Description du diagramme de classes

- **Main :**
  La classe sert à lancer le programme. Elle contient un ManagerVue ainsi qu'un GestionnaireJoueur.

- **ModeleManager :**
  La classe permet de gérer toutes les autres classes.
  Elle possède une Carte et un DeplaceurJoueur ainsi qu'un numNiveau.
  Elle possède une méthode *getCarte* permettant de récupérer la Carte ainsi qu'une méthode *gestionTouche* qui gère les KeyEvents et les actions à effectuer en conséquence.

- **Observateur :**
  Cette interface définit la méthode *update*.

- **Package Vues :**
  1. **ManagerVue :**
     Le ManagerVue permet de gérer les différentes vues à charger. Il possède une HashMap de String,Parent, la première Stage ainsi que la Stage actuelle (currentView). Il a aussi la SceneShowed, qui est la scène montrée.
     Il possède des méthodes pour régler la taille de la fenêtre ainsi que pour charger, décharger et montrer une vue.
  2. **End :**
     Class regroupant le code behind de la page End.fxml
  3. **GameOver**
     Class regroupant le code behind de la page GameOver.fxml
  4. **Start**
     Class regroupant le code behind de la page Start.fxml
  5. **FenetreDeJeu**
     Class regroupant le code behind de la page FenetreDeJeu.fxml
     Elle possède un boucleur de jeu, un Timer et une liste d'IAGarde
     La classe contient les méthodes pour afficher la carte ainsi que pour ajouter une IA à la liste d'IA

- ## Package Joueur :

1. **GestionnaireJoueur :**
   La classe possède un joueurActuel, représentant le joueur en train de jouer, ainsi qu'une ListeJoueurs. Il est là pour gérer les actions en rapport avec les joueurs.
6. **Joueur :**
   La classe représente un joueur. Elle continent un nom ainsi qu'un Score. De plus, elle possède une StringProperty, permettant d'afficher correctement le score sur la vue.
   Elle possède une méthode *refreshInformations* permettant de recharger les infosJoueurs pour la vue.
2. **JoueurSerializable :**
   Cette classe à juste pour but d'être sauvegardée dans un fichier. Comme le Joueur et le Score possède une StringProperty, il est impossible de les sérialiser. Cette classe joue juste le rôle d'intermédiaire.
3. **ListeJoueurs :**
   Cette classe contient deux liste de joueurs. Une observable et une normale. La liste normale est utilisée pour sauvegarder.

- ## Package Persistance :

1. **LoaderJoueur :**
   Cette classe permet de charger les joueurs. Il possède une méthode statique *loadJoueurs* permettant de charger les joueurs.
2. **SaverJoueurs :**
   Cette classe permet de sauvegarder les joueurs. Il possède une méthode statique *saveJoueurs* permettant de sauvegarder les joueurs.

- ## Package Colisionneurs :

1. **Colisionneur :**
   Cette interface précise la méthode utilisée par tous les collisionneurs
2. **ColisionneurObjet :**
   Cette classe définit la méthode *isOkayToMove* en fonction des objets.
3. **ColisionneurGarde :**
   Cette classe définit la méthode *isOkayToMove* en fonction des gardes et de leur vue.
4. **ColisionneurMur :**
   Cette classe définit la méthode *isOkayToMove* en fonction des murs.
5. **ColisionneurSortie :**
   Cette classe définit la méthode *isOkayToMove* en fonction des sorties.

- ## Package Deplaceurs :

1. **Deplaceur :**
   Cette interface précise la méthode utiliser pour déplacer les entités.
2. **DeplaceurJoueur :**
   La classe permet de déplacer le joueur.
   Elle possède la méthode de l'interface.
   Elle possède un colisionneurObjet, un colisionneurMur, un colisionneurGarde et un colisionneurSortie.
   Elle possède aussi une méthode *traitementMouvement* permettant de déplacer le joueur en fonction d'une touche pressée.
3. **DeplaceurGarde :**
   La classe permet de déplacer un garde.
   Elle possède la méthode de l'interface.
   Elle possède une PersoJoueur représentant le personnage joué par le joueur.

- ## Package Boucleurs :

1. **Boucleur :**
   La classe représente le boucleur, implémentant les boucles de jeu.
   Elle possède un BooleanProperty gameOver utilisée quand le joueur est en GameOver. Celui-ci bloque la boucle.
   Elle possède aussi une liste d'observateurs ainsi qu'une méthode *subscribe* et *notify*
2. **BoucleurJeu :**
   La classe hérite de Boucleur.
   Elle définit la boucle du jeu.

- ## Package Utils :

1. **Timer :**
   Le Timer représente le compte à rebours et implémente l'interface Observateur.
   Il possède un int(cptAct utillisé pour la boucle) le temps, ainsi qu'une StringProperty utilisée pour l'affichage ainsi qu'un boucleurJeu.
   Il possède aussi une méthode *refreshTimer* permettant d'actualiser la StringProperty pour la vue.
2. **Score :**
   Le Score représente le score du joueur.
   Il possède un int représentant le score ainsi qu'une StringProperty servant à afficher le score correctement.
   Il possède aussi une méthode *refreshScore* permettant d'actualiser la StringProperty pour la vue ainsi qu'une méthode addScore utilisée pour ajouter des points au Score.

3. **IAGarde :**
   Le Timer représente le compte à rebours et implémente l'interface Observateur.
   L'IAGarde représente l'IA des gardes.
   Elle possède un boucleurJeu, un int(cptAct utillisé pour la boucle), un déplaceurGarde, un Garde ainsi que la Carte
4. **LecteurDeCarte :**
   La classe est utilisée pour lire une carte depuis un fichier texte et pour renvoyer un tableau de 2 dimensions représentant la carte.
   Elle possède la longueur et la hauteur de la carte.
   Elle possède une méthode *lireCarte* permettant de lire un fichier et de retourner un tableau.
5. **Carte :**
   La classe représente un niveau (la carte) du jeu.
   Elle contient une longeurP et une largeurP (dimensions de la carte) ainsi que d'une liste d'Entités contenant toutes les Entités actuellement sur la carte.
   Elle possède plusieurs méthodes, comme *destroy*, permettant de détruire un Objet, *clearCarte*, permettant de vider la liste d'Entités, *clearVueGarde* , permettant d'enlever toutes les VueGarde de la carte, *lireCarte*, permettant de lire la carte ainsi que *loadNewLevel*, permettant de charger un nouveau niveau.
6. **Position :**
   La position représente les coordonnées d'une Entité.
   Elle continent la coordonnée x et y.

- ## Package Acteurs :

1. **Entité :**
   Classe abstraite représentant toute Entité.
   Elle possède une imageView représentant le Sprite, un boolean pour savoir si l'Entité est solide (non traversable) ainsi qu'une Position.

Toutes les classes suivantes héritent d'Entité.

2. **Sortie :**
   Classe représentant une sortie.
3. **Mur :**
   Classe représentant un mur.
4. **Objet :**
   Classe représentant un objet à ramasser
5. **VueGarde :**
   Classe représentant la vision (champ de vision) d'un garde.

6. **Personnage :**
   Classe représentant un personnage.
   Possède une direction, utilisée pour tourner le Sprite
   Possède une méthode *changeDirection* pour changer la rotation du Sprite
7. **PersoJoueur :**
   Hérite de Personnage.
   Classe représentant le personnage joué par le joueur.
8. **Garde :**
   Hérite de Personnage.
   Classe représentant un Garde
   Possède une distanceVue, un compteur de Garde statique (utilisé pour les ids) ainsi qu'un id pour identifier le Garde.