

**Cabrol Morgane – Glangine geoffrey**

**Cégep de Matane**

**2015**

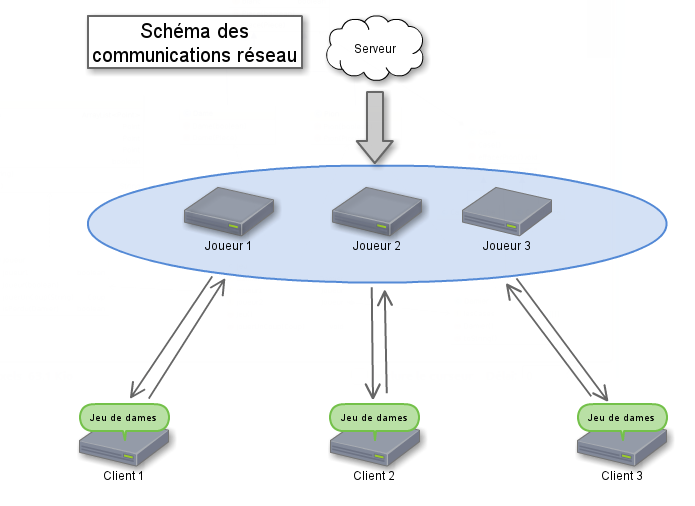
***Jeu de dames***

**Graphique**

**Projet**

# Architecture

Voici notre diagrame de vulgariation des communications réseau.



# Communication

Nous avons choisi d’implémenter le protocole TCP car sur un jeu de dames, c’est du tour par tour et les données envoyées sont importantes. En effet, si un coup et mal transmi au second client ou au serveur, ça risque de créer des conflits.

Sur le serveur, lors d’une connection, il est créé un Thread d’écoute pour écouter les entrées du client qui viens de se connecter. Puis le «Client » est placé dans un tableau et communiquera avec le prochain client à être placé dans ce tableau. Une partie est créée pour ces deux clients et ils peuvent jouer ensemble.

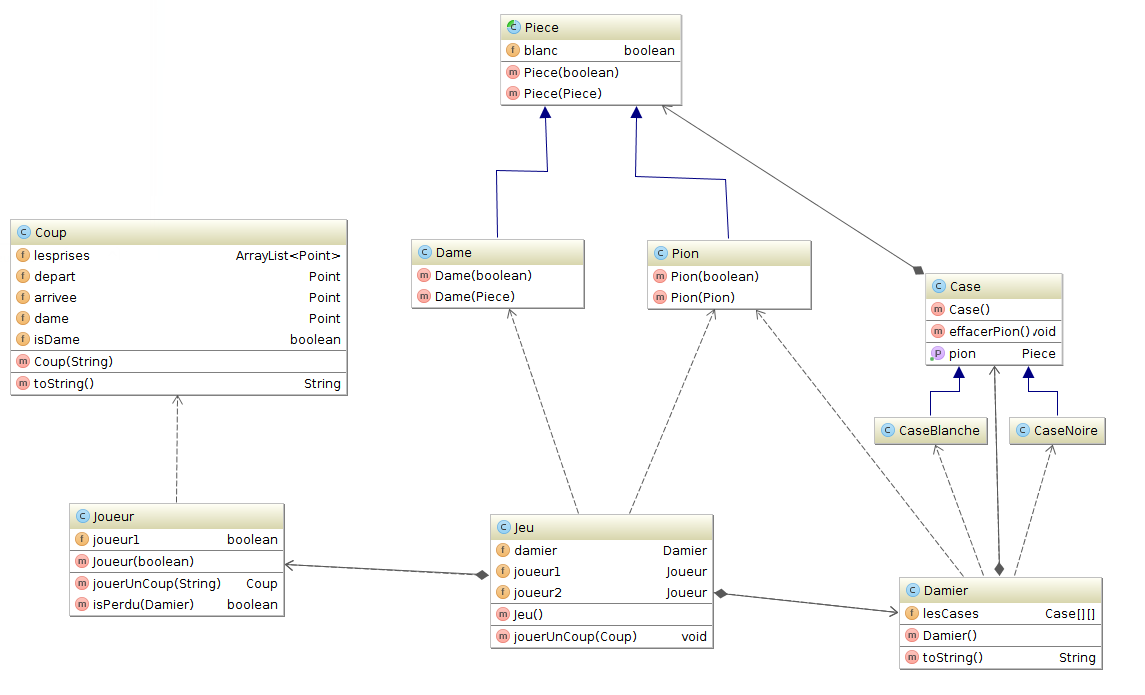
Nous aurions pu amméliorer encore ce protocole en faisant tourner chaque partie sur un Thread Et ainsi il n’y aurrais pas eut de tableau de client mais un tableau de partie et Nous n’aurions pas eût à vérifier quel client joue avec lequel et ça aurrait été beuacoup plus simple de fermer les communications et de supprimer la partie du tableau à la fin de la partie.

