

Module AOC – Master 2 MITIC

Rapport TP : Métronome

Binôme : Dimitri LANOE – Dounia SAKHRAOUI

2014/2015

1 .Introduction

Dans le but de la mise en œuvre des patrons de conception en Orienté Objet, il nous ait proposé de mettre en œuvre une application de type métronome. Le projet proposé se traduit par la réalisation de deux versions distinctes. Ce document explique dans un premier temps, les étapes de conception et d'implémentation du métronome ainsi que les tests effectués.

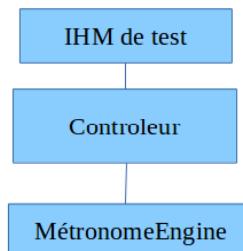
2 .Problématique

Ce qui est demandé de faire dans ce projet est de réaliser une application de type métronome. Il s'agit d'un appareil qui effectue une action spécifique à intervalles réguliers afin d'indiquer le tempo de manière fiable et stable. Dans ce projet, l'action à opérer est l'émission d'un signal sonore ou lumineux.

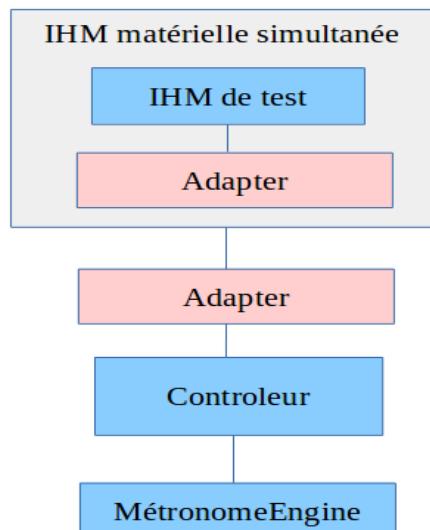
3 .Conception

La conception de ce métronome passe par plusieurs étapes, dont 2 étapes majeures qui ont pour résultat la réalisation de la version 1 puis la réalisation de la version 2 de l'appareil.

- **La version 1 :** c'est une version de test, elle permet de savoir si le développement effectué répond au besoins de l'utilisateur ou pas. Cette version est fonctionnelle sur ordinateur.

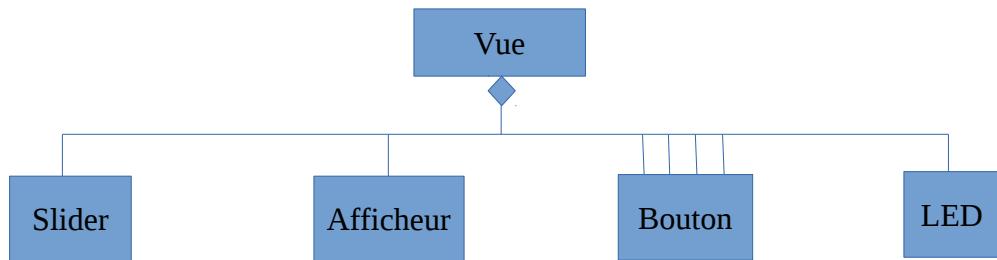


- **La version 2 :** elle représente la version finale qui sera livrée. Elle est constituée de la version 1 améliorée. Cette version est conçue de manière à pouvoir s'adapter à un équipement (matériel physique).



