AOC M2 MIAGE
ORGEVAL Thomas & BOURGEOIS Bastien

AOC
ACTIVE OBJECT

—

Année 2022-2023

INFORMATIONS

Vous trouverez notre projet sur github à l'adresse https://github.com/Bourval/ACO-Observer en java avec la javadoc. Ainsi que les différents schéma réalisés avec plantUML que vous retrouverez également sur ce document et dans le repo ainsi que le jar.

CONCEPTION

Introduction

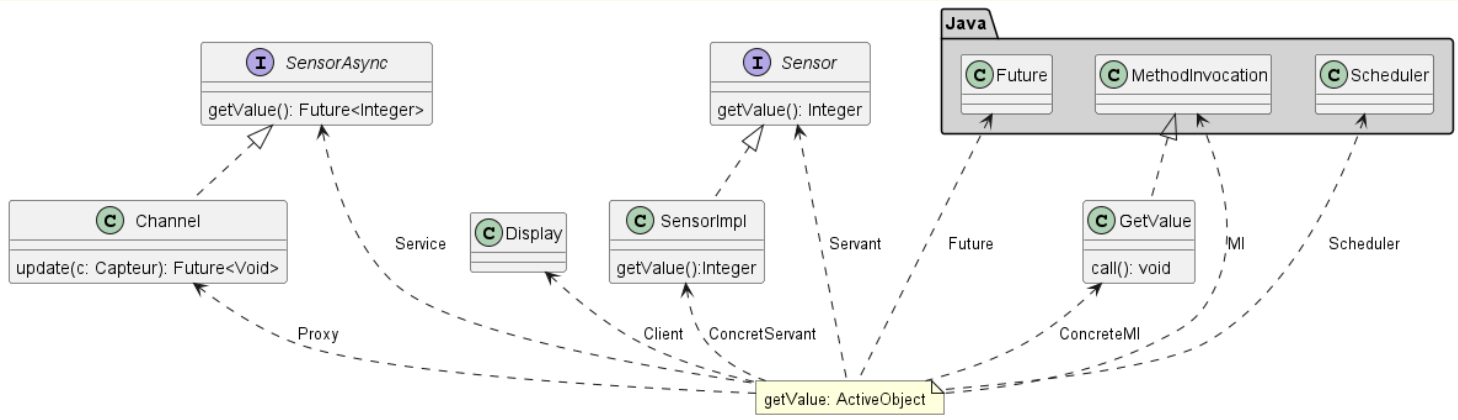
Dans ce projet, nous avons mis en place un service de diffusion de données de capteurs en utilisant le patron de conception Active Object. Le but de ce projet est de mettre en œuvre une diffusion parallèle d'Observer.

Projet

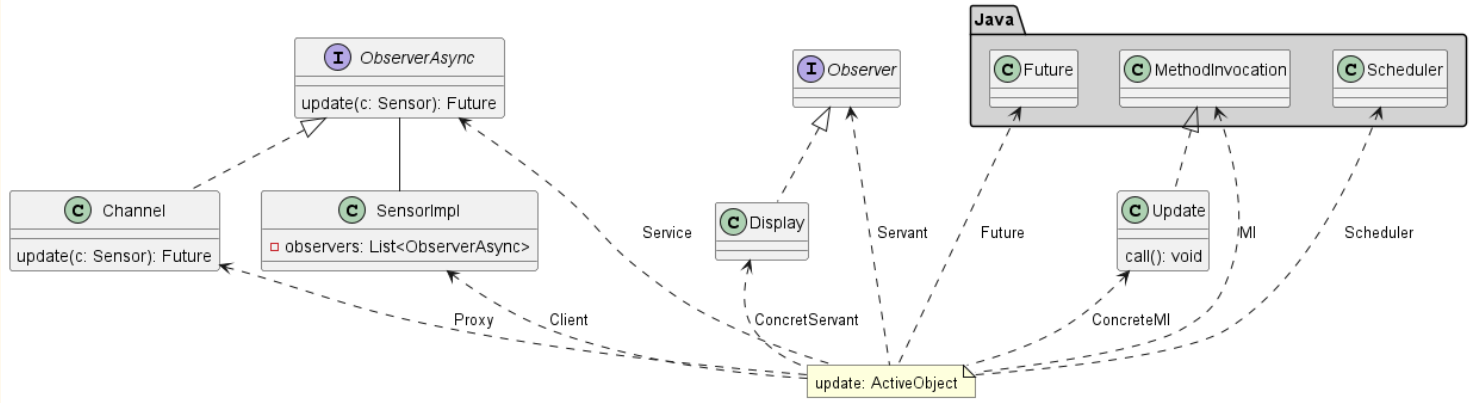
Notre solution consiste à diffuser un flux de valeurs à des objets abonnés exécutés dans des threads différents de la source du service. Le Capteur envoie une séquence croissante d'entiers via un compteur incrémenté à intervalle fixe. Les Canaux transmettent l'information aux Afficheurs, avec des délais de transmission aléatoires. Les Afficheurs récupèrent les valeurs diffusées pour les afficher. L'ensemble de politique de diffusion Observer comprend trois types de diffusions : atomique, séquentielle et par époque.

