TP Test unitaire

Clément Hue

Question 1

Classe de comportement	Sous-domaine d'entrée	Représentant sélectionne	Résultat classifie
Équilatéral	x,y,z ≥ 0 et x=y=z	(3,3,3)	Categorie.équilatéral
Isocèle	x,y,z ≥ 0 et (x=y et y!=z ou x=z et z!=y ou y=z et z!= x)	(4,4,7)	Categorie.Isocele
Quelconque	x,y,z ≥ 0 et (x!=y et x!=z et z!=y)	(7,5,9)	Categorie.Quelconque

Classe de comportement	Sous-domaine d'entrée	Représentant sélectionne	Résultat constructeur
invalide	x,y,z ≤ 0	(-5,-2,-8)	Exeption IllegalArgumentException
invalide	x+y < z ou x+z < y ou z+y < x	(4,4,9)	Exeption IllegalArgumentException
valide	x,y,z > 0	(4,4,5)	new Triangle

Question 2

Test au limite_

Classe de comportement	Sous-domaine d'entrée	Représentant sélectionne	Résultat classifie
Isocèle	x,y,z ≥ 0 et (x=y ou x=z ou y=z)	(4,4,8)	Categorie.isocèle
Quelconque	x!=y!=z	(1,,2,3)	Catégorie quelconque

Classe de comportement	Sous-domaine d'entrée	Représentant sélectionne	Résultat constructeur
valide	x,y,z > 0	(8,4,4)	new Triangle
invalide	x,y,z ≤ 0	(4,3,8)	Exeption IllegalArgumentExce

			ption
invalide	x,y,z ≤ 0	(0,0,0)	Exeption IllegalArgumentException

Question 4

```
test boite noir
le triangle est créé a:3 b:3 c:3 true
le triangle est créé a:8 b:4 c:4 false
le triangle est équilatéral a: 3 b:3 c:3 true
le triangle est isocèle a: 4 b:4 c:7 true
le triangle est quelconque a: 7 b:5 c:9 true
le triangle est isocèle a: 4 b:4 c:8 true
le triangle n'est pas créé a:-5 b:-2 c:-8 true
le triangle n'est pas créé a:4 b:4 c:9 true
le triangle est créé a:4 b:4 c:5 true
le triangle est créé a:4 b:4 c:8 true
le triangle n'est pas créé a:4 b:3 c:8 true
le triangle n'est pas créé a:0 b:0 c:0 true
le triangle est quelconque a: 1 b:2 c:3 true
```

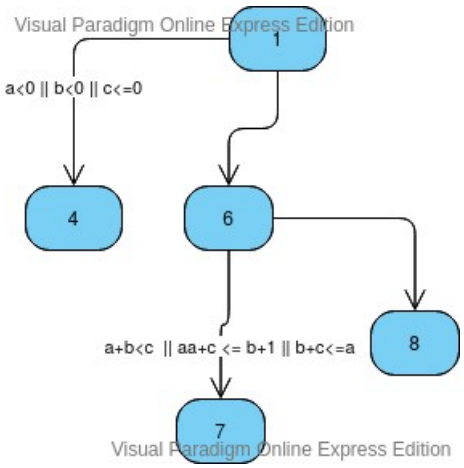
Le test pour le triangle (8,4,4) n'est pas passé, il devait être valide, mais il ne l'est pas

