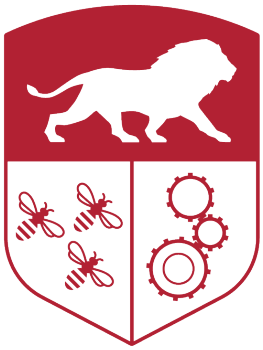

CENTRALE LYON

Applications concurrentes, mobiles et réparties
en Java
BE : Jeu de l'Araignée

Compte-Rendu BE Java - Jeu de l'Araignée

Élèves :
Jeremy Luccioni

Enseignants :
Stéphane Derrode

4 octobre 2025

Table des matières

1 Introduction

Ce rapport présente l'implémentation du jeu de l'Araignée développée en Java dans le cadre du BE de programmation orientée objet. Le jeu oppose deux joueurs qui tentent d'aligner trois pions sur une grille 3×3, d'abord en les plaçant (phase de placement), puis en les déplaçant vers des cases adjacentes (phase de déplacement).

L'implémentation complète comprend les deux phases du jeu et propose une interface graphique développée avec Java Swing.

2 Architecture du projet

2.1 Diagramme de classes UML

Le projet suit une architecture MVC (Modèle-Vue-Contrôleur) qui sépare clairement la logique métier de l'interface graphique.

