Universidad de Buenos Aires
Facultad de Ingeniería

75.52 Taller de Programación II
1^{er} Cuatrimestre 2011

Trabajo práctico Go

Fecha de entrega: 3 de Junio

Tutora: Patricia Calvo

Integrantes:

Apellido,Nombre	Padrón Nro.	E-mail
Bukaczewski, Veronica	86.954	vero13@gmail.com
De Antoni, Matías	88.506	mdeantoni87@gmail.com
Garbarini, Lucía	88.300	lu.teddy@gmail.com
Pandolfo, Lucas	88.581	lucashpandolfo@gmail.com

Índice

1. Objetivo

Desarrollar un sistema que permita jugar partidas de **Go**, en un tablero reducido y considerando el juego finalizado **a la primera muerte** (capture Go).

2. Requerimientos funcionales

El sistema debe permitir a dos jugadores humanos en la misma computadora jugar entre si. También debe permitir como posibles participantes del juego a alguna aplicación externa (como ser **gnugo**). Adicionalmente se deben incluír estrategias de juego para que una sola persona pueda desarrollar una partida contra el sistema.

3. Manual de usuario

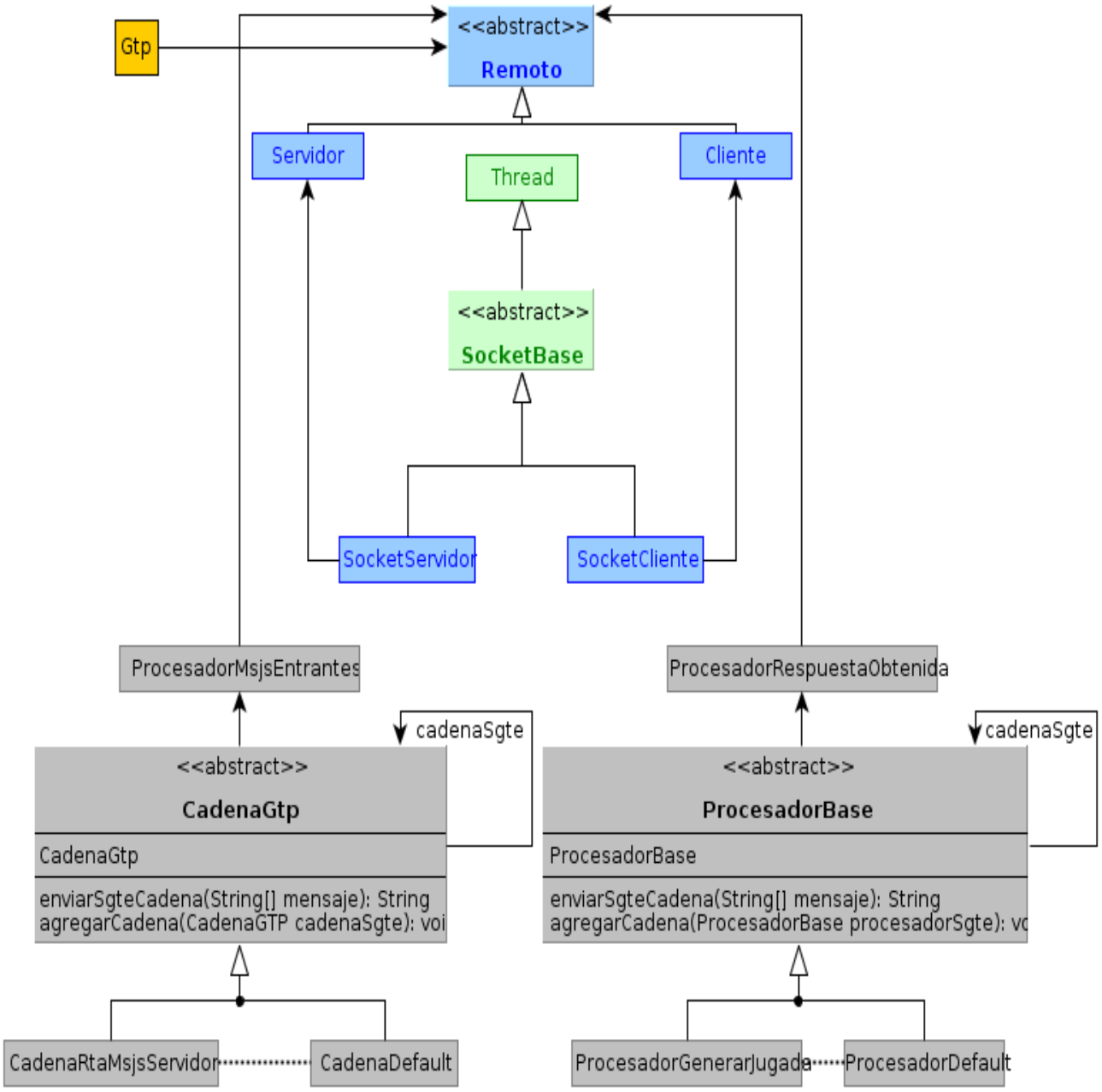
4. Detalles de implementación

4.1. Vista

4.2. Protocolo de comunicaciones

La clase remoto es la encargada del manejo del protocolo de texto Go(gtp), si el modelo desea enviar un mensaje lo debe hacer por esta clase. Como se puede observar en el diagrama servidor y cliente son hijos de remoto; esto se debe a que el proceso de mensajes entrantes y saliente es el mismo en ambos casos, la única diferencia existente es el momento inicial donde se establece la conexión. Para el proceso de los mensajes, tenemos dos cadenas de responsabilidades; una encargada de los mensajes respuestas(los cuales se caracterizan por el

inicio con el caracter =) y los mensajes comandos. Estas cadenas procesan dichos mensajes y le informan al modelo, para que el mismo decida como se debe continuar.
La versión del protocolo utilizada es la 2, a pesar que el protocolo fue implementado totalmente; para los alcances del trabajo práctico solo se utilizan los comandos más importantes.



4.3. Estrategias implementadas

Actualmente se cuenta con cuatro estrategias de juego que serán utilizadas posteriormente para elaborar estrategias mas avanzadas:

4.3.1. EstrategiaComputadoraAtacar

Esta estrategia intenta ocupar casilleros adyacentes a las cadenas con menor grado de libertad del oponente, intentando capturarlas.

4.3.2. EstrategiaComputadoraDefender

Esta estrategia intenta ocupar casilleros adyacentes a las cadenas propias con menor grado de libertad, intentando evitar que sean capturadas.

4.3.3. EstrategiaAtaqueCuidadoso

Esta estrategia es una combinación de las estrategias "EstrategiaComputadoraAtacar" y "EstrategiaComputadoraDefender". Primero verifica que las cadenas propias no estén en peligro de ser capturadas. Si se encuentra una cadena propia en riesgo aplica la estrategia "EstrategiaComputadoraDefender", en caso contrario aplica "EstrategiaComputadoraAtacar".

