École Centrale Lyon

UE PRO
LaTeX Approfondi
Rapport

BE Agaignée

Élèves :
Naoya SENOO
Marwane AGREBI

Enseignant :
Stéphane Derrode

1er octobre 2024

Table des matières

1 Introduction

Ce rapport explique la conception, l'implémentation et le processus de développement du jeu "Jeu Araignée", développé en utilisant Java. Ce jeu est un jeu de plateau compétitif où deux joueurs placent et déplacent leurs pions à tour de rôle, dans le but de remplir certaines conditions pour gagner.

Tout d'abord, une vue d'ensemble de la structure du code est présentée, suivie d'un diagramme UML qui illustre visuellement la structure des classes et les relations entre les objets. Ensuite, chaque fonctionnalité du jeu, de son lancement à sa fin, est décrite en détail. En particulier, l'interface utilisateur est expliquée, mettant en évidence comment les utilisateurs interagissent avec le jeu, en utilisant les éléments de l'interface tels que la disposition des écrans et les boutons.

Ce rapport met également l'accent sur la transition entre la phase 1 et la phase 2 du jeu, ainsi que sur la conception de l'interface utilisateur, pensée pour offrir une expérience conviviale. Les méthodes utilisées pour afficher les conditions de victoire et guider la progression des joueurs sont également abordées.

Ce rapport vise à fournir une compréhension systématique du processus de développement du "Jeu Araignée" et à explorer en détail la conception de l'interface utilisateur et l'implémentation des mécanismes du jeu en Java.

2 Fonctionnement

2.1 Introduction

Dans ce chapitre, nous allons détailler le fonctionnement d'une application Java. Cela inclut la structure générale du code, la gestion des ressources, et le flux d'exécution.

2.2 Architecture de l'application

L'architecture d'une application Java repose généralement sur les deux couches suivantes :

- **Interface utilisateur (IHM)** : Cette couche gère l'interaction avec l'utilisateur en Utilisant la bibliotèque Swing.
- **Logique métier** : La partie centrale où sont traitées les règles de gestion pour le jeu .

2.3 Exécution du programme

L'exécution du programme s'articule autour de plusieurs étapes, décrites ci-dessous :

1. **StartingPage** : C'est la fenêtre principale où vous avez le choix entre les options suivantes :
   - Commencer le jeu
   - Quitter l'application
   - Accéder à l'aide pour mieux comprendre le fonctionnement du jeu
2. **HelpPage** : Cette page fournit les informations nécessaires pour connaître les règles du jeu :
   - Le jeu se joue sur une grille de 3x3 cases, avec deux joueurs. Chaque joueur joue avec un symbole, soit "X" soit "O". Le premier joueur utilise les "X".
   - **Première phase** : Chaque joueur place son symbole (X ou O) dans une case de la grille, trois fois chacun, en alternance. Le gagnant est celui qui parvient à aligner trois symboles identiques (horizontalement, verticalement ou en diagonale). Si aucun joueur n'y parvient, il y a égalité ("draw").
   - **Deuxième phase** : Si l'égalité est déclarée, chaque joueur peut déplacer l'une de ses trois cases vers une case adjacente libre. Le jeu continue jusqu'à ce qu'un joueur réussisse à aligner trois cases avec son symbole. À ce moment-là, une fenêtre s'ouvre pour annoncer le gagnant, avec deux options : rejouer ou quitter.
3. **Fenêtre de gagnant** : Lorsque le gagnant est déterminé, une fenêtre spéciale apparaît pour indiquer le joueur victorieux. Cette fenêtre présente deux boutons :
   - Rejouer : pour relancer une nouvelle partie
   - Quitter : pour fermer l'application

2.4 Structure de Code

Nous avons écrit le code en respectant plusieurs principes de programmation, notamment la séparation des responsabilités. Les classes sont divisées en deux groupes distincts :

— **Interfaces Homme-Machine (IHM)** : Ces classes sont responsables de la gestion de l'interaction utilisateur, c'est-à-dire l'affichage des interfaces graphiques. Nous avons principalement utilisé l'héritage de la classe `JFrame` pour construire les différentes fenêtres de l'application, ce qui nous permet de gérer efficacement l'affichage et les événements associés.
— **Logique Métier** : Ces classes encapsulent la logique du programme, c'est-à-dire la manière dont le jeu fonctionne en interne. Elles s'occupent du traitement des données, de la gestion des règles du jeu, et de la coordination entre les différentes étapes du jeu. Ici, nous avons fait usage de l'interface `ActionListener` pour gérer les actions des boutons et assurer la réactivité de l'application. Ce groupe assure que le code reste flexible et réutilisable.

