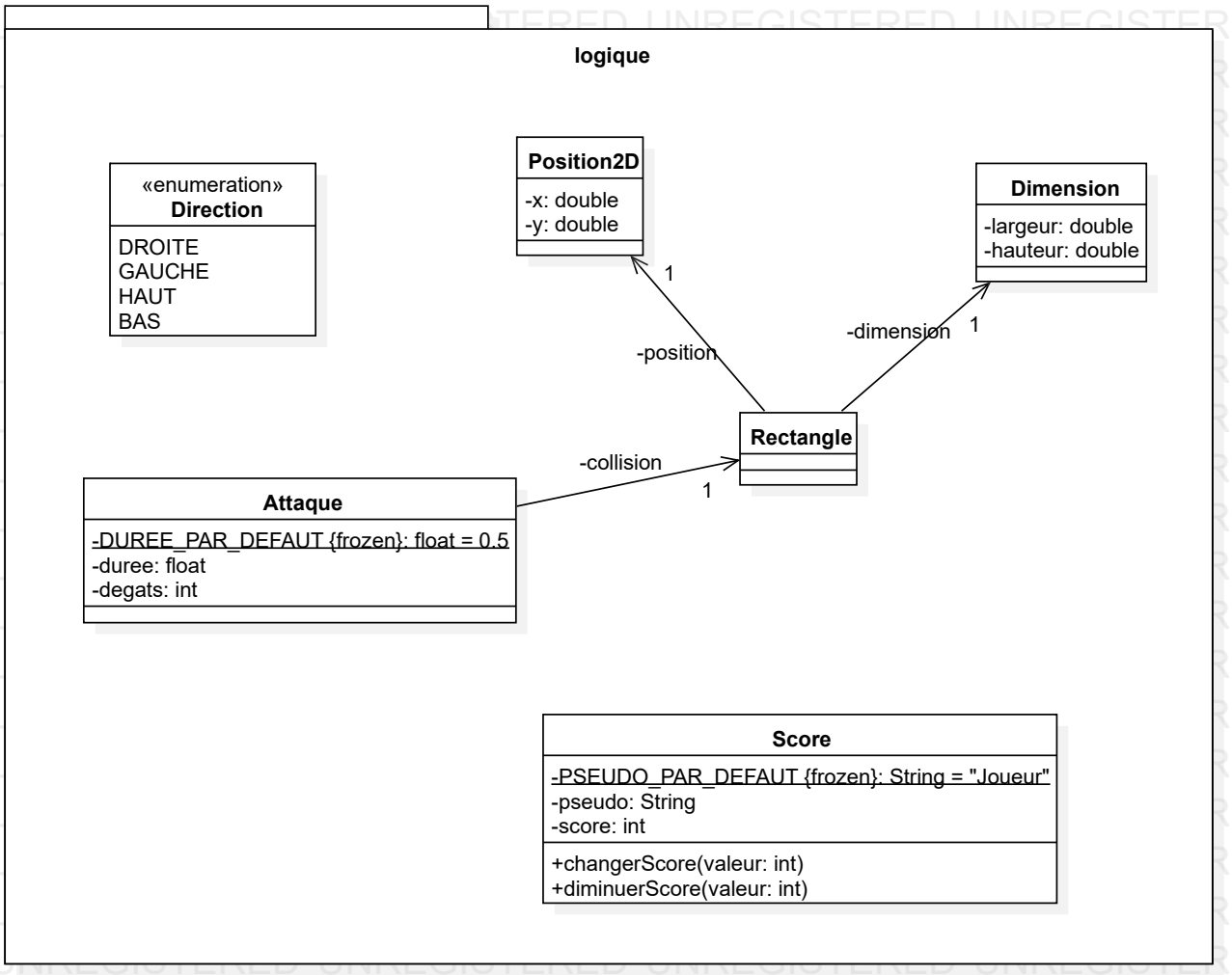
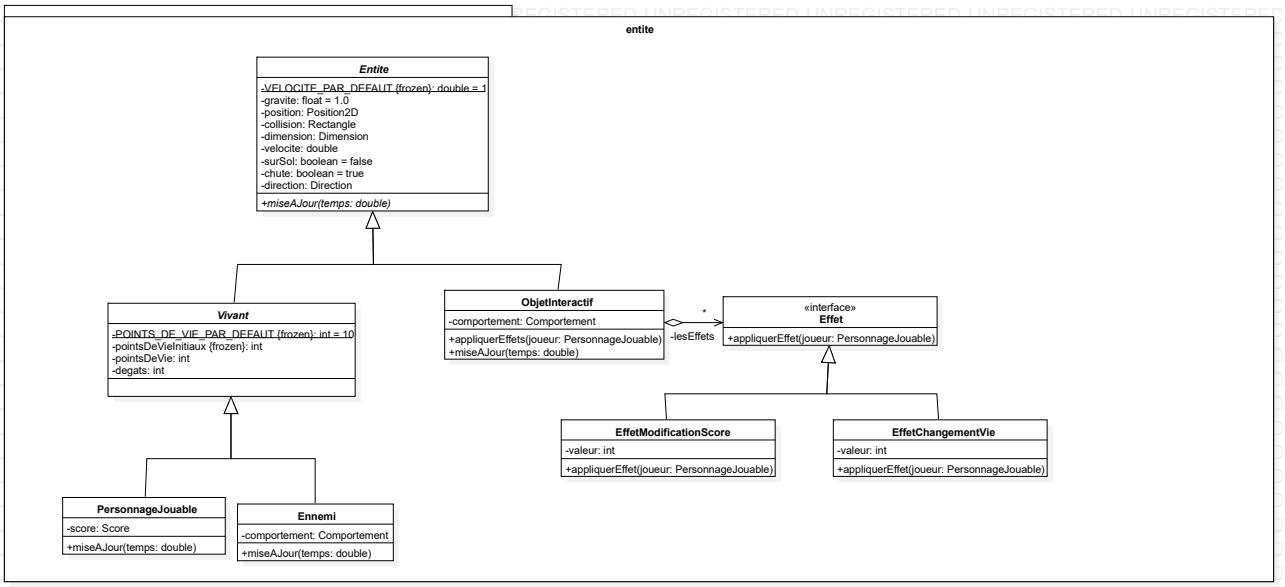
Jérémy TREMBLAY B1
Guillaume TOUMAZET A ?

