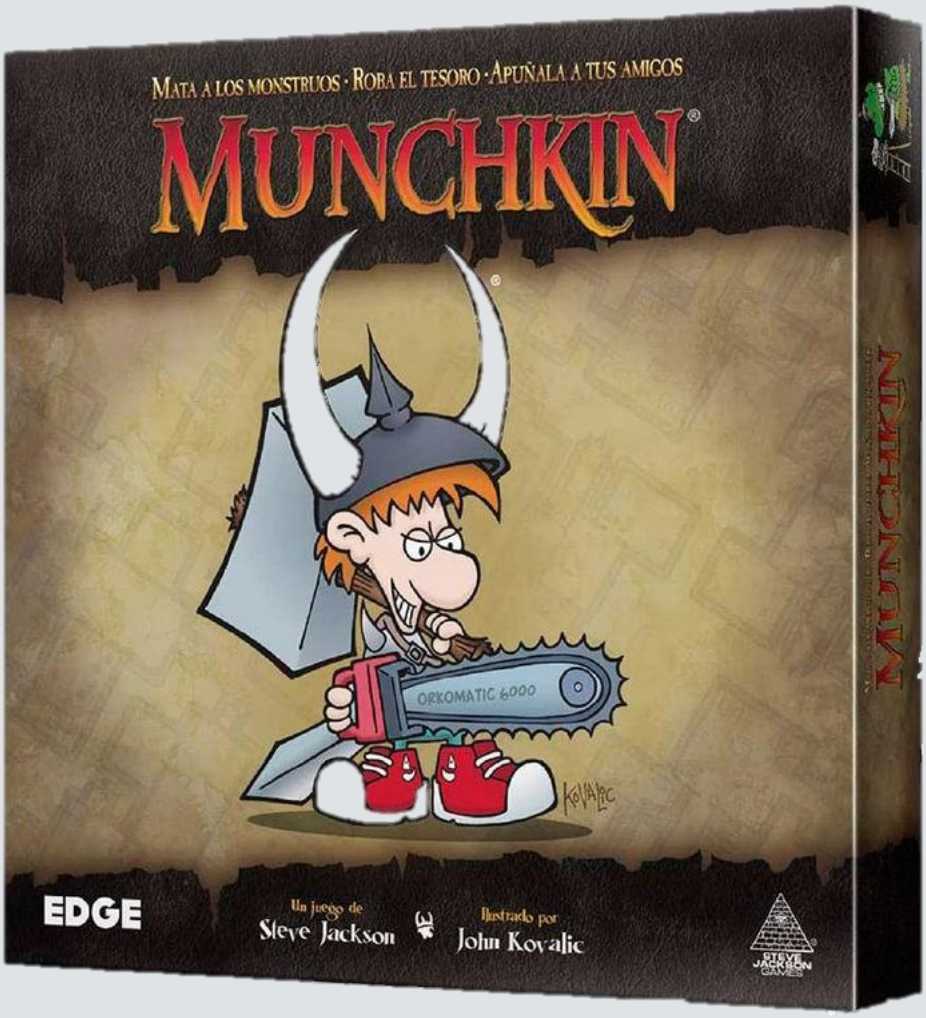
Rapport de Conception AP4B

MUNCHKIN EN JAVA



BONNET Rémi, MORALES Léon, MORIN Alban

UTBM – A2023 - version 2

SOMMAIRE

PRESENTATION

Dans le cadre de l'UE Programmation Orientée Objet AP4B, il nous est demandé de réaliser un projet en groupe afin de mettre en pratique les compétences apprises au long du semestre. Il s'agit d'un jeu de société à recréer à l'aide du langage JAVA. Dans un premier temps nous devons conceptualiser le jeu à l'aide de diagrammes UML en respectant les bonnes pratiques vues en cours. La partie programmation ne commence qu'une fois cette conception terminée.

Parmi les sujets possibles, nous avons choisi celui du Munchkin car nous étions déjà familiers avec la version physique. Faire quelques parties réelles nous a permis de facilement lister les différentes fonctionnalités à implémenter. Nous avons fait le choix de garder le maximum de règles d'origine possible. La notice du jeu est consultable ici.

