Universidad ORT

Facultad Ingeniería

Programación 2

Obligatorio 1 – Grupo M2A

Rodolfo Agustín Silva
(187061)

Andrés Lacaño
(158910)

(158910) (187061)

Mayo 2015

Índice

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Casos de uso

Caso de uso: a) Registrar jugador

Actor: Usuario

Curso normal: 1) El usuario ingresa a la opción 1

2) Ingresa el nombre

3) Ingresa la edad

4) Ingresa el alias

5) Se registra al sistema el jugador

Curso alternativo: 3) Si la edad no es un valor numérico, el sistema pide reingresar la edad

4) Si el alias ya existe y está en uso, el sistema pide reingresar otro

Caso de uso: c) Ranking de jugadores

Actor: Usuario

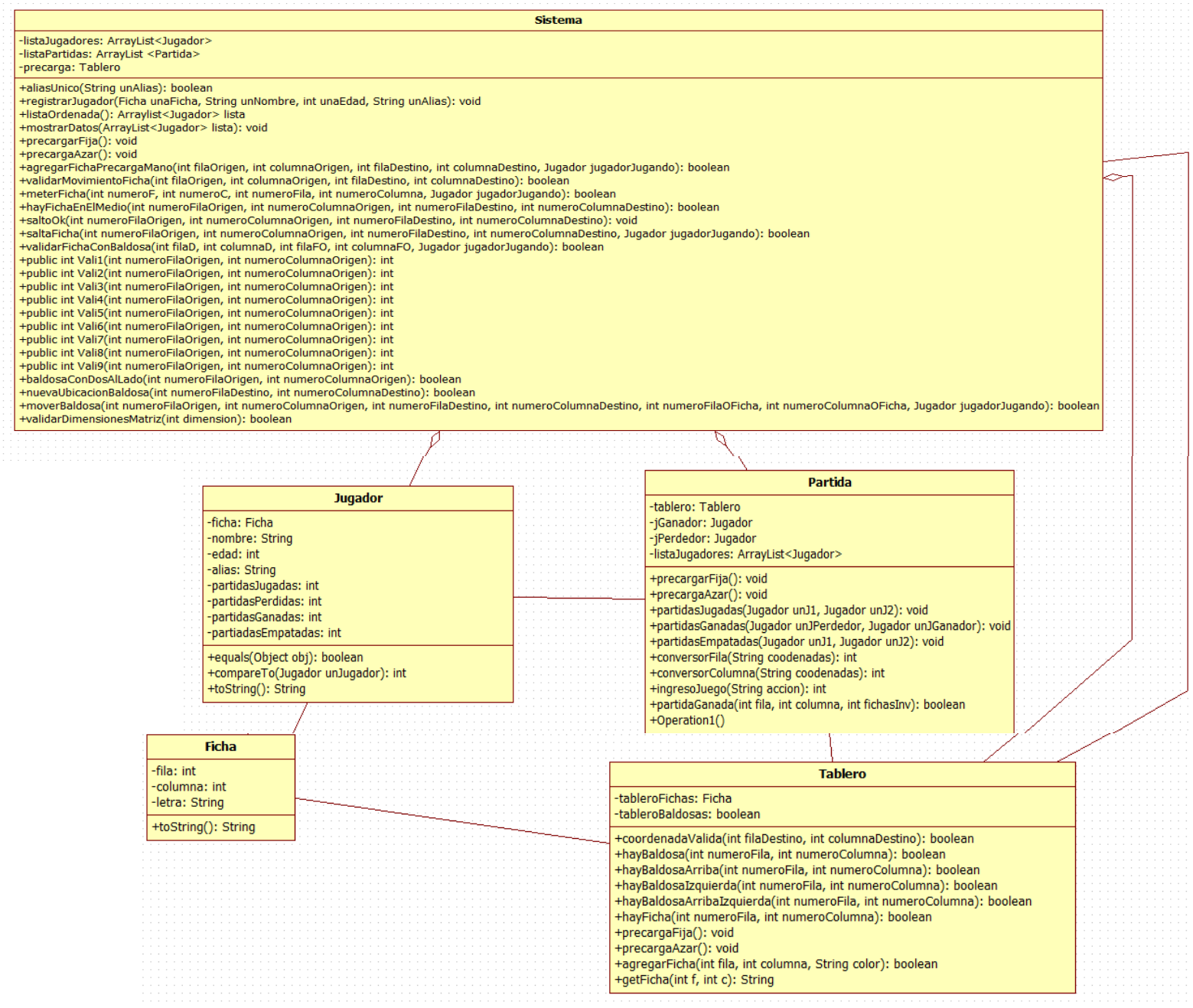
Curso normal: 1) El usuario elige la opción de consulta de jugadores

2) El sistema muestra en pantalla una lista de jugadores ordenada por partidas ganadas

Curso alternativo: 2) Si no se ha realizado ninguna partida, el sistema dice que no hay jugadores para mostrar un ranking

