# AOC : Rapport de Validation

Hugo Bannier
Antoine Leval

January 28, 2019

**Abstract**

Ce projet consistait à réaliser un service de diffusion de données de capteur en s'appuyant sur le patron de conception Active Object, ainsi que les autres patrons de conception vus en cours ACO et AOC. Le code source du projet est téléchargeable à l'adresse suivante : https://github.com/Viinyard/AOC_PROJET/tree/dev.

## 1 Implémentation

Pour mettre en place l'architecture du projet nous avons dû suivre plusieurs pattron de conception :

- Active Object
- Proxy
- Observable/Observer
- Strategy

Le Pattron de Conception Active Object doit être effectif dans deux sens, pour lancer l'update de l'afficheur, et pour récupérer la valeur du générateur à partir de l'afficheur. Les deux actions étant asynchrone.

Pour simuler le côté asynchrone de l'application nous avons dû utiliser le pattron de conception Proxy, qui ajoute un temps de latence entre chaque action Update et GetValue.

Pour gérer la mise à jour automatique de l'afficheur à chaque changement d'état du générateur nous avons implémenté le pattron de conception Observable / Observer.

## 2 Les Diffusions

### 2.1 La diffusion atomique

Pour la diffusion atomique on devait veiller à ce que tous les observers reçoivent la même valeur, et qu'ils reçoivent toutes les valeurs.

Pour faire cela nous avons rendû l'appel de la méthode métant à jour la valeur du générateur bloquante.

istic > aoc > m3 > diffusion > AtomiqueStrategy.java

```
List<Future<Void>> enAttentes = this.observable.getObservers().stream()
        .map((it) -> it.update(this.generator))
        .collect(Collectors.toList());

enAttentes.forEach(it -> {
    try {
        it.get();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
```

```
    }
});
```

En appelant la méthode `get` du `Futur<Void>` renvoyé par la méthode `update` de notre `observer`. On s'assure que la méthode qui met à jour la valeur du générateur est bloquante, et donc qu'aucune autre valeur ne sera envoyé avant que chaque afficheur se soit mis à jour.

L'action bloquante ici est le `update`, mais si l'on suit le fil d'exécution du thread qui exécutera la `method invocation update`, on s'aperçoit que la `method invocation getValue` est bloquante pour ce processus.

istic > aoc > m3 > afficheur > Afficheur.java

```
@Override
public void update(GeneratorAsync g) {
    try {
        this.value = Optional.of(g.getValue().get());
    } catch (InterruptedException | ExecutionException e) {
        e.printStackTrace();
    }
    this.repaint();
}
```

Ici on peut voir que le processus qui éxecutera la `method invocation update` sera bloqué par le `g.getValue().get()` et donc que le `Futur<Void> update` ne sera fini tant que la `method invocation getValue` ne sera terminée et aura renvoyé sa valeur.

Cette méthode n'est pas optimale car on bloque un processus du ThreadPool pour rien. On pourrait par exemple appeler dans la méthode `update` de l'afficheur appeler un `Futur<Void>` au lieu d'un `Futur<Long>` qui attend la valeur du générateur.

Le `Futur<Void>` s'occupera une fois appelé de mettre changer la valeur de l'afficheur et d'ordonner sa mise à jour.

La méthode `call()` du `method invocation setValue` (au lieu de `getValue`) appelerait par exemple `afficheur.setValue(maValeur);`. Et dans l'afficheur on aurait une méthode :

```
public void setValue (long maValeur) {
    this.maValeur = maValeur;
    this.repaint();
}
```

Ainsi l'appele de la méthode Update asynchrone ne serait pas bloquante.

