



# Projet Active Object

**Mamady CONDE**
**Yannick NAMOUR**
**M2 CCN 2018/2019**

**1 . Introduction**

L'implémentation de la conception Active Object est une technique pour la conception par Objets qui a pour objectif la mise en oeuvre des architectures asynchrones c'est-à-dire de mettre en place le patron de conception Active Object afin de permettre des appels asynchrones sur un modèle synchrone.

L'objectif de ce projet est de réaliser un service de diffusion de données d'un Générateur en utilisant les mécanismes de programmation par threads, le patron de conception Observer et le patron de conception Active Object.

**2 . Architecture et Conception**

**2 . 1. Diagramme d'instance**

Le but de l'application est de concevoir un programme qui génère et envoie des valeurs pour ensuite les afficher sur les différents afficheur existants. Nous avons mis en place un système de générateur qui génère et transmet des valeurs aux différents canaux qui, à leur tour se chargent de créer des Method Invocation permettant l'envoie des valeurs aux afficheurs. La diffusion des valeurs se fait soit :

- de manière automatique, c'est-à-dire tous les observateurs voient la suite complète des valeurs prises par le sujet mais pas au même moment
- de manière séquentielle, c'est-à-dire les observateurs voient les même suites de valeurs généré par le générateur mais pas du l'ordre du générateur

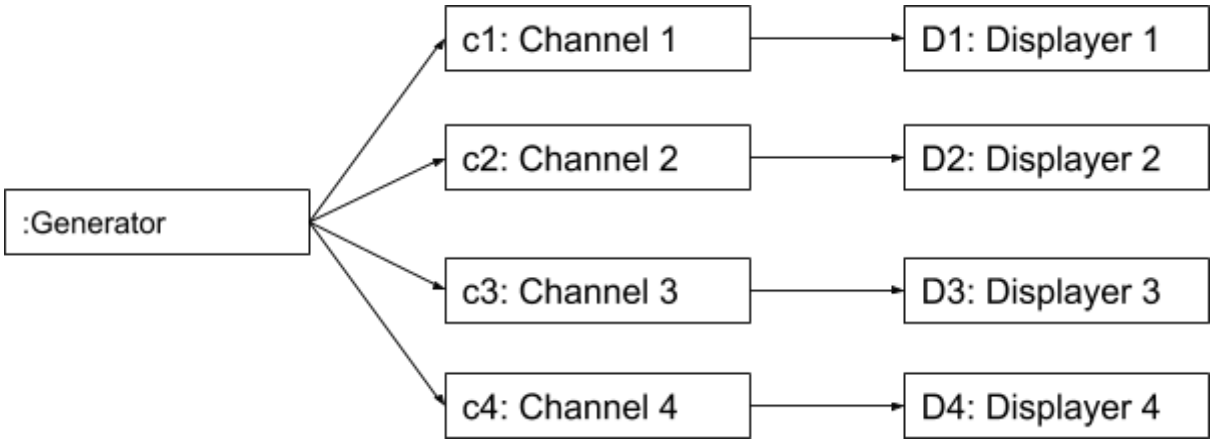



*Fig 1: Diagramme d'instance.*

**2. 2 . Mode Synchrone**

Nous avons commencé l'implémentation du projet par le mode synchrone. Un mode dans lequel le Générateur communique directement avec l'afficheur sans le canal.

