Saif-Eddine ALJANE
Ouways VELLORE SHERIEF
Institution : IUT de Villetaneuse
BUT1 INFORMATIQUE
Groupe : Némée

Rapport de la SAE S2.01 : Dev.application

Année : 2024-2025

Composition du groupe :

- **VELLORE SHERIEF Ouways (Némée)**
- **ALJANE Saif-Eddine(Néméé)**

Instruction de lancement :

1 - Compiler les 12 fichiers .java

2 - Exécuter la commande java JeuEchec

Diagramme UML (sans variable ni méthode) :

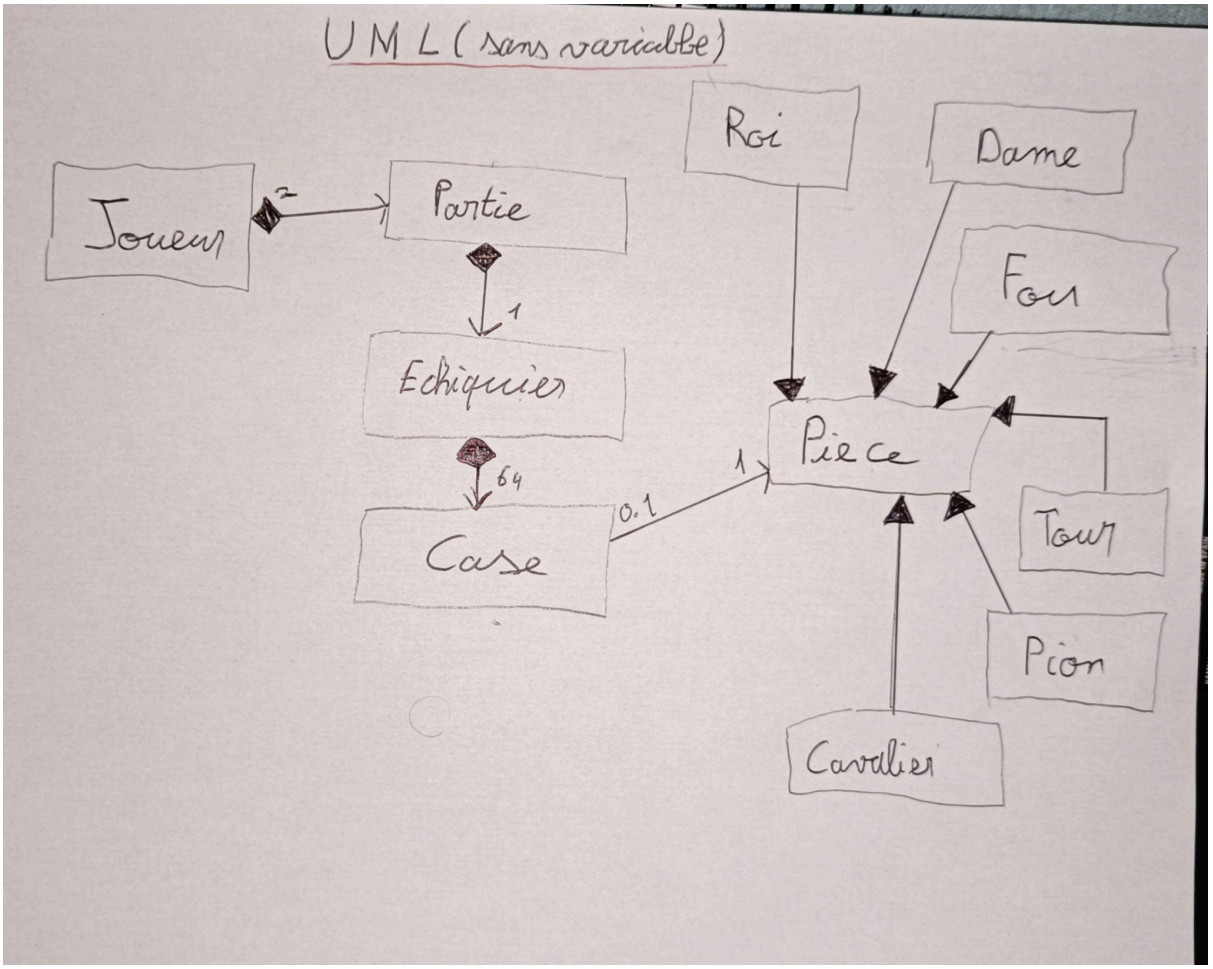
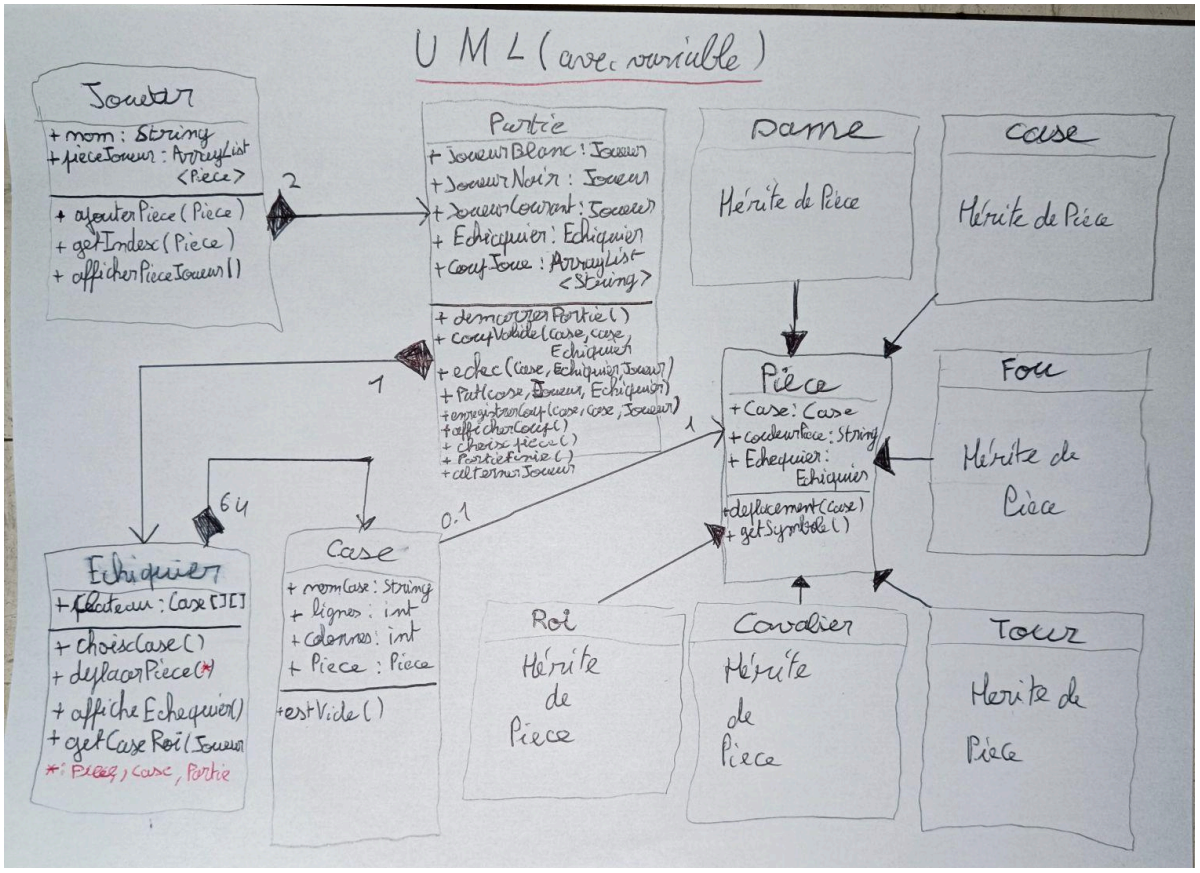