1. Diagramme de classe

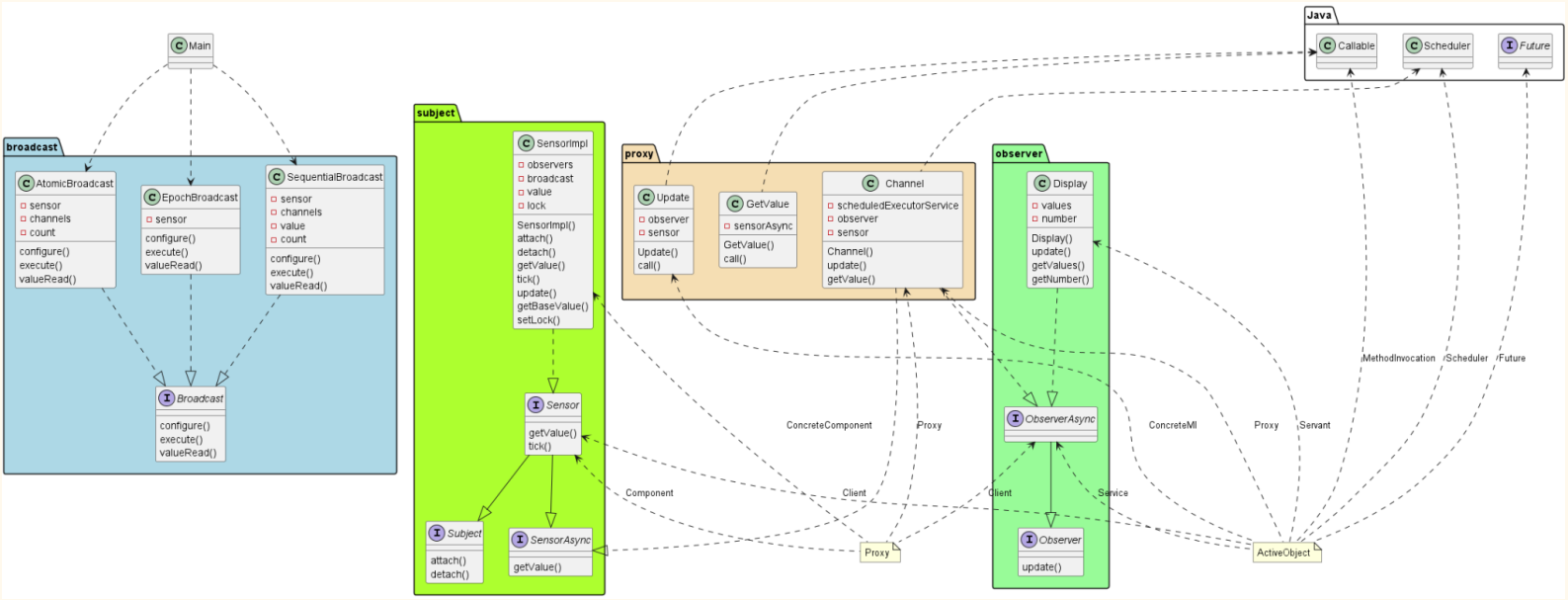
A. GetValue



B. Update



C. Update et GetValue avec les 3 broadcasts : Sequential, Atomic et Epoch



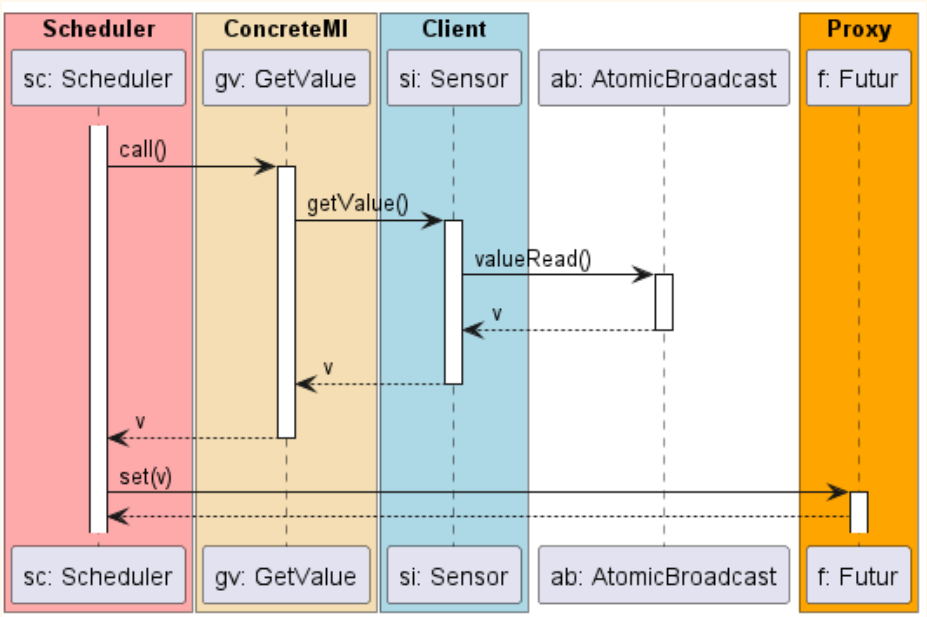
Le diagramme de classe montre comment le design patterns Active Object & Proxy ont été utilisés pour la diffusion Atomique.

2. Diagramme de séquence

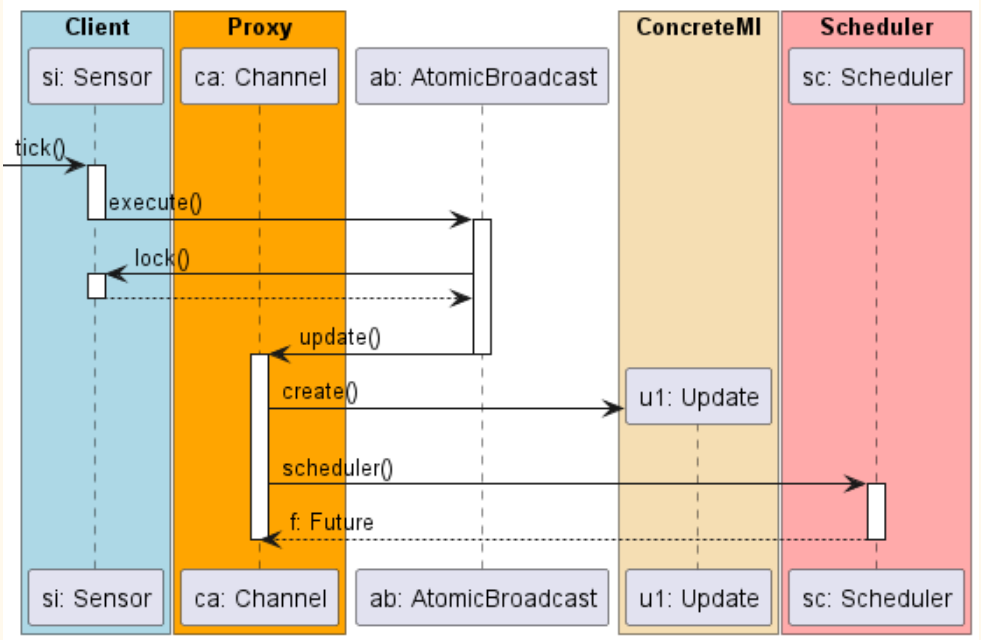
Les diagrammes de séquence montrent la séquence d'interactions et de messages entre les différentes classes implémentées dans l'architecture de diffusion de données pour les capteurs. Ils permettent de visualiser la façon dont les données sont diffusées aux observateurs à partir des capteurs, en utilisant les stratégies de diffusion définies.

Par exemple, dans le cas de la diffusion atomique, un diagramme de séquence pourrait montrer comment le capteur envoie une notification à tous les observateurs attachés en même temps, en leur transmettant la même valeur. Dans le cas de la diffusion séquentielle, un diagramme de séquence pourrait montrer comment le capteur envoie une notification séquentielle à chaque observateur attaché, en transmettant la même valeur à chaque fois, mais dans une séquence différente de celle du sujet.