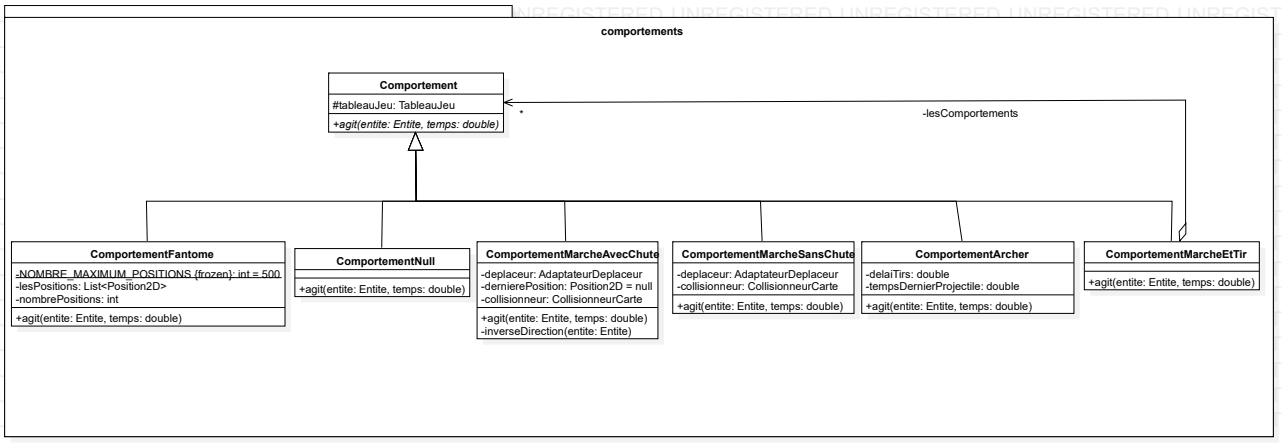
Description du diagramme de classes



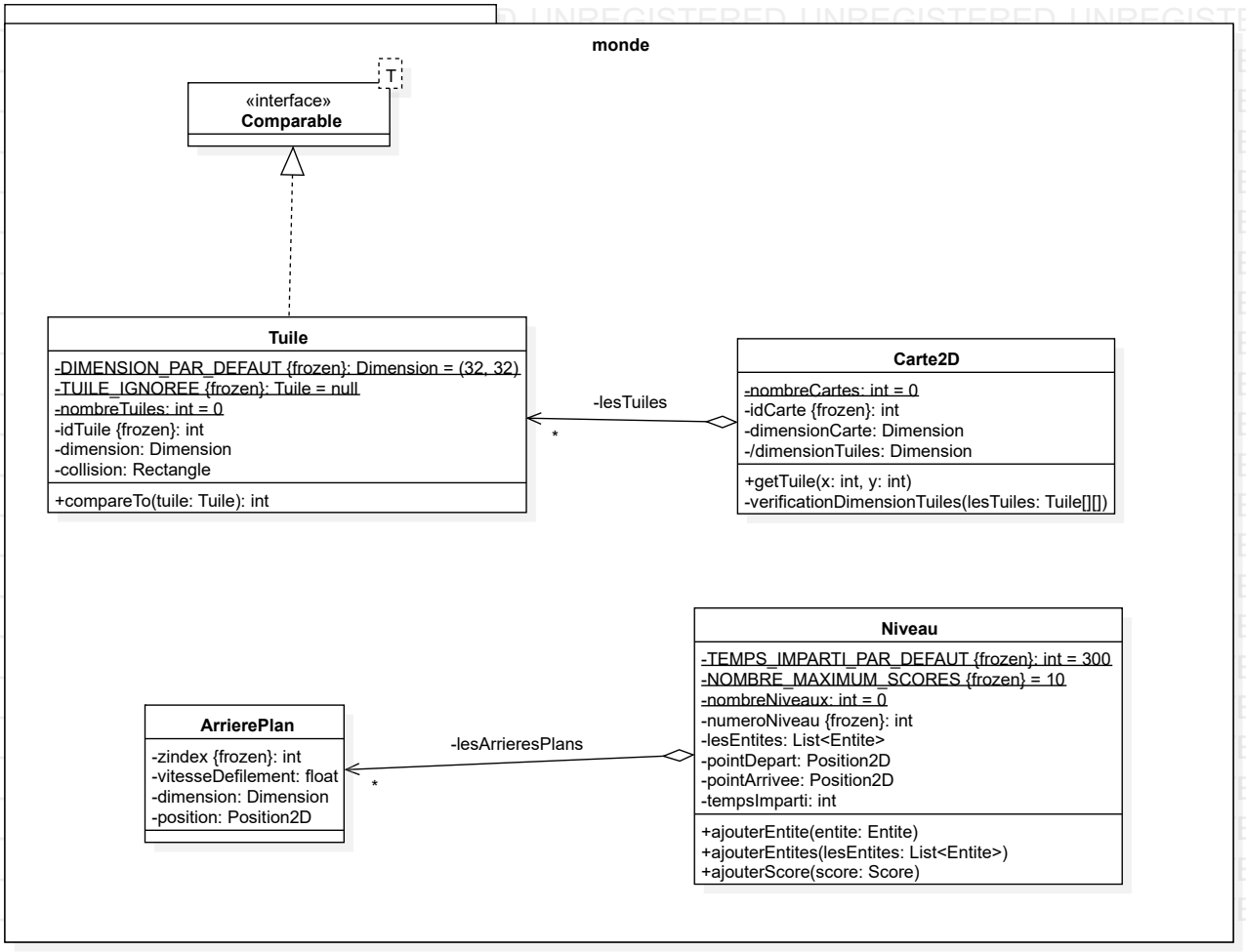
Nous avons des classes basiques qui permettent à notre jeu de fonctionner correctement. La position contient simplement des double correspondant aux coordonnées en x et y d'un objet. Un rectangle possède une dimension et une position. Une score n'est qu'un entier couplé d'un pseudo (pour savoir qui a réalisé ce score).



Notre jeu est composé de la façon suivante : chaque élément qui peut bouger est une entité. Une entité a alors un rectangle qui détermine sa zone de collision, une position, une dimension, et des *booleens* indiquant si elles sont sur le sol ou qu'elles tombent. Un vivant possède des points de vie, et un nombre de dégâts, et un ennemi a un comportement (c'est une stratégie permettant de changement dynamiquement l'attitude que vont avoir les ennemis dans notre jeu). Un personnage a un score représentant sa performance.

