title: Dossier de tests author: Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Dossier de tests

Nom du projet : SAE_2.01-02-05_RPG-SIMULATOR	Version : 0.0.02a
Document : Dossier de test	Date : 18/05/2023
Responsables de la rédation : Lucas DA SILVA FERREIRA & Baptiste FOURNIE	

Sommaire

1. Introduction

Ce travail est un projet lié a la SAE 2.01, 2.02 et 2.05. Il consiste à la réalisation d'un simulateur de RPG, ce document est un dossier de test qui consiste a decrire les procédures de tests mises en oeuvre pour le projet uniquement sur le package modèle comme il à été demandé. Le but de ce dossier de test est de verifier si les classes et méthodes ne comportent pas d'erreurs et correspondent a la spécification demandé.

2. Description de la procédure de test

Ces tests seront appliqué de manière récursif à chaque ajout de méthode, il sera de la forme d'un cycle en V récursive en somme. Tout les tests faits seront des tests unitaires qui utiliseront la stratégie de la "Boite noire"

3. Description des informations à enregistrer pour le test

3.1. Campagne de test

Produit testé :	RPG SIMULATOR
Configuration logicielle :	IntelliJ IDEA 2022.3.2, OpenJDK 19.02, Windows 10 Professionnal
Configuration matérielle :	INTEL CORE I7-11800H 16x4.6Ghz, RAM 2x32Gb 3200Mhz DDR4, Geforce RTX 3070 ti
Date de début : 18/05/2023	Date de finalisation : 8/06/2023
Tests à appliquer :	**CLASSE Quete.java TESTS / RESULTATS** • Test unitaire sur la méthode "extraitPrecond()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "extraitPos()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "testprecondition() en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "distanceQuete()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "queteProche()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "quetePossible()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "compareTo()" en utilisant la méthode des boites noires. Résultat ici **CLASSE Parcours.java TESTS / RESULTATS** • Test unitaire sur la méthode "extraitQuete()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "queteFinPossibleEfficace()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "compareTo()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "quetesPossibles()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "ajouteDuree()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "getQueteActuelle()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "queteFinPossibleExhaustive()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "getCle()" en utilisant la méthode des boites noires. Résultat ici

	• Test unitaire sur la méthode "compareTo()" en utilisant la méthode des boites noires. VERSION 2.0Résultat ici • Test unitaire sur la méthode "ajouterDeplacement()" en utilisant la méthode des boites noires. Résultat ici **CLASSE Classement.java TESTS / RESULTATS** • Test unitaire sur la méthode "ajout()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "afficherClassement()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "afficherClassement()" en utilisant la méthode des boites noires. VERSION 2.0Résultat ici • Test unitaire sur la méthode "regulationParcours()" en utilisant la méthode des boites noires. Résultat ici **CLASSE Algorithme.java TESTS / RESULTATS** • Test unitaire sur la méthode "solutionGloutonneEfficace()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "solutionGloutonneExhaustive()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "solutionSpeedrunEfficace()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "SolutionSpeedrunExhaustive()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "SolutionEfficaceNbQuete()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "SolutionExhaustiveNbQuete()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "SolutionDeplacementEfficace()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "SolutionDeplacementExhaustive()" en utilisant la méthode des boites noires. Résultat ici • Test unitaire sur la méthode "choixAlgo()" en utilisant la méthode des boites noires. Résultat ici
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

3.2. Tests

CLASSE "Quete.java" :

METHODE "extraitPrecond()" :

- Choix de partition : Les deux listes contenant que des entiers naturels privés de 0 peut etre chacun divisé en trois sous ensemble. A1 la liste est égale à None , A1 la liste contient un entier , A2 la liste

contient deux entiers. a represente la premiére liste , b représente la deuxiéme liste.Quand a n'existe pas , b est obligé d'être None sinon il deviendrait la premiére liste.

Identification du test : extraitPrecond()	Version : 1.0
Description du test :	Test Unitaire sur la méthode "extraitPrecond()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	a	b	Résultat attendu
P0	a = None ou ensemble vide	b = None ou ensemble vide	[0,0,0,0]
P1	a contient un entier	b = None ou ensemble vide	[i,0,0,0]
P2	a contient un entier	b contient un entier	[i,0,y,0]
P3	a contient un entier	b contient 2 entiers	[i,0,y,z]
P4	a contient 2 entiers	b = None ou ensemble vide	[i,y,0,0]
P5	a contient 2 entiers	b contient 1 entier	[i,y,z,0]
P6	a contient 2 entiers	b contient 2 entiers	[i,y,z,x]

Classe	a	b	Résultat
P0	None	None	[0,0,0,0]
P1	(1)	None	[1,0,0,0]
P2	(1)	(3)	[1,0,3,0]
P3	(1)	(3,4)	[1,0,3,4]
P4	(1,2)	None	[1,2,0,0]
P5	(1,2)	(3)	[1,2,3,0]
P6	(1,2)	(3,4)	[1,2,3,4]

retour au sommaire

METHODE "extraitPos()" :

- Choix de partition : l'ensemble des entiers naturels n'a pas besoin d'être partitionné.