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Diagrama de clases UML



Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Datos de prueba

Descripción del dato a probar	Resultado esperado	Resultado obtenido
Se disponen de 5 opciones en el menú principal. Se elige la opción 5-Salir	Se finalice el programa	ok
Se disponen de 5 opciones en el menú principal. Se elige una opción invalida, ej. 6	Pide que reingreses una opción valida	ok
Se disponen de 5 opciones en el menú principal. Se elige una opción invalida, ej. 0	Pide que reingreses una opción valida	ok
Se disponen de 5 opciones en el menú principal. Se elige una opción invalida, ej. #	Pide que reingreses una opción valida	ok
Se disponen de 5 opciones en el menú principal. Se elige la opción 1-Registrar jugador	Nos deja registrar un jugador	ok
Se disponen de 5 opciones en el menú principal. Se elige la opción 1-Registrar jugador con nombre, edad (entre 0 y 100), alias(Prueba, 20, Pru)	Se registra el jugador en el sistema	ok
Se disponen de 5 opciones en el menú principal. Se elige la opción 1-Registrar jugador con nombre, edad (entre 0 y 100), alias (, 20, Pru)	No se registra el jugador, no se aceptan espacios en blanco. Se pide que reingrese	ok
Se disponen de 5 opciones en el menú principal. Se elige la opción 1-Registrar jugador con nombre, edad (entre 0 y 100), alias (Prueba , 4000, Pru)	No se registra el jugador, edad no valida blanco. Se pide que reingrese	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 1-Registrar jugador con nombre, edad (entre 0 y 100), alias previamente un jugador registrado con los siguientes datos: (Hola, 21, Pru) Se dispone a registrar un segundo jugador (Prueba , 4000, Pru)	No se registra el jugador, el alias ya está en uso. Se pide que reingrese	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se ingresa una opción invalida ej.0	Opción invalida, se pide que reingrese una valida	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se ingresa una opción invalida ej.4	Opción invalida, se pide que reingrese una valida	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se ingresa una opción invalida ej.-1	Opción invalida, se pide que reingrese una valida	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se ingresa una opción invalida ej./	Opción invalida, se pide que reingrese una valida	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se ingresa una opción válida ej.1-Precarga al azar	Se ingresa en la opción elegida	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se ingresa una opción válida ej.2-Precarga a mano	Se ingresa en la opción elegida	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se ingresa una opción válida ej.3-Precarga fija	Se ingresa en la opción elegida	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 1-Precarga al azar	Al iniciar la partida el tablero va a tener las fichas repartidas en posiciones al azar	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga a mano	Se pide al jugador blanco que ingrese dentro del tablero las posiciones de sus 6 fichas, luego las 6 del jugador rojo	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga a mano Se ingresa una coordenada valida de la primer ficha (posiciones validas dentro de fila E a H y columnas de 5 a 9) ej. E6	Se ingresa las fichas en el tablero a jugar a futuro	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga a mano Se ingresa una coordenada invalida de la primer ficha (posiciones validas dentro de fila E a H y columnas de 5 a 9) ej. E2	Se pide que se reingresen las coordenadas de la ficha	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga a mano Se ingresa una coordenada invalida de la primer ficha (posiciones validas dentro de fila E a H y columnas de 5 a 9) ej. E&	Se pide que se reingresen las coordenadas de la ficha	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga a mano Se ingresa una coordenada valida de la primer ficha (posiciones validas dentro de fila E a H y columnas de 5 a 9) ej. A7	Se pide que se reingresen las coordenadas de la ficha	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga a mano Se ingresa una coordenada valida de la primer ficha (posiciones validas dentro de fila E a H y columnas de 5 a 9) ej. &12	Se pide que se reingresen las coordenadas de la ficha	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga a mano Se ingresa una coordenada valida de la primer ficha (posiciones validas dentro de fila E a H y columnas de 5 a 9) ej. E	Se pide que se reingresen las coordenadas de la ficha	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga a mano Se ingresa una coordenada valida de la primer ficha (posiciones validas dentro de fila E a H y columnas de 5 a 9) ej. 5	Se pide que se reingresen las coordenadas de la ficha	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 2-Precarga, se disponen de 3 opciones dentro. Se selecciona la opción 2-Precarga fija	Se ingresa las fichas de forma predeterminada en el tablero a jugar a futuro	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, sin haber registrado previamente dos jugadores en el sistema	No te permite jugar baldosas, te pide que ingreses dos jugadores mínimo en el sistema	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema	Te permite ingresar a la opción de jugar baldosas	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se despliega la lista de jugadores en este caso hay 2. Se elige un jugador valido ej.1	Se selecciona el jugador	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se despliega la lista de jugadores en este caso hay 2. Se elige un jugador invalido ej.3	Te pide que elijas un jugador existente	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se despliega la lista de jugadores en este caso hay 2. Se elige un jugador invalido ej.3	Te pide que elijas un jugador existente	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se despliega la lista de jugadores en este caso hay 2. Se elige un jugador invalido ej.&	Te pide que elijas un jugador existente	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se despliega la lista de jugadores en este caso hay 2. Se elige un jugador invalido ej.-4	Te pide que elijas un jugador existente	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se despliega la lista de jugadores en este caso hay 2. El primer jugador elige al jugador 1 del sistema. El jugador 2 también elige al jugador 1 del sistema	No permite seleccionar al jugador 2 el jugador uno del sistema porque ya está en uso y le pide que elija uno que este disponible	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. No se ingresó previamente al menú de Precarga	Las fichas en el tablero se distribuyen al azar	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se ingresó previamente al menú de Precarga y se seleccionó la precarga a mano	Las fichas se distribuyen en el tablero de la forma que los jugadores las posicionaron en precarga a mano	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se ingresó previamente al menú de Precarga y se seleccionó la precarga fija	Las fichas se distribuyen en el tablero de forma ya predeterminada	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se muestran dos tipos de tablero 1-Sin bordes, 2-Con bordes Se elige uno de los dos, ej. 1	Cuando se muestre el tablero solo se van a ver las coordenadas y las baldosas	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se muestran dos tipos de tablero 1-Sin bordes, 2-Con bordes Se elige una opción invalida ej. 3	Se pide que ingreses una opción valida	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se muestran dos tipos de tablero 1-Sin bordes, 2-Con bordes Se elige una opción invalida ej. 0	Se pide que ingreses una opción valida	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se muestran dos tipos de tablero 1-Sin bordes, 2-Con bordes Se elige una opción invalida ej. #	Se pide que ingreses una opción valida	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se ingresa la cantidad de fichas invertidas necesarias para ganar (1 a 6) ej.2	Se acepta la cantidad de fichas invertidas	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se ingresa la cantidad de fichas invertidas necesarias para ganar (1 a 6) ej.8	No se acepta la cantidad de fichas invertidas y se pide que reingrese un valor valido	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se ingresa la cantidad de fichas invertidas necesarias para ganar (1 a 6) ej.0	No se acepta la cantidad de fichas invertidas y se pide que reingrese un valor valido	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 3-Jugar baldosas, habiendo registrado previamente dos jugadores en el sistema. Se ingresa la cantidad de fichas invertidas necesarias para ganar (1 a 6) ej.%	No se acepta la cantidad de fichas invertidas y se pide que reingrese un valor valido	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. M E5E6	Se mueve la ficha blanca del jugador 1	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. m e5E6	Se pide que se reingrese en mayúscula la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. M E5E7	Movimiento invalida, se pide que reingrese	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. M E5E15	Movimiento invalida, se pide que reingrese	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. M E5 E6	Movimiento invalida, se pide que reingrese	
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. ME5E6	Movimiento invalida, se pide que reingrese	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. F E5E6	Movimiento invalida, se pide que reingrese	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. M E5	Movimiento invalida, se pide que reingrese	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. M E5&6	Movimiento invalida, se pide que reingrese	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. M &5E6	Movimiento invalida, se pide que reingrese	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero ej. M E5E6E7	Movimiento invalida, se pide que reingrese	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero en una posición ya ocupada por otra suya ej. M E5F5	Movimiento invalido, se pide que reingrese	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una ficha suya en el tablero en una posición ya ocupada por una del oponente ej. M E5E6	Movimiento invalido, se pide que reingrese	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S E7E9	Se realiza la jugada, la ficha se desplaza a esa posición y se invierte quedando “b”	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S E7e9	Se pide que se reingrese en mayúscula la jugada	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S E15E9	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S E7 E9	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. SE7E9	Movimiento invalida, se pide reingrese la jugada	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. SE7E9	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S E7E8	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. K E7E8	Movimiento invalida, se pide reingrese la jugada	

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S E7&9	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S &7E9	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S E7E9E8	Movimiento invalida, se pide reingrese la jugada	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S E7	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a saltar una ficha suya en el tablero ej. S 7E	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero, moviendo una ficha ej. M E9DE5	Se acepta el movimiento y se mueve una baldosa, moviendo una ficha encima	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero, saltando una ficha ej. M E9D8F8	Se acepta el movimiento y se mueve una baldosa, haciendo saltar la ficha encima y esta se invierte a "b"	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero, saltando una ficha ej. M E9D8f8	Se pide que se reingrese en mayúscula la jugada	ok

Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero ej. M E9D8E6	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero ej. M E9D10F8	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero ej. H E9D8E8	Movimiento invalida, se pide reingrese la jugada	ok

Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero ej. ME9D8E8	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero ej. M E9 D8 E8	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero ej. M E9&8E8	Movimiento invalida, se pide reingrese la jugada	ok

Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero ej. M E9D8E8G8H4	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero ej. M E40D8E8	Movimiento invalida, se pide reingrese la jugada	ok

Se arrancó la partida con los dos jugadores, en precarga fija. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a mover una baldosa en el tablero, intentando realizarlo utilizando las fichas rojas del oponente ej. M E9F4F5	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a solicitar el empate ej. E (el jugador 2 lo acepta)	Se muestra el cartel de solicitud de empate, al haber sido aceptada, se termina la partida	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a solicitar el empate ej. E (el jugador 2 no acepta)	Se muestra el cartel de solicitud de empate, al no haber sido aceptada, se continua con la partida manteniéndose en turno del jugador que lo solicito (jugador 1)	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a solicitar el empate de forma invalida ej. E9	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a solicitar el empate de forma invalida ej. E &	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a solicitar el empate de forma invalida ej. P	Movimiento invalida, se pide reingrese la jugada	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a solicitar el empate de forma invalida ej. P	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a solicitar el empate de forma invalida ej. e	Se pide que se reingrese en mayúscula la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a abandonar la partida ej. X	Se acepta la jugada, el jugador 1 se rinde y el jugador 2 es el ganador	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a abandonar la partida de forma invalida ej. X 10	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a abandonar la partida de forma invalida ej. X?	Movimiento invalida, se pide reingrese la jugada	ok
Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a abandonar la partida de forma invalida ej. T	Movimiento invalida, se pide reingrese la jugada	ok

Se arrancó la partida con los dos jugadores. Los movimientos disponibles que tienen los jugadores son, mover ficha, saltar ficha, mover baldosa, solicitar empate y abandonar. El jugador 1 corresponde a las fichas blancas y el 2 a las rojas. El jugador uno se dispone a abandonar la partida de forma invalida ej. x	Se pide que se reingrese en mayúscula la jugada	ok
Se arrancó la partida con los dos jugadores, en precarga fija. Se eligió un total de 1 fichas invertidas para ganar Se realizan los siguientes movimientos: jugador 1, jugador 2 a continuación		
S F8H8, M F5F6, B E9H10G9, M G8G9, M G7G8, M G5G6, M G8H9, M H6G7, M H5H6, S F7F5, M H6H7 (ver imagen1)	Gana el jugador 1 con las fichas en forma horizontal, se acaba la partida	ok
Se arrancó la partida con los dos jugadores, en precarga fija. Se eligió un total de 1 fichas invertidas para ganar Se realizan los siguientes movimientos: jugador 1, jugador 2 a continuación		
S F8F6, M F7F8, M F6F7, M H6G6, M H5H6, M G6F6, M H6H7 (ver imagen 2)	Gana el jugador 1 con las fichas verticalmente, se acaba la partida	ok

