TRUONG-Timmy
G54900

# Rapport Projet ATLIR5 : Skyjo (Aout/Septembre)

## Introduction

Pour la première session, le projet consistait à choisir un sujet de projet et de l'implémenter en y incorporant un client serveur et une base de données. Ainsi, pour la seconde session, le projet consiste en une correction du projet choisi en première session.

J'ai voulu reprendre la dernière remise effectuée par notre groupe. Cependant, vu « l'état » dans lequel notre projet était, il m'était incompréhensible. Ainsi, j'ai décidé de refaire un tout nouveau projet mais en gardant les idées principales établies dans le projet en première session.

## Description générale du projet

Pour le projet, le jeu choisi a été « Skyjo », un jeu de cartes (voir section « Règles du jeu »). Ainsi, le projet contient une architecture Client-Serveur multi-clients et multi-parties, et aussi une partie « SGBD ».

De manière générale, plusieurs clients peuvent se connecter et affronter un autre client en jouant une partie de « Skyjo ». Cependant, ceux-ci peuvent également voir les résultats des anciennes parties et aussi les anciens joueurs des dernières parties.

Enfin, le projet global se divise en 3 projets différents : un pour le serveur, un pour le client, un pour les messages envoyés entre le client et le serveur. Le projet contenant les messages a été inclus dans les deux autres projets afin que le client et le serveur puissent justement comprendre les types de messages envoyés. Dans le serveur, une partie « console » a également été ajoutée afin de pouvoir voir les données dans la base de données, une manière plus rapide de pouvoir voir les données ajoutées.

## Règles du jeu

Pour les règles complètes du jeu : *https://www.regledujeu.fr/regle-du-skyjo/*

Remarque : L'année passée, nous avons effectué le projet sur le jeu de cartes « Skyjo » mais seule une partie du jeu a été implémentée, et non pas l'entièreté du jeu officiel car cela aurait été trop long.

TRUONG-Timmy
G54900

# Bibliothèques/Outils employé(e)s

Les bibliothèques utilisées pour le projet sont :

- Les bibliothèques « java fx » afin de pouvoir créer un affichage pour le client.
- Le « Scene Builder » a également été utilisé pour l'affichage.
- Pour le « SGBD », le fichier jar, afin de pouvoir faire fonctionner le « SGBD » a été intégré au projet.
- Utilisation de « SQLite » pour créer les tables du projet.

# Mode d'emploi

## *Projet en général :*

Pour lancer mon projet « Skyjo » (projet Ant), cela nécessite quelques étapes :

1) Il faut lancer le serveur en faisant un « run » du projet serveur pour pouvoir lancer le serveur.
2) Il faut lancer un client en faisant un « run » du projet client pour pouvoir lancer un client. Ainsi, lorsque le client sera connecté au serveur. Le client aura le choix : jouer une partie, quitter le jeu (donc le serveur), ou visualiser les anciennes parties ou les anciens joueurs.
3) – Si le client choisit de jouer contre un autre client, celui-ci verra une fenêtre d'attente, cela signifie que celui-ci doit attendre un autre joueur. Cependant, si le nombre de joueurs est suffisant, la fenêtre d'attente s'affichera, mais le jeu en lui-même s'affichera directement après.
   – Si le client choisit de visualiser les anciennes parties et anciens joueurs, alors 2 fenêtres s'afficheront avec les anciennes parties et les anciens joueurs.
   – Si le client choisit de quitter, il va simplement quitter le serveur
4) Ainsi, si deux clients sont dans une partie (deux fenêtres différentes), ceux-ci pourront y jouer. A la fin de la partie, les données des joueurs et de la partie sont sauvegardées dans le « SGBD ».

## *Jouer au jeu :*

En ce qui concerne le jeu en lui-même, les clients ont un affichage avec leur jeu et celui de leur adversaire, la pioche et la défausse. Ceux-ci ont également deux boutons dans lesquels se trouvent un rappel des règles et un rappel/explication de comment interagir avec l'affichage pour pouvoir jouer au jeu.

Pour l'interaction du jeu et donc pouvoir y jouer, le client devra faire un clic droit sur une des cartes dans sa main.

Ainsi, 3 options s'offrent à lui :

- Le joueur peut échanger la carte sur laquelle il a fait un clic droit contre celle de la défausse.
- Si le joueur a pioché avant, celui-ci peut échanger la carte sur laquelle il a fait un clic droit avec la carte piochée.
- Si le joueur a pioché avant, celui-ci peut décider de révéler la carte qu'il a cliquée (celle-ci doit être cachée) et mettre la carte piochée sur la défausse.

## Description des classes

### *Partie Client :*

- AbstractClientSkyjo : classe reprise dans le cours, contient les méthodes que doit avoir un client pour se connecter à un serveur. Cette classe sera héritée par la classe ClientSkyjo.
- ClientSkyjo : classe représentant un client voulant se connecter au serveur Skyjo et qui va gérer les messages envoyés par le serveur.
- ViewClientSkyjo : classe créant un client et qui créée l'affichage du client et le met à jour.
- ViewGameSkyjo : classe créant l'affichage du jeu tout entier.
- ViewHiddenCardSkyjo : classe créant une carte cachée du jeu.
- ViewIntroductionMenuSkyjo : classe créant le menu d'introduction du jeu quand un client se connecte au serveur.
- ViewMessageSkyjo : classe créant les messages du jeu adressés aux joueurs et met à jour les messages en fonction de la situation.
- ViewPilesSkyjo : classe créant la défausse et la pioche et les interactions avec ceux-ci.
- ViewPlayerSkyjo : classe créant la vue d'un joueur : ses points et ses cartes.
- ViewTextFieldNameController : classe qui représente les interactions avec la fenêtre de nom.
- ViewValueCardSkyjo : classe créant une carte Skyjo visible, avec son nombre et sa couleur en fonction du nombre.
- Observable : classe pour le design pattern Oberver. - Observer : classe pour le design pattern Oberver.

