PROJET AOC

Observer asynchrone

Ramadan Soumaila
Waberi Houssein Galib
Master 2 Ingénierie Logicielle
Année universitaire 2017/2018

1 Introduction

Ce projet avait lieu pour le cours d'aoc, consiste à mettre en place le patron de conception Active Object afin de permettre des appels asynchrones sur un modèle synchrone. Pour ce faire, nous avons fait la conception du modèle synchrone et asynchrone puis l'implémentation du patron de conception Active Object qui s'appuie sur la bibliothèque standard oracle.

Repository : https://github.com/ismaelrami/m2il_aoc_observer_asynchrone

2 Choix techniques

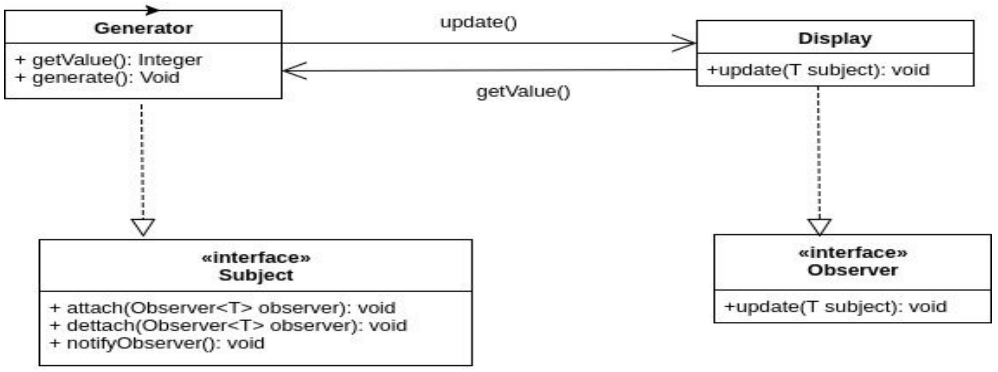
L'IHM de notre programme est crée en utilisant le framework **JavaFX.** Il sépare le code de l'IHM grâce à l'injection de dépendances en utilisant l'architecture MVC.

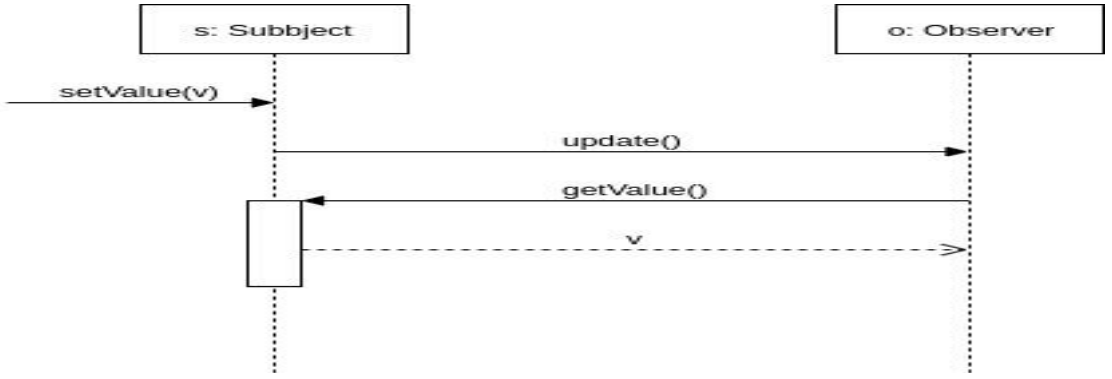
Le fichier fxml pour JavaFX est crée grâce au logiciel **Gluon** qui génère des interfaces et simplifie l'écriture du fichier. Ce logiciel permet également d'associer facilement des controllers aux actions accessible depuis l'interface.

3 Architecture et Conception

3.1 Mode synchrone

Nous avons commencé en toute premier en mode synchrone sans le canal.





3.2 Mode asynchrone

Modèle M1

Le modèle M1 est l'implémentation d'observer asynchrone. Pour ce modèle, le générateur génère un nombre puis notifie son afficheur (DisplayerImpl) pour afficher la valeur dans la console.

