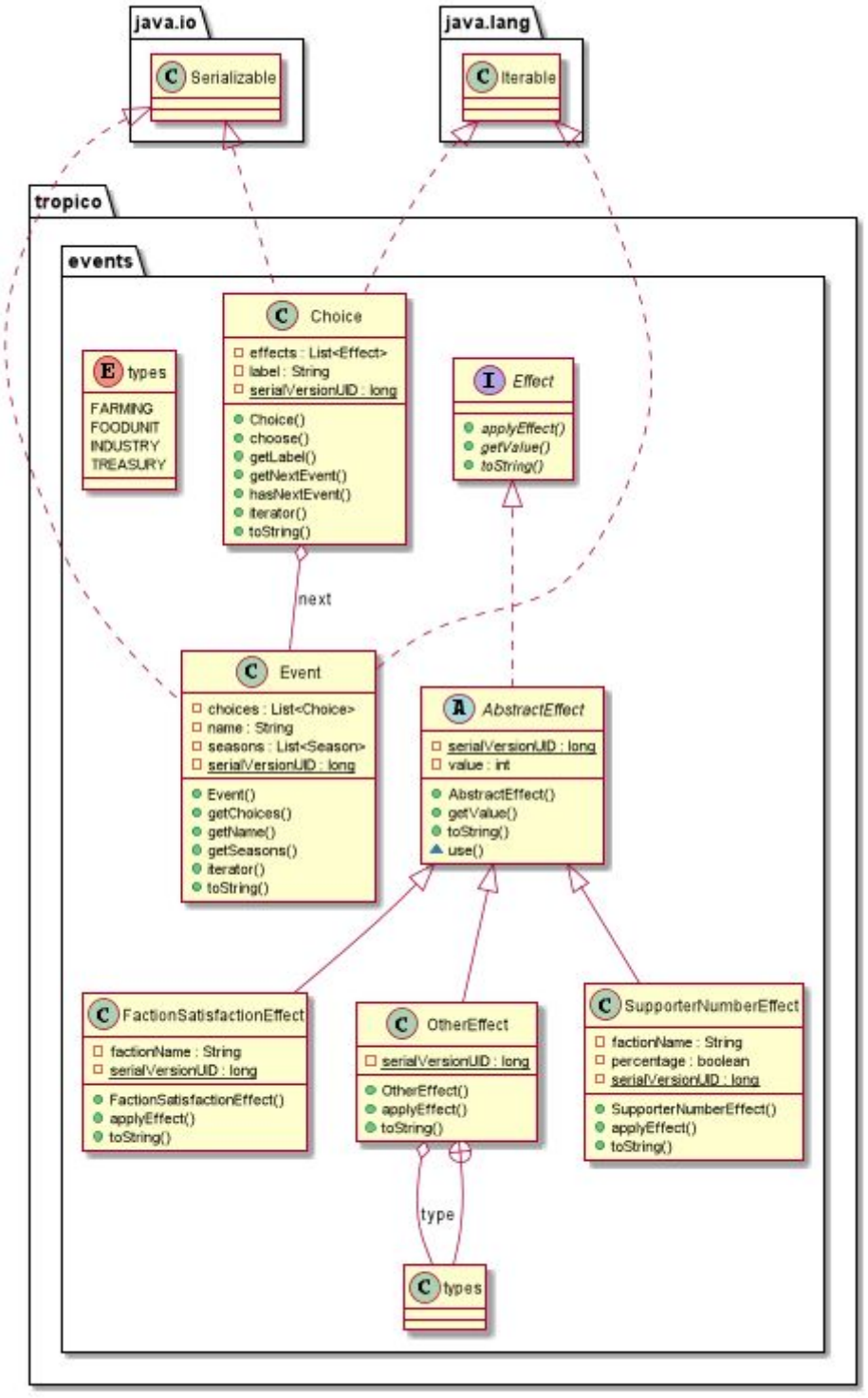
