Rapport AOC

Master 2 Ingénierie Logicielle

June Benvegnu-Sallou
Emmanuel Loisance

Introduction

L'application développée ici permet de mettre en évidence le pattern Active Object. Un générateur de nombre sur 4 afficheurs a été mis en place ainsi que la possibilité de mettre en évidence 2 algorithmes de diffusion des nombres générés. Ces deux algorithmes font appel à :

- La 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 valeur, il y a donc une attente.
- La cohérence séquentielle: il n'y a pas d'attente, le traitement se fait pas à pas. L'affichage se fait indépendamment pour chaque observateur.

L'architecture mise en place consiste à avoir des observateurs asynchrones avec des opérations d'appel synchrone et une exécution asynchrone via le retour d'une promesse par mise en oeuvre du pattern Active Object dans le JDK d'Oracle.

Choix techniques

Nous avons choisi d'utiliser JavaFX pour développer l'application rapidement avec les fichiers ".fxml" et les annotations FXML qui permettent d'accéder facilement aux différents composants de l'application et d'interagir avec eux via un binding.

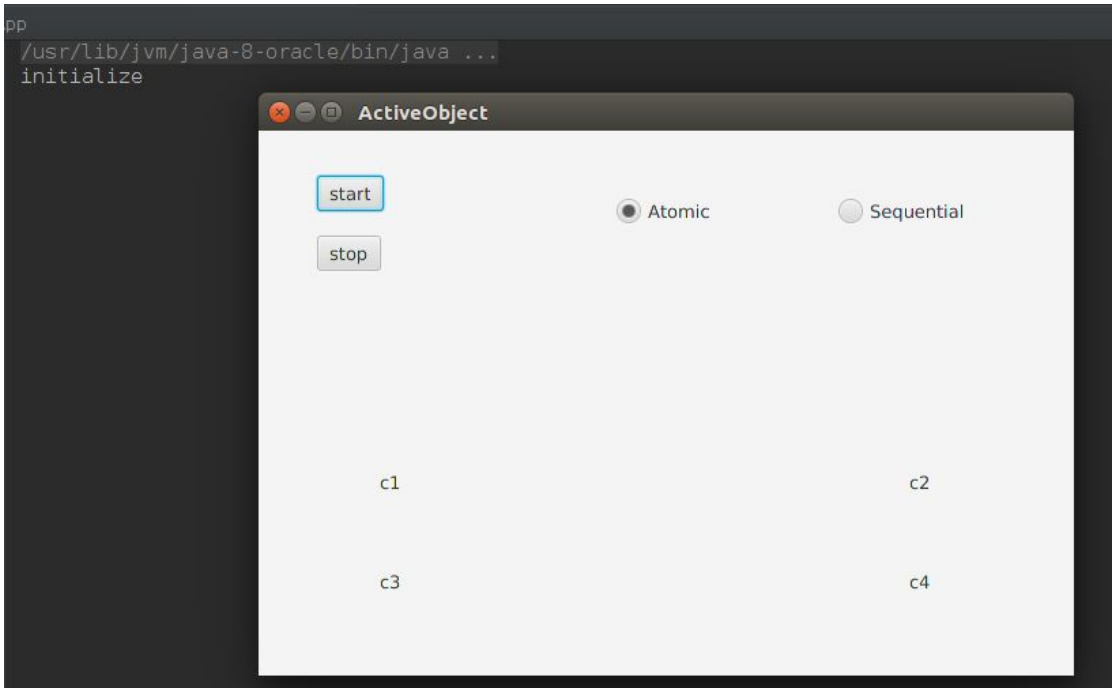
La classe "WindowController" a été créée pour mettre tout le code nécessaire aux fichiers ".fxml" de la fenêtre renseignée dans l'attribut "fx:controller". La classe implémente l'interface "Initializable" qui permet de lancer les traitements une fois que les différents composants nécessaires à JavaFX soient correctement chargés.