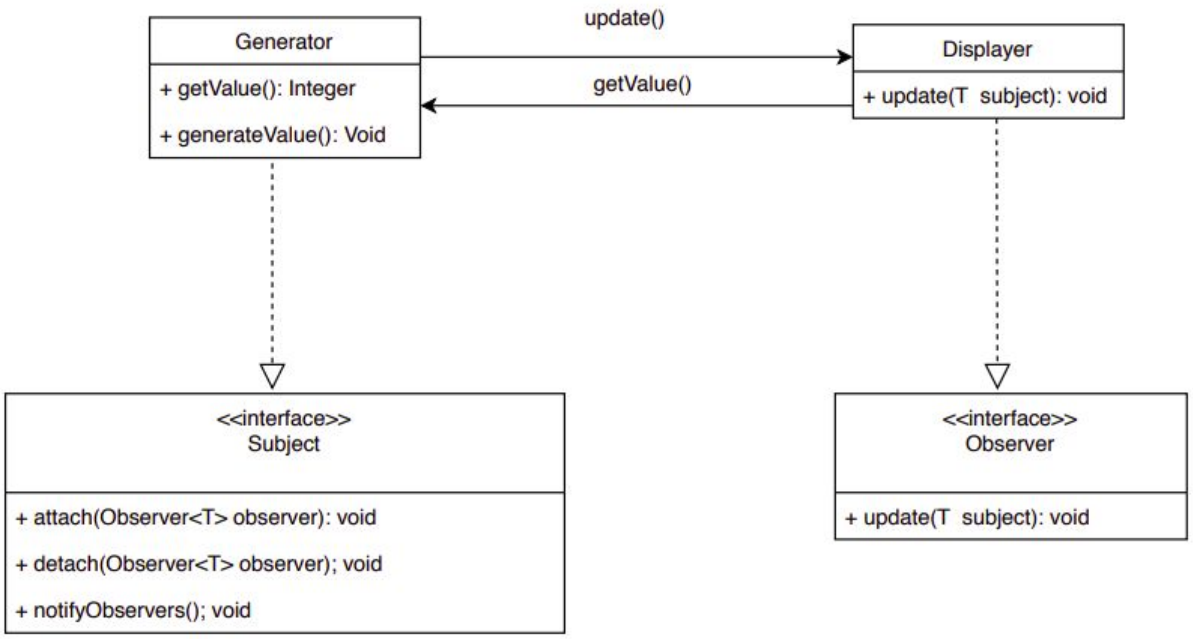



*Fig 2: Diagramme de classe.*

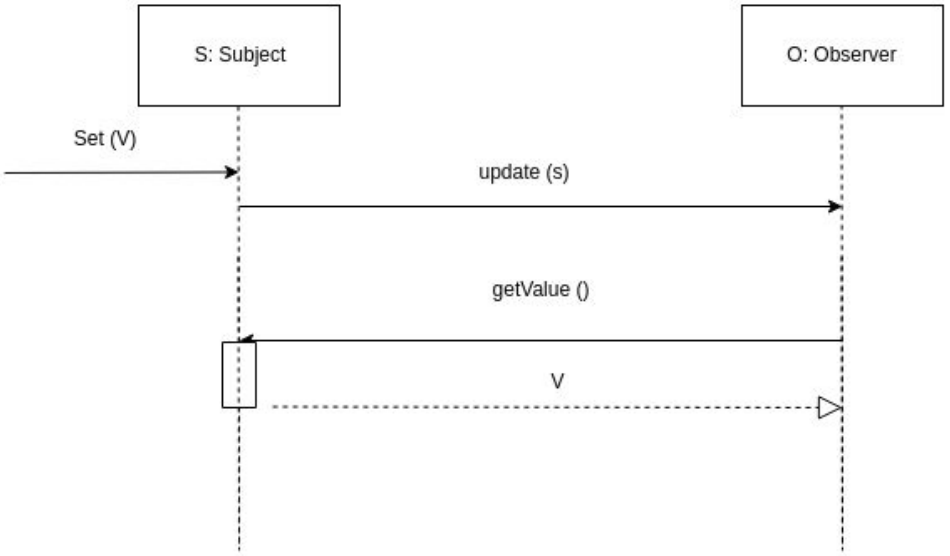



*Fig 3: Diagramme de séquence.*

### 2 . 3 . Mode Asynchrone

Ce mode se décline en deux modèles: le modèle M1 et le modèle M3.

- **Modèle M1:**

Le modèle M1 est l'implémentation d'observer asynchrone. Pour ce modèle, le générateur, qui est ici représenté par le Subject , génère un nombre puis notifie à travers le canal son afficheur (DisplayerImpl qui implémente Observer) pour afficher la valeur dans la console.