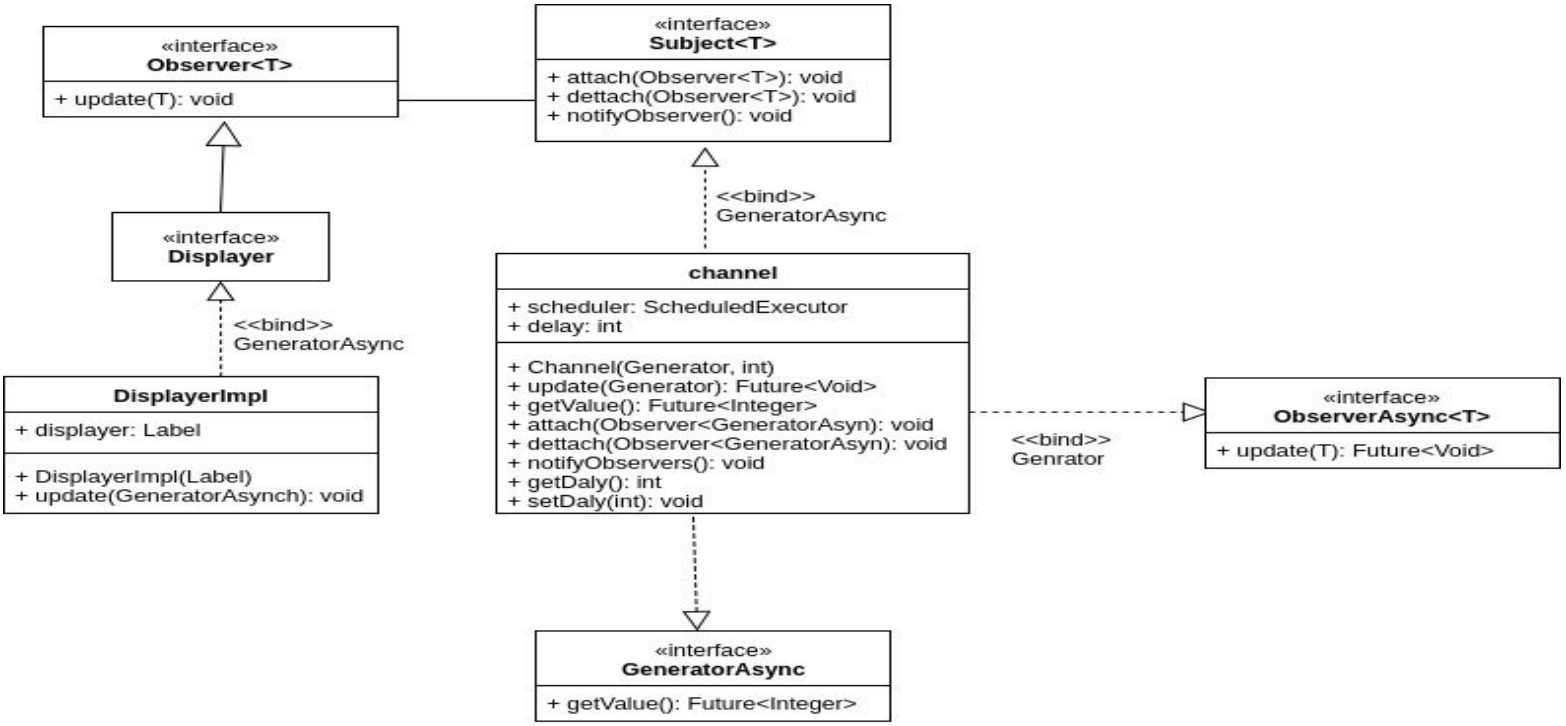


Diagramme de classe du modèle M1

Modèle M3

Ce modèle implémente le patron de conception Active Object en mode observer asynchrone avec opération d'appel synchrone et exécution asynchrone par la mise en oeuvre dans le JDK et Oracle.