Identification du test : extraitPos()	Version : 1.0
Description du test :	Test unitaire sur la méthode "extraitPos()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	extraitPos	Résultat attendu
P0	"(i,y)"	[i,y]

Classe	a	Résultat
P0	"(2,4)"	[2,4]

retour au sommaire

METHODE "testprecondition()" :

- Choix de partition : le tableau precond peut etre partionné en trois ensembles, A1 ne contient que des 0, A2 ne contient pas que des 0 et A3 ne contient 0.

Identification du test : testprecondition()	Version : 1.0
Description du test :	Test unitaire sur la méthode "testprecondition() en utilisant la méthode des boites noires."
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	extraitPos	Résultat attendu
P0	"(i,y)"	[i,y]

Classe	a	Résultat
P0	"(2,4)"	[2,4]

retour au sommaire

METHODE "distanceQuete()" :

- Choix de partition : Pour les deux colonnes ont peut les partitionnés en 3 parties, on considère que x et y sont des entiers positifs, x1 est égale à x2, x1 est inférieur à x2, x1 est supérieur à x2 et y1 est égale à y2, y1 est inférieur à y2, y1 est supérieur à y2.

Identification du test : distanceQuete()	Version : 1.0
Description du test :	Test unitaire sur la méthode "distanceQuete()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	x	y	Résultat attendu
P0	x1=X2	y1=y2	0
P1	x1=X2	y1>y2	abs(y1 - y2)
P2	x1=X2	y1<y2	abs(y1 - y2)
P3	x1>X2	y1=y2	abs(x1 - x2)
P4	x1>X2	y1>y2	abs(x1 - x2) + abs(y1 - y2)
P5	x1>X2	y1<y2	abs(x1 - x2) + abs(y1 - y2)
P6	x1<X2	y1=y2	abs(x1-x2)
P7	x1<X2	y1>y2	abs(x1 - x2) + abs(y1 - y2)
P8	x1<X2	y1<y2	abs(x1 - x2) + abs(y1 - y2)

Classe	x	y	Résultat attendu
P0	x1,x2=2	y1,y2=2	0
P1	x1,x2=2	y1=2,y2=1	1
P1	x1,x2=2	y1=1,y2=2	1
P3	x1=2,x2=1	y1,y2=2	1
P4	x1=2,x2=1	y1=2,y2=1	2
P5	x1=2,x2=1	y1=1,y2=2	2
P6	x1=1,x2=2	y1,y2=2	1
P7	x1=1,x2=2	y1=2,y2=1	2
P8	x1=1,x2=2	y1=1,y2=2	2

retour au sommaire

METHODE "queteProche()" :

- Choix de partition : On considère que la distance est un entier positif! Elle peut être partitionné en 3. Distance Q1 = Distance Q2 ou Distance Q1 > Distance Q2 ou Distance Q1 < Distance Q2.

Identification du test : queteProche()	Version : 1.0
Description du test :	Test unitaire sur la méthode "queteProche()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	distance	Résultat attendu
P0	Q1 <Q2	(Q1)
P1	Q1 = Q2	(Q1,Q2)
P2	Q1 >Q2	(Q2)

Classe	distance	Résultat
P0	Q1 = 1, Q2 =2	(Q1)
P1	Q1 = 1, Q2 = 1	(Q1,Q2)
P2	Q1 = 2, Q2 = 1	(Q2)

retour au sommaire

METHODE "quetePossible()" :

- Choix de la partition : I0, I1, I2 et I3 représente chacun un indice du tableau précondition. On considère dans ce choix de partition que true veut dire que l'id de la quête necessaire est faite.

Identification du test : quetePossible()	Version : 1.0
Description du test :	Test unitaire sur la méthode "quetePossible()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

campagne de test :	

Classe	I0	I1	I2	I3	Résultat attendu
P0	true	ensemble de la partition	true	ensemble de la partition	true
P1	true	ensemble de la partition	ensemble de la partition	true	true
P2	ensemble de la partition	true	true	ensemble de la partition	true
P3	ensemble de la partition	true	ensemble de la partition	true	true
P4	true	ensemble de la partition	false	false	false
P5	ensemble de la partition	true	false	false	false
P6	false	false	ensemble de la partition	true	false
P7	false	false	true	ensemble de la partition	false
P8	false	false	false	false	false

liste id = {1,2,3,4}

Classe	I0	I1	I2	I3	Résultat
P0	1	0	3	2	true
P1	1	0	6	2	true
P2	6	1	2	0	true
P3	6	1	3	2	true
P4	1	0	8	9	false
P5	6	1	8	0	false
P6	8	9	6	1	false
P7	8	9	0	0	false
P8	8	9	6	7	false

retour au sommaire

METHODE "compareTo()" :

- Choix de la partition : Les ID sont des entiers positif. Il peut être partitionné en 3 parties. l'ID de this est égale a la quete pointé qui est this. l'ID de la quete this est inférieur a la quete pointé. l'ID de la quete this est supérieur a la quete pointé.