Se arrancó la partida con los dos jugadores, en precarga fija. Se eligió un total de 2 fichas invertidas para ganar Se realizan los siguientes movimientos: jugador 1, jugador 2 a continuación		
S H5H7, S H6H8, B H5D6E5, M H8H9, M G7G6, S G5G7, M G6G5, M G7F6, M H7H8, M F6E5, B E9I9H8, B E8D5E5. (ver imagen 3)	Gana el jugador2 con las fichas diagonalmente, se acaba la partida	OK
Se arrancó la partida con los dos jugadores, en precarga fija. Se eligió un total de 2 fichas invertidas para ganar Se realizan los siguientes movimientos: jugador 1, jugador 2 a continuación		
S F8F6, M F7F8, S G7E9, M G8G7, B H9D8E9, S F5F7, B H8D6E5, M F7E8,B H7D7D6 M E8E9, B D6C8D7, B E5I5G5 (ver imagen 4)	Gana el jugador2 con las fichas diagonalmente (en sentido opuesto al caso anterior), se acaba la partida	OK
Se dispone de 5 opciones en el menú principal. Se elige la opción 4- Ranking sin haber registrado previamente ningún jugador en ele sistema	No te permite entrar a la opción porque no tiene al menos un jugador en el sistema	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 4- Ranking habiendo registrado dos jugadores previamente (P, 20 Jugador1) y (K, 21 Jugador2), no se ha jugado ninguna partida aun	Ordena los jugadores por alias	ok

Se dispone de 5 opciones en el menú principal. Se elige la opción 4- Ranking, habiendo registrado dos jugadores previamente (P, 20 Jugador1) y (K, 21 Jugador2), sabiendo que ya se jugaron dos partidas de baldosas	Muestra la lista ordenada por partidas ganadas, y en caso de tener la misma cantidad de partidas ganadas, ordena por alias	ok
Se dispone de 5 opciones en el menú principal. Se elige la opción 4- Ranking, habiendo registrado dos jugadores previamente (P, 20 Jugador1) y (K, 21 Jugador2), sabiendo que ya se jugaron dos partidas de baldosas y las dos las gano el Jugador1	Muestra la lista ordenada por partidas ganadas	ok

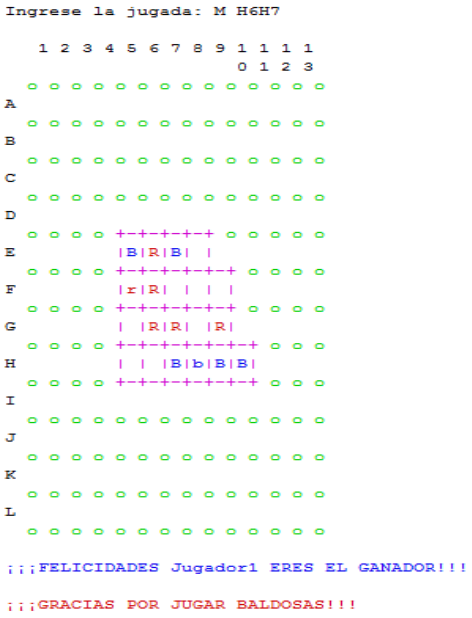


Imagen 1

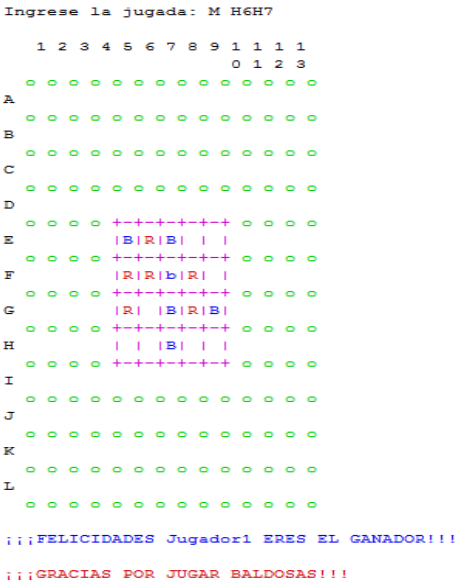


Imagen 2

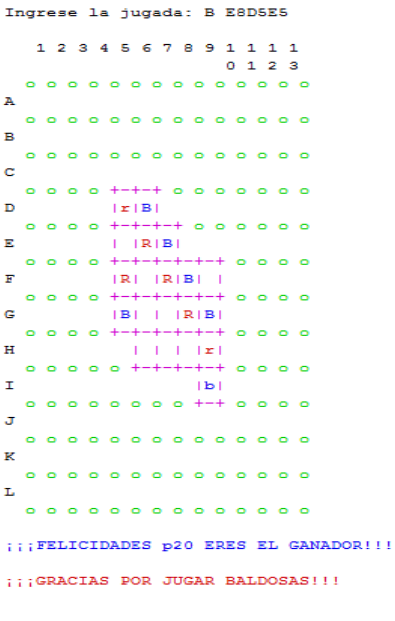


Imagen 3

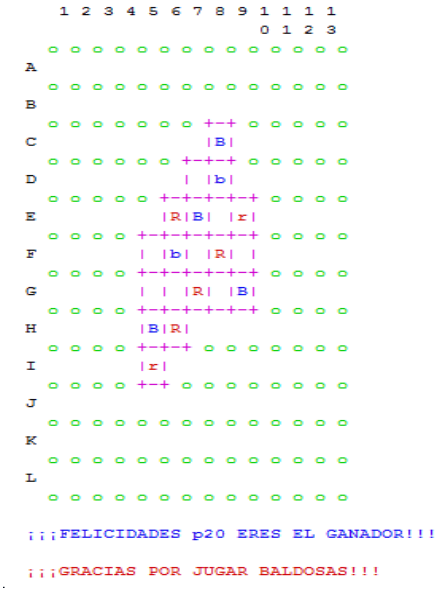
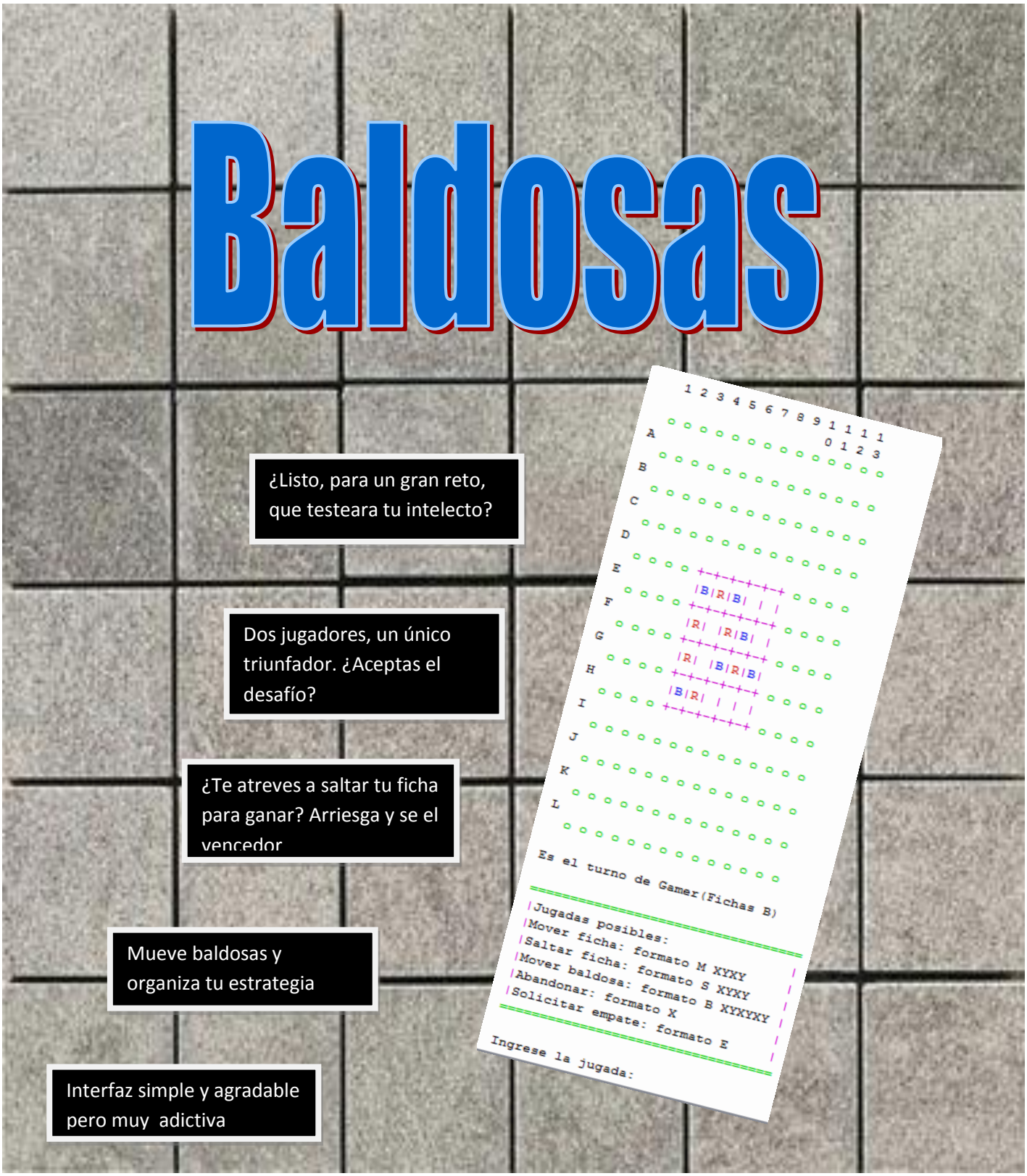


