### Projet LU2IN002 - 2020-2021

|  |
| --- |
| Numéro du groupe de TD/TME : |

|  |  |  |
| --- | --- | --- |
| *Nom :* Ramdani | *Nom :* | *Nom :* |
| *Prénom :* Cyrena | *Prénom :* Miguel | *Prénom :* |
| *N° étudiant :* 3805942 | *N° étudiant :* 28600492 | *N° étudiant :* |

|  |
| --- |
| *Thème choisi (en 2 lignes max.)* |
| Le thème est inspiré du quidditch, un sport de balle fictif issu de la saga Harry Potter créé par J.K Rowling. Il s’agit d’un jeu multijoueur qui vous invite à chevaucher vos balais volant et défier vos amis. |

|  |
| --- |
| *Description des classes et de leur rôle dans le programme (2 lignes max par classe)* |
| * **Information** : classe static printant sur la ligne de commande les informations sur le jeu, équipe, ou les balles. * **Start** : Donne accès au menu d’accueil sur ligne de commande et démarre le Jeu en * **TropDErreurException** : exception lance si l’utilisateur faire trop d’erreur en rentrant ses configurations avant de commencer le jeu. * **Jeu** : Commence le fil de jeu, initialise la fenêtre de jeu, initialise les 2 joueurs, le nombre de balles)par sélection ou de façon aléatoire), ajoute les objets de jeu. * **KeyInput**: permet de gérer l’utilisation du clavier et permet de déplacer les joueurs respectifs. * **GameObject**: classe abstraite donnant les prérequis de chaque objet apparaissant sur le terrain (position, affichage, taille, identifiant, vitesse). * **UseGameO**: Permet de gérer tous les GameObject en les mettant dans une ArrayList. Détecte les collisions entre objets et permet de les ajouter/supprimer/update. * **Joueur**: chaque joueur a un nom, une équipe et une couleur. Il apparait sous la forme d’un carré. S’il entre en contact avec un Cognard il sera immobilisé pendant un délai. * **Balle**: chaque balle rebondit sur les rebords. Elles apparaissent tous avec une forme circulaire mais on des tailles, couleurs, comportements différents. * **Souaffle**: initialise au centre du terrain (magenta), est mobile seulement après un contact avec un joueur et perd de la vitesse progressivement. En contact avec l’un des goals il disparait et faire gagner un point à l’autre équipe. * **Cognard**: avec cette option les Cognards apparaissent (en noir) aléatoirement et se déplacent toujours la même vitesse (avec parfois des changements de trajectoire) * **VifOr**: avec cette option le Vif d’Or apparait (en jaune) aléatoirement à intervalle irrégulier en se téléportant. L’attraper fait gagner le match. * **Saveable :** Un fichier est saveable si on peut en enregistrer les donner dans un fichier. On enregistre differentes choses en fonction de la classe dont il s’agit |

|  |
| --- |
| *Schéma UML des classes vision fournisseur (dessin “à la main” scanné ou photo acceptés)* |
|  |

|  |  |
| --- | --- |
| *Checklist des contraintes prises en compte:* | *Nom(s) des classe(s) correspondante(s)* |
| Classe contenant un tableau ou une ArrayList | * UseGameO : ArrayList<GameObject> object * Jeu |
| Classe avec membres et méthodes statiques | * Information |
| Classe abstraite et méthode abstraite | * GameObject : tick(),render(Graphics g) * Balle |
| Interface | * Saveable |
| Classe avec un constructeur par copie ou clone() | * Cognard * Joueur |
| Définition de classe étendant Exception | * TropDErreurException |
| Gestion des exceptions | * Start * Jeu |
| Utilisation du pattern singleton | * Jeu |

|  |
| --- |
| *Présentation de votre projet (max. 2 pages) : texte libre expliquant en quoi consiste votre projet.* |
| Un match de quidditch. :  il faut gagner |

|  |
| --- |
| *Copier / coller vos classes et interfaces à partir d'ici :* |