Identification du test : compareTo()	Version : 1.0
Description du test :	Test unitaire sur la méthode "compareTo()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	id	Résultat attendu
P0	this.id >parquete.id	>0
P1	this.id =parquete.id	=0
P2	this.id <parquete.id	<0

Classe	id	Résultat attendu
P0	this = 2 par = 1	1
P1	this = 2 par = 2	=0
P2	this= 1 par = 2	-1

retour au sommaire

CLASSE "Parcours.java" :

METHODE "extraitQuete()" :

- Choix de la partition : Une TreeSet de toute les quetes du Scénario.

Identification du test : extraitQuete()	Version : 1.0
Description du test :	Test unitaire sur la méthode "extraitQuete()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut

Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	tree set	Résultat attendu
P0	"(qi,qy,qz)"	(qy,qz)

quete 0 = quete fin

Classe	tree set	Résultat
P0	"(q0,q1,q2)"	(q1,q2)

retour au sommaire

METHODE "queteFinPossibleEfficace()" :

- Choix de la partition : l'experience acqueri par le joueur et une liste de quete possible. Une partie ou l'experience est infèrieure au prérequis de la quete, un ou elle est égale et une autre ou elle est supérieure. Avec des parties ou des quetes sont possibles ou non.

Identification du test : queteFinPossibleEfficace()	Version : 1.0
Description du test :	Test unitaire sur la méthode "queteFinPossibleEfficace()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Experience	Quete Possible	Résultat attendu
P0	p.exp>qFin.exp	true	true
P1	p.exp=qFin.exp	true	true
P2	p.exp<qFin.exp	peu importe	false
P3	peu importe	false	false

liste id = {1,2,3}

Classe	Experience	Quete Possible	Résultat
P0	p = 300, qFin =200	precond [1,0,3,0]	true
P1	p = 200, qFin =200	precond [1,0,3,0]	true

Classe	Experience	Quete Possible	Résultat
P2	p = 150, qFin =200	precond [4,0,3,0]	false
P3	p = 200, qFin =200	precond [4,0,3,0]	false

retour au sommaire

METHODE "compareTo()" :

- Choix de la partition : comparaison de durée, this inférieur a la durée du parcours, this égale a la duree du parcours et this supérieur a la durée du parcours.

Identification du test : compareTo()	Version : 1.0
Description du test :	Test unitaire sur la méthode "compareTo()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	duree	Résultat attendu
P0	this.duree>parparcours.duree	>0
P1	this.duree=parparcours.duree	=0
P2	this.duree<parparcours.duree	<0

Classe	duree	Résultat attendu
P0	this = 20 par = 10	10
P1	this = 20 par = 20	=0
P2	this= 10 par = 20	-10

retour au sommaire

METHODE "quetesPossibles()" :

- Choix de la partition : On regarde les quêtes réalisables et on regarde s'il sont rajouter dans les quêtes possibles

Identification du test : quetesPossibles()	Version : 1.0

Description du test :	Test unitaire sur la méthode "quetesPossibles()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	nbQuetesRéalisables	Résultat attendu
P0	0	ensemble ne change pas
P1	1	ensemble ajoute la dite quete
P2	plusieurs	ensemble ajoute les dites quetes

queteFaite = 1,Quete1 quetePossible = (Quete2) ensQueteNonFaite = (Quete3,Quete4,Quete2)

Classe	nbQuetesRéalisables	Résultat attendu
P0	0	(Quete2)
P1	(Quete4)	(Quete2,Quete4)
P2	(Quete4,Quete3)	(Quete2,Quete4,Quete3)

retour au sommaire

- Choix de la partition : On ajoute le temps de la quete en terme de distance par rapport a la quete precedente et de durée.