Imagen 4

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Folleto promocional de Baldosas

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)



Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Cambios de lógica para el juego

Como primer cambio para la lógica, agregaríamos a cada jugador una única nueva ficha que al ser colocada volviera todas las fichas invertidas del rival en minúscula sea “b” o “r” a forma normal, es decir “B” y “R”.
Como segundo cambio de lógica para el juego, agregaríamos que cada jugador tenga la opción dentro del juego iniciado de poder cambiar el icono de sus fichas, por otra letra o un símbolo a su elección, sin interferir con el tablero. Esto no contaría como cambio de turno al ser realizado.

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Listado impreso de clases

Clase Sistema

```
package Dominio;


import java.util.ArrayList;

import java.util.Collections;


public class Sistema {


    private ArrayList<Jugador> listaJugadores;

    private ArrayList<Partida> listaPartidas;

    private Tablero precargar;


    public ArrayList<Jugador> getListaJugadores() {

        return listaJugadores;

    }


    public void agregarJugador(Jugador unJugador) {

        listaJugadores.add(unJugador);

    }


    public ArrayList<Partida> getListaPartidas() {

        return listaPartidas;

    }


    public void agregarPartida(Partida unaPartida) {

        unaPartida.setTablero(precargar);

        listaPartidas.add(unaPartida);

    }
```

```
public Tablero getPrecarga() {
    return precargar;
}

public Sistema() {
    listaJugadores = new ArrayList<Jugador>();
    listaPartidas = new ArrayList<Partida>();
}

public Partida partidaActual() {
    return listaPartidas.get(listaPartidas.size() - 1);
}

public boolean aliasUnico(String unAlias) {
    boolean ok = false;
    for (int i = 0; i < this.listaJugadores.size(); i++) {
        if (this.listaJugadores.get(i).getAlias().equals(unAlias)) {
            ok = true;
        }
    }
    return ok;
}

public void registrarJugador(Ficha unaFicha, String unNombre, int unaEdad, String unAlias) {
    Jugador unJugador = new Jugador(unaFicha, unNombre, unaEdad, unAlias, 0, 0, 0);
    this.listaJugadores.add(unJugador);
}
```

```
    public ArrayList<Jugador> listaOrdenada() {
        ArrayList<Jugador> lista = new ArrayList<Jugador>();
        for (int i = 0; i < this.getListaJugadores().size(); i++) {
            Jugador unJ = this.getListaJugadores().get(i);
            lista.add(unJ);
        }
        Collections.sort(lista);
        return lista;
    }

    public void mostrarDatos(ArrayList<Jugador> lista) {
        for (int i = 0; i < lista.size(); i++) {
            Jugador unJugador = lista.get(i);
            System.out.println("El jugador " + unJugador.getAlias()
                    + " jugo un total de " + unJugador.getPartidasJugadas() + " veces"
                    + " y gano un total de " + unJugador.getPartidasGanadas() + " partidas");
        }
    }

    public void precargarFija() {
        partidaActual().precargarFija();
    }

    public void precargaAzar() {
        partidaActual().precargaAzar();
    }

    public boolean agregarFichaPrecargaMano(String fila, String columna, String color) {
        return partidaActual().getTablero().agregarFicha(Partida.conversorFila(fila),
Partida.conversorColumna(columna), color);
```

```
    }


    public void moverFicha(int filaOrigen, int columnaOrigen, int filaDestino, int columnaDestino,
Jugador jugadorJugando) {

        Ficha fichaAux = partidaActual().getTablero().getTableroFichas()[filaOrigen][columnaOrigen];

        if (jugadorJugando.getFicha().getLetra().equals(fichaAux.getLetra().toUpperCase())) {

            if (partidaActual().getTablero().hayBaldosa(filaDestino, columnaDestino)) {

                if (validarMovimientoFicha(filaOrigen, columnaOrigen, filaDestino, columnaDestino)) {

                    partidaActual().getTablero().getTableroFichas()[filaDestino][columnaDestino] =
fichaAux;

                    partidaActual().getTablero().getTableroFichas()[filaOrigen][columnaOrigen] = null;

                }

            }

        }

    }


    public boolean validarMovimientoFicha(int filaOrigen, int columnaOrigen, int filaDestino, int
columnaDestino) {

        boolean ok = false;

        if ((Math.abs(filaOrigen - filaDestino) == 1 && Math.abs(columnaOrigen - columnaDestino) == 1) ||
(Math.abs(filaOrigen - filaDestino) == 0 && Math.abs(columnaOrigen - columnaDestino) == 1) ||
(Math.abs(filaOrigen - filaDestino) == 1 && Math.abs(columnaOrigen - columnaDestino) == 0)) {

            if (partidaActual().getTablero().coordenadaValida(filaDestino, columnaDestino)) {

                ok = true;

            }

        }

        return ok;

    }


    public boolean meterFicha(int numeroF, int numeroC, int numeroFila, int numeroColumna, Jugador
jugadorJugando) {

        boolean ok = false;

        if (partidaActual().getTablero().coordenadaValida(numeroFila, numeroColumna)) {
```

```
            if (partidaActual().getTablero().hayBaldosa(numeroFila, numeroColumna)) {

                if (partidaActual().getTablero().hayFicha(numeroF, numeroC)) {

                    if (!partidaActual().getTablero().hayFicha(numeroFila, numeroColumna)) {

                        if
(jugadorJugando.getFicha().getLetra().equals(partidaActual().getTablero().getTableroFichas()[numeroF][num
eroC].getLetra().toUpperCase())) {

                            ok = true;

                        }

                    }

                }

            }

        }

        return ok;

    }


    public boolean hayFichaEnElMedio(int numeroFilaOrigen, int numeroColumnaOrigen, int
numeroFilaDestino, int numeroColumnaDestino) {

        int fila;

        int columna;

        if (numeroFilaOrigen == numeroFilaDestino) {

            fila = numeroFilaOrigen;

        } else {

            fila = Math.max(numeroFilaOrigen, numeroFilaDestino) - 1;

        }

        if (numeroColumnaOrigen == numeroColumnaDestino) {

            columna = numeroColumnaOrigen;

        } else {

            columna = Math.max(numeroColumnaOrigen, numeroColumnaDestino) - 1;

        }

        return partidaActual().getTablero().hayFicha(fila, columna);

    }
```

```
    public boolean saltoOk(int numeroFilaOrigen, int numeroColumnaOrigen, int numeroFilaDestino, int
numeroColumnaDestino) {

        boolean ok = false;

        if ((Math.abs(numeroFilaOrigen - numeroFilaDestino) == 2 && Math.abs(numeroColumnaOrigen -
numeroColumnaDestino) == 2) || (Math.abs(numeroFilaOrigen - numeroFilaDestino) == 0 &&
Math.abs(numeroColumnaOrigen - numeroColumnaDestino) == 2) || (Math.abs(numeroFilaOrigen -
numeroFilaDestino) == 2 && Math.abs(numeroColumnaOrigen - numeroColumnaDestino) == 0)) {

            if ((partidaActual().getTablero().coordenadaValida(numeroFilaDestino, numeroColumnaDestino))
&& (hayFichaEnElMedio(numeroFilaOrigen, numeroColumnaOrigen, numeroFilaDestino, numeroColumnaDestino))) {

                ok = true;

            }

        }

        return ok;

    }


    public boolean saltaFicha(int numeroFilaOrigen, int numeroColumnaOrigen, int numeroFilaDestino, int
numeroColumnaDestino, Jugador jugadorJugando) {

        boolean ok = false;

        if (partidaActual().getTablero().coordenadaValida(numeroFilaDestino, numeroColumnaDestino)) {

            if (partidaActual().getTablero().hayBaldosa(numeroFilaDestino, numeroColumnaDestino)) {

                if (!partidaActual().getTablero().hayFicha(numeroFilaDestino, numeroColumnaDestino)) {

                    if (partidaActual().getTablero().hayFicha(numeroFilaOrigen, numeroColumnaOrigen)) {

                        Ficha fichaAux =
partidaActual().getTablero().getTableroFichas()[numeroFilaOrigen][numeroColumnaOrigen];

                        if
(jugadorJugando.getFicha().getLetra().equals(fichaAux.getLetra().toUpperCase())) {

                            if (saltoOk(numeroFilaOrigen, numeroColumnaOrigen, numeroFilaDestino,
numeroColumnaDestino)) {

                                if (fichaAux.getLetra().equals("R") || fichaAux.getLetra().equals("B")) {

                                    fichaAux.setLetra(fichaAux.getLetra().toLowerCase());

                                } else {

                                    fichaAux.setLetra(fichaAux.getLetra().toUpperCase());

                                }
```

```
partidaActual().getTablero().getTableroFichas()[numeroFilaDestino][numeroColumnaDestino] = fichaAux;

partidaActual().getTablero().getTableroFichas()[numeroFilaOrigen][numeroColumnaOrigen] = null;
                                ok = true;
                            }
                        }
                    }
                }
            }
        }
        return ok;
    }

    public boolean validarFichaConBaldosa(int filaD, int columnaD, int filaFO, int columnaFO, Jugador
jugadorJugando) {
        boolean ok = false;
        if (partidaActual().getTablero().coordenadaValida(filaFO, columnaFO)) {
            Ficha fichaAux = partidaActual().getTablero().getTableroFichas()[filaFO][columnaFO];
            if (fichaAux != null &&
jugadorJugando.getFicha().getLetra().equals(fichaAux.getLetra().toUpperCase())) {
                if (saltoOk(filaFO, columnaFO, filaD, columnaD) || validarMovimientoFicha(filaFO,
columnaFO, filaD, columnaD)) {
                    ok = true;
                }
            }
        }
        return ok;
    }

    public int Vali1(int numeroFilaOrigen, int numeroColumnaOrigen) {
        int cont = 0;
```

```
        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen - 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen - 1]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen + 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen + 1]
== true) {

            cont++;

        }

        return cont;

    }



    public int Vali2(int numeroFilaOrigen, int numeroColumnaOrigen) {

        int cont = 0;