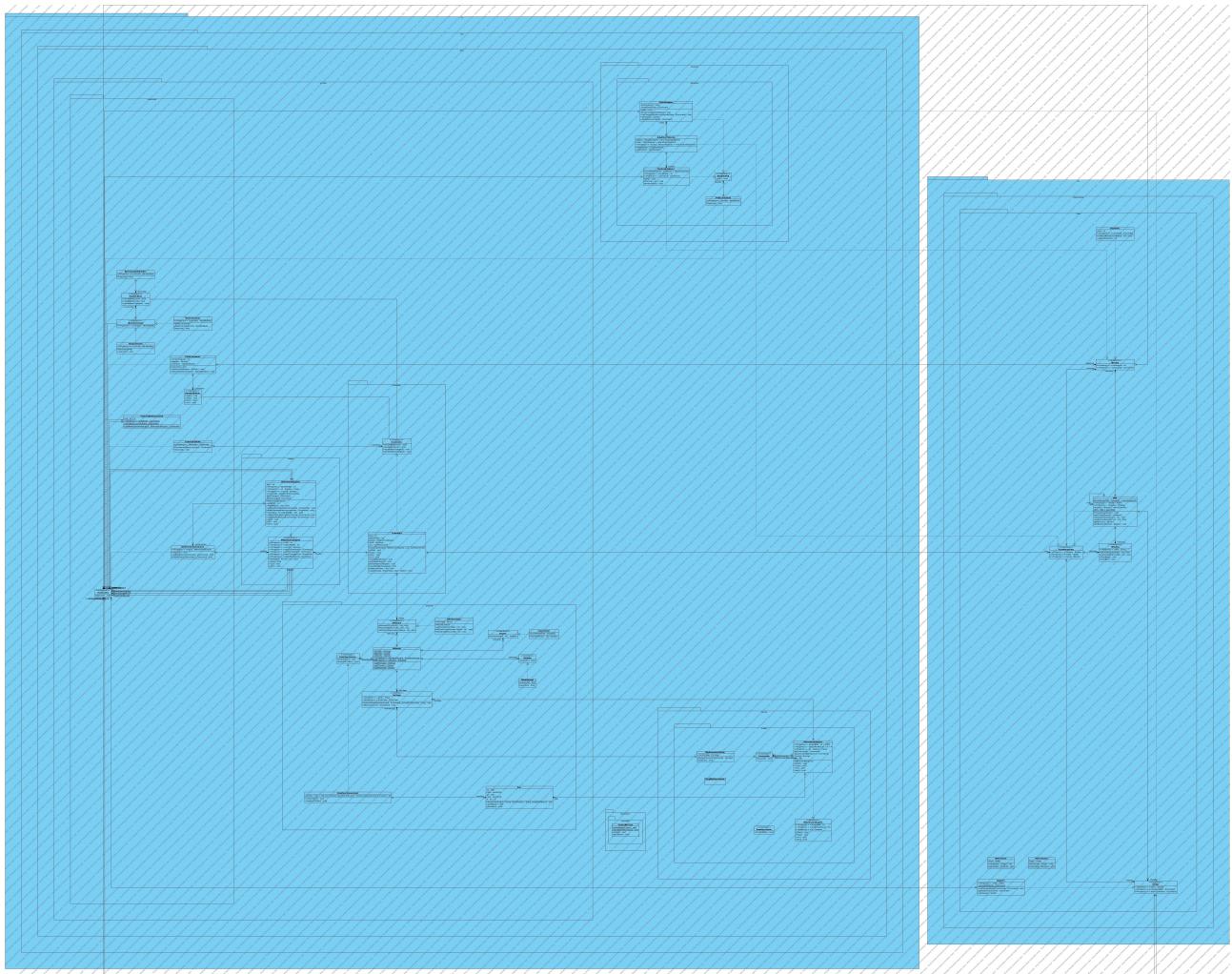
3.1 Conception de l'IHM

La Conception de l'IHM est effectué indépendamment de la plate-forme du métronome. Cette étape consiste à concevoir une vue statique qui permet de présenter les composants d'interaction du métronome.



La vue est constituée d'un **Slider**, pour régler le tempo (le nombre de temps par minute), un **Afficheur**, pour afficher la valeur du tempo en cours, 4 **Boutons**, deux pour démarrer ou arrêter le métronome, et deux pour incrémenter ou décrémenter le nombre de temps par mesure.

3.2 Diagrammes de classes



3.3 Patrons de conception

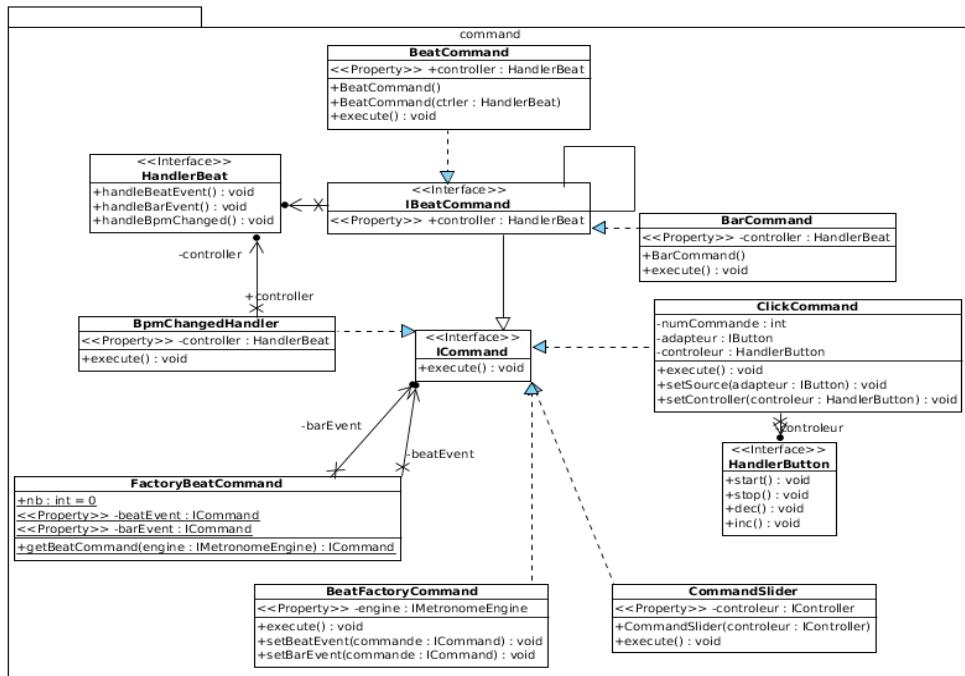
Les patrons de conception définissent les interactions entre des classes et des objets. C'est des outils indépendant du langage de programmation et du paradigme. Ils sont généralement utilisé comme solution à un problème courant en conception. Ils permettent l'amélioration de l'efficacité, la robustesse, le temps de développement et la lisibilité du code.