L'héritage et l'implémentation ont joué un rôle clé dans notre approche de développement. L'utilisation de `JFrame` et `ActionListener` comme classes mères a permis de simplifier la gestion de l'interface graphique et des événements utilisateurs. Cette séparation des responsabilités permet de maintenir un code propre, facilement modifiable et extensible, en plus de favoriser la lisibilité et la maintenance du programme.

2.5 Commentaire :

De plus, le code est accompagné de commentaires qui décrivent chaque étape du processus. Ces commentaires permettent de clarifier les intentions de chaque partie du code, facilitant ainsi la compréhension et la maintenance. Par exemple, chaque bloc de code important est précédé d'un commentaire qui explique son rôle, ce qui est essentiel pour aider d'autres développeurs à comprendre le fonctionnement global du programme sans avoir à déchiffrer chaque ligne de code individuellement.

2.6 Conclusion

Une application Java repose sur une architecture modulaire et un cycle d'exécution bien défini. Sa capacité à gérer les ressources de manière automatique et à gérer les exceptions en fait un langage robuste pour le développement d'applications complexes.

3 Diagramme UML : Diagramme de Classe

Le langage de modélisation unifié (UML) est un langage standardisé utilisé pour spécifier, visualiser, construire et documenter les artefacts d'un système logiciel. Il permet aux développeurs et aux concepteurs de représenter graphiquement la structure et le comportement d'un système, facilitant ainsi la communication entre les parties prenantes. Parmi les différents types de diagrammes UML, le diagramme de classe est l'un des plus essentiels. Il illustre les classes du système, leurs attributs, leurs méthodes, et les relations entre elles. Ce diagramme est fondamental pour la conception orientée objet, car il permet de modéliser la structure statique du système et de définir clairement les responsabilités de chaque classe.

3.1 Éléments d'un Diagramme de Classe

- **Classe :** Une classe est une description d'un ensemble d'objets ayant des attributs et des comportements communs. Elle est représentée par un rectangle divisé en trois parties : le nom de la classe, les attributs et les méthodes.
- **Attributs :** Les attributs décrivent l'état d'une classe. Ils sont généralement listés sous le nom de la classe.
- **Méthodes :** Les méthodes (ou opérations) définissent les comportements d'une classe et sont listées sous les attributs.
- **Relations :** Les relations entre classes peuvent inclure :
  - *Association :* Représente une relation entre deux classes. Une ligne simple est utilisée pour l'indiquer.
  - *Héritage :* Indique qu'une classe hérite d'une autre classe. Elle est représentée par une flèche avec un triangle.
  - *Composition :* Une relation forte où une classe fait partie d'une autre classe. Elle est représentée par un losange noir.
  - *Agrégation :* Une relation plus faible que la composition. Elle est représentée par un losange vide.

3.2 Interfaces et leur Implémentation

Les interfaces sont des types abstraits qui définissent un ensemble de méthodes qu'une classe doit implémenter. Voici quelques points clés concernant les interfaces :

- **Définition :** Une interface est une collection de méthodes abstraites (sans corps) que les classes doivent implémenter. Elle est représentée dans un diagramme de classe par un rectangle avec le nom de l'interface, généralement en italique.
- **Implémentation :** Lorsqu'une classe implémente une interface, elle s'engage à fournir des implémentations pour toutes les méthodes de l'interface. Cela est représenté par une flèche avec un trait plein allant de la classe à l'interface, avec une notation spécifique (un triangle ouvert).
- **Utilité :** Les interfaces permettent de créer des systèmes modulaires et flexibles, favorisant la réutilisabilité du code et facilitant les tests unitaires.