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen - 1]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen + 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen + 1]
== true) {

            cont++;

        }
```

```
        return cont;

    }


    public int Vali3(int numeroFilaOrigen, int numeroColumnaOrigen) {

        int cont = 0;

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen + 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen + 1]
== true) {

            cont++;

        }

        return cont;

    }


    public int Vali4(int numeroFilaOrigen, int numeroColumnaOrigen) {

        int cont = 0;

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen - 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen + 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen + 1]
== true) {

            cont++;

        }

        return cont;

    }
```

```
    public int Vali5(int numeroFilaOrigen, int numeroColumnaOrigen) {

        int cont = 0;

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen + 1]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen - 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen - 1]
== true) {

            cont++;

        }

        return cont;

    }


    public int Vali6(int numeroFilaOrigen, int numeroColumnaOrigen) {

        int cont = 0;

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen - 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen - 1]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen + 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        return cont;
```

```
    }


    public int Vali7(int numeroFilaOrigen, int numeroColumnaOrigen) {

        int cont = 0;

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen - 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen - 1]
== true) {

            cont++;

        }

        return cont;

    }


    public int Vali8(int numeroFilaOrigen, int numeroColumnaOrigen) {

        int cont = 0;

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen - 1][numeroColumnaOrigen]
== true) {

            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen + 1]
== true) {

            cont++;

        }

        return cont;

    }


    public int Vali9(int numeroFilaOrigen, int numeroColumnaOrigen) {

        int cont = 0;

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen + 1][numeroColumnaOrigen]
== true) {
```

```
            cont++;

        }

        if (partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen - 1]
== true) {

            cont++;

        }

        return cont;

    }


    public boolean baldosaConDosAlLado(int numeroFilaOrigen, int numeroColumnaOrigen) {

        boolean ok = false;

        int cont = 0;

        if (numeroFilaOrigen != 0 && numeroColumnaOrigen != 0 && numeroFilaOrigen != 11 &&
numeroColumnaOrigen != 12) {

            cont = Vali1(numeroFilaOrigen, numeroColumnaOrigen);

        } else {

            if (numeroFilaOrigen == 0 && numeroColumnaOrigen != 0 && numeroColumnaOrigen != 12) {

                cont = Vali2(numeroFilaOrigen, numeroColumnaOrigen);

            } else {

                if (numeroFilaOrigen == 0 && numeroColumnaOrigen == 0) {

                    cont = Vali3(numeroFilaOrigen, numeroColumnaOrigen);

                } else {

                    if (numeroFilaOrigen != 0 && numeroFilaOrigen != 11 && numeroColumnaOrigen == 0) {

                        cont = Vali4(numeroFilaOrigen, numeroColumnaOrigen);

                    } else {

                        if (numeroFilaOrigen == 11 && numeroColumnaOrigen != 0 && numeroColumnaOrigen !=
12) {

                            cont = Vali5(numeroFilaOrigen, numeroColumnaOrigen);

                        } else {

                            if (numeroFilaOrigen != 11 && numeroFilaOrigen != 0 && numeroColumnaOrigen ==
12) {

                                cont = Vali6(numeroFilaOrigen, numeroColumnaOrigen);
```

```
                            } else {
                                if (numeroFilaOrigen == 11 && numeroColumnaOrigen == 12) {
                                    cont = Vali7(numeroFilaOrigen, numeroColumnaOrigen);
                                } else {
                                    if (numeroFilaOrigen == 11 && numeroColumnaOrigen == 0) {
                                        cont = Vali8(numeroFilaOrigen, numeroColumnaOrigen);
                                    } else {
                                        if (numeroFilaOrigen == 0 && numeroColumnaOrigen == 12) {
                                            cont = Vali9(numeroFilaOrigen, numeroColumnaOrigen);
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
        if (cont <= 2) {
            ok = true;
        }
        return ok;
    }

    public boolean nuevaUbicacionBaldosa(int numeroFilaDestino, int numeroColumnaDestino) {
        boolean ok = false;
        int cont = 0;
        if (partidaActual().getTablero().coordenadaValida(numeroFilaDestino, numeroColumnaDestino)) {
            if (numeroFilaDestino != 0 && numeroColumnaDestino != 0 && numeroFilaDestino != 11 &&
numeroColumnaDestino != 12) {
```

```
                cont = Vali1(numeroFilaDestino, numeroColumnaDestino);
            } else {
                if (numeroFilaDestino == 0 && numeroColumnaDestino != 0 && numeroColumnaDestino != 12) {
                    cont = Vali2(numeroFilaDestino, numeroColumnaDestino);
                } else {
                    if (numeroFilaDestino == 0 && numeroColumnaDestino == 0) {
                        cont = Vali3(numeroFilaDestino, numeroColumnaDestino);
                    } else {
                        if (numeroFilaDestino != 0 && numeroFilaDestino != 11 && numeroColumnaDestino ==
0) {
                            cont = Vali4(numeroFilaDestino, numeroColumnaDestino);
                        } else {
                            if (numeroFilaDestino == 11 && numeroColumnaDestino != 0 &&
numeroColumnaDestino != 12) {
                                cont = Vali5(numeroFilaDestino, numeroColumnaDestino);
                            } else {
                                if (numeroFilaDestino != 11 && numeroFilaDestino != 0 &&
numeroColumnaDestino == 12) {
                                    cont = Vali6(numeroFilaDestino, numeroColumnaDestino);
                                } else {
                                    if (numeroFilaDestino == 11 && numeroColumnaDestino == 12) {
                                        cont = Vali7(numeroFilaDestino, numeroColumnaDestino);
                                    } else {
                                        if (numeroFilaDestino == 11 && numeroColumnaDestino == 0) {
                                            cont = Vali8(numeroFilaDestino, numeroColumnaDestino);
                                        } else {
                                            if (numeroFilaDestino == 0 && numeroColumnaDestino == 12) {
                                                cont = Vali9(numeroFilaDestino, numeroColumnaDestino);
                                            }
                                        }
                                    }
```

```
                                }
                            }
                        }
                    }
                }
            }
        }
        if (cont >= 1) {
            ok = true;
        }
        return ok;
    }


    public boolean moverBaldosa(int numeroFilaOrigen, int numeroColumnaOrigen, int numeroFilaDestino, int
numeroColumnaDestino, int numeroFilaOFicha, int numeroColumnaOFicha, Jugador jugadorJugando) {
        boolean ok = false;
        int largo = partidaActual().getTablero().getTableroBaldosas().length;
        int ancho = partidaActual().getTablero().getTableroBaldosas()[0].length;
        if (partidaActual().getTablero().coordenadaValida(numeroFilaOrigen, numeroColumnaOrigen)) {
            if (partidaActual().getTablero().coordenadaValida(numeroFilaDestino, numeroColumnaDestino)) {
                if
(partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen] == true) {
                    for (int i = 0; i < 10; i++) {


                    }
                    if
(partidaActual().getTablero().getTableroBaldosas()[numeroFilaDestino][numeroColumnaDestino] == false) {
                        if (baldosaConDosAlLado(numeroFilaOrigen, numeroColumnaOrigen)) {
                            if (nuevaUbicacionBaldosa(numeroFilaDestino, numeroColumnaDestino)) {
                                if (validarFichaConBaldosa(numeroFilaDestino, numeroColumnaDestino,
numeroFilaOFicha, numeroColumnaOFicha, jugadorJugando)) {
```

```
partidaActual().getTablero().getTableroBaldosas()[numeroFilaDestino][numeroColumnaDestino] = true;


partidaActual().getTablero().getTableroBaldosas()[numeroFilaOrigen][numeroColumnaOrigen] = false;

                                    if (saltaFicha(numeroFilaOFicha, numeroColumnaOFicha,
numeroFilaDestino, numeroColumnaDestino, jugadorJugando)) {

                                        ok = true;

                                    } else {

                                        moverFicha(numeroFilaOFicha, numeroColumnaOFicha,
numeroFilaDestino, numeroColumnaDestino, jugadorJugando);

                                        ok = true;

                                    }

                                }

                            }

                        }

                    }

                }

            }

        }

        return ok;

    }


    public boolean validarDimensionesMatriz(int dimension) {

        boolean esValida;

        switch (dimension) {

            case 1:

                esValida = true;

                break;

            case 2:

                esValida = true;

                break;

            case 3:
```

```
                esValida = true;
                break;
            default:
                esValida = false;
        }
        return esValida;
    }
}
```