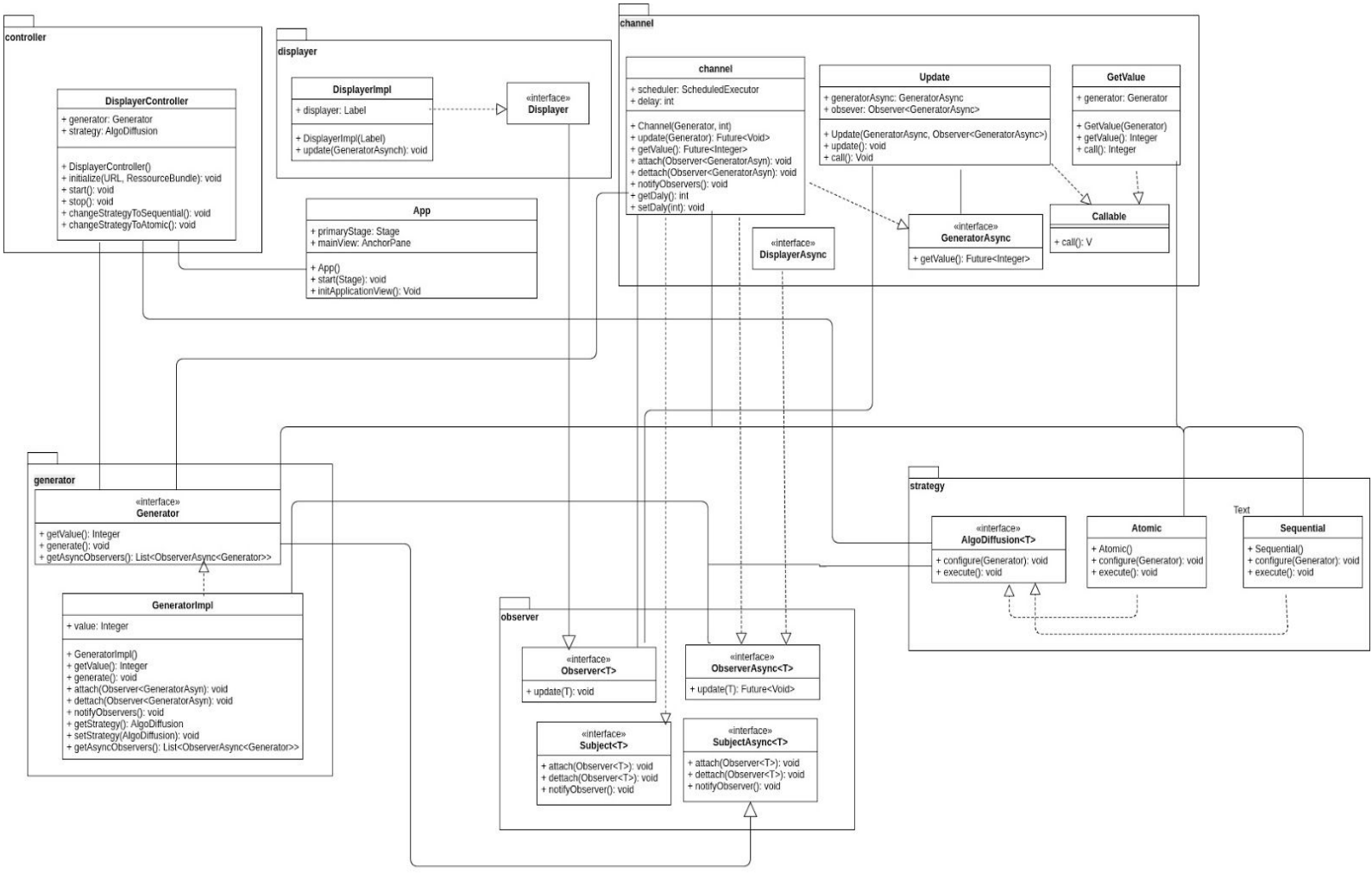


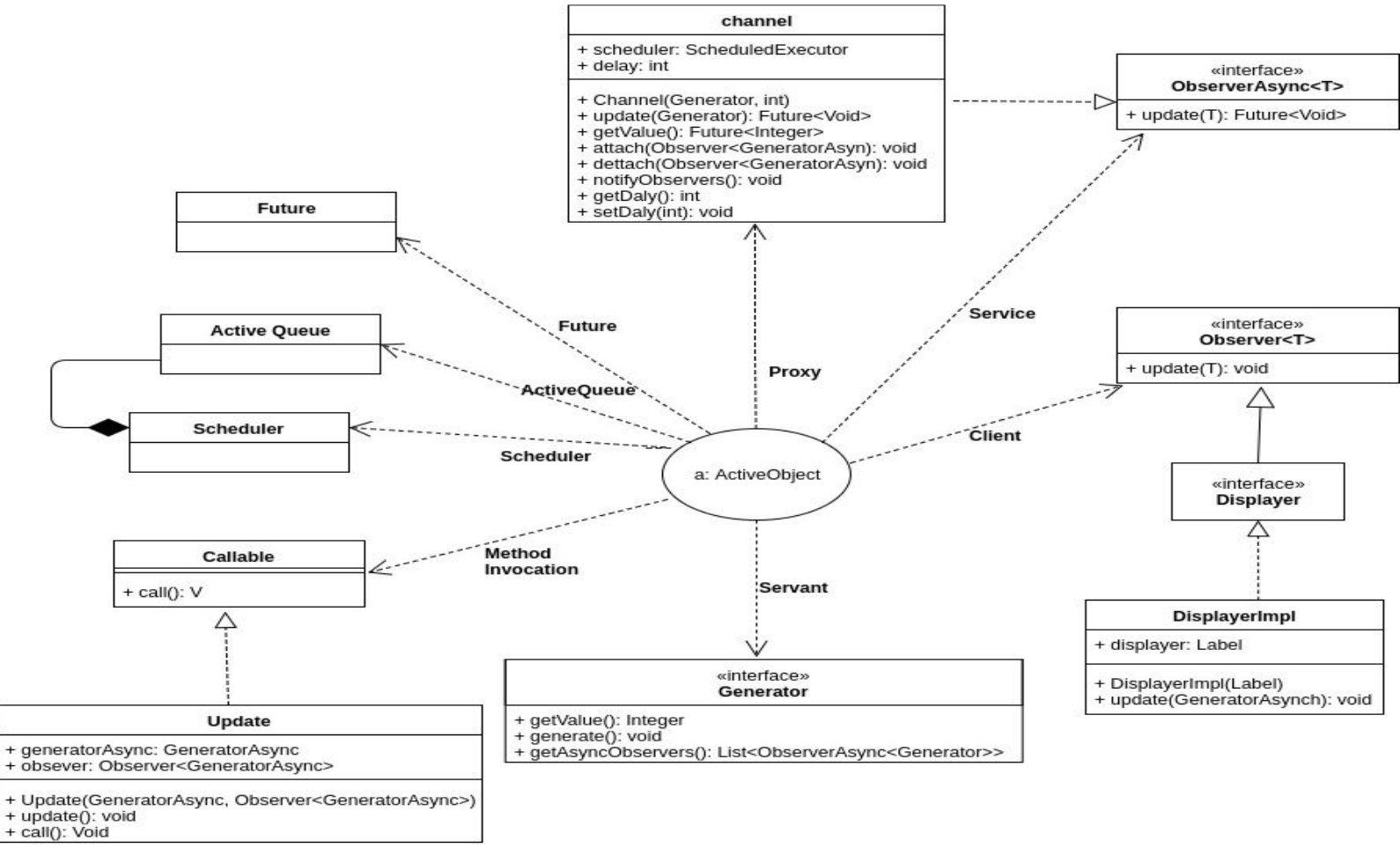
Diagramme de classe du modèle M3

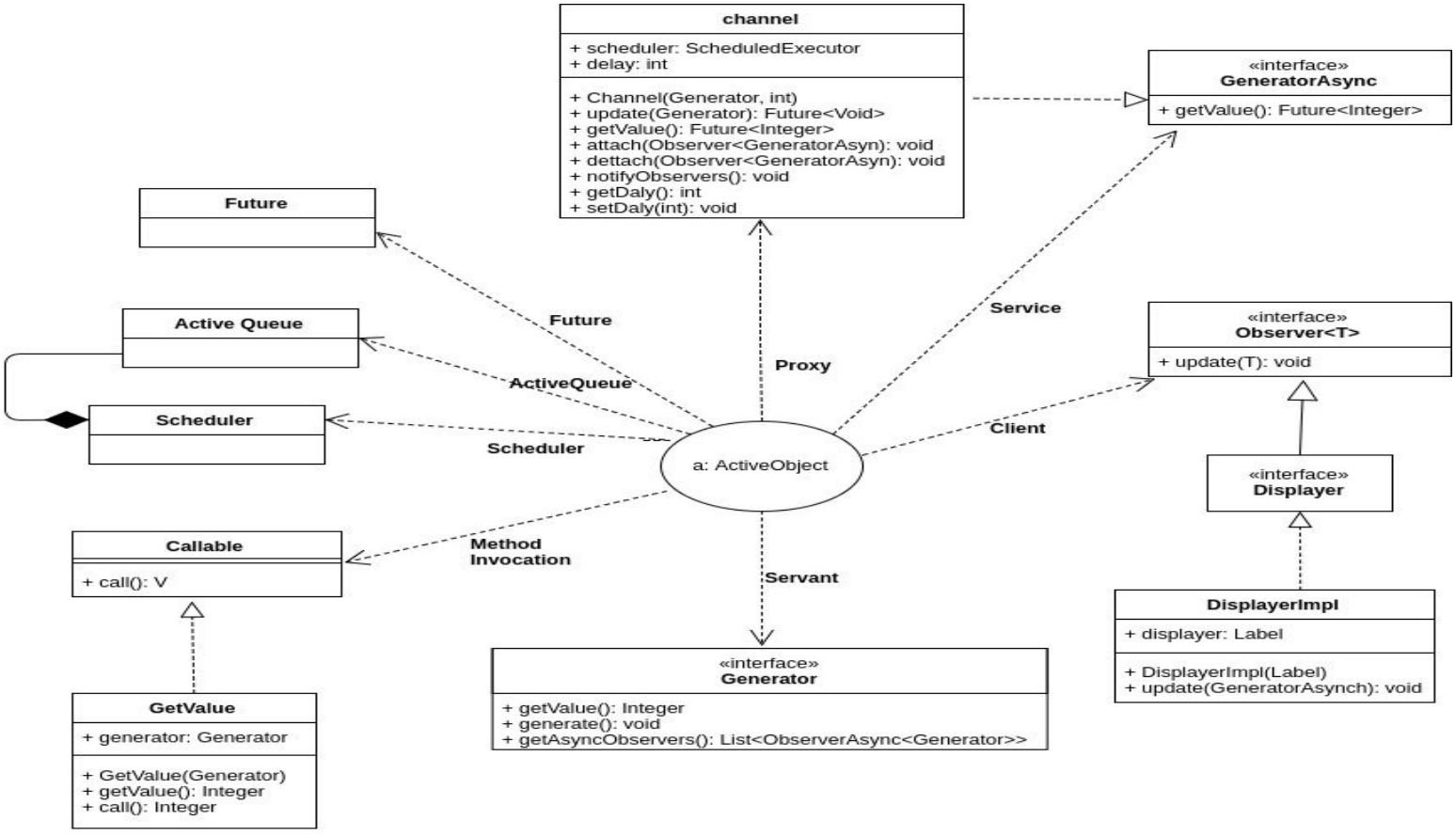
Ce modèle utilise deux algorithmes de diffusion qui sont l'algorithme séquentielle et atomique. Ces deux algorithmes ont pour but de concevoir une stratégie du générateur pour le programme.

Cohérence atomique : tous les observateurs voient la suite complète des valeurs prises par le sujet. Le sujet ne peut faire de traitement que lorsque tous les observateurs ont lu la dernière valeurs.

Cohérence séquentielle : il n'y a pas d'attente, le traitement se fait pas à pas. L'affichage se fait indépendamment pour chaque observateur.

Le patron de conception est utilisé deux fois, car pour les appels asynchrones nous avons besoin dans les deux sens. L'utilisation de la première application du patron est la fonction Update et la deuxième est la fonction GetValue.