Voici ci-dessous le diagramme de classe pour notre application Java :

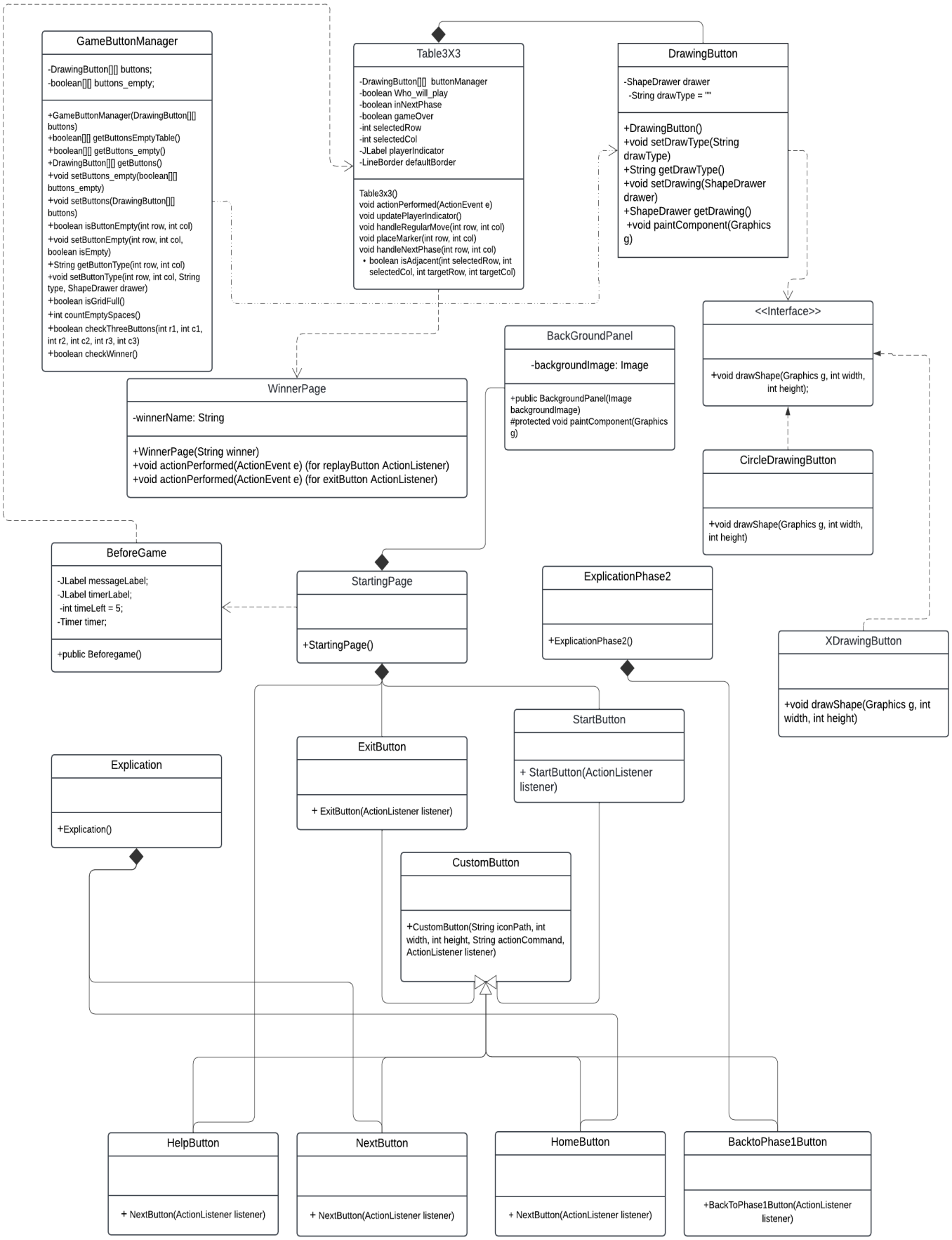

FIGURE 1 – Diagramme de Classe.