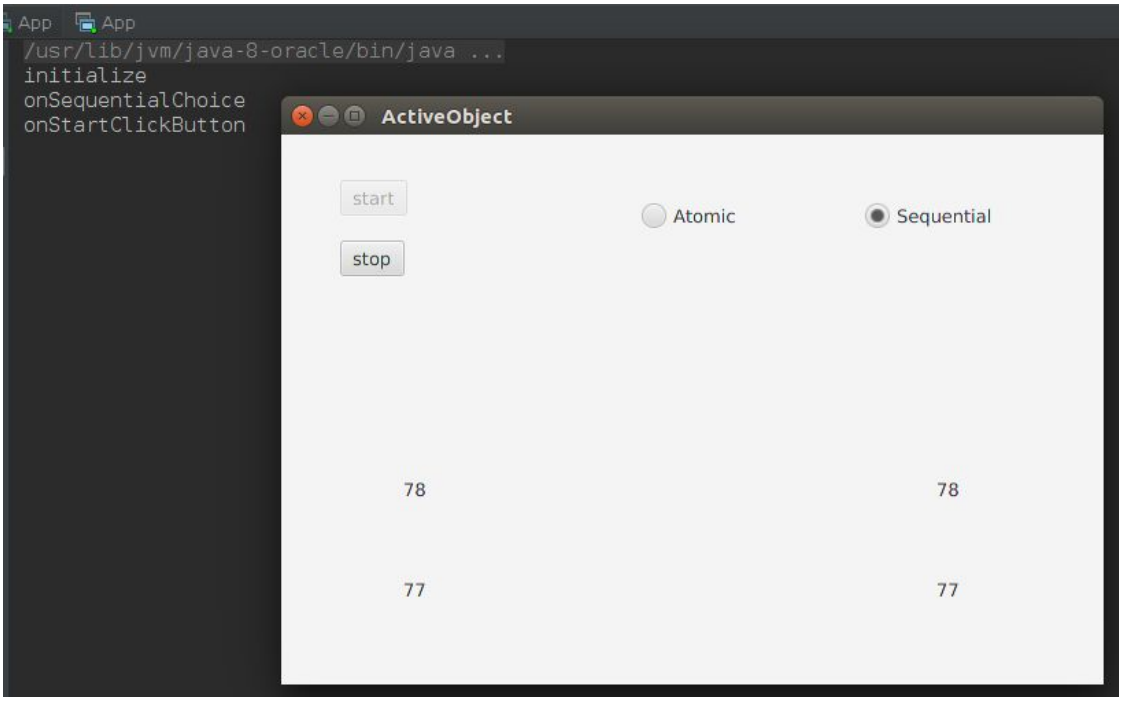
L'interface modélisée est simple, un bouton radio pour chaque algorithme de diffusion, un bouton pour démarrer et un bouton pour stopper le générateur et enfin un label pour chaque afficheur. Au lancement de l'application, l'algorithme par défaut est "Atomic", il est alors possible de démarrer le générateur avec le bouton "start". Le bouton est ensuite grisé jusqu'à ce que le bouton "stop" soit cliqué. Des possibilités d'améliorations pour l'interface sont possibles et envisageables mais non réalisées par manque de temps.

