ÓSCAR ÁLVAREZ LUCAS

Practica 3 Programación

26 DE SEPTIEMBRE DE 2025

1º DAW

Francisco Romero Vargas

oalvluc0702/Programacion: Repositorio de la asignatura programación

ÍNDICE

ENLACE AL REPOSITORIO

oalvluc0702/Programacion: Repositorio de la asignatura programación

1.ARRAYS BIDIMENSIONALES

EJERCICIO 1

DEFINE UN ARRAY DE NÚMEROS TIPO DOUBLE DE 3 FILAS POR 7 COLUMNAS CON NOMBRE DOUB Y ASIGNA LOS VALORES SEGÚN LA SIGUIENTE TABLA. MUESTRA EL CONTENIDO DE TODOS LOS ELEMENTOS DEL ARRAY DISPUESTOS EN FORMA DE TABLA COMO SE MUESTRA EN LA FIGURA.

Array num	Columna 0	Columna 1	Columna 2	Columna 3	Columna 4	Columna 5
Fila 0	0	30	2			5
Fila 1	75				0	
Fila 2			-2	9		11

Para este ejercicio he decidido utilizar un número “PlaceHolder” para hacer la sustitución con un espacio en blanco para que no saliera 0

```
C:\Users\daw\.jdks\openjdk-25\bin\java.exe "-javaagent:E:\IntelliJ IDEA Community Edition 2
Array      Columna 0   Columna 1   Columna 2   Columna 3   Columna 4   Columna 5
FILA   0 ||0         ||30        ||2         ||          ||          ||5         ||
FILA   1 ||75        ||          ||          ||          ||0         ||          ||
FILA   2 ||          ||          ||-2        ||9         ||          ||11        ||

Process finished with exit code 0
```

EJERCICIO 2

ESCRIBE UN PROGRAMA QUE SOLICITE 20 NÚMEROS ENTEROS. ESTOS NÚMEROS DEBEMOS DE INTRODUCIRLO EN UN ARRAY DE 4 FILAS POR 5 COLUMNAS. EL PROGRAMA MOSTRARÁ LAS SUMAS PARCIALES DE FILAS Y EN LAS COLUMNAS EL MAYOR NÚMERO DE LA COLUMNA. LA SUMA TOTAL DEBE APARECER EN LA ESQUINA INFERIOR DERECHA.

					Σ fila 0
					Σ fila 0
					Σ fila 0
					Σ fila 0
Σ columna 0	Σ columna 1	Σ columna 2	Σ columna 3	Σ columna 4	**TOTAL**

Para este ejercicio he decidido usar los bucles para calcular los sumatorio, el máximo de columna y el total, no hay vuelta de hoja.

```
|1        ||2        ||3        ||1        ||2        ||9        |
|3        ||4        ||5        ||6        ||7        ||25       |
|3        ||2        ||1        ||4        ||5        ||15       |
|6        ||7        ||4        ||3        ||2        ||22       |
|6        ||7        ||5        ||6        ||7        ||102      |
Process finished with exit code 0
```

EJERCICIO 3

MODIFICA EL PROGRAMA ANTERIOR DE TAL FORMA QUE LOS NÚMEROS QUE SE INTRODUCEN EN EL ARRAY SE GENEREN DE FORMA ALEATORIA (VALORES ENTRE 1 Y 1999).

Ahora para hacer que se generen de manera aleatorio, solo he modificado que en vez de meter el número por pantalla tu mismo, se generan automáticamente con el método Math.random().

```
C:\Users\daw\.jdks\openjdk-25\bin\java.exe "-javaagent:E:\IntelliJ IDEA Community
|447      ||1682      ||1436      ||1193      ||1997      ||6755      |
|1854     ||1946      ||1614      ||1158      ||1246      ||7818      |
|1857     ||564       ||1339      ||19        ||579       ||4358      |
|1493     ||993       ||14        ||868       ||1978      ||5346      |
|1857     ||1946      ||1614      ||1193      ||1997      ||32884     |
Process finished with exit code 0
```

EJERCICIO 4

MODIFICA EL PROGRAMA ANTERIOR DE TAL FORMA QUE LAS SUMAS PARCIALES Y LA SUMA TOTAL APAREZCAN EN LA PANTALLA CON UN PEQUEÑO RETRASO, DANDO LA IMPRESIÓN DE QUE EL ORDENADOR SE QUEDA “PENSANDO” ANTES DE MOSTRAR LOS NÚMEROS.

