Rapport Projet IPO / Groupe 2 / Lilian Sartiaux & David Ribeiro

**Partie 1** : Implémentation des méthodes pour déplacer la grenouille via une lecture de touche clavier et savoir quand la partie est gagnée ou perdue.

* Implémentation de la méthode *move()* dans la classe Frog.(Difficulté : Nous avons rencontré des difficultés pour faire bouger la grenouille et à cause du fonctionnement de l’objet case.)
* **Implémentation des méthodes pour vérifier si le jeu est perdu ou gagné**. (Nous avons utilisé des méthodes existantes dans les Interfaces, pour garantir la cohérence du projet.)

**Partie 2** : Implémentation des méthodes nécessaires à l’affichage des voitures et de leur déplacement en continu. (Classes Environment, Lane, Car).

* *Nous avons cherché à modifier le moins possible le squelette et notamment les relations entre les classes du projet (via les import. en début de fichier).*
* *Nous avons déduit les paramètres des méthodes et ce qu’elles renvoient en fonction des appels de méthode faits un peu partout dans le projet .*
* Dans le constructeur Lane : *update()* déplace les voitures déjà existantes sur la voie pour libérer la première case et que *mayAddCar()* fonctionne.
* *Environment.Update()* se contente de répéter *Lane.Update()* pour chacune de ses voies : ainsi, le « gros du travail » se fait dans *Lane.Update().*
* Dans *Lane.Update(),* le timer n’est réinitialisé à 0 que lorsque les voitures se sont déplacées.
* Utilisation des méthodes random importées dans la classe Game (dans les constructeurs des classes Car et Lane).
* Difficulté : Après l’implémentation des méthodes d’update, à l’exécution nous avions une fenêtre grise avec la grenouille mais sans voiture. Nous savions que ce n’était pas dû aux méthodes update, car nous nous étions assurés qu’il y avait systématiquement un appel à addToGraphics pour chaque voie du tableau de voies dans Environment.
* Difficulté : Certaines voitures sont buguées et sont invisibles ou alors la hitbox des voiture est elle aussi bugué.

**Partie 3** : Implémentation des méthodes nécessaires à un jeu « infini ». (EnviInf et FrogInf).

* Dans EnviInf : on a un tableau des lignes affichées et un tableau qui recense toutes les lignes.
* Les méthodes update actualisent le tableau infini (lanesInf) en permanence ( Difficulté : Nous n’avons pas réussi a afficher les lignes qui sont sensées descendre, mais elles sont quand même présentes dans le tableau lanes).
* Dans la classe FrogInf, nous avons changer la restriction qui empêchait la grenouille de monté et qui restait bloqué dans la fenêtre.
* Dans la classe Game on a rajoute des attributs score et limite qui définissent le score du joueur.

**Partie 4** : Implémentation des méthodes pour avoir un timer. (Classe Game).

* Utilisation de librairies conçues pour mesurer le temps écoulé
* Difficulté : Arrêter le chronomètre après que la partie soit perdue. Solution :

import java.time.Instant; import java.time.Duration;

Utilisation d’un booléen pour stopper le chronomètre, autrement le chrono continue de tourner.

* On affiche « le temps d’arrivée – le temps de départ » (car ces méthodes sont des horloges, donnent l’heure qu’il est).
* Affichage en secondes via la méthode .*toMillis()*.