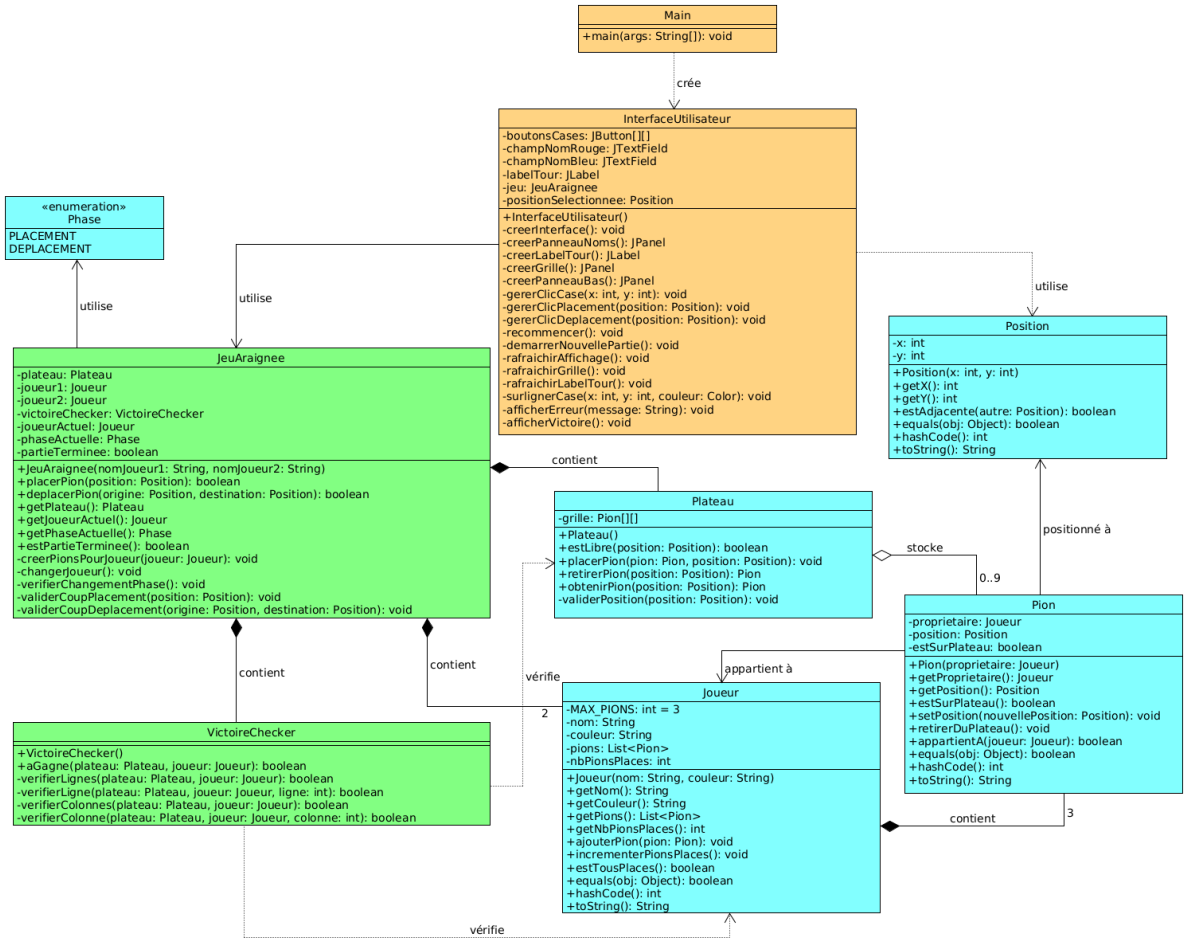

FIGURE 1 – Diagramme de classes UML du projet

2.2 Description des classes

Le projet est organisé en 9 classes principales :

2.2.1 Classes métier

— **Position** : Représente une position (x, y) sur la grille 3×3. Gère la validation des coordonnées et vérifie l'adjacence entre positions (déplacements horizontaux et verticaux uniquement, pas de diagonales).
— **Pion** : Modélise un pion appartenant à un joueur. Connaît sa position actuelle et son propriétaire.
— **Joueur** : Représente un joueur avec son nom, sa couleur (Rouge ou Bleu), et ses 3 pions. Suit le nombre de pions déjà placés.
— **Plateau** : Gère la grille 3×3 de jeu. Permet de placer, retirer et obtenir les pions, et vérifie si une position est libre.
— **Phase** : Énumération des deux phases du jeu (PLACEMENT et DEPLACEMENT).

2.2.2 Classes de logique

— **VictoireChecker** : Vérifie si un joueur a gagné en alignant 3 pions sur une ligne ou une colonne. Les diagonales ne sont pas considérées comme victoire.
— **JeuAraignee** : Classe centrale qui orchestre le jeu. Gère les tours des joueurs, les changements de phase, valide les coups et détecte les victoires. Délègue la vérification de victoire à VictoireChecker.

2.2.3 Classes d'interface

— **InterfaceUtilisateur** : Interface graphique en Java Swing. Affiche la grille de jeu, gère les clics utilisateur, et délègue toute la logique à JeuAraignee.
— **Main** : Point d'entrée du programme, lance l'interface graphique.

2.3 Principes de conception

L'architecture respecte plusieurs principes de la programmation orientée objet :
— **Séparation des responsabilités** : Chaque classe a une responsabilité unique et bien définie.
— **Encapsulation** : Les attributs sont privés, l'accès se fait via des getters/setters appropriés.
— **Composition** : JeuAraignee compose Plateau, Joueurs et VictoireChecker plutôt que d'hériter.
— **Validation centralisée** : Toute la validation des coups est dans JeuAraignee, l'interface ne fait qu'afficher.

3 Fonctionnalités implémentées

3.1 Phase de placement

Les joueurs placent alternativement leurs 3 pions sur la grille. L'interface affiche le nombre de pions restants à placer pour chaque joueur. La validation empêche de placer un pion sur une case déjà occupée.