Para este ejercicio el truco era usar el thread.sleep para simular el parón, tuve que buscarlo en la documentación de Java y en StackOverflow para solucionar el problema.

```
|1209      ||1820      ||725       ||1344      ||1859      |
```

```
C:\Users\daw\.jdks\openjdk-25\bin\java.exe "-javaagent:E:\IntelliJ IDEA Co
|1209      ||1820      ||725       ||1344      ||1859      ||6957      |
|1066      ||1288      ||1998      ||1709      ||1931      ||7992      |
|1644      ||613       ||290       ||1271      ||894       ||4712      |
|52        ||91        ||1210      ||1897      ||1805      ||5055      |
|1644      ||1820      ||1998      ||1897      ||1931      ||34006     |
Process finished with exit code 0
```

Es complicado representarlo con capturas, pero hace el retraso.

EJERCICIO 5

CREAR UN PROGRAMA QUE CUANDO SE LE INTRODUZCA NÚMEROS ENTEROS RELLENE UN ARRAY DE 6 FILAS POR 10 COLUMNAS CON NÚMEROS ENTEROS POSITIVOS COMPRENDIDOS ENTRE 0 Y 1000 (AMBOS INCLUIDOS). A CONTINUACIÓN, EL PROGRAMA DEBERÁ: DAR LA POSICIÓN DEL NÚMERO, MÁXIMO Y MÍNIMO, LA SUMA TOTAL DE TODAS LAS FILAS Y COLUMNAS, LA SUMA DE TODAS LAS COLUMNAS LA SUMA DE TODAS LAS FILAS.

Para hacer este ejercicio he usado muchos bucles recorriendo los arrays dimensionales y mucha cantidad de variables que hacían falta.

```
C:\Users\daw\.jdks\openjdk-25\bin\java.exe "-javaagent:E:\IntelliJ IDEA Communit
la suma de la fila 1 es 3819
la suma de la fila 2 es 5339
la suma de la fila 3 es 4584
la suma de la fila 4 es 5796
la suma de la fila 5 es 5968
la suma de la fila 6 es 4297
la suma de la columna 1 es 2567
la suma de la columna 2 es 3247
la suma de la columna 3 es 2967
la suma de la columna 4 es 3918
la suma de la columna 5 es 3041
la suma de la columna 6 es 2768
la suma de la columna 7 es 2883
la suma de la columna 8 es 1684
la suma de la columna 9 es 3983
la suma de la columna 10 es 2745
El número máximo es 985 y se encuentra en la posición de fila 4 columna 10
El número mínimo es 3 y se encuentra en la posición de fila 5 columna 8
La suma de todas las filas y columnas en total es 29803
Process finished with exit code 0
```

EJERCICIO 6

MODIFICA EL PROGRAMA ANTERIOR DE TAL FORMA QUE NO SE REPITA NINGÚN NÚMERO EN EL ARRAY ADEMÁS DE QUE TIENE QUE ESTAR COMPRENDIDO EN UN RANGO ENTRE 20-40.

Este ejercicio era complicado porque tenías que controlar que el bucle no se quedará ejecutándose infinitamente cuando ya no se quedase sin números que no se repitan del 20-40, para ello he implementado un contador de intentos máximos para que si llega a esas iteraciones salga del bucle y ejecute todo.

```
C:\Users\daw\.jdks\openjdk-25\bin\java.exe "-javaagent:E:\IntelliJ IDEA Community Edition 20
la suma de la fila 1 es 304
la suma de la fila 2 es 291
la suma de la fila 3 es 35
la suma de la fila 4 es 0
la suma de la fila 5 es 0
la suma de la fila 6 es 0
la suma de la columna 1 es 92
la suma de la columna 2 es 71
la suma de la columna 3 es 52
la suma de la columna 4 es 44
la suma de la columna 5 es 58
la suma de la columna 6 es 47
la suma de la columna 7 es 66
la suma de la columna 8 es 61
la suma de la columna 9 es 79
la suma de la columna 10 es 60
El número máximo es 40 y se encuentra en la posición de fila 1 columna 9
El número mínimo es 20 y se encuentra en la posición de fila 2 columna 5
La suma de todas las filas y columnas en total es 630
Process finished with exit code 0
```

