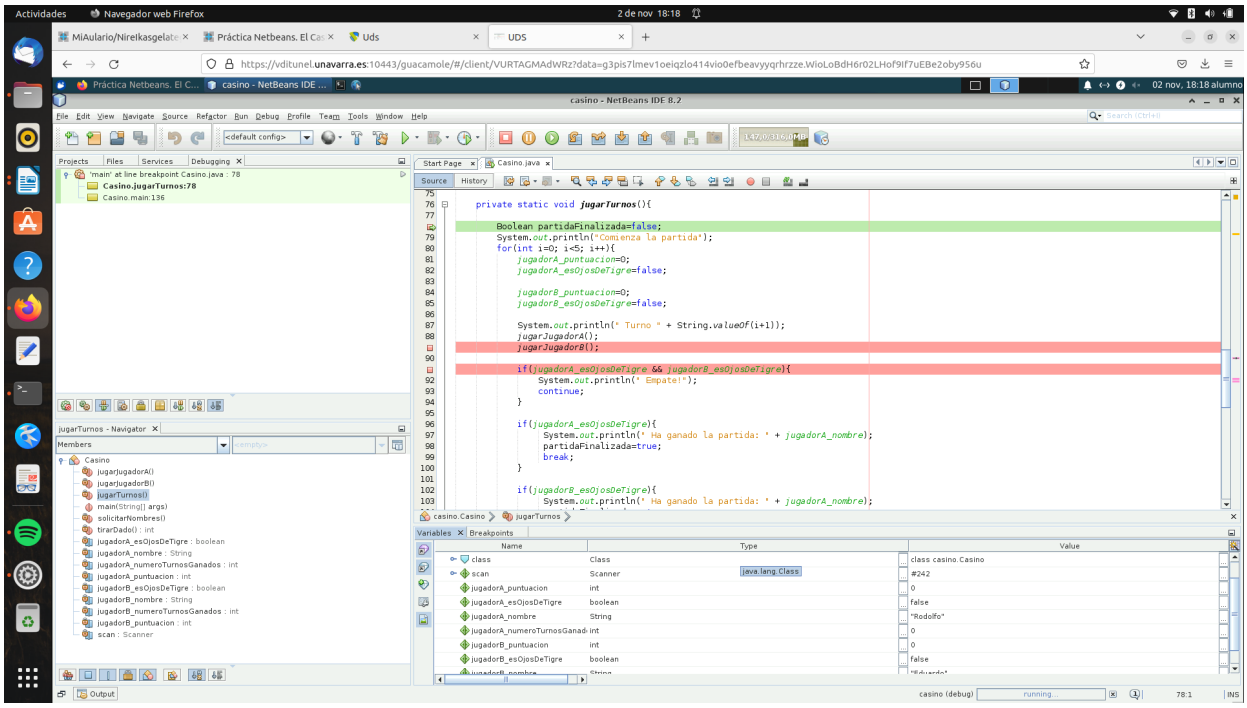
PRÁCTICA 5: EL CASINO:

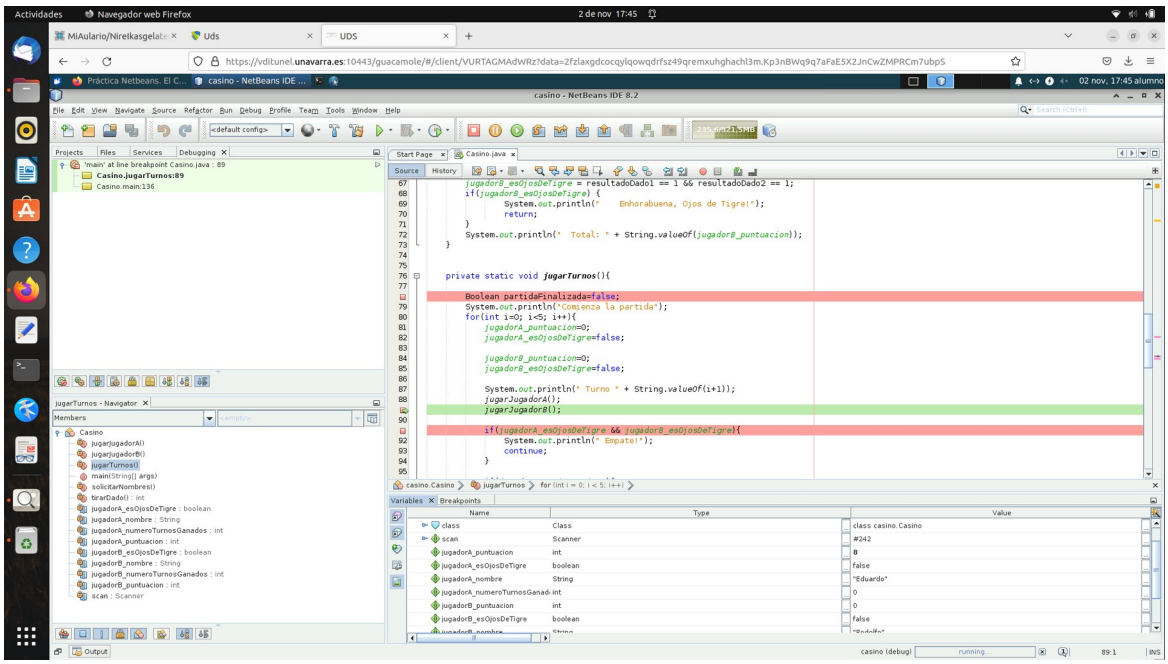
4 EJERCICIOS:

Ejercicio 1: Utilice el depurador para inspeccionar el resultado de la ejecución del código.

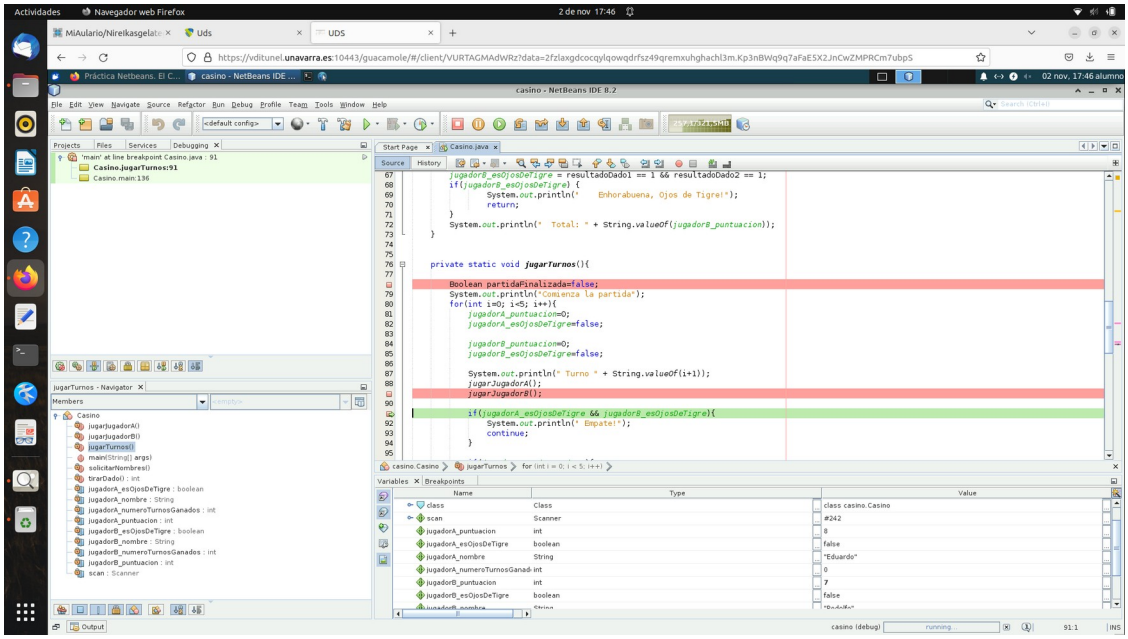
En primer lugar, he añadido tres break-points a jugarTurnos(). Para añadir el break-point únicamente hay que seleccionar el número de la línea, e instantáneamente se pintará de rojo dicha instrucción. A continuación hay que seleccionar Debug Project en la sección Debug. El primero break-point. lo sitúo en la primera instrucción de jugarTurnos() y así poder visualizar el valor inicial de la puntuación de ambos jugadores (0).



En segundo lugar, se ha añadido un break-point en la instrucción jugarJugadorB() para así poder visualizar el cambio de valor de la variable de la puntuación del jugador A (8), que se ejecuta en la línea anterior al break-point.



Por último, se sitúa el último break-point despues de jugarJugadorB(), para poder ver cómo se ha modificado la puntuación del jugadorB (7).



Ejercicio 2: Reescriba la solución haciendo uso del paradigma Orientado a Objetos.

Clase	Atributos	Métodos	Responsabilidad
Casino	Jugador, Jugador, Partida	jugarTurnos()	Realiza los 5 turnos de la partida siempre y cuando no haya Ojos de Tigre
Partida	Jugador, Jugador, finPartida(booleano), Dado,Dado	JugarTurno(), partidaFinalizada()	Realiza un turno de la partida. Se comprueba que no se haya terminado la partida
Jugador	Nombre, puntuación, esOjosDeTigre, numeroTurnosGanados	obtenerNombre(),obtenerPuntuacion(), sumarPuntuacion()...	Devuelve atributos de jugador, reinicia atributos, y los modifica
Dado	Resultado	TirarDado(), obtenerResultado()	Tira el dado y devuelve el resultado

Ejercicio 3: Comparación entre diferentes paradigmas.

Tras comparar ambos códigos, tanto el paradigma imperativo y el orientado a objetos, resulta mucho más sencillo el paradigma orientado a objetos al tener un código mucho más dividido entre distintas clases y modularizado. Las ventajas, como se ha mencionado anteriormente es que al estar todo más dividido es mucho más fácil encontrar un fallo y entender realmente lo que te realiza el programa. Una posible desventaja es que te requiere hacer un análisis previo para dividir todo el código en distintas clases.