Question 5

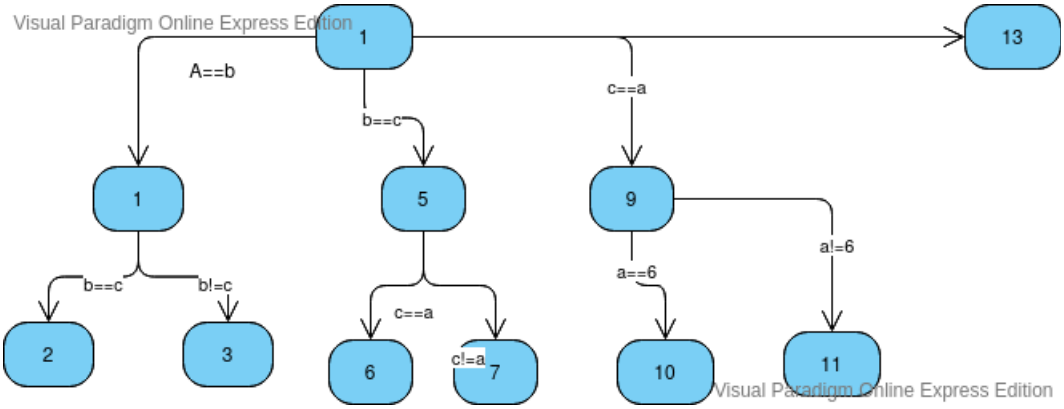
Non, il faut laisser les tests à l'extérieur, sinon ils deviennent couplé au code d'une part et cela disperserais les tests d'autre part.
Le seul intérêt de mettre le test dans la classe serait de tester les méthodes privés, mais cela ne devrait jamais arriver. Tester l'api public est plus pertinent pour permettre du refactoring derrière.

Question 6

constructeur



classifie



Question 7

On peut effectuer la couverture par chemin car c'est la plus fine.
Il peut être utile de décomposer les conditions, pour avoir des tests encore plus fin.

Question 8u

constructeur

Chemin	DT	Résultat attendu
(1,4)	(-4,8,2)	Exception
(1,6,7)	(3,3,7)	Exception
(1,6,8	(4,4,8)	New Triangle

Classifie

Chemin	DT	Résultat attendu
(1,1,2)	(3,3,3)	Equilateral
(1,1,3)	(3,3,5)	Isocele
(1,5,6)	(4,4,4)	Equilateral
(1,5,7)	(5,3,3)	Isocele
(1,9,10)	(6,2,6)	Isocele
(1,9,11)	(8,6,8)	Isocele
(1,13)	(4,5,6)	Quelconque

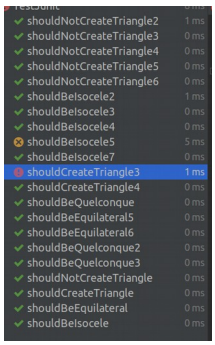
Question 9 et 10

```
test boite noir
le triangle est créé a:3 b:3 c:3 true
le triangle est créé a:8 b:4 c:4 false
le triangle est équilatéral a: 3 b:3 c:3 true
le triangle est isocèle a: 4 b:4 c:7 true
le triangle est quelconque a: 7 b:5 c:9 true
le triangle est isocèle a: 4 b:4 c:8 true
le triangle n'est pas créé a:-5 b:-2 c:-8 true
le triangle n'est pas créé a:4 b:4 c:9 true
le triangle est créé a:4 b:4 c:5 true
le triangle est créé a:4 b:4 c:8 true
le triangle n'est pas créé a:4 b:3 c:8 true
le triangle n'est pas créé a:0 b:0 c:0 true
le triangle est quelconque a: 1 b:2 c:3 true
test boite blanche
le triangle n'est pas créé a:-4 b:8 c:2 true
le triangle n'est pas créé a:3 b:3 c:7 true
le triangle est créé a:4 b:4 c:8 true
le triangle est équilatéral a: 3 b:3 c:3 true
le triangle est équilatéral a: 4 b:4 c:4 true
le triangle est isocèle a: 3 b:3 c:5 true
le triangle est isocèle a: 5 b:3 c:3 true
le triangle est isocèle a: 6 b:2 c:6 false
le triangle est isocèle a: 8 b:6 c:8 true
le triangle est quelconque a: 4 b:5 c:6 true
```

Le test pour le triangle (6,2,6) ne passe pas.
Le test boite noir, et boite blanche ont détecté des erreurs différentes, ils se complétent.

Question 11 et 12

Junit ma détecté 2 tests faux



Le cas où l'on créer un triangle(6,2,6), on s'attend à ce qu'il soit isocèle, mais il est équilatéral.
Le cas ou l'on créer le triangle(8,4,4), on obtient une exception.
On obtient les mêmes résultats que la façon manuelle.