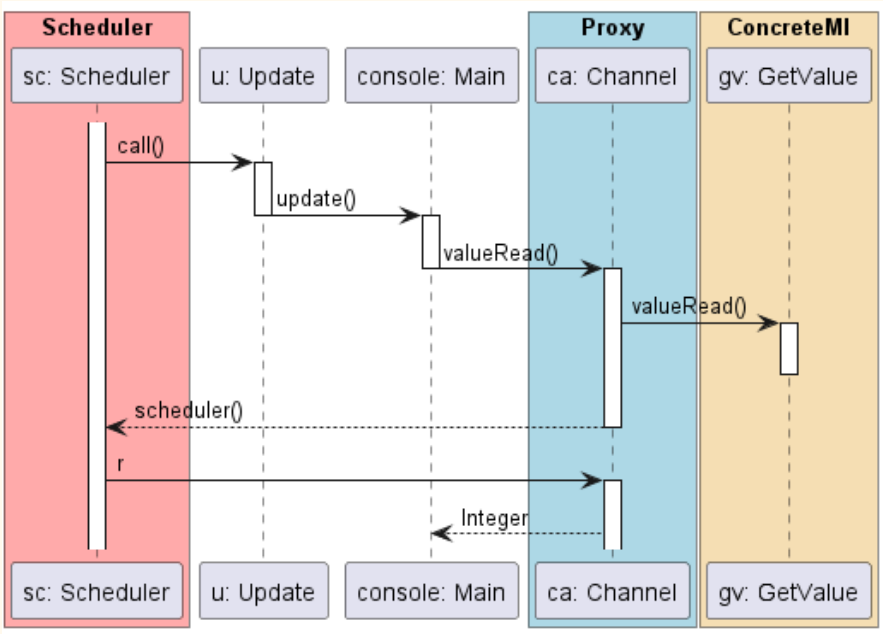
A. Diffusion atomique GetValue



B. Diffusion atomique Update



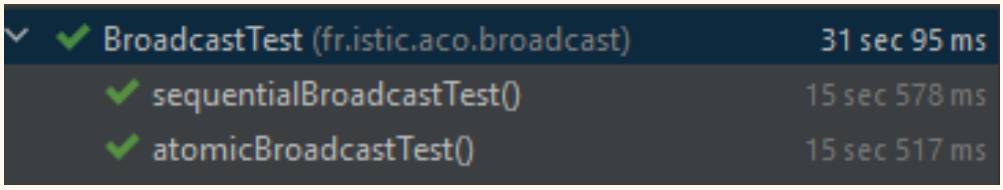
C. Diffusion séquentiel GetValue



VALIDATION

Tests

Pour valider notre projet, nous avons créé une classe de test pour comparer les résultats des Afficheurs. Il est important de noter que pour la diffusion par époque, certains Afficheurs peuvent être vides lors de l'exécution en raison de l'incohérence de cette méthode de diffusion. Il est possible de relancer le programme plusieurs fois pour avoir des échantillons supplémentaires.



Test 1

```
-- Atomic broadcast:
Display 0, values: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
Display 1, values: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
Display 2, values: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
-- Sequential broadcast:
Display 0, values: [1, 4, 7, 10, 14, 16, 19, 22, 26, 29, 33, 36, 40, 44, 47, 50]
Display 1, values: [1, 4, 7, 10, 14, 16, 19, 22, 26, 29, 33, 36, 40, 44, 47, 50]
Display 2, values: [1, 4, 7, 10, 14, 16, 19, 22, 26, 29, 33, 36, 40, 44, 47, 50]
-- Epoch broadcast:
Display 0, values: []
Display 1, values: []
Display 2, values: []
-- Epoch broadcast 2:
Display 0, values: [2, 2, 4, 5, 5, 6]
Display 1, values: []
Display 2, values: []
-- Epoch broadcast 3:
Display 0, values: [1, 2, 3, 5]
Display 1, values: [2, 3, 5, 7]
Display 2, values: [3, 4, 5, 8]
```

Test 2

```
-- Atomic broadcast:
Display 0, values: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
Display 1, values: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
Display 2, values: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
-- Sequential broadcast:
Display 0, values: [1, 3, 7, 10, 12, 15, 19, 23, 27, 31, 35, 40, 44, 48]
Display 1, values: [1, 3, 7, 10, 12, 15, 19, 23, 27, 31, 35, 40, 44, 48]
Display 2, values: [1, 3, 7, 10, 12, 15, 19, 23, 27, 31, 35, 40, 44, 48]
-- Epoch broadcast:
Display 0, values: [1, 2, 3]
Display 1, values: [1, 3, 4]
Display 2, values: [2, 5, 6]
-- Epoch broadcast 2:
Display 0, values: [2, 3]
Display 1, values: [3, 4]
Display 2, values: [4, 5]
-- Epoch broadcast 3:
Display 0, values: [2]
Display 1, values: [2]
Display 2, values: [4]
```

CONCLUSION

En résumé, ce projet a permis de mettre en pratique les patrons de conception AOC et ACO pour simuler un service de diffusion de données de capteurs. Les différentes diffusions proposées ont été mises en œuvre et validées grâce à des tests pour vérifier leur fonctionnement.