

PROJET ITIC

ATRMOUH Boujamaa
MAZAN Kamil

Année 2023-2024 - 24 décembre 2023

Table des matieres

Table des matières

PROJET ITIC.....	0
Table des matieres	1
Test Fonctionnel.....	2
Valeur (valeur : Object C \rightarrow Object O).....	2
Test Structurel	3
IsSolved (void \rightarrow void)	3
Rapport et tests	4
Pourcentage de Couverture à travers clEmma (Eclipse).....	4
Pourcentage de Couverture à travers de tests Mutationnels	5

Test Fonctionnel

Valeur (valeur : Object C \rightarrow Object O)

Après une analyse des valeurs d'entrée, et établissement des partitions suivantes :

Paramètre d'entrée	Code_Partition	Condition de partition
Object C	Partition1_1	Object C absente
	Partition1_2	Object C presente
	Partition1_3	Object C is null
	Partition1_4	Object C not null
This.Object	Partition2_1	This.object is empty
	Partition2_2	This.object is not empty

Voici les résultats des tests JUnit, effectuées :

Code test	Résultat obtenu	Test réussit ou échoué	Automatisation (code)			
			Classe de test (junit)	Opération de test (junit)	Préfixe	Postfixe
Test1	Exception	réussit	testValeur	valeurTest1	container = null C = null O = null	container = null C = null O = null
Test2	Exception	réussit	testValeur	valeurTest2	container = null C = null O = null	container = null C = null O = null
Test3	ErreurConteneur	réussit	testValeur	valeurTest3	C = null O = null	C = null O = null
Test4	ErreurConteneur	réussit	testValeur	valeurTest4	container = null C = null O = null	container = null C = null O = null
Test5	Exception	réussit	testValeur	valeurTest5	container = null C = null O = null	container = null C = null O = null
Test6	Exception	réussit	testValeur	valeurTest6	container = null C = null	container = null C = null
Test7	Object O	échoué	testValeur	valeurTest7	container = null C = null O = null	container = null C = null O = null
Test8	Object O	réussit	testValeur	valeurTest8	container = null C = null O = null	container = null C = null O = null

Les résultats sont les mêmes pour testEtat4.jar et pour testEtat7.jar.

Un test a échoué, car il a été supposé que la clé existait mais était nulle, et que le conteneur n'était pas vide. Selon le cahier des charges, une clé ne peut pas être vide, donc nous attendions une exception. Cependant, le test a retourné une valeur, ce qui indique que la clé peut être vide. Par conséquent, il est nécessaire de corriger la fonction d'ajout ou le constructeur du conteneur afin de vérifier si la clé que l'on tente d'ajouter est vide. Dans le cas où la clé est vide, l'ajout dans le conteneur ne devrait pas être autorisé.

Test Structurel

IsSolved (void → void)

Après une analyse en profondeur sur différents critères, comportant All-nodes ou encore Tous-les-utilisations, nous avons pu identifier 9 chemins, cependant après analyse symbolique il se trouve que certains d'entre eux ne sont pas exécutables :

All-nodes, all-arcs, tous-les-définitions (Chemin 1) :

Code Chemin	Objective du chemin				Conclusion de l'analyse
Ch1	All-nodes, All-arcs, Tous-les-définitions				impossible
Chemin	Symboles				Condition
	currentRow	currentColumn	dimension	candidate	
1	currentRow0	currentColumn0	dimension0	candidate0	
2	0	currentColumn0	dimension0	candidate0	
2'	0	currentColumn0	dimension0	candidate0	
3	0	0	dimension0	candidate0	$0 < \text{dimension0}$
3'	0	0	dimension0	candidate0	
2"	1	0	dimension0	candidate0	$0 >= \text{dimension0}$
2'	1	0	dimension0	candidate0	
9	true				$1 >= \text{dimension0}$

Tous-les-utilisations (Chemin 7) :

Code Chemin	Objective du chemin				Conclusion de l'analyse
Ch7	Tous-les-utilisations				impossible
Chemin	Symboles				Condition
	currentRow	currentColumn	dimension	candidate	
1	currentRow0	currentColumn0	dimension0	candidate0	
2	0	currentColumn0	dimension0	candidate0	
2'	0	currentColumn0	dimension0	candidate0	
3	0	0	dimension0	candidate0	$0 < \text{dimension0}$
3'	0	0	dimension0	candidate0	
4	0	0	dimension0	Box(0,0)	
5	0	0	dimension0	Box(0,0)	
3"	0	1	dimension0	Box(0,0)	$\text{Box}(0,0).isSolved()$
3'	0	1	dimension0	Box(0,0)	
2"	1	1	dimension0	Box(0,0)	$1 >= \text{dimension0}$
2'	1	1	dimension0	Box(0,0)	
3	1	1	dimension0	Box(0,0)	$1 < \text{dimension0}$
3'	1	1	dimension0	Box(0,0)	
4	1	1	dimension0	Box(1,1)	$1 < \text{dimension0}$
5	1	1	dimension0	Box(1,1)	
6	1	1	dimension0	Box(1,1)	$! \text{Box}(1,1).isSolved()$
end	false				

Ici, nous voyons bien qu'il y a contradiction dans les conditions, la dimension ne peut pas être à la fois inférieure à 0 et supérieur ou égale à 0. Ainsi, les chemins ne sont jamais parcourus.