4 Réalisation

4.1 Démarrage du jeu

Lorsque le jeu est lancé, un écran de démarrage s'affiche comme illustré à la figure 2. Cet écran comporte un bouton "Start", un bouton "Exit" et un bouton "Help". Si l'on clique sur le bouton "Help", l'écran bascule vers une page expliquant les règles du jeu.

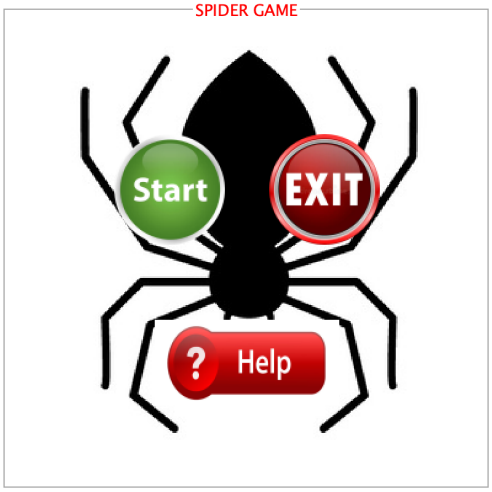

FIGURE 2 – Page de démarrage

4.2 Écran d'explication des règles

En cliquant sur le bouton "Help", une explication de la phase 1 du jeu est d'abord affichée (figure 3). Sur la partie droite de cet écran, un bouton fléché vers la droite permet de passer à l'explication de la phase 2. Au bas de l'écran, un bouton "Home" permet de revenir à l'écran de démarrage, tandis qu'un bouton "Start" permet de commencer le jeu.

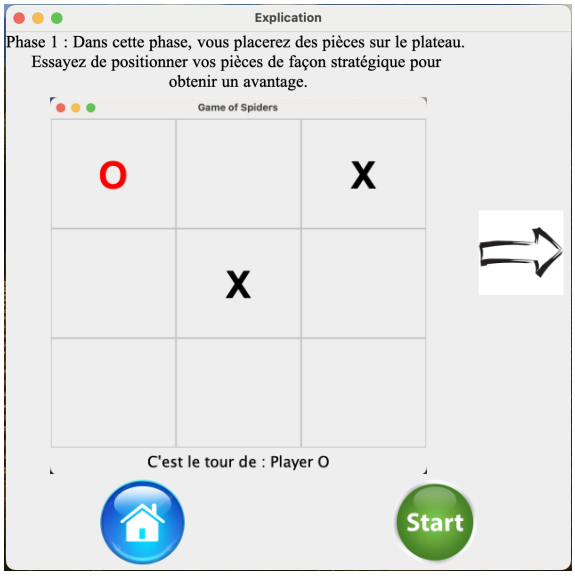

FIGURE 3 – Page d'explication de la phase 1

Dans l'écran d'explication de la phase 2 (figure 4), un bouton fléché vers la gauche permet de revenir à l'explication de la phase 1. De plus, comme pour l'écran de la phase 1, des boutons "Home" et "Start" sont présents en bas de l'écran.

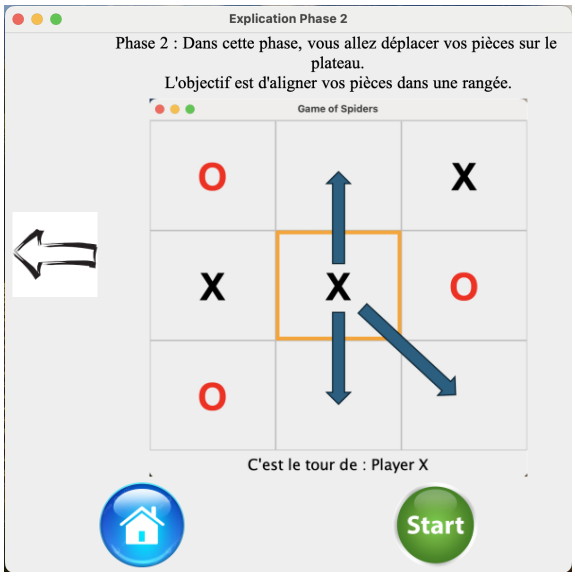

FIGURE 4 – Page d'explication de la phase 2

4.3 Début du jeu

En cliquant sur le bouton « Start », le compte à rebours commence et l'écran illustré dans la figure 5 apparaît. Une fois que les joueurs ont compris quel joueur utilisera quelle pièce, le jeu passe à l'écran de jeu. Sur l'écran de jeu, le premier joueur est le « joueur 1 (X) » et les mots « C'est le tour du joueur 1 (X) » sont affichés en bas de l'écran pour indiquer son tour. Après que le joueur X a placé une pièce, c'est au tour du joueur 2(O), qui apparaît à l'écran.