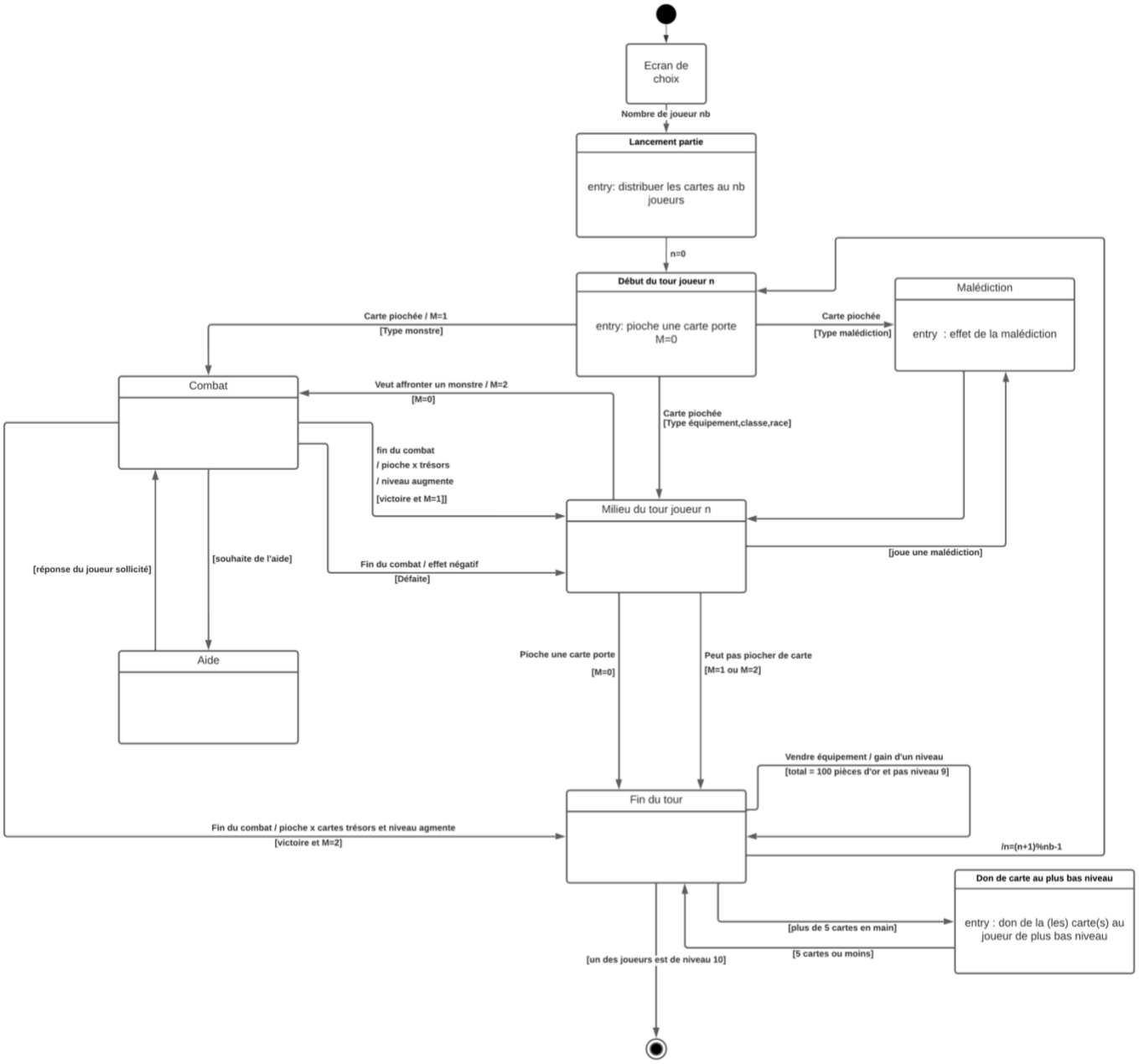
Avec le souhait d'adapter le jeu à l'UTBM (les personnages, objets, …), nous avons créé des cartes inédites pour notre version de Munchkin. Les spécificités de chaque carte restent similaires au jeu original.

Au début de chaque nouvelle partie, l'utilisateur choisit le nombre de joueurs (minimum 2). La partie prend fin dès lors qu'un joueur atteint le niveau 10 et celui-ci est déclaré vainqueur.

NB : L'unique changement entre la version 1 et 2 de ce rapport est le diagramme de classe.

