Rapport Projet IPO / Groupe 1 / Lola LETERRIER & Zahara BEROKHROKH

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

- **Implémentation de la méthode *move()* dans la classe Frog.** (Difficulté : Nous avons rencontré des difficultés pour comprendre comment fonctionne la lecture de touches de clavier.)
- **Implémentation des méthodes pour vérifier si le jeu est perdu ou gagné**. (Nous avons utilisé des méthodes existantes dans d'autres classes, pour garantir la cohérence du projet.)

Partie 2 : Implémentation des méthodes nécessaires à l'affichage des voitures et de leur déplacement en continu, selon un certain tempo. (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 (voir commentaires dans les fichiers du projet).*
- Dans le constructeur Lane : *moveCars()* déplace les voitures déjà existantes sur la voie pour libérer la première case et que *mayAddCar()* fonctionne.
- L'objectif du booléen en paramètre dans la méthode *move()* (classe Car) est de faire un appel à la méthode *addToGraphics* (méthode en private) pour chaque voiture, qu'elle se déplace ou non.
- *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 : Nous pensions que isSafe (dans la classe Lane) devait vérifier si la case était perdante et non pas seulement libre.
- 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.
  Correction très simple : nous n'avions pas rempli le tableau de voitures dans le constructeur de Lane, du coup il n'y avait aucune voiture à afficher !

Partie 3 : Implémentation des méthodes nécessaires à un jeu « infini ». (Classes EnviInf, Lane, Car)

- Dans EnviInf : on a un tableau des lignes affichées et un tableau qui recense toutes les lignes.
- Implémentation de la méthode setOrd dans la classe Car : la position de la voiture va changer, on crée une nouvelle case, sur la même abscisse, sur l'ordonnée qu'on donne.

- Ensuite on modifie les ordonnées des voies. A l'échelle de l'environnement, c'est ce qui permet de faire afficher certaines lignes et en faire disparaître d'autres.
- Les méthodes update actualisent le tableau infini (lanesInf) en permanence
- Dans la classe FrogInf, un nouvel attribut : ord et une méthode (getOrd()) pour renvoyer le score à la fin de la partie, dans la méthode testLose().

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

- Utilisation de librairies conçues pour mesurer le temps écoulé

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

- Difficulté : Arrêter le chronomètre après que la partie soit perdue. Solution : 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 *.toSeconds()*.

Partie 5 : Tentative d'implémentation des méthodes pour avoir un jeu à 2 joueurs.

Problèmes rencontrés :

- Séparation de lecture de touches de clavier : comment faire pour attribuer un ensemble de touches à chaque joueur ?
- Comment avoir 2 joueurs ? Notre idée était de séparer l'écran en 2, nous avons réussi à ne faire qu'un côté de l'écran.