Proceso de desarrollo UT2 Problema 4. DAVID GARCÍA VALERO

Lo primero que he hecho ha sido el control de validez del input del usuario, para ello me he asegurado que en el caso del número de intentos este número fuese entero y número del 1 al 10. Este ha sido uno de los pasos más difíciles, pues he tenido que, a través de prueba y error (y depurar), verificar que siempre que fuese erróneo, tuviese que introducir otra vez la variable; y que si fuese correcto, no pidiese más de un valor erróneamente.

Este control de datos luego lo pasé a las variables de los intentos, modificándolo, para que fuese incorrecto en caso de no ser un número del 0 al 99 (dos dígitos máximo).

Para implementar la generación de números aleatorios asigné una variable para este número aleatorio (numSecreto) y le di el valor de un número aleatorio con Math.random. Para que tuviese dos dígitos lo multipliqué por 100, y por int para que fuese un número entero.
Teniendo ya el número aleatorio, había que implementar la lógica de las pistas, estuve barajando varias posibilidades, pero la que más me convenció fue separar las condicionales para si el número secreto era mayor a 50 o menor a 50; luego puse otras tres condicionales para las dos posibilidades.
Supuse que si excedían o sobraban 25 unidades era lejano o frío (intento >= // <= numSecretoInt+25)
Que si excedían o sobraban 15 unidades era media distancia o templado (intento >= // <= numSecretoInt+15)
Y que si excedían o sobraban 1 unidades era cercano o caliente (intento >= // <= numSecretoInt+1)

Si bien para implementarlo empecé con un switch, vi que iba a ser una tarea repetitiva e ineficiente, cambié el método y simplemente hice un while para que todos los valores fuesen seguidos y no tener que repetir tanto, así pude depurar más tarde de una manera más efectiva.

En el caso de que los intentos se acaben, el programa se acabará, ya que las siguientes líneas solo funcionan gracias a condicionales que tienen como instancia que el número de inputs sea mayor.
Me costó averiguar cómo hacer para que el programa finalizara en el caso de dar con el número secreto exitosamente.
Tuve que cambiar un poco mi paradigma, al final di con que si ponía una condicional antes de cada siguiente input de intento, funcionando dicha condicional si el número del intento anterior fuese distinto al número secreto. Cerré estos condicionales antes del final del while que englobaba todos los inputs de intentos.

Hice el primer ejercicio opcional, asigné un método para entrar a éste modo sin cambiar la parte obligatoria. Mediante un input eliges la dificultad y el switch valora cada caso, cambiando el número de intentos y el rango de dígitos. Solo había que cambiar el valor en dígitos de Math.random y el número de intentos a emplear.

Una vez había hecho todo esto, había terminado el ejercicio, pero me di cuenta que algo iba muy mal, ya que tenía 2400 líneas de código. Tenía que usar una herramienta para hacer que todas las estructuras que se repetían demasiado hacerlas más pequeñas. Cada vez que hacía un intento, se repetía su número seguido a la derecha.
Esto debería haberlo pensado antes, pero no veía otra forma de hacerlo al principio, y esto ha hecho que se convierta en una bola, tarde, pues, comprendí que lo más adecuado sería un for.
Y para eliminar la estructura repetitiva de las pistas también me serviría.

Básicamente tuve que hacer todo de cero (dejo el archivo del código anterior también por si quieres mirarlo), pero mejor, ya que fue mucho más fácil manejar y visualizar el código de esta manera.
Añadí el for para que el mínimo de intentos fuese 1 y el máximo 10, y cada bucle aumentase en 1 los intentos completados. El formato de los print también dejó de ser repetitivo de ésta forma.
Cambié la forma de adivinar el número y ahora lo hice mediante la diferencia del número input y número secreto, si existe mucha diferencia será frío, si existe media templado y si existe menos caliente.
También cambié el formato para que no resultase tan repetitivo, con un print si era mayor o menor y un

println si es frio, templado o caliente.
Verifiqué el resultado en no más de 10 líneas, cosa que en el anterior código repetía a cada tanto.

Ahora en la parte opcional los intentos serían en fácil 10, en normal 6, y en difícil 4, asignadas mediante un switch con un previo input del usuario para valorar qué dificultad es.

En conclusión, tuve que rehacer el ejercicio ya que el propósito del mismo era la repetición y lo había hecho mal, ahora está mucho mejor con estructuras repetitivas adecuadas. El código pasó de 2300 líneas a 100 líneas la parte obligatoria y 400 líneas en total junto a la parte opcional.

Después de todo el tiempo invertido decidí que sería divertido añadir los puntos por ganar.
Las variables son la dificultad de rango de dígitos escogida (10 fácil, 100 normal, 1000 difícil), la dificultad en cuanto a los intentos escogida (1 (Fácil)*10 2 (Normal)*10 o 3 (Difícil)*10) y el número de intentos restante (+ número de intentos restante *500)

A modo constructivo, el ejercicio podría añadir o tener la funcionalidad para introducir tu nombre o nickname y que tu puntuación se guardase, para que así el siguiente que quisiese hacer el reto, tuviese un aliciente.