Pràctica 1 - Dominó

Software Distribuït

Xavi Moreno Liceras
Pablo Martínez Martínez

Introducció

En aquest document informarem sobre la primera pràctica realitzada en Software Distribuït. Es donarà detall sobre tot allò que s'ha implementat, excepte el protocol. Aquest, el protocol, està especificat en el següent enllaç https://gist.github.com/eloipuertas/9392391. L'informe està dividit en tres parts:

- Client
- Servidor
- Llibreria

Totes aquelles classes que es vegin de color blau en el diagrama de classes, es que són classes implementades en la llibreria.

Client

Vista

Per tal de realitzar la part del client en aquesta pràctica, he generat un mini framework, en el qual treballar, i així separar les vistes que es podien presentar. D'aquesta manera s'aconsegueix modularitzar tota la vista i així no encapsular-ho tot en una única classe.

El framework que hem montat, és semblant a la idea d'Android, hi ha un **ViewController**, que conté un bucle el qual va mostrant Views, mentre hi hagin Views disponibles. Tota **View** (classe abstracte) té dos mètodes a implementar:

- getTitle(): La View heretada ha de retornar en aquest mètode un títol de la View.
- run(): En aquest mètode s'ha d'incloure tota la vista d'aquella View, és a dir, tot allò que es volgui presentar per pantalla. A més aquest mètode s'ha de retornar un objecte de tipus Class, que serà la següent View que es mostrarà, si aquesta Class és nul·la, llavors el ViewController, s'encarregarà de cridar a la View que hi havia en la pila.

El **Bundle** es una instància que té el ViewController, per tal de compartir informació entre Views. D'aquesta manera una View pot passar informació a l'altre. En la pràctica no s'ha arribat a utilitzar, però hem vist que podria tenir alguna utilitat.

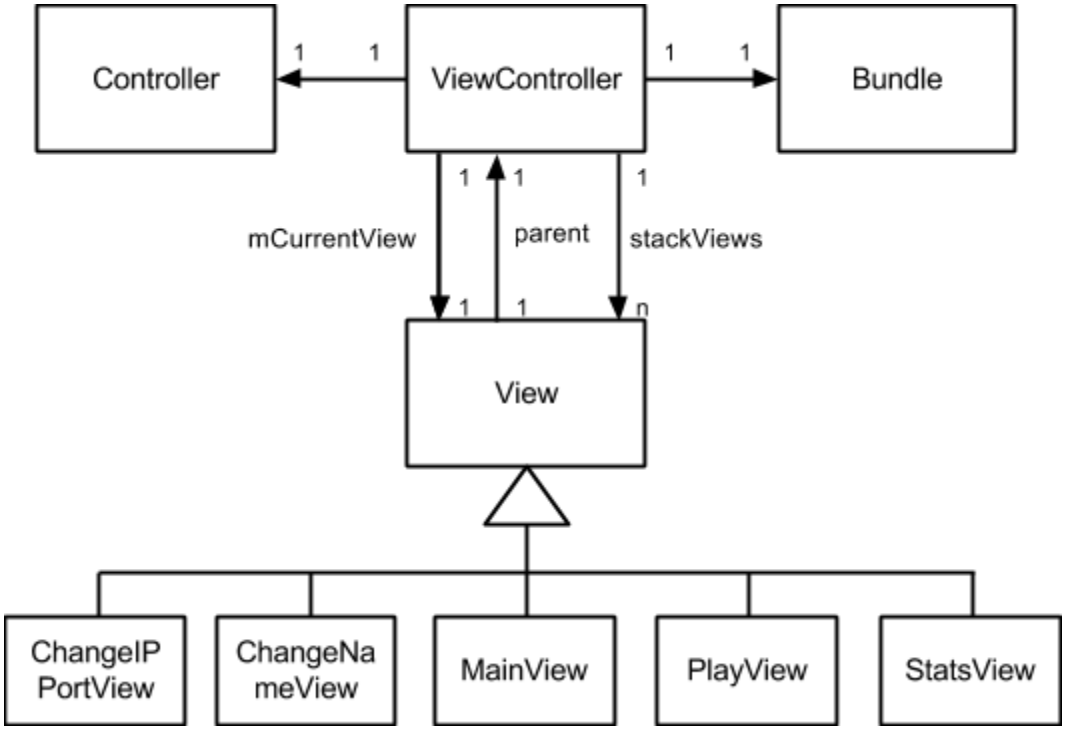


Figura 1

Hem creat 5 Views, que són les següents:

- ChangeIPPortView: View que permet canviar la ip i port del servidor al qual es vol conectar.
- ChangeNameView: View per poder canviar el nom del player del client.
- MainView: View principal, aquest és el primer View que s'executa. Aquest conté un menú que permet interactuar amb l'usuari per tal de jugar, canviar la ip i el port, canviar el nom d'usuari, o veure taules d'estadístiques.
- PlayView: Aquesta View permet jugar al joc del Dominó a través de línies de comandes. Aquesta s'encarrega de fer crides a GameController, la qual implementa el protocol a seguir.
- StatsView: View que permet visualitxar dades com una taula amb les partides jugades, o veure IPs que han donat problemes de connexió.