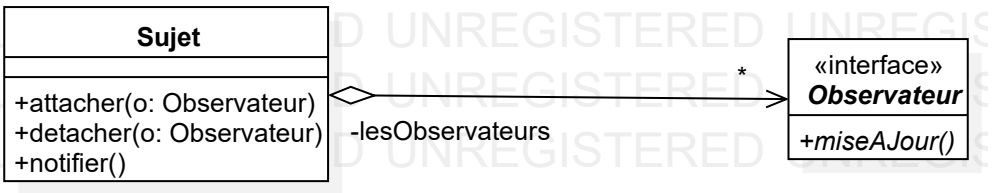


La stratégie des comportements est illustrée plus en détails ici.

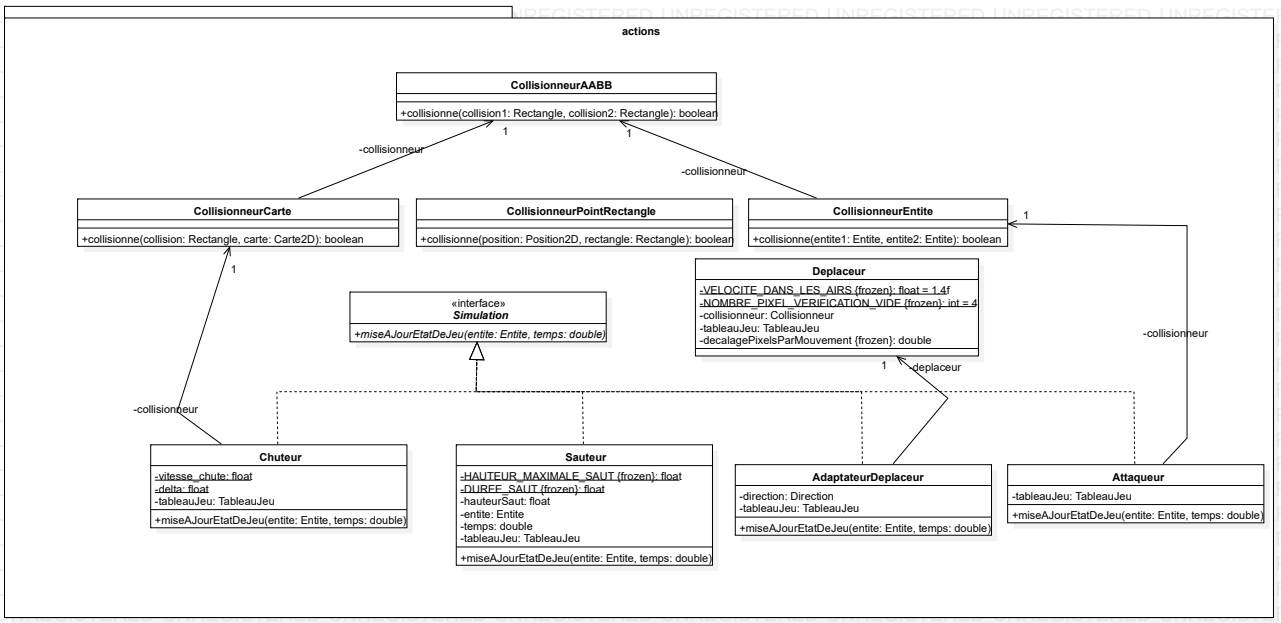


Nos mondes sont gérés de la façons suivante : c'est des tuiles disposées sur une carte à deux dimensions. Ainsi la classe tuile a un identifiant unique fixée lors de sa création et une zone de collision, ainsi qu'une dimension. Une carte comporte un tableau à deux dimensions de tuiles qui représente la carte. On connaît sa dimension. Un niveau possède une carte, ainsi qu'une collection d'entités et d'arrières plans.