Diagramme UML (avec variable) :



Methode par Class :

Joueur :

+ **ajouterPiece**(Piece piece) :
  ajoute une Pièce dans la liste de pièce du Joueur ;

+ **getIndex**(Piece piece):
  donne l'index de la piece passer en paramètre

+ **afficherPieceJoueur**():
  affiche les piece du Joueur

Echiquier :

+ **choixCase**():
  permet de faire en sorte que l'utilisateur puisse choisir une case

+ **deplacerPiece**(Piece p,Case destination,Partie p):
  permet de deplacer la piece à la case de destination

+ **getCaseRoi**(Piece piece):

retourne l’index de la pièce

Case :

+ **estVide**():
  permet de vérifier si une case est vide ou pas

Piece :

+ **deplacement**(Case caseDestination);
  verifie si la case est dans les case de deplacement légal de la pièce

+ **getSymbo**l():
  retourne le symbole qui permet d’identifier la pièce

Partie :

+ **demarrerPartie**():
  Initialise tous l'échiquier , place les pièces , et initialise les joueurs .

+ **coupValide**(Case case,Case case , Echiquier echecquier):
  vérifie si le coup est possible sur l'échiquier (pas d’obstacle sur le chemin)

+ **echec**(Case case,Case case,Joueur joueur):
  vérifie si le roi du joueur passé en paramètre est en échec

+ **pat**(Case case,Case case,Joueur joueur):
  vérifie si le roi du joueur passé en paramètre est en pat

+ **enregistrerCoup**(Case case,Case,Joueur joueur):
  permet d'enregistrer le coup précédent dans une list

+ **afficherCoup**():
  affiche les coup qui sont stockés dans la liste des coups

+ **ChoixPiece**(Echicquier echecquier,Joueur joueurCourant):
  permet de faire en sorte que l’utilisateur puisse choisir une pièce

+ **PartieFini**(Echequier echecquier):
  permet de vérifier que la partie est fini (si un des roi et en situation d’echec et mat ou de pat)

+ **aleterneJoueur**():
  permet d’alterner de joueur courant entre le joueur blanc et le joueur noir

Répartition de création des class :

Tour: Ouways
Fou: Ouways
Dame: Saif-Eddine
Roi: Saif-Eddine
Pion: Ouways et Saif-Eddine
Cavalier: Saif-Eddine
Piece: Ouways
Case:Saif-Edddine
Echequier: Saif-Eddine
Joueur: Ouways
Partie: Ouways et Saif-Eddine

Pour JeuEchec(le main) nous l'avons fait à deux ,nous avons aussi fait à deux les diagrammes UML et la structure du jeu .

Réussite et Problème :

Dans notre jeu, ce qui marche est la création de l'échiquier des pièces et de leur placement sur l'échiquier. Pour ce qui est de la logique de déplacement des pièces, chaque pièce sait où elle peut se déplacer (grâce au polymorphisme). Nous avons aussi une vue d'ensemble de la partie, cela nous permet de voir si un déplacement est réellement possible. L'affichage de l'échiquier avec les symboles modélisant les pièces fonctionne, il se met à jour après chaque coup légal. Et pour finir, la gestion des captures est réussie.
Pour ce qui ne marche pas, nous avons constaté que le jeu ne réussit pas à détecter une situation de fin de partie. Nous avons mis les pièces dans une situation d'échec et la partie continué .