UVSQ université PARIS-SACLAY | IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY

R2.03-QUALITE DE DEVELOPPEMENT

DOSSIER DE TESTS

SAE 2.01 & 2.02

Yassine Elkalki & Mohamed Lamine LY
2022-2023 INF1-C

RPG_Version_IHM	Version 1.0
Document : Dossier de tests	Date : 08/06/2023
Responsable de la rédaction : Yassine Elkalki & Mohamed Lamine LY	

Dossier de tests

1. Introduction

Ce dossier de tests associé au développement des classes du composant modèle du projet RPG_Version_IHM.

Dans ce RPG simplifié, un joueur reçoit une liste de quêtes que l'on appelle scénario. Le but est donc de réaliser la quête finale du scénario. Mais pour cela, il faut avoir rempli des préconditions ainsi qu'avoir accumulé suffisamment de points d'expérience. Il faut donc proposer la meilleure solution pour finir le scénario !

Il est important de tester les différentes fonctionnalités pour s'assurer de son bon fonctionnement. Ce dossier de test vise à évaluer les classes du composant modèle .

Nous allons vérifier les opérations des classes Position, Quête et Solution.

2. Description de la procédure de test

La procédure de test du projet RPG repose sur la méthode de partitionnement de données. Cette méthode vise à diviser l'ensemble des données disponibles en plusieurs sous-ensembles distincts. Cela permet d'évaluer la performance du modèle développé en utilisant des données qui n'ont pas été utilisées lors de son entraînement.

3. Description des informations à enregistrer pour les tests

1. Campagne de test

Définition du contexte des tests en s'appuyant sur le type de tableau suivant :

Produit testé : Produit RPG_Version_IHM	
Configuration logicielle : JUnit 5.9	
Configuration matérielle : Windows 10 et 11	
Date de début : 24/04/2023	Date de finalisation : 08/06/2023
Responsable de la campagne de test : Yassine Elkhalki	

2. Tests

Classe Position (PositionTest) :

Ce test vérifie la fonctionnalité de la méthode `deplacement` de la classe Position en comparant ses résultats avec les réponses attendues. Si tous les tests réussissent, aucun message d'erreurs ne sera affiché dans la console.

Données de tests pour la méthode deplacement():

Cas	Acteurs (paramètres)		Résultat attendue
	Position appelante	**Position parametre**	
Cas 1	Position(0,0)	Position(0,0)	0
Cas 2	Position(0,0)	Position(5,5)	10
Cas 3	Position(0,0)	Position(3,1)	4
Cas 4	Position(5,5)	Position(0,0)	10
Cas 5	Position(5,5)	Position(5,5)	0
Cas 6	Position(5,5)	Position(3,1)	6
Cas 7	Position(3,1)	Position(0,0)	4
Cas 8	Position(3,1)	Position(5,5)	6
Cas 9	Position(3,1)	Position(3,1)	0

Les deux tests qui suivent, testent la fonctionnalité des méthodes getX et getY qui sont les accesseurs de la classe.

Données de tests pour la méthode getX():

Cas	Acteurs (paramètres)	Résultat attendue
	Position	
Cas 1	Position(0,0)	0
Cas 2	Position(5,5)	5
Cas 3	Position(3,1)	3

Données de tests pour la méthode getY():

Cas	Acteurs (paramètres)	Résultat attendue
	Position	
Cas 1	Position(0,0)	0
Cas 2	Position(5,5)	5
Cas 3	Position(3,1)	1

Classe Solution (SolutionTest) :

Cette classe contient les méthodes de tests pour les méthode solutionEfficace et solutionExhaustive. Ces tests vérifient partiellement la justesse des solution renvoyées par les méthodes.

Par exemple, on vérifiera pour les deux solutions :

- La bonne présence de la quête finale
- Les longueurs de solutions (= au nombre de quêtes pour la solution exhaustive & <= au nombre de quêtes pour la solution efficace)
- Pour la solution exhaustive, si toutes les quêtes sont bien présentes dans la solution.

Données de tests pour la méthode solutionEfficace():

Cas	Acteurs (paramètres)	Résultat attendue
	Scenario	
Cas 1	scenario 0	longueur : <= 5 quête finale : oui
Cas 2	scenario 1	longueur : <= 6 quête finale : oui
Cas 3	scenario 2	longueur : <= 10 quête finale : oui
Cas 4	scenario 3	longueur : <= 8 quête finale : oui
Cas 5	scenario 4	longueur : <= 10 quête finale : oui

Données de tests pour la méthode solutionEfxhaustive():

Cas	Acteurs (paramètres)	Résultat attendue
	Scenario	
Cas 1	scenario 0	longueur : = 5 quête finale : oui toutes les quêtes : oui
Cas 2	scenario 1	longueur : = 6 quête finale : oui toutes les quêtes : oui
Cas 3	scenario 2	longueur : = 10 quête finale : oui toutes les quêtes : oui
Cas 4	scenario 3	longueur : = 8 quête finale : oui toutes les quêtes : oui
Cas 5	scenario 4	longueur : = 10 quête finale : oui toutes les quêtes : oui

Classe Quete (QueteTest) :

Dans cette classe de test, il y a des méthodes de tests pour la quasi totalité des méthodes. Voici son dossier de test pour les méthodes les plus importantes.

Données de tests pour la méthode estQueteFinale():

Cas	Acteurs (paramètres)	Résultat attendue
	Quete	
Cas 1	`Quete("1|(4, 3)|()|2|100|explorer pic de Bhanborim")`	false
Cas 2	`Quete("0|(1, 1)|((3,4),)|4|350|vaincre Araignée lunaire")}`	true

Données de tests pour la méthode estRealisee():

Cas	Acteurs (paramètres)	Résultat attendue
	Quete	
Cas 1	`Quete("1|(4, 3)|()|2|100|explorer pic de Bhanborim").setChRealisee(true)`	true
Cas 2	`Quete("0|(1, 1)|((3,4),)|4|350|vaincre Araignée lunaire")}`	false

4. Conclusion

En conclusion, le dossier de test utilisant JUnit a été un succès indéniable. Les tests automatisés ont permis de vérifier de manière rigoureuse et efficace le bon fonctionnement de notre application. Grâce à JUnit, nous avons pu détecter et corriger des erreurs avant même leur apparition en production, ce qui a contribué à améliorer la qualité de notre code et à réduire les risques d'anomalies.

L'utilisation de JUnit a également permis d'optimiser notre processus de développement en accélérant les cycles de tests. Les tests automatisés ont été exécutés de manière fiable et cohérente, éliminant ainsi les erreurs humaines et économisant un temps précieux pour notre équipe de développement.

De plus, JUnit a favorisé la collaboration au sein de notre équipe en offrant une plateforme commune pour l'exécution et le partage des tests. Cela a permis une meilleure communication et une compréhension partagée des exigences et des comportements attendus de notre application.

En conclusion, l'intégration de JUnit dans notre processus de développement a considérablement renforcé la robustesse et la qualité de notre code, en réduisant les erreurs et en améliorant notre productivité. Les tests automatisés ont apporté une confiance supplémentaire dans notre application, ce qui a été essentiel pour atteindre nos objectifs et satisfaire les attentes de nos utilisateurs. JUnit est sans aucun doute un outil précieux dans le domaine du test logiciel, et nous continuerons à l'utiliser pour garantir la qualité et la fiabilité de nos futurs projets.