Question 13

Il met 8ms pour exécuter les tests

Question 14

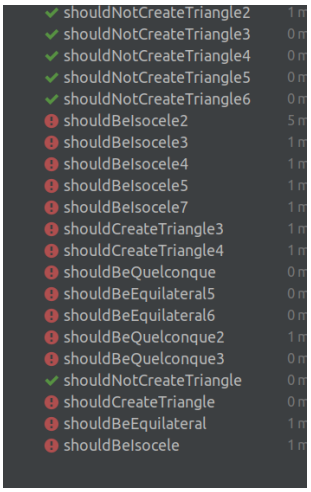
On peut exécuter 450000 fois les tests par heure.

Question 15

1) on inverse le signe de a

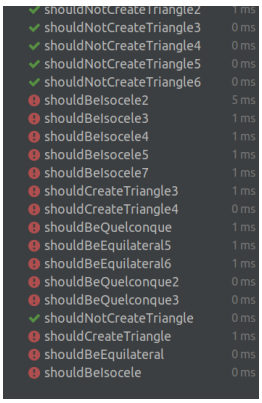
```
if (a>0 || b<0 || c<=0)
```

mutation détecté



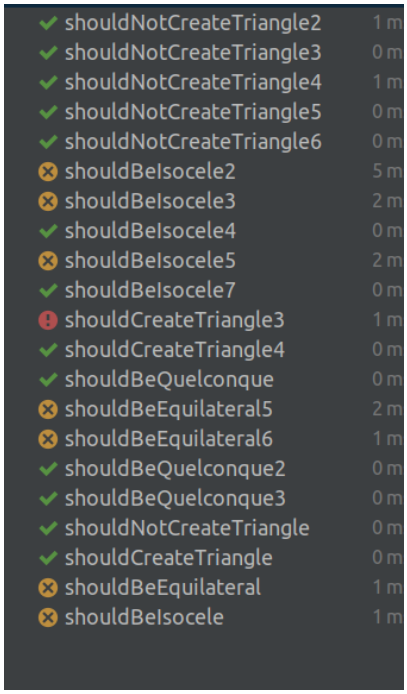
2) a -b au lieu de a +b

```
if (a - b < c || a + c <= b + 1 || b + c <= a)
```



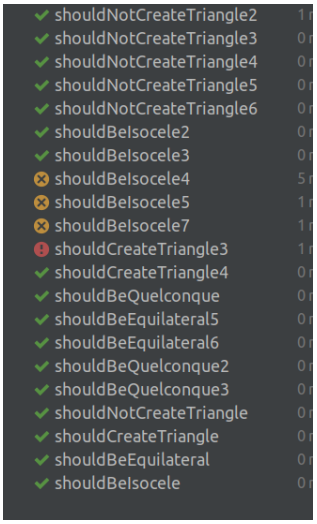
3) b!=c au lieu de b==c

```
if (a == b) {
    if (b!=c) return Categorie.Equilateral ;
```



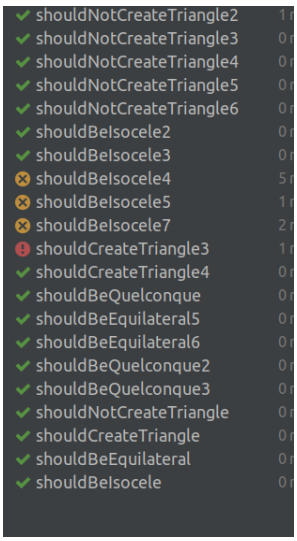
4) c!=a au lieu de c==a

```
if (b == c) {
    if (c != a) return Categorie.Equilateral ;
```