EJERCICIO 7

MODIFICA EL PROGRAMA DEL EJERCICIO 6 PARA QUE: 1. LOS NÚMEROS NO SE REPITAN (COMO EN EL EJERCICIO ANTERIOR). 2. LOS NÚMEROS ESTÉN COMPRENDIDOS EN UN RANGO DINÁMICO (EL USUARIO INTRODUCE EL VALOR MÍNIMO Y MÁXIMO). 3. AL FINAL, EL PROGRAMA MUESTRE: LA MEDIA ARITMÉTICA DE TODOS LOS NÚMEROS DEL ARRAY. LA POSICIÓN DE TODOS LOS NÚMEROS PRIMOS QUE HAYA EN EL ARRAY. UNA REPRESENTACIÓN GRÁFICA EN CONSOLA DE CADA FILA, DONDE CADA NÚMERO SE REPRESENTE CON UN NÚMERO DE * PROPORCIONAL A SU VALOR DENTRO DEL RANGO DADO (POR EJEMPLO, SI EL RANGO ES 10-20 Y APARECE EL 15, SE MOSTRARÁN 5 *).

Este ejercicio tenía la complejidad de los asteriscos que al final lo he hecho con una fórmula para calcular la proporción en base a unos asteriscos máximos. Y reservando espacios dinámicamente para poder cuadrar la gráfica

Para los números primos he usado el método de un bucle hasta la raíz cuadrada, porque si encuentra un divisor exacto significa que tiene más divisores.

```
C:\Users\daw\.jdks\openjdk-25\bin\java.exe "-javaagent:E:\IntelliJ IDEA Community Edition 2025.2.1\lib\idea_rt.jar=53300" -Dfile.encoding=UT
Dime un número entero para el máximo
20
Dime un número entero para el mínimo
10
se ha encontrado un número primo, este es: 19 se encuentra en la Fila 1 y Columna 4
se ha encontrado un número primo, este es: 11 se encuentra en la Fila 1 y Columna 6
se ha encontrado un número primo, este es: 13 se encuentra en la Fila 1 y Columna 8
se ha encontrado un número primo, este es: 17 se encuentra en la Fila 2 y Columna 1

El número máximo es 20 y se encuentra en la posición de fila 1 columna 10
El número mínimo es 10 y se encuentra en la posición de fila 1 columna 7
La suma de todas las filas y columnas en total es 165
La media aritmética es: 15
|18       ||12       ||14       ||19       ||15       ||11       ||10       ||13       ||16       ||20       |
|17       ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        |
|0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        |
|0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        |
|0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        |
|0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        ||0        |

1   : ********    **          ****        *********   *****       *                     ***         ******      **********
2   : *******
3   :
4   :
5   :
6   :

Process finished with exit code 0
```

EJERCICIO 8

REALIZAR EL JUEGO DE LA “BÚSQUEDA DEL TESORO” DONDE SI TE ACERCAS AL TESORO TE VA AVISANDO DE QUE HAY UN TESORO CERCA, PERO AGREGANDO QUE TENEMOS DOS POSIBLES TESOROS EN EL JUEGO Y UNO DE ELLOS ES UN IMPOSTOR

Este programa es la búsqueda del tesoro, en el propio programa voy explicando en los comentarios para que sirve cada función y que hace exactamente, mi forma de plantearlo fue crear primero una función que me mostrase un tablero impreso de la longitud que tu le especifiques dentro de una variable en el código.

Dependiendo de lo que valga la variable dentro del array te va a imprimir una cosa u otra, 0 es la arena, 1 es el tesoro falso 2 es el tesoro que es verdadero y 3 es el personaje (en mi caso he elegido un loro por hacer referencia a que somos un pirata)

Luego definí una función que me ayudara a mostrar el mapa pero que no revelase los tesoros como lo hacía la otra función que ya tenía para mostrar todo el tablero completo.

