Thomas Sulpice

Lounis Krimed

**Rapport Projet Java**

1. Présentation du code

Le projet contient 4 packages : game, main, representation et univers.

| Package | Classes , Interfaces |
| --- | --- |
| game | Game |
| main | Main |
| representation | Event, Node, NodeDecorator, DecisionNode, CombatNode, InnerNode, ChanceNode, TerminalNode |
| univers | Entity, Character, Monster, Mage, Archer, Swordsman, Skill, SkillAOE, SkillSingle, MonsterDB, NodeDB, SkillDB, SkillTest, MonsterTest |

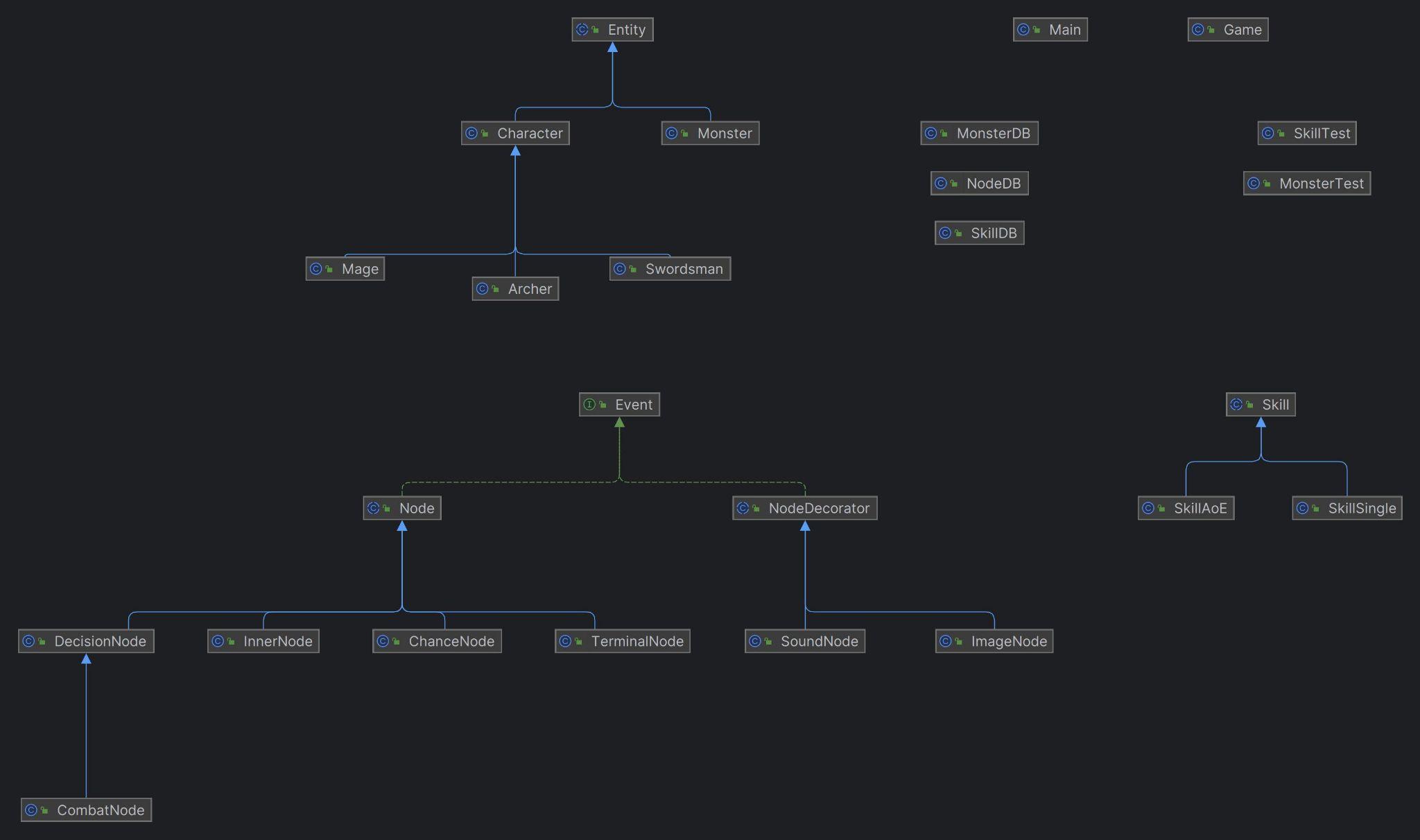
****

Diagramme UML de l’ensemble des classes du Projet généré par le plugin Diagrams d’IntelliJ

Les classes DecisionNode, InnerNode, ChanceNode et TerminalNode héritent naturellement de Node car elles doivent toutes implémenter certains attributs et méthodes, tels que la description et la méthode display.

La classe CombatNode hérite elle de DecisionNode. En effet, lors d’un combat, le joueur doit sélectionner une attaque lors de son tour. Il semble alors assez naturel de créer une nouvelle classe héritant de DecisionNode afin de simplifier le déroulement des combats.

Afin de permettre à chaque type de Node de pouvoir aussi afficher une image et jouer un son, on implémente un pattern design Decorator. Ainsi, les classes Node et NodeDecorator implémentent l’interface Event, et les classes ImageNode et SoundNode héritent donc de NodeDecorator.

Les classes Character et Monster devant toutes deux comporter certains attributs et méthodes en commun, tels que leur nom, leur niveau, leurs attaques, etc, elles héritent d’une même classe Entity.

La classe Character comporte elle même trois sous-classes, Archer, Mage et Swordsman.

Enfin, la classe Skill comporte deux sous-classes SkillAOE et SkillSingle, afin de différencier les attaques à cibles multiples des attaques à cible unique. En effet, une attaque à cibles multiples pourrait par exemple contenir un attribut en plus qui définirait le nombre maximal de cibles touchées.

1. Extensions éventuelles

Parlons des modifications que nous pourrions ajouter au code avec plus de temps.

Évidemment, nous pourrions compléter l’histoire et ajouter une véritable interface au jeu.

Nous pourrions ajouter de nouvelles classes de personnage, implémenter le système de précision dans les combats, permettre au joueur de combattre plusieurs monstres en même temps et donc modifier la classe SkillAOE.

1. Difficultés rencontrées

Le projet n’était en lui-même pas très compliqué en termes de niveau de programmation, mais avoir une vue d’ensemble des classes à réaliser était difficile au début du projet, d’autant plus que nous avions pas encore tous les outils nécessaires pour réaliser un projet cohérent.

Par exemple, l’utilité de la classe InnerNode n’était pas évidente au moment de la remise du premier jalon, et nous avons donc écrit les autres classes indépendamment de celle-ci.

Les vraies difficultés résidaient plutôt dans les concepts et les bibliothèques non mentionnées en cours, en particulier le pattern design Decorator et l’affichage d’images ou de sons.