Concept Objet – M2 Novembre 2019

**Projet « nom du projet »**

BOMY Clara – CHRISTIAENS Mathilde





Table des matières

[Introduction 3](#_Toc23088833)

[1 Description du projet 3](#_Toc23088834)

[1.1 Brève description 3](#_Toc23088835)

[1.2 Fonctionnalités 4](#_Toc23088836)

[1.2.1 La carte 4](#_Toc23088837)

[1.2.2 Simulation 4](#_Toc23088838)

[1.2.3 Individus 4](#_Toc23088839)

[1.2.4 Fonctionnalités supplémentaires 4](#_Toc23088840)

[2 Conception 4](#_Toc23088841)

[2.1 Architecture du projet 4](#_Toc23088842)

[2.2 Intégration des concepts objets fondamentaux 4](#_Toc23088843)

[2.3 Diagramme de classes 5](#_Toc23088844)

[2.4 Visualisation de la simulation 5](#_Toc23088845)

[2.5 Implémentation des fonctionnalités supplémentaires 5](#_Toc23088846)

[3 Déroulement d’une simulation 5](#_Toc23088847)

[3.1 Déroulé d’un tour pour un personnage 5](#_Toc23088848)

[3.2 Fonctionnement de la boucle principale 5](#_Toc23088849)

[4 Analyse 5](#_Toc23088850)

[4.1 Choix faits en termes d’organisation 5](#_Toc23088851)

[4.2 Répartition des rôles 6](#_Toc23088852)

[4.3 Bilan 6](#_Toc23088853)

[4.3.1 Difficultés rencontrées 6](#_Toc23088854)

[4.3.2 Points de satisfaction 6](#_Toc23088855)

[Conclusion 6](#_Toc23088856)

# Introduction

Dans le cadre de notre cours de *Concept Objet*, il nous a été demandé de réaliser une simulation discrète à pas de temps constant mettant en place un système multi-agents (SMA). Ce projet nous permettra ainsi de mettre en application les nombreux concepts traités durant le cours.

Nous avons réalisé « nom Projet », projet mettant en scène un microcosme dans lequel quatre populations inspirées de l’univers de Game Of Thrones évoluent sur une « carte », *Westeros*. Tout comme dans l’œuvre originale, ces factions seront amenées à se battre ou à s’entraider dépendamment de leurs rencontres et de leurs affinités. A l’issu de la simulation, nous pourrons alors découvrir la faction la plus apte à régner sur *Westeros*.

Pour en faciliter la gestion, nous avons fait le choix de travailler à deux sur ce projet. Après hésitation avec le Python, le langage de programmation utilisé est le Java, langage sur lequel nous étions toutes deux le plus à l’aise.

*Nous allons donc vous décrire notre projet, tout d’abord en expliquant les règles de notre simulation, ensuite en montrant l’architecture de notre projet, et enfin nous parlerons de certaines fonctions ou code intéressants.*

# Description du projet

## Brève description

Ce projet consiste en une simulation discrète à pas de temps constant, adaptée de l’univers de *Game Of Thrones* couplé à un système tiré des jeux de rôle où la réussite d’une action dépend de lancers de dés. Comme dans la série *Game Of Thrones*, nous retrouvons ainsi les célèbres familles *Stark, Lannister, Targaryen* et les *Sauvageons*, sans oublier les redoutables *Marcheurs Blancs* – éléments perturbateurs pouvant compromettre la survie des factions.

Doté de points d’attaque, de vie et d’expérience, chaque individu évolue sur une carte composée d’obstacles et peut être amené à aider voire à combattre jusque mort s’ensuive les autres personnages. Pour se protéger, chaque population hormis les Marcheurs Blancs possède une safezone qui lui est propre.

En théorie, la simulation se termine lorsqu’il ne reste plus qu’une seule population sur la carte. Néanmoins, la simulation pouvant s’éterniser, un mode de calcul basé sur les caractéristiques finales des individus encore en vie nous permet de déterminer la population gagnante.

## Les règles de la simulation

### Les individus

Ainsi, les personnages de notre simulation comportent cinq populations différentes que l’on peut diviser en deux sous-catégories : les humains et les Marcheurs Blancs. Elles peuvent toutes se déplacer sur la carte, rencontrer et attaquer d’autres personnages. Notons qu’elles ont en commun des points de vie et des points d’attaque.

La différence entre humain et Marcheur Blanc se fait au niveau des caractéristiques et de la façon d’attaquer. Contrairement aux humains, les Marcheurs Blancs ne possèdent par exemple pas de safezone (zone de récupération), ni de points d’expériences car on considère qu’ils sont déjà assez puissants dès le début.

#### Les Humains

Les Lannister, les Stark, les Sauvageons et les Targaryen représentent les quatre populations humaines évoluant sur la carte. Ces factions sont divisées en deux catégories :

* Les familles du Nord (=Northener) dans lesquels on trouve les Stark et les Sauvageons.
* Les familles du Sud (=Southerner) regroupant les Lannister et les « Targaryen.

Tous ces individus sont des humains qui possèdent chacun un nom, un niveau, des points d’expérience, des points d’endurance et une safezone. Notons que les valeurs attribuées à ces caractéristiques peuvent être différentes en fonction de la faction considérée.

#### Les Marcheurs Blancs

Nous retrouvons, pour finir, une autre population : les Marcheurs Blancs. Tous les x tours, un Marcheur Blanc prendra place sur la carte.

Chaque Marcheur Blanc possède un micro climat pouvant influencer les caractéristiques des autres factions.