### *Partie Message :*

- MessageData : classe représentant un message de la part d'un client afin de pouvoir voir les données dans le « SGBD ».
- MessageDropHitCard : classe représentant un message de la part d'un client disant qu'il souhaite révéler la carte choisie et mettre la carte piochée sur la défausse.
- MessageEndGame : classe représentant un message de la part d'un client disant que le jeu est fini.
- MessageExchangeDiscardCard : classe représentant un message de la part d'un client disant qu'il veut échanger sa carte avec celle de la défausse.

- MessageExit : classe représentant le message de la part d'un client qui souhaite quitter le serveur.
- MessageHitCard : classe représentant le message de la part d'un client qui souhaite piocher une carte.
- MessageKeepHitCard : classe représentant le message de la part d'un client qui souhaite échanger une carte avec la carte piochée.
- MessageReadyToPlay : classe représentant le message de la part d'un client disant s'il veut jouer une partie.
- MessageDataDb : classe représentant le message de la part du serveur contenant les données des tables Player et Game.
- MessageInGame : classe représentant le message de la part du serveur disant qu'un joueur est en jeu contre un autre.
- MessageInWait : classe représentant le message de la part du serveur disant qu'un joueur est en attente.
- MessageInformationsGame : classe représentant le message de la part du serveur avec les informations actuelles du jeu.
- MessageNames : classe représentant le message de la part du serveur avec les noms des joueurs.

### *Partie Serveur :*

- AbstractServer : classe reprise dans le cours, contenant les méthodes à avoir pour qu'un serveur interagisse avec des clients.
- ConnectionToClient : classe reprise dans le cours, représentant une connexion avec le serveur.
- ServerSkyjo : classe représentant le serveur de jeu Skyjo, acceptant les différents clients voulant se connecter au serveur.
- Les classes dans le package g54900.Skyjo.Model créent le jeu et fait fonctionner le jeu.
- Les classes dans les packages g54900.jdbc servent à l'interaction avec la base de données (tables Game et Player).

## Modifications générales apportées

Pour la seconde session, beaucoup de modifications ont été apportées.

### *Projet en général :*

Le projet fonctionne plus qu'en première session et est plus structuré et divisé en plusieurs packages. Il contient tout ce qui a été demandé.

### *Partie Client-Serveur :*

TRUONG-Timmy
G54900

Le client-serveur reste basé sur le même principe qu'en première session mais certaines méthodes ont été modifiées et le modèle du jeu a été repris de mon ancien projet et pas de celui d'un de mes coéquipiers dans mon ancien groupe. *Partie SGBD :*

Pour la partie « SGBD », deux tables sont créées cette fois-ci (Game et Player). Le « SGBD » s'est inspiré de la structure vue en cours. De plus, lorsque deux joueurs ont fini de jouer, les données de la partie et des joueurs sont sauvegardées directement à la fin de la partie.

*Partie Affichage :*

Des modifications ont été apportées au niveau de l'affichage. Dans le menu, les clients peuvent jouer mais aussi voir les données des anciennes parties et des anciens joueurs. Lorsque ceux-ci veulent jouer, ceux-ci peuvent maintenant mettre leurs noms dans le jeu. Une fenêtre d'attente s'affiche également maintenant quand le nombre de joueurs est insuffisant.

L'affichage du jeu reste pratiquement le même, deux boutons ont été ajoutés afin que les joueurs puissent avoir un rappel des règles et un rappel de comment interagir avec l'affichage pour pouvoir jouer au jeu.

Enfin, l'utilisation du « SceneBuilder » ne se fait pas sur tout le projet mais seulement sur la fenêtre d'attente et sur la fenêtre où les joueurs peuvent mettre leurs noms. Le reste a été codée de manière « brute ».

*Partie rapport :*

Le rapport est plus structuré et détaillé.

## Bibliographies/ Références

Aucune partie ou classe en dehors du cours a été utilisée.

Pour la partie Client-Serveur, la structure du projet s'est totalement inspirée de la structure vue en cours. Certaines classes ont été reprises du cours, notamment les classes « AbstractServer », « AbstractClient », « ConnectionToClient », et ont été modifiées.

Pour la partie « SGBD », le projet a repris la structure vue en cours. Certaines classes (notamment les classes Demo, Controller, View dans le cours) ont été reprises et modifiées afin de pouvoir les incorporer à mon projet.

## Conclusion/Résultat

En conclusion, selon moi, le projet fonctionne à 95% (ceci est bien évidemment mon estimation personnelle) car lorsque je lance des clients, une partie est bien créée avec les bons joueurs correspondants, mais parfois l'affichage du jeu pour les deux clients rencontre un problème. Cela n'arrive pas souvent mais je n'ai pas réussi à comprendre la cause. Il faut relancer le serveur et les clients si le problème d'affichage apparaît.