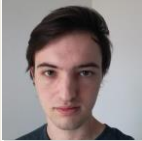
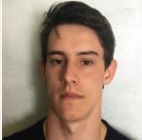
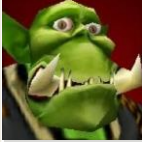




A large, thick, black L-shaped frame that starts at the top-left corner and extends towards the bottom-right corner, framing the central text.

PROJET 7WONDERS

Équipe Top4 + 1

Expertise des membres

	NOM Prénom	Expertise
	TOGNETTI Yohann	Fonctionnalités et éléments du jeu ainsi que son déroulement.
	MARTIN D'ESCRIENNE Yann	Fonctionnalités et éléments du jeu ainsi que son déroulement.
	JEROME Maxime	Communication serveur de statistiques – jeu, Statistiques
	BEN FREDJ Yasmine	Intelligence artificielle et fonctionnalités du jeu.
	MARRO Sébastien	Mainteneur du repo Github, éléments du jeu.

Synthèse du projet

- ✓ L'application demandée est fonctionnelle, toutes les fonctionnalités du jeu ont été implémentées, des fonctionnalités hors-sujet prévues ou bonus ont été abandonnées
- ✓ Nous étions confiant sur la bonne réalisation de ce projet malgré quelques légers retards sur certaines itérations
- ✓ Le code et les tests sont en général de bonne qualité, mis à part quelques classes notamment ***CardFactory*** et ses tests.

Organisation et justification générale du code

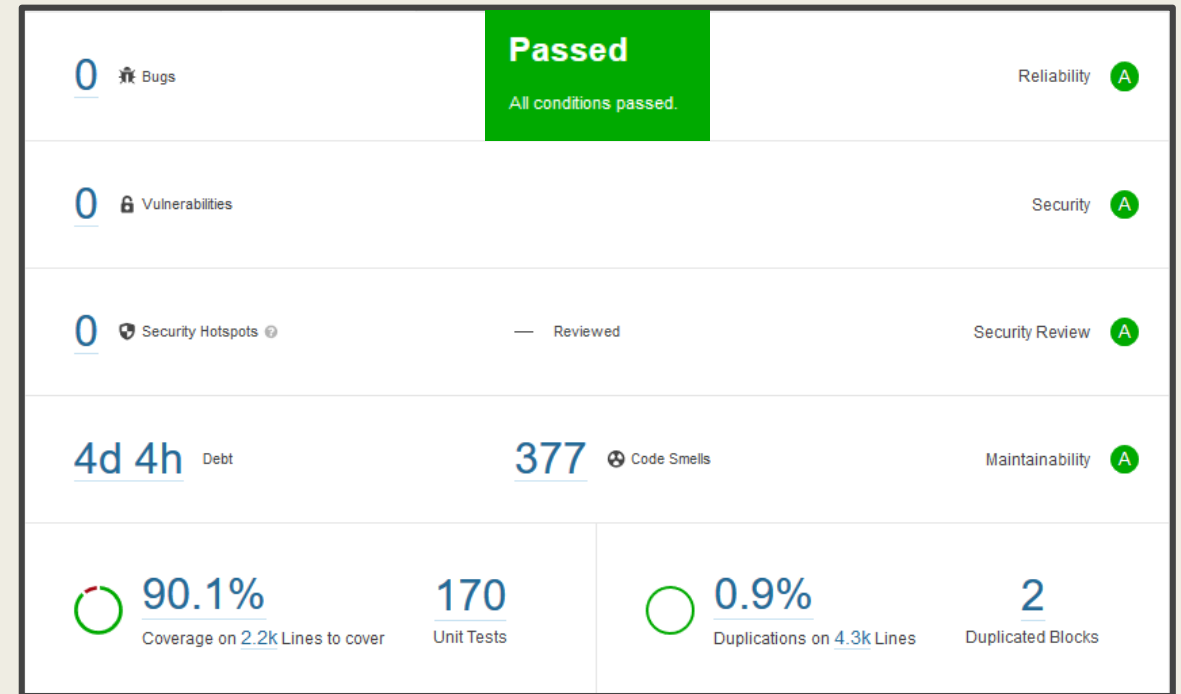
- Le module *client* représente les différentes intelligences artificielles du jeu
- Le module *commun* représente les données accessibles entre tous les modules
- Le module *gameserver* représente le jeu et son déroulement
- Le module *statsserver* représente le serveur de gestion de statistiques
- Le jeu demande séquentiellement à chaque joueur son action et attend la réponse de chacun, afin de simuler les choix en temps réel de chaque joueur
- Notre code essaye au maximum d'utiliser des interfaces, héritages et de la modularité afin d'avoir un faible couplage et une bonne cohésion.

