

Universidad ORT

Escuela de Ingeniería

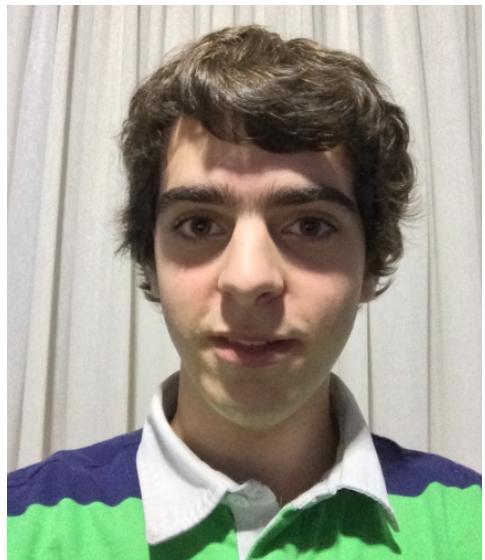
Obligatorio 1

Programación 2

Autores:

Fernando Agustín Hernández Gobertti (173631) – Ingeniería en Telecomunicaciones

Sebastián Effa Gallego (193248) – Ingeniería en Electrónica



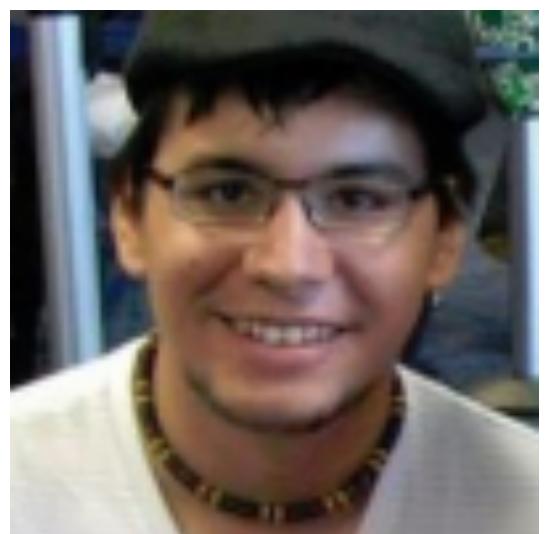
Tutores:

Nicolás Fornaro (Teórico)

Matías Núñez (Práctico)

Fecha Límite de Entrega:

21 de Octubre del 2014



Índice del Trabajo Realizado:

Casos de Uso	3
<i>Pedido a: Ingreso de jugador</i>	3
<i>Pedido c: Consulta de jugadores</i>	5
Diagrama de Clases (UML: Unified Modelling Language)	6
Datos de Prueba	7
Folleto del juego para promoción del producto	21, 22
<i>Página 1. PORTADA</i>	21
<i>Página 2. INTERIOR</i>	22
Modificaciones a la lógica del juego	23
1. <i>Rotación del Panel Móvil</i>	23
2. <i>Eliminación temporal del Panel Móvil</i>	24
Clases utilizadas (y su respectivo código)	25
Clase Interfaz	60
Clase Juego	32
Clase Jugador	44
Clase Perfección	25
Clase Tablero	47

Casos de Uso

Pedido a: *Ingreso de jugador (indicando nombre, edad y alias de cada uno).*

Actor: Operador / Usuario / Jugador

Caso de Uso: Agregar un jugador

Especificación del Caso de Uso:

Curso Normal:

- El sistema le solicita que ingrese el nombre al Jugador 1.
- El usuario ingresa el nombre del Jugador 1.
- El sistema pide el ingreso de la edad del Jugador 1.
- El usuario digita la edad del Jugador 1.
- El sistema requiere el alias del Jugador 1.
- El usuario entra el alias del Jugador 1.
- El proceso se repite idénticamente para el Jugador 2.
- El sistema muestra una lista con los datos de los dos Jugadores ingresados y solicita al usuario confirmación de datos.
- Una vez confirmado, procederá al Pedido b descripto en la letra del Obligatorio.

Cursos Alternativos:

- ✓ El usuario ingresa una cadena de caracteres en vez de un entero al solicitársele la edad de alguno de los Jugadores.

El sistema despliega un mensaje de advertencia o error y le solicita al usuario nuevamente el dato.

- ✓ El usuario registra ambos jugadores y resultan tener el mismo nombre, edad y alias (ser el mismo jugador).

El sistema muestra un mensaje indicándole al usuario su error y le solicita un nuevo ingreso de datos de ambos jugadores.

- ✓ El usuario recapacita sobre algún dato erróneo ingresado y decide editar los datos ingresados.

El sistema elimina todo lo anteriormente registrado y solicita un nuevo ingreso de datos al usuario.

- ✓ El usuario ingresa datos que coinciden con los de un Jugador previamente ingresado.

El sistema despliega los puntos guardados por el Jugador ingresado y así el usuario podrá continuar con su progreso en el juego.

- ✓ El usuario no ingresa una de las opciones indicadas como posibles.

El sistema despliega un mensaje de advertencia indicándole al usuario su error y le solicita su ingreso nuevamente.

Pedido c: Consulta de jugadores (mostrando la lista de jugadores, ordenada alfabéticamente e incluyendo la cantidad de partidas jugadas y ganadas)

Actor: Operador / Usuario / Jugador

Caso de Uso: Consulta de Jugadores

Especificación del Caso de Uso:

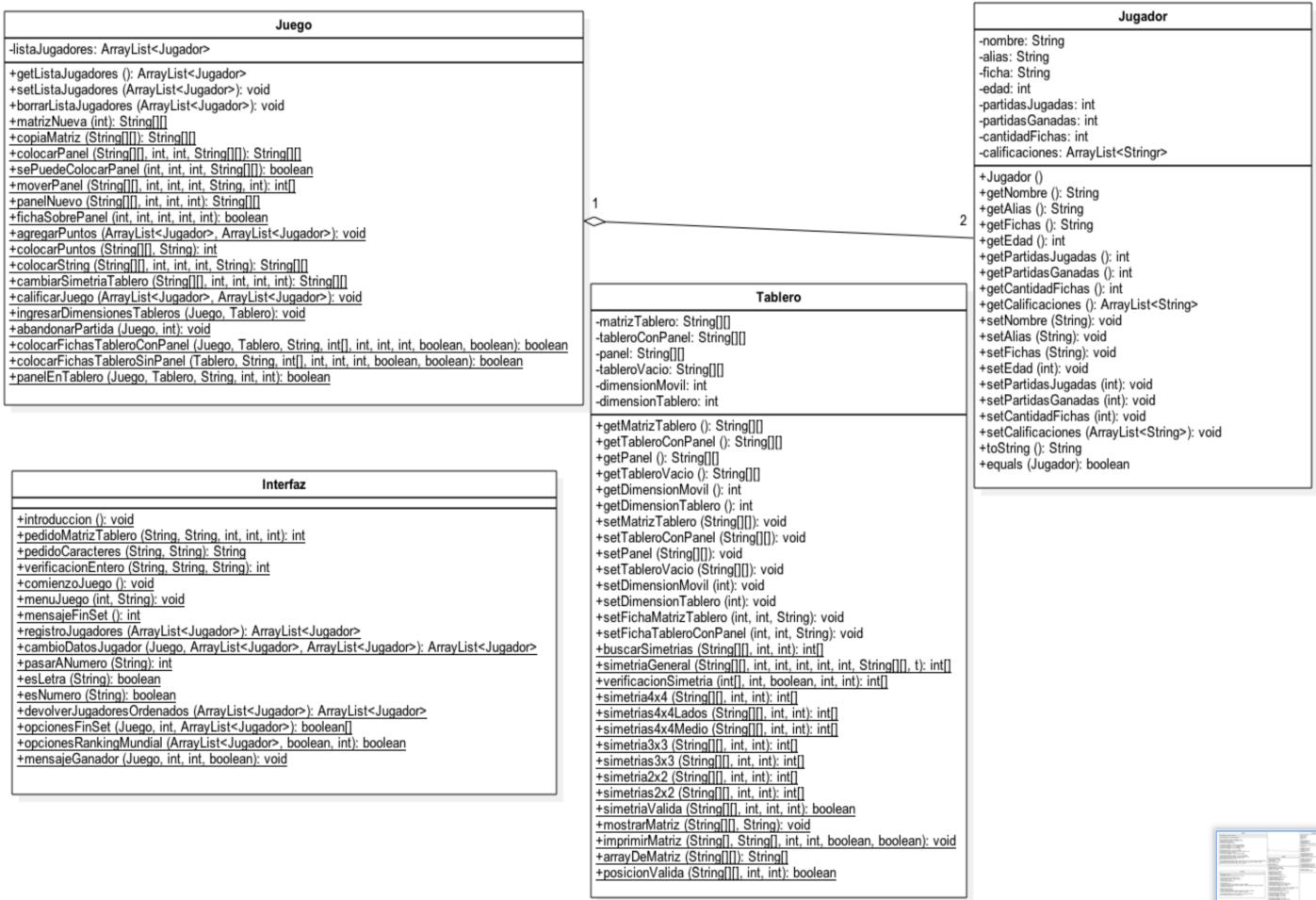
Curso Normal:

- Al finalizar la partida y anunciar el ganador (o empate, según en caso), el sistema despliega un pequeño menú donde le brinda al usuario la opción de visualizar la lista ordenada de jugadores que hayan alguna vez jugado al juego.
 - La lista de Jugadores mostrada por el sistema posee información relativa a los datos ingresados por el usuario, el último tipo de ficha jugada (Blancas o Negras) por el Jugador, las partidas jugadas y las partidas ganadas (puntos) por el mismo al igual que una lista de calificaciones respectivas a las agregadas por ese mismo jugador en distintas ocasiones de juego (distintas partidas).
 - A continuación, el sistema despliega otro menú en el que el usuario se encuentra posibilitado a calificar el juego (guardando su comentario para su debida muestra en partidas futuras), jugar nuevamente o abandonar el juego.

Cursos Alternativos:

- ✓ El usuario no ingresa una de las opciones indicadas posibles o disponibles.
El sistema despliega un mensaje de error evidenciando el error e insta al usuario un nuevo ingreso de la opción deseada.

Diagrama de Clases (UML: *Unified Modelling Language*)



Nota: Creación en *StarUML* Versión 2.0.0-beta10 para MAC OS X

Datos de Prueba

DATO INGRESADO	DATO ESPERADO	RESULTADO
Se ingresa el nombre del jugador 1 Juan	<p>Lo acepta y pide el siguiente dato:</p> <p>PARTIDA 1</p> <p>JUGADOR 1_____</p> <p>Jugará con las fichas BLANCAS</p> <p>Ahora ingrese su NOMBRE</p> <p>Juan</p> <p>NOMBRE aceptado.</p> <p>Por favor, ingrese su EDAD a continuación</p> <p> </p>	OK
Se ingresa la edad del jugador 1 en enteros 23	<p>Lo acepta y pide el siguiente dato:</p> <p>Por favor, ingrese su EDAD a continuación</p> <p>23</p> <p>Edad recibida</p> <p>Ingrese su ALIAS</p> <p> </p>	OK
Se ingresa la edad del jugador 1 como un String (cadena de caracteres) Veinte	<p>Se informa al usuario y se vuelve a pedir el dato:</p> <p>Por favor, ingrese su EDAD a continuación</p> <p>Veinte</p> <p>Debe ingresar su edad en enteros</p> <p>Por favor, ingrese su EDAD a continuación</p>	OK

<p>Se ingresa una edad irreal o imposible (mayor a 120) en números enteros 2343</p>	<p>Se informa al usuario y se vuelve a pedir el dato:</p> <p>Por favor, ingrese su EDAD a continuación 2343</p> <p>No puede engañar a este programa. Debe ingresar su VERDADERA edad</p> <p>Por favor, ingrese su EDAD a continuación </p>	<p>OK</p>
<p>Se ingresa el alias del jugador 1 Juan23</p>	<p>Lo acepta y pide el siguiente dato:</p> <p>Ingresé su ALIAS Juan23</p> <p>ALIAS ingresado.</p>	<p>OK</p>
<p>Se repite el proceso con los datos del jugador 2</p>	<p>Igual que con jugador 1.</p>	<p>OK</p>
<p>Ambos jugadores concuerdan en sus datos de Nombre, Edad y Alias</p>	<p>Se informa al jugador del suceso (un jugador no puede jugar contra sí mismo) y se solicitan los datos nuevamente:</p> <p>JUGADOR 2 _____</p> <p>Jugará con las fichas NEGRAS</p> <p>Ahora ingrese su NOMBRE Juan</p> <p>NOMBRE aceptado.</p> <p>Por favor, ingrese su EDAD a continuación 23</p> <p>Edad recibida</p> <p>Ingresé su ALIAS Juan23</p> <p>ALIAS ingresado.</p> <p>No puede jugar contra si mismo. Ingrese nuevamente</p>	<p>OK</p>
<p>El / Los jugador/es ingresados ya jugaron previamente</p>	<p>Se despliega la información de partidas jugadas y ganadas previamente guardadas. Si no jugaron ninguna, las partidas ganadas y jugadas de ambos jugadores resulta 0.</p>	<p>OK</p>

	<p>Muchas gracias. Sus datos están siendo procesados.</p> <p>Los datos ingresados fueron:</p> <pre>[PARTIDAS JUGADAS: 1 FICHAS ÚLTIMAS JUGADAS: Blancas Nombre: Juan Edad: 18 Alias: Juan171 PARTIDAS GANADAS: 0 CALIFICACIÓN/ES: [Ninguna Calificación] , PARTIDAS JUGADAS: 1 FICHAS ÚLTIMAS JUGADAS: Negras Nombre: Fernando Edad: 18 Alias: SagitariusWithin PARTIDAS GANADAS: 1 CALIFICACIÓN/ES: [Ninguna Calificación]</pre> <p>]</p> <p>Están de acuerdo con los datos ingresados?</p> <p>1- Si 2- No</p>	
Se muestra un menú con dos opciones (1 o 2) para confirmar los datos	<p>Si se ingresa la opción 1, el programa sigue. Si se ingresa la opción 2, se vuelven a solicitar los datos.</p> <p>Están de acuerdo con los datos ingresados? 1- Si 2- No 1</p> <p>Datos procesados.</p> <p>Están de acuerdo con los datos ingresados? 1- Si 2- No 2</p> <p>Les solicitaremos nuevamente sus datos</p> <p>JUGADOR 1_____</p> <p>Jugará con las fichas BLANCAS</p> <p>Ahora ingrese su NOMBRE</p>	OK
Al desplegarse el menú se ingresan opciones incoherentes o Strings en vez de enteros 25 Veinticinco	<p>Se vuelve a mostrar el menú y se solicita nuevamente la opción.</p>	OK

	<p>Están de acuerdo con los datos ingresados? 1- Si 2- No Veinte</p> <p>Debe ingresar enteros 1 o 2.</p> <p>Están de acuerdo con los datos ingresados? 1- Si 2- No 23</p> <p>Debe ingresar opcion 1 o 2 únicamente. Disculpe las molestias.</p> <p>Los datos ingresados fueron: [PARTIDAS JUGADAS: 0 FICHAS ÚLTIMAS JUGADAS: Blancas Nombre: sdfsd Edad: 2 Alias: sdfsd PARTIDAS GANADAS: 0 CALIFICACIÓN/ES: [Ninguna Calificación] , PARTIDAS JUGADAS: 0 FICHAS ÚLTIMAS JUGADAS: Negras Nombre: sdfs Edad: 2 Alias: sdfsd PARTIDAS GANADAS: 0 CALIFICACIÓN/ES: [Ninguna Calificación]]</p> <p>Están de acuerdo con los datos ingresados? 1- Si 2- No</p>	
Se ingresa la dimensión del tablero y de panel de acuerdo a las normas. Dimensión tablero: 8 Dimensión panel: 3	<p>Lo acepta y comienza el juego:</p> <p>Cuál es la dimensión del tablero en la que desea jugar? Nota: Se aceptarán tableros de 4, 6 u 8 piezas de lado solamente. 8</p> <p>Dimension asignada.</p> <p>Cuál es la dimensión del tablero en la que desea jugar? Nota: Se aceptarán tableros de 2, 3 o 4 piezas de lado solamente. 3</p> <p>Dimension asignada.</p>	OK
Se ingresa la dimensión del tablero y del panel sin estar de acuerdo a las normas. Dimensión tablero: 9 Dimensión panel: 5	<p>Se informa al usuario y se vuelve a pedir el dato:</p> <p>TABLERO PRINCIPAL_____</p> <p>Cuál es la dimensión del tablero en la que desea jugar? Nota: Se aceptarán tableros de 4, 6 u 8 piezas de lado solamente. 9</p> <p>Debe ingresar valores 4, 6 u 8 refiriendo a la dimension que desee para el tablero.</p> <p>Cuál es la dimensión del tablero en la que desea jugar? Nota: Se aceptarán tableros de 4, 6 u 8 piezas de lado solamente. </p>	OK

	<p>TABLERO MÓVIL_____</p> <p>Cuál es la dimensión del tablero en la que desea jugar? Nota: Se aceptarán tableros de 2, 3 o 4 piezas de lado solamente. 5</p> <p>Debe ingresar valores 2, 3 o 4 refiriendo a la dimensión que desee para el tablero.</p>																																																																																													
Se ingresa la dimensión del tablero y del panel como String Ocho	<p>Se informa al usuario y se vuelve a pedir el dato:</p> <p>TABLERO PRINCIPAL_____</p> <p>Cuál es la dimensión del tablero en la que desea jugar? Nota: Se aceptarán tableros de 4, 6 u 8 piezas de lado solamente. Ocho</p> <p>Debe ingresar 4, 6 u 8 según sean las dimensiones deseadas. Ingrese nuevamente, por favor.</p>	OK																																																																																												
Se ingresa una dimensión del tablero de 4.	<p>Se le corresponde automáticamente con un panel de 2x2:</p> <p>Cuál es la dimensión del tablero en la que desea jugar? Nota: Se aceptarán tableros de 4, 6 u 8 piezas de lado solamente. 4</p> <p>Dimension asignada.</p> <p>La dimensión única disponible del Panel Móvil para la dimensión del Tablero elegida es 2x2</p>	OK																																																																																												
Se ingresa una jugada en el formato estipulado. Colocar ficha: jc3	<p>Lo acepta y coloca la ficha en la posición indicada:</p> <div style="border: 1px solid black; padding: 10px;"> <p>Ingrese una jugada válida jc3</p> <p>Su ficha ha sido colocada.</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>1</td><td>2</td><td>3</td><td>4</td> </tr> <tr> <td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td> </tr> </table> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>A</td><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td> </tr> <tr> <td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td> </tr> </table> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>B</td><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td> </tr> <tr> <td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td> </tr> </table> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>C</td><td> </td><td> </td><td> </td><td>B</td><td> </td><td> </td> </tr> <tr> <td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td> </tr> </table> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>D</td><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td> </tr> <tr> <td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td> </tr> </table> </div>	1	2	3	4	+	-	+	-	+	-	+	-	+	-	+	-	A							+	-	+	-	+	-	+	-	+	-	+	-	B							+	-	+	-	+	-	+	-	+	-	+	-	C				B			+	-	+	-	+	-	+	-	+	-	+	-	D							+	-	+	-	+	-	+	-	+	-	+	-	OK
1	2	3	4																																																																																											
+	-	+	-	+	-	+	-	+	-	+	-																																																																																			
A																																																																																														
+	-	+	-	+	-	+	-	+	-	+	-																																																																																			
B																																																																																														
+	-	+	-	+	-	+	-	+	-	+	-																																																																																			
C				B																																																																																										
+	-	+	-	+	-	+	-	+	-	+	-																																																																																			
D																																																																																														
+	-	+	-	+	-	+	-	+	-	+	-																																																																																			

<p>La posición donde se quiere colocar la ficha ya contiene otra ficha o no existe.</p> <p>Colocar ficha: jc9 – jm3</p>	<p>Se informa al usuario y se vuelve a pedir la jugada.</p> <pre>Ingrese su intención a continuación y presione ENTER 1 2 3 4 +---+ +---+ +---+ +---+ A +---+ +---+ +---+ +---+ B +---+ +---+ +---+ +---+ C B +---+ +---+ +---+ +---+ D +---+ +---+ +---+ +---+</pre> <p>Ingrese una jugada válida jc3</p> <p>Su jugada no es válida. Ingrese nuevamente a continuación.</p> <p>OK</p> <pre>Ingrese su intención a continuación y presione ENTER 1 2 3 4 +---+ +---+ +---+ +---+ A +---+ +---+ +---+ +---+ B +---+ +---+ +---+ +---+ C B +---+ +---+ +---+ +---+ D +---+ +---+ +---+ +---+</pre> <p>Ingrese una jugada válida jc9</p> <p>Su jugada no es válida. Ingrese nuevamente a continuación.</p>
---	--

<p>Se ingresa una jugada contraria al formato estipulado. Colocar ficha: j33 – jcc – j3c</p>	<p>Se informa al usuario y se vuelve a pedir el dato:</p> <pre>Ingrese su intención a continuación y presione ENTER 1 2 3 4 +++ --- +++ A +++ --- +++ B +++ --- +++ C B +++ --- +++ D +++ --- +++ </pre> <p>Ingrese una jugada válida j33</p> <p>Su jugada no es válida. Ingrese nuevamente a continuación.</p>	OK
<p>Se ingresa una jugada correspondiente a mover el panel antes de haber colocado el panel</p>	<p>Se informa al usuario del suceso y se vuelve a pedir la jugada:</p> <pre>Ingrese su intención a continuación y presione ENTER 1 2 3 4 +++ --- +++ A +++ --- +++ B +++ --- +++ C B +++ --- +++ D +++ --- +++ </pre> <p>Ingrese una jugada válida md3</p> <p>ADVERTENCIA: El panel no se encuentra en el tablero. Debe ingresar una opción válida. Su jugada no es válida. Ingrese nuevamente a continuación.</p>	OK

<p>Se ingresa una jugada en el formato estipulado. Colocar panel: pb1</p>	<p>Lo acepta y coloca el panel en la posición indicada:</p> <pre>Ingrese una jugada válida pb1</pre> <table border="0"> <thead> <tr> <th></th> <th style="text-align: center;">1</th> <th style="text-align: center;">2</th> <th style="text-align: center;">3</th> <th style="text-align: center;">4</th> </tr> </thead> <tbody> <tr> <td>A</td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> <tr> <td>B</td> <td style="text-align: center;">#</td> <td style="text-align: center;">#</td> <td style="text-align: center;">#</td> <td style="text-align: center;">#</td> </tr> <tr> <td>C</td> <td style="text-align: center;">#</td> <td style="text-align: center;">#</td> <td style="text-align: center;">#</td> <td style="text-align: center;">#</td> </tr> <tr> <td>D</td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> </tbody> </table>		1	2	3	4	A					B	#	#	#	#	C	#	#	#	#	D					<p>OK</p>
	1	2	3	4																							
A																											
B	#	#	#	#																							
C	#	#	#	#																							
D																											
<p>Se ingresa una jugada contraria al formato estipulado. Colocar panel: p21 - pba</p>	<p>Se informa al usuario y se vuelve a pedir la jugada:</p> <pre>Ingrese su intención a continuación y presione ENTER 1 2 3 4</pre> <table border="0"> <thead> <tr> <th></th> <th style="text-align: center;">1</th> <th style="text-align: center;">2</th> <th style="text-align: center;">3</th> <th style="text-align: center;">4</th> </tr> </thead> <tbody> <tr> <td>A</td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> <tr> <td>B</td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> <tr> <td>C</td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;">B</td> </tr> <tr> <td>D</td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> </tbody> </table> <pre>Ingrese una jugada válida p21</pre> <p>Su jugada no es válida. Ingrese nuevamente a continuación.</p>		1	2	3	4	A					B					C				B	D					<p>OK</p>
	1	2	3	4																							
A																											
B																											
C				B																							
D																											

	<p>Si es posible realizar el movimiento, mueve el panel, de lo contrario, se vuelve a pedir la jugada:</p> <pre>Ingrese su intención a continuación y presione ENTER 1 2 3 4 +++ --- +++ --- +++ --- +++ --- +++ A # # # # # --- --- --- +++ B # # # # # # # # # # --- +++ --- +++ C # # # # # B # # # # # --- +++ --- +++ D +++ --- +++ --- +++ --- +++ --- +++</pre> <p>Ingrese una jugada válida md2</p> <pre>1 2 3 4 +++ --- +++ --- +++ --- +++ --- +++ --- +++ A +++ --- +++ --- # # # # # B # # # # +++ --- +++ --- # # # # # C # # # # +++ --- +++ --- # # # # # D +++ --- +++ --- +++ --- +++ --- +++</pre>	
<p>Se ingresa una jugada en el formato estipulado luego de colocar el panel.</p> <p>Mover panel: mi1</p>	<p>Se informa al usuario y se vuelve a pedir la jugada:</p>	<p>OK</p>

	<p>Ingrese su intención a continuación y presione ENTER</p> <table border="0"> <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table> <p>+++ --- +++ --- ++- --- +++ --- +++ --- +++</p> <table border="0"> <tr><td>A</td><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> <tr><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td></tr> </table> <table border="0"> <tr><td>B</td><td> </td><td> </td><td>#</td><td>#</td><td>#</td><td>#</td></tr> <tr><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td></tr> </table> <table border="0"> <tr><td>C</td><td> </td><td> </td><td>#</td><td>#</td><td>#</td><td>#</td></tr> <tr><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td></tr> </table> <table border="0"> <tr><td>D</td><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> <tr><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td></tr> </table> <p>Ingrese una jugada válida pc2</p> <p>ADVERTENCIA: El panel ya se encuentra en el tablero. Debe ingresar una opción válida. Su jugada no es válida. Ingrese nuevamente a continuación.</p>	1	2	3	4	A							#	#	#	#	#	#	#	B			#	#	#	#	#	#	#	#	#	#	#	C			#	#	#	#	#	#	#	#	#	#	#	D							---	---	---	---	---	---	---	
1	2	3	4																																																											
A																																																														
#	#	#	#	#	#	#																																																								
B			#	#	#	#																																																								
#	#	#	#	#	#	#																																																								
C			#	#	#	#																																																								
#	#	#	#	#	#	#																																																								
D																																																														
---	---	---	---	---	---	---																																																								
<p>Se ingresa una jugada correspondiente a colocar el panel cuando éste ya se encuentra en el tablero</p>	<p>Se informa al usuario del suceso y se vuelve a pedir la jugada:</p> <p>Ingrese su intención a continuación y presione ENTER</p> <table border="0"> <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table> <p>+++ --- +++ --- ++- --- +++ --- +++ --- +++</p> <table border="0"> <tr><td>A</td><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> <tr><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td></tr> </table> <table border="0"> <tr><td>B</td><td> </td><td> </td><td>#</td><td>#</td><td>#</td><td>#</td></tr> <tr><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td></tr> </table> <table border="0"> <tr><td>C</td><td> </td><td> </td><td>#</td><td>#</td><td>#</td><td>#</td></tr> <tr><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td><td>#</td></tr> </table> <table border="0"> <tr><td>D</td><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> <tr><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td></tr> </table> <p>Ingrese una jugada válida pc2</p> <p>ADVERTENCIA: El panel ya se encuentra en el tablero. Debe ingresar una opción válida. Su jugada no es válida. Ingrese nuevamente a continuación.</p>	1	2	3	4	A							#	#	#	#	#	#	#	B			#	#	#	#	#	#	#	#	#	#	#	C			#	#	#	#	#	#	#	#	#	#	#	D							---	---	---	---	---	---	---	OK
1	2	3	4																																																											
A																																																														
#	#	#	#	#	#	#																																																								
B			#	#	#	#																																																								
#	#	#	#	#	#	#																																																								
C			#	#	#	#																																																								
#	#	#	#	#	#	#																																																								
D																																																														
---	---	---	---	---	---	---																																																								
<p>Se ingresa una jugada en el formato estipulado. Rendirse: x</p>	<p>Se finaliza la partida y se muestran los resultados y un menú solicitando si se quieren mostrar los resultados a escala global:</p>	OK																																																												

	<p>Ingrese su intención a continuación y presione ENTER</p> <table border="0"> <tr> <td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td></tr> </table> <pre> ++- -+- +-- +-+ +-+ -+- +-- +-+ +-+ A ++- -+- +-- +-# # # # # B # # # # ++- -+- +-- +-# # # # # C # # # # ++- -+- +-- +-# # # # # D ++- -+- +-- +-+ +-+ -+- +-- +-+ +-+ </pre> <p>Ingrese una jugada válida</p> <p>x</p>	1	2	3	4	
1	2	3	4			
<p>Se ingresa una jugada en un formato no estipulado. Rendirse: x11</p>	<p>Se informa al usuario y se vuelve a pedir la jugada:</p> <p>Ingrese su intención a continuación y presione ENTER</p> <table border="0"> <tr> <td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td></tr> </table> <pre> ++- -+- +-- +-+ +-+ -+- +-- +-+ +-+ A ++- -+- +-- +-# # # # # B # # # # ++- -+- +-- +-# # # # # C # # # # ++- -+- +-- +-# # # # # D ++- -+- +-- +-+ +-+ -+- +-- +-+ +-+ </pre> <p>Ingrese una jugada válida</p> <p>x11</p> <p>Su jugada no es válida. Ingrese nuevamente a continuación.</p>	1	2	3	4	<p>OK</p>
1	2	3	4			

<p>Menú de si se desean ver los resultados a escala global</p>	<p>Si se selecciona la opción 1, se muestran los resultados a escala global. Si se selecciona la opción 2, se pasa directamente al siguiente menú.</p> <pre>Desearía ver sus resultados de nuestro juego a escala global??? 1 - Si, me encantaría. 2 - No, gracias. 1 RANKING MUNDIAL: [PARTIDAS JUGADAS: 1 FICHAS ÚLTIMAS JUGADAS: Negras Nombre: dfsdf Edad: 1 Alias: sdfsd PARTIDAS GANADAS: 0 CALIFICACIÓN/ES: [Ninguna Calificación] , PARTIDAS JUGADAS: 1 FICHAS ÚLTIMAS JUGADAS: Blancas Nombre: fdf Edad: 34 Alias: sdfsd PARTIDAS GANADAS: 1 CALIFICACIÓN/ES: [Ninguna Calificación]]</pre>	<p style="text-align: right;">OK</p>
<p>En el menú de resultados a escala global, se ingresa una opción incoherente respecto a las habilitadas.</p> <p>6 - Seis</p>	<p>Se informa al usuario y se vuelve a pedir el dato:</p> <div style="border: 1px solid black; padding: 5px;"> <p>QUÉ DESEA HACER A CONTINUACIÓN?</p> <ol style="list-style-type: none"> 1. Jugar nuevamente. 2. Calificar este juego o sobreescribir una calificación previa. Sus opiniones son siempre tenidas en cuenta. 3. Abandonar el juego. </div> <p>6</p> <p>Dato recibido. Ingrese una opcion correcta dentro del rango brindado. Digite una vez más.</p>	<p style="text-align: right;">OK</p>
<p>Menú final</p>	<p>Si se selecciona la opción 1, comienza una nueva partida. Si se selecciona la opción 2, se solicita una calificación para el juego, luego de ingresarla se vuelve a mostrar el menú. Si se ingresa la opción 3, finaliza el programa.</p> <div style="border: 1px solid black; padding: 5px;"> <p>QUÉ DESEA HACER A CONTINUACIÓN?</p> <ol style="list-style-type: none"> 1. Jugar nuevamente. 2. Calificar este juego o sobreescribir una calificación previa. Sus opiniones son siempre tenidas en cuenta. 3. Abandonar el juego. </div> <p>3</p> <p>Dato recibido. Muchas gracias por jugar. Lo esperamos nuevamente pronto!!! BUILD SUCCESSFUL (total time: 9 minutes 57 seconds)</p>	<p style="text-align: right;">OK</p>

<p>En el menú final, se ingresa una opción incoherente respecto a las disponibles.</p> <p>6 - Seis</p>	<p>Se informa al usuario y se vuelve a pedir el dato:</p> <p>Desearía ver sus resultados de nuestro juego a escala global???</p> <p>1 - Si, me encantaría. 2 - No, gracias.</p> <p>Seis</p> <p>Debe ingresar un valor válido.</p> <p>Desearía ver sus resultados de nuestro juego a escala global???</p> <p>1 - Si, me encantaría. 2 - No, gracias.</p>	<p>OK</p>
<p>Se ingresa la calificación de los jugadores respecto a la partida</p>	<p>Se informa y agradece al usuario. Su historial calificaciones son desplegadas al final de la siguiente partida jugada.</p> <p>2</p> <p>Dato recibido.</p> <p>JUGADORES:</p> <p>Ingresen su calificación.</p> <p>NOTA: La calificación puede ser tanto un valor a escala personal como un comentario. Ambos son agradecidos.</p> <p>Muy buen juego</p> <p>Muchas gracias</p> <p>CALIFICACIONES AÑADIDAS...</p> <p>QUÉ DESEA HACER A CONTINUACIÓN?</p> <p>1. Jugar nuevamente.</p> <p>2. Calificar este juego o sobreescribir una calificación previa.</p> <p>Sus opiniones son siempre tenidas en cuenta.</p> <p>3. Abandonar el juego.</p>	<p>OK</p>
<p>Se termina la partida</p>	<p>Ambos jugadores reciben una partida jugada más. El ganador recibe una partida ganada más. En caso de empate, ningún jugador recibe ninguna partida ganada.</p> <p>Ingrese una jugada válida</p> <p>x</p> <p>EL JUGADOR 2 SE RETIRA.</p> <p>EL GANADOR FUE EL JUGADOR 1!!!!!!</p> <p>FIN DE LA PARTIDA!!!</p> <p>Los resultados fueron...</p> <p>[</p> <p>PARTIDAS JUGADAS: 1 FICHAS ÚLTIMAS JUGADAS: Negras</p> <p>Nombre: dfsdf Edad: 1 Alias: sdfsfdf</p> <p>PARTIDAS GANADAS: 0 CALIFICACIÓN/ES: [Ninguna Calificación]</p> <p>,</p> <p>PARTIDAS JUGADAS: 1 FICHAS ÚLTIMAS JUGADAS: Blancas</p> <p>Nombre: fdf Edad: 34 Alias: sdfsfdf</p> <p>PARTIDAS GANADAS: 1 CALIFICACIÓN/ES: [Ninguna Calificación]</p> <p>]</p> <p>Desearía ver sus resultados de nuestro juego a escala global???</p> <p>1 - Si, me encantaría. 2 - No, gracias.</p>	<p>OK</p>

<p>Simetrías</p>	<p>Si al colocar una ficha, se encuentra una simetría, se cambian todas las fichas de la simetría por fichas de quién la realizó (todas blancas o todas negras). Se validan simetrías en el tablero, y si es que el panel está colocado sobre el tablero, también se validan simetrías sobre el panel y entre el tablero y el panel.</p>	OK
-------------------------	--	---

Ingrese su intención a continuación y presione ENTER
 1 2 3 4

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

A | B | N | | | |

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

B | B | | | | |

las posiciones marcadas con asteriscos se sustituyen por fichas negras.

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

1 2 3 4

C | | | | | | |

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

A | N | N | | | |

D | | | | | | |

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

B | N | N | | | |

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

C | | | | | | |

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

D | | | | | | |

Ingrese una jugada válida

jb2

Su ficha ha sido colocada.

Se ha encontrado la siguiente simetría

1 2 3 4

+++ --- +-- +-- +-- +-- +-- +-- +--

A | * | * | | | |

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

B | * | * | | | |

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

C | | | | | | |

+++ --- +-- +-- +-- +-- +-- +-- +-- +--

Folleto del juego para promoción del producto:

Página 1. PORTADA

Más de 200 nominaciones en los Best Game Awards 2014 celebrado en Finlandia

“UN DESPLIEGUE INCREÍBLE DE DINAMISMO Y VERSATILIDAD”
GamesForEveryone

“JAMÁS VISTO ANTES. DEFINITIVAMENTE MUST PLAY” GamersWayOfLife

¿QUÉ ESTÁS ESPERANDO PARA JUGARLO?

“MEJOR JUEGO JAMÁS CREADO” GAMERQUARTER

EL JUEGO ESTÁ COMENZANDO!!!

J U E G O 0 0 0

ENFRÉNTATE A TUS AMIGOS MÁS CERCANOS EN UN TORNEO QUE EXIGIRÁ AL MÁXIMO DE SUS NOCIONES DE ESTRATEGIA Y PERCEPCIÓN SENSORIAL.

EMBÁRCATE EN ESTA BATALLA POR LA VICTORIA Y LOGRA ENCABEZAR NUESTRAS LISTAS MUNDIALES.

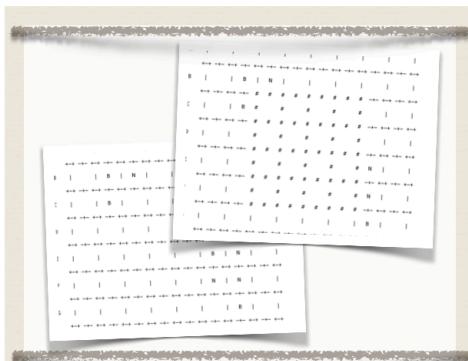
SMARTPERFECCION
Cuaureim 1451, Montevideo, Uruguay
(598) 2 902 15 05
www.SmartPerfeccion.com

	1	2	3	4	5	6	7	8
A								
B			# B # N # B #					
C			# N #	# N #				
D			# B # N # B #					
E			N					
F			B					
G			N					
H			I					

¡PRUEBE ESTE NUEVO E INGENIOSO JUEGO, ÚNICO EN SU CLASE Y PRIMERO EN EL MERCADO!

Nota: Creado en Pages versión 5.5 (2109) para MAC OS X.

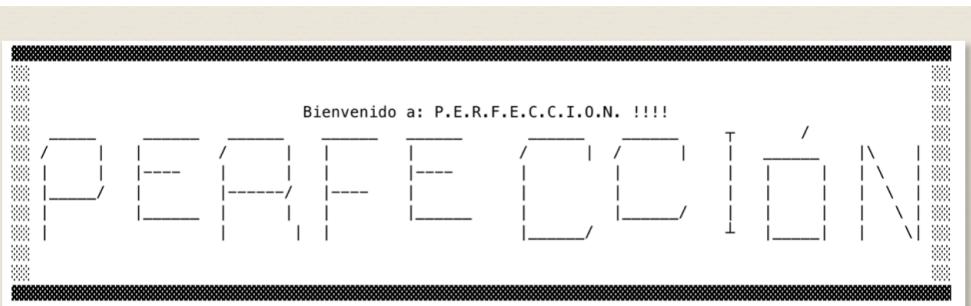
Página 2. INTERIOR



El juego trata de generar simetrías mediante el ingreso de fichas del tipo Blancas o Negras (Jugador 1 o Jugador 2) para así abarcar la mayor cantidad de espacio del tablero con fichas propias.

Los jugadores poseerán un panel móvil de uso opcional que tapará las fichas de debajo de donde éste se coloque. Se podrá a su vez colocarle fichas encima que se moverán junto con el panel.

La partida termina cuando ambos jugadores colocaron todas sus fichas. El ganador resultará quien posee más cantidad de fichas visibles en el tablero.



Al comenzar la partida, se le solicitará su registro y la elección del tamaño del tablero y panel. Existen varios tamaños disponibles a su agrado.

¡¡¡Una vez listo, LA DIVERSIÓN COMENZARÁ!!!

JUGADOR 1, este es su turno:
RECUERDE que su turno será válido solo si ingresa sus intenciones de la forma a continuación explicitada
* Para ingresar una ficha al tablero en una POSICIÓN VACÍA ingrese de la forma 'J-Fila-Columna' (sin guiones) refiriéndose a la fila y columna donde quiera ingresar la ficha respectivamente.
** Para dejar la partida y así rendirse ingrese 'X'.
*** Para colocar el Panel Móvil ingresar de la forma 'P-Fila-Columna' (sin guiones)
**** indicando así la posición del borde superior izquierdo del mismo
RECUERDE: Las filas deben ser ingresadas en caracteres (estando ordenadas comenzando por A) y las y las columnas en números (comenzando por el 1).

¿PODRÁ CONVERTIRSE EN NUESTRO MEJOR JUGADOR?

Al finalizar la partida, poseerá varias opciones a su gusto para proseguir, teniendo la posibilidad de ver su posición en el ranking mundial de nuestro juego.

QUÉ DESEA HACER A CONTINUACIÓN?

1. Jugar nuevamente.
2. Calificar este juego o sobreescribir una calificación previa.
Sus opiniones son siempre tenidas en cuenta.
3. Abandonar el juego.

Nota: Creado en Pages versión 5.5 (2109) para MAC OS X.

Modificaciones a la lógica del juego:

1. Rotación del Panel Móvil:

Opción extra dentro de las posibilidades que el jugador poseerá en su turno. De esta manera, además de colocarlo o mover el panel, el jugador sería capaz de rotarlo (ya sea 90, 180 o 270 grados a elección), variando la posición de las fichas colocadas encima del mismo y posibilitando nuevas simetrías.

```
■ JUGADOR 1, este es su turno:  
■ RECUERDE que su turno será válido solo si ingresa sus intenciones de la forma a continuación explicitada  
■ * Para ingresar una ficha al tablero en una POSICIÓN VACÍA ingrese de la forma 'J-Fila-Columna' (sin  
■ guiones) refiriéndose a la fila y columna donde quiera ingresar la ficha respectivamente.  
■ ** Para dejar la partida y así rendirse ingrese 'X'.  
■ *** Para desplazar el Panel Móvil ingresar de la forma 'M-Dirección-Cantidad de Posiciones' (sin guiones)  
■ **** Para rotar el panel móvil ingrese 'R-(Cantidad de giros de 90 grados en sentido horario)' (sin  
■ guiones) que deseé realizar  
■ RECUERDE: Las filas deben ser ingresadas en caracteres (estando ordenadas comenzando por A) y las  
■ y las columnas en números (comenzando por el 1).
```

Así, se podría implementar una nueva posibilidad de ingreso de forma “Rn”, siendo n el número (entero positivo distinto de 0) de rotaciones de 90 grados realizados (1,2,3,4,...) en sentido horario. Se observa sin embargo que efectuar n giros de cierta cantidad de grados y realizar n+3k (con k entero) giros de la misma cantidad de grados es equivalente.

2. Eliminación temporal del Panel Móvil:

Opción extra colocada a su vez dentro de las posibilidades que el jugador tendrá en su turno. En este caso, el jugador estará posibilitado a remover el panel móvil (y la revelación de sus correspondientes fichas tapadas) además de su correspondiente colocación o movilidad del mismo.

JUGADOR 1, este es su turno:

RECUERDE que su turno será válido solo si ingresa sus intenciones de la forma a continuación explicitada

* Para ingresar una ficha al tablero en una POSICIÓN VACÍA ingrese de la forma 'J-Fila-Columna' (sin

* guiones) refiriéndose a la fila y columna donde quiera ingresar la ficha respectivamente.

** Para dejar la partida y así rendirse ingrese 'X'.

*** Para desplazar el Panel Móvil ingresar de la forma 'M-Dirección-Cantidad de Posiciones' (sin guiones)

**** Para remover el panel móvil del tablero ingrese 'E'. Recuerde que podrá ingresarlo en futuros turnos

RECUERDE: Las filas deben ser ingresadas en caracteres (estando ordenadas comenzando por A) y las y las columnas en números (comenzando por el 1).

De esta manera, la opción ingresada podría ser de la forma "E" lo que brindaría la información al sistema de retirar el panel móvil hasta un posible reingreso posterior. En el momento de su reincorporación, el panel conservaría la última posición registrada antes de su retiro al igual que la posición de las fichas que se encontraban encima del mismo.

Clases utilizadas (y su respectivo código):

Clase Perfección:

```
package obligatorio;
// PROGRAMACIÓN 2 - Obligatorio 1
// Fernando Hernández (173631 - Ingeniería en Telecomunicaciones) -
// Sebastián Effa (193248 - Ingeniería en Electrónica)

import java.util.*;

public class Perfeccion {

    //MÉTODO MAIN DEL OBLIGATORIO
    public static void main(String[] args) {
        //Variables inicializadas únicamente al comienzo del programa
        boolean gameOver = false;
        int partida = 1;
        ArrayList<Jugador> listaTodosJugadores = new
ArrayList<Jugador>();

        Interfaz.introduccion();

        //Repetición hasta el término del juego
        while (!gameOver) {

            //Variables reinicializadas al comienzo de cada juego
            (iniicalizadas en pocas líneas para ahorrar espacio del método)
            int opcionJuego = 0, puntosJugador1 = 0, puntosJugador2 = 0,
            turno = 1, filaPanel = 0, columnaPanel = 0, fichasJugador1 = 0,
            fichasJugador2;
            boolean rendicion = false, setOver = false, movilEnTablero =
            false, tieneFichasJugador1 = true, tieneFichasJugador2 = true;
            boolean opcionCorrecta = false, optionCorrecta = false,
            finSet = false, posicionesLibres = true;
            String opcion, movimiento = " ";
            int[] datos = new int[3];
            Juego game = new Juego();
            Tablero table = new Tablero();

            System.out.println("\n\nPARTIDA " + partida);
```

```
//Solicitud datos de los Jugadores

game.setListaJugadores(Interfaz.registroJugadores(listaTodosJugadores));
    Interfaz.cambioDatosJugador(game, game.getListajugadores(),
listaTodosJugadores);

    //Solicitud de datos y creación de las Matrices a utilizar
    game.ingresarDimensionesTableros(game, table);

    Interfaz.comienzoJuego();

    //Repetición hasta que ambos jugadores se quedan sin fichas
    (hasta que termina la partida)
    while (!setOver) {
        while (!opcionCorrecta) {
            if (!movilEnTablero) { //Si el panel móvil NO se
encuentra en el tablero
                Interfaz.menuJuego(turno, " *** Para colocar el
Panel Móvil ingresar de la forma 'P-Fila-Columna' (sin guiones)
\n"
+ " " *** indicando así la posición del
borde superior izquierdo del mismo
");
                table.imprimirMatriz(table.getMatrizTablero(),
table.getPanel(), filaPanel, columnaPanel, movilEnTablero, false);
            }
            if (movilEnTablero) { //Si el panel móvil se
encuentra en el tablero
                Interfaz.menuJuego(turno, " *** Para desplazar el
Panel Móvil ingresar de la forma 'M-Dirección-Cantidad de Posiciones'
(sin guiones) ");
                table.imprimirMatriz(table.getTableroConPanel(),
table.getPanel(), filaPanel, columnaPanel, movilEnTablero, false);
            }
            opcion = Interfaz.pedidoCaracteres("Ingrese una
jugada válida", "").toUpperCase().trim().replaceAll(" ", "");
            //Todo a mayúsculas, sin espacios
            movimiento = opcion.substring(0, 1); // El movimiento
puede ser únicamente 'J', 'X y ('P' o 'M')
            if ((movimiento.equals("X")) && (opcion.length() ==
1)) { //Si el jugador desea rendirse
                game.abandonarPartida(game, turno);
                rendicion = true;
                setOver = true;
            }
        }
    }
}
```

```

        opcionCorrecta = true;
    }
    if (opcion.length() == 3) {
        if (movilEnTablero) {
            if (movimiento.equals("J")) { //Si el
jugador desea ingresar una ficha estando el panel móvil en el tablero
                try {
                    opcionCorrecta =
game.colocarFichaTableroConPanel(game, table, opcion, datos, turno,
filaPanel, columnaPanel,
                                            tieneFichasJugador1,
tieneFichasJugador2);
                } catch (Exception e) {
                    opcionCorrecta = false;
                }
            }
            if (movimiento.equals("P")) { //Si el
jugador desea ingresar el panel móvil ya estando éste en el tablero
(jugada errónea)
                System.out.println("\nADVERTENCIA: El
panel ya se encuentra en el tablero. Debe ingresar una opción
válida.\n");
            }
            if (movimiento.equals("M")) { //Si el
jugador desea mover el panel móvil ya estando éste en el tablero
                try {
                    String direccion =
opcion.substring(1, 2);
                    int distancia =
Interfaz.pasarANumero(opcion.substring(2, 3));
                    if ((direccion.equals("I") ||
direccion.equals("D") || direccion.equals("A") || direccion.equals("B"))
&&
Interfaz.esNumero(opcion.substring(2, 3))) {
                        //Array que contiene en la primer
posición si se movió el panel, y en las siguientes donde se movió
                        int[] panelMovido = {0, 0, 0};
                        panelMovido =
game.moverPanel(table.getTableroConPanel(), table.getPanel().length,
filaPanel, columnaPanel,
                                            direccion, distancia);
                        if (panelMovido[0] == 1) {
                            opcionCorrecta = true;
                            filaPanel = panelMovido[1];
                        }
                    }
                }
            }
        }
    }
}

```

```
        columnaPanel =
panelMovido[2];

table.setTableroConPanel(Juego.colocarPanel(table.getPanel(), filaPanel,
columnaPanel, table.getMatrizTablero()));

table.setPanel(game.panelNuevo(table.getTableroConPanel(), filaPanel,
columnaPanel, table.getPanel().length));
}
if (panelMovido[0] == 0) {
System.out.println("\nADVERTENCIA: No se puede mover el panel como
indicó.\n");
}
} else {

System.out.println("\nADVERTENCIA: Ingrese un movimiento válido para el
panel.\n");
}

} catch (Exception e) {
opcionCorrecta = false;
}
}
if (opcionCorrecta) {

table.imprimirMatriz(table.getTableroConPanel(), table.getPanel(),
filaPanel, columnaPanel, movilEnTablero, false);
}
}
}
if (!movilEnTablero) {
if (movimiento.equals("J")) { //Si el jugador
desea colocar una ficha NO estando el panel móvil en el tablero
try {
opcionCorrecta =
game.colocarFichaTableroSinPanel(game, table, opcion, datos, turno,
filaPanel, columnaPanel,
tieneFichasJugador1,
tieneFichasJugador2);
} catch (Exception e) {
}
}
if (movimiento.equals("P")) { //Si el jugador
desea colocar el panel móvil NO estando éste en el tablero
```

```

try {
    filaPanel =
Interfaz.pasarANumero(opcion.substring(1, 2)) - 1;
    columnaPanel =
Interfaz.pasarANumero(opcion.substring(2, 3)) - 1;

table.setTableroConPanel(game.copiaMatriz(table.getMatrizTablero()));
    opcionCorrecta =
game.sePuedeColocarPanel(table.getPanel().length, filaPanel,
columnaPanel, table.getMatrizTablero())
    &&
Interfaz.esLetra(opcion.substring(1, 2)) &&
Interfaz.esNumero(opcion.substring(2, 3));
    if (opcionCorrecta) {

table.setTableroConPanel(Juego.colocarPanel(table.getPanel(), filaPanel,
columnaPanel, table.getTableroConPanel()));
        movilEnTablero = true;
    }
} catch (Exception e) {
    opcionCorrecta = false;
}
}
if (movimiento.equals("M")) { //Si el jugador
desea mover el panel móvil NO estando éste en el tablero (jugada errónea)
    System.out.println("\nADVERTENCIA: El panel
no se encuentra en el tablero. Debe ingresar una opción válida.\n");
}
if (opcionCorrecta && !movimiento.equals("P")) {
//Si la opción se ingresó adecuadamente y ésta NO se refiere a colocar el
panel

table.imprimirMatriz(table.getMatrizTablero(), table.getPanel(),
filaPanel, columnaPanel, movilEnTablero, false);
}
if (opcionCorrecta && movimiento.equals("P")) {
//Si la opción se ingresó adecuadamente y ésta se refiere a colocar el
panel

table.imprimirMatriz(table.getTableroConPanel(), table.getPanel(),
filaPanel, columnaPanel, movilEnTablero, false);
}
if (!opcionCorrecta) { //Si la opcion NO se ingreso
adecuadamente

```

```
        System.out.println("Su jugada no es válida.  
Ingrese nuevamente a continuación.");  
    }  
}  
} //Al finalizar cada turno de cada jugador  
opcionCorrecta = false;  
turno++;  
tieneFichasJugador1 =  
game.getListaJugadores().get(0).getCantidadFichas() > 0; //Cantidad de  
fichas restantes del Jugador 1  
tieneFichasJugador2 =  
game.getListaJugadores().get(1).getCantidadFichas() > 0; //Cantidad de  
fichas restantes del Jugador 2  
  
//Se juega hasta que ambos jugadores se queden sin fichas  
o alguno se rinda, lo que es equivalente a que no haya posiciones libres  
en el tablero  
if ((!movilEnTablero) &&  
(game.contarPuntos(table.getMatrizTablero(), " ") == 0)) {  
    System.out.println("No hay posiciones disponibles,  
puede colocar el panel para seguir o ingresar 'X' para abandonar");  
//Si no está el panel móvil colocado y no hay posiciones libres en el  
tablero  
}  
if ((movilEnTablero) &&  
(game.contarPuntos(table.getMatrizTablero(), " ") +  
game.contarPuntos(table.getPanel(), " ") == 0)) {  
    posicionesLibres = false; //Si está el panel móvil  
colocado y no hay posiciones libres ni en el tablero ni en el panel móvil  
}  
if ((!posicionesLibres) || (!tieneFichasJugador1 &&  
!tieneFichasJugador2)){  
    setOver=true;  
}  
}  
  
if (movilEnTablero) {  
    puntosJugador1 =  
game.contarPuntos(table.getTableroConPanel(), "B");  
    puntosJugador2 =  
game.contarPuntos(table.getTableroConPanel(), "N");  
} else {  
    puntosJugador1 =  
game.contarPuntos(table.getMatrizTablero(), "B");  
    puntosJugador2 =  
game.contarPuntos(table.getMatrizTablero(), "N");
```

```
        }
        game.agregarPuntos(listaTodosJugadores,
game.getListaJugadores());
        Interfaz.mensajeGanador(game, puntosJugador1, puntosJugador2,
rendicion);
        System.out.println("FIN DE LA PARTIDA!!! \nLos resultados
fueron... \n");

game.setListaJugadores(Interfaz.devolverJugadoresOrdenados(game.getListaJ
ugadores()));
        System.out.println(game.getListaJugadores());
        partida++;
        while (!optionCorrecta) { //Menú de escala o ranking global
            int option = Interfaz.verificacionEntero("Desearía ver
sus resultados de nuestro juego a escala global???\\n 1 - Si, me
encantaría. \\t"
+ " 2 - No, gracias.", "Debe ingresar un valor
válido.", "");
            optionCorrecta =
Interfaz.opcionesRankingMundial(listaTodosJugadores, optionCorrecta,
option);
        }

        while (!finSet) { //Menú final de la Partida
            opcionJuego = Interfaz.mensajeFinSet();
            boolean[] finPartidas = Interfaz.opcionesFinSet(game,
opcionJuego, listaTodosJugadores);
            finSet = finPartidas[0];
            gameOver = finPartidas[1];
        }
        game.borrarListaJugadores(game.getListaJugadores());
    }
}
```

Fin de Clase Perfección

Clase Juego:

```
package obligatorio;
// PROGRAMACIÓN 2 - Obligatorio 1
// Fernando Hernández (173631 - Ingeniería en Telecomunicaciones) -
// Sebastián Effa (193248 - Ingeniería en Electrónica)

import java.util.*;

public class Juego {

    //Atributos de la clase Juego
    private ArrayList<Jugador> listaJugadores;

    //Métodos de la clase Juego

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

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

    public void borrarListaJugadores(ArrayList<Jugador> listaJugadores) {
        this.listaJugadores.clear();
    }

    //Método para crear e inicializar una Matriz de Strings (apropiarla
    para la partida)
    public static String[][] matrizNueva(int dimension) {
        String[][] matriz = new String[dimension][dimension];
        for (int fila = 0; fila < matriz.length; fila++) {
            for (int columna = 0; columna < matriz[0].length; columna++)
{
                matriz[fila][columna] = " ";
            }
        }
        return matriz;
    }

    //Método para copiar una matriz
    public static String[][] copiaMatriz(String[][] matriz) {
```

```

String[][] matrizCopia = new
String[matriz.length][matriz[0].length];
    for (int fila = 0; fila < matriz.length; fila++) {
        for (int columna = 0; columna < matriz[0].length; columna++)
{
            matrizCopia[fila][columna] = matriz[fila][columna];
        }
    }
    return matrizCopia;
}

//Método para colocar el panel en el tablero
public static String[][] colocarPanel(String[][] panel, int fila, int
columna, String[][] tablero) {
    String[][] copiaTablero = copiaMatriz(tablero);
    if (((fila + panel.length) <= tablero.length) && ((columna +
panel[0].length) <= tablero[0].length)) {
        for (int i = 0; i < panel.length; i++) {
            for (int j = 0; j < panel[0].length; j++) {
                copiaTablero[fila + i][columna + j] = panel[i][j];
            }
        }
    }
    return copiaTablero;
}

//Método para comprobar si se puede colocar el panel en cierta
posición
public static boolean sePuedeColocarPanel(int dimension, int fila,
int columna, String[][] tablero) {
    return (((fila + dimension) <= tablero.length) && ((columna +
dimension) <= tablero[0].length) && fila >= 0 && columna >= 0);
}

//Método que retorna la nueva posición del panel y si se puede mover
public static int[] moverPanel(String[][] tableroConPanel, int
dimension, int filaPanel, int columnaPanel, String direccion, int
distancia) {
    int filaNueva = 0;
    int columnaNueva = 0;
    int[] retorno = {0, 0, 0};
    if (direccion.equals("I")) {
        filaNueva = filaPanel;
        columnaNueva = (columnaPanel - distancia);
    }
}

```

```

if (direccion.equals("D")) {
    filaNueva = filaPanel;
    columnaNueva = (columnaPanel + distancia);
}
if (direccion.equals("A")) {
    filaNueva = (filaPanel - distancia);
    columnaNueva = columnaPanel;
}
if (direccion.equals("B")) {
    filaNueva = (filaPanel + distancia);
    columnaNueva = columnaPanel;
}
if (sePuedeColocarPanel(dimension, filaNueva, columnaNueva,
tableroConPanel)) {
    retorno[0] = 1;
}
retorno[1] = filaNueva;
retorno[2] = columnaNueva;
return retorno;
}

//Método para actualizar el panel una vez se coloca o se mueve sobre
el tablero
public static String[][] panelNuevo(String[][] tableroConPanel, int
filaPanel, int columnaPanel, int dimension) {
    String[][] panelNuevo = matrizNueva(dimension);
    for (int fila = filaPanel; fila < (filaPanel + dimension);
fila++) {
        for (int columna = columnaPanel; columna < (columnaPanel +
dimension); columna++) {
            panelNuevo[fila - filaPanel][columna - columnaPanel] =
tableroConPanel[fila][columna];
        }
    }
    return panelNuevo;
}

//Método para comprobar si una ficha se coloca sobre el panel (si no
es así, la ficha también se coloca en el tablero sin panel)
public static boolean fichaSobrePanel(int filaFicha, int
columnaFicha, int filaPanel, int columnaPanel, int dimension) {
    return (filaFicha >= filaPanel && filaFicha <= filaPanel +
dimension && columnaFicha >= columnaPanel && columnaFicha <= columnaPanel +
dimension);
}

```

```
//Método para agregar puntos a los jugadores
public void agregarPuntos(ArrayList<Jugador> listaTodosJugadores,
ArrayList<Jugador> listaJugadores) {

    try {

        listaJugadores.get(0).setPartidasJugadas(listaJugadores.get(0).getPartidasJugadas() + 1);

        listaJugadores.get(1).setPartidasJugadas(listaJugadores.get(1).getPartidasJugadas() + 1);
        if (listaTodosJugadores.isEmpty()) {
            listaTodosJugadores.add(listaJugadores.get(0));
            listaTodosJugadores.add(listaJugadores.get(1));
        } else {
            Iterator<Jugador> it = listaTodosJugadores.iterator();
            while (it.hasNext()) {
                Jugador unJugador = it.next();
                if (unJugador.equals(listaJugadores.get(0))) {

                    unJugador.setPartidasJugadas(unJugador.getPartidasJugadas() + 1);

                    unJugador.setPartidasGanadas(listaJugadores.get(0).getPartidasGanadas());

                    unJugador.setFicha(listaJugadores.get(0).getFicha());
                } else {
                    listaTodosJugadores.add(listaJugadores.get(0));
                }
                if (unJugador.equals(listaJugadores.get(1))) {

                    unJugador.setPartidasJugadas(unJugador.getPartidasJugadas() + 1);

                    unJugador.setPartidasGanadas(listaJugadores.get(1).getPartidasGanadas());

                    unJugador.setFicha(listaJugadores.get(1).getFicha());
                } else {
                    listaTodosJugadores.add(listaJugadores.get(1));
                }
            }
        }
    } catch (Exception e) {
    }
}
```

```

//Método para contar puntos por partida
public static int contarPuntos(String[][] matriz, String criterio) {
    int puntos = 0;
    for (int fila = 0; fila < matriz.length; fila++) {
        for (int columna = 0; columna < matriz[0].length; columna++) {
            if (matriz[fila][columna].equals(criterio)) {
                puntos++;
            }
        }
    }
    return puntos;
}

//Método para colocar asteriscos donde se encontró la simetría o
numerales donde se encuentra el panel (en una copia del tablero)
public static String[][] colocarString(String[][] tablero, int fila,
int columna, int dimension, String string) {
    String[][] copiaTablero = copiaMatriz(tablero);
    for (int i = fila; i < (fila + dimension); i++) {
        for (int j = columna; j < (columna + dimension); j++) {
            if (string.equals("*")) {
                copiaTablero[i][j] = string;
            }
            if (string.equals("#")) { //agregar si fila es impar
                if (i % 2 == 0 && (columna % 2 == 0 || (columna % 2
!= 0 && j != columna))) {
                    copiaTablero[i][j] = " " + string + " ";
                }
                if (i % 2 != 0 && j % 2 == 0 && i != fila) {
                    copiaTablero[i][j] = " " + string + " ";
                }
            }
        }
    }
    return copiaTablero;
}

//Método para sustituir las fichas en una simetría
public static String[][] cambiarSimetriaTablero(String[][] tablero,
int fila, int columna, int dimensionSimetria, int turno, boolean
tableroSinPanel,
    int filaPanel, int columnaPanel, int dimensionPanel) {
    String[][] copiaTablero = copiaMatriz(tablero);
    for (int i = fila; i < (fila + dimensionSimetria); i++) {

```

```

        for (int j = columna; j < (columna + dimensionSimetria); j++)
    {
        if (turno % 2 != 0 && !tableroSinPanel){
            copiaTablero[i][j] = "B";
        }
        if (turno % 2 == 0 && !tableroSinPanel){
            copiaTablero[i][j] = "N";
        }
        if(tableroSinPanel && !Juego.fichaSobrePanel(i, j,
filaPanel, columnaPanel, dimensionPanel)){
            if (turno % 2 != 0){
                copiaTablero[i][j] = "B";
            }
            if (turno % 2 == 0){
                copiaTablero[i][j] = "N";
            }
        }
    }

    return copiaTablero;
}

//Método desplegado en caso de querer realizar un comentario sobre el
juego
public static void calificarJuego(ArrayList<Jugador>
listaTodosJugadores, ArrayList<Jugador> listaJugadores) {
    listaJugadores.get(0).getCalificaciones().clear();
    listaJugadores.get(1).getCalificaciones().clear();
    String calificacion = Interfaz.pedidoCaracteres("JUGADORES:
\nIngresen su calificación. \nNOTA: La calificación puede ser tanto un
valor a escala personal como un comentario. Ambos son agradecidos.",
"Muchas gracias");
    Iterator<Jugador> it = listaTodosJugadores.iterator();
    while (it.hasNext()) {
        Jugador unJugador = it.next();
        if
(unJugador.getAlias().equals(listaJugadores.get(0).getAlias())))
            unJugador.getCalificaciones().add(calificacion);
    }
}
System.out.println("\nCALIFICACIONES AÑADIDAS...");
```

```

//Método para ingresar las dimensiones de los Tableros a usar
//Tablero Fijo y Panel Móvil)
public static void ingresarDimensionesTableros(Juego game, Tablero
table) {
    table.setDimensionTablero(Interfaz.pedidoMatrizTablero("\nTABLERO
PRINCIPAL_____", "4, 6 u 8", 4, 6, 8));

    table.setMatrizTablero(game.matrizNueva(table.getDimensionTablero()));

    table.setTableroConPanel(game.copiaMatriz(table.getMatrizTablero()));

    table.setTableroVacio(game.matrizNueva(table.getDimensionTablero()));
    if (table.getDimensionTablero() == 4) {
        System.out.println("\nLa dimensión única disponible del Panel
Móvil para la dimensión del Tablero elegida es 2x2");
        table.setDimensionMovil(2);
    }
    if (table.getDimensionTablero() == 6) {

        table.setDimensionMovil(Interfaz.pedidoMatrizTablero("\nTABLERO
MÓVIL_____", "2 o 3", 2, 3, 173631));
    }
    if (table.getDimensionTablero() == 8) {

        table.setDimensionMovil(Interfaz.pedidoMatrizTablero("\nTABLERO
MÓVIL_____", "2, 3 o 4", 2, 3, 4));
        table.setPanel(game.matrizNueva(table.getDimensionMovil()));

        game.getListaJugadores().get(0).setCantidadFichas((table.getDimensionMovil()
*table.getDimensionMovil()
+
table.getDimensionTablero()*table.getDimensionTablero()) / 2);

        game.getListaJugadores().get(1).setCantidadFichas((table.getDimensionMovil()
*table.getDimensionMovil()
+
table.getDimensionTablero()*table.getDimensionTablero()) / 2);
    }

    //Método desplegado al ser el movimiento una X y así querer el
    jugador de turno retirarse de la partida
    public static void abandonarPartida(Juego game, int turno) {
        if (turno % 2 != 0) { //Si el turno es del Jugador 1 (Blancas)

```

```
System.out.println("\nEL JUGADOR 1 SE RETIRA.\n EL GANADOR  
FUE EL JUGADOR 2!!!!!!");  
  
game.getListaJugadores().get(1).setPartidasGanadas(game.getListaJugadores()  
().get(1).getPartidasGanadas() + 1);  
}  
if (turno % 2 == 0) { //Si el turno es del Jugador 2 (Negras)  
    System.out.println("\nEL JUGADOR 2 SE RETIRA.\n EL GANADOR  
FUE EL JUGADOR 1!!!!!!");  
  
game.getListaJugadores().get(0).setPartidasGanadas(game.getListaJugadores()  
().get(0).getPartidasGanadas() + 1);  
}  
}  
  
//Método para colocar una Ficha (y su consiguiente verificación de  
simetría) en un tablero con panel (movilEnTablero == true)  
public static boolean colocarFichaTableroConPanel(Juego game, Tablero  
table, String opcion, int[] datos, int turno, int filaPanel, int  
columnaPanel,  
        boolean tieneFichasJugador1, boolean tieneFichasJugador2) {  
  
    boolean opcionCorrecta = false;  
    int filaFicha = Interfaz.pasarANumero(opcion.substring(1, 2)) -  
1;  
    int columnaFicha = Interfaz.pasarANumero(opcion.substring(2, 3))  
- 1;  
    if(Interfaz.esLetra(opcion.substring(1, 2)) &&  
Interfaz.esNumero(opcion.substring(2, 3))){  
        if (table.posicionValida(table.getTableroConPanel(),  
filaFicha, columnaFicha)) {  
            if ((turno % 2 != 0) && (tieneFichasJugador1)){  
                table.setFichaTableroConPanel(filaFicha,  
columnaFicha, "B");  
                if (!Juego.fichaSobrePanel(filaFicha, columnaFicha,  
filaPanel, columnaPanel, table.getPanel().length)) {  
                    table.setFichaMatrizTablero(filaFicha,  
columnaFicha, "B");  
                }  
                System.out.println("Su ficha a sido colocada.");  
                opcionCorrecta = true;  
            }  
        }  
    }  
    game.getListaJugadores().get(0).setCantidadFichas(game.getListaJugadores()  
().get(0).getCantidadFichas() - 1);  
}
```

```
if ((turno % 2 ==0) && (tieneFichasJugador2)){
    table.setFichaTableroConPanel(filaFicha,
columnaFicha, "N");
    if (!Juego.fichaSobrePanel(filaFicha, columnaFicha,
filaPanel, columnaPanel, table.getPanel().length)) {
        table.setFichaMatrizTablero(filaFicha,
columnaFicha, "N");
    }
    System.out.println("Su ficha a sido colocada.");
    opcionCorrecta = true;
}

game.getListaJugadores().get(1).setCantidadFichas(game.getListaJugadores(
).get(1).getCantidadFichas() - 1);
}
if((turno%2!= 0 && !tieneFichasJugador1) || (turno%2==0
&& !tieneFichasJugador2)){
    System.out.println("No tiene Fichas disponibles.
Ingrese otra jugada");
}
datos =
table.buscarSimetrias(table.getTableroConPanel(),filaFicha,columnaFicha);
if (datos[0] != 0) {
    System.out.println("Se ha encontrado la siguiente
simetría\n");

table.imprimirMatriz(Juego.colocarString(table.getTableroVacio(),
datos[1], datos[2], datos[0], "*"), table.getPanel(), filaPanel,
columnaPanel, true, true);
if (turno % 2 != 0) {
    System.out.println("\nLas posiciones marcadas con
asteriscos se sustituyen por fichas BLANCAS.\n");
}
if (turno % 2 == 0) {
    System.out.println("\nLas posiciones marcadas con
asteriscos se sustituyen por fichas NEGRAS.\n");
}

table.setTableroConPanel(Juego.cambiarSimetriaTablero(table.getTableroCon
Panel(), datos[1], datos[2], datos[0], turno, false,
filaPanel, columnaPanel,
table.getPanel().length));

table.setMatrizTablero(Juego.cambiarSimetriaTablero(table.getMatrizTabler
o(), datos[1], datos[2], datos[0], turno, true,
```

```
        filaPanel, columnaPanel,
table.getPanel().length));
    }

table.setPanel(game.panelNuevo(table.getTableroConPanel(), filaPanel,
columnaPanel, table.getPanel().length));
}
}

return opcionCorrecta;
}

//Método para colocar una Ficha (y su consiguiente verificación de
simetría) en un tablero sin panel (movilEnTablero == false)
public static boolean colocarFichaTableroSinPanel(Juego game, Tablero
table, String opcion, int[] datos, int turno, int filaPanel, int
columnaPanel,
boolean tieneFichasJugador1, boolean tieneFichasJugador2) {

    boolean opcionCorrecta = false;
    int filaFicha = Interfaz.pasarANumero(opcion.substring(1, 2)) -
1;
    int columnaFicha = Interfaz.pasarANumero(opcion.substring(2, 3)) -
1;
    if (table.posicionValida(table.getMatrizTablero(), filaFicha,
columnaFicha) && Interfaz.esLetra(opcion.substring(1, 2)) &&
Interfaz.esNumero(opcion.substring(2, 3))) {
        if ((turno % 2 != 0) && (tieneFichasJugador1)) {
            table.setFichaMatrizTablero(filaFicha, columnaFicha,
"B");
            System.out.println("Su ficha ha sido colocada.");
            opcionCorrecta = true;
        }
        game.getListaJugadores().get(0).setCantidadFichas(game.getListaJugadores(
).get(0).getCantidadFichas() - 1);
    }
    if ((turno % 2 == 0) && (tieneFichasJugador2)) {
        table.setFichaMatrizTablero(filaFicha, columnaFicha,
"N");
        System.out.println("Su ficha ha sido colocada.");
        opcionCorrecta = true;
    }
    game.getListaJugadores().get(1).setCantidadFichas(game.getListaJugadores(
).get(1).getCantidadFichas() - 1);
}
```

```

        if((turno%2!= 0 && !tieneFichasJugador1) || (turno%2==0 &&
!tieneFichasJugador2)){
            System.out.println("No tiene Fichas disponibles.
Ingrese otra jugada");
        }
        datos = table.buscarSimetrias(table.getMatrizTablero(),
filaFicha, columnaFicha);
        if (datos[0] != 0) {
            System.out.println("Se ha encontrado la siguiente
simetría\n");

table.imprimirMatriz(Juego.colocarString(table.getTableroVacio(),
datos[1], datos[2], datos[0], "*"), table.getPanel(),
filaPanel, columnaPanel, false, true);
        if (turno % 2 != 0) {
            System.out.println("\nlas posiciones marcadas con
asteriscos se sustituyen por fichas blancas.\n");
        }
        if (turno % 2 == 0) {
            System.out.println("\nlas posiciones marcadas con
asteriscos se sustituyen por fichas negras.\n");
        }

table.setMatrizTablero(Juego.cambiarSimetriaTablero(table.getMatrizTabler
o(), datos[1], datos[2], datos[0], turno, false,
filaPanel, columnaPanel,
table.getPanel().length));
        }
    }
    return opcionCorrecta;
}

//Método usado en el main para colocar el panel móvil en el tablero
fijo
public static boolean panelEnTablero(Juego game, Tablero table,
String opcion, int filaPanel, int columnaPanel) {
    boolean movilEnTablero = false;
    filaPanel = Interfaz.pasarANumero(opcion.substring(1, 2)) - 1;
    columnaPanel = Interfaz.pasarANumero(opcion.substring(2, 3)) - 1;

table.setTableroConPanel(game.copiaMatriz(table.getMatrizTablero()));
    boolean opcionCorrecta =
game.sePuedeColocarPanel(table.getPanel().length, filaPanel,
columnaPanel, table.getMatrizTablero());
    if (opcionCorrecta) {

```

```
        table.setTableroConPanel(Juego.colocarPanel(table.getPanel(),
filaPanel, columnaPanel, table.getTableroConPanel()));
        movilEnTablero = true;
    }
    return movilEnTablero;
}

}
```

Fin de Clase Juego

Clase Jugador:

```
package obligatorio;
// PROGRAMACIÓN 2 - Obligatorio 1
// Fernando Hernández (173631 - Ingeniería en Telecomunicaciones) -
// Sebastián Effa (193248 - Ingeniería en Electrónica)

import java.util.*;

public class Jugador {

    //Atributos de la clase Jugador
    private String nombre;
    private int edad;
    private String alias;
    private String ficha;
    private int partidasGanadas;
    private int partidasJugadas;
    private ArrayList<String> calificaciones;
    private int cantidadFichas;

    //Constructor por Defecto (sin Parámetros)
    public Jugador(){
        this.setNombre("Sin nombre.");
        this.setEdad(0);
        this.setAlias("Sin alias");
        this.setFicha("Sin ficha");
        this.setPartidasGanadas(0);
        this.setPartidasJugadas(0);
        this.setCantidadFichas(0);
    }

    //Métodos de la clase Jugador
    public String getNombre(){
        return nombre;
    }

    public int getEdad(){
        return edad;
    }

    public String getAlias(){
        return alias;
    }
}
```

```
public String getFicha(){
    return ficha;
}

public int getPartidasGanadas(){
    return partidasGanadas;
}

public int getPartidasJugadas(){
    return partidasJugadas;
}

public int getCantidadFichas(){
    return cantidadFichas;
}

public ArrayList<String> getCalificaciones() {
    return calificaciones;
}

public void setNombre(String unNombre){
    this.nombre=unNombre;
}

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

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

public void setFicha(String unaFicha){
    this.ficha=unaFicha;
}

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

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

```
public void setCantidadFichas (int unaCantidadFichas){
    this.cantidadFichas=unaCantidadFichas;
}

public void setCalificaciones(ArrayList<String> calificaciones) {
    this.calificaciones = calificaciones;
}

@Override
public String toString(){
    return "\nPARTIDAS JUGADAS: "+this.getPartidasJugadas()+"\n"
FICHAS ÚLTIMAS JUGADAS: "+this.getFicha()
        +"\n"+ "Nombre: "+this.getNombre() +" Edad:
"+this.getEdad()+" Alias: "+this.getAlias()
        +"\nPARTIDAS GANADAS: "+this.getPartidasGanadas()+"\n"
CALIFICACIÓN/ES: "+this.getCalificaciones()+"\n\n";
}

//Redefinición del Método equals(Object o) en clase Jugador
public boolean equals(Jugador o){
    boolean ok = this.getNombre().equals(o.getNombre()) &&
this.getEdad()==o.getEdad() && this.getAlias().equals(o.getAlias());
    return ok;
}
```

Fin de Clase Jugador

Clase Tablero:

```
package obligatorio;
// PROGRAMACIÓN 2 - Obligatorio 1
// Fernando Hernández (173631 - Ingeniería en Telecomunicaciones) -
// Sebastián Effa (193248 - Ingeniería en Electrónica)

public class Tablero {

    //Atributos de la clase Tablero
    private String[][] matrizTablero;
    private String[][] tableroConPanel;
    private int dimensionMovil;
    private int dimensionTablero;
    private String[][] panel;
    private String[][] tableroVacio;

    //Métodos de la clase Tablero
    public String[][] getTableroVacio() {
        return tableroVacio;
    }

    public void setTableroVacio(String[][] tableroVacio) {
        this.tableroVacio = tableroVacio;
    }

    public String[][] getPanel() {
        return panel;
    }

    public void setPanel(String[][] panel) {
        this.panel = panel;
    }

    public int getDimensionMovil() {
        return dimensionMovil;
    }

    public void setDimensionMovil(int dimensionMovil) {
        this.dimensionMovil = dimensionMovil;
    }

    public int getDimensionTablero() {
```

```
        return dimensionTablero;
    }

    public void setDimensionTablero(int dimensionTablero) {
        this.dimensionTablero = dimensionTablero;
    }

    public String[][] getMatrizTablero() {
        return matrizTablero;
    }

    public void setMatrizTablero(String[][] matrizTablero) {
        this.matrizTablero = matrizTablero;
    }

    public void setFichaMatrizTablero(int fila, int columna, String letra) {
        this.matrizTablero[fila][columna] = letra;
    }

    public String[][] getTableroConPanel() {
        return tableroConPanel;
    }

    public void setTableroConPanel(String[][] tableroConPanel) {
        this.tableroConPanel = tableroConPanel;
    }

    public void setFichaTableroConPanel(int fila, int columna, String letra) {
        this.tableroConPanel[fila][columna] = letra;
    }

    //Método para buscar simetrías (utiliza simetrías debajo)
    public static int [] buscarSimetrias(String [][] tablero, int filaFicha, int columnaFicha){
        int [] datos = {0,0,0}; //Array cuyo primer valor refiere a la
        //dimension de la simetría (0 es que no hay), segundo y tercer valor son la
        //posición
        datos = simetrias4x4Lados(tablero, filaFicha, columnaFicha);
        if(datos[0] == 0){
            datos = simetrias4x4Lados(tablero, (filaFicha-1),
        columnaFicha);
        }
        if(datos[0] == 0){
```

```

        datos = simetrias4x4Lados(tablero, (filaFicha-2),
columnaFicha);
    }
    if(datos[0] == 0){
        datos = simetrias4x4Lados(tablero, filaFicha, (columnaFicha-
1));
    }
    if(datos[0] == 0){
        datos = simetrias4x4Lados(tablero, (filaFicha-2),
(columnaFicha-2));
    }
    if(datos[0] == 0){
        datos = simetrias4x4Medio(tablero, filaFicha, columnaFicha);
    }
    if(datos[0] == 0){
        datos = simetrias3x3(tablero, filaFicha, columnaFicha);
    }
    if(datos[0] == 0){
        datos = simetrias2x2(tablero, filaFicha, columnaFicha);
    }
    return datos;
}

//Método general de detección de simetrías utilizado en los métodos
posteriores
public static int [] simetriaGeneral (String [][] matrizAuxiliar, int
simetria, int filaFicha, int columnaFicha,
int pasoFila, int pasoColumna, String [][] tablero, int []
datos){
    if((datos[0]==0) &&
(simetriaValida(tablero,simetria,filaFicha,columnaFicha))){
        for(int fila = filaFicha; fila < pasoFila; fila++){
            for(int columna = columnaFicha; columna < pasoColumna;
columnas++){
                matrizAuxiliar[fila - filaFicha][columna -
columnaFicha] = tablero[fila][columna];
            }
        }
        if(simetria == 2){
            datos = simetria2x2 (matrizAuxiliar, filaFicha,
columnaFicha);
        }
        if(simetria == 3){
            datos = simetria3x3 (matrizAuxiliar, filaFicha,
columnaFicha);
        }
    }
}

```

```

        }
        if(simetria == 4){
            datos = simetria4x4 (matrizAuxiliar, filaFicha,
columnaFicha);
        }
    }
    return datos;
}

//Método general de verificación de simetría utilizado en los métodos
posteriores
public static int [] verificacionSimetria(int [] datos, int simetria,
boolean condicion, int filaFicha, int columnaFicha){
    if(condicion){
        if(simetria == 2){
            datos[0] = 2;
            datos[1] = filaFicha;
            datos[2] = columnaFicha;
        }
        if(simetria == 3){
            datos[0] = 3;
            datos[1] = filaFicha;
            datos[2] = columnaFicha;
        }
        if(simetria == 4){
            datos[0] = 4;
            datos[1] = filaFicha;
            datos[2] = columnaFicha;
        }
    }
    return datos;
}

//Método para detectar las simetrías de 4x4
public static int [] simetria4x4 (String [][] aux4x4, int filaFicha,
int columnaFicha){
    int [] datos = {0,0,0}; //Array cuyo primer valor refiere a la
dimension de la simetría (0 es que no hay), segundo y tercer valor son la
posición
    if(filaFicha>=0 && columnaFicha>=0){
        try{
            datos = verificacionSimetria(datos, 4, aux4x4
[0][0].equals(aux4x4 [0][3]) && aux4x4 [0][1].equals(aux4x4 [0][2]) &&
aux4x4 [1][0].equals(aux4x4 [1][3]) && aux4x4
[1][1].equals(aux4x4 [1][2])) &&

```

```

        aux4x4 [2][0].equals(aux4x4 [2][3]) && aux4x4
[2][1].equals(aux4x4 [2][2]) &&
            aux4x4 [3][0].equals(aux4x4 [3][3]) && aux4x4
[3][1].equals(aux4x4 [3][2])), filaFicha, columnaFicha);

        datos = verificacionSimetria(datos, 4, datos[0] == 0 &&
aux4x4 [0][0].equals(aux4x4 [3][0]) && aux4x4 [1][0].equals(aux4x4
[2][0])) &&
                aux4x4 [0][1].equals(aux4x4 [3][1]) && aux4x4
[1][1].equals(aux4x4 [2][1]) &&
                    aux4x4 [0][2].equals(aux4x4 [3][2]) && aux4x4
[1][2].equals(aux4x4 [2][2]) &&
                        aux4x4 [0][3].equals(aux4x4 [3][3]) && aux4x4
[1][3].equals(aux4x4 [2][3])), filaFicha, columnaFicha);
            }
        catch(NullPointerException e){
            datos [0] = 0;
        }
    }
    return datos;
}

//Método para verificar las simetrías de 4x4 donde la pieza colocada
se encuentre en sus extremos
public static int [] simetrias4x4Lados(String [][] tablero, int
filaFicha, int columnaFicha){
    int [] datos = {0,0,0}; //el primer valor refiere a la dimensión
de la simetría (0 es que no hay), segundo y tercer valor son la posición
String [][] aux4x4 = new String [4][4];

    if(simetriaValida(tablero,4,filaFicha,columnaFicha)){
        for (int fila = filaFicha; fila < (filaFicha + 4); fila++) {
            for (int columna = columnaFicha; columna < (columnaFicha
+ 4); columna++) {
                aux4x4[fila - filaFicha][columna - columnaFicha] =
tablero[fila][columna];
            }
        }
        datos = simetria4x4 (aux4x4, filaFicha, columnaFicha);
    }

    datos = simetriaGeneral(aux4x4, 4, filaFicha-3, columnaFicha,
filaFicha+1, columnaFicha+4, tablero, datos);
    datos = simetriaGeneral(aux4x4, 4, filaFicha, columnaFicha-3,
filaFicha+4, columnaFicha+1, tablero, datos);
}

```

```

        datos = simetriaGeneral(aux4x4, 4, filaFicha-3, columnaFicha-3,
filaFicha+1, columnaFicha+1, tablero, datos);

        for (int fila = 0; fila < 3; fila++) {
            for (int columna = 0; columna < 3; columna++) {
                try{
                    if(aux4x4[fila][columna].equals(" ")){
                        datos [0] = 0;
                    }
                }catch (Exception e){
                    datos [0] = 0;
                }
            }
        }
        return datos;
    }

    //Método para verificar el resto de las simetrías de 4x4 que no se
    //verificaron en el método simetrias4x4Lados(String[][][], int, int)
    public static int [] simetrias4x4Medio(String [][] tablero, int
filaFicha, int columnaFicha){
        int [] datos = {0,0,0}; //Array cuyo primer valor refiere a la
        //dimensión de la simetría (0 es que no hay), segundo y tercer valor son la
        //posición
        String [][] aux4x4 = new String [4][4];
        if(simetriaValida(tablero,4,(filaFicha-1),(columnaFicha-1))){
            for (int fila = (filaFicha-1); fila < (filaFicha + 3);
fila++) {
                for (int columna = (columnaFicha-1); columna <
(columnaFicha + 3); columna++) {
                    aux4x4[fila - (filaFicha-1)][columna - (columnaFicha-
1)] = tablero[fila][columna];
                }
            }
            datos = simetria4x4 (aux4x4, (filaFicha-1), (columnaFicha-
1));
        }
        datos = simetriaGeneral(aux4x4, 4, filaFicha-2, columnaFicha-1,
filaFicha+2, columnaFicha+3, tablero, datos);
        datos = simetriaGeneral(aux4x4, 4, filaFicha-2, columnaFicha-2,
filaFicha+2, columnaFicha+2, tablero, datos);
        datos = simetriaGeneral(aux4x4, 4, filaFicha-1, columnaFicha-2,
filaFicha+3, columnaFicha+2, tablero, datos);
    }
}

```

```

        for (int fila = 0; fila < 4; fila++) {
            for (int columna = 0; columna < 4; columna++) {
                try{
                    if(aux4x4[fila][columna].equals(" ")){
                        datos [0] = 0;
                    }
                }catch (Exception e){
                    datos [0] = 0;
                }
            }
        }
        return datos;
    }

    //Método para detectar las simetrías de 3x3
    public static int [] simetria3x3 (String [][] aux3x3, int filaFicha,
int columnaFicha){
        int [] datos = {0,0,0}; //Array cuyo primer valor refiere a la
dimension de la simetria (0 es que no hay), segundo y tercer valor son la
posición
        if(filaFicha>=0 && columnaFicha>=0){
            try{
                datos = verificacionSimetria (datos, 3,
aux3x3[0][0].equals(aux3x3[0][2]) && aux3x3[1][0].equals(aux3x3[1][2]) &&
aux3x3[2][0].equals(aux3x3[2][2]), filaFicha,
columnaFicha);
                datos = verificacionSimetria (datos, 3,
aux3x3[0][0].equals(aux3x3[2][0]) && aux3x3[0][1].equals(aux3x3[2][1]) &&
aux3x3[0][2].equals(aux3x3[2][2]), filaFicha,
columnaFicha);
            }
            catch(NullPointerException e){
                datos [0] = 0;
            }
        }
        return datos;
    }

    //Método para verificar las simetrías de 3x3 respecto a la última
pieza colocada
    public static int [] simetrias3x3(String [][] tablero, int filaFicha,
int columnaFicha){

```

```

        int [] datos = {0,0,0}; //Array cuyo primer valor refiere a la
dimension de la simetria (0 es que no hay), segundo y tercer valor son la
posición
        String [][] aux3x3 = new String [3][3];
        if(simetriaValida(tablero,3,filaFicha,columnaFicha)){
            for (int fila = filaFicha; fila < (filaFicha + 3); fila++) {
                for (int columna = columnaFicha; columna < (columnaFicha
+ 3); columna++) {
                    aux3x3[fila - filaFicha][columna - columnaFicha] =
tablero[fila][columna];
                }
            }
            datos = simetria3x3 (aux3x3, filaFicha, columnaFicha);
        }

        datos = simetriaGeneral(aux3x3, 3, filaFicha-2, columnaFicha,
filaFicha+1, columnaFicha+3, tablero, datos);
        datos = simetriaGeneral(aux3x3, 3, filaFicha-2, columnaFicha-2,
filaFicha+1, columnaFicha+1, tablero, datos);
        datos = simetriaGeneral(aux3x3, 3, filaFicha, columnaFicha-2,
filaFicha+3, columnaFicha+1, tablero, datos);
        datos = simetriaGeneral(aux3x3, 3, filaFicha-1, columnaFicha,
filaFicha+2, columnaFicha+3, tablero, datos);
        datos = simetriaGeneral(aux3x3, 3, filaFicha-1, columnaFicha-2,
filaFicha+2, columnaFicha+1, tablero, datos);
        datos = simetriaGeneral(aux3x3, 3, filaFicha-2, columnaFicha-1,
filaFicha+1, columnaFicha+2, tablero, datos);
        datos = simetriaGeneral(aux3x3, 3, filaFicha, columnaFicha-1,
filaFicha+3, columnaFicha+2, tablero, datos);
        datos = simetriaGeneral(aux3x3, 3, filaFicha-1, columnaFicha-1,
filaFicha+2, columnaFicha+2, tablero, datos);

        for (int fila = 0; fila < 3; fila++) {
            for (int columna = 0; columna < 3; columna++) {
                try{
                    if(aux3x3[fila][columna].equals(" ")){
                        datos [0] = 0;
                    }
                }
                catch (Exception e){
                    datos [0] = 0;
                }
            }
        }
    }
    return datos;
}

```

```
}

//Método para detectar las simetrías de 2x2
public static int [] simetria2x2(String [][] aux2x2, int filaFicha,
int columnaFicha){
    int [] datos = {0,0,0}; //Array cuyo primer valor refiere a la
dimension de la simetria (0 es que no hay), segundo y tercero son la
posición
    if(filaFicha>=0 && columnaFicha>=0){
        try{
            datos = verificacionSimetria(datos, 2, aux2x2
[0][0].equals(aux2x2 [0][1]) && aux2x2 [1][0].equals(aux2x2 [1][1]),
filaFicha, columnaFicha);
        }
        catch(Exception e){
            datos [0] = 0;
        }
        try{
            datos = verificacionSimetria(datos, 2, aux2x2
[0][0].equals(aux2x2 [1][0]) && aux2x2 [0][1].equals(aux2x2 [1][1]),
filaFicha, columnaFicha);
        }
        catch(Exception e){
            datos [0] = 0;
        }
    }
    return datos;
}

//Método para verificar las simetrías de 2x2 respecto a la última
pieza colocada
public static int [] simetrias2x2(String [][] tablero, int filaFicha,
int columnaFicha){
    int [] datos = {0,0,0}; //Array cuyo primer valor refiere a la
dimension de la simetria (0 es que no hay), segundo y tercero son la
posición
    String [][] aux2x2 = new String [2][2];
    if(simetriaValida(tablero,2,filaFicha,columnaFicha)){
        for (int fila = filaFicha; fila < (filaFicha + 2); fila++) {
            for (int columna = columnaFicha; columna < (columnaFicha
+ 2); columna++) {
                aux2x2[fila - filaFicha][columna - columnaFicha] =
tablero[fila][columna];
            }
        }
    }
}
```

```

        datos = simetria2x2 (aux2x2, filaFicha, columnaFicha);
    }

    datos = simetriaGeneral(aux2x2, 2, filaFicha-1, columnaFicha,
filaFicha+1, columnaFicha+2, tablero, datos);
    datos = simetriaGeneral(aux2x2, 2, filaFicha-1, columnaFicha-1,
filaFicha+1, columnaFicha+1, tablero, datos);
    datos = simetriaGeneral(aux2x2, 2, filaFicha, columnaFicha-1,
filaFicha+2, columnaFicha+1, tablero, datos);

    for (int fila = 0; fila < 2; fila++) {
        for (int columna = 0; columna < 2; columna++) {
            try{
                if(aux2x2[fila][columna].equals(" ")){
                    datos [0] = 0;
                }
            }catch (Exception e){
                datos [0] = 0;
            }
        }
    }
    return datos;
}

//Método para verificar si una simetría es válida
public static boolean simetriaValida (String[][] matriz, int
dimensionSimetria, int fila, int columna){
    boolean validez = false;
    boolean hayBlancas = false;
    boolean hayNegras = false;
    if((fila + (dimensionSimetria - 1)) < matriz.length && (columna +
(dimensionSimetria - 1)) < matriz[0].length){
        validez=true;
    }
    if(fila>=0 && columna>=0 && validez){
        for (int f = fila; f < (fila + dimensionSimetria); f++) {
            for (int c = columna; c < (columna + dimensionSimetria);
c++) {
                if(matriz[f][c].equals(" ")){
                    validez = false;
                }
                if(matriz[f][c].equals("B")){
                    hayBlancas = true;
                }
                if(matriz[f][c].equals("N")){

```



```

        if (columna != matrizfila.length - 1) {
            System.out.print(separacion);
        }
    }
    System.out.println("\n");
}
}

//Método para decorar una matriz dada a modo del ejemplo presentado y
mostrarla en pantalla
public static void imprimirMatriz(String[][] matriz, String [][] panel, int filaP, int columnaP, boolean estaPanel, boolean vacio) {
    String[][] matrizNueva = new String[matriz.length * 2 +
1][matriz[0].length * 2 + 1];
    String[] array = arrayDeMatriz(matriz);
    int posicionArray = 0;
    for (int fila = 0; fila < matrizNueva.length; fila++) {
        for (int columna = 0; columna < matrizNueva[0].length;
columna++) {
            if (fila % 2 == 0) {
                if(columna % 2 == 0){
                    matrizNueva[fila][columna] = "+-+";
                }
                if(columna % 2 !=0){
                    matrizNueva[fila][columna] = "-+-";
                    if(columna == matrizNueva[0].length - 1){
                        matrizNueva[fila][columna] = "+";
                    }
                }
            }
            else {
                if (columna % 2 == 0) {
                    matrizNueva[fila][columna] = " | ";
                } else {
                    matrizNueva[fila][columna] =
array[posicionArray];
                    if (vacio){
                        if (estaPanel && filaP <= fila && fila <=
(filaP + panel.length*2 + 1) && columnaP <= columna && columna <
(columnaP + panel[0].length*2 + 1)){
                            matrizNueva[fila][columna] = " ";
                        }
                    }
                    posicionArray++;
                }
            }
        }
    }
}

```

```

        }
    }
}
if(estaPanel){
    matrizNueva =
Juego.colocarString(matrizNueva, filaP*2, columnaP*2, (panel.length*2)+1, "#");
}
mostrarMatriz(matrizNueva, " ");
}

//Método para llenar un array con información brindada por una matriz
dada
public static String[] arrayDeMatriz(String[][] matriz) {
    String[] array = new String[matriz.length * matriz.length];
    int posicionArray = 0;
    for (int fila = 0; fila < matriz.length; fila++) {
        for (int columna = 0; columna < matriz.length; columna++) {
            array[posicionArray] = matriz[fila][columna];
            posicionArray++;
        }
    }
    return array;
}

//Método para verificar si se puede colocar una ficha en la posición
indicada
public static boolean posicionValida (String[][] matriz, int fila,
int columna){
    boolean validez = false;
    try{
        if(matriz[fila][columna].equals(" ") && fila<matriz.length &&
columna<matriz[0].length)
            validez=true;
    }
    catch(Exception e){}
    return validez;
}
}

```

Fin de Clase Tablero

Clase Interfaz:

```
for (int i = 0; i < 2; i++) {
```

```
System.out.println("
```

");

}

```
System.out.println(" " );
```

```
System.out.println("\n
```

Les damos la

bienvenida a este juego único en su clase");

```
System.out.println("\nPara comenzar a jugar, le  
solicitaremos el ingreso de datos del Jugador y las condiciones de juego  
que desee");
```

}

```
//Método para solicitar dimensión de un tablero  
public static int pedidoMatrizTablero(String titulo, String  
mensaje, int valor1, int valor2, int valor3) {  
    Scanner in = new Scanner(System.in);  
    boolean dimensionCorrecta = false;  
    int dimension = 0;
```

```
System.out.println(titulo);
while (!dimensionCorrecta) {
    dimension = verificacionEntero("\nCuál es la dimensión
del tablero en la que desea jugar? \nNota: Se aceptarán tableros de " +
mensaje + " piezas de lado solamente.",
    "\nDebe ingresar " + mensaje + " según sean las
dimensiones deseadas. \nIngrese nuevamente, por favor.", "");
    if ((dimension != valor1) && (dimension != valor2) &&
dimension != valor3) {
        System.out.println("\nDebe ingresar valores " +
mensaje + " refiriendo a la dimension que desee para el tablero.");
    }
    if ((dimension == valor1) || (dimension == valor2) ||
(dimension == valor3)) {
        System.out.println("\nDimension asignada.\n");
        dimensionCorrecta = true;
    }
}
return dimension;
}

//Método para la solicitud de cadenas de caracteres o Strings
public static String pedidoCaracteres(String pedido, String
devolucion) {
    Scanner caracteres = new Scanner(System.in);
    System.out.println("\n" + pedido);
    String dato = caracteres.nextLine();
    System.out.println("\n" + devolucion);
    return dato;
}
```



```

//Método para pedir y mostrar datos en el Menu del Juego
public static void menuJuego(int turno, String opcion3) {

System.out.println("\n\n" + "JUEGO DE TIC-TAC-TOE" + "\n");
if(turno%2!=0)
    System.out.println("■ JUGADOR 1, este es su turno:" + "\n");
if(turno%2==0)
    System.out.println("■ JUGADOR 2, este es su turno:" + "\n");
System.out.println("■ RECUERDE que su turno será válido solo si ingresa sus intenciones de la forma a continuación explicitada" + "\n");
+ "\n■ * Para ingresar una ficha al tablero en una POSICIÓN VACÍA ingrese de la forma 'J-Fila-Columna' (sin guiones) refiriéndose a la fila y columna donde quiera ingresar la ficha respectivamente." + "\n";
+ "\n■ ** Para dejar la partida y así rendirse ingrese 'X'. " + "\n" + opcion3
+ "\n■ RECUERDE: Las filas deben ser ingresadas en caracteres (estando ordenadas comenzando por A) y las y las columnas en números (comenzando por el 1).");

System.out.println("■ JUEGO DE TIC-TAC-TOE" + "\n");
}

```

```
        System.out.println("\n\nIngrese su intención a continuación  
y presione ENTER");  
    }  
  
    //Método para mostrar y solicitar datos correspondientes al fin  
de la partida (set)
```

```
    public static int mensajeFinSet() {  
        boolean opcionCorrecta = false;
```

```
System.out.println("\n\n_____  
_____|");  
System.out.println("|| QUÉ DESEA HACER A CONTINUACIÓN?  
||"  
    + "\n|| 1. Jugar nuevamente.  
||"  
    + "\n|| 2. Calificar este juego o sobreescribir una  
calificación previa. ||"  
    + "\n||     Sus opiniones son siempre tenidas en  
cuenta. ||"  
    + "\n|| 3. Abandonar el juego.  
||");
```

```
System.out.println("_____  
_____|");  
int opcion = Interfaz.verificacionEntero("", "Por favor,  
inserte una opción válida.\nIngrese nuevamente a continuación", "Dato  
recibido.");  
while (!opcionCorrecta) {  
    if (opcion != 1 && opcion != 2 && opcion != 3) {
```

```
        System.out.println("Ingrese una opcion correcta  
dentro del rango brindado. Digite una vez más.");  
        opcion = Interfaz.verificacionEntero("", "Por  
favor, inserte una opción válida.\nIngrese nuevamente a continuación",  
"Dato recibido.");  
    } else {  
        opcionCorrecta = true;  
    }  
}  
return opcion;  
}  
  
//Método para el Ingreso de datos del Jugador  
public static ArrayList<Jugador> registroJugadores(ArrayList  
<Jugador> listaTodosJugadores) {  
    boolean jugadorSolitario = true;  
    ArrayList<Jugador> listaAuxiliar = new ArrayList<>();  
    ArrayList<String> califPredeterminada=new ArrayList<>();  
    califPredeterminada.add("Ninguna Calificación");  
    while(jugadorSolitario){  
        listaAuxiliar.clear();  
        for (int cantidadJugadores = 1; cantidadJugadores < 3;  
cantidadJugadores++) {  
            boolean edadCorrecta = false;  
            System.out.println("\nJUGADOR " + cantidadJugadores  
+ " _____ \n");  
            Jugador participante = new Jugador();  
            if (cantidadJugadores == 1) {  
                System.out.println("Jugará con las fichas  
BLANCAS");
```

```
    participante.setFicha("Blancas");

}

if (cantidadJugadores == 2) {
    System.out.println("Jugará con las fichas NEGRAS");
    participante.setFicha("Negras");
}

String nombreJugador = Interfaz.pedidoCaracteres("Ahora
ingrese su NOMBRE", "NOMBRE aceptado.");
participante.setNombre(nombreJugador);

while (!edadCorrecta) {
    int edadJugador = Interfaz.verificacionEntero("Por
favor, ingrese su EDAD a continuación", "Debe ingresar su edad en
enteros", "");
    participante.setEdad(edadJugador);
    if (edadJugador > 120 || edadJugador < 1) {
        System.out.println("No puede engañar a este
programa. Debe ingresar su VERDADERA edad");
    } else {
        edadCorrecta = true;
    }
}
System.out.println("Edad recibida");

String aliasJugador =
Interfaz.pedidoCaracteres("Ingrese su ALIAS", "ALIAS ingresado.");
participante.setAlias(aliasJugador);
participante.setCalificaciones(califPredeterminada);
listaAuxiliar.add(participante);
}
```

```
        if(listaAuxiliar.get(0).equals(listaAuxiliar.get(1))){
            System.out.println("\nNo puede jugar contra si mismo.");
            Ingrese nuevamente");
        }
        else
            jugadorSolitario = false;
    }
    if(!listaTodosJugadores.isEmpty()){
        Iterator <Jugador> it = listaTodosJugadores.iterator();
        while(it.hasNext()){
            Jugador jugador = it.next();
            if(jugador.equals(listaAuxiliar.get(0))){
                listaAuxiliar.get(0).setPartidasJugadas(jugador.getPartidasJugadas());
                listaAuxiliar.get(0).setPartidasGanadas(jugador.getPartidasGanadas());
            }
            if(jugador.equals(listaAuxiliar.get(1))){
                listaAuxiliar.get(1).setPartidasJugadas(jugador.getPartidasJugadas());
                listaAuxiliar.get(1).setPartidasGanadas(jugador.getPartidasGanadas());
            }
        }
        System.out.println("\nMuchas gracias. Sus datos están
siendo procesados.");
        return listaAuxiliar;
    }
}
```

```
//Método para brindarle al Jugador la posibilidad de cambiar  
sus datos una vez ingresados  
public static ArrayList<Jugador> cambioDatosJugador(Juego  
unJuego, ArrayList<Jugador> jugadores, ArrayList <Jugador>  
listaTodosJugadores) {  
    Scanner in = new Scanner(System.in);  
    int opcion = 0;  
    while (opcion != 1) {  
        System.out.println("\n Los datos ingresados fueron: " +  
"\n" + jugadores);  
        opcion = Interfaz.verificacionEntero("Están de acuerdo  
con los datos ingresados?" + "\n 1- Si" + "\t 2- No", "Debe ingresar  
enteros 1 o 2.", "");  
        if (opcion == 1) {  
            System.out.println("Datos procesados.");  
        }  
        if (opcion == 2) {  
            System.out.println("Les solicitaremos nuevamente  
sus datos");  
  
        unJuego.setListaJugadores(Interfaz.registroJugadores(listaTodosJugadores)  
    );  
    }  
    if (opcion != 1 && opcion != 2) {  
        System.out.println("Debe ingresar opcion 1 o 2  
únicamente. Disculpe las molestias.\n");  
    }  
}  
return jugadores;  
}
```

```
//Método para pasar un String a número entero
public static int pasarANumero(String letra) {
    char l = letra.charAt(0);
    int num = l - 64;
    if (num < 0) {
        num = Integer.parseInt("") + 1;
    }
    return num;
}

//Método para verificar si un carácter es letra
public static boolean esLetra(String letra) {
    boolean esLetra = true;
    char l = letra.charAt(0);
    int num = l - 64;
    if (num < 0) {
        esLetra = false;
    }
    return esLetra;
}

//Método para verificar si un carácter es número
public static boolean esNumero(String numero) {
    boolean esNumero = false;
    char l = numero.charAt(0);
    int num = l - 64;
    if (num < 0) {
        esNumero = true;
    }
}
```

```
        return esNúmero;  
    }
```

//Método que devuelve un ArrayList de Jugadores dado en forma ordenada

```
    public static ArrayList<Jugador>  
devolverJugadoresOrdenados(ArrayList<Jugador> listaJugadores) {  
        //Criterio Interno, Inner Class Anónima (se define el criterio localmente pero sin dar el nombre)  
        Collections.sort(listaJugadores, new Comparator() {  
            public int compare(Object param1, Object param2) {  
                Jugador participante1 = (Jugador) param1;  
                Jugador participante2 = (Jugador) param2;  
                int aux =  
participante1.getNombre().compareTo((Jugador)  
participante2.getNombre());  
                if (aux == 0) {  
                    aux =  
participante1.getAlias().compareTo((Jugador) participante2.getAlias());  
                }  
                return aux;  
            }  
        });  
        return listaJugadores;  
    }  
  
    //Método que despliega las opciones del Fin del Set (Jugar nuevamente, calificar o abandonar el juego)
```

```
public static boolean [] opcionesFinSet(Juego game, int
opcionJuego, ArrayList<Jugador> listaTodosJugadores){
    // Inicialización de un Array de booleanos que
determina el valor de la variable finSet(posicion 1) y gameOver (posicion
2) del main
    boolean [] finPartidas = new boolean[2];
    if (opcionJuego == 1) {
        System.out.println("SU NUEVO JUEGO
COMENZARÁ EN BREVE...\\n\\n");
        finPartidas[0] = true;
    }
    if (opcionJuego == 2) {
        game.calificarJuego(listaTodosJugadores,
game.getListaJugadores());
    }
    if (opcionJuego == 3) {
        System.out.println("Muchas gracias por
jugar. Lo esperamos nuevamente pronto!!!\"");
        finPartidas[0] = true;
        finPartidas[1] = true;
    }
    return finPartidas;
}

//Método que verifica la confirmación del despliegue del
Ranking Mundial y lo verifica en caso afirmativo
public static boolean opcionesRankingMundial(ArrayList<Jugador>
listaTodosJugadores, boolean optionCorrecta, int option){
    if (option == 1) {
        System.out.println("RANKING MUNDIAL:");
    }
}
```

```
        listaTodosJugadores =  
Interfaz.devolverJugadoresOrdenados(listaTodosJugadores);  
  
System.out.println(listaTodosJugadores);  
        optionCorrecta = true;  
    }  
    if (option == 2) {  
        optionCorrecta = true;  
    }  
    if (option != 1 && option != 2) {  
        System.out.println("Debe ingresar un  
valor válido y apropiado (1 o 2).\nIngrese nuevamente, por favor.");  
    }  
    return optionCorrecta;  
}  
  
//Método que muestra quien fue el ganador de la partida y el  
consiguiente incremento de las partidas ganadas del vencedor  
public static void mensajeGanador(Juego game, int  
puntosJugador1, int puntosJugador2, boolean rendicion){  
    if (puntosJugador1 > puntosJugador2 && !rendicion) {  
        System.out.println("EL GANADOR FUE EL JUGADOR  
1!!!!!!\n\n");  
  
game.getListaJugadores().get(0).setPartidasGanadas(game.getListaJugadores()  
().get(0).getPartidasGanadas()+1);  
    }  
    if (puntosJugador1 < puntosJugador2 && !rendicion)  
{
```

```
System.out.println("EL GANADOR FUE EL JUGADOR  
2!!!!!\n\n");  
  
game.getListaJugadores().get(0).setPartidasGanadas(game.getListaJugadores()  
(()).get(0).getPartidasGanadas()+1);  
}  
if (puntosJugador1 == puntosJugador2 && !rendicion)  
{  
    System.out.println("EMPATE !!!!!!\n\n");  
}  
}  
  
}
```

Fin de Clase Interfaz