La classe World est la classe principale où seront chargés ou initialisés tous les éléments de base nécessaires au fonctionnement du jeu, cette classe possède plusieurs méthodes essentielles tels que SpawnEnemy qui gère la création d'un objet de type EnemyDistance ou EnemyHandToHand, sa position, ses stats proviennent d'un fichier xml et elles subiront des modifications au fur et à mesure de la partie pour que la difficulté croît au fil du temps, aussi les ennemis apparaîtront de plus en plus fréquemment au fur et à mesure que le jeu continue. Les entités créées seront stockées dans une ArrayList qui aura tous ses éléments dessinés, le Player est aussi considéré comme une entité et est en première place dans cette ArrayList, lorsqu'un ennemi meurt, il est retiré de cette ArrayList et disparaît donc de l'écran.

Lors de la mort du joueur, l'écran de Game Over apparaît et des statistiques telles que le nombre d'ennemis tués et le temps survécu apparaissent.
Ces mêmes statistiques sont également stockées dans des fichiers XML grâce à la méthode SaveGameData qui, pour chaque partie de l'utilisateur, stocke la date de la partie, les ennemis tués et le temps de jeu. Les méthodes CreateOrUpdatePlayerProfile et CountGamesSaved permettent de stocker le nombre de parties effectués par un joueur afin d'afficher dans une page HTML, via transformation XLST, les informations sur un joueur, la liste des ennemis du jeu ou encore les meilleurs scores effectués.
Cependant le nom du joueur n'est pas rentré manuellement par le joueur, le programme se charge de récupérer le nom de l'utilisateur qui est connecté à l'ordinateur.

La classe Sprite fera le lien entre la position de l'image et la position réelle de l'entité afin d'avoir une même position pour l'entité et son sprite. C'est aussi cette classe qui appellera les méthodes Move du joueur et de l'ennemi.

La classe Button gère un bouton, soit un objet affiché à l'écran qui lorsqu'on passe dessus s'assombrit et lorsqu'on clique dessus provoque une action. Cette action est associée à un bouton grâce à la méthode AddAction qui prend l'action en question en argument. Cela permet de facilement instancier des boutons ayant des actions différentes, plutôt que de créer une sous classe de bouton par action.

La classe abstraite Entity gère tout ce qui a une hitbox et qui est susceptible de bouger. Player, Projectile et Enemy en héritent donc, au contraire Button n'en hérite pas car les boutons sont statiques.

La classe abstraite Enemy hérite de Entity. Elle sert d'intermédiaire aux deux types d'ennemis. Elle possède notamment une méthode pour empêcher les ennemis de se superposer et envoie de l'expérience au joueur lorsqu'un ennemi meurt. Elle gère aussi les dégâts que subissent les ennemis.

La classe EnemyDistance hérite de Enemy, elle traite le cas des ennemis de type « Distance » qui souhaitent rester à une certaine distance du joueur pour lui tirer des projectiles dessus à intervalle régulier.

La classe HandToHandEnemy hérite de Enemy, elle traite le cas des ennemis de type « Corps à corps » qui se rapproche constamment du joueur et lui inflige des dégâts en le touchant.

La classe Player hérite de Entity, et s'occupe de tout ce qui est relatif au joueur, que ce soit ses déplacements, son attaque ou encore les améliorations de caractéristiques qu'il peut choisir après chaque montée de niveau. Elle gère également les dégâts que le joueur subit et arrête la partie lorsque le joueur n'a plus de point de vie. Cette classe possède la classe privée Weapon qui gère les projectiles tirés à la souris par le joueur. L'idée de cette sous classe est de permettre au joueur de changer d'arme et donc de projectiles, qui pourraient avoir des stats différentes. Il y a par exemple la possibilité d'effectuer des tirs à tête chercheuse mais nous n'avons pas utilisé cette fonctionnalité avec les réglages actuels, puisque le joueur ne change jamais d'arme.

La classe Projectiles hérite de Entity, le projectile aura sa propre hitbox et subira des dégâts à chaque ennemi touché, jusqu'à sa destruction. Le projectile s'assurera aussi qu'un ennemi à distance ne peut blesser qu'un joueur et qu'un joueur ne peut blesser qu'un ennemi. Ses points de vies représentent le nombre de cibles qu'il peut toucher avant d'être détruit.

La classe Behavior est une classe énumérée qui possède les différents comportements d'ennemi possibles, ici DISTANCE et HAND_TO_HAND. Elle permet de facilement attribuer des types de comportements aux ennemis et peut aisément être modifiée.

La classe EnemyData va stocker des attributs qui seront lu dans un fichier xml (ici Enemies.xml), c'est cette classe qui sera réutilisée dans World.SpawnEnemy pour créer les ennemis avec les statistiques correspondantes.

La classe EnemyLoader va charger un fichier xml et attribuer les éléments du fichier à un objet de type EnemyData en donnant des valeurs par défaut s'il n'y en a pas dans le xml.

XML :

Les fichiers XML sont tous groupés dans le dossier /XML, ils servent à lire des données pour les injecter dans le jeu et aussi à sauvegarder les données à la fin d'une partie

- Enemies.xml : Contient la liste ennemis avec leurs attributs (Name, Rectangle_X, Rectangle_Y, Size, Type, HP, AttackDamage, Speed, XPValue)

- GameList.xml : Contient la liste de games jouées par l'utilisateur sous la forme Username + Date (jour + heure)

- PlayerProfile.xml : Contient le profil des utilisateurs ayant joué au jeu sous la forme Username + GamesPlayed

- Saves.xml : Contient l'ensemble des parties jouées par les joueurs avec les statistiques collectées en fin de partie. Chaque save contient le nom du joueur, le nombre d'ennemis tués, le temps de jeu et la date

XSD :

Les fichiers XSD ont pour buts de vérifier la cohérence entre le fichier XML et son schéma associé.

- Enemies.xsd : Vérifie Enemies.xml

- PlayerProfile.xsd : Vérifie PlayerProfile.xml

- Saves.xsd : Vérifie Saves.xml

XSLT :

Les fichiers XSLT permettent de générer un fichier, XML ou HTML, en sélectionnant certaines données d'un fichier XML d'entrée.

- Enemies.xsl : Crée une page HTML avec, pour chaque type d'ennemi, son nom, son type d'attaque, ses hp, ses dégâts, sa speed et son XP Value

- GameList.xsl : Crée un fichier XML avec l'ensemble des parties sous la forme Username + Date à partir de Saves.xml

- HighScores.xsl : Crée une page HTML qui récapitule les parties en triant par nombres d'ennemis tués au cours d'une partie pour définir les meilleurs parties

- PlayerProfile.xsl : Crée une page HTMl qui, pour chaque joueur donne son nom et son nombre de parties jouées

Utilisation de DOM/XMLReader et sérialisation pour importer/exporter des données du jeu.

L’utilisation des ces technologies permettent de lire dans le fichier Enemies.xml pour affecter les bons attributs aux ennemis au lancement du jeu puis, à la fin de la partie, permet d’exporter les statistiques de la partie pour les stocker dans Saves.xml et incrémenter le nombre de parties du joueur dans PlayerProfile.xml.