Y ya por último hablando de funciones agregué 2 funciones, una para comprobar si el número de fila y columna es válido tanto para el inicio del juego como para el movimiento del propio jugador

Y otra para acabar la partida si la posición siguiente cuando se mueve es igual a un tesoro, ya sea falso o verdadero.

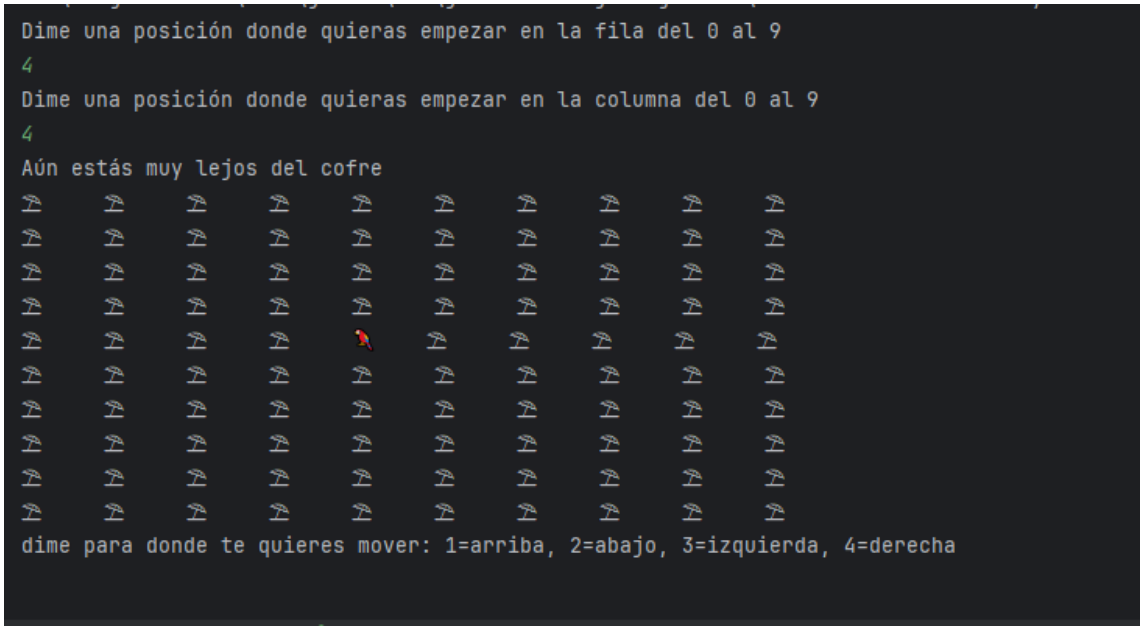
Las posiciones de los tesoros tenían que generarse automáticamente por lo cual hice 2 variables que almacenaran un valor aleatorio que equivaliese al número de casillas de la cual se compone el tablero.

