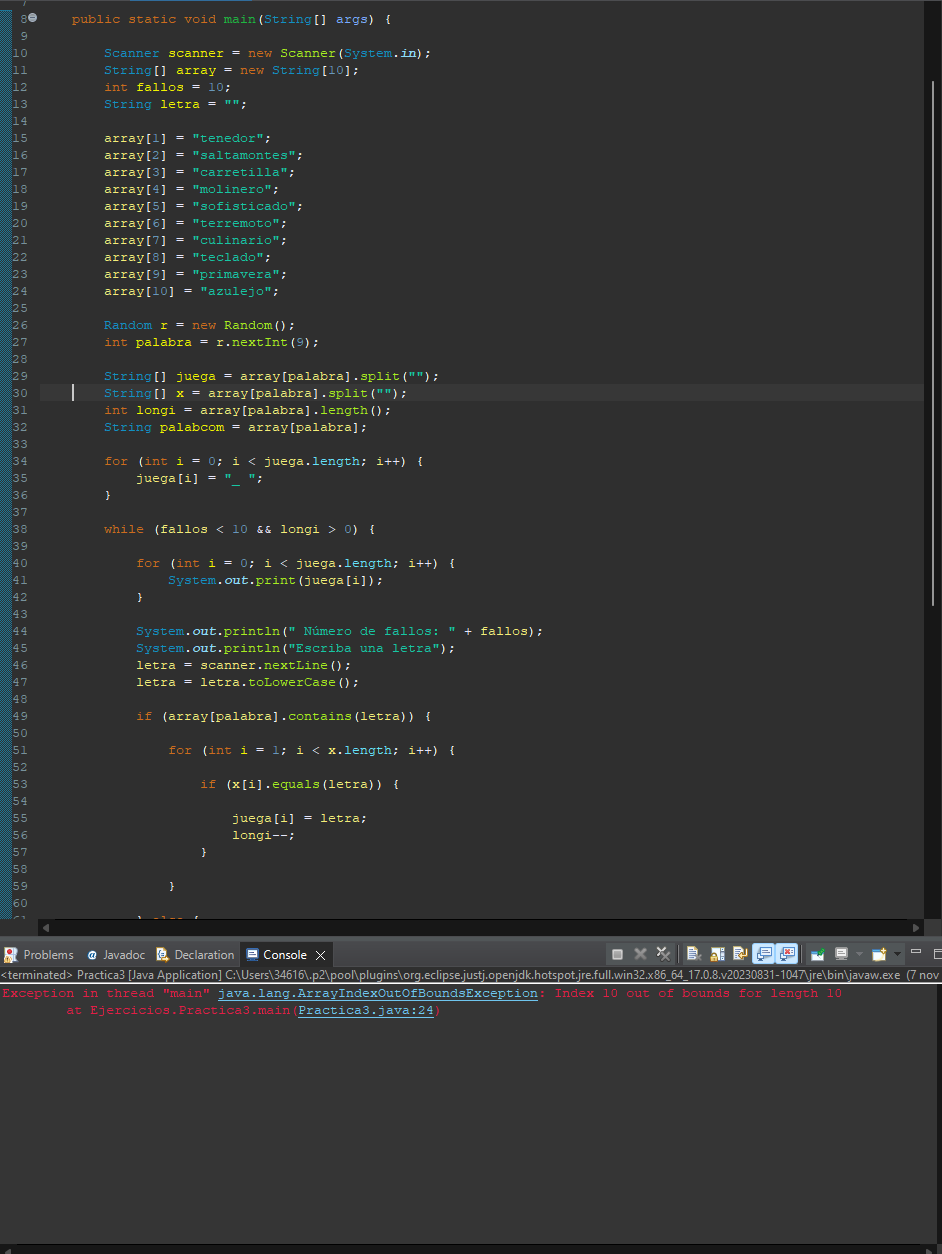
**Práctica 3: Depuración con Eclipse**



**Hecho por: Natalia Martínez García**

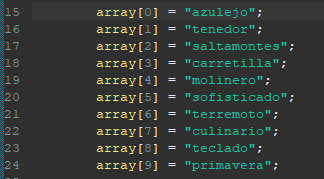
Nos encontramos ante un ejercicio hecho en Eclipse. Se trata del juego del ahorcado, pero cuando lo ejecutamos nos sale un error:



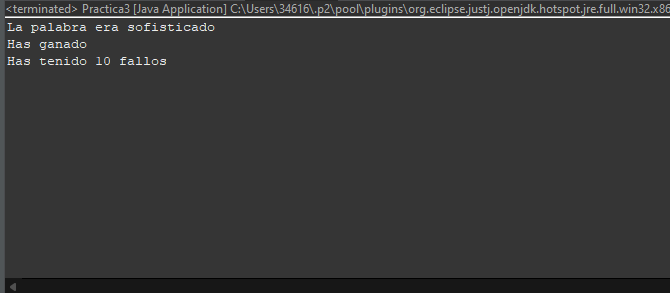
El error nos dice que da el fallo en línea 24, empezamos a depurar en la primera línea de código para comprender mejor este error:

Nos podemos dar cuenta al ir depurando con el F6 que en el “array”, que tiene 10 espacios, no se usa el espacio número 0 y se está intentando poner la palabra “azulejo” en el espacio 10, que no existe:

Por tanto, vamos a cambiar la posición de la palabra “azulejo” a la posición 0:



Una vez arreglado el error, volvemos a ejecutar el programa:



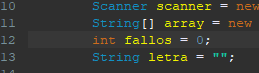
Podemos ver que el juego ha finalizado sin dejarnos jugar, además de la contradicción de que hemos ganado y a la vez hemos tenido 10 fallos.

Volvemos a depurar empezando justo debajo de lo que hemos arreglado:

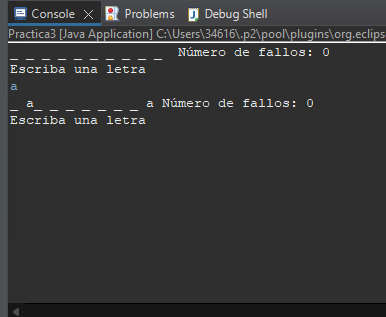
Vemos que cuando tiene que entrar en el “while” para empezar a jugar, hay algo que hace que no entre y se ejecute. Vemos cual es la condición para que entre:

La condición para que el while se ejecute es que el número de fallos sea menor de 10 y que la variable “longi” sea mayor que 0. Veamos lo que valen esas variables.

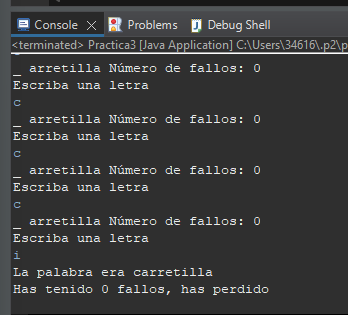
Como podemos apreciar, la variable longi, vale 9, por lo tanto, es mayor que 0. Sin embargo, la variable fallos tiene el valor de 10. Al tener este valor, hace que no se pueda ejecutar el while. Para arreglar este problema vamos a cambiar el valor inicial de la variable fallos a 0.



Si volvemos a ejecutar el programa, vemos que ya podemos jugar:

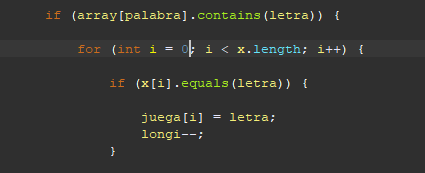


Mientras jugamos nos encontramos un error nuevo. Nos hemos dado cuenta de que la palabra escondida es “carretilla”, hemos puesto todas las letras menos la “c”. Todas las demás letras se han mostrado, pero al poner la letra “c” no nos dice que la letra no está, sin embargo, no la muestra y nos pide otra letra otra vez. Por probar, hemos puesto otra letra al azar y ha terminado el juego, con 0 fallos y encima hemos perdido.