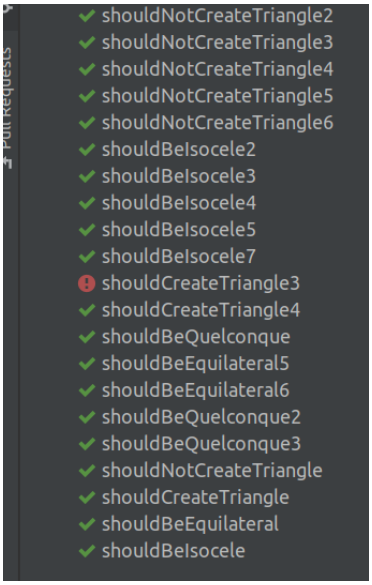
5) return quelconque au lieu de isocèle

```
if (b == c) {
    if (c == a) return Categorie.Equilateral ;
    else return Categorie.Quelconque ;
```



6) a==10 au lieu de a==6

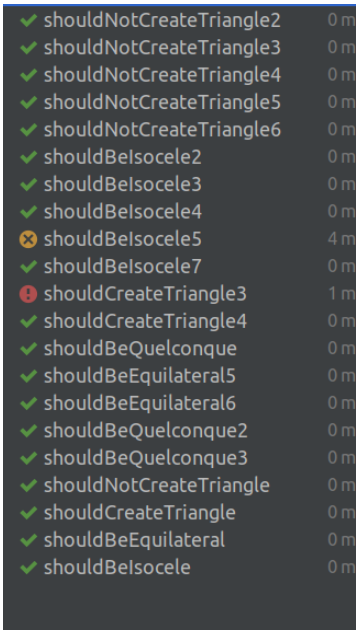
```
if (c == a) {
    if (a == 10) return Categorie.Equilateral ;
```



la seul est erreur que l'on perçoit est celle, déjà présente sans mutant, donc les tests ne sont pas assez fin pour détecter ce changement

7) && au lieu de ||

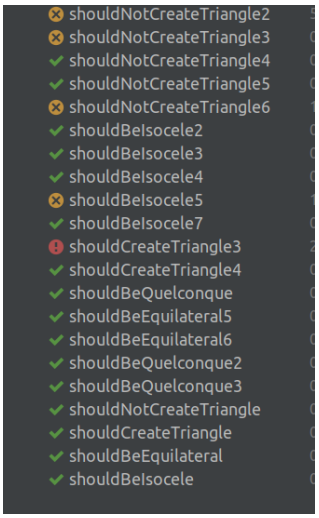
```
if (a<0 && b<0 || c<=0)
    throw new IllegalArgumentException () ;
```



Idem qu'au dessus le mutant n'est pas détecté

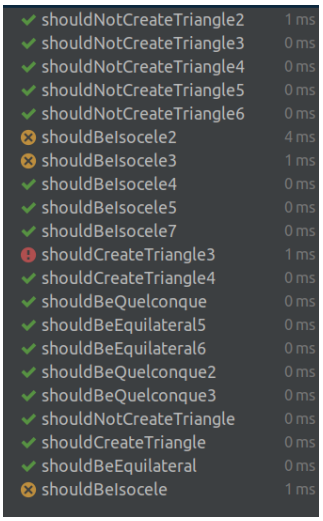
8) && au lieu de ||

```
if (a + b < c && a + c <= b + 1 || b + c <= a)
    throw new IllegalArgumentException () ;
```



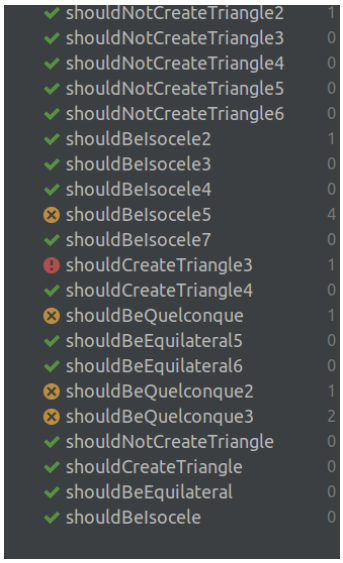
9) a==c au lieu de a==b

```
if (a == c) {
    if (b==c) return Categorie.Equilateral ;
```



10) équilatéral au lieu de isocèle

```
    return Categorie.Equilateral ;
}
```



Question 16

Sur les 10 mutants, 2 n'ont pas été détectés, les tests sont pas totalement fiable, mais permettre à priori de détecter 80 % des changements.