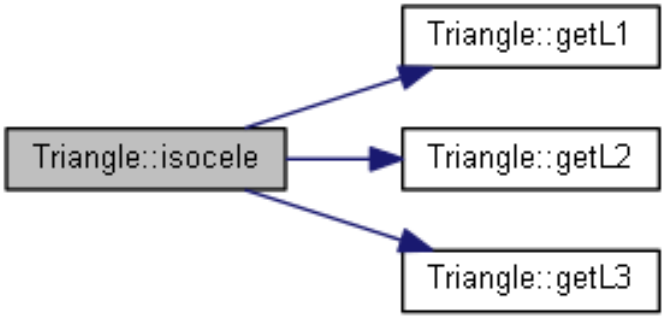
2.3 Géométrie d'un triangle

Ce problème étend la classe `Triangle` avec des caractéristiques de géométrie.

Écrivez une méthode `perimetre` qui calcule et renvoie le périmètre du triangle.



Écrivez une méthode `isocele` qui teste et renvoie `Vrai` si l'instance courante est un triangle isocèle, `Faux` sinon. (Un triangle est *isocèle* si au moins deux côtés ont la même longueur.)



Validez vos méthodes avec la solution.

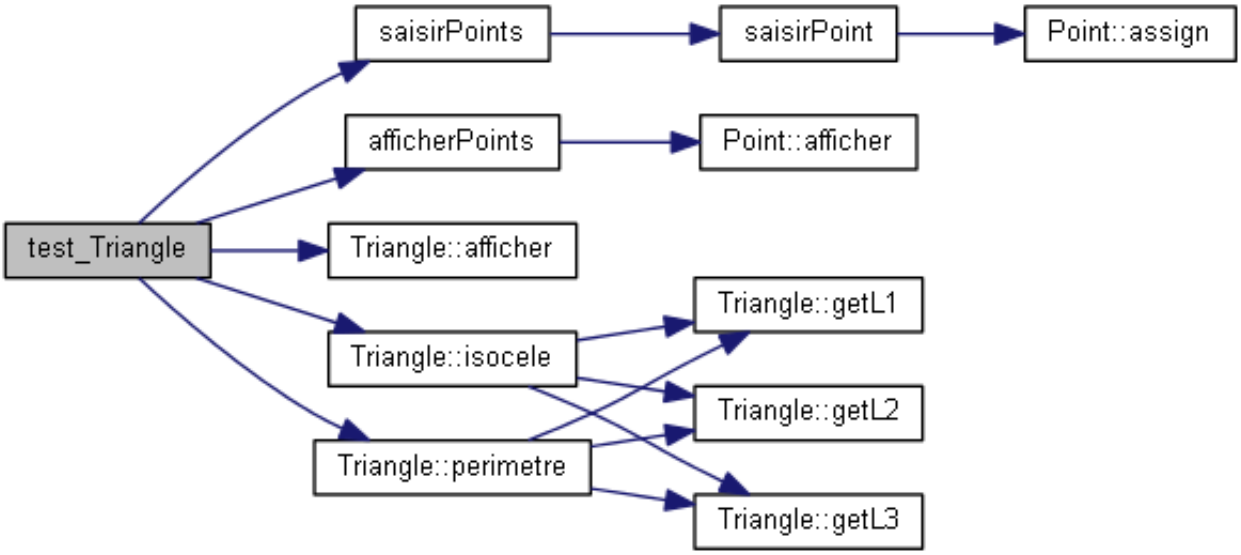
Validez la définition de votre classe avec la solution.

Solution Java @[Triangle.java]

```java
public class Triangle{
private double m_l1, m_l2, m_l3;

public Triangle(double a, double b, double c){
  assign(a, b, c);
}

public Triangle(Point p1, Point p2, Point p3){
  assign(p1.distance(p2), p2.distance(p3), p3.distance(p1));
}

public void assign(double a, double b, double c){
  m_l1 = (a >= 0 ? a : 0.0);
  m_l2 = (b >= 0 ? b : 0.0);
  m_l3 = (c >= 0 ? c : 0.0);
}

public double getL1(){
  return m_l1;
}

public double getL2(){
  return m_l2;
}

public double getL3(){
  return m_l3;
}

public void afficher(){
  System.out.print("<"+getL1()+","+getL2()+","+getL3()+">");
}
```

```
public String str(){
  return "<"+getL1()+","+getL2()+","+getL3()+">";
}

public double perimetre(){
  return getL1() + getL2() + getL3();
}

public boolean isocele(){
  return (getL1() == getL2())
      || (getL2() == getL3())
      || (getL3() == getL1());
}
}
```


Complétez la procédure `test_Triangle` avec le calcul et l'affichage des caractéristiques de géométrie.



Testez. Exemples d'exécution :

```
Premier point (x,y)? (-3.5,0)
Deuxieme point (x,y)? (3.5,0)
Troisieme point (x,y)? (0,4)
  P1 = (-3.5,0), P2 = (3.5,0), P3 = (0,4)
  Longueurs = <7,5.31507,5.31507>
  Perimetre = 17.6301
  Isocele = 1
```

```
Premier point (x,y)? (0,0)
Deuxieme point (x,y)? (3,0)
Troisieme point (x,y)? (0,4)
  P1 = (0,0), P2 = (3,0), P3 = (0,4)
  Longueurs = <3,5,4>
  Perimetre = 12
```

```
  Isocele = 0
```

Validez votre procédure avec la solution.

Solution Java @[pgtriangle.java]

```
import java.util.Scanner;
import java.util.Locale;
public class PGTriangle{
static void saisirPoint(String txt,Point p){
  Scanner cin = new Scanner(System.in);
  cin.useLocale(Locale.US);
  System.out.print(txt);
  double x = cin.nextDouble();
  double y = cin.nextDouble();
  p.assign(x,y);
}

static void saisirPoints(Point p1,Point p2,Point p3){
  saisirPoint("Premier point (x,y)? ",p1);
  saisirPoint("Deuxieme point (x,y)? ",p2);
  saisirPoint("Troisieme point (x,y)? ",p3);
}

static void afficherPoints(Point p1,Point p2,Point p3){
  System.out.print("  P1 = "+p1.str());
  System.out.print(", P2 = "+p2.str());
  System.out.println(", P3 = "+p3.str());
}

static void test_Points(){
  Point p1 = new Point();
  Point p2 = new Point();
  Point p3 = new Point();
  saisirPoints(p1, p2, p3);
  afficherPoints(p1, p2, p3);
}

static void test_Triangle0(){
  Point p1 = new Point();
  Point p2 = new Point();
  Point p3 = new Point();
  saisirPoints(p1, p2, p3);
  afficherPoints(p1, p2, p3);
  Triangle tr = new Triangle(p1, p2, p3);
  System.out.print("  Longueurs = ");
  tr.afficher();
  System.out.println();
}

static void test_Triangle(){
  Point p1 = new Point();
  Point p2 = new Point();
  Point p3 = new Point();
  saisirPoints(p1, p2, p3);
  afficherPoints(p1, p2, p3);
```

```
  Triangle tr = new Triangle(p1, p2, p3);
  System.out.println("  Longueurs = "+tr.str());
  System.out.println("  Perimetre = "+tr.perimetre());
  System.out.println("  Isocele = "+tr.isocele());
}
public static void main(String[] args){
  test_Points();
  test_Triangle0();
  test_Triangle();
}
}
```

3 Références générales

Comprend ■