METHODE "ajouteDuree()" :

Identification du test : ajouteDuree()	Version : 1.0
Description du test :	Test unitaire sur la méthode "ajouteDuree()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	chQuetesFaites	Résultat attendu
P0	ensemble vide	parquete(pos y + pos x + duree parquete)
P1	possède une quete actuelle	abs(parquete(actuelle(pos x + pos y)-ancienne(pos x + pos y) + duree parquete))

Parquete position = (2,2) Parquete duree = 10 Queteactuelle position = (4,1) Ancienne duree = 10

Classe	chQuetesFaites	Résultat attendu
P0	ensemble vide	14
P1	possède quete actuelle	23

retour au sommaire

METHODE "getQueteActuelle()" :

- Choix de la partition : recupère un entier positif lorsque une quete est prise.

Identification du test : getQueteActuelle()	Version : 1.0
Description du test :	Test unitaire sur la méthode "getQueteActuelle()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	chQueteFaite	Résultat attendu
P0	ensemble vide	IDQueteActuelle = -10
P1	possède un ou plusieurs quètes	IDQueteActuelle >= 0

Classe	chQueteFaite	Résultat attendu
P0	ensemble vide	-10
P1	chQueteFaite = {1.Q1}	1

retour au sommaire

METHODE "queteFinPossibleExhaustive()" :

- Choix de la partition : Analyse en fonction de l'experience si la quete est possible. L'expérience est divisé en 3 partition d'équivalence. La situation ou l'expérience actuelle est supérieur a la quete de fin. Celle ou elle est égale ou celle ou elle est inférieure. Elle regarde si la quete est possible ou non et regarde si le nombre de quete du scénario moins un est supérieur ou égale au nombre de quete deja faite.

Identification du test :	Version : 1.0

queteFinPossibleExhaustive()	
Description du test :	Test unitaire sur la méthode "queteFinPossibleExhaustive()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	NbQueteFaite	Quete Possible	Experience	Résultat Attendu
P0	(NbQueteScenario-1)=NbQueteFaite	True	pfinexp>=qfinexp	True
P1	(NbQueteScenario-1)=NbQueteFaite	True	pfinexp<qfinexp	False
P2	(NbQueteScenario-1)>NbQueteFaite	True	pfinexp>=qfinexp	False
P3	(NbQueteScenario-1)>NbQueteFaite	True	pfinexp<qfinexp	False
P4	(NbQueteScenario-1)>NbQueteFaite	False	pfinexp>=qfinexp	False
P5	(NbQueteScenario-1)>NbQueteFaite	False	pfinexp<qfinexp	False

Classe	NbQueteFaite	Quete Possible	Experience	Résultat Attendu
P0	4=NbQueteFaite = NbQueteScenario, NbQueteScenario = 5	True	pfinexp = 350,qfinexp = 350	True
P1	4=NbQueteFaite = NbQueteScenario, NbQueteScenario = 5	True	pfinexp = 300,qfinexp = 350	False
P2	5=NbQueteScenario,3 = NbQueteFaite	False	pfinexp = 350,qfinexp = 350	False
P3	5=NbQueteScenario,3 = NbQueteFaite	True	pfinexp = 300,qfinexp = 350	False

Classe	NbQueteFaite	Quete Possible	Experience	Résultat Attendu
P4	5=NbQueteScenario,3 = NbQueteFaite	False	pfinexp = 350,qfinexp = 350	False
P5	5=NbQueteScenario,3 = NbQueteFaite	False	pfinexp = 300,qfinexp = 350	False

retour au sommaire

METHODE "getCle()" :

- Choix de la partition : parComparatif est un string. On peut le partionné en 3. Soit parCompartif est égal à "duree", soit "nbQuete" ou soit "deplacements".

Identification du test : getCle()	Version : 1.0
Description du test :	Test unitaire sur la méthode "getCle()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	parComparatif	Résultat attendu
P0	parcomparatif="duree "	this.duree
P1	parcomparatif="nbQuete "	this.quetefaites.size()
P2	parcomparatif ="deplacements"	this.deplacements

parcours.duree = 30 parcours.deplacements = 20 parcours.quetesfaites.size() = 5

Classe	parComparatif	Résultat
P0	parcomparatif="duree "	30
P1	parcomparatif="nbQuete "	5
P2	parcomparatif ="deplacements"	20

retour au sommaire

METHODE "compareTo()" :

- Choix de la partition : nbduree ,nbQueteFaite, nbDeplacements sont tous des entiers positifs. durée peut etre partionné en 3, this inférieur a la durée du parcours, this égale a la duree du parcours et this supérieur a la durée du parcours.(meme chose pour nbQueteFaite et Deplacements) et pour parComparatif c'est un string. Il peut etre partionné en 3: Soit parCompartif est égal à "duree", soit "nbQuete" ou soit "deplacements".

