Projet Qualité Logiciel

« Java Discrete Behavior Framework Simulator »

Version	Auteur	Revue	Date
0.1	F. Vernier	S. Cimpan	27/03/2020
0.2	F. Vernier	S. Cimpan	28/03/2020
0.3	F. Vernier	S. Cimpan	10/04/2020
0.4	S. Cimpan	F. Vernier	17/04/2020

Description :

Le projet porte sur un simulateur de comportement. Ce dernier permet d'invoquer des méthodes d'objet à des instants définis par des lois de probabilité.

Équipe :

Vous vous organiserez en 2 équipes équilibrées par groupe de TP.

Votre projet comprend 3 grandes tâches :

1. Pise en main du projet – reverse Engineering,
2. Plan de test et compte rendu de test,
3. Reengineering,

qui ne sont pas indépendantes. Afin de vous organiser, je vous donne les dépendances

- le package « discreteBeaviorSimulator » dépend du package « action »
- le package «action» dépend du package « timer »
- pour chaque package le reverse Engineering doit être fait avant les tests
- pour chaque package les tests doivent être fait avant le reengineering.

Contraintes :

- Logiciel & plug-in :
  - Java version 8
  - Eclipse IDE version 2020-03
    - EclEmma version 3.1.3
    - SonnarLint version 5.0
    - ObjectAid version current
  - Bibliothèque
    - JUnit 5
  - Moose
  - jdt2famix
- Normalisation :
  - Architecture Projet
    - Architecture standard de projet Maven sous Eclipse
  - Code Java
    - Respect des normes d'écriture Eclipse
    - Tout élément public doit être associé à un commentaire de type JavaDoc
    - Tout élément privé doit être associé à un commentaire de type Java Classique
      - Variable : commentaire de fin de ligne
      - Attribut : commentaire de fin de ligne
      - Méthode : commentaire multi lignes
  - Test
    - L'ensemble des tests doivent être dans un répertoire dédié
    - Chaque package du projet a son package associé dans le répertoire test
  - Langue
    - L'ensemble code et commentaires doit être en anglais.
  - Documment

Objectif :

1. revese Engineering
    1. réaliser le diagramme de classes avant et après refactoring
    2. réaliser un diagramme de séquence
    3. documenter le code au format javadoc
2. test
    1. définition d'un plan de test
    2. implémentation des tests
    3. exécution des tests et réalisation d'un compte rendu de test
3. Reengineering
    1. augmentation de la qualité du code
    2. refactoring du package « action »

Livrable :

- Rapport qualité fourni par SonarLint avant et après refactoring
- Différentes vues Moose avant et après refactoring mettant en évidence la compréhension structurelle et qualitative du projet.
- La conception UML, les diagrammes de classes avant et après refactoring ainsi qu'un exemple de diagramme de séquence
- Code documenté, refactorisé et intégrant les jeux de test
- Le plan de test et les comptes-rendus de test au format XML et HTML

Évaluation :

Vous serez évalué sur différents critères :

- reverse Engineering : la clarté des diagrammes UML et de la javadoc mettra en évidence votre compréhension du système
- test : la qualité et couverture du plan de test est primordiale, son implémentation peut être partielle
- Re-engineering : l'évolution des métriques de qualité, le diagramme de Classes et le code final mettront en évidence le travail sur ce dernier point

Annexe :