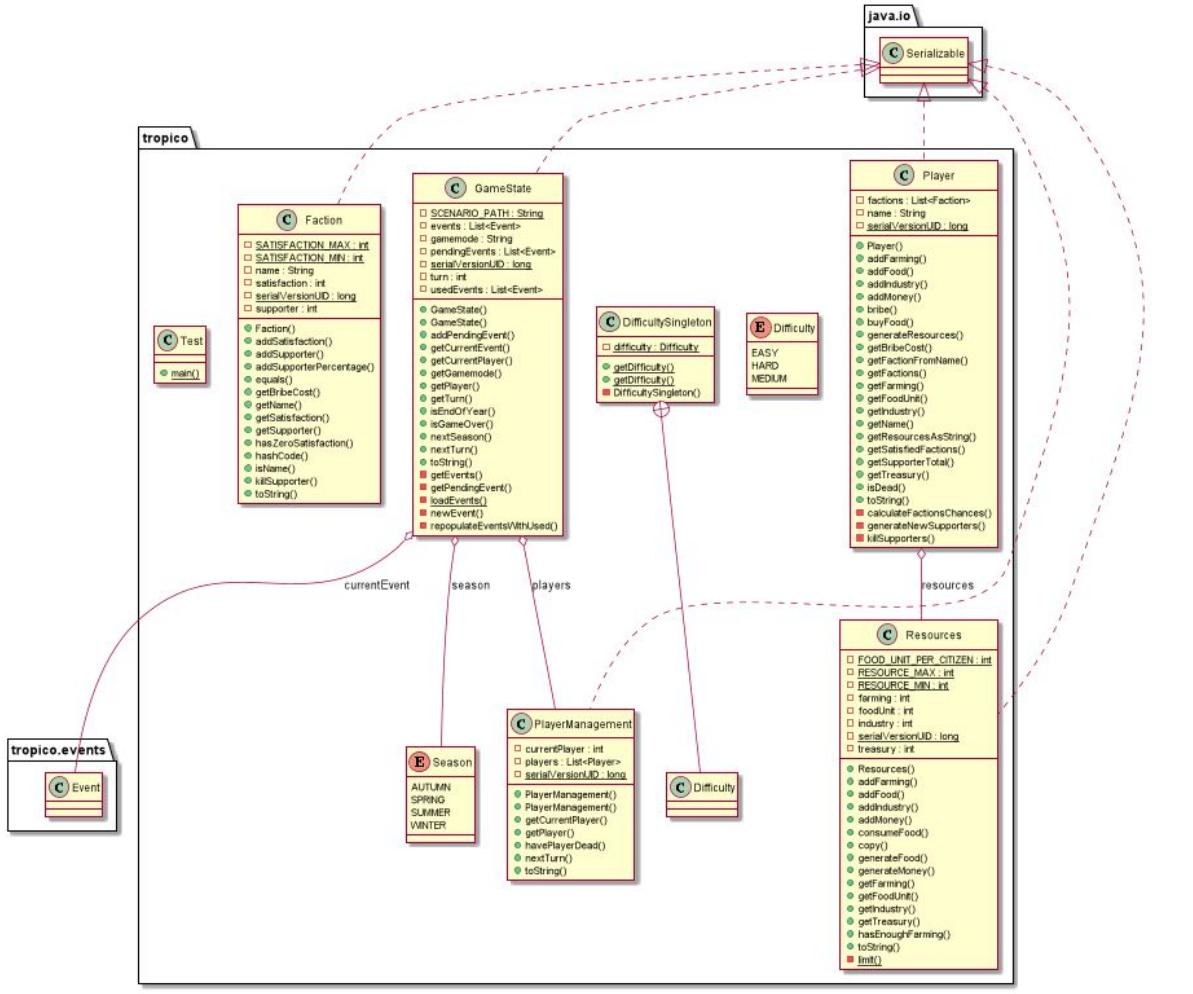
Cahier de Conception Technique