Identification du test : compareTo()	Version : 2.0
Description du test :	Test unitaire sur la méthode "compareTo()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	nbduree	nbQueteFaite	
P0	this.duree>parparcours.duree	ensemble de la partition	
P1	this.duree=parparcours.duree	ensemble de la partition	
P2	this.duree<parparcours.duree	ensemble de la partition	
P3	ensemble de la partition	this.nbQueteFaite>parparcours.nbQueteFaite	
P4	ensemble de la partition	this.nbQueteFaite=parparcours.nbQueteFaite	
P5	ensemble de la partition	this.nbQueteFaite<parparcours.nbQueteFaite	
P6	ensemble de la partition	ensemble de la partition	this.nbD
P7	ensemble de la partition	ensemble de la partition	this.nbD
P8	ensemble de la partition	ensemble de la partition	this.nbD

Classe	nbduree	nbQueteFaite	nbDeplacement	parComparatif	Résultat attendu
P0	this = 20 par = 10	this = 2 par = 1	this = 10 par = 5	duree	10
P1	this = 20 par = 20	this = 2 par = 2	this = 10 par = 10	duree	0

Classe	nbduree	nbQueteFaite	nbDeplacement	parComparatif	Résultat attendu
P2	this = 10 par = 20	this = 1 par = 2	this = 5 par = 10	duree	-10
P3	this = 20 par = 10	this = 2 par = 1	this = 10 par = 5	nbQuete	1
P4	this = 20 par = 20	this = 2 par = 2	this = 10 par = 10	nbQuete	0
P5	this = 10 par = 20	this = 1 par = 2	this = 5 par = 10	nbQuete	-1
P6	this = 20 par = 10	this = 2 par = 1	this = 10 par = 5	deplacements	5
P7	this = 20 par = 20	this = 2 par = 2	this = 10 par = 10	deplacements	0
P8	this = 10 par = 20	this = 1 par = 2	this = 5 par = 10	deplacements	-5

retour au sommaire

METHODE "ajouterDeplacement()" :

- Choix de la partition : Il y a deux partitions, soit ChQueteFaites est vide soit elle possède une quete actuelle.

Identification du test : ajouterDeplacement()	Version : 1.0
Description du test :	Test unitaire sur la méthode "ajouterDeplacement()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	chQuetesFaites	Résultat attendu
P0	ensemble vide	parquete(pos y + pos x)
P1	possède une quete actuelle	abs(parquete(actuelle(pos x + pos y)-ancienne(pos x + pos y)))

Parquete position = (2,2) Queteactuelle position = (4,1) ancien dépalcement = 4

Classe	chQuetesFaites	Résultat attendu
P0	ensemble vide	4
P1	possède quete actuelle	7

retour au sommaire

CLASSE "Classement.java" :

METHODE "ajout()" :

- Choix de la partition : Les clés ne sont que des entiers positifs. Et on peut les partitionnés en deux, soit la clé du parcours est deja dans le classement sois il n'est pas déja présent.

Identification du test : ajout()	Version : 1.0
Description du test :	Test unitaire sur la méthode "ajout()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	ContainsKey	Résultat Attendu
P0	True	ajout parcours
P1	False	création d'une nouvelle liste avec une nouvelle clé et le parcours

clé_parcours = 20

Classe	ContainsKey	Résultat Attendu
P0	clé_parcours dans le classement	Ajout du parcours dans la clé 20
P1	clé_parcours n'est pas dans le classement	Création d'une nouvelle liste avec la clé et le parcours

retour au sommaire

METHODE "afficherClassement()" :

- Choix de la partition : un ou des parcours qui on la plus petite clé

Identification du test : afficherClassement()	Version : 1.0
Description du test :	Test unitaire sur la méthode "afficherClassement()" en utilisant la

	méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	parcours	Résultat Attendu
P0	un parcours possède la plus petite	affiche un seul parcours avec la clé la plus petite
P1	plusieurs parcours possède la plus petite	affiche tout les parcours avec la clé la plus petite

- **/!\ abscence de tableau de données, ceci est un test visuel! /!**

retour au sommaire

METHODE "afficherClassement()" :

- Choix de la partition : un ou des parcours qui on la plus petite clé. parchoix == 1 ou 2.