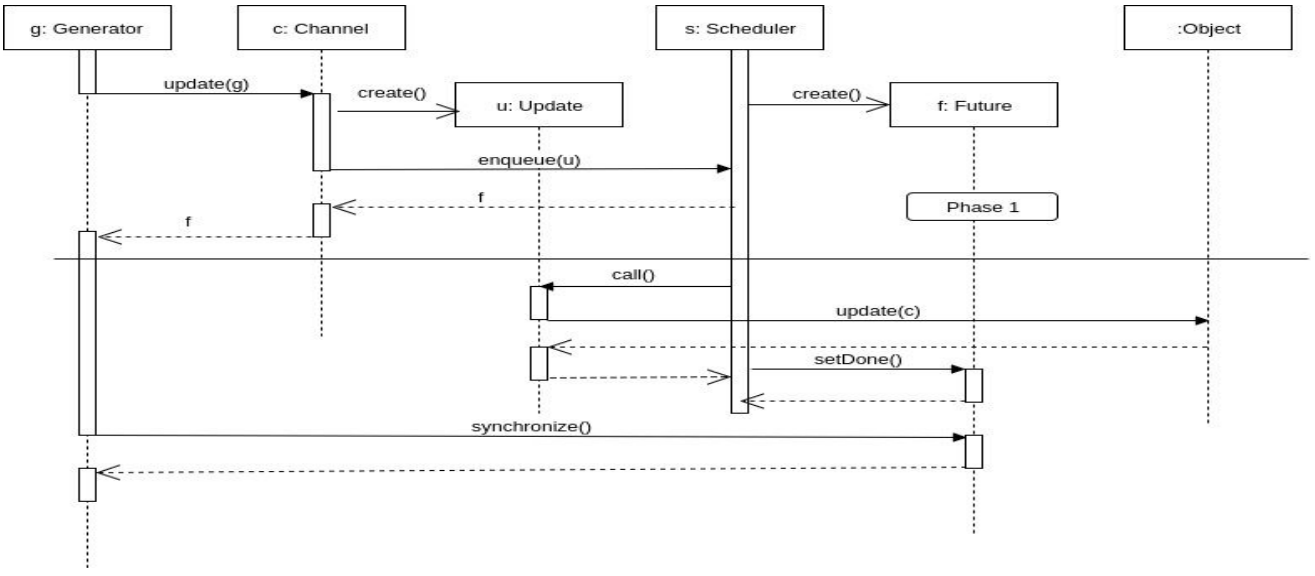


Diagramme de séquence du modèle M3

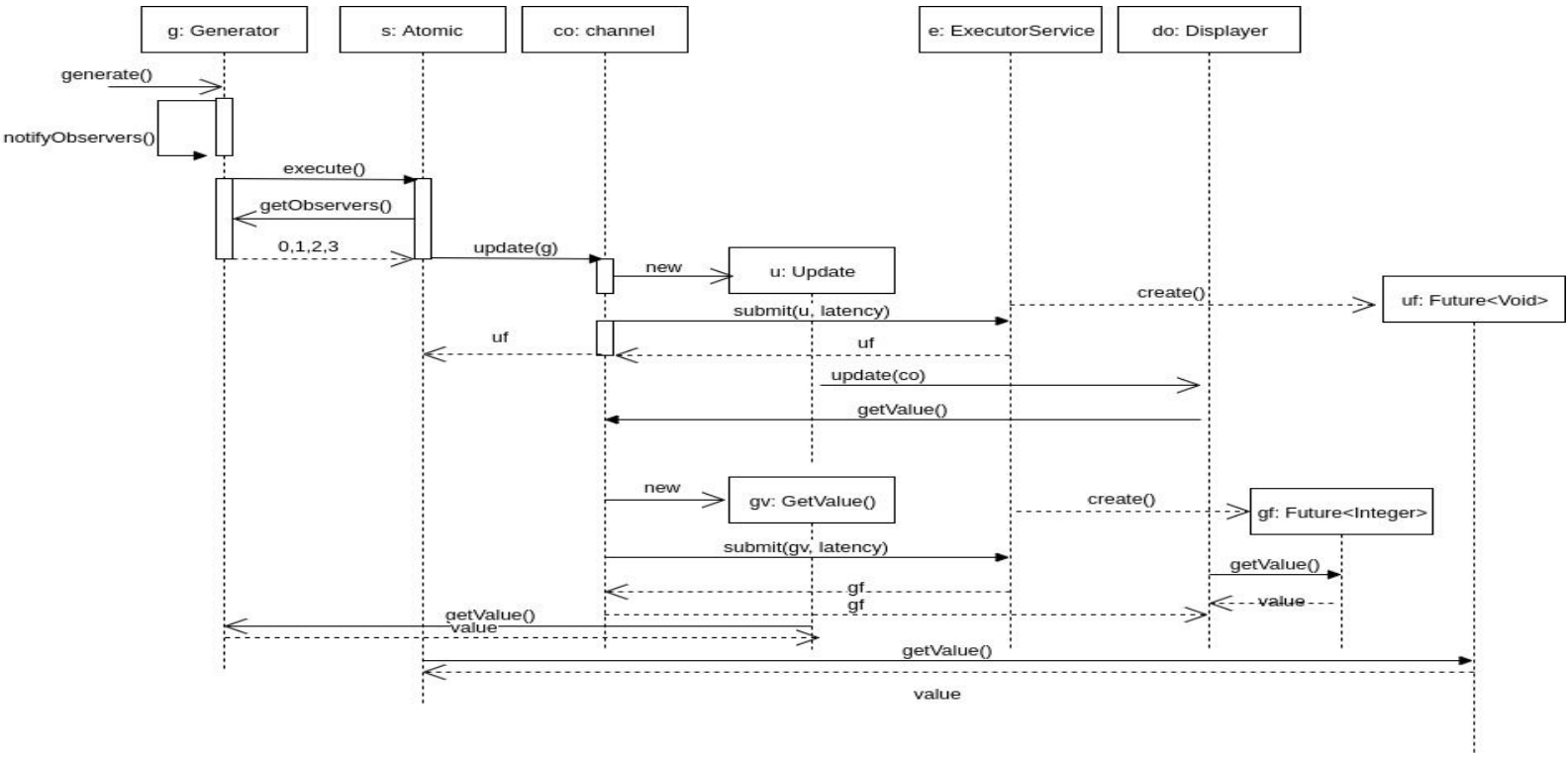
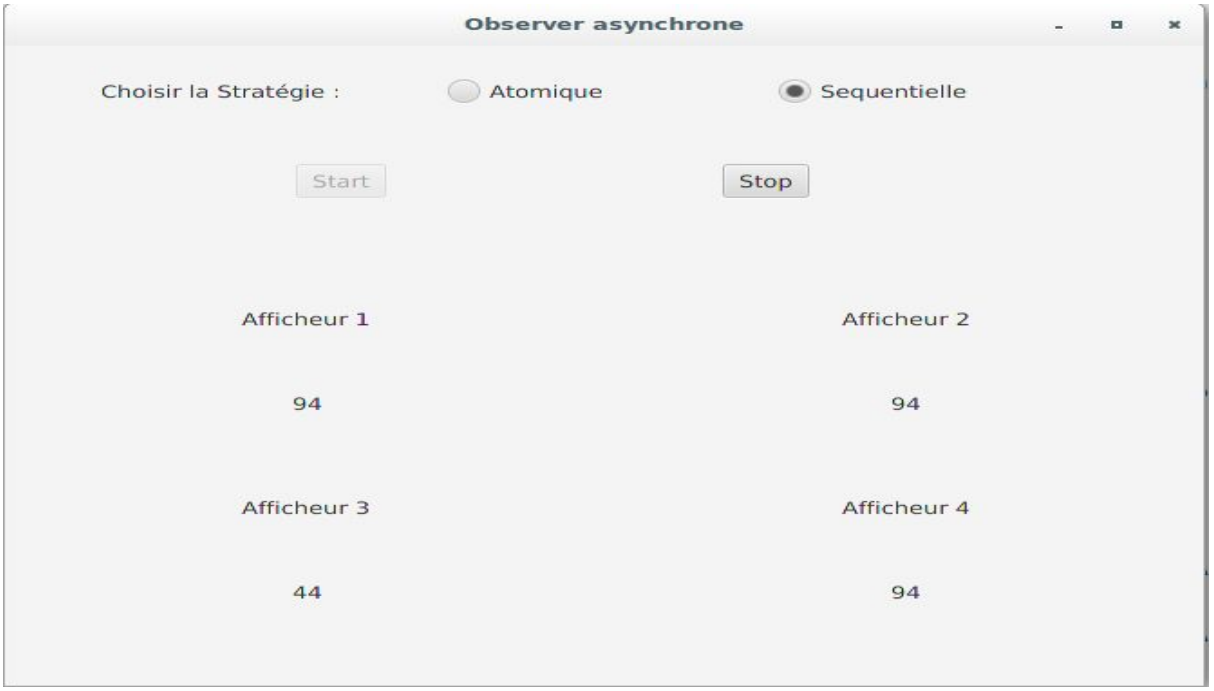


Diagramme de séquence du modèle M3

4 Tests

L'interface de notre application offre à l'utilisateur de choisir la stratégie. Nous avons deux boutons permettant de démarrer et d'arrêter la génération et l'affichage des valeurs. Nous avons aussi quatres afficheurs affichent les valeurs générées aléatoirement selon le choix de l'algorithme.

Exemple d'IHM JavaFX avec la strategie atomique



Exemple d'IHM JavaFX avec la stratégie sequentielle