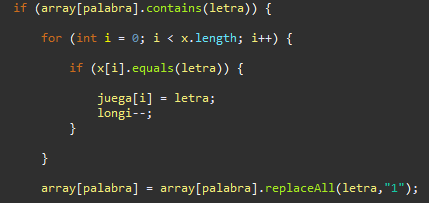


Volvemos a depurar poniendo la interrupción en el while.

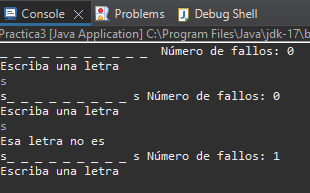
Al depurar el programa nos hemos dado cuenta de que cuando el programa analiza cada una de las letras de la palabra oculta para saber si coincide con la letra que hemos dicho, en vez de empezar por la primera letra, que sería la letra 0, empieza siempre en la segunda letra que sería el 1. La variable que se encarga de saber que letra estamos analizando es la “i” del “for”. Para arreglar esto, solo tenemos que cambiar el contenido de la variable i por un “0”.



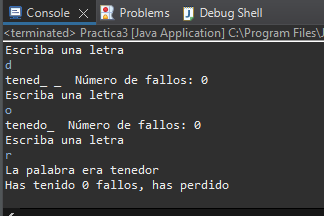
El error de cuando hemos probado una letra al azar se debe a que esa letra la contenía la palabra oculta. En este código nos hemos dado cuenta de que para saber si hemos ganado, hay un contador que ya hemos visto antes “longi” que tiene la cantidad de letras que hay en la palabra oculta, esta variable va disminuyendo cuando acertamos una letra. La condición del “while” que hemos visto antes, es para saber si hemos ganado o perdido. Si los fallos llegan a 10, hemos perdido, y si la variable “longi” llega a 0 significa que hemos acertado la palabra, por lo tanto, hemos ganado. El fallo de este programa es que, si repites una letra que se encuentra en la palabra oculta como antes, el juego se va a ejecutar como si esa letra no se hubiera puesto antes. La variable longi va a disminuir, por lo tanto, si ponemos todo el rato una letra correcta muchas veces, la variable “longi” va a seguir restándose hasta que llegue a 0, por lo tanto, vamos a ganar siempre el juego si hacemos eso, aunque nos falten letras por poner. Para arreglar esto, vamos a cambiar la letra con la cual la hemos comparado para que no se repita. Yo le he puesto el valor de un 1, porque sí, pero no se le debe poner una letra ya que podría interferir con el juego.



Con este arreglo vemos que al poner una letra repetida, la va a contar como un fallo, por lo tanto, ya no va a haber ningún error.



Si volvemos a ejecutar el programa y lo jugamos, vemos que ocurre un último fallo:

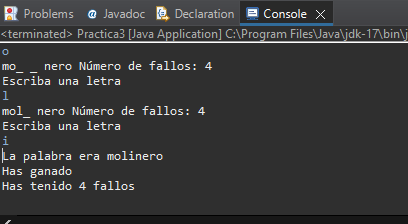


A pesar de que hemos ganado, nos dice que hemos perdido. Hacemos un último debug para saber dónde está el fallo:

Podemos ver que aunque la variable fallos tiene valor de 10, entra en el “if” que imprime por consola que hemos ganado. La condición para entrar en el if es que la variable “longi” no sea igual a 0, sino entra al “else” que imprime por consola que hemos perdido. En este caso vemos que nuestra variable longi vale 9, que inicialmente valía 10. Hemos introducido la letra “t” varias veces. Como la letra “t” pertenece a la palabra oculta, la variable longi ha pasado a valer 9. A continuación hemos introducido la letra “t” hasta llegar a los 10 fallos.

La solución para este error es cambiar la condición del “if” a (longi == 0), ya que si la variable llega a 0 significa que hemos ganado.

Una vez resuelto este problema, podemos ver que el programa ya funciona perfectamente:



En conclusión, al realizar esta práctica, he comprendido mejor como se usa el debug, he podido comprender mejor el código y he podido darme cuenta de un fallo que tenía mi código del que no me había dado cuenta (lo de que si ponías una palabra repetida todo el rato podías ganar todo el rato) y lo he podido arreglar mientras hacía esta práctica.