Controlador-Model

El client conté un controlador (**Controller**) que manega dades de tota la aplicació. Aquest controla:

- Dades d'estadístiques (**Stats**): Conté informació sobre les partides jugades.
- Controlador del joc
- Nom d'usuari
- IP i port del servidor que es vol conectar

Pel que fa al controlador del joc, l'hem encapsulat en **'GameController'**, la qual hereta de AbstractProtocol, una classe implementada en la nostra llibreria. Aquest controlador conté una serie de funcions que es comuniquen amb el servidor. A més aquesta classe és encarregada de portar la lògica del joc per part del client, és a dir, inserir i borrar les fitxes de les llistes corresponents. Aquest control el fa a través de la clase **DominoGame**, la qual conté el taulell i la mà del client.

Aquest un diagrama de classes de la part del controlador del client:

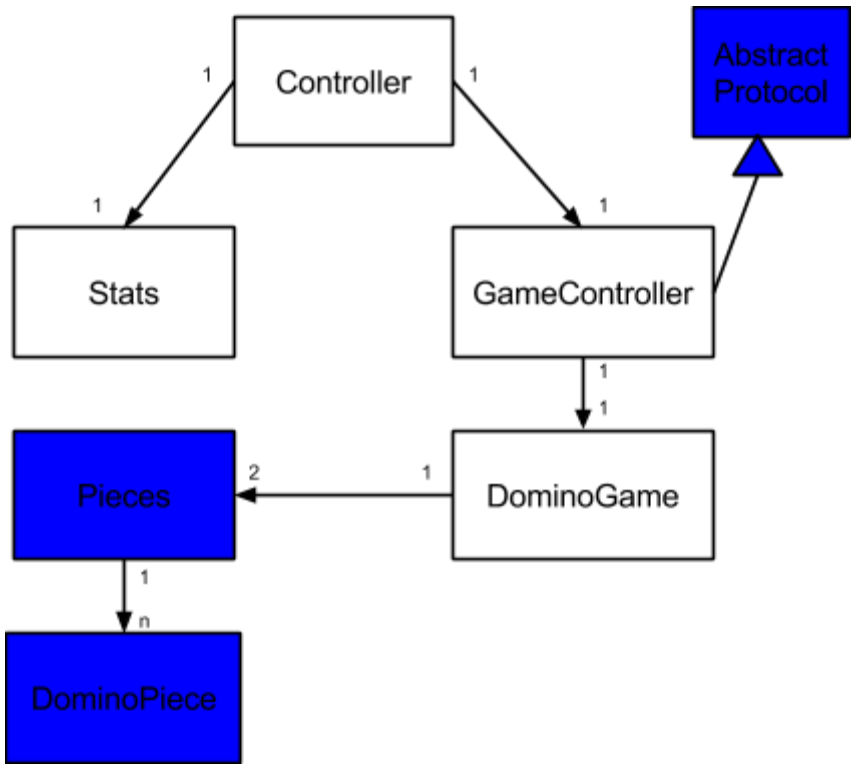


Figura 2

Llibreria

En la llibreria hem inclòs codi, el qual es reutilitzat tant per la part del servidor com per a la part del client. En la lliberia tenim dos parts, una el model i l'altre el protocol.

Model

Pel que fa al model, tant el client com el servidor, hem reutilitzat un parell de classes essencials tant per l'un com per l'altre, que són: **Pieces** i **DominoPiece**:

- DominoPiece: Classe que conté informació sobre una fitxa i mètodes per saber si es doble o per invertir la fitxa.
- Pieces: Aquesta classe reuneix un conjunt de fitxes de dominó i permet inserir de forma ordenada o no a la llista. Conté mètodes útils com per exemple una per saber quina puntuació suma la mà acomulada, entre altres.

Connection

El paquet connection de la llibreria conté la classe **AbstractProtocol** que es el pare de la comunicació entre client i servidor. Aquest métode conté una instancia de **ComUtils**.
El métode principal d'aquesta classe es readFrame(). Aquest métode discrimina el frame en funció del seu header i fa la crida amb parametres a una funció abstracta que client i servidor hauran d'implementar.
Seguint aquest model de disseny la funció *helloFrameRequest()* será cridada explicitament per el client, i li arribarà a traves de *readFrame*() al servidor. Així el servidor farà una crida explicita de la funció *helloFrameResponse()* i al client li arrivarà la crida a traves de *readFrame()*. I així successivament.

