***Conquête***

*implémentation système de tours*

# I.Classe Action

La classe Action sera une classe abstraite servant de base aux différentes actions possibles par les joueurs

3 attributs :

* Power power
* Int positionX
* Int positionY

*(Une classe Positition aurait pu être utilisée ici, mais il me semble préférable de la garder pour l’interface)*

🡺Sera une classe de base pour ces sous-classes :

1. ActionMakeAlliance
2. ActionBreakAlliance
3. ActionAttack
4. ActionMove
5. ActionConstruct
6. ActionCreateUnits
7. ActionDestroyBuilding
8. ActionDestroyUnits

*[penser à faire un ActionTypes, comme pour Building et Resource]*

# II.ActionManager

ActionManager sera celui qui créera les Action avec des méthodes statiques. Elle sera instanciable et aura comme attribut la Map

* renverra une exception (de type IllegalArgumentException ou une créée pour l’occasion) si les consitions ne sont pas remplies
* Renverra la Classe Action[…] demandée sinon

Ex :



# III. Classe GameLoop (nom non définitif)

C’est cette classe qui va gérer le déroulement du jeu.

Elle possèdera un tableau de 8 ArrayList, chaque arrayList étant une arralist d’une des actions des joueurs (dans le même ordre que dans I.)

Donc :

🡺 Tant que le joueur à des actions/ qu’il décide de jouer :

1. Le joueur choisit de faire une action avec sa souris
2. L’actionManager est appelé pour décider si l’action est possible ou non.
   1. Si non, rien ne se passe vraiment (le message du throw sera envoyé en pop-up sur l’interface)
   2. Si oui, Le nombre d’actions du joueur est décrémenté et l’action est créée et ajoutée dans l’arrayList de la bonne case
3. On recommence

🡺 Les autres « joueurs » font de même, exactement le même processus

🡺Fin du tour

1. Itération dans le tableau d’arrayList
   1. Itération dans l’arrayList, c’est à ce moment que les actions sont réelement réalisées, et dans l’ordre convenu dans le cahier des charges (si je ne me trompe pas)
2. Ajout/diminution ressources 🡺 Si norriture < 0, soldats qui meurent auto