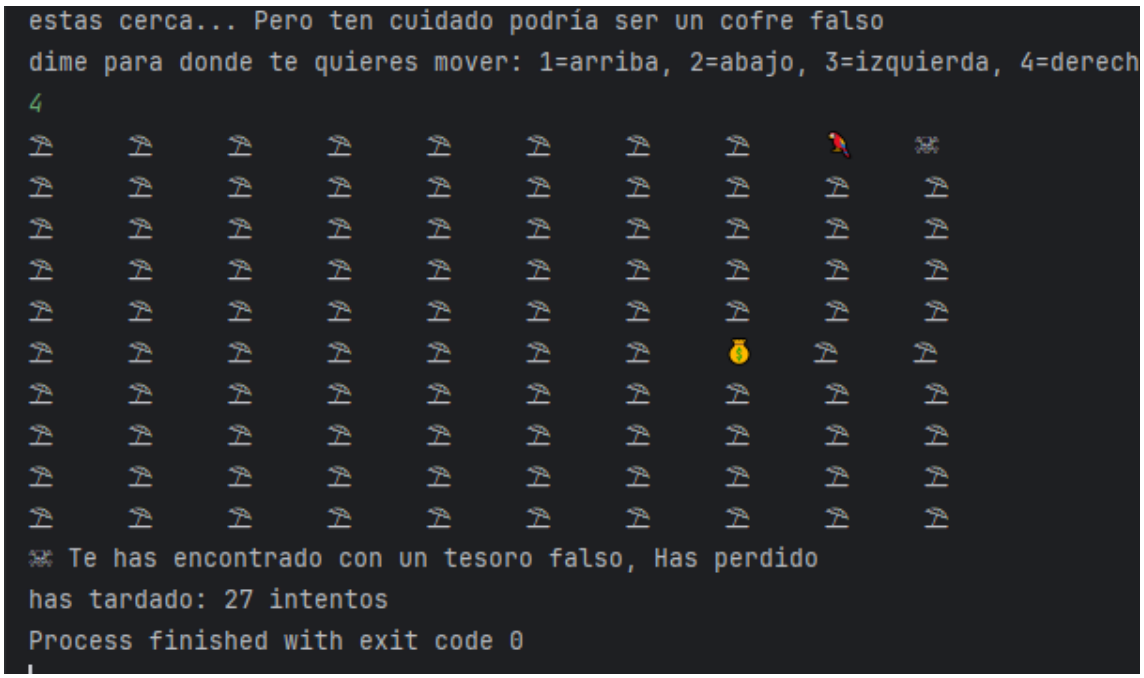
Si tienes la buena o mala suerte de caer en un tesoro, se llama a la función, te sale un mensaje de aviso de que has ganado o perdido la partida y te muestra el número de intentos que te ha costado para llegar al tesoro, los intentos es una variable que se va incrementando cada vez que realizas una acción para encontrar un tesoro como moverse por el mapa.

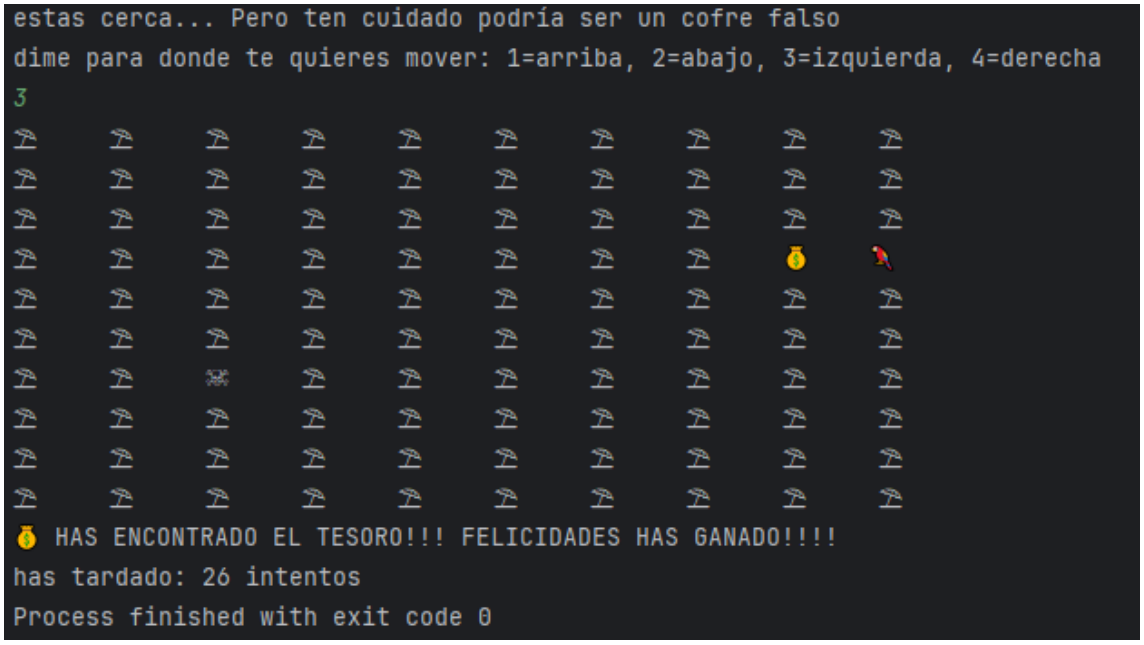
El movimiento lo he planteado con una estructura Switch case que restaba y sumaba posiciones comprobando si eran válidas con la función que ya tenía creada anteriormente para validar.

