Université Du Québec À Montréal

Rapport: projet pratique 2 - phase2
Jeu Takénoko

Équipe: J

1. Yazid Iharkane (IHAY23088102)
2. Farahnaz Kashandeh K.
3. Kokou Eyram Joel Afetse (AFEK11069706)

Travail présenté à:
Madame Imen Benzarti
Dans le cadre du cours:
Inf5153 - Génie logiciel: conception

Hiver 2020

Table de matière

Page

1- Introduction

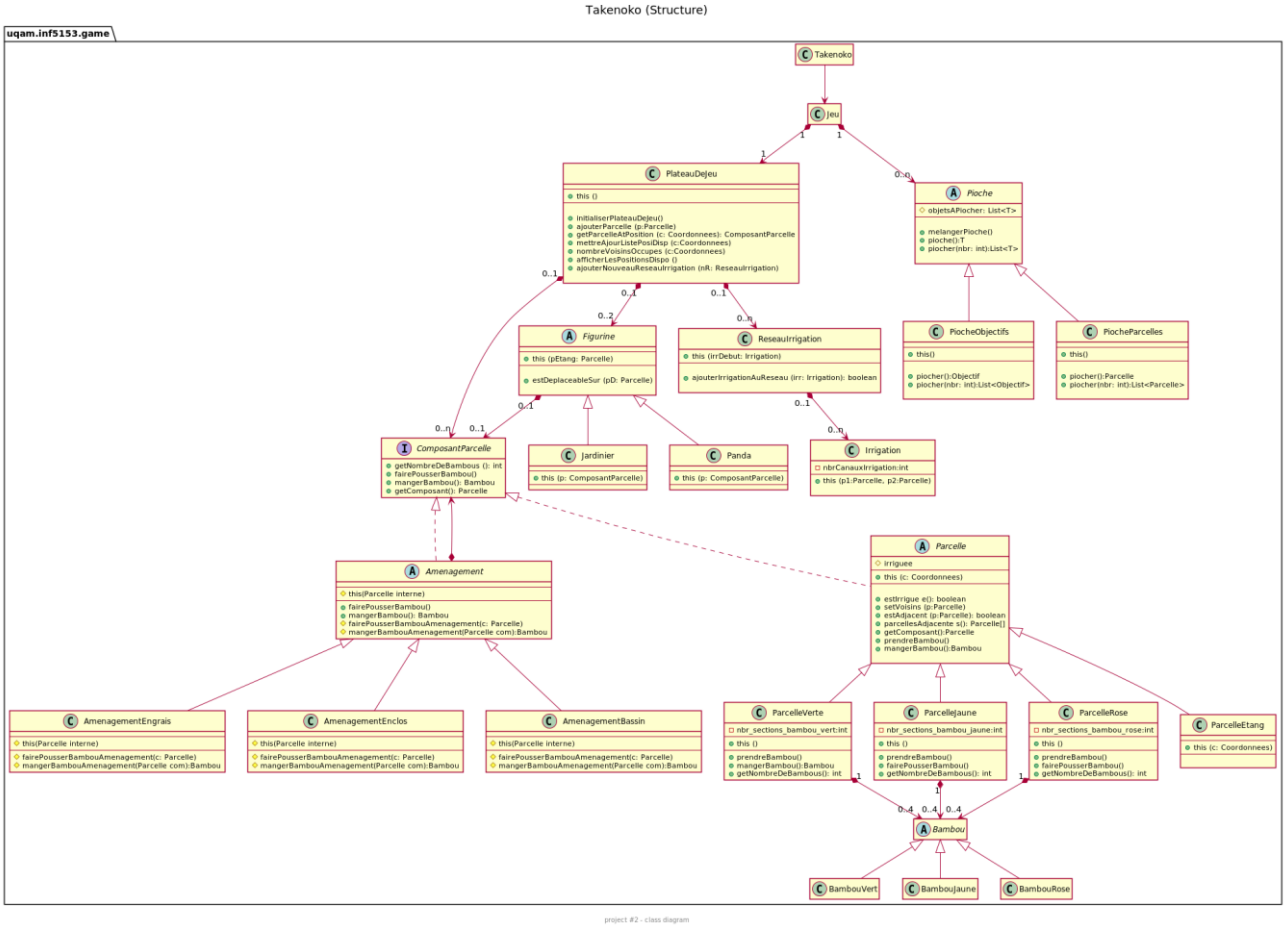
Notre objectif est de réaliser la conception et le développement de jeu de société «Takeneko». Pour l'atteindre, le travail était devisé en deux phases. La première phase a été consacrée à la production de la version minimale de jeu. Par contre, la deuxième qui est le sujet de ce travail pratique est consacrée à la finalisation de jeu, en respectant les règles du génie logiciel et en appliquant les patrons de conception en cas de besoin.