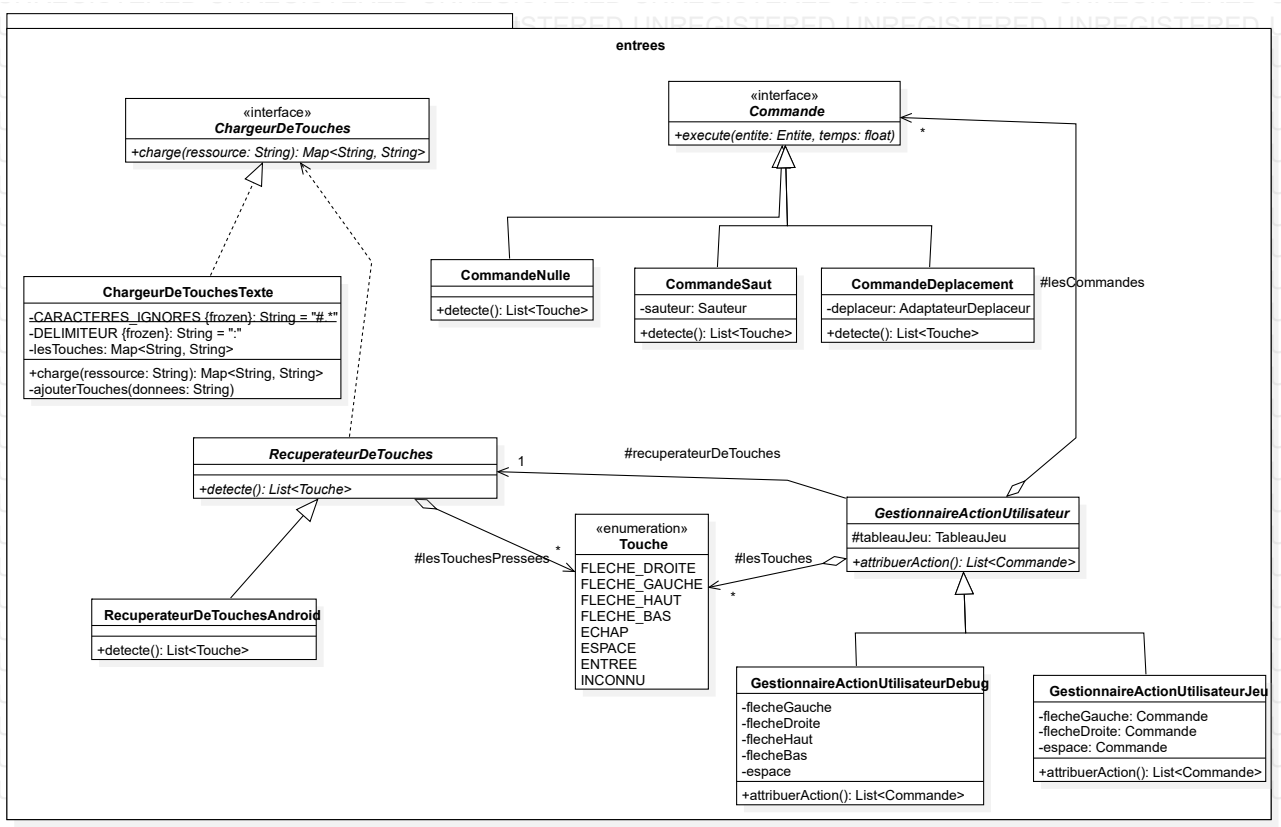
Un arrière-plan n'a qu'une position, une vitesse de défilement et un niveau d'importance qui va jouer lors de l'affichage, ainsi qu'une dimension. On a alors des méthodes pour ajouter ou supprimer des entités pour lorsqu'elles apparaissent ou qu'elles meurent. On connaît le point de départ du niveau (ou va apparaître le personnage), et le point où il va finir. Il possède un numéro unique ainsi qu'un temps maximal pour être fini.