Patrons de conception (Design Patterns)

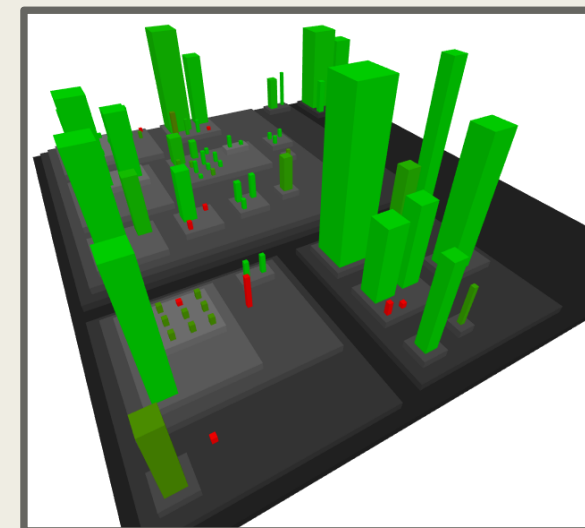
- (Strategy) Les joueurs (*IA*) implémentent tous une interface qui définit les actions possibles, cela permet de rendre plus simple les modifications de celles-ci et la maintenabilité côté jeu. Les effets et les coûts ont aussi ce DP, cela permet une meilleure maintenabilité, manipulation et compréhension de code
- (Template) Les statistiques (*Dealers & Objects*) ont besoin d'un DP Template pour une meilleure maintenabilité, manipulation et compréhension de code
- (Singleton) La gestion de l'aléatoire est représenté par un singleton afin d'éviter tout bug provenant de la PRNG Java, aussi, l'objet des statistiques est un singleton, ce qui permet de ne pas mettre en argument cet objet dans des fonctions qui, a priori, ne doivent pas le mettre
- (Factory) La génération des paquets de cartes et des merveilles se font en fonction du nombre de joueurs dans la partie
- (Decorator) Les demandes faites au joueur passent par un DP Decorator pour vérifier la réponse de l'IA.

Analyse métrique

- La couverture des tests est très suffisante
- Aucun bug, aucune vulnérabilité et aucun hotspot de sécurité n'ont été trouvé par *SonarQube*
- La duplication de code est très faible
- La dette est faible par rapport aux *Code Smells* fournis



Coverage 3d donné par SonarQube



Notre projet est un bon projet !

- ✓ Une bonne maintenabilité de code avec les interfaces
- ✓ Un code commenté et une documentation explicite et à jour
- ✓ Un coverage supérieur à 90% et des tests complets
- ✓ Une bonne gestion des patrons de conception
- ✓ Une organisation en sous package et package avec dépendance

Pourquoi notre projet est-il un mauvais projet ?

- X Le retard sur certaines itérations a entraîné des dettes techniques qui ont dû être corrigées sur les suivantes, entraînant ainsi une perte de temps.
- X Certaines classes ont accès à des champs mal gérés comme Player qui a accès à ces voisins sans abstraction.
- X Notre projet manque de fichiers de ressources externes pour les cartes ou encore les merveilles
- X Les factories n'ont pas une implémentation pertinente des fonctions (switch-cases).
- X Classes inutiles (*ChoiceMaterialEffect*, *EarnWithCardEffect*, ...)

Rétrospective du projet

Pour un temps plein sur le projet nous aurions gardé :

- L'utilisation de SonarQube
- L'utilisation du kanban de github et l'utilisation des tickets
- L'utilisation des branches de développement

Nous aurions amélioré ou aimé mettre en place :

- La mise en place de plus de tests croisés
- Mise en place d'une vraie séparation client - serveur pour les joueurs et le jeu
- Mise en place d'une base de donnée pour la gestion des statistiques
- Un coverage toujours à 70% à chaque itérations.
- L'utilisation d'un diagramme de classe

Nous aurions arrêté de :

- Faire des commits pour régler les bugs d'un précédent commit



**Merci pour votre
attention**