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Clase Tablero

```
package Dominio;


import java.util.Random;


public class Tablero {


    private Ficha[][] tableroFichas;

    private boolean[][] tableroBaldosas;


    public Ficha[][] getTableroFichas() {

        return tableroFichas;

    }


    public void setTableroFichas(Ficha[][] tableroFichas) {

        this.tableroFichas = tableroFichas;

    }


    public boolean[][] getTableroBaldosas() {

        return tableroBaldosas;

    }


    public void setTableroBaldosas(boolean[][] tableroBaldosas) {

        this.tableroBaldosas = tableroBaldosas;

    }


    public Tablero(Ficha[][] tabF, boolean[][] tabB) {

        this.setTableroFichas(tabF);

        this.setTableroBaldosas(tabB);

    }
```

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

```
public Tablero() {
    Ficha[][] tFichas = new Ficha[12][13];
    this.setTableroFichas(tFichas);
    boolean[][] tBaldosas = new boolean[12][13];
    this.setTableroBaldosas(tBaldosas);
    for (int i = 4; i < 8; i++) {
        for (int j = 4; j < 9; j++) {
            this.getTableroBaldosas()[i][j] = (true);
        }
    }
}

public boolean coordenadaValida(int filaDestino, int columnaDestino) {
    boolean ok = false;
    if ((filaDestino >= 0 && filaDestino <= 11) && (columnaDestino >= 0 && columnaDestino <= 12)) {
        ok = true;
    }
    return ok;
}

public boolean hayBaldosa(int numeroFila, int numeroColumna) {
    return this.getTableroBaldosas()[numeroFila][numeroColumna];
}

public boolean hayBaldosaArriba(int numeroFila, int numeroColumna) {
    if (numeroFila == 0) {
        return false;
    } else {
        return this.getTableroBaldosas()[numeroFila - 1][numeroColumna];
```

```
    }
}

public boolean hayBaldosaIzquierda(int numeroFila, int numeroColumna) {
    if (numeroColumna == 0) {
        return false;
    } else {
        return this.getTableroBaldosas()[numeroFila][numeroColumna - 1];
    }
}

public boolean hayBaldosaArribaIzquierda(int numeroFila, int numeroColumna) {
    if (numeroFila == 0 || numeroColumna == 0) {
        return false;
    } else {
        boolean[][] mat = this.getTableroBaldosas();
        return mat[numeroFila - 1][numeroColumna - 1];
    }
}

public boolean hayFicha(int numeroFila, int numeroColumna) {
    return (this.getTableroFichas()[numeroFila][numeroColumna] != null);
}

public void precargaFija() {
    this.getTableroFichas()[4][4] = new Ficha(4, 4, true, "B");
    this.getTableroFichas()[4][6] = new Ficha(4, 6, true, "B");
    this.getTableroFichas()[5][7] = new Ficha(5, 7, true, "B");
    this.getTableroFichas()[6][6] = new Ficha(6, 6, true, "B");
    this.getTableroFichas()[6][8] = new Ficha(6, 8, true, "B");
```

```
        this.getTableroFichas()[7][4] = new Ficha(7, 4, true, "B");
        this.getTableroFichas()[4][5] = new Ficha(4, 5, false, "R");
        this.getTableroFichas()[5][4] = new Ficha(5, 4, false, "R");
        this.getTableroFichas()[5][6] = new Ficha(5, 6, false, "R");
        this.getTableroFichas()[6][4] = new Ficha(6, 4, false, "R");
        this.getTableroFichas()[6][7] = new Ficha(6, 7, false, "R");
        this.getTableroFichas()[7][5] = new Ficha(7, 5, false, "R");
    }

    public void precargaAzar() {
        int i = 1;
        Random rnd = new Random();
        while (i < 7) {
            int fila = (int) (rnd.nextDouble() * 4 + 4);
            int columna = (int) (rnd.nextDouble() * 5 + 4);
            if (tableroFichas[fila][columna] == null) {
                tableroFichas[fila][columna] = new Ficha(fila, columna, true, "B");
                i++;
            }
        }
        while (i < 13) {
            int fila = (int) (rnd.nextDouble() * 4 + 4);
            int columna = (int) (rnd.nextDouble() * 5 + 4);
            if (tableroFichas[fila][columna] == null) {
                tableroFichas[fila][columna] = new Ficha(fila, columna, false, "R");
                i++;
            }
        }
    }
```

```
    public boolean agregarFicha(int fila, int columna, String color) {
        boolean ok = false;
        if (this.coordenadaValida(fila, columna)) {
            if (hayBaldosa(fila, columna) && !hayFicha(fila, columna)) {
                this.getTableroFichas()[fila][columna] = new Ficha(fila, columna, true, color);
                ok = true;
            }
        }
        return ok;
    }

    public String getFicha(int f, int c) {
        return tableroFichas[f][c].getLetra();
    }
}
```