La distancia para que me diga si está lejos o está cerca la hice guardando la posición de los tesoros verdadero y falso y con una formula matemática que consiste en restar la distancia del tesoro y la distancia del jugador todo pasado a números enteros negativos y sumando los valores de la X y la Y se podía calcular la distancia real que había entre un jugador y cualquiera de los 2 tesoros.

Voy a poner unos ejemplos de las diferentes salidas que tiene el código







EJERCICIO 9

ESCRIBE UN PROGRAMA QUE, DADA UNA POSICIÓN EN UN TABLERO DE AJEDREZ, NOS DIGA A QUÉ CASILLAS PODRÍA SALTAR UN CABALLO QUE SE ENCUENTRA EN ESA POSICIÓN. COMO SE INDICA EN LA FIGURA, EL CABALLO SE MUEVE SIEMPRE EN FORMA DE L. EL TABLERO CUENTA CON 64 CASILLAS. LAS COLUMNAS SE INDICAN CON LAS LETRAS DE LA “A” A LA “H” Y LAS FILAS SE INDICAN DEL 1 AL 8.

Este programa me pide que haga un tablero de ajedrez y que le diga donde coloco al caballo para que me diga a que casillas puede saltar.

Plantear este ejercicio fue muy sencillo, primero necesitaba una matriz de 8x8 que contuviese las letras y columnas, por eso me ayudé de 2 arrays sencillos para luego juntarlos en un array bidimensional al cual llamé conjunto.

Igual que en el resto de los ejercicios tenía una función para mostrarme el tablero y que el jugador no se sintiera tan perdido.

Luego tenía otra función que me ayudaba a encontrar que el número que me había introducido el usuario fuera un número que existiese en el tablero de ajedrez y validase el posicionamiento de la pieza.

Y la función más importante de este programa al cual le escribí en 2 arrays los movimientos del caballo referenciados 1 por 1 por fila y columna, por ejemplo, en cuanto a fila el caballo se puede mover 2 filas para arriba o para abajo, que equivaldría a -2 o 2 y en columnas se puede mover una a la derecha y a la izquierda que equivale a 1 y -1.

Luego encuentra donde está el caballo y guarda su posición tanto en fila como columna y usamos esas posiciones para ir imprimiendo los movimientos sumándole a la posición en filas y posición en columnas los elementos del array.

Luego comprueba si ese movimiento se pasa del rango de la tabla, si se pasa del rango del tablero, no se imprimirá

Ahora voy a mostrar unas cuantas salidas del ejercicio para comprobar que efectivamente hace su función

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:E:\IntelliJ IDEA Community Edition 2025.2.2\lib\idea_rt.jar=49318" -Dfile
 .encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath E:\proyectos\Programacion\out\production\practica3
 ejercicio9
A8      B8      C8      D8      E8      F8      G8      H8
A7      B7      C7      D7      E7      F7      G7      H7
A6      B6      C6      D6      E6      F6      G6      H6
A5      B5      C5      D5      E5      F5      G5      H5
A4      B4      C4      D4      E4      F4      G4      H4
A3      B3      C3      D3      E3      F3      G3      H3
A2      B2      C2      D2      E2      F2      G2      H2
A1      B1      C1      D1      E1      F1      G1      H1
Dime la posición donde quieres introducir el caballo
d5
El caballo puede moverse a las siguientes posiciones E7 F6 F4 E3 C3 B4 B6 C7
Process finished with exit code 0
```

Si probamos poniendo el caballo en las esquinas, como podemos solo imprime los posibles movimientos

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:E:\IntelliJ IDEA Communi
 .encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classp
 ejercicio9
A8      B8      C8      D8      E8      F8      G8      H8
A7      B7      C7      D7      E7      F7      G7      H7
A6      B6      C6      D6      E6      F6      G6      H6
A5      B5      C5      D5      E5      F5      G5      H5
A4      B4      C4      D4      E4      F4      G4      H4
A3      B3      C3      D3      E3      F3      G3      H3
A2      B2      C2      D2      E2      F2      G2      H2
A1      B1      C1      D1      E1      F1      G1      H1
Dime la posición donde quieres introducir el caballo
a1
El caballo puede moverse a las siguientes posiciones B3 C2
Process finished with exit code 0
```