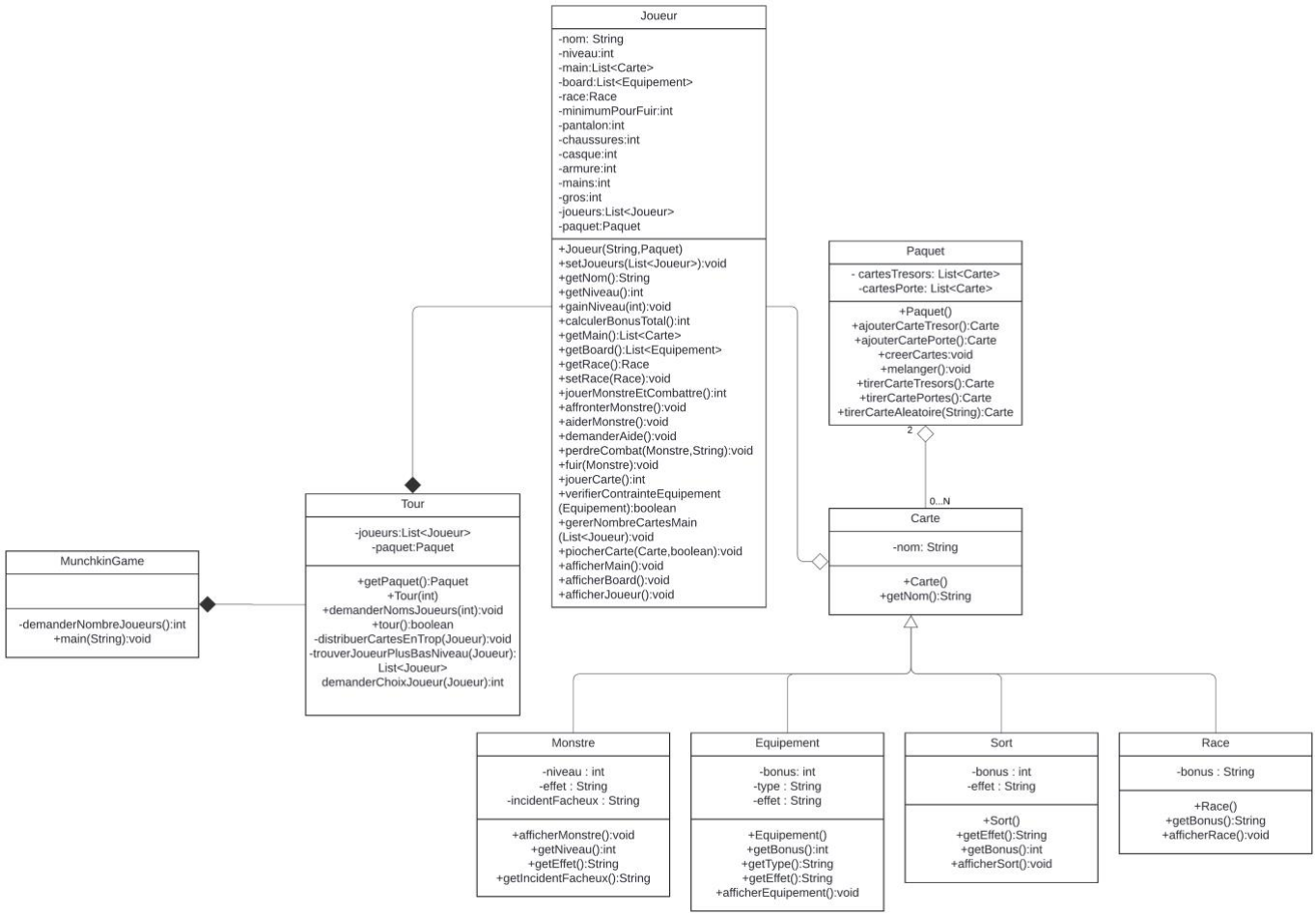
CONCEPTION UML

ETATS TRANSITION



La variable M sert à savoir si le joueur a déjà combattu un monstre ou non lors de son tour.

DIAGRAMME DE CLASSES



Relations entre les classes :

Joueur --> Carte

Cette agrégation suggère que les cartes peuvent exister indépendamment des joueurs, et un joueur peut posséder une collection de cartes. Si un joueur est supprimé, ses cartes associées ne sont pas nécessairement détruites. Cela offre plus de flexibilité par rapport à la composition et reflète une relation plus faible entre les deux classes.

Joueur -->Tour de jeu

La composition signifie que les objets **`Joueur`** sont des composants essentiels du tour de jeu. Si un tour de jeu est créé, il est associé à une collection de joueurs. Si le tour de jeu est détruit, tous les joueurs associés à ce tour sont également détruits. Cette relation de composition reflète une forte dépendance entre les objets TourDeJeu et Joueur, où les joueurs sont considérés comme faisant partie intégrante du tour de jeu.

Le reste des agrégations reprend suit la même logique que la relation Joueur --> Carte

Les types de Carte

Les monstres, équipements, sorts et races pouvant être pioché par le joueur héritent de façon évidente de la classe carte.