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Clase Partida

```
package Dominio;


import java.util.ArrayList;


public class Partida {


    private Tablero tablero;

    private Jugador jGanador;

    private Jugador jPerdedor;

    private ArrayList<Jugador> listaJugadores;


    public Tablero getTablero() {

        return tablero;

    }


    public void setTablero(Tablero unTablero) {

        tablero = unTablero;

    }


    public Jugador getJGanador() {

        return jGanador;

    }


    public void setJGanador(Jugador unJGanador) {

        jGanador = unJGanador;

    }


    public Jugador getJPerdedor() {

        return jPerdedor;
```

```
    }

    public void setJPerdedor(Jugador unJPerdedor) {
        this.jPerdedor = unJPerdedor;
    }

    public ArrayList<Jugador> getListaJugadores() {
        return listaJugadores;
    }

    public void setListaJugadores(ArrayList<Jugador> listaJugadores) {
        this.listaJugadores = listaJugadores;
    }

    public Partida() {
        Jugador unJ = new Jugador();
        Tablero unT = new Tablero();
        this.setTablero(unT);
        this.setJGanador(unJ);
        this.setJPerdedor(unJ);
        this.listaJugadores = new ArrayList<>();
    }

    public Partida(Tablero unTablero, Jugador unJGanador, Jugador unJPerdedor, ArrayList<Jugador>
listaJugadores) {
        this.setTablero(unTablero);
        this.setJGanador(unJGanador);
        this.setJPerdedor(unJPerdedor);
        listaJugadores = new ArrayList<>();
    }
```

```
public void precargarFija() {
    tablero.precargaFija();
}

public void precargaAzar() {
    tablero.precargaAzar();
}

public void partidasJugadas(Jugador unJ1, Jugador unJ2) {
    unJ1.setPartidasJugadas(unJ1.getPartidasJugadas() + 1);
    unJ2.setPartidasJugadas(unJ2.getPartidasJugadas() + 1);
}

public void partidasGanadas(Jugador unJPerdedor, Jugador unJGanador) {
    this.setJPerdedor(unJPerdedor);
    this.setJGanador(unJGanador);
    unJPerdedor.setPartidasPerdidas(unJPerdedor.getPartidasPerdidas() + 1);
    unJGanador.setPartidasGanadas(unJGanador.getPartidasGanadas() + 1);
}

public void partidasEmpatadas(Jugador unJ1, Jugador unJ2) {
    unJ1.setPartidasEmpatadas(unJ1.getPartidasEmpatadas() + 1);
    unJ1.setPartidasEmpatadas(unJ2.getPartidasEmpatadas() + 1);
}

public static int conversorFila(String coordenadas) {
    int numeroFila = coordenadas.charAt(0) - 65;
    return numeroFila;
}
```

```
public static int conversorColumna(String coordenadas) {
    int numeroColumna;
    if (coordenadas.length() == 1) {
        numeroColumna = coordenadas.charAt(0) - 49;
    } else {
        try {
            numeroColumna = Integer.parseInt(coordenadas);
        } catch (NumberFormatException ex) {
            numeroColumna = -1;
        }
    }
    return numeroColumna;
}

public int ingresoJuego(String accion) {
    int seRealiza = 0;
    String h1 = accion.toUpperCase();
    String h2 = h1.substring(0, 1);
    if (h2.equals("M") || h2.equals("S") || h2.equals("B")) {
        String h = h1.substring(0, 2);
        if (h.equals("M ") && (accion.length() >= 5 && accion.length() <= 8)) {
            seRealiza = 1;
        } else {
            if (h.equals("S ") && (accion.length() >= 5 && accion.length() <= 8)) {
                seRealiza = 2;
            } else {
                if (h.equals("B ") && (accion.length() > 7 && accion.length() < 12)) {
                    seRealiza = 3;
                }
```

```
            }
        }
    } else {
        if (h2.equals("X") || h2.equals("E")) {
            String h = h1.substring(0, 1);
            if (h.equals("X") && accion.length() == 1) {
                seRealiza = 4;
            } else {
                if (h.equals("E") && accion.length() == 1) {
                    seRealiza = 5;
                }
            }
        } else {
            seRealiza = 6;
        }
    }
    return seRealiza;
}

public boolean partidaGanada(int fila, int columna, int fichasInv) {
    int cantFichas = 1;
    int cantFichasInv = 0;
    boolean esInv = false;
    if (this.tablero.getTableroFichas()[fila][columna].getLetra().equals("r")
            || this.tablero.getTableroFichas()[fila][columna].getLetra().equals("b")) {
        esInv = true;
    }
    int i = fila + 1;
    while ((i < 12) && (this.tablero.hayFicha(i, columna))
```

```
                &&
this.tablero.getTableroFichas()[i][columna].getLetra().toUpperCase().equals(this.tablero.getTableroFichas
()[fila][columna].getLetra().toUpperCase())) {


            if (this.tablero.getTableroFichas()[i][columna].getLetra().equals("r")

                    || this.tablero.getTableroFichas()[i][columna].getLetra().equals("b")) {

                cantFichasInv++;

            }

            cantFichas++;

            i++;

        }

        i = fila - 1;

        while ((i >= 0) && (this.tablero.hayFicha(i, columna))

                &&
this.tablero.getTableroFichas()[i][columna].getLetra().toUpperCase().equals(this.tablero.getTableroFichas
()[fila][columna].getLetra().toUpperCase())) {


            if (this.tablero.getTableroFichas()[i][columna].getLetra().equals("r")

                    || this.tablero.getTableroFichas()[i][columna].getLetra().equals("b")) {

                cantFichasInv++;

            }

            cantFichas++;

            i--;

        }

        if ((cantFichas >= 4) && (cantFichasInv >= fichasInv)) {

            return true;

        }

        cantFichas = 1;

        if (esInv) {

            cantFichasInv = 1;

        } else {

            cantFichasInv = 0;
```

```
        }
        int j = columna + 1;
        while ((j < 13) && (this.tablero.hayFicha(fila, j))
                &&
this.tablero.getTableroFichas()[fila][j].getLetra().toUpperCase().equals(this.tablero.getTableroFichas()[
fila][columna].getLetra().toUpperCase())) {

            if (this.tablero.getTableroFichas()[fila][j].getLetra().equals("r")
                    || this.tablero.getTableroFichas()[fila][j].getLetra().equals("b")) {
                cantFichasInv++;
            }
            cantFichas++;
            j++;
        }
        j = columna - 1;
        while ((j >= 0) && (this.tablero.hayFicha(fila, j))
                &&
this.tablero.getTableroFichas()[fila][j].getLetra().toUpperCase().equals(this.tablero.getTableroFichas()[
fila][columna].getLetra().toUpperCase())) {

            if (this.tablero.getTableroFichas()[fila][j].getLetra().equals("r")
                    || this.tablero.getTableroFichas()[fila][j].getLetra().equals("b")) {
                cantFichasInv++;
            }
            cantFichas++;
            j--;
        }
        if ((cantFichas >= 4) && (cantFichasInv >= fichasInv)) {
            return true;
        }
        cantFichas = 1;
        if (esInv) {
```

```
            cantFichasInv = 1;

        } else {

            cantFichasInv = 0;

        }

        i = fila + 1;

        j = columna + 1;

        while ((j < 13) && (i < 12) && (this.tablero.hayFicha(i, j))

                &&
this.tablero.getTableroFichas()[i][j].getLetra().toUpperCase().equals(this.tablero.getTableroFichas()[fil
a][columna].getLetra().toUpperCase())) {

            cantFichas++;


            if (this.tablero.getTableroFichas()[i][j].getLetra().equals("r")

                    || this.tablero.getTableroFichas()[i][j].getLetra().equals("b")) {

                cantFichasInv++;

            }

            j++;

            i++;

        }

        j = columna - 1;

        i = fila - 1;

        while ((j >= 0) && (i >= 0) && (this.tablero.hayFicha(i, j))

                &&
this.tablero.getTableroFichas()[i][j].getLetra().toUpperCase().equals(this.tablero.getTableroFichas()[fil
a][columna].getLetra().toUpperCase())) {

            cantFichas++;


            if (this.tablero.getTableroFichas()[i][j].getLetra().equals("r")

                    || this.tablero.getTableroFichas()[i][j].getLetra().equals("b")) {

                cantFichasInv++;

            }

            j--;
```

```
            i--;
        }
        if ((cantFichas >= 4) && (cantFichasInv >= fichasInv)) {
            return true;
        }

        cantFichas = 1;
        if (esInv) {
            cantFichasInv = 1;
        } else {
            cantFichasInv = 0;
        }
        i = fila - 1;
        j = columna + 1;
        while ((j < 13) && (i >= 0) && (this.tablero.hayFicha(i, j))
                &&
this.tablero.getTableroFichas()[i][j].getLetra().toUpperCase().equals(this.tablero.getTableroFichas()[fil
a][columna].getLetra().toUpperCase())) {
            cantFichas++;

            if (this.tablero.getTableroFichas()[i][j].getLetra().equals("r")
                    || this.tablero.getTableroFichas()[i][j].getLetra().equals("b")) {
                cantFichasInv++;
            }
            j++;
            i--;
        }
        j = columna - 1;
        i = fila + 1;
        while ((j >= 0) && (i < 12) && (this.tablero.hayFicha(i, j))
```

```
                &&
this.tablero.getTableroFichas()[i][j].getLetra().toUpperCase().equals(this.tablero.getTableroFichas()[fil
a][columna].getLetra().toUpperCase())) {

            cantFichas++;



            if (this.tablero.getTableroFichas()[i][j].getLetra().equals("r")

                    || this.tablero.getTableroFichas()[i][j].getLetra().equals("b")) {

                cantFichasInv++;

            }

            j--;

            i++;

        }

        return ((cantFichas >= 4) && (cantFichasInv >= fichasInv));

    }

}
```