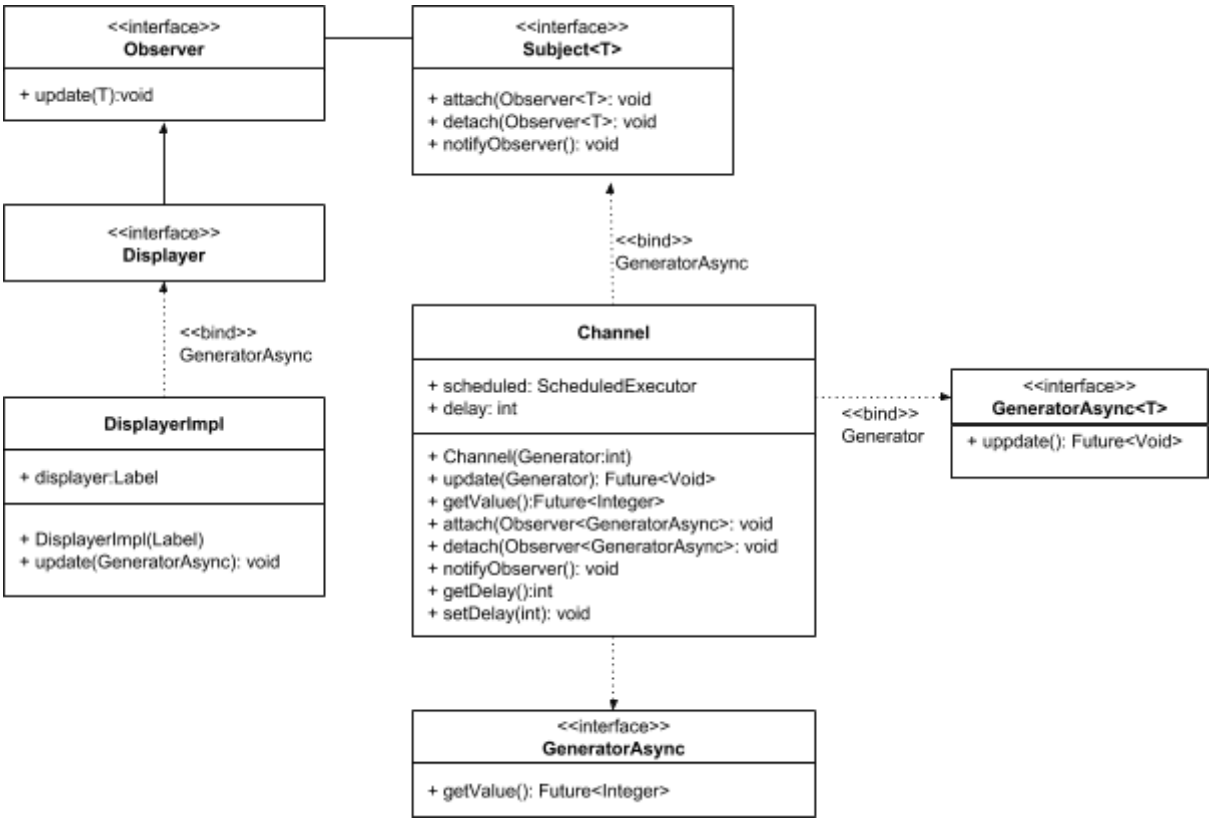



*Fig 4: Diagramme de Classe du modèle M1.*

- **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.

Dans ce modèle l'utilisation du patron de conception apparaît deux fois, l'une pour la fonction Update et l'autre pour la fonction GetValue.