Par DOS SANTOS Sébastien et OGER Corentin

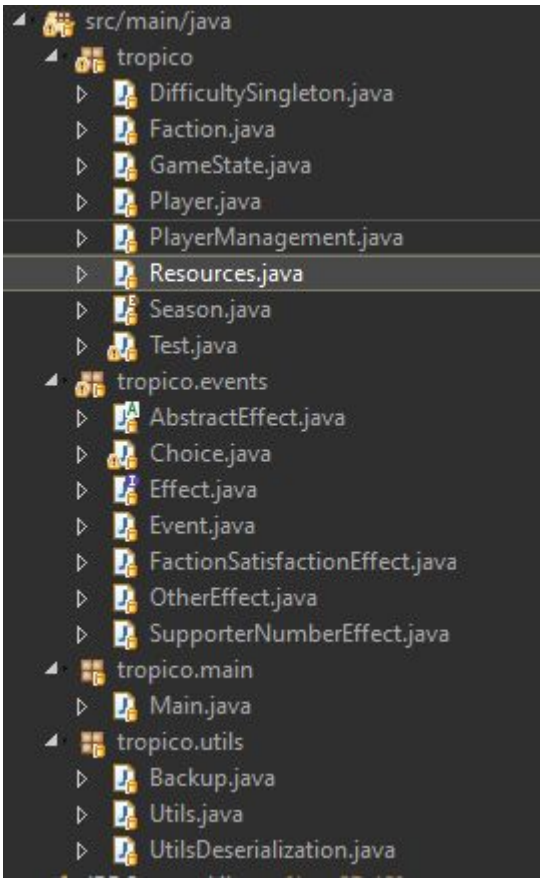
EVENTS's Class Diagram



TROPICO's Class Diagram



Architecture du projet:



Dans un premier package se trouve de nombreuses classes du jeu contenant la majorité des informations : joueurs, difficulté, ressources etc.

Dans le second, tropico.events, on trouve tout ce qui est liés aux événements, c'est-à-dire les événements en eux-mêmes, les choix, et les effets. Pour les effets, il était plus pratique de faire une interface ainsi qu'une classe abstraite, pour éviter de dupliquer du code.

Tropico.main contient la classe main, où les actions, (liées aux gamestate) sont appelées.

Notre main comporte les différentes boucles de jeu tel qu'un menu ou bien la boucle principale du jeu.

Nous avons décidé de créer une classe GameState contenant les variables utiles pendant la partie tel que les événements, la saison ou bien les joueurs.

Enfin, tropico.utils contient des utilitaires, telle qu'une classe de désérialisation pour les json, une classe utils contenant quelques fonctions générales, ou encore la classe backup, utilisée pour la sauvegarde.

Difficultés rencontrées:

Nous avons rencontré quelques difficultés lors de ce projet.
Premièrement, lié à la période d'isolation que nous vivons, il a été plus difficile de se voir, et il a donc fallu s'adapter, en se servant uniquement d'outils de communication à distance. Nous avons donc communiqué via discord, en vocal, comme à l'écrit, et il en va de même avec nos professeurs. De plus, nous avons créé un projet git pour faciliter les échanges et la mise en commun.
Deuxièmement, des difficultés d'ordre technique. En parlant de difficulté justement, celle-ci fut un problème car nous ne savions pas comment la gérer efficacement. Il nous a alors été suggéré d'utiliser un singleton, ce qui nous a été d'une grande aide. La désérialisation nous a fait découvrir le fonctionnement des json, et l'affichage, que nous avions au départ voulu faire sur javafx, a été abandonné au profit d'une interface sur console. Celle-ci, bien que moins gracieuse, aura su nous faire gagner du temps.

Répartition des tâches:

Cela a plus ou moins été du 50/50.
Le début du projet, notamment sa structure et ses fondations, ont été gérés par Sébastien. Il a aussi géré la désérialisation, la gestion des joueurs ou encore de la sauvegarde.
De mon côté, Corentin, comme vous pouvez le constater j'écris ce rapport, bien que je ne sois pas seul à le faire, et je me suis aussi chargé d'une partie de la javadoc, du readme et changelog. Côté programmation, je me suis occupé principalement des effets, d'améliorer certaines méthodes, et de la gestion de la difficulté.
Pour le reste, multijoueur, saisons, boucle principale du jeu, etc. Il serait difficile de l'attribuer à l'un de nous en particulier.

