Laboratoire 9 - Disques

Leonard Cseres, Aladin Iseni

20 janvier 2025

Table des matières

1 Introduction

Ce projet a pour objectif de créer une application graphique permettant la gestion et la manipulation de disques via une interface utilisateur simple. Afin d'assurer la maintenabilité, la lisibilité et l'extensibilité, une architecture modulaire a été adoptée. Cette approche garantit que chaque composant est indépendant tout en permettant une interaction fluide entre eux.

---

2 Conception et Architecture

2.1 Structure

Le projet est organisé en plusieurs packages logiques, chacun ayant une responsabilité clairement définie :

- `model` : Contient les classes représentant la logique métier, notamment les disques (`Disk`) et leur gestionnaire (`DiskManager`).
- `view` : Comprend les éléments graphiques (ex. `DrawingPanel`, `MainFrame`), responsables de l'affichage et de l'interaction utilisateur.
- `action` : Définit des classes dédiées aux actions spécifiques (création et déplacement de disques) pour séparer les comportements et simplifier le code.
- `util` : Regroupe les utilitaires partagés, comme la gestion des couleurs via la classe `ColorPalette`.

2.1.1 Justification

1. **Modularité** : Chaque package et chaque classe ont une responsabilité unique, ce qui rend le code plus facile à comprendre et à modifier.
2. **Réutilisation** : Les actions (`Action`, `CreateAction`, `MoveAction`) encapsulent des comportements spécifiques, facilitant leur réutilisation ou leur extension pour de nouvelles fonctionnalités.
3. **Testabilité** : Les composants bien isolés permettent de tester chaque élément indépendamment (ex. tester la logique de `DiskManager` sans dépendre de l'interface graphique).

3 Caractéristiques Principales

3.1 Création de Disques

- L'utilisateur peut dessiner un disque en cliquant et en déplaçant la souris.
- La taille du disque est calculée dynamiquement en fonction des points de départ et d'arrivée.
- Cette fonctionnalité est implémentée dans la classe `CreateAction`, ce qui garantit une encapsulation claire du processus.

3.2 Déplacement de Disques

- En maintenant la touche **Shift** enfoncée, un utilisateur peut déplacer un disque en cliquant dessus et en le faisant glisser.
- La position du disque est recalculée en tenant compte d'un décalage entre la souris et le centre du disque, pour un comportement fluide.
- La classe `MoveAction` gère cette logique, séparant le déplacement de la logique d'affichage.

3.3 Suppression de Disques

- Un clic droit sur un disque supprime le disque supérieur à cet emplacement.

---

4 Tests Effectués

Création de disques	
Vérification de la taille et de la position des disques créés	Vrai
Validation de la gestion des couleurs à chaque création	Vrai
Déplacement de disques	
Test de l'interaction fluide avec le disque sélectionné	Vrai
Validation que le disque reste au premier plan lors du déplacement	Vrai
Suppression de disques	
Vérification de la suppression correcte du disque supérieur à un emplacement donné	Vrai
Interaction utilisateur	
Vérification des combinaisons de clics pour dessiner, déplacer et supprimer des disques	Vrai

5 Conclusion

L'architecture modulaire et orientée objet garantit un code clair, extensible et fiable. La séparation des responsabilités facilite l'ajout de nouvelles fonctionnalités, comme de nouvelles actions. Les tests valident le bon fonctionnement et réponds au cahier des charges.

A Annexes

A.1 Listing Java

c.f. page suivante.