FIGURE 5 – Page de préparation de jeu (5 secondes)

FIGURE 6 – Page de Phase 1

4.4 Passage de la phase 1 à la phase 2

Lorsque les deux joueurs ont placé chacun leurs trois pions, un message s'affiche pour signaler la transition vers la phase 2 (figure 7).

FIGURE 7 – Passage de la Phase 1 à la Phase 2

4.5 Déclaration de victoire

Si un joueur parvient à aligner ses pions verticalement, horizontalement ou en diagonale, un message s'affiche pour annoncer la victoire de ce joueur, et une nouvelle page est affichée (figure 8).

FIGURE 8 – Page de déclaration de victoire

4.6 Réinitialisation du jeu ou sortie

Lorsque l'un des joueurs remporte la partie, un écran indiquant le vainqueur s'affiche en grand. Au bas de cet écran se trouvent un bouton "Reset" et un bouton "Exit" (figure 9). En cliquant sur le bouton "Reset", l'écran revient à celui de la figure 6, permettant ainsi de démarrer une nouvelle partie.

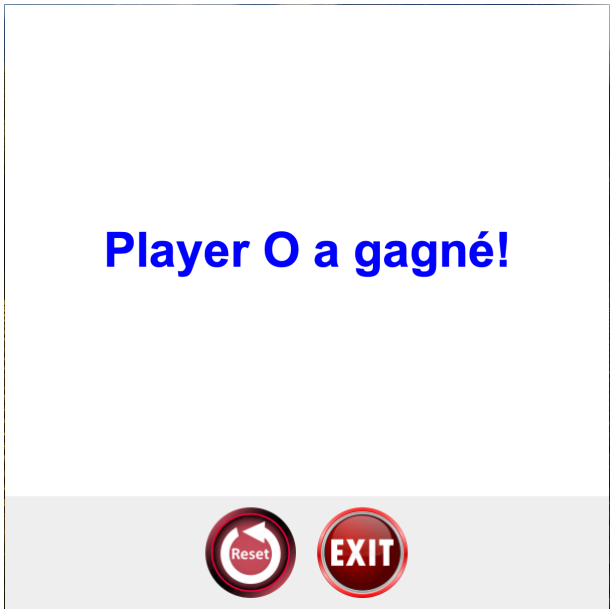

FIGURE 9 – Page de déclaration de victoire et de réinitialisationdu jeu

5 Conclusion

Le développement du jeu "Jeu Araignée" a permis de mieux comprendre plusieurs aspects de la programmation en Java. En particulier, on a appris l'importance de diviser le code en plusieurs classes, ce qui facilite grandement la maintenance future du programme. De plus, on a acquis des compétences dans la gestion des ActionListeners et des ActionCommands, ainsi que dans la création d'interfaces utilisateurs intuitives et efficaces.

Bien que ce projet soit conçu comme un projet ponctuel sans plan immédiat d'améliorations futures, si on avait plus de temps à consacrer, il serait intéressant de rendre le jeu plus dynamique. Par exemple, permettre aux joueurs de placer et déplacer les pions en cliquant et maintenant le bouton de la souris serait une amélioration notable. De même, des animations plus dynamiques lors des victoires, comme le traçage d'une ligne sur les trois pions alignés, pourraient enrichir l'expérience utilisateur.

Ce projet a offert l'opportunité d'approfondir la compréhension de la structure d'un programme interactif et de la manière dont les différentes composantes, telles que l'interface utilisateur et la logique de jeu, doivent fonctionner de manière cohérente. Il représente une étape importante dans le développement en tant que programmeur Java, et on est satisfait du résultat final atteint dans ce cadre limité.