Ce qui a été implémenté:

La grande majorité des fonctionnalités demandées ont été ajoutées. Il reste néanmoins quelques bugs, certains connus, d'autres non.
Pour n'en citer qu'une partie, nous avons implémenté les événements, le choix de la difficulté et du scénario, le multijoueur, les événements chaînés etc.
On peut aussi citer le bonus sur la répartition des partisans, que nous avons trouvé intéressant de rajouter.
Pour ne pas allonger ce rapport, il sera plus facile de parler de ce qui n'a pas été implémenté.

Ce qui reste à implémenter:

Il manque peu de choses à implémenter, en effet, dans les tâches obligatoires, il manque seulement les événements qui affectent l'autre joueur lors d'une partie à deux. Cependant, nous avons créé très peu d'événements, il est donc possible qu'il y est des bugs que nous n'avions pas remarqué, et ce manque d'événements est un point faible de notre projet.

De plus on peut y ajouter les bonus que nous avons décidé de ne pas faire (à l'exception de celui cité précédemment).

Contenu supplémentaire et choix techniques :

Nous avons parfois choisi certaines options par rapport à d'autres, qui n'étaient pas forcément prévues dans la consigne.
Par exemple, nous avons décidé qu'en multijoueur, la partie est finie quand il n'y a plus qu'un joueur.
De plus, le nombre de morts parmi les partisans en fin d'année est calculé selon la nourriture. Nous trouvions cela plus cohérent que de faire en fonction de l'agriculture (ce qui est tout de même fait pour la génération de nouveaux partisans).
Dans le même thème, pour faire raccord avec le bonus sur la répartition des partisans, lorsqu'un partisan meurt en fin d'année (ou plusieurs), la faction qui perd ce partisan est calculée selon son nombre de partisans par rapport au total. Une grande faction aura donc plus de chance de perdre un partisan qu'une petite.
Enfin, il est possible grâce à notre structure, d'ajouter facilement du contenu, comme des événements ou des scénarios. Nous avons même un script

python qui permet via une autre interface console de créer le json des événements. Il est ainsi possible de “modder” notre projet assez facilement.

Il y a d’autres fonctionnalités pratiques dont on pourrait parler ici, mais elles sont moindres et ne valent pas le coût d’en discuter longuement.

Perspectives d’évolutions:

Cette section est destinée aux perspectives d’évolutions, et des conseils pour un développeur souhaitant modifier notre projet.
Premièrement, comme dit plus tôt, nous n’avons pas d’interface graphique. Il serait donc intéressant d’en créer une. La tâche ne sera pas facile pour autant, la majorité se transforme plutôt bien, mais cela pourrait être plus difficile, car certaines classes affichent des informations qui sont simples à afficher sur console.
Une autre idée serait de rajouter les contenus manquants, certains sont relativement facile à intégrer, mais d’autres peuvent causer plus de problèmes. Il est aussi possible de corriger les bugs (voir le readme).
Ensuite, le manque de contenu est un problème, mais il est très facile à régler grâce à notre structure. Il est donc conseillé de rajouter du contenu tel que conseillé dans le readme.
Pour le multijoueur, on ne peut actuellement jouer qu’à deux, mais en changeant légèrement la fonction “havePlayerDead” de la classe PlayerManagement, il doit être possible de faire en sorte que le jeu soit jouable à plus que deux.
Enfin, il y a un champ “next” dans la classe choice, mais aussi dans les json. Cela peut poser problème dans le cadre d’effet en chaîne, qui rendrait le json bien moins lisible. Ce travail est plus conséquent, mais il en vaut le coût.

Conclusion:

Merci d’avoir lu ce rapport, nous espérons que notre travail vous aura plu, de notre côté, il aura été sympathique (même si la documentation n’est pas notre partie favorite).

Nous vous souhaitons une bonne journée,

Corentin & Sébastien