USE CASE

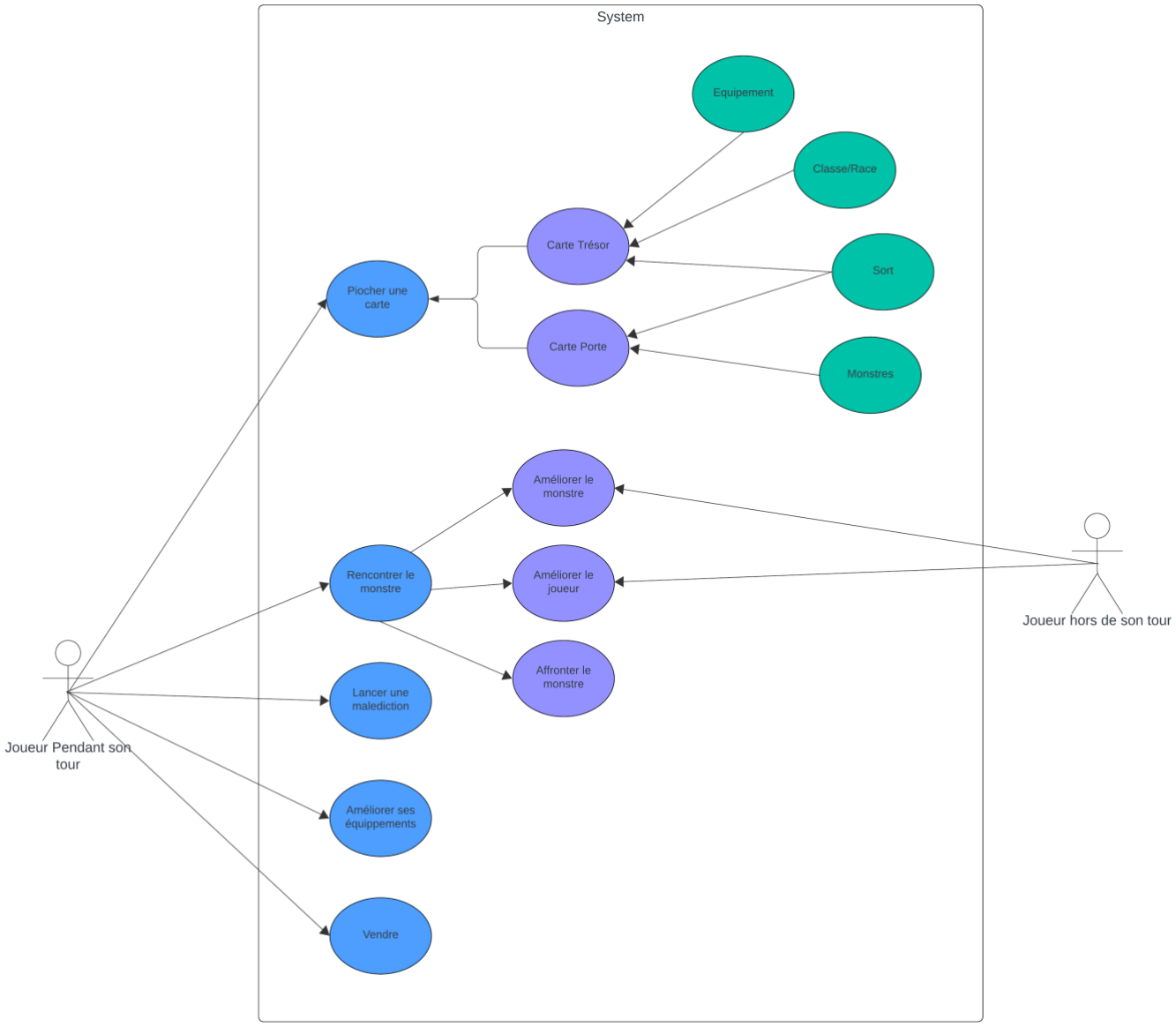

System
Equipement
Classe/Race
Carte Trésor
Sort
Piocher une carte
Carte Porte
Monstres
Améliorer le monstre
Rencontrer le monstre
Améliorer le joueur
Affronter le monstre
Lancer une malediction
Améliorer ses équippements
Vendre
Joueur Pendant son tour
Joueur hors de son tour

DIAGRAMME DE SEQUENCE

Nous avons fait le choix de ne pas réaliser de diagramme de séquence pour notre projet pour les raisons suivantes :

- ➔ Il ressemblerait beaucoup au diagramme d'états transitions mais en moins lisible et en apportant peu d'informations pertinentes.
- ➔ Ce n'est pas forcément adapté à notre cas de figure, un jeu en local sur une seule machine avec des tours qui se répètent.
- ➔ Le nombre d'acteurs et leurs interactions mutuelles est très limité. Celles-ci sont déjà décrite par le diagramme d'ET pour la plupart.

REFERENCES

Image de couverture : https://fr.nauticamilanonline.com/jeu-de-societe-munchkin-b44cc/