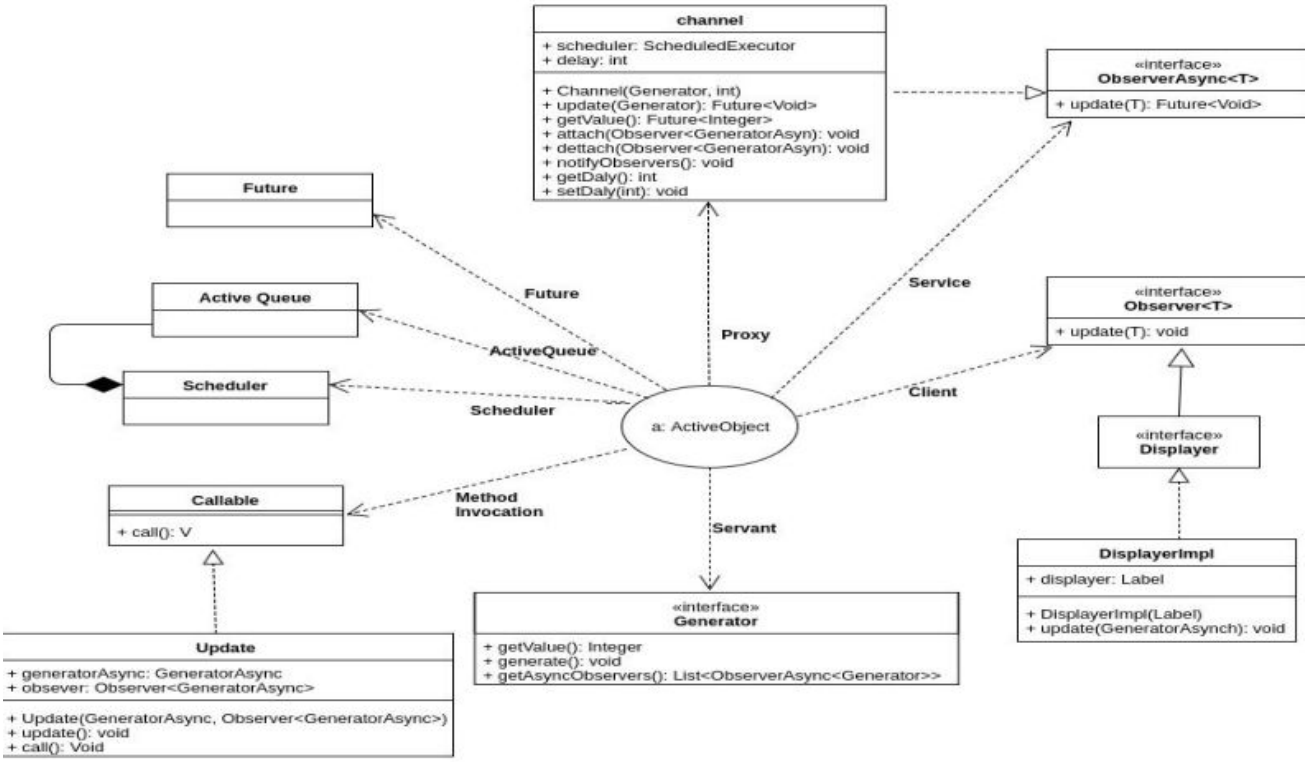



*Fig 5: Patron de conception pour la fonction Update.*

