**Snakes and Ladders**

Especificación de requerimientos funcionales y diagrama de clases

Danilo Vicente Erazo Meza

Jhan Carlos Carvajal Bastidas

Facultad de Ingeniería, Universidad ICESI

Algoritmos y Programación II

Juan M. Reyes

Abril 2021

# Requerimientos funcionales

El sistema debe estar en la capacidad de:

**Req1. Presentar al usuario la cuadrícula de juego.** La cuadrícula debe tener **n** filas por **m** columnas, dentro de las cuales hay **s** serpientes y **e** escaleras. (cada una de las casillas puede identificarse a través de un número. La numeración inicia en la casilla inferior izquierda con el número 1, sigue en la casilla inmediatamente a la derecha y así hasta terminar la fila. Luego sube en esa misma columna y se regresa hacia la izquierda, intercalando así la dirección en cada fila).

**Req2. Gestionar serpientes.** Las serpientes en el juego unen una casilla con otra cualquiera en una fila inferior. Las serpientes se identifican con letras mayúsculas del alfabeto iniciando en A. En una interfaz por consola, una serpiente puede ser representada a través de la letra que la identifica tanto en la casilla donde inicia como en la casilla donde termina.

**Req3. Gestionar escaleras.** Las escaleras en el juego unen una casilla con otra cualquiera en una fila superior. Las escaleras están numeradas desde 1 hasta el número total de escaleras, siendo cada número el identificador de dicha escalera. Una escalera puede ser representada a través del número que la identifica tanto en la casilla donde inicia como en la casilla donde termina.

**Req4. Presentar y operar menú de opciones.** El programa debe tener un menú con 3 opciones. La primera opción es para jugar, la segunda opción es para ver el tablero de posiciones y la tercera opción es salir del programa.

* **Req4.1. Recibir los datos en una sola línea.** Cuando se escoge la primera opción del menú el programa debe recibir en una sola línea la siguiente información: número de filas, columnas, serpientes, escaleras y un símbolo para cada uno de los jugadores (símbolos disponibles: \* ! O X % $ # + &).
* **Req4.1.1. Asignar aleatoriamente símbolos a cada jugador.** El usuario puede optar por introducir una línea donde en lugar de los símbolos para cada jugador se encuentre el número de jugadores, en este caso el programa deberá asignar aleatoriamente uno de los símbolos disponibles a cada jugador.
* **Req4.2. Crear cuadrícula de juego para iniciar el juego.** Cuando el usuario elige jugar, se crea un juego con una cuadrícula de tamaño **nxm**, con **s** serpientes y **e** escaleras ubicadas aleatoriamente uniendo cualquiera de las casillas del tablero, con las siguientes **restricciones**: ninguna escalera inicia en la casilla 1, ninguna serpiente inicia en la casilla n x m, y ninguna casilla de inicio o fin de escalera o serpiente debe coincidir con otro inicio o fin de escalera o serpiente.
* **Req4.3. Mostrar el tablero de posiciones.** Cuando se escoge la segunda opción el programa mostrará un listado de los nombres o nicknames de los jugadores, sus símbolos y sus respectivos puntajes.
* **Req4.4. Salir del programa.** Cuando se escoge la tercera opción (salir del juego) del menú el programa debe finalizar.

**Req5. Mostrar tablero con las posiciones actuales de los jugadores.** Una vez se despliegue esta visualización de la cuadrícula, el programa queda esperando un salto de línea para iniciar y mostrar el primer tablero. Los tableros del juego irán mostrando la posición de los jugadores en las casillas, pero ya no deben mostrar los números de las casillas, aunque sí las escaleras y las serpientes.

**Req5.1. Mostrar cuadrícula numerada inicial.** Si en lugar de simplemente ingresar un salto de línea al programa, se escribe la palabra **num** y luego se da salto de línea, el programa mostrará la misma cuadrícula que muestra al inicio, con las casillas numeradas, las serpientes y las escaleras. Esperará entonces un salto de línea para mostrar el tablero actual (con las serpientes, escaleras y posición actual de los jugadores) y continuar con el juego.

**Req5.2. Empezar modo simulación.** Si en lugar de simplemente ingresar un salto de línea al programa, se escribe la palabra **simul** y luego se da salto de línea, el programa empezará en modo simulación, que consiste en ir mostrando lo que cada jugador a su turno juega, con el tablero correspondiente de cada nueva posición, esperando 2 segundos entre cada jugada, pero sin esperar ningún salto de línea.

**Req5.3.** **Regresar al menú cuando se recibe la palabra menu.** Si en lugar de simplemente ingresar un salto de línea al programa, se escribe la palabra **menu** y luego se da salto de línea, el juego se corta sin terminar y el programa regresa al menú principal, mostrándose sus opciones.

**Req6.** **Lanzar dado.** Cuando el juego empieza y el jugador en turno ingresa un salto de línea, entonces el jugador lanza el dado, y obtiene un número entre 1 y 6 que es indicado a continuación. Finalmente se realiza el cambio de posición de acuerdo al resultado obtenido

Si el jugador cae en un inicio de serpiente o de escalera, debe bajar o subir, respectivamente, por dicho elemento del juego y quedar en la casilla correspondiente.

**Req7. Mostrar el jugador que ganó el juego.** El juego termina cuando uno de los jugadores llegue a la última casilla, se logra al caer en la última posición u obtener una posición mayor. En ese caso debe decir: El jugador **Z** ha ganado el juego, con **Y** movimientos. Donde **Z** es el símbolo del jugador ganador y **Y** es la cantidad de veces que el jugador lanzó el dado en ese juego. A continuación de este mensaje, se pide el nombre o **nickname** del jugador ganador

**Req8. Regresar al menú principal.** El programa debe regresar al menú principal cuando algún jugador gana el juego (algún jugador llegó a la casilla final).

**Req9.** **Calcular puntaje de los jugadores.** se calcula un puntaje para el usuario que es igual a la cantidad de movimientos multiplicado por la cantidad total de casillas del tablero. Este puntaje debe ser almacenado por el sistema.

# Diagrama de clases

