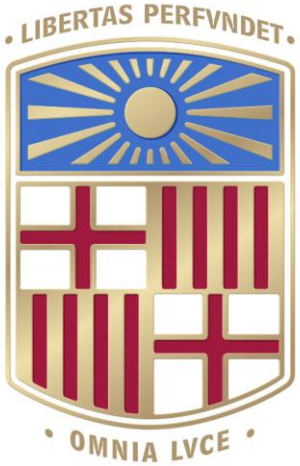

UNIVERSITAT DE BARCELONA

ASSIGNATURA: *SOFTWARE DISTRIBUÏT*

PRÀCTICA 1

ALUMNOS:

ALEJANDRO ALFONSO HERNÁNDEZ

VÍCTOR MAESTRO LARA

Data d'Entrega: 02/04/2019

INDEX

Introducció

L'objectiu d'aquesta primera pràctica de Software Distribuït és realitzar una aplicació distribuïda utilitzant una comunicació **client-servidor** en Java. L'aplicació definida en classe és el joc del **BlackJack** simplificat, el qual consta d'un jugador o client i un servidor o crupier.

El BlackJack és un joc de cartes en el que els jugadors intentaran assolir una puntuació el més proper possible a 21 sense passar-se. El joc es jugat amb una baralla típica de poker i els valors de cada carta corresponen al seu valor excepte en el cas del AS que pren un valor de 1 o 11 segons les altres cartes que es tinguin a la mà.

La baralla conté un total de 52 cartes: A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3 i 2. Per a cada pal clubs (♣), diamonds (♦), hearts (♥) and spades (♠).

S'ha definit un protocol a classe que serà el que utilitzarem en la nostra implementació per a la comunicació entre client i servidor.

El servidor haurà de estar implementat amb la capacitat d'acceptar múltiples clients al mateix temps. Per això haurà de ser multi-thread. Un thread per a cada client nou connectat.

Diagrama d'estats

Primerament hem definit un conjunt d'estats en el que es possible estar durant una partida. Per això ens hem basat en les accions que un jugador pot prendre seguint el flux de una partida normal.

Tant el **client com el servidor** tenen comandes de comunicació que poden o no ser compartides. Per exemple trobem que el **client** pot fer **STRT** i no **INIT** però el servidor si que pot fer **INIT** i no **STRT**. Al mateix temps també comparteixen comandes com el **SHOW**.

Podem dir que el diagrama d'estats d'un joc és la combinació d'estats possibles als que es pot arribar segons les decisions presses pel client en un estat concret de la partida.

A continuació s'adjunta una imatge del diagrama d'estats del nostre joc. Aquest inclou diferents possibilitats segons les accions presses pel client i la situació de la partida. Aquestes diferents opcions estan diferenciades per colors en les fletxes entre els diferents estats, i indiquen que *s'ha passat del flux original d'una partida a un altre diferent*. La imatge també es pot trobar adjunta en la carpeta de diagrames amb nom **"diagrama de estados v2.0.png".**

Diagrama de classes

Primerament explicarem com està organitzada la nostra aplicació de client i servidor.

Tant Client com Servidor tenen classes comunes com la classe ComUtils.java que inclou tots els mètodes que permeten tant llegir o escriure amb els sockets de comunicació.

Així mateix, trobem classes principals que es comparteixen en el model:

Classes	**Informació**
GamePlayer	Super classe de Player i Dealer que conté la informació de les cartes d'un jugador.
Player	El jugador que juga com a client.
Dealer	El crupier que juga com a servidor.
Card	La classe carta per identificar cadascuna de les diferents cartes de la baralla
Suit	El pal d'una carta.
Rank	El valor d'una carta.
Winner	El guanyador d'una partida.
Command	Les possibles comandes que existeixen en una partida tant pel servidor com el client.

Com ja sabem el client ha de establir la connexió amb el servidor en una adreça IP i port especificats pel servidor. En el cas del client la connexió s'estableix a la classe **Client.java** i la lògica del joc la porta la classe **GameClientThread.java** hem decidit que quan s'estableixi una nova connexió el client creí un fil nou per a la partida, que tot hi que no es necessari hem cregut convenient fer-ho en cas que un futur es volgués permetre que el mateix client pogués jugar en múltiples partides al mateix temps.

D'altra banda per part del Servidor, el **Server.java** accepta les sol·licituds dels clients i crea un fil nou per a cadascun d'ells per tal d'atendre'ls. La lògica es pot trobar en la classe **GameLogic.java** que es la que porta el seguiment de l'estat del joc.

Així mateix, hem decidit implementar el nostre Client amb una interfície de menú per facilitar als jugadors la pressa de decisions i al mateix temps eliminar errors que podrien sorgir per part del client per enviar comandes no possibles en un estat concret de la partida, tot hi que fem la comprovació en el servidor que la comanda rebuda sigui possible per l'estat de la partida actual.

Els diagrames de classe es poden trobar a la carpeta Diagrames amb els noms: **"Diagrama de clases Client.png"** i **"Diagrama de clases Server.png"** respectivament.

Hem afegit un conjunt de classes per comunicar errors, tant pel client com pel servidor:

Classes	**Informació**
MaxCashExceedException	Exception thrown when Client exceeds max cash
NoExistRankException	Exception thrown when a Card Rank does not exist
NoExistSuitException	Exception thrown when a Card Suit does not exist
NoClientCommandException	Error thrown when a client sends a Command that is not known by the server
NotEnoughCashException	Error thrown when player's initial bet is lower than the minium
NotEnoughCoinsException	Create a new Error for not having enough coins to bet or start a new game
PlayerCheatsException	Error thrown when player tried to cheat on the server cause the cards that he send us are different than the server side has
ServerSideErrorException	Exception thrown when a Server loses the track of the game state
UnkownActionException	Exception thrown when the action taken is not known
WrongActionException	Error thrown when the client sends to the server an action that can not be taken for the current game state
NoExistWinnerException	Exception thrown when a Winner does not exist
UnexpectedCommandExcpetion	Exception thrown when a Command from the server was not expected
UnknownActionException	Exception thrown when the action taken is not known in client side

Resum sessió de Test

- Errors detectats en el nostre client, cap.
- Errors detectats en el nostre servidor, cap.

D'altra banda el conjunt d'excepcions que hem implementat ens ha facilitat la feina per poder informar a la resta de grups sobre la presència de errors.

En el nostre servidor cap grup a tingut cap problema en la comunicació a excepció d'aquells que presentaven algun problema en la seva implementació del client.

Per part del nostre client hem pogut detectar errors amb altres grups per problemes en la implementació del protocol en l'altre grup.

- El grup F3, presentava problemes en la lectura i enviament dels chars de les cartes (el suit i el rank).
- El grup F5, presentava problemes en la implementació del seu protocol ja que utilitzava String variables per a la comunicació.
- El grup F9, aquest grup no va testejar amb cap altre grup ni el servidor ni el client per tant no hem pogut comprovar la comunicació.

Conclusions

Hem implementat tant el client com el servidor seguint les pautes indicades en el protocol. Amb controls d'error i amb excepcions per informar tant al client com al servidor de l'existència dels mateixos.

El client ha sigut implementat amb els diferents modes de joc i el servidor amb multi-thread per poder acceptar múltiples clients al mateix temps.

D'altra banda no hem pogut implementar el selector, tot hi que el vam començar vam tenir problemes amb el buffer i vam decidir cancel·lar el seu desenvolupament.