Mais nous avons gardé la méthode bloquante de l'update car celle-ci nous offre un meilleur contrôle sur les afficheur, cela nous permet d'être notifié lorsque l'afficheur a reçu la valeur et s'est mis à jour. Ce qui est indispensable pour mettre en place certaines strategie de diffusion demandée dans ce projet.

```
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 0
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 838ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 4093ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 2175ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 1496ms latency
[pool-1-thread-1] INFO istic.aoc.m3.active.Canal - GET 0 with 5446ms latency
[pool-7-thread-1] INFO istic.aoc.m3.active.Canal - GET 0 with 3321ms latency
[pool-5-thread-1] INFO istic.aoc.m3.active.Canal - GET 0 with 823ms latency
[pool-5-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[0]
[pool-3-thread-1] INFO istic.aoc.m3.active.Canal - GET 0 with 3269ms latency
[pool-7-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[0]
[pool-1-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[0]
[pool-3-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[0]
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 1
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 1 with 2208ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 1 with 5157ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 1 with 1643ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 1 with 5408ms latency
```

Ici on peut voir la trace de log de la strategie atomique de diffusion. On observe que le générateur modifie la valeur à 0, envois un update à chaque observer avec un temps de latence aléatoire. Ensuite chaque afficheur envois un getValue au générateur avec un temps de latence également, puis enfin ils se mettent à jour. Seulement une fois que chaque afficheur se soit mis à jour le générateur envois la prochaine valeur : 1 et envois les updates à chaque observers, etc..

## 2.2 La diffusion séquentielle

Le principe de la diffusion séquentielle est que chaque observer (ou afficheur) doivent recevoir la même valeur, mais la séquence n'est pas forcément la séquence réelle du sujet (le générateur).

istic > aoc > m3 > diffusion > SequentielStrategy.java

```java
@Override
public void execute() {
    if(this.enAttentes.stream().allMatch(f -> f.isDone())) {
        this.enAttentes = this.observable.getObservers().stream()
            .map((it) -> it.update(new MementoGenerator(this.generator.getValue())))
            .collect(Collectors.toList());
    }
}
```

Nous devons donc mettre à jour chaque afficheur avec la même valeur. Pour faire cela, a chaque fois que la valeur du générateur est mise à jour, on regarde en premier lieu si tous les afficheurs sont disponibles pour être mis à jour (c'est à dire si il n'y a pas de Futur<Void> de non terminés sur les observers).

Si tous les observers sont disponibles alors on les mets tous à jour avec la même valeur.

Pour faire cela on utilise un Memento sur le générateur. De cette façon on peut mettre à jour tous les afficheurs avec une copie du générateur, renvoyant la même valeur à chacun. La stratégie implémente `Generator`, car la méthode update de l'observer asynchrone prend un

```java
class MementoGenerator implements Generator {
    private long value;

    public MementoGenerator(long value) {
        this.value = value;
    }

    @Override
    public long getValue() {
        return this.value;
    }

    @Override
    public void setValue(long value) {}
}
```

De cette manière, lorsque l'afficheur appelera getValue sur le canal, nous somme certain que tous les canaux renverrons la même valeur.

Une fois que tous les afficheurs ont fini de se mettre à jour, le processus update appelant est libéré, et le `Futur<Void> update(..)` se fini. La prochaine valeur sera donc envoyé.

```
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 0
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 1840ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 2051ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 1369ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 657ms latency
[pool-7-thread-1] INFO istic.aoc.m3.active.Canal - GET 0 with 1378ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 1
[pool-5-thread-1] INFO istic.aoc.m3.active.Canal - GET 0 with 3621ms latency
[pool-1-thread-1] INFO istic.aoc.m3.active.Canal - GET 0 with 1041ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 2
[pool-7-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[0]
```

```
[pool-3-thread-1] INFO istic.aoc.m3.active.Canal - GET 0 with 4279ms latency
[pool-1-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[0]
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 3
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 4
[pool-5-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[0]
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 5
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 6
[pool-3-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[0]
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 7
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 7 with 1338ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 7 with 999ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 7 with 2753ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 7 with 539ms latency
```

Si on observe la trace de log de la stratégie séquentielle on peut voir que le générateur met la valeur à 0. Aucun update n'est en cours, alors tous les observers sont mis à jour, chaque observeur envoyé par l'update ont la valeur 0.

Ensuite tant que les afficheurs sont encore en train de se mettre à jour, le générateur prend les valeurs 1, 2, 3, 4, 5 et 6 mais comme le dernier afficheur n'avait pas fini de se mettre à jour ces changements de valeurs ne sont pas notifié aux observers.

Ensuite le dernier afficheur (le Rouge) se met à jour, tous les `Futur<Void> update(..)` sont terminés, le générateur reçoit la valeur 7, cette fois tous les observers sont notifiés de ce changement de valeur.