Clase Jugador

```
package Dominio;


public class Jugador implements Comparable<Jugador> {


    private Ficha ficha;

    private String nombre;

    private int edad;

    private String alias;

    private int partidasJugadas;

    private int partidasPerdidas;

    private int partidasGanadas;

    private int partidasEmpatadas;


    public Ficha getFicha() {

        return ficha;

    }


    public void setFicha(Ficha unaFicha) {

        this.ficha = unaFicha;

    }


    public String getNombre() {

        return nombre;

    }


    public void setNombre(String unNombre) {

        this.nombre = unNombre;

    }
```

```
public int getEdad() {
    return edad;
}

public void setEdad(int unaEdad) {
    this.edad = unaEdad;
}

public String getAlias() {
    return alias;
}

public void setAlias(String unAlias) {
    this.alias = unAlias;
}

public int getPartidasJugadas() {
    return partidasJugadas;
}

public void setPartidasJugadas(int unaPartidasJugadas) {
    this.partidasJugadas = unaPartidasJugadas;
}

public int getPartidasPerdidas() {
    return partidasPerdidas;
}

public void setPartidasPerdidas(int unaPartidasPerdidas) {
    this.partidasPerdidas = unaPartidasPerdidas;
```

```
    }

    public int getPartidasGanadas() {
        return partidasGanadas;
    }

    public void setPartidasGanadas(int unaPartidasGanadas) {
        this.partidasGanadas = unaPartidasGanadas;
    }

    public int getPartidasEmpatadas() {
        return partidasEmpatadas;
    }

    public void setPartidasEmpatadas(int unaPartidaEmpatada) {
        this.partidasEmpatadas = unaPartidaEmpatada;
    }

    public Jugador() {
        this.setNombre("Sin nombre");
        this.setEdad(0);
        this.setAlias("Sin alias");
        this.setPartidasJugadas(0);
        this.setPartidasPerdidas(0);
        this.setPartidasGanadas(0);
    }

    public Jugador(Ficha unaFicha, String unNombre, int unaEdad, String unAlias, int unaPartidasJugadas,
int unaPartidasPerdidas, int unaPartidasGanadas) {
        this.setFicha(unaFicha);
```

```
        this.setNombre(unNombre);
        this.setEdad(unaEdad);
        this.setAlias(unAlias);
        this.setPartidasJugadas(unaPartidasJugadas);
        this.setPartidasPerdidas(unaPartidasPerdidas);
        this.setPartidasGanadas(unaPartidasGanadas);
    }

    @Override
    public String toString() {
        return "\nNombre del jugador :" + this.getNombre()
                + "\nEdad del jugador :" + this.getEdad()
                + "\nAlias del jugador :" + this.getAlias()
                + "\nPartidas jugadas :" + this.getPartidasJugadas()
                + "\nPartidas perdidas :" + this.getPartidasPerdidas()
                + "\nPartidas ganadas :" + this.getPartidasGanadas();
    }

    @Override
    public boolean equals(Object obj) {
        Jugador unJugador = ((Jugador) obj);
        return this.getAlias().equals(unJugador.getAlias());
    }

    @Override
    public int compareTo(Jugador unJugador) {
        int retorno = unJugador.getPartidasGanadas() - this.getPartidasGanadas();
        if (retorno == 0) {
            retorno = this.getAlias().compareTo(unJugador.getAlias());
        }
```

```
        return retorno;

    }

}
```

Andres Lacaño (158910)
Rodolfo Agustín Silva (187061)

Clase Ficha

```
package Dominio;


public class Jugador implements Comparable<Jugador> {


    private Ficha ficha;

    private String nombre;

    private int edad;

    private String alias;

    private int partidasJugadas;

    private int partidasPerdidas;

    private int partidasGanadas;

    private int partidasEmpatadas;


    public Ficha getFicha() {

        return ficha;

    }


    public void setFicha(Ficha unaFicha) {

        this.ficha = unaFicha;

    }


    public String getNombre() {

        return nombre;

    }


    public void setNombre(String unNombre) {

        this.nombre = unNombre;

    }
```

```
public int getEdad() {
    return edad;
}

public void setEdad(int unaEdad) {
    this.edad = unaEdad;
}

public String getAlias() {
    return alias;
}

public void setAlias(String unAlias) {
    this.alias = unAlias;
}

public int getPartidasJugadas() {
    return partidasJugadas;
}

public void setPartidasJugadas(int unaPartidasJugadas) {
    this.partidasJugadas = unaPartidasJugadas;
}

public int getPartidasPerdidas() {
    return partidasPerdidas;
}

public void setPartidasPerdidas(int unaPartidasPerdidas) {
    this.partidasPerdidas = unaPartidasPerdidas;
```

```
    }

    public int getPartidasGanadas() {
        return partidasGanadas;
    }

    public void setPartidasGanadas(int unaPartidasGanadas) {
        this.partidasGanadas = unaPartidasGanadas;
    }

    public int getPartidasEmpatadas() {
        return partidasEmpatadas;
    }

    public void setPartidasEmpatadas(int unaPartidaEmpatada) {
        this.partidasEmpatadas = unaPartidaEmpatada;
    }

    public Jugador() {
        this.setNombre("Sin nombre");
        this.setEdad(0);
        this.setAlias("Sin alias");
        this.setPartidasJugadas(0);
        this.setPartidasPerdidas(0);
        this.setPartidasGanadas(0);
    }

    public Jugador(Ficha unaFicha, String unNombre, int unaEdad, String unAlias, int unaPartidasJugadas,
int unaPartidasPerdidas, int unaPartidasGanadas) {
        this.setFicha(unaFicha);
```

```
        this.setNombre(unNombre);
        this.setEdad(unaEdad);
        this.setAlias(unAlias);
        this.setPartidasJugadas(unaPartidasJugadas);
        this.setPartidasPerdidas(unaPartidasPerdidas);
        this.setPartidasGanadas(unaPartidasGanadas);
    }

    @Override
    public String toString() {
        return "\nNombre del jugador :" + this.getNombre()
                + "\nEdad del jugador :" + this.getEdad()
                + "\nAlias del jugador :" + this.getAlias()
                + "\nPartidas jugadas :" + this.getPartidasJugadas()
                + "\nPartidas perdidas :" + this.getPartidasPerdidas()
                + "\nPartidas ganadas :" + this.getPartidasGanadas();
    }

    @Override
    public boolean equals(Object obj) {
        Jugador unJugador = ((Jugador) obj);
        return this.getAlias().equals(unJugador.getAlias());
    }

    @Override
    public int compareTo(Jugador unJugador) {
        int retorno = unJugador.getPartidasGanadas() - this.getPartidasGanadas();
        if (retorno == 0) {
            retorno = this.getAlias().compareTo(unJugador.getAlias());
        }
```

```
        return retorno;

    }

}
```