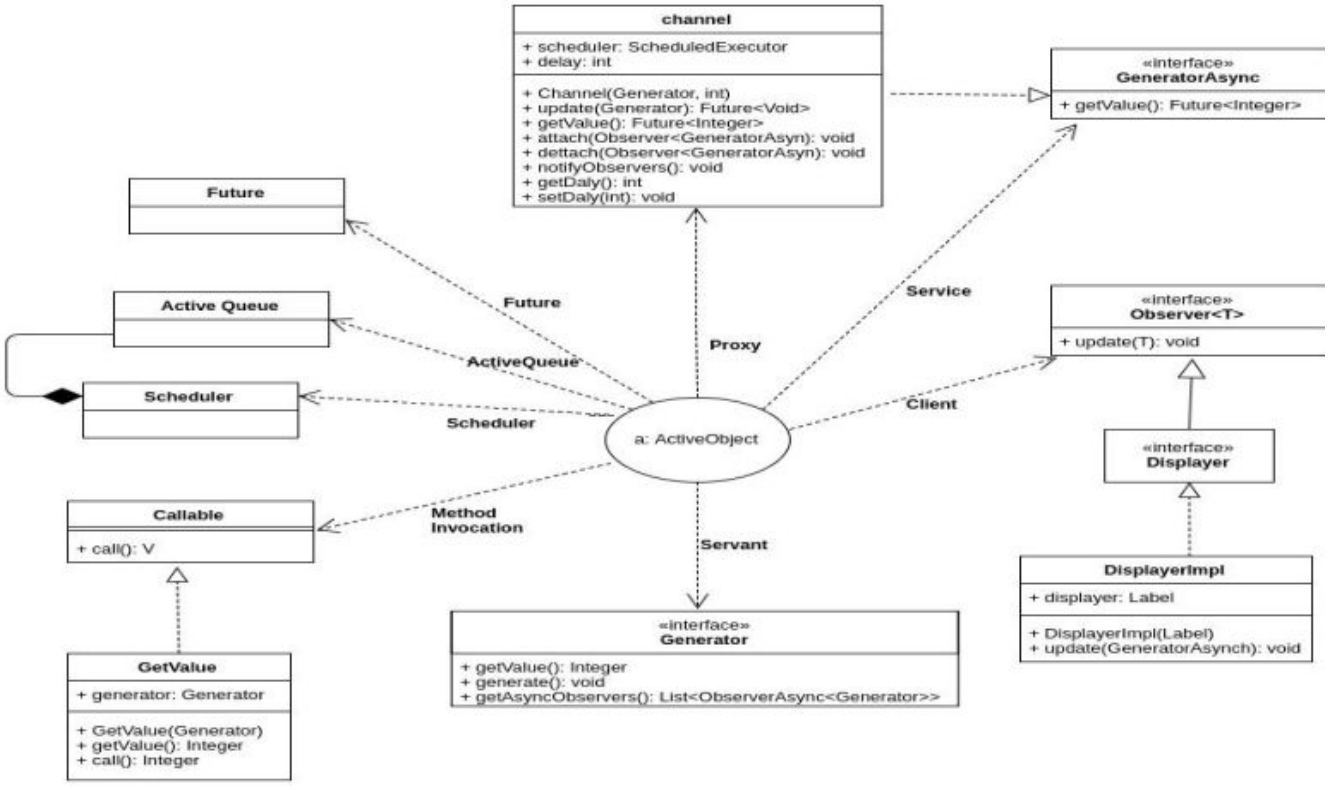
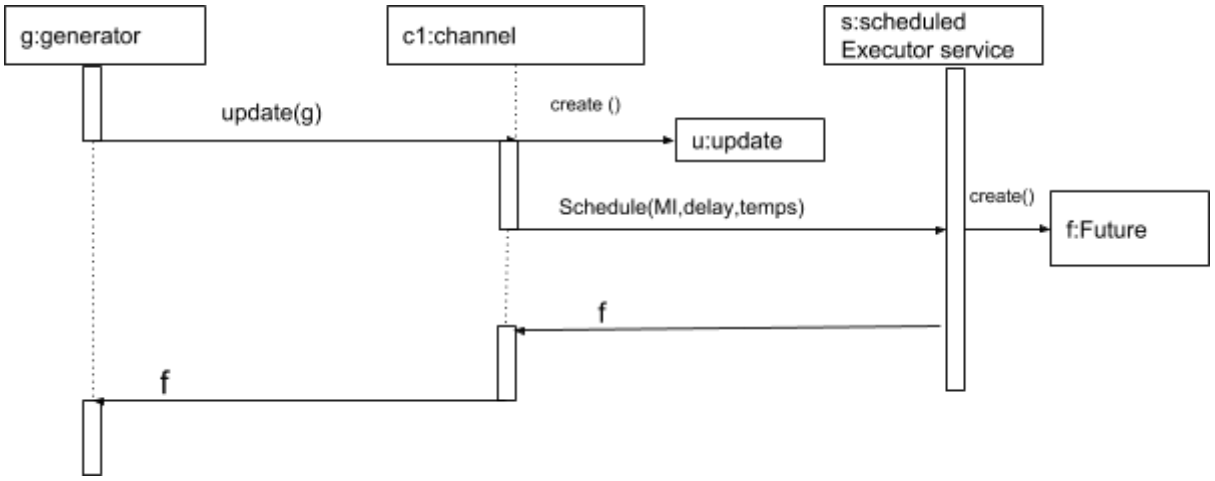