Difficultés

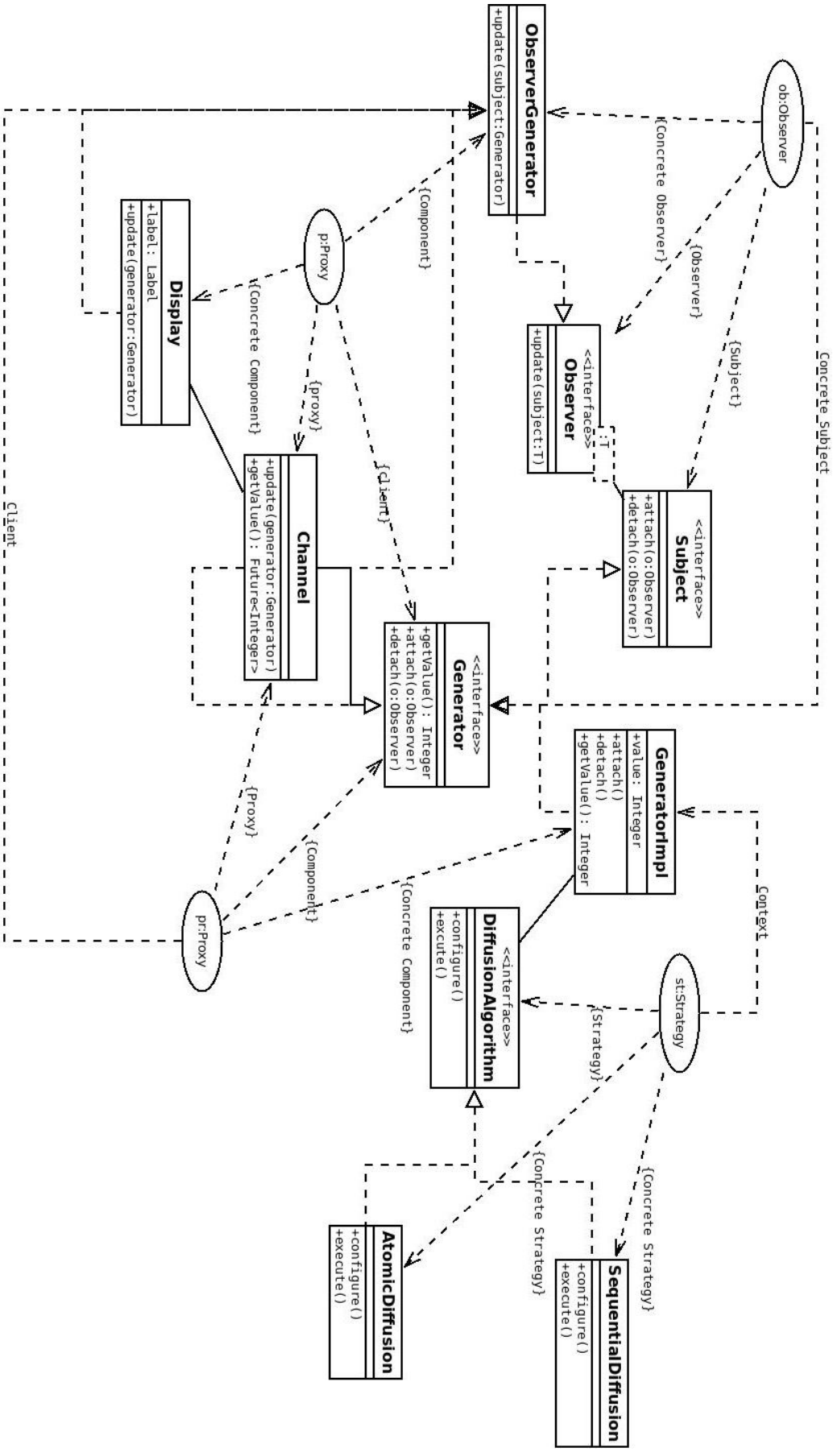
Nous avons rencontré plusieurs obstacles lors de la mise en place de l'architecture, des problèmes de compréhension du fonctionnement avec un niveau d'abstraction qui doit être assez élevé dans un premier temps et par la suite avec l'application de ces notions d'asynchronisme. Avec notamment la classe Channel qui devait implémenter 3 interfaces afin d'être à la fois un sujet et un observateur.

