Développement initiatique

Sujet 7 : Objets et classes : course de voitures

1 Course simple

Nous allons d'abord écrire un programme qui organise une course entre deux voitures, en ligne droite d'un point de départ à un point d'arrivée. Ecrire les classes suivantes.

La classe `Voiture`

Une voiture est caractérisée par son nom (chaîne de caractères), sa position sur un axe à partir d'un point origine (entier positif ou nul) et sa vitesse (entier positif ou nul). Pour la simulation à l'écran, l'axe sera horizontal et le point origine tout à fait à gauche de l'écran.

La classe `Voiture` a donc trois attributs : `nom`, `position` et `vitesse`. Elle possède le constructeur et les méthodes suivants :

```
public Voiture(String unNom, int uneVitesse)
/* pré-requis : (à compléter)
 * action : crée une voiture de nom unNom et de vitesse uneVitesse
 *          placée à l'origine
 */
public String toString()
/* résultat :  retourne une chaîne de caractères contenant les caractéristiques
 *             de this (sous la forme de votre choix)
 */
public String toString2()
/* résultat :  retourne une chaîne de caractères formée d'une suite d'espaces
 *             suivie de la première lettre du nom de this, suivie d'un retour
 *             à la ligne, le nombre d'espaces étant égal à la position de this.
 */
public String leNom()
/* résultat :  retourne le nom de this
 */
public boolean depasse(int limite)
/* résultat :  retourne vrai si et seulement si la position de this est
 *             supérieure ou égale à limite
 */
public void avance()
/* pré-requis : (à compléter)
 * action :  fait avancer this d'une distance égale à sa vitesse */
```

```
public void auDepart()
/* action :  place this au départ de la course (à l'origine) */
```

Conseil : La méthode `charAt(int i)` invoquée par un objet de la classe `String` renvoie le i^e caractère de la chaîne (en commençant à l'indice 0).

La classe `Course`

Une course (entre 2 voitures) est caractérisée par les 2 voitures en compétition, qui sont des instances de la classe `Voiture`, ainsi que par la longueur de la piste, qui est un entier strictement positif.

Le déroulement d'une course se passe de la façon suivante. Au début, les 2 voitures sont placées sur la ligne de départ. A chaque étape, l'une des 2 voitures, choisie aléatoirement, avance. La course s'arrête dès qu'une voiture a franchi la ligne d'arrivée.

La classe `Course` a donc 3 attributs : `voit1`, `voit2` et `longueurPiste`. Elle possède le constructeur et les méthodes suivants :

```
public Course (Voiture uneVoit1, Voiture uneVoit2, int longueur)
/* pré-requis : (à compléter)
 * action :     (à compléter)
 */
public String toString()
/* résultat :  retourne une chaîne de caractères contenant les caractéristiques
 *             de this (sous la forme de votre choix)
 */
public Voiture deroulement()
/*
 * action : Simule le déroulement d'une  course entre this.voit1 et this.voit2
 *     sur une piste de longueur this.longueurPiste.
 *     this.voit1 et this.voit2 sont d'abord placées sur la ligne de départ.
 *     Ensuite, jusqu'à ce qu'une voiture franchisse la ligne d'arrivée, l'une
 *     des deux voitures est itérativement sélectionnée aléatoirement et avance.
 *     Un affichage des deux voitures (représentées par la première lettre de leur
 *     nom) à leur position respective à chaque étape permet de suivre la course.
 * résultat : la voiture gagnante.
 */
```

Conseils :

- La méthode de classe `Ut.randomMinMax(int min, int max)` permet de renvoyer un nombre entier pseudo-aléatoire compris entre `min` et `max`.
- Dans la console, pour simuler l'évolution des voitures, vous pouvez utiliser les procédures `Ut.clearConsole()` et `Ut.pause(int timeMilli)` entre deux étapes de la course.

La classe `MainCourse`

La classe `MainCourse` permet d'exécuter des courses entre des voitures. Dans la version simple, la procédure principale appelle une fonction `courseAller` qui crée deux instances `v1` et `v2` de la classe `Voiture` et une instance de la classe `Course` mettant en compétition `v1` et `v2`. `courseAller` déclenche ensuite le déroulement de la course entre `v1` et `v2` et affiche finalement le nom de la voiture gagnante.

Tester votre programme.

2 Course avec un aller-retour

Modifier les classes précédentes et écrire une fonction `courseAllerRetour` dans la classe `MainCourse` pour effectuer une course avec un aller-retour, le point de départ devenant également le point d'arrivée.

Ajouter à la classe `Voiture` l'attribut entier `sens` qui représente le sens de déplacement sur l'axe : 1 si la voiture s'éloigne de l'origine et -1 si elle s'en rapproche. Ajouter également à cette classe la méthode :

```
public void faitDemiTour()
/* action :  fait faire un demi-tour à this */
```

Ajouter dans la classe `Course` une méthode `deroulementAR` et dans la classe `Voiture` une méthode `avanceAR` qui gère le demi-tour lors du dépassement de la limite de la piste (à votre choix).

3 Extensions possibles

1. Donner aux 2 voitures des vitesses différentes et des probabilités différentes d'être choisies pour avancer (en équilibrant les chances pour une course équitable !).
2. Défavoriser une voiture au départ, mais pour équilibrer les chances, permettre à une voiture d'accélérer quand son écart avec la voiture de tête devient trop important (pour cela, il vous faudra rajouter des méthodes à la classe `Voiture` pour permettre à une voiture d'accélérer et pour pouvoir évaluer l'écart entre 2 voitures).
3. Choisir comme itinéraire de la course k longueurs de piste, où k est un entier positif quelconque (pair ou impair !). Ce type de course généralise à la fois la version de base ($k = 1$) et la course aller-retour ($k = 2$). Il faut prévoir un rebondissement au bout des $k - 1$ premiers parcours de piste et une arrivée au bout du k^e parcours.

   Pour cette version, il faut reprendre et unifier les codes écrits pour la version de base et pour la version aller-retour.
4. Simuler une course entre un nombre quelconque de voitures (utiliser un tableau d'instances de la classe `Voiture`).
5. Faire évoluer la voiture dans un plan : remplacer l'attribut `position` par les coordonnées `x` et `y`, et l'attribut `sens` par une direction pouvant avoir 4 valeurs possibles (les 4 points cardinaux).

Pour l'affichage, on peut utiliser une matrice de caractères ou bien donner accès aux coordonnées des voitures en ajoutant des accesseurs `getLigne()` et `getColonne()` dans la classe `Voiture`.