Identification du test : afficherClassement()	Version : 2.0
Description du test :	Test unitaire sur la méthode "afficherClassement()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	parcours	parchoix	nbSolution	Résultat Attendu
P0	un parcours possède la plus petite	1	nbsolution<=parcourAfficher	affiche un seul parcours avec la clé la plus petite
P1	plusieurs parcours possède la plus petite	1	nbsolution<=parcourAfficher	affiche tout les parcours avec la clé la plus petite

Classe	parcours	parchoix	nbSolution	Résultat Attendu
P2	un parcours possède la plus grande clé	2	nbsolution<=parcourAfficher	affiche un seul parcours avec la clé la plus grande
P3	plusieurs parcours possède la plus grande clé	2	nbsolution<=parcourAfficher	affiche tout les parcours avec la clé la plus grande
P4	ensemble de la partition	ens. de la p.	nbsolution>parcourAfficher	affiche le ou les parcours mais lève une exception

- **/!\ abscence de tableau de données, ceci est un test visuel! /!**

retour au sommaire

METHODE "regulationClassement()" :

- Choix de la partition : SolutionParcours est un int positif comme nbSolutionsTotale. SolutionParcours est partionné en 2 : soit SolutionTotale <parsolutions ou SolutionTotale =parsolution; Le parChoix est un int et peut etre partionné en 2: parChoix = 1 (meilleurs choix) ou parChoix = 2 (pires choix).parcoursgetCle est un int positif. il peut etre partionné en 2 : parcours est plus petit que un autre parcours dans le Classement ou soit parcours est plus grand que un autre parcours dans le Classement

Identification du test : regulationParcours()	Version : 1.0
Description du test :	Test unitaire sur la méthode "regulationParcours()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	solutionsTotale	par choix	parcours.getCle	Résultat Attendu

---

Classe	solutionsTotale	par choix	parcours.getCle	Résultat Attendu
P0	SolutionTotale <parsolutions	peu importe	peu importe	classementajouteparcours
P1	SolutionTotale =parsolutions	1	parcours est plus petit que un autre parcours dans le Classement	Classement ajoute parcours et vire un parcours moins bon
P2	SolutionTotale =parsolutions	1	parcours est plus grand que un autre parcours dans le Classement	Classement ne change pas
P3	SolutionTotale =parsolutions	2	parcours est plus petit que un autre parcours dans le Classement	Classement e change pas
P4	SolutionTotale =parsolutions	2	parcours est plus grand que un autre parcours dans le Classement	Classement ajoute parcours et vire un parcours meilleure

parcours dans le classement se nommera parcoursClass

Classe	solutionsTotale	par choix	parcours.getCle	Résultat Attendu
P0	SolutionTotale = 2 par solution = 1	2	parcours.getCle = 10 parcoursClass = 8	classement.ajout(Parcours)
P1	SolutionTotale = 1 par solution = 2	1	parcours.getCle = 8 parcoursClass = 10	Classement ajoute parcours et vire un parcours moins bon
P2	SolutionTotale = 1 par solution = 2	1	parcours.getCle = 10 parcoursClass = 8	Classement ne change pas
P3	SolutionTotale = 1 par solution = 2	2	parcours.getCle = 8 parcoursClass = 10	Classement ne change pas
P4	SolutionTotale = 1 par solution = 2	2	parcours est plus grand que un autre parcours dans le Classement	parcours.getCle = 10 parcoursClass = 8

CLASSE "Algorithme.java" :

METHODE "solutionGloutonneEfficace()" :

- Choix de la partition : Il y a deux partitions, soit il y a plusieurs possibilités de parcours glouton, soit il n'y en a qu'un.

Identification du test : solutionGloutonneEfficace()	Version : 1.0
Description du test :	Test unitaire sur la méthode "solutionGloutonneEfficace()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Scenario	Résultat attendu
P0	Plusieur possibilités de parcours glouton	Parcours
P1	Qu'une seule possibilité de parcours glouton	Parcours distance minimal

Classe	Scenario	Résultat attendu
P0	Scenario0	Parcours(1,2,4,3,0)
P1	Scenario1	parcours(1,4,3,2,5,0)

retour au sommaire

METHODE "solutionGloutonneExhaustive()" :

- Choix de la partition : Il y a deux partitions, soit il y a plusieurs possibilités de parcours glouton, soit il n'y en a qu'un.

Identification du test : solutionGloutonneExhaustive()	Version : 1.0
Description du test :	Test unitaire sur la méthode "solutionGloutonneExhaustive()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Scenario	Résultat attendu
P0	Plusieur possibilités de parcours gloutonExhaustive	Parcours
P1	Qu'une seule possibilité de parcours gloutonExhaustive	Parcours distance minimal

---

Classe	Scenario	Résultat attendu
P0	Scenario0	Parcours(1,2,4,3,0)
P1	Scenario1	Parcours(1,4,3,2,5,0)

retour au sommaire

METHODE "solutionSpeedrunEfficace()" :

- Choix de la partition : Pour tous les autres méthodes de SolutionsAlgorithme . On va considérer qu'il n'a pas vraiment de partition (Ce qui est vrai avec les scénarios donnés) . On va vérifier grace aux résultats de David Auger si à chaque Algorithme. La méthode trouve la valeur minimale (durée, déplacements , nbQuete) et la valeur maximale sur les 5 premiers Scenarios. Donc il n'y aura pas de premiers tableaux de partitions on passera directement au tableaux de données. (Désolé on pas trouvé d'autres façon).

