Etude de conception

CURT Elien - DAUBREGE Simon - LACOMBE Dorian

I. Principe du projet

L'ensemble du projet est codé dans le langage **Java.**

La première partie est une application de simulation d'un "hôpital fantastique" nommé Fantasy Hospital. Dans cette simulation, l'utilisateur prend le rôle d'un médecin qui a pour but de gérer cet hôpital. La simulation fonctionne sous forme de tour où toutes les 30 secondes un certain nombre d'actions s'effectue par elles-même et le joueur a la possibilité d'effectuer 2 actions toutes les **30 secondes**.

La deuxième partie est aussi une simulation mais cette fois, d'une colonie de lycanthropes. Dans cette dernière, l'utilisateur a aussi des actions mais limitées, c'est-à-dire que la meute se gère "toute seule" et les actions qu'elle effectue sont aléatoires.

II. Fonctionnement

<u>Partie hôpital fantastique</u>
Pour cette simulation nous avons tout d'abord le package "creatures".
La classe la plus importante "Creature" qui est la classe abstraite mère de chacun des types de Créature (présent dans le package "specific"). Cette classe comporte l'ensemble des fonctions que les créatures ont en commun. Pour les fonctions propres à certains types de créature, elles sont dans leur classe respective.
Il y a aussi la classe Maladie, elle représente une condition affectant une créature avec un type défini et un niveau de gravité.
Son objectif est de pouvoir gérer l'état de santé d'une créature en modifiant progressivement l'évolution de la maladie ou de vérifier si une maladie atteint un niveau critique ou létal.
Nous avons ensuite les interfaces :

- ClientVIPPrioritaire : destinée à des classes, comme les créatures, qui peuvent bénéficier du statut VIP prioritaire ce qui influence leur réaction au temps d'attente.

- CreatureBestiale : cette interface introduit une mécanique où la mort de la créature peut avoir des répercussions biologiques sur son environnement.
- HabitantDuTriage : adaptée pour les créatures impliquées dans un processus de triage, où l'attente peut avoir des effets variables selon le contexte.
- MortVivant : appliqué aux créatures ayant des propriétés spéciales qui leur permettent de survivre à la mort.

Passons au package main :

- HopitalFantastique : La classe HopitalFantastique gère les services, le personnel et les créatures présentes. Elle contient toutes les fonctions relatives à la gestion, par exemple : ajouterService(ServiceMedical service), setServices(), getMedecins() et setMedecins(), getNombreCreatures(), afficherServices() et afficherCreatures().
- InterfaceHopital : gère l'interaction avec l'utilisateur dans le cadre de la simulation. Elle permet de réaliser différentes actions liées à la gestion des services et des créatures de l'hôpital à travers un menu interactif comme : afficherMenu() ou resetActions().
- Simulation : gère l'ensemble de la simulation de l'Hôpital Fantastique. Elle crée et gère l'hôpital, y ajoute des créatures, des services médicaux, et simule des événements aléatoires affectant les créatures. Les Thread permettent un fonctionnement par tours avec des actions aléatoires sur les créatures et un menu interactif pour l'utilisateur.

Pour le package services :

- ServiceMedical : représente le service médical de l'hôpital au sein de la simulation. Cette classe gère plusieurs aspects liés au service, notamment sa capacité, la liste des créatures qu'il accueille, ainsi que leur traitement.
- Crypte : extends de la classe ServiceMedical, elle représente un service médical spécifique destiné aux créatures de type MortVivant, elle hérite donc de tous les attributs et possède niveauVentilation et temperature spécifique à ce service.
- CentreDeQuarantaine : La classe CentreDeQuarantaine représente un service médical spécialisé dans la gestion des créatures contaminées ou malades

Enfin, le package values :

- TypeBudget : La classe TypeBudget est une énumération qui définit différents types de budget possibles pour les services médicaux. Chaque élément de l'énumération représente un type de budget spécifique avec un label associé.
- TypeSexe : TypeSexe est une enum aussi, elle définit les différents sexe possible chez les créatures. Chaque élément de l'énumération représente un type de sexe spécifique avec un label associé.
- TypeMaladie : TypeMaladie est encore une enum, elle définit les différentes maladies diagnostiquées . Chaque élément de l'énumération représente une maladie spécifique avec un label associé.

Partie meute de lycanthrope :

Le package lycanthrope :

- Classe Colonie : modélise la colonie composée de meutes de lycanthropes. Chaque meute est un groupe structuré de lycanthropes ayant des caractéristiques spécifiques et interagissant selon des règles de hiérarchie et d'événements aléatoires. La classe inclut des fonctionnalités pour gérer les meutes, les lycanthropes, et une simulation d'événements aléatoires ou initiés par l'utilisateur.
- Classe Couple : représente un couple de lycanthropes composé d'un mâle et d'une femelle. Elle sert principalement à modéliser les interactions reproductives entre les lycanthropes dans une meute. Elle comporte des mécanismes de vérification du sexe des individus, d'affichage des caractéristiques des membres du couple, et de gestion de la reproduction.
- Classe Lycanthrope2 : représente un lycanthrope dans un système orienté objet. Cette classe, Lycanthrope2, inclut plusieurs fonctionnalités riches comme la gestion des attributs biologiques, sociaux (rangs et domination), et l'intégration dans un groupe (meute). La classe comprend, dans la procédure "hurler", un itérateur qui parcourt les lycanthropes de la meute pour les faire entendre un hurlement d'appartenance.
- Classe Meute : représente une liste de lycanthropes, elle gère la taille de la meute, les tris par ordre de rang (du plus haut au plus faible) et gère les couples.

Le package values :

- Enum TypeAge : énumération qui représente les différentes catégories d'âge pour un lycanthrope (Jeune, Adulte, Vieux)
- Enum TypeHurlement : énumère les différents type de hurlement des lycanthropes (hurlement : d'appartenance, de domination, de soumission, d'agressivité)
- Enum TypeRangDomination : énumère l'ensemble des rangs de domination des lycanthropes, plus le rang d'une des créatures est élevé plus il pourra en soumettre d'autres.

III. Conclusion

Cette étude de conception propose une architecture claire et bien structurée pour la simulation d'un hôpital fantastique et d'une colonie de lycanthropes. En s'appuyant sur une programmation orientée objet et en exploitant des concepts comme l'héritage, les interfaces et les énumérations, le projet donne une simulation dynamique et immersive. Chaque composant est conçu pour être modulaire, facilitant la gestion des interactions complexes entre entités et permettant des évolutions futures.