On peut observer que le générateur change bien de valeur, mais que les afficheurs récupèrent bien la valeur que le générateur avait au moment du update car ([0]).

## 2.3 La diffusion causale

La diffusion causale n'oblige pas tous les observers à recevoir la même valeur, mais si un afficheur reçoit la valeur n, cela implique qu'il reçoit la valeur n - 1. Il doivent donc chacun, et individuellement recevoir toutes les valeurs, dans l'ordre. C'est à dire que l'afficheur 1 ne doit pas recevoir 5 puis 4 puis 6.

Pour faire cela on a utilisé un pattron de conception Memento sur les valeurs du generateur. Le gardien est ici le Generator, pour une question de simplicité d'implémentation.

```java
public class MementoGenerator implements Generator {

    private Queue<Long> mementoValue = new LinkedList<>();

    @Override
    public long getValue() {
        return this.mementoValue.poll();
    }

    @Override
    public void setValue(long value) {
        this.mementoValue.add(value);
    }
}
```

On envoie les update dans l'ordre, mais ils ne sont pas exécuté de façon synchrone, il nous faut donc un pattron de conception Memento sur les valeurs renvoyé par le getValue, peut importe la valeur du générateur au moment ou l'update est envoyé nous devons renvoyer au moment du get la dernière valeur en attente (Queue de type FIFO).

```java
@Override
public void execute() {
    this.observable.getObservers().forEach((it) -> {
        this.hmMemento.putIfAbsent(it, new MementoGenerator());
        Generator gen = this.hmMemento.get(it);
        gen.setValue(generator.getValue());
        it.update(gen);
```

```
    });
}
```

Chaque observer a son propre générateur, chaque générateur possède un memento sur les valeurs renvoyés, de façon à ne pas en louper une seule et les renvoyer dans le bon ordre. Ici chaque generateur est plus ou moins indépendant et peuvent avoir de gros écarts de valeurs entre eux.

Pour pallier au fait qu'ici nous envoyons tous les updates et que l'exécutor service peut vite déborder, et bloquer la création d'un Thread pour faire un Get, ce qui créerait un interbloquage, nous avons implémenté deux éxécutor service, un pour les updates, et un pour les gets.

```
private ScheduledExecutorService executorServiceUpdate = new
    ScheduledThreadPoolExecutor(4);
private ScheduledExecutorService executorServiceGet = new ScheduledThreadPoolExecutor(4);
```