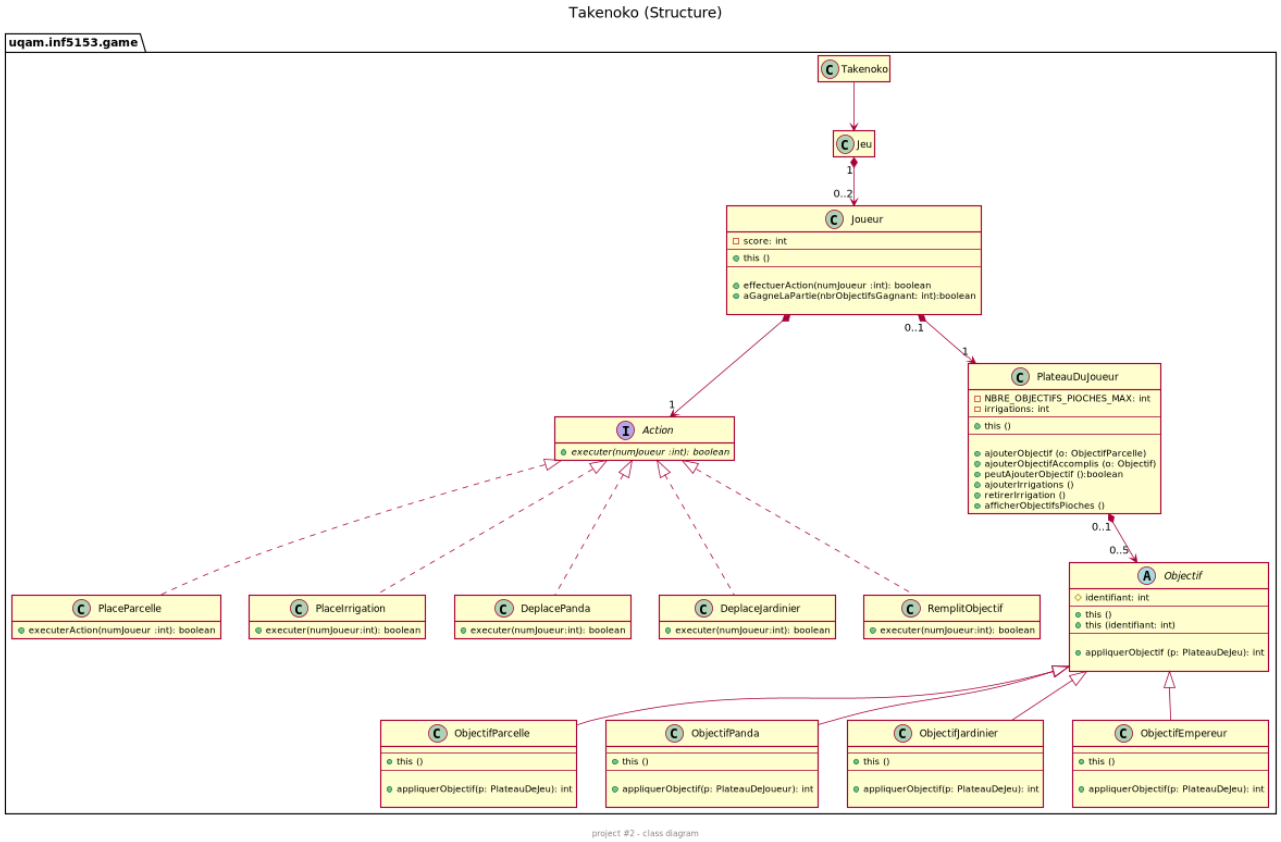
Ce rapport contient trois parties essentielles. Dans la première, nous avons illustré notre conception en utilisant le diagramme de classes, ainsi que celui de séquences. Dans la deuxième, nous avons décrit les points forts et les points faibles de notre conception. En fin, dans la troisième, nous avons parlé des évolutions prioritaires qui pourront être mises en place si nous déciderons de finaliser la construction de ce jeu.

2- Modèles

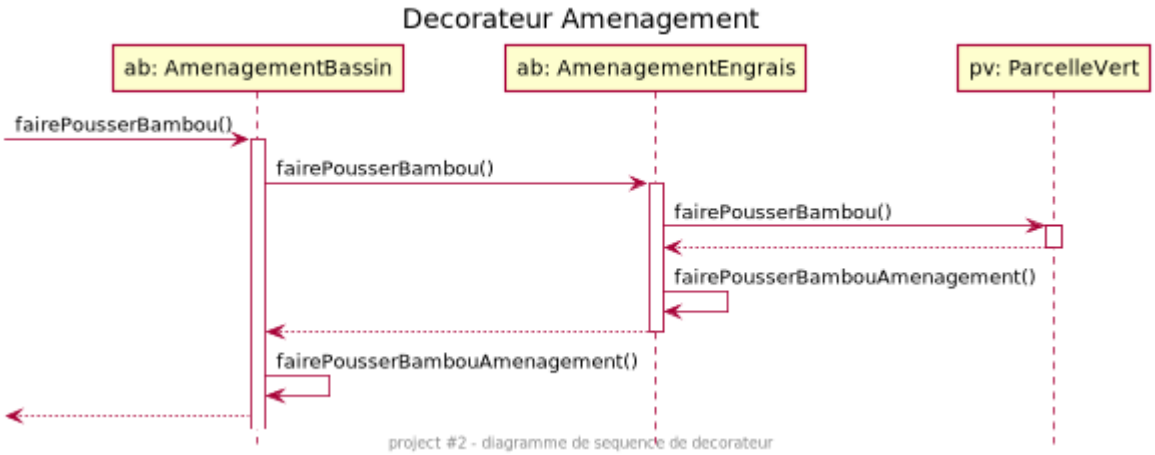
2-1- Les modèles de conceptions pertinents