4.3.4. EstrategiaAtaqueCuidadosoMasInteligente

Similar a la estrategia anterior, pero primero verifica si existe alguna cadena del oponente con grado de libertad 1. Si existe, verifica que ese grado de libertad no se deba a un ojo. Si no se deba a un ojo se procede a capturar al grupo. Si no se cumplen estas condiciones, se aplica la estrategia "EstrategiaAtaqueCuidadoso".

4.3.5. EstrategiaMiniMax

Implementa una estrategia del tipo **MiniMax**. En cada turno, arma una lista de todos los casilleros vacíos y despliega un árbol de jugadas por cada posible casillero. La profundidad hasta la cual despliega elárbol de jugadas es configurable. Al llegar a la profundidad deseada, se aplica la función de evaluación a los tableros resultantes.

La función de evaluación tiene en cuenta las siguientes variables:

- Grados de libertad de MAX: Se cuentan todos los casilleros adyacentes a cada cadena de MAX libres (no se cuentan los repetidos). Se quiere que sea lo mayor posible.
- Cantidad de ojos de MAX
- Grados de libertad de la cadena mas corta de MAX: la variable a la que se le da más importancia.
- Grados de libertad de la cadena mas corta de MIN.
- Grados de libertad de la cadena mas larga de MIN.
- Ojos de MIN.

Por encima de las variables arriba mencionadas, se encuentra la condición de que alguna de alguna de las cadenas de MIN tenga grado 1. En ese caso se da por ganada la partida (en esa rama del árbol de jugadas).

Al desplegar el árbol de jugadas, si en algún nivel todos los movimientos son inválidos, se da por finalizada la partida y no se sigue avanzando hasta los niveles mas profundos.

4.3.6. Mejoras posibles

La implementación MiniMax presentada se puede mejorar teniendo en cuenta lo siguiente:

- Si se es el primero en jugar, no es necesario descender en el árbol de jugadas. Sería conveniente comenzar con jugadas precalculadas.
- Se tienen en cuenta como jugadas posibles todos los casilleros libres del tablero. Se podría tener en cuenta solamente los casilleros adyacentes a todas las cadenas presentes en el tablero solamente, limitando así las jugadas, pero reduciendo el procesamiento, eventualmente dándonos la posibilidad de descender un poco mas en el árbol de jugadas.
- La cantidad de niveles que se baja en el árbol de jugadas es fija. Se puede parametrizar en función de las jugadas posibles. Al principio de la partida, al haber muchas posibilidades se elige por ejemplo descender hasta el nivel 3, pero a medida que quedan menos posibilidades (la mitad del tablero puede ser un caso) podríamos aumentar un nivel sin perder mucha velocidad de respuesta.
- La función de evaluación actualmente es una suma pesada de diferentes variables. Se pueden implementar diferentes funciones de evaluación mas sofisticadas.