Projet Dev

Création d'un prototype à partir de code existant et d'une documentation

---

Objectif

Vous disposez d'un code de départ pour le projet.

La première partie de votre travail consiste à :

- Le comprendre, à l'aide de diagrammes UML fournis.
- Le compléter, pour obtenir un prototype du jeu.

Remarque : le code fourni est à vocation pédagogique, certains choix ne sont pas les meilleurs en terme d'efficacité

Etape 1 : traduction des associations

Diagramme à exploiter : diagramme de classes

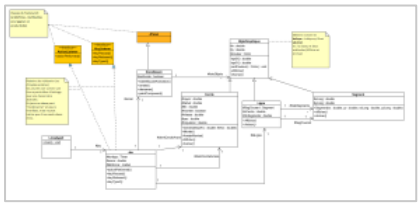

Figure 1: diagramme de classes de l'application.

Chaque association d'un diagramme de classes doit être implémentée (traduite en code source) lors de la programmation. Le diagramme peut contenir des informations qui spécifient clairement comment cette implémentation doit se faire : c'est le cas avec le diagramme qui vous est fourni.

Figure 2: une association avec des indications pour l'implémentation.

A faire : complétez ou modifiez le code source fourni pour que la traduction des associations soit conforme à ce que spécifie le diagramme de classes. Soyez précis, exploitez tous les éléments donnés par le diagramme de classes (nom des rôles, multiplicités, navigabilité, visibilité).

Indications :

- le code fourni est exécutable, mais n'affiche rien. Après cette étape il n'affichera toujours rien : soyez patients, il faut réaliser l'étape suivante pour voir apparaître les « objets graphiques ».
- Pour traduire une mutliplicité *, utilisez une ArrayList.

Etape 2 : programmation de méthodes à partir de « spécifications »

Diagrammes à exploiter : diagrammes de classes, de séquences et d'activités

L'objectif de cette étape est de **programmer des méthodes selon des spécifications données par des diagrammes de séquences et d'activités**. Lorsqu'elle sera terminée vous devriez voir les objet

1. Mettre en place les **appels de méthodes** entre classes nécessaires à l'animation et l'affichage. Vous n'avez **pas à écrire pour le moment le code précis de l'affichage** en lui-même.
   Le diagramme de séquences vous indique en partie, sur un exemple, comment vous devez procéder.

2. Ecrire le code de la méthode `ajouterObjet(ObjetGraphique unObjet)` de la classe ZoneDessin. Son rôle est de placer l'objet passé en paramètre dans la liste des objets graphiques connus par ZoneDessin, c'est-à-dire la liste des objets qui doivent être dessinés.

3. **Compléter** le code des méthodes d'animation et d'affichage selon les **indications données en commentaires dans le code** et la description faite dans le **diagramme d'activités**. Vous devez :
   - Créer, afficher et animer une ligne de segments
   - Créer, afficher et animer le cercle : l'animation de la chute sous l'effet de la gravité est faite, vous devez gérer l'appui sur la touche qui permet de donner une impulsion vers le haut

Indication : essayez de suivre le déroulement logique du processus d'animation et d'affichage, pour ne rien oublier (ou pour debugger). C'est-à-dire :

- Vérifiez que les listes d'objets à afficher sont bien créées et initialisées
- Vérifiez que chaque méthode d'animation est complète
- Vérifiez que chaque méthode d'affichage est complète

Remarque : la ligne est créée avec une position initiale en x valant 800. Cela signifie qu'elle n'est pas visible, elle s'affiche « juste à côté » de la zone dessin, en dehors de l'écran. Dans un premier temps, si vous voulez vérifier qu'elle s'affiche et que vous n'avez pas programmé l'animation, vous pouvez décaler un peu sa position initiale pour la ramener dans l'écran.

Etape 3 : conception puis implémentation

Votre programme ne permet pas, pour le moment, de vérifier si la ligne passe dans le cercle ou non. Vous devez maintenant **concevoir et programmer** cette partie du jeu.

Démarche :

1. Concevez l'architecture qui permettra d'apporter cette fonctionnalité à votre application (quelles méthodes ajouter ou utiliser dans les classes ? Nécessité de créer de nouvelles classes ? Quelles communications entre classes sont nécessaires ? …)

2. Concevez l'algorithme de la méthode qui doit vérifier si la ligne est dans le cercle ou non.

Représentez avec les diagrammes UML adéquats votre solution.

3. Programmez ce que vous avez conçu.