A- Digramme de classe





B- Diagramme de séquence décrivant le patron décorateur



2-2- La justification de pertinence de modèles de conception utilisés

Les modifications faites dans cette deuxième étape de travail se divisent en deux principales parties. Premièrement, dans la première phase de travail, quatre principes SOLID ont été respectés. À savoir les principes S, O, L et D. Aussi, nous avons appliqué, à notre conception, les patrons GRASP suivant: spécialité de l'information, faible couplage, forte cohésion, contrôleur et polymorphisme. Ce qui nous a permis d'avoir une conception ouverte à l'extension et fermée à la modification. En effet, nous avons ajouté des fonctionnalités sans modifier la structure générale de notre code. Par exemple, nous avons une classe abstraite **Figurine** et nous avons ajouté la classe **Panda** sans modifier les autres classes de logiciel. Aussi, nous avons une classe abstraite **Objectif** et la classe concrète **ObjectifParcelle**. Nous avons ajouté les deux classes concrètes **ObjectifJardinier** et **ObjectifPanda** sans toucher aux autres classes. À la fin de la première partie, la seule classe qui avait plusieurs responsabilités était la classe **Joueur**. Pour remédier à ce problème, dans le cadre de cette deuxième partie, nous avons implémenté le patron de conception **Stratégie** et le patron GRASP **Indirection.** En effet, nous avons eu besoin d'un patron, avec lequel un comportement de classe ou son algorithme peut être modifié au moment de l'exécution. Toujours dans ce contexte, nous avons créé une interface **Action** et cinq classes concrètes à savoir: **DeplaceJardinier**, **DeplacePanda**, **PlaceParcelle**, **PlaceIrrigation** et **RemplitObjectif** qui implémentent cette interface et représentent diverses stratégies. Ce modèle nous a aidé à avoir considérablement des classes avec la responsabilité unique et aussi à avoir de faible couplage entre la classe **Joueur** et les autres éléments. Deuxièmement, l'une des tâches à faire dans le cadre de la deuxième phase de travail, c'est d'ajouter les aménagements aux parcelles. Pour le faire, nous avons utilisé le patron de conception **Décorateur**. Ce patron nous a permis d'ajouter des nouvelles fonctionnalités à l'objet existant parcelle sans modifier sa structure. Ce type de patrons relève du modèle structurel. Il agit en tant qu'enveloppeur de la classe existante. Pour l'implémenter, nous avons créé une classe décoratrice **Aménagement** qui enveloppe la classe d'origine **Parcelle** et fournit des fonctionnalités supplémentaires en conservant la signature des méthodes de classe.

3- Analyse critique du projet:

3-1- Forces

- Le projet est bien découpé avec différents concepts distincts.
- Le projet a été découpé de manière à ce que nous puissions ajouter de nouvelles fonctionnalités sans modifier notre code actuel. En gros, il est ouvert à l'extension. Donc il y a la possibilité d'ajouter les conditions climatiques.
- Nous avons réussi à implémenter les différentes fonctionnalités demandées
- Notre projet est suffisamment testé.
- La classe Joueur avait beaucoup de responsabilités avant l'implémentation du patron Stratégie, On a profité de ce patron pour créer un classe Action et éviter un couplage immédiat entre plusieurs éléments et aussi on a appliqué le patron Décorateur qui rend notre code plus cohérant.
- Cette conception est évaluative, et on peut l'appliquer aux jeux semblables.
- Notre encapsulation dans la majorité de classes est asses bien, particulièrement dans les classes concrètes qui implémentent l'interface **Action** et qui héritent de la classe **Amenagement**.

3-2- Faiblesses

- Notre encapsulation a été un peu brisé à certains endroits dans le code.
- Nous pensons que la cohésion entre certains nos classes n'est pas assez forte.

4- Les évolutions prioritaires qui doivent être mise en place par la suite

A la fin de cette deuxième phase de travail, il reste à ajouter les objectifs Empereur et les conditions Climatiques. Donc, notre suggestion pour un éventuel ajout, c'est de développer les fonctionnalités qui permettront d'utiliser les conditions climatiques.