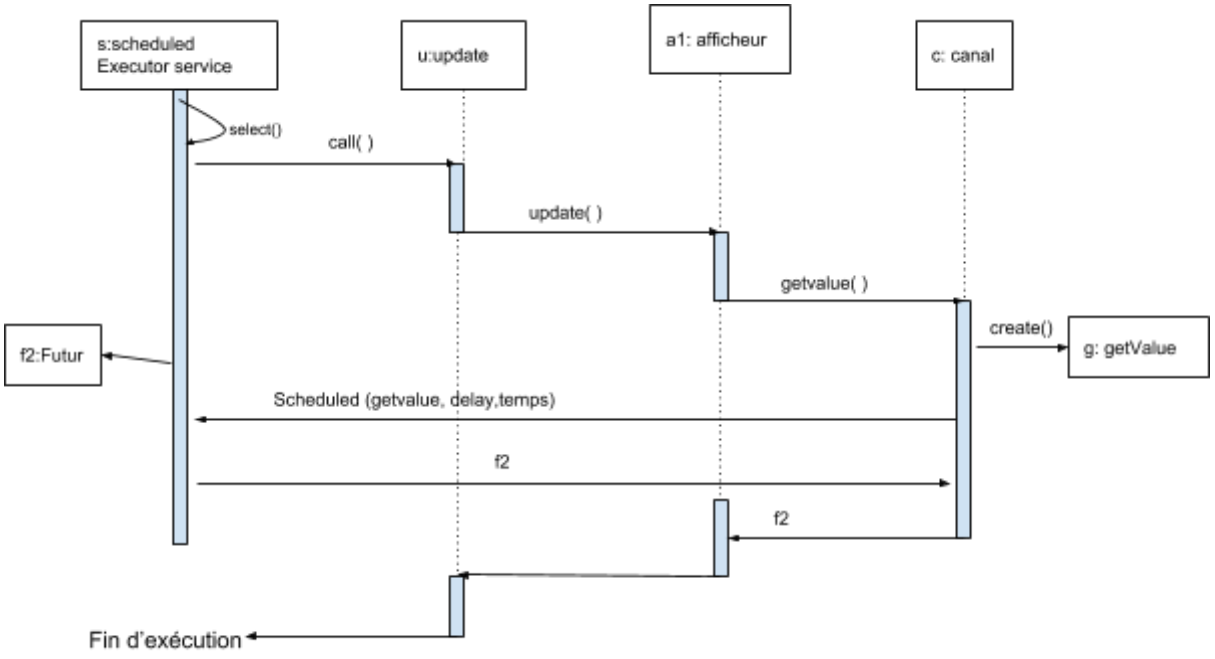

*Fig 6: Patron de Conception pour la fonction GetValue.*

## Diagramme de séquence

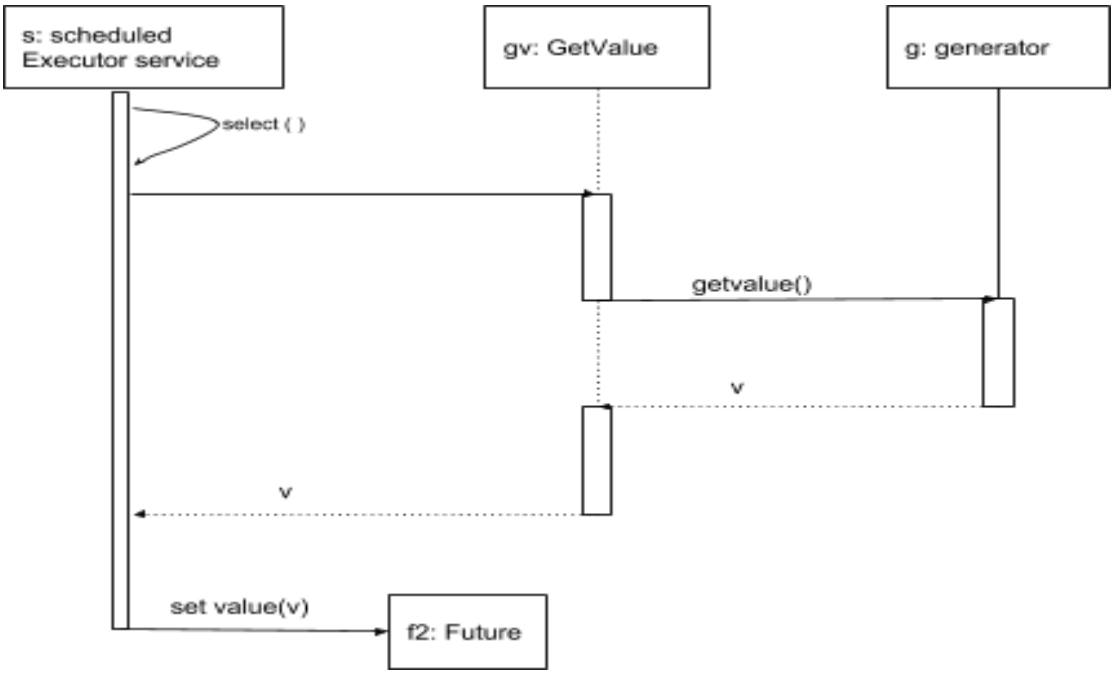
**Phase 1 :** pour la méthode update




**Phase 2a:** pour la méthode update et phase 1 pour la méthode getvalue




**Phase 2b:** pour la méthode update et 2a pour la méthode getvalue




**3 . Tests**

Nous observons effectivement la diffusion des valeurs générées de manière aléatoire par le Générateur.

Mais en console nous n'observons pas de différence entre l'affichage en stratégie séquentielle et en atomique.

```
Valeur du Generateur: 85
Valeur Afficheur 85
Valeur Afficheur 85
Valeur Afficheur 85
Valeur Afficheur 85
Valeur du Generateur: 21
Valeur Afficheur 21
Valeur Afficheur 21
Valeur Afficheur 21
Valeur Afficheur 21
```

*Fig 7: Affichage Séquentiel*

```
Valeur du Generateur: 32
Valeur Afficheur 32
Valeur Afficheur 32
Valeur Afficheur 32
Valeur Afficheur 32
Valeur du Generateur: 48
Valeur Afficheur 48
Valeur Afficheur 48
Valeur Afficheur 48
Valeur Afficheur 48
```

*Fig 8: Affichage Atomique*