#### Les déplacements des individus

A chaque étape, et pour chaque individu, le mode de déplacement est tiré aléatoirement […] – ce qui se passe

#### Les rencontres entre individus

Au cours de la simulation, les différents personnages peuvent être amenés à se rencontrer. Avec nos cinq types de population, il existe plusieurs cas :

* Rencontre de même population : ce qui se passe
* Entre humains :
* Entre Marcheurs Blancs :
* Rencontre de populations différentes :
* Entre deux familles du nord : ce qui se passe
* Entre deux familles du sud : ce qui se passe
* Rencontre avec un Marcheur Blanc : ce qui se passe

#### Les attaques

Lors de « mauvaises » rencontres, des attaques peuvent se produire. Elles correspondent à des combats au « tout par tour » durant lesquelles l’individu attaque une seule fois par tour. A chaque tour, le nombre de points de vie changent en fonction de l’attaque reçue. Plus l’attaquant aura d’XP, plus il retranchera des points de vie à son adversaire.

L’attaque reçue est déterminée à l’aide d’un système de lancer de dés aboutissant soit à un échec, à un succès ou à un succès critique. Notons que lors d’un succès critique, les populations humaines lancent une super attaque pouvant permettre de retourner la situation. Chaque faction possèdent sa propre super attaque :

* Targaryen : super attaque qui permet de tuer instantanément leur adversaire à l’aide de leurs dragons.
* Lannister :
* Stark : blabla
* Sauvageons

Le combat se termine lorsqu’un des deux personnages trépasse. Le vainqueur récupère alors l’ensemble des points d’expérience de son adversaire qui disparaît de la simulation.

Notons qu’aucun combat ne peut avoir lieu dans une safezone si celle-ci est propre à l’un des deux personnages.

### La carte

Comme dit précédemment, les personnages évoluent tous sur une carte. La carte est un plateau assez classique : elle consiste en un carré de 20 x 20 caractères qui comporte plusieurs obstacles, ajoutés aléatoirement lors de sa création. A l’initialisation d’une simulation, les différents personnages se positionnent aléatoirement sur la carte et les safezones sont ensuite attribuées.

Les safezones se trouvent aux quatre coins de la carte et sont délimitées par des symboles correspondant à la première lettre de la faction propriétaire. Ainsi, chaque population hormis les Marcheurs Blancs possède sa propre safezone.

La carte se divise également en plusieurs zones climatiques (zone chaude, tempérée, fraiche ou glaciale) pouvant avoir une influence sur les personnages.

### La simulation

Nous avons mis en place une simulation discrète à pas de temps constant. A chaque étape, chaque individu encore vivant se déplace pour évoluer sur la carte. A chaque début de tour, l’ordre de prise en charge des personnages changent. De plus, tous les x tours, un Marcheur Blanc arrivera sur la Carte afin de redynamiser la simulation si besoin.

La simulation se termine lorsqu’il ne reste que des Marcheurs Blancs, que des familles du Nord ou que des familles du Sud.

### Fonctionnalités supplémentaires

* Arrivée périodique de marcheurs blancs, éléments perturbateurs de la simulation pouvant mettre à mal l’avancée des humains
* Attribution d’un nom aux personnages pour faciliter le suivi de la simulation
* Pattern Singleton pour Westeros et Game Board

# Conception

## architecture du projet

Utilisation de packages (diagramme de packages ?)

Classes utilitaires = FileManager et UserInterface

Gestionnaire du jeu = GameMaster

## Intégration des concepts objets fondamentaux

* Héritage (partis individus)
* Encapsulation (portée des attributs et des méthodes, getters, setters)
* Polymorphisme (move, attaques, meet)

Types de polymorphisme :

* Meet = paramètres méthode qui change
* Super attack = méthodes appelées sont différentes en fonction de la classe

## Diagramme de classes

## Visualisation de la simulation

Affichage console + système de logs

## Implémentation des fonctionnalités supplémentaires

Mise en place du design pattern singleton (pour gameboard et gamemaster)

# Déroulement d’une simulation

## Déroulé d’un tour pour un personnage

diagramme d’états-transitions (e.g., à propos des états pris par les individus en

fonction de ce qui leur arrive)

## Fonctionnement de la boucle principale

diagramme d’activité qui représentera les instructions importantes de l’algorithme “global” de la simulation (ceci comprend la boucle principale manipulant le pas de temps et la boucle qui gère le déplacement des individus).

# Analyse personnelle

## Répartition des rôles

Matrice de responsabilités

## Choix faits en termes d’organisation

*Avant de programmer, j’ai commencé à réfléchir à un sujet original, cohérent et dans lequel je pouvais mettre à profit toutes les notions que j’ai pu acquérir lors du cours de Java. Une fois le concept trouvé, je me suis intéressée aux classes et aux liens entre les différentes classes. Suite à cela, j’ai établi le diagramme de classe global du projet dans lequel j’ai y placé les différentes méthodes et attributs des classes. Bien que ce travail de préparation m’ait pris environ deux semaines, j’ai pu avoir une vue d’ensemble sur le projet.*

*Une fois mon projet validé, j’ai ensuite réfléchi en détail aux fonctionnements de chacune des méthodes afin de pouvoir les programmer rapidement et efficacement. Cela m’a pris environ une semaine, toutefois, il m’a permis d’économiser du temps pour la suite.*

*Concernant la programmation, j’ai souhaité commencer par coder puis débugger les fonctions terminales avant de remonter aux fonctions plus générales de façon à gagner du temps.*

## Bilan

### Difficultés rencontrées

### Points de satisfaction

# Conclusion

D’une façon générale, ce projet présente l’ensemble des notions présentées en cours de Concept Object : héritage, encapsulation, polymorphisme, diagrammes UML, simulation, etc. Ainsi, ce projet nous a apporté une expérience supplémentaire dans le domaine du développement.

En guise d’améliorations du projet, il serait intéressant de :