Interface graphique





M1 : Observateur asynchrone avec appel asynchrone d'opérations



M3 : Pattern Active Object + JDK Oracle

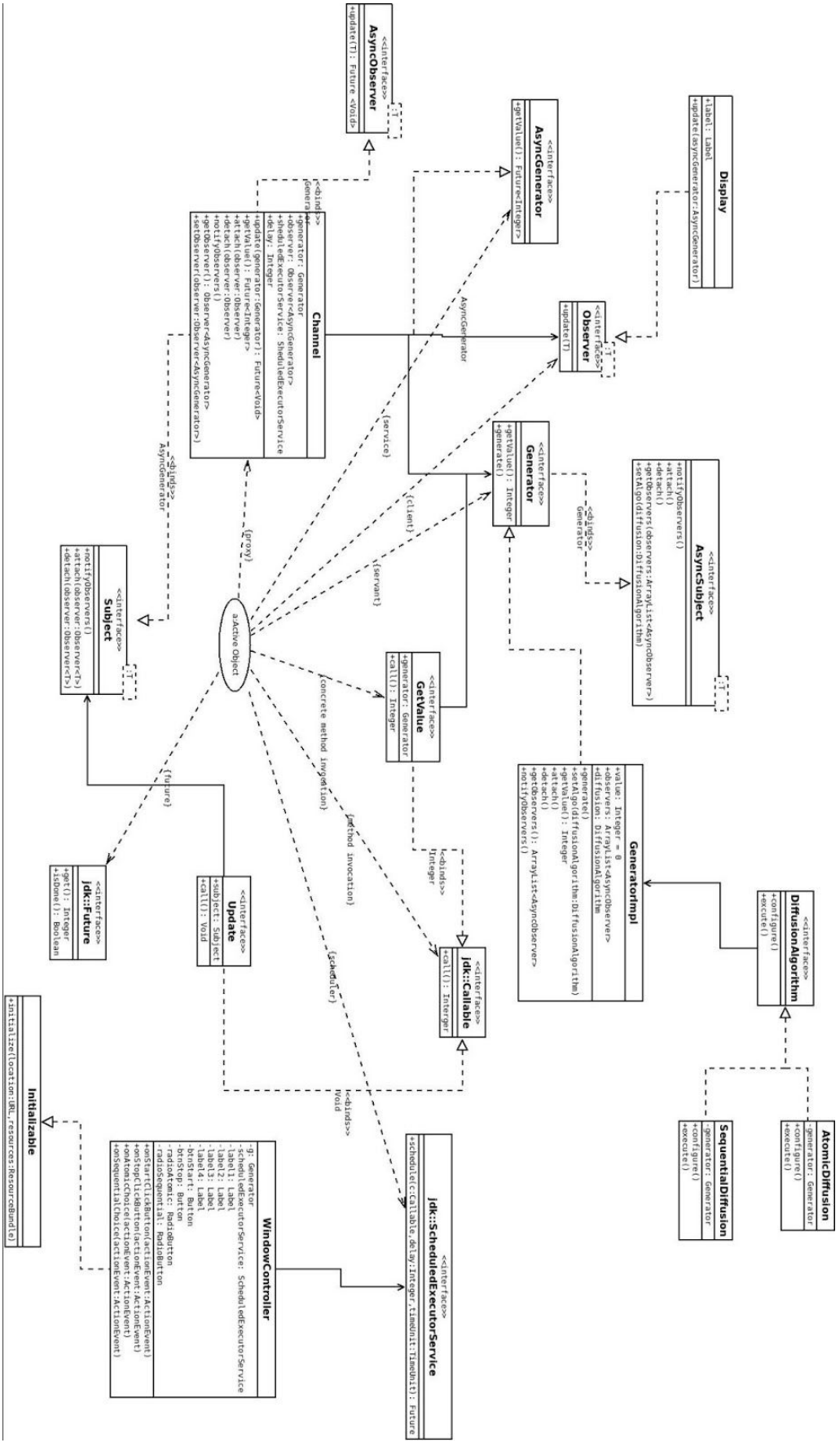


Diagramme de séquence “Update”

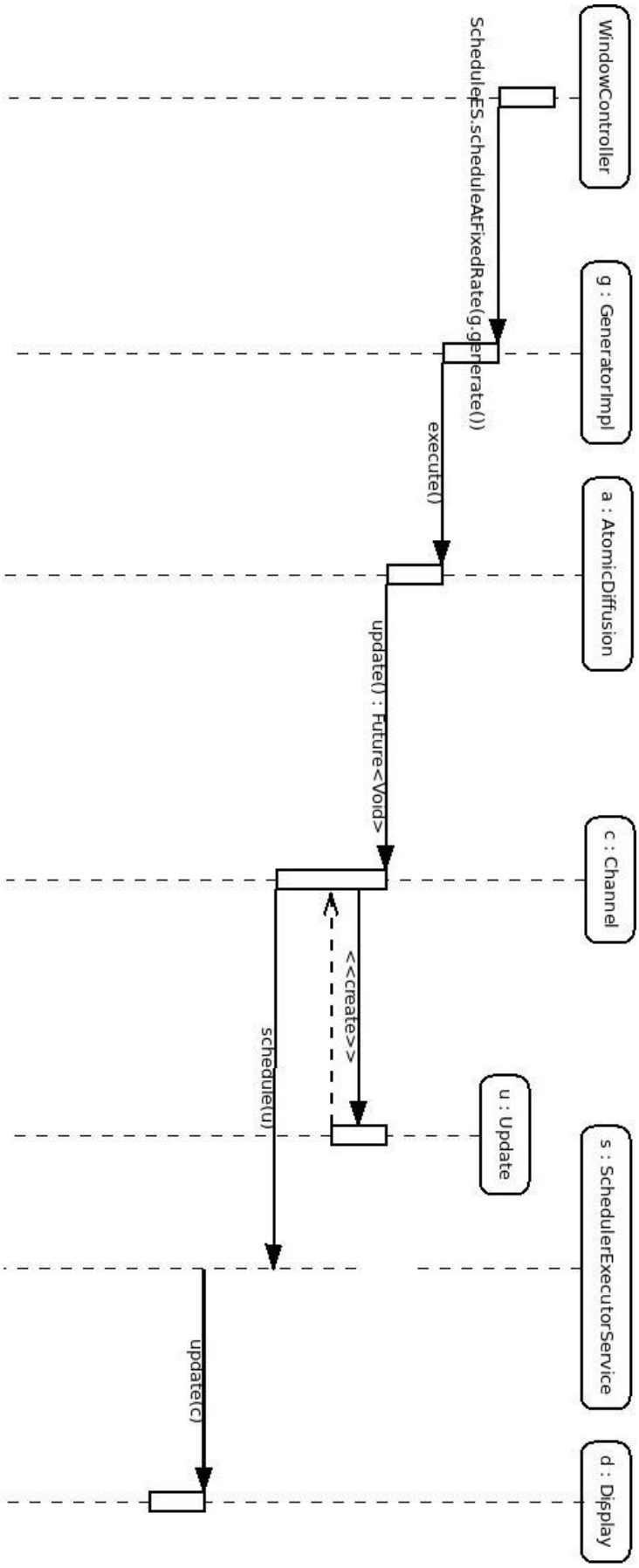


Diagramme de séquence “GetValue”