Nom du projet: SAE 2.01	Version: 1.0
Document: Dossier de tests	Date: 22/05/2023
Responsable de la rédaction: Ouvrard Maxence	

Dossier de tests

1. Introduction

Le document suivant à pour but de tester les fonctionnalités principales du niveau 1 de la SAE RPG. Il va se centrer sur les méthodes de la classe Aventure, c'est-à-dire la classe la plus importante du projet.

2. Description de la procédure de test

La classe Java doit être capable de réaliser des opérations et de savoir faire des comparaisons afin de trouver la quête que le joueur doit ensuite réaliser. Pour ce faire, il doit analyser des préconditions, savoir faire des calculs de distance, savoir si la solution que nous souhaitons avoir est efficace ou exhaustive.

3. Description des informations à enregistrer pour les tests

Campagne de test

Produit testé: La classe java Aventure	
Date de début: 22/05/23	Date de finalisation: 07/06/23
Tests à appliquer: calculDeplacement(), niveau1(), choixProchaineQuete(), assezExperimente(), preconditionRemplies() et accesQuete()	
Responsable de la campagne de test: Ouvrard Maxence	

Tests

Identification du test: calculDeplacement()	Version: 1.0
Description du test: Commençons en testant la fonction calculDeplacement(), en vérifiant tous les cas possibles.	
Ressources requises: Le logiciel InteliJ et la bibliothèque junit.	

Responsable: Ouvrard Maxence		
Cas testé		Résultats attendus
P1	avec posFin plus petit que posDebut	un entier positif
P2	avec posFin plus grand que posDebut	un entier positif
P3	avec posFin égal à PosDebut	0

Résultat du test

Référence du test appliqué: calculDeplacement()			
Responsable: Gravier Kylian			
Date de l'application du test: 27/05			
Cas testé		Résultat attendu	Résultat du test
P1			
P2			
P3			

Tests

Identification du test: assezExperimente()		Version: 1.0
Description du test: Nous allons tester la méthode assezExperimente()..		
Ressources requises: Le logiciel InteliJ et la bibliothèque junit.		
Responsable: Ouvrard Maxence		
Cas testé		Résultats attendus
P1	parJoueur.getExperience() < quete.getExp()	false
P2	parJoueur.getExperience() > quete.getExp()	true
P3	parJoueur.getExperie	true

	nce() = quete.getExp()	

Résultats de tests

Référence du test appliqué: assezExperimente()			
Responsable: Gravier Kylian			
Date de l'application du test: 27/05			
Cas testé		Résultat attendu	Résultat de test
P1			
P2			
P3			

Tests

Identification du test: accesQuete()		Version: 1.0
Description du test: Continuons en testant la fonction permettant de vérifier qu'une quete est accessible.		
Ressources requises: Le logiciel InteliJ et la bibliothèque junit.		
Responsable: Gravier Kylian		
Cas testé		Résultats attendus
P1	toutes les preconditions à 0	false
P2	les deux premières preconditions à 0	false
P3	les deux dernières preconditions à 0	false
P4	aucune quête réalisée	false
P5	les préconditions ont été réalisés	true

Résultats de tests

Référence du test appliqué: accesQuete()			
Responsable: Ouvrard Maxence			
Date de l'application du test: 27/05			
Cas testé		Résultat attendu	Résultat du test
P1			
P2			
P3			
P4			
P5			

Tests

Identification du test: preconditionsRemplies()		Version: 1.0
Description du test: Testons la méthode preconditionsRemplies() qui doit permettre de retourner un booleen si toutes les préconditions sont remplies pour une quete donnée.		
Ressources requises: Le logiciel InteliJ et la bibliothèque junit.		
Responsable: Gravier Kylian		
Cas testé		Résultats attendus
P1	pas de précondition	true
P2	quete 0 et préconditions respectées	true
P3	quete 0 et préconditions non respectées	false
P4	quete et préconditions respectées	true
P5	quete et préconditions non respectées	false

Résultats de tests

Référence du test appliqué: preconditionsRemplies()			
Responsable: Ouvrard Maxence			
Date de l'application du test: 27/05			
Cas testé		Résultat attendu	Résultat du test
P1			
P2			
P3			
P4			
P5			

Tests

Identification du test: niveau1()		Version: 1.0
Description du test: Testons la méthode niveau1() qui doit permettre de retourner une solution efficace ou exhaustive.		
Ressources requises: Le logiciel InteliJ et la bibliothèque junit.		
Responsable: Gravier Kylian		
Cas testé		Résultats attendus
P1	parChoix = 1	solution efficace
P2	parChoix = 2	solution exhaustive

Résultats de tests

Référence du test appliqué: niveau1()			
Responsable: Gravier Kylian			
Date de l'application du test: 27/05			
Cas testé		Résultat attendu	Résultat du test
P1			
P2			

Tests

Identification du test: testPrecon()		Version: 1.0
Description du test: Testons la méthode testPrecon() qui se situe dans la classe Quete et qui vérifie si une quête nécessite des préconditions ou non.		
Ressources requises: Le logiciel InteliJ et la bibliothèque junit.		
Responsable: Ouvrard Maxence		
Cas testé		Résultats attendus
P1	pas de preconditions	true
P2	nécessite des préconditions	false

Résultats de tests

Référence du test appliqué: niveau1()			
Responsable: Ouvrard Maxence			
Date de l'application du test: 28/05			
Cas testé		Résultat attendu	Résultat du test
P1			
P2			

4. Conclusion: