JCoinche
| Java Programming

Raphaël Beltrand
Baptiste Tiers

About JCoinche :

The “*Jeu de Coinche*” is a french card game derived from the “*Belote*” introducing a new auction named “*Coinche*”.

This project is developed in **Java**, on a Client/Server architecture.

In this documentation, you will find all the necessary information you need to build it from source files, how it works, as well as detailed information about the different parts of it.

Summary

1. **Requirements**
2. **Getting ready**
3. **UML Diagrams**
   - ➢ Server
   - ➢ Client
4. **Protocol**
5. **Credits**

Requirements

The project uses *Maven* to build.

It is a Java project running on *JRE 9*. It must be build with *JDK 9*. Make sure your environment variables are set accordingly.

Getting Ready

To install the game, use maven at the root of the repository :

```
#> mvn package
```

The server and client jar files are located in the **target/** directory.

The server can be launched via command line to specify a PORT argument, if you want to bypass the default 4242 port.

```
#> jcoinche-server [PORT]
```

The client can also be launched via command line to specify an IP and a Port argument, if you want to bypass the default 127.0.0.1 IP and 4242 port.

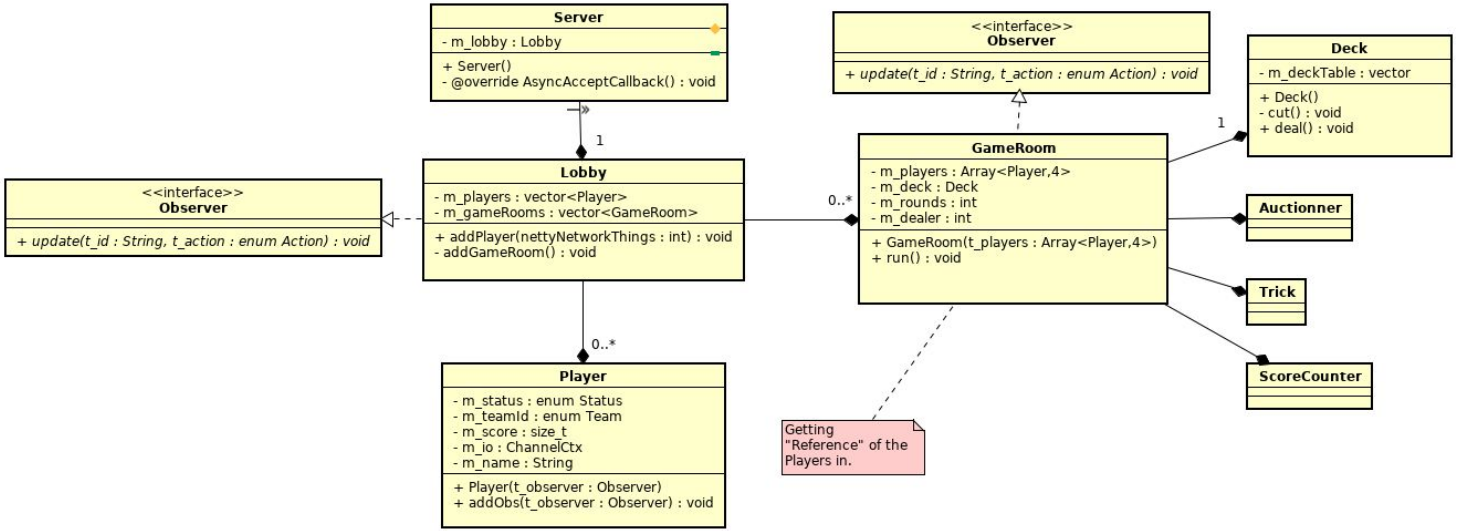
```
#> jcoinche-client [IP] [PORT]
```

UML DIAGRAMS

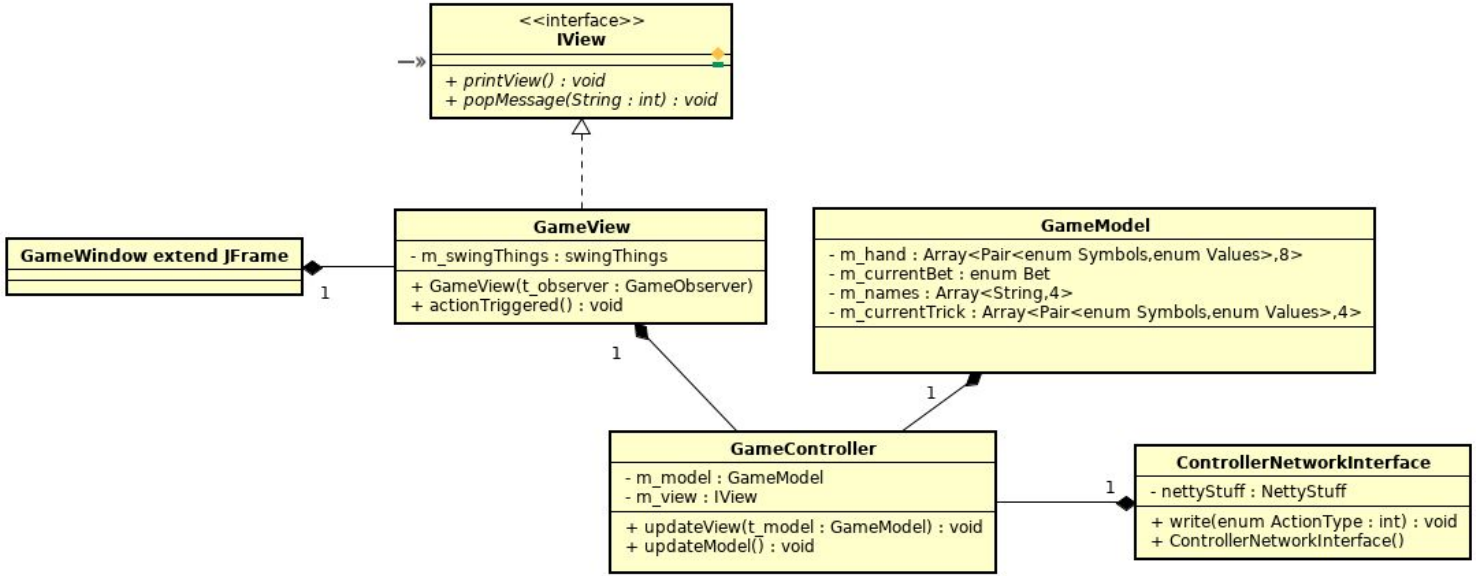
1. Server

2. Client

Server


Server
- m_lobby : Lobby
+ Server()
- @override AsyncAcceptCallback() : void
1
<<interface>>
Observer
+ update(t_id : String, t_action : enum Action) : void
Lobby
- m_players : vector<Player>
- m_gameRooms : vector<GameRoom>
+ addPlayer(nettyNetworkThings : int) : void
- addGameRoom() : void
0..*
Player
- m_status : enum Status
- m_teamId : enum Team
- m_score : size_t
- m_io : ChannelCtx
- m_name : String
+ Player(t_observer : Observer)
+ addObs(t_observer : Observer) : void
<<interface>>
Observer
+ update(t_id : String, t_action : enum Action) : void
GameRoom
- m_players : Array<Player,4>
- m_deck : Deck
- m_rounds : int
- m_dealer : int
+ GameRoom(t_players : Array<Player,4>)
+ run() : void
0..*
1
Deck
- m_deckTable : vector
+ Deck()
- cut() : void
+ deal() : void
Auctionner
Trick
ScoreCounter
Getting
"Reference" of the
Players in.

Client


<<interface>>
IView
+ printView() : void
+ popMessage(String : int) : void
GameView
- m_swingThings : swingThings
+ GameView(t_observer : GameObserver)
+ actionTriggered() : void
GameWindow extend JFrame
GameModel
- m_hand : Array<Pair<enum Symbols,enum Values>,8>
- m_currentBet : enum Bet
- m_names : Array<String,4>
- m_currentTrick : Array<Pair<enum Symbols,enum Values>,4>
GameController
- m_model : GameModel
- m_view : IView
+ updateView(t_model : GameModel) : void
+ updateModel() : void
ControllerNetworkInterface
- nettyStuff : NettyStuff
+ write(enum ActionType : int) : void
+ ControllerNetworkInterface()
1
1
1
1
1

Protocol

The protocol is provided by the Google Protobuf protocol generator. Following is more details about how we organized the protocol for JCoinche.

- ***BaseCommand*** : Used for packet interfacing
- ***Auction*** : Sends a Packet containing the auction put by a player
- ***Move*** : Sends a Packet containing the move made by a player
- ***player_state*** : Contains all the player information to display
- ***bet_state*** : Contains the current game auction information
- ***static_game_state*** : Contains game information to display
- ***request*** : Used to pull information from the Client
- ***switch_mode*** : Informs the Client of a game screen change
- ***ServResponse*** : Sends a Server response to the Client
- ***Disconnect*** : Informs the parties of a Client disconnection

Credits

This project is brought to you by **La Pirogue Du Fun ™**