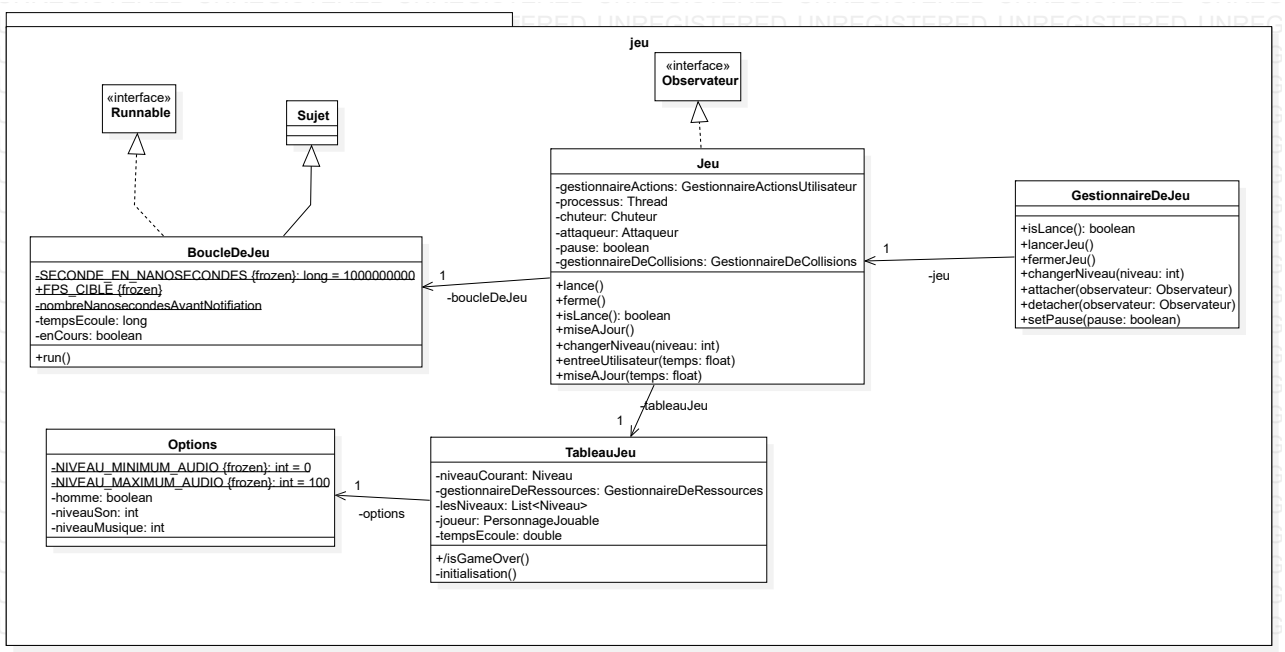
Cet observateur est celui utilisé par notre boucle de jeu pour se mettre à jour. La vue est également notifiée de la sorte. Elle s'abonne au jeu et est ensuite notifiée à intervalles réguliers.