L'analyse symbolique nous permet d'établir 7 tests distincts :

Automatisation (code)						
Code test	Résultat obtenu	Test réussit ou échoué	Classe de test	Opération de test	Préfixe	Postfixe
Test1	false	réussit	testIsSolved	isSolvedTest1	this.Puzzle = null	this.Puzzle = null
Test2	true	réussit	testIsSolved	isSolvedTest2	this.Puzzle = null	this.Puzzle = null
Test3	false	réussit	testIsSolved	isSolvedTest3	this.Puzzle = null	this.Puzzle = null
Test4	true	réussit	testIsSolved	isSolvedTest4	this.Puzzle = null	this.Puzzle = null
Test5	false	réussit	testIsSolved	isSolvedTest5	this.Puzzle = null	this.Puzzle = null
Test6	false	réussit	testIsSolved	isSolvedTest6	this.Puzzle = null	this.Puzzle = null
Test7	false	réussit	testIsSolved	isSolvedTest7	this.Puzzle = null	this.Puzzle = null

Rapport et tests








Pourcentage de Couverture à travers clEmma (Eclipse)

Ici, nous avons décidé d'utiliser testEtat7.jar. Car après l'analyse fonctionnel, nous avons constaté que cette version fonctionnait bien mieux.





test (21 déc. 2023 11:04:41)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 Code		52 %		23 %	116	261	276	638	60	193	3	11
Total	1 210 of 2 552	52 %	104 of 136	23 %	116	261	276	638	60	193	3	11




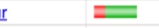


Code

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 src		24 %		23 %	99	130	201	266	43	62	3	6
 test		81 %		n/a	17	131	75	372	17	131	0	5
 target/generated-sources/annotations		n/a		n/a	0	0	0	0	0	0	0	0
 target/generated-test-sources/test-annotations		n/a		n/a	0	0	0	0	0	0	0	0
Total	1 210 of 2 552	52 %	104 of 136	23 %	116	261	276	638	60	193	3	11




test

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 tests.fonctionnels		78 %		n/a	17	112	75	311	17	112	0	3
 tests.structurels		100 %		n/a	0	19	0	61	0	19	0	2
Total	229 of 1 261	81 %	0 of 0	n/a	17	131	75	372	17	131	0	5

tests.fonctionnels

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 testAjouter		80 %		n/a	14	79	58	234	14	79	0	1
 testValeur		73 %		n/a	1	23	16	61	1	23	0	1
 testCreerConteneur		66 %		n/a	2	10	1	16	2	10	0	1
Total	229 of 1 066	78 %	0 of 0	n/a	17	112	75	311	17	112	0	3

tests.structurels

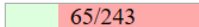
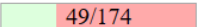
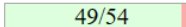
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 testIsSolved		100 %		n/a	0	12	0	44	0	12	0	1
 testParseSolvedBoxesFromInputString		100 %		n/a	0	7	0	17	0	7	0	1
Total	0 of 195	100 %	0 of 0	n/a	0	19	0	61	0	19	0	2

Nous constatons que nos tests fonctionnels sont couverts à 78% par l'exécuteur. Et les tests structurels à 100%.

Pourcentage de Couverture à travers de tests Mutationnels

Pit Test Coverage Report

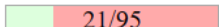
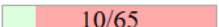
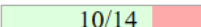
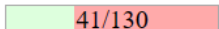
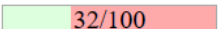
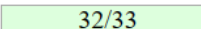
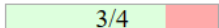
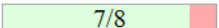
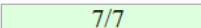
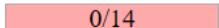
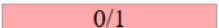
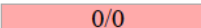
Project Summary

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
4	27%  65/243	28%  49/174	91%  49/54

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage	Test Strength
sudoku	4	27%  65/243	28%  49/174	91%  49/54

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
Box.java	22%  21/95	15%  10/65	71%  10/14
Puzzle.java	32%  41/130	32%  32/100	97%  32/33
SudokuUtils.java	75%  3/4	88%  7/8	100%  7/7
exemple_sudoku.java	0%  0/14	0%  0/1	0%  0/0

Line coverage : Ici on peut voir que nos tests gèrent 65 lignes sur 243 lignes de codes ce qui représente 27% sachant qu'on teste uniquement 2 méthodes. Mais vu que ces dernières en appel d'autres, ce résultat est tout à fait plausible.

Mutation Coverage : Nos tests qui gèrent uniquement 65 lignes de codes gèrent 49 mutations sur 174 possibles trouver par le PIT sur l'ensemble du code.

Test Strength : Nos tests ont permis de gérer 49 mutations sur 54 mutations trouver par le PIT, ce qui correspond à 91% de mutations.