## Requisitos del Proyecto:

*"Tiene que haber barreras aleatorias de, entre 1 y 3 casillas, que aparecerán durante 5 segundos y luego desaparecerán. Tiene que haber, al menos, 2 tamaños de tablero y 2 velocidades. Tienen que aparecer objetos aleatoriamente cada x segundos (entre 5 y 20 seg) que, al comerlos sumas 5 puntos. Cada X segundos (entre 20 y 40) la velocidad se aumentará durante 5 segundos."*

Propios:

- Distintos modos de juego para variar la jugabilidad, quería introducir que los muros se pudieran quitar, que la serpiente pasara de un lado a otro del tablero en vez de chocarse, tuberías… Tenía bastantes ideas, pero no he podido implementar todas.
- Hacer una aplicación lo más intuitiva posible, que permitiera al usuario interaccionar con ella para alternar los modos. -> generar un menú principal.
- Que la aplicación usara los principios SOLID (se ha intentado).
- Archivar los datos del récord en un file, para que sobrevivieran a distintas ejecuciones del juego.
- Establecer una pantalla final, para informar al usuario que el juego había finalizado.
- Distintos estilos para personalizar el juego.
- Aplicar la mayor cantidad de conceptos vistos en clase.

## Elaboración del proyecto:

Para comenzar el proyecto, he partido de los requisitos propuestos por el profesor, pero también pensé en distintas opciones que quería implementar. Los primeros días, el trabajo estuvo enfocado en ver distintos juegos de Snake para ver su interacción con el usuario y comprender lo máximo posible la herramienta principal del proyecto: Java.Swing.

La metodología empleada podríamos clasificarla dentro de agile, ya que el objetivo primero era crear algo que funcionara y luego ir construyendo y mejorando alrededor de esto.

Una vez hecho esto, preparé los primeros esquemas a mano, con el desarrollo de clases y como estas iban a trabajar entre sí. Y así construí la primera Serpiente. Primero se movía sola y todas las casillas estaban en una posición. Más tarde conseguí que las casillas se siguieran unas a las otras y que se pudiera mover a voluntad. Para más tarde conseguir que se moviera sola, pero que pudiera elegir yo el movimiento. Esto lo conseguí creando un mando, que tuviera la

dirección y que la serpiente siempre se moviera. El usuario podría modificar esta dirección con las teclas “asdw” gracias a un KeyListener. Con la serpiente hecha, añadí formas de aumentar el tamaño de la serpiente y generar la comida en posiciones donde la serpiente pudiera estar.

En este punto tenía gran parte del proyecto realizado, por eso intente mejorar la estructura general de la aplicación para que fuera más sencilla de interpretar y usarse. Siguiendo (mejor dicho, intentando seguir) los Principios SOLID, que pautan como las aplicaciones deben generarse. No voy a explicarlos, porque lo haría mal, pero recomiendo echarles un vistazo porque son muy interesantes y ayudan mucho. Aquí reestructure las clases con una estructura parecida a la final (ver esquema de clases). Además, empecé a realizar los esquemas en digital en vez de en papel.

La creación de muros supuso bastantes problemas a priori, porque quería que los muros tuvieran distintas direcciones y que su generación fuera totalmente aleatoria. Al final encontré la forma con distintas enumeraciones y clases. La idea desde el principio fue que el juego tuviera la posibilidad de generar más de un muro, pero la realidad es que no he llegado a implementar esta idea. El programa genera un muro aleatorio y cuando pasa el tiempo de vida de este, lo borra y genera otro. El siguiente objetivo fue que detectara la zona ocupada para que no genere muros por encima del cuerpo de la serpiente o fuera del tablero.

Por último, tenía que comunicar a la aplicación, cuando el juego había terminado y ya tenia hecho casi todo el trabajo. A partir de aquí, empecé a desarrollar distintos menús de juego para transformar la aplicación en algo que pudiéramos considerar una aplicación más que un juego sencillo. Y con esto vinieron muchos problemas. Tratamientos de threads, objetos que generan objetos, reiniciar parámetros globales, etc. Pero al final se salvaron los problemas bastante bien.

Para establecer los valores del juego he empleado variables estáticas, se podría hacer de otras maneras, pero al final consideré esta como la mejor opción. Ya que te facilitaba mucho el acceso a ellas. Pensé en un principio que las variables de los objetos fueran privadas, aquí se pueden ver resquicios en la clase Posición y las que heredan de esta, que arrastran métodos getter y setter que se podrían omitir. Pero no esta mal dejarlos, por si en algún momento se quiere retomar la aplicación.

En conclusión, estoy muy contento con el trabajo realizado, porque he ido añadiendo casi todos los conceptos que hemos trabajado en clase y esto me ha permitido trabajarlos y entenderlos mejor. Me hubiera gustado hacer alguna cosa más, pero creo que el trabajo esta representado. Han sido muchas horas de trabajo y sinceramente no se si volveré a jugar nunca a un snake. Pero es mi primera aplicación y en general estoy satisfecho con el resultado.

## Realización de pruebas:

### Pruebas Unitarias:

Al estar el trabajo desarrollado casi en su totalidad en Java.Swing y sobre un entorno gráfico, es bastante difícil generar un plan de pruebas que permita de forma rápida garantizar la seguridad de la aplicación en la totalidad de su ejecución. Por esto, a la hora de ir codificando se han realizado pruebas a pequeña escala introduciendo distintas posibilidades. Reflejo de esto son la utilización de algunos bloques "try…catch" y ajustes a los parámetros de los bucles.

Algunas pruebas para testear métodos concretos han sido por ejemplo,s el alterar el fichero del menú final, desde fuera de la aplicación, introduciéndole parámetros que no deberían estar, como letras a la puntuación y demás.

El trabajo realizado, ha ido acompañado de muchas horas de prueba y error y "pegándose" uno con el código.

### Pruebas de funcionalidad:

Estas han ido enfocadas a probar cada vez que implementaba algo, que esto funcionase y que el resto del programa también lo hacía. Por ello, siempre se ha tenido la máxima de crear las distintas partes de la aplicación por separado, e irlas incluyendo una vez se hayan solucionado los problemas que pudieran tener.

Debido a esto, surgieron muchos problemas a la hora de juntar distintas partes a la aplicación principal, porque no era la misma ejecución. Uno de ellos fue el crear distintos JFrame y que cada uno tuviera sus respectivos escuchadores. El principal se bloqueaba, aunque funcionaran por separado, entre ellos se bloqueaban. Por esto se empleó el uso de distintos threads tipo "Daemon" que paliaran esta situación.

## Conclusión:

El trabajo ha sido muy enriquecedor porque me ha permitido comprender como funciona y se debe desarrollar una aplicación. Aunque el resultado no fuera el óptimo, creo que es bastante bueno y que se observa el trabajo realizado. Me aporta un buen punto de partida a la hora de desarrollar futuras aplicaciones.