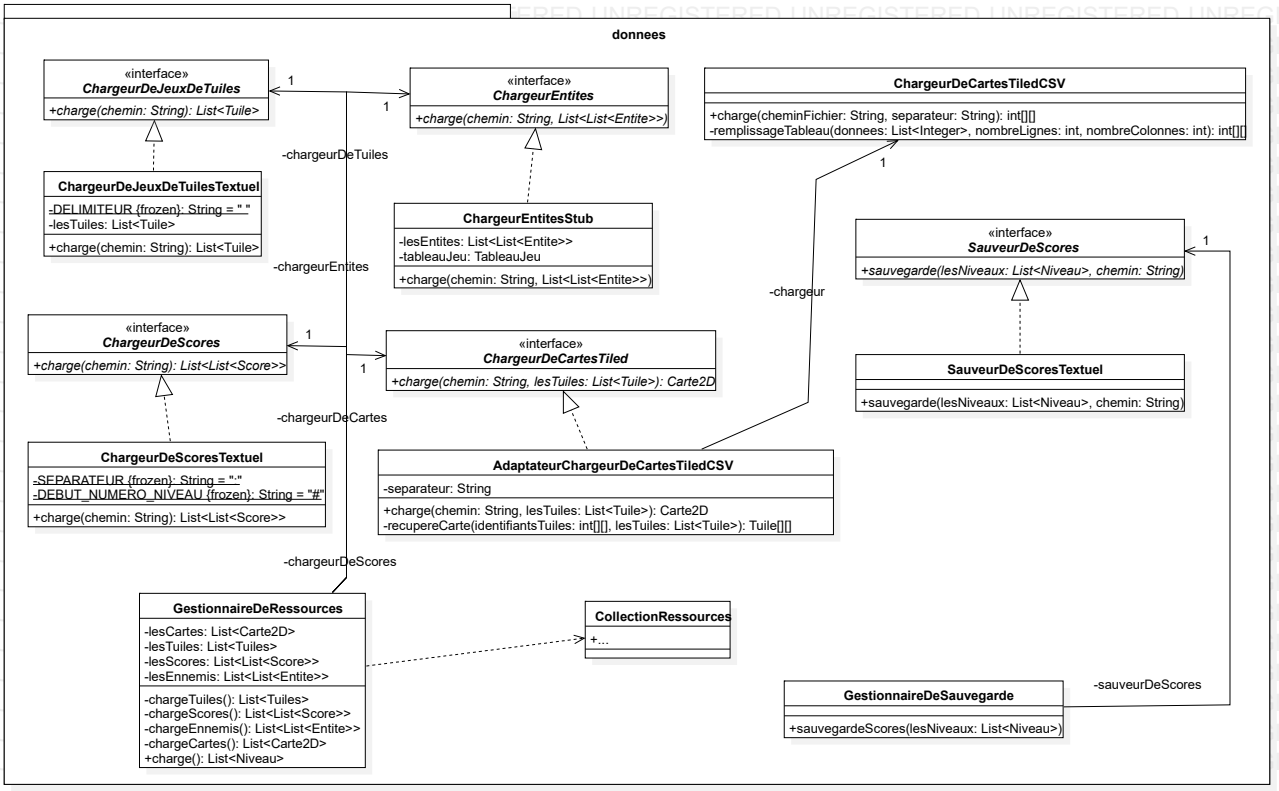
Ces actions permettent à notre jeu de fonctionner parfaitement. Une simulation vient mettre à jour le jeu, et prend un temps en paramètre (pour faire des calculs) et une entité (une simulation agit sur cette entité). Nous avons ainsi un chuteur qui va venir faire tomber une entité en continu. Un sauteur permet à une entité de s'élever dans les airs en suivant la fonction mathématique du cosinus. Un déplaceur a pour objectif de simplement déplacer une entité suivant une direction (d'où l'adaptateur pour avoir la direction). Enfin l'attaqueur n'est pas encore implémenté. Toutes ces actions sont régies par des collisions. On a ainsi un colisionneurAABB qui vient simplement détecter s'il y a collision entre deux rectangles quelconque. Un collisionneur carte précise s'il y a une collision avec la carte par rapport à un rectangle donné.



Pour détecter les entrées du joueur et réagir en conséquence, nous nous sommes servis du patron de conception Commande qui vient encapsuler un appel de méthode permettant de le paramétriser. Une commande encapsule simplement l'appel à une méthode (exemple : appel du Sauteur ou de l'attaqueur). Pour détecter les touches, on utilise un recupérateur de touches android qui place les touches pressées dans une liste. Cette liste est ensuite récupérée de manière continue par un GestionnaireActionUtilisateur dont le but est de simplement retourner une liste de commandes qui vont correspondre aux touches pressées (qui se trouvent dans la liste). Les commandes sont ensuites executées dans le jeu.



Un tableau de jeu (ou game board en anglais) est une classe qui regroupe nos différents niveaux, le personnage, le niveau courant et bien d'autres données. Un jeu possède un tableau de jeu. Un jeu possède une boucle de jeu qui est un sujet. Le jeu est abonné à cette boucle et est notifié à intervalles réguliers. Il se met alors à jour en récupérant les commandes du gestionnaire d'action et en les traitant, puis applique un effet de gravité sur chaque entité. Des options sont présnetes. Enfin le gestionnaire de jeu est une façade qui permet de centraliser les appels de méthodes .



Des chargeurs sont présents et permettent de charger nos différentes données. Le chargeur de cartes charge nos fichiers CSV et créé les cartes associées. Le chargeur de scores ne fait que charger les scores dans le fichier. Les tilesets (ou jeux de tuiles) sont chargés également. Les entités sont simplement chargées depuis un Stub pour le moment, mais nos stratégies de chargement laisse la possibilité de modifier complètement notre modèle et son comportement. La classe CollectionRessources contient toutes nos données, chemins et flux vers les fichiers necessaires. Le sauveur de scores est implémenté mais seulement pour des fichiers textuels.