Y vamos a intentar introducirle un valor que no es válido

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:E:\IntelliJ IDEA Community Edition 2025.2.2\lib\idea_rt.jar=49449" -Dfile
 .encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath E:\proyectos\Programacion\out\production\practica3
 ejercicio9
A8      B8      C8      D8      E8      F8      G8      H8
A7      B7      C7      D7      E7      F7      G7      H7
A6      B6      C6      D6      E6      F6      G6      H6
A5      B5      C5      D5      E5      F5      G5      H5
A4      B4      C4      D4      E4      F4      G4      H4
A3      B3      C3      D3      E3      F3      G3      H3
A2      B2      C2      D2      E2      F2      G2      H2
A1      B1      C1      D1      E1      F1      G1      H1
Dime la posición donde quieres introducir el caballo
a500
Esa posición no es válida, vuelve a pedir
```

EJERCICIO 10

REALIZA EL JUEGO DE LAS TRES EN RAYA.

Este ejercicio, a pesar de que parece bastante complicado es el que menos me ha costado de los 3 juegos que he realizado en esta práctica.

El planteamiento que he empleado es sencillo, primero necesitaba un tablero y una función que me lo mostrase dependiendo del valor que exista dentro de la posición del tablero.

Necesitaba una función muy grande que me comprobara cuando se acaba una partida y que me diera los ganadores.

Y necesitaba una función que me comparase si había algún empate, que es básicamente cuando no quedan casillas sin rellenar y ninguno de los dos jugadores a conseguido un tres en raya.

Primero para la función que acabe la partida me he decantado por un método de sumatorio de filas columnas y diagonales.

Si el sumatorio de una fila, columna o diagonal es igual a 0 significa que el jugador 1 en este caso ha hecho un 3 en raya ya que como valor por defecto para mostrar un guion bajo en el tablero he puesto un 99

Y si el sumatorio de una fila, columna o diagonal es igual a 3 significa que en esa fila columna o diagonal hay 3 1 y por eso suma 3, por lo cual habría ganado el jugador 2.

El empate sería de forma que ningún sumatorio de filas y columnas da alguno de los valores esperados y encima ya no quedan huecos en blanco.

Luego para los turnos de los jugadores he hecho una variable que controle cuando termine el turno del jugador 1 y si el turno del jugador 1 termina, significa que empieza el turno del jugador 2.

He comparado también que las casillas que decían los jugadores por fila y columna estuvieran vacías para que no pudiesen sobrescribir a la casilla pudiendo dar lugar a una partida infinita.

Si en el turno del jugador 1 la partida acaba por que bien hay un empate o ha ganado, llamaremos a la función acabar partida y haremos un break porque si no espera a que el turno del jugador 2 se ejecute también antes de decir que ha ganado (darme cuenta de esto me ha dado muchos quebraderos de cabeza)

Ya una vez que tenemos controlado el tema de los turnos el flujo del juego se ejecuta normalmente.

Ahora voy a poner algunos ejemplos de partidas con el script

```
Dime un número de columna 1-3
1
O         O         _
_         X         _
X         _         _
Te toca jugador 1:
Dime un número de fila 1-3
1
Dime un número de columna 1-3
3
O         O         O
_         X         _
X         _         _
Ha ganado el jugador 1
```

Aquí gana el jugador 1

```
Te toca jugador 2:
Dime un número de fila 1-3
2
Dime un número de columna 1-3
3
O         O         _
X         X         X
O         _         _
Ha ganado el jugador 2
```

Aquí gana el jugador 2

```
Te toca jugador 1:
Dime un número de fila 1-3
3
Dime un número de columna 1-3
1
O         X         X
O         _         _
O         _         _
Ha ganado el jugador 1
```

Aquí se gana de columna

```
Te toca jugador 1:
Dime un número de fila 1-3
3
Dime un número de columna 1-3
3
O         X         X
_         O         _
_         _         O
Ha ganado el jugador 1

Process finished with exit code 0
```

Aquí el jugador gana de diagonal

```
Te toca jugador 1:
Dime un número de fila 1-3
3
Dime un número de columna 1-3
2
O         X         O
O         X         X
X         O         O
Ha habido empate
```

Y aquí por último ha habido un empate