Identification du test : solutionSpeedrunEfficace()	Version : 1.0
Description du test :	Test unitaire sur la méthode "solutionSpeedrunEfficace()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Scenario	Résultat
P0	scenario 0	minduree = 27 , maxduree = 30
P1	scenario 1	minduree = 34 , maxduree = 40
P2	scenario 2	minduree = 80 , maxduree = 106
P3	scenario 3	minduree = 53 , maxduree = 72
P4	scenario 4	minduree = 95 , maxduree =167

retour au sommaire

METHODE "SolutionSpeedrunExhaustive()" :

- Choix de la partition : Voir ici

Identification du test : SolutionSpeedrunExhaustive()	Version : 1.0

Description du test :	Test unitaire sur la méthode "SolutionSpeedrunExhaustive()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Scenario	Résultat
P0	scenario 0	minduree = 36 , maxduree = 40
P1	scenario 1	minduree = 34 , maxduree = 40
P2	scenario 2	minduree = 91 , maxduree = 117
P3	scenario 3	minduree = 64 , maxduree = 74
P4	scenario 4	minduree = 115 , maxduree =171

retour au sommaire

METHODE "SolutionEfficaceNbQuete()" :

- Choix de la partition : Voir ici

Identification du test : SolutionEfficaceNbQuete()	Version : 1.0
Description du test :	Test unitaire sur la méthode "SolutionEfficaceNbQuete()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Scenario	Résultat
P0	scenario 0	minnbQuete = 4 , maxnbQuete = 4
P1	scenario 1	minnbQuete= 5 , maxnbQuete = 6
P2	scenario 2	minnbQuete = 9 , maxnbQuete= 9
P3	scenario 3	minnbQuete = 6 , maxnbQuete = 8
P4	scenario 4	minnbQuete = 7 , maxnbQuete =10

retour au sommaire

- Choix de la partition : Voir ici

METHODE "SolutionExhaustiveNbQuete()" :

Identification du test : SolutionExhaustiveNbQuete()	Version : 1.0
Description du test :	Test unitaire sur la méthode "SolutionExhaustiveNbQuete()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Scenario	Résultat
P0	scenario 0	minnbQuete = 5 , maxbQuete = 5
P1	scenario 1	minnbQuete= 6 , maxnbQuete = 6
P2	scenario 2	minnbQuete = 10 , maxnbQuete= 10
P3	scenario 3	minnbQuete = 8 , maxnbQuete = 8
P4	scenario 4	minnbQuete = 10 , maxnbQuete =10

retour au sommaire

METHODE "SolutionDeplacementEfficace()" :

- Choix de la partition : Voir ici

Identification du test : SolutionDeplacementEfficace()	Version : 1.0
Description du test :	Test unitaire sur la méthode "SolutionDeplacementEfficace()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Scenario	Résultat
P0	scenario 0	minDeplacements = 14 , maxDeplacements = 20
P1	scenario 1	minDeplacements= 17, maxDeplacements = 23
P2	scenario 2	minDeplacements = 35 , maxDeplacements= 57

Classe	Scenario	Résultat
P3	scenario 3	minDeplacements = 26 , maxDeplacements = 36
P4	scenario 4	minDeplacements = 49 , maxDeplacements =107

retour au sommaire

METHODE "SolutionDeplacementExhaustive()" :

- Choix de la partition : Voir ici

Identification du test : SolutionDeplacementExhaustive()	Version : 1.0
Description du test :	Test unitaire sur la méthode "SolutionDeplacementExhaustive()" en utilisant la méthode des boites noires.
Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	Scenario	Résultat
P0	scenario 0	minDeplacements = 20 , maxDeplacements = 24
P1	scenario 1	minDeplacements= 17, maxDeplacements = 23
P2	scenario 2	minDeplacements = 39 , maxDeplacements= 65
P3	scenario 3	minDeplacements = 28 , maxDeplacements = 38
P4	scenario 4	minDeplacements = 55 , maxDeplacements =111

retour au sommaire

METHODE "choixAlgo()" :

- Choix de la partition : la partition est que le type de solution est soit Efficace ou exhaustive, donc soit 1 soit 2, puis l'objectif est soit de speedrun soit le deplacement et soit le nombre de quete, respectivement 1, 2 ou 3.

Identification du test : choixAlgo()	Version : 1.0
Description du test :	Test unitaire sur la méthode "choixAlgo()" en utilisant la méthode des boites noires.