Dans les traces de log on peut voir que dans un premier temps plusieurs update sont envoyés, puis à la fin que tous les gets sont fait dans l'ordre, et qu'aucune valeur n'est manquée par les afficheurs.

```
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 0
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 1523ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 3798ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 2509ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 0 with 1787ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 1
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 1 with 4560ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 1 with 5479ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 1 with 5248ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 1 with 5323ms latency
[pool-1-thread-1] INFO istic.aoc.m3.active.Canal - GET 1 with 5289ms latency
[pool-7-thread-1] INFO istic.aoc.m3.active.Canal - GET 1 with 4939ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 2
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 2 with 4761ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 2 with 4532ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 2 with 4331ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 2 with 5467ms latency
[pool-5-thread-1] INFO istic.aoc.m3.active.Canal - GET 2 with 4104ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 3
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 3 with 2499ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 3 with 1851ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 3 with 2222ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 3 with 1250ms latency
[pool-3-thread-1] INFO istic.aoc.m3.active.Canal - GET 3 with 4434ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 4
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 4 with 1427ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 4 with 2156ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 4 with 1008ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 4 with 1606ms latency
[pool-7-thread-2] INFO istic.aoc.m3.active.Canal - GET 4 with 5135ms latency
[pool-3-thread-2] INFO istic.aoc.m3.active.Canal - GET 4 with 4046ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 5
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 5 with 4750ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 5 with 4171ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 5 with 2159ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 5 with 3762ms latency
[pool-5-thread-2] INFO istic.aoc.m3.active.Canal - GET 5 with 4764ms latency
[pool-5-thread-3] INFO istic.aoc.m3.active.Canal - GET 5 with 2430ms latency
[pool-1-thread-3] INFO istic.aoc.m3.active.Canal - GET 5 with 1870ms latency
[pool-1-thread-2] INFO istic.aoc.m3.active.Canal - GET 5 with 4936ms latency
[pool-1-thread-4] INFO istic.aoc.m3.active.Canal - GET 5 with 2415ms latency
[pool-7-thread-3] INFO istic.aoc.m3.active.Canal - GET 5 with 5159ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 6
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 6 with 1277ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 6 with 3744ms latency
```

```
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 6 with 4419ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 6 with 2617ms latency
[pool-3-thread-3] INFO istic.aoc.m3.active.Canal - GET 6 with 2253ms latency
[pool-5-thread-4] INFO istic.aoc.m3.active.Canal - GET 6 with 1601ms latency
[pool-7-thread-4] INFO istic.aoc.m3.active.Canal - GET 6 with 2623ms latency
[pool-3-thread-4] INFO istic.aoc.m3.active.Canal - GET 6 with 2982ms latency
[pool-5-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[0]
[pool-5-thread-1] INFO istic.aoc.m3.active.Canal - GET 6 with 1179ms latency
[pool-7-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[0]
[pool-1-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[0]
[pool-1-thread-1] INFO istic.aoc.m3.active.Canal - GET 6 with 2726ms latency
[Thread-0] INFO istic.aoc.m3.generator.GeneratorImpl - SET 7
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 7 with 523ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 7 with 2270ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 7 with 2249ms latency
[Thread-0] INFO istic.aoc.m3.active.Canal - UPDATE 7 with 620ms latency
[pool-1-thread-3] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[1]
[pool-1-thread-3] INFO istic.aoc.m3.active.Canal - GET 7 with 2587ms latency
[pool-7-thread-1] INFO istic.aoc.m3.active.Canal - GET 7 with 4274ms latency
[pool-5-thread-3] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[1]
[pool-5-thread-3] INFO istic.aoc.m3.active.Canal - GET 7 with 1379ms latency
[pool-5-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[2]
[pool-5-thread-4] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[3]
[pool-1-thread-4] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[2]
[pool-1-thread-4] INFO istic.aoc.m3.active.Canal - GET 7 with 2014ms latency
[pool-3-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[0]
[pool-3-thread-1] INFO istic.aoc.m3.active.Canal - GET 7 with 1291ms latency
[pool-3-thread-3] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[1]
[pool-3-thread-2] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[2]
[pool-7-thread-4] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[1]
[pool-7-thread-4] INFO istic.aoc.m3.active.Canal - GET 7 with 4637ms latency
[pool-5-thread-3] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[4]
[pool-3-thread-3] INFO istic.aoc.m3.active.Canal - GET 7 with 4519ms latency
[pool-5-thread-1] INFO istic.aoc.m3.active.Canal - GET 7 with 5015ms latency
[pool-3-thread-2] INFO istic.aoc.m3.active.Canal - GET 7 with 1841ms latency
[pool-7-thread-2] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[2]
[pool-7-thread-2] INFO istic.aoc.m3.active.Canal - GET 7 with 715ms latency
[pool-3-thread-4] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[3]
[pool-3-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[4]
[pool-1-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[3]
[pool-3-thread-4] INFO istic.aoc.m3.active.Canal - GET 7 with 2210ms latency
[pool-1-thread-1] INFO istic.aoc.m3.active.Canal - GET 7 with 841ms latency
[pool-5-thread-2] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[5]
[pool-1-thread-3] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[4]
[pool-1-thread-4] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[5]
[pool-7-thread-2] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[3]
[pool-7-thread-2] INFO istic.aoc.m3.active.Canal - GET 7 with 4288ms latency
[pool-5-thread-4] INFO istic.aoc.m3.active.Canal - GET 7 with 1939ms latency
[pool-1-thread-2] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[6]
[pool-1-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Blanc MAJ : Optional[7]
[pool-7-thread-3] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[4]
[pool-3-thread-2] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[5]
[pool-7-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[5]
[pool-3-thread-4] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[6]
[pool-5-thread-4] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[6]
[pool-7-thread-4] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[6]
[pool-3-thread-3] INFO istic.aoc.m3.afficheur.Afficheur - Rouge MAJ : Optional[7]
[pool-5-thread-1] INFO istic.aoc.m3.afficheur.Afficheur - Vert MAJ : Optional[7]
[pool-7-thread-2] INFO istic.aoc.m3.afficheur.Afficheur - Bleu MAJ : Optional[7]
```