Totes les funcionalitats estan degudament comentades tant al codi com al JavaDoc.

Servidor

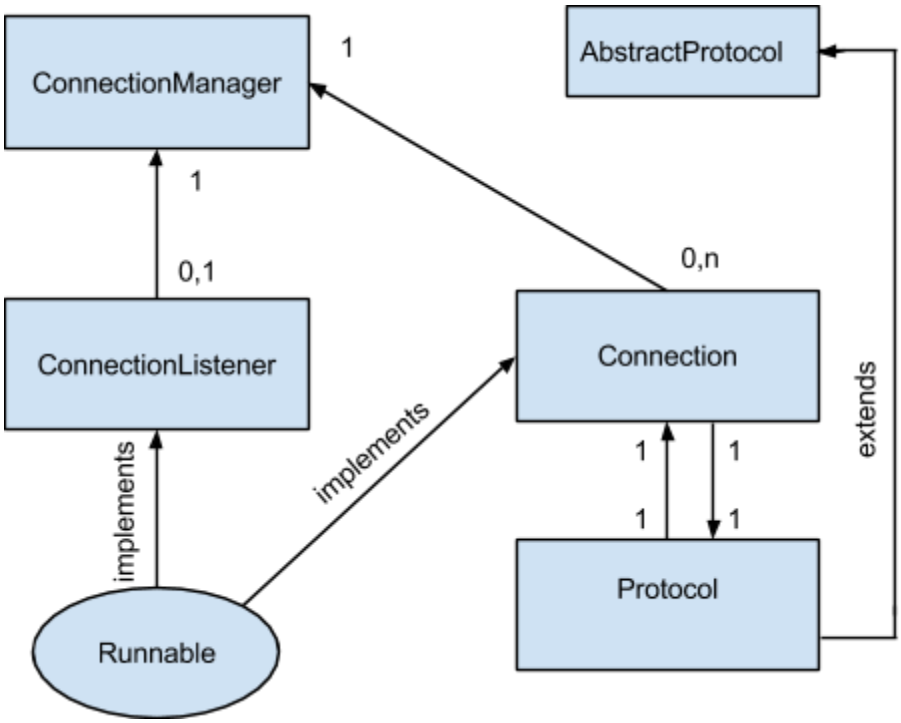
El servidor està organitzat amb el paradigma MVC, per tant, tenim tres paquets:

- **Model**
- **View**
- **Controller**

Model

Es divideix en dos paquets:

- Game: Conté només una classe homònima que conté totes les funcions corresponents a la lògica del joc.
- Connections: En aquest paquet tenim quatre classes. ConnectionManager, ConnectionListener, Connection i Protocol.
    - **ConnectionManager**: S'encarrega de gestionar les connexions. Conte un HashMap amb totes elles. A més també conté entre 0 i 1 ConnectionListener. Aquesta classe tracta els errors relacionats amb la inicialització i finalització de les conexions.
    - **ConnectionListener**: Es una classe que implementa la interficie Runnable, per tant es tracta d'un Thread que escolta les peticions de conexió. A més notifica a ConnectionManager quan hi ha una nova conexió perque aquesta ho gestioni.
    - **Connection**: Thread propi de cada client.
    - **Protocol**: Classe que hereta de AbstractProtocol, implementa totes les funcions abstractes.



View

Dins de la view tenim la classe **ServerMainWindow** i **Log**. El Log es un objecte que es va repartint per totes les classes del servidor, per tant qualsevol classe pot escriure en ell. Dins de ServerMainWindow, a part del mètode main, també es carrega una GUI que permet al usuari monitoritzar l'activitat al servidor.

Controller

El package controller conté la classe **ServerController**. Aquest conté mètodes d'activació i desactivació de logs. Així com encendre i apagar el ConnectionListener i desconectar clients.

Test

El dia 13 de Març vam realitzar un testeig amb tota la classe i així comprobar si el nostre client-servidor, funcionava amb la resta de la classe. Aquest va ser el resultat obtingut.

SI -> Funciona
NO -> No funciona

El nostre client amb altres servidors:

1 - NO Error al enviar fitxa vàlida
2 - SI
3 - NO Error al enviar fitxa vàlida
4 - SI
5 - SI
6 - NO fitxas duplicadas
7 - NO Error al enviar una fitxa vàlida
8 - NO lógica, envía movimiento inválido cuando no lo es
9 - NO Error al fi de partida, m'he quedat sense mà i no m'ha dit partida acabada
10 - SI
11 - NO Error al enviar una fitxa vàlida
12 - NO

Els demés clients al nostre servidor:

1 - SI
2 - SI
3 - SI
4 - SI
5 - SI
6 - SI
7 - SI
8 - SI
9 - SI
10 - SI
11 -
12 - SI