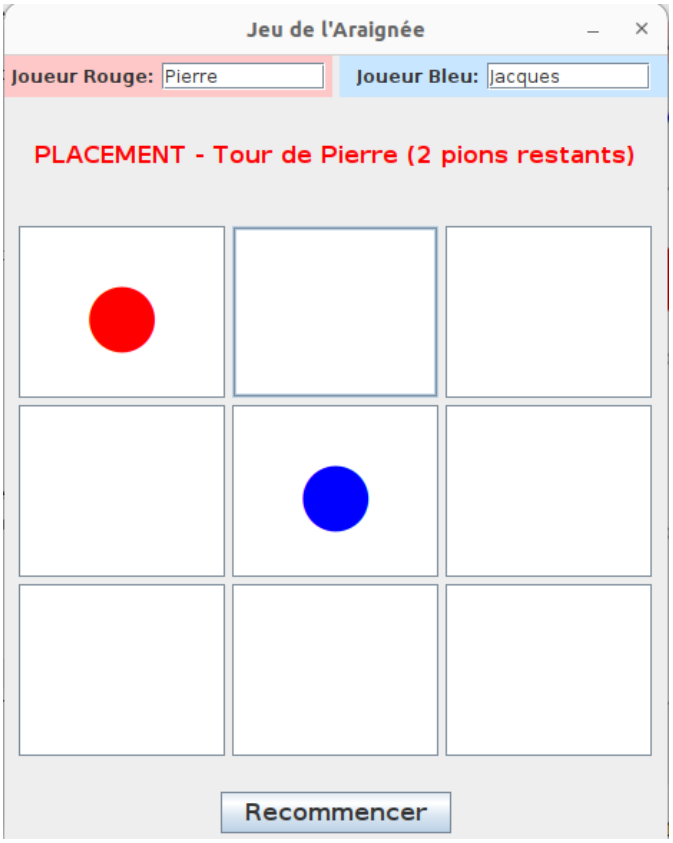

FIGURE 2 – Phase de placement en cours - Tour du Joueur Rouge (Pierre) avec 2 pions restants à placer

FIGURE 3 – Message d'erreur lors d'une tentative de placement sur une case déjà occupée

3.2 Phase de déplacement

Une fois tous les pions placés, le jeu passe automatiquement en phase de déplacement. Les joueurs peuvent déplacer un de leurs pions vers une case adjacente libre (haut, bas, gauche, droite uniquement).

Fonctionnalité de sélection/désélection :

— Clic sur un pion : sélection (surlignage en jaune)
— Clic sur le même pion : désélection
— Clic sur un autre pion du même joueur : changement de sélection
— Clic sur une case vide : tentative de déplacement