Mais malheureusement, nous n’avons pas eût le temps de faire cela.

# Conception

Diagramme UML

Ce diagramme représente seulement une petite partie du code, en effet il correspond juste à la logique du jeu de dames côté serveur.



# Code

Une des seules choses à connaitre pour pouvoir continuer le développement de notre projet est le format utilisé pour notre XML

<?xml version="1.0"?>

<coup>

    <PionJoue>

        <PositionDepart>

            <ligne>4</ligne>

            <colone>1</colone>

        </PositionDepart>

        <PositionArrive>

            <ligne>4</ligne>

            <colone>9</colone>

        </PositionArrive>

    </PionJoue>

    <Prises>

        <Prise>

            <ligne>5</ligne>

            <colone>2</colone>

        </Prise>

        <Prise>

            <ligne>5</ligne>

            <colone>4</colone>

        </Prise>

        <Prise>

            <ligne>5</ligne>

            <colone>6</colone>

        </Prise>

        <Prise>

            <ligne>5</ligne>

            <colone>8</colone>

        </Prise>

    </Prises>

</coup>

# SOLUTION

Tout au long de notre projet, nous avons rencontré plusieurs bugs ou problèmmes, en voici quelques uns.

## Compromis ou problème 1

Un problème d’affichage qui arrive de temps en temps. En effet le damier ne s’affiche pas tout le temps au lancement du jeu. Jusqu’à présent, ça ne nous dérangeait pas car ça ne nous empêchait pas de travailler et de jouer avec notre jeu. Mais maintenant que nous avons presque terminé il faut rendre le produit opérationnel à 100% sans bogues (même mineurs). De plus le même problème est apparu lors du passage de l’écran de jeu à l’écran de fin de partie. Nous avons donc recherché d’où venait ce bug et nous en avons déduit qu’il fallait faire un appel à la méthode pack() de notre fenêtre après avoir changé son panel !

En effet, j’ai appris que la méthode pack() servait à redimensionner un objet de façon à ce que ses préférences de taille soit conformes au préférences de taille des composants qui composent cet objet.  
Voilà quelques heures de perdues qui se résolvent en une seule petite ligne de code ! :)

Ma fenêtre.pack()

## Compromis ou problème 2

Nous avons eût un petit problème de MVC en effet, vers le milieu du projet nous nous sommes rendus compte que nos classes de Contrôle contenaient du code de vue. Il y en avait trop et le problème était trop important pour être refactorisé. Nous avons donc décidé de refaire des classes de Contôle pour le côté serveur. Ce qui est dommage car nous aurions pu utiliser notre code de contrôle pour les deux côtés à la fois. De plus, nous avons perdu beaucoup de temps à essayer d’amméliorer cela.

## Compromis ou problème 3

Nous avons rencontré un problème avec le mouseListener de ma case car je l’ajoutait dans ma classe « panneau » qui est un dossier graphique et je ne pouvais donc pas utiliser les méthodes de ma classe « Jeu » sauf en créant un objet Jeu dans ma classe « Panneau » ce qui n’aurait pas été logique…

Solution : Finalement j’ai créé une classe « CaseEcouteur » qui implémente la classe « MouseListener » et qui récupère une Case de Jeu , Un damier de Jeu et aussi L’image de la case. Finalement cette classe permet de faire le lien entre ma partie graphique et ma partie Jeu lors de l’évènement clique sans qu’il n’y ait d’incohérence.

# Ressources

## Liens vers les outils

### Librairie JDOM2 pour le XML

<http://www.jdom.org/>