Ressources requises :	IntelliJ IDEA 2022.3.2, la machine cité plus haut
Responsable de la campagne de test :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE

Classe	typesolution	objectifsolution	Résultat
P0	1	1	solutionSpeedrunEfficace(parScenario)
P1	1	2	solutionDeplacementEfficace(parScenario)
P2	1	3	solutionEfficaceNbQuete(parScenario)
P3	2	1	solutionSpeedrunExhaustive(parScenario)
P4	2	2	solutionDeplacementExhaustive(parScenario)
P5	2	3	solutionExhaustiveNbQuete(parScenario)

retour au sommaire

3.3. Résultats

CLASSE "Quete.java" :

METHODE "extraitPrecond()" :

Identification du test :	extraitPrecond()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	18/05/2023
Résultat :	OK
Occurences des résultats :	Systèmatique

retour au sommaire

METHODE "extraitPos()" :

Identification du test :	extraitPos()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	18/05/2023
Résultat :	OK
Occurences des résultats :	Systèmatique

retour au sommaire

METHODE "testprecondition()" :

Identification du test :	testprecondition()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	18/05/2023
Résultat :	OK
Occurences des résultats :	Systèmatique

retour au sommaire

METHODE "distanceQuete()" :

Identification du test :	distanceQuete()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	18/05/2023
Résultat :	OK
Occurences des résultats :	Systèmatique

retour au sommaire

METHODE "queteProche()" :

Identification du test :	queteProche()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	18/05/2023
Résultat :	OK
Occurences des résultats :	Systèmatique

retour au sommaire

METHODE "quetePossible()" :

Identification du test :	quetePossible()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	18/05/2023

Résultat :	OK
Occurences des résultats :	Systèmatique

retour au sommaire

METHODE "compareTo()" :

Identification du test :	compareTo()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	20/05/2023
Résultat :	OK
Occurences des résultats :	Systèmatique

retour au sommaire

CLASSE "Parcours.java" :

METHODE "extraitQuete()" :

Identification du test :	extraitQuete()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	20/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "queteFinPossibleEfficace()" :

Identification du test :	queteFinPossibleEfficace()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	20/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "compareTo()" :

Identification du test :	compareTo()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	23/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "quetesPossibles()" :

Identification du test :	quetesPossibles()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	24/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "ajouteDuree()" :

Identification du test :	ajouteDuree()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	24/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "getQueteActuelle()" :

Identification du test :	getQueteActuelle()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	26/05/2023
Résultat :	OK

Occurences des résultats :	Systématique

retour au sommaire

METHODE "queteFinPossibleExhaustive()" :

Identification du test :	queteFinPossibleExhaustive()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	26/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "getCle()" :

Identification du test :	getCle()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	30/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "compareTo()" :

Identification du test :	compareTo()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	30/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "ajouterDeplacement()" :

Identification du test :	ajouterDeplacement()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	31/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

CLASSE "Classement.java" :

METHODE "ajout()" :

Identification du test :	ajout()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	27/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "afficherClassement()" :

Identification du test :	afficherClassement() VERSION 1.0
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	27/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "afficherClassement()" :

Identification du test :	afficherClassement() VERSION 2.0
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	27/05/2023
Résultat :	OK

Occurences des résultats :	Systématique

retour au sommaire

METHODE "regulationClassement()" :

Identification du test :	regulationClassement() VERSION 1.0
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	03/06/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

CLASSE "Algorithme.java" :

METHODE "solutionGloutonneEfficace()" :

Identification du test :	solutionGloutonneEfficace()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	28/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "solutionGloutonneExhaustive()" :

Identification du test :	solutionGloutonneExhaustive()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	28/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "solutionSpeedrunEfficace()" :

Identification du test :	solutionSpeedrunEfficace()

Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	28/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "SolutionSpeedrunExhaustive()" :

Identification du test :	SolutionSpeedrunExhaustive()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	28/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "SolutionEfficaceNbQuete()" :

Identification du test :	SolutionEfficaceNbQuete()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	30/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "SolutionExhaustiveNbQuete()" :

Identification du test :	SolutionExhaustiveNbQuete()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	31/05/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "SolutionDeplacementEfficace()" :

Identification du test :	SolutionDeplacementEfficace()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	01/06/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "SolutionDeplacementExhaustive()" :

Identification du test :	SolutionDeplacementExhaustive()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	01/06/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

METHODE "choixAlgo()" :

Identification du test :	choixAlgo()
Responsable :	Lucas DA SILVA FERREIRA & Baptiste FOURNIE
Date d'application du test :	03/06/2023
Résultat :	OK
Occurences des résultats :	Systématique

retour au sommaire

3.4. Conclusions

Chacune des méthodes des classes du package modéle testé marchent . Ils n’y pas de bugs qui ont été trouvés. Les tests unitaires sont finies.Aucune erreur à été détécté. Nous pouvons passé à la crétion de l'interface utilisateur.