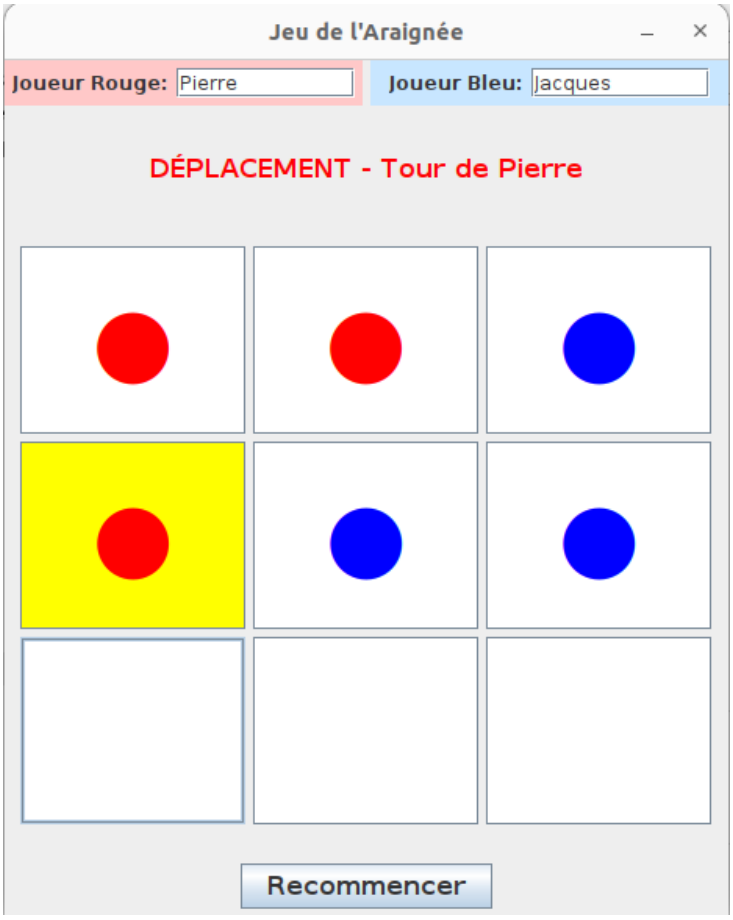

FIGURE 4 – Phase de déplacement - pion sélectionné en jaune

FIGURE 5 – Message pour rejouer

3.3 Détection de victoire

Le jeu détecte automatiquement un alignement de 3 pions sur une ligne ou une colonne (les diagonales ne comptent pas). Un message de victoire s'affiche et propose de rejouer.

FIGURE 6 – Message de victoire

3.4 Fonctionnalités supplémentaires

— **Personnalisation** : Les joueurs peuvent entrer leurs noms avant de commencer
— **Bouton Recommencer** : Permet de relancer une partie à tout moment
— **Affichage dynamique** : Le label indique toujours le joueur actuel et la phase en cours
— **Gestion d'erreurs** : Messages clairs en cas de coup invalide

4 Qualité du code

4.1 Commentaires

Le code est commenté de manière équilibrée :
— JavaDoc sur les méthodes publiques importantes (placerPion, deplacerPion, aGagne)
— Commentaires de classe expliquant le rôle de chaque classe
— Commentaires internes uniquement pour les logiques non-évidentes
— Pas de sur-commentaire : le code est auto-explicatif avec des noms de variables et méthodes clairs

4.2 Nommage

Respect des conventions Java :
— Classes en PascalCase (JeuAraignee, VictoireChecker)
— Méthodes et variables en camelCase (placerPion, joueurActuel)
— Constantes en MAJUSCULES (MAX_PIONS)
— Noms descriptifs et en français pour plus de clarté

4.3 Organisation

Le code est structuré et lisible :
— Méthodes courtes et focalisées sur une seule tâche
— Attributs privés avec getters appropriés
— Validation centralisée dans des méthodes dédiées
— Pas de code dupliqué

5 Difficultés rencontrées et solutions

5.1 Gestion de la sélection en phase de déplacement

Problème : Initialement, impossible de désélectionner un pion une fois sélectionné, ce qui obligeait à faire un déplacement même en cas d'erreur.

Solution : Implémentation d'une logique de sélection/désélection intelligente : un clic sur le pion déjà sélectionné le désélectionne, et permet de changer de pion sans avoir à déplacer.

5.2 Rafraîchissement de l'affichage

Problème : Le surlignage jaune persistait sur plusieurs cases simultanément.

Solution : Rafraîchissement complet de la grille avant chaque nouveau surlignage, en remettant systématiquement toutes les cases en blanc.

5.3 Validation des déplacements

Problème : Nécessité de vérifier qu'un déplacement se fait uniquement vers une case adjacente (pas de diagonales).

Solution : Méthode `estAdjacente()` dans la classe Position qui calcule les différences de coordonnées et valide le déplacement.

6 Améliorations possibles

— Historique des coups avec possibilité d'annuler
— Sauvegarde/chargement de parties
— Timer pour limiter le temps de réflexion
— Animation des déplacements de pions
— Mode sombre pour l'interface

— Statistiques (nombre de parties jouées, victoires par joueur) avec implémentation d'une DB

7 Conclusion

Ce projet a permis de mettre en pratique les concepts de la programmation orientée objet en Java : encapsulation, composition, séparation des responsabilités et d'utiliser Java Swing.

Le jeu est entièrement fonctionnel avec les deux phases implémentées, une interface graphique intuitive, et une gestion robuste des erreurs. Le code est propre, bien structuré et respecte les bonnes pratiques de développement Java.

Lien du dépôt Git : `https://github.com/Kadzzzzz/Jeu-Areignee.git`