Dans ce TP, le choix s'est porté sur l'utilisation des patterns suivants :

Command, Observer et Adapter.

3.3.1 Command

C'est un patron de conception de type comportemental. Il permet de séparer le code initiateur de l'action du code de l'action elle-même.



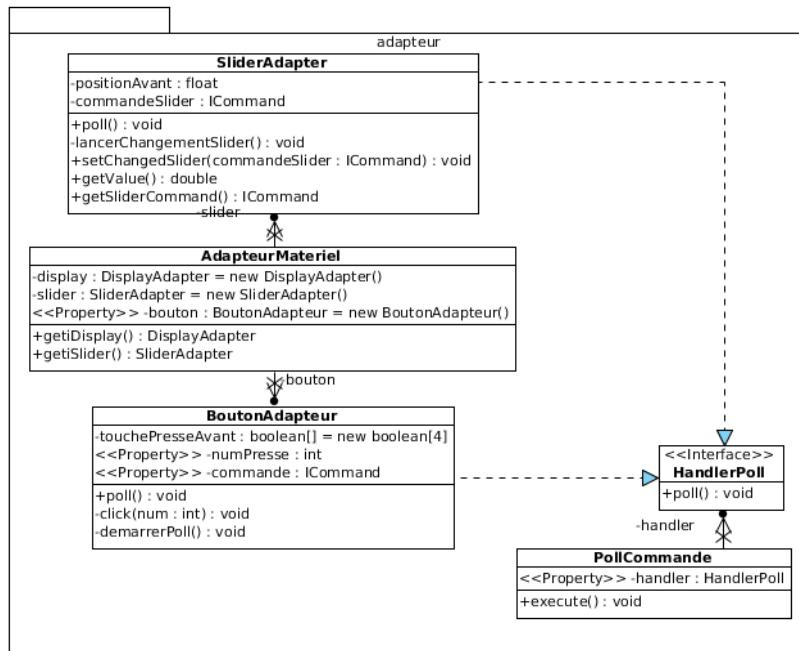
3.3.2 Observer

Le patron de conception observer est utilisé pour envoyer un signal à des modules qui jouent le rôle d'observateurs. En cas de notification, les observateurs effectuent alors l'action adéquate en fonction des informations qui parviennent depuis les modules qu'ils observent.

3.3.3 Adapter

Le patron de conception Adapter est un moyen commode de faire fonctionner un objet avec une interface qu'il ne possède pas. L'idée est de concevoir une classe supplémentaire qui se charge d'implémenter la bonne interface (L'Adaptateur) et d'appeler les méthodes correspondantes dans l'objet à utiliser (L'adapté).

Ce patron de conception a été utilisé afin de construire la version 2 du métronome.



4 .Implémentation

4.1 Outils utilisés

L'implémentation a été effectué en utilisant le langage Java. L'interface du métronome quand à elle a été réalisée à l'aide de JavaFX.

L'IHM statique a été réalisée à l'aide de JavaFX Scène Builder.

4.2 Tests et Validation

Les tests ont été réalisés en utilisant le framework JUnit.

4.2.1 Tests effectués

4.2.1.1 Tests du Métronome Engine

Les tests se sont rapporté aux méthodes permettant d'allumer, d'éteindre le métronome, d'incrémenter et de décrémenter la valeur du tempo. La classe « MetronomeEngineTest » effectue ces tests, en veillant à respecter les contraintes liées aux valeurs maximales du tempo.

4.2.1.2 Tests du contrôleur

Ces tests implémentés par la classe « ControllerTest » permettent de vérifier que le contrôleur remplit bien ses missions lorsqu'un événement survient.

4.2.1.3 Tests du matériel

Cette série de tests vise l'afficheur, le clavier, l'horloge et la molette.

- **Afficheur** : à travers la classe « AfficheurImplTest », les tests permettent de voir si l'afficheur affiche une valeur demandée et s'il est capable d'afficher et d'éteindre les deux LEDs à la demande.

- **Clavier** : la classe « `ClavierImplTest` » permet de vérifier que le clavier peut modifier l'interface, par exemple, vérifie si les touches des deux flèches de directions modifient la valeur du tempo.

- **Horloge** : la classe « `HorlogeImplTest` » vérifie l'activation périodique de l'horloge.

- **Molette** : la classe « `MoletteImplTest` » vérifie la position de la molette.

4.2.2 Tests réussis

Les tests précédent ont tous était passé avec succès.

4.3 JavaDoc

L'intégralité de la JavaDoc est stockée dans le dossier `<Doc>` situé à la racine du projet.

5 .Difficultés rencontrées

Les principales difficultés ont été rencontrées dans la phase d'analyse. La mise en place des patrons de conception n'est pas évidante de prime à bord. Les phases d'implémentation et de test ont été réalisées avec aisance.

6 .Conclusion

Ce projet nous a permis d'enrichir notre expérience dans le domaine de la conception orienté objet. En effet, l'emploi des patrons de conception nous a permis de concevoir des applications dont le code est lisible, facilement extensible et qui peut évoluer.

Ce projet nous a également permis d'appliquer la notion d'injection de dépendance en utilisant le frame-work JavaFX.