TRABAJO MULTIMEDIA: VIDEOJUEGO SNAKE.

Participantes:

- **Jesús Manuel Palomo Vega (jmpv0005) DNI: 77379337-T**
- **Celia Macías Torres (cmt00010) DNI: 77367311-A**

Instrucciones de uso y funcionamiento:

Nuestro trabajo se trata del clásico videojuego Snake, que consiste en controlar una serpiente que se desplaza por un mapa intentando comer la mayor cantidad de “frutas” posible sin salirse del mapa ni morderse a sí misma.

El usuario podrá controlar a dicha serpiente mediante las flechas del teclado o, en su defecto, las teclas W, A, S y D y pausar la partida en cualquier momento pulsando la tecla espacio, la cual también nos permitirá reiniciar la partida en caso de haber terminado.

El juego incluirá un HUD o barra de estado que nos informará de la puntuación y longitud de la serpiente, que aumentarán cada vez que coma una fruta, y del tiempo de la partida actual, además de notificaciones para ciertos eventos del juego como la pausa o el fin del mismo. Además, el juego cuenta con clips de sonido tales como una música de fondo, un efecto de sonido para cuando la serpiente logra comer una fruta y otro para cuando la partida termina.

Por último, cada vez que la serpiente coma una fruta, los colores del juego cambiarán.

Clases y sus métodos:

Clase SnakeMulti:

1. **arrancarPartida()** → Método al que llamaremos cada vez que iniciemos una nueva partida. Inicializa todos los valores a su valor por defecto e inicia la reproducción de la pista musical de fondo.
2. **actionPerformed(ActionEvent arg0)** → Éste método será el que controlará el desplazamiento de la serpiente, recibirá una orden de dirección y el método comprobará que el cuadrante hacia dónde se va a dirigir la cabeza de la serpiente sea válido, en caso de no serlo, ya sea por exceder los límites de la pantalla de juego o por intentar acceder a un cuadrante ocupado por una parte del cuerpo de la misma serpiente, el juego terminará automáticamente. También nos permitirá aumentar la puntuación y la longitud de la serpiente si ésta logra comerse una fruta, además de activar un efecto sonoro. En dicho caso, la fruta desaparecerá y aparecerá en otro cuadrante completmente aleatorio de la pantalla.
3. **noEnroscado(int x, int y)** → Éste será un método auxiliar al que llamaremos en el anterior método. En el, comprobaremos el cuadrante hacia donde quiere desplazarse la cabeza de la serpiente y

comprobaremos si coincide con alguna de las casillas ocupadas por la misma, las cuales se guardan en una lista. En caso de coincidencia, el método devolverá false, en caso de no coincidir, devolverá true.

4. **keyPressed(KeyEvent p)** → Éste será el método que determine el movimiento que queremos que la serpiente realice en función de la tecla que pulsemos, admitirá tanto la pulsación de las flechas como de las letras W, A, S y D. Asimismo, no permitirá realizar el movimiento antagonista al actual, por ejemplo, si vamos hacia la derecha, no nos dejará ir hacia la izquierda, ya que la serpiente volvería sobre sí misma y perderíamos el juego automáticamente. También nos permitirá pausar el juego pulsando la tecla espacio o reiniciarlo en caso de haber perdido.
5. **sonido(String archivo)** → Con éste método, podremos acceder a las pistas de audio del paquete de sonidos de nuestro proyecto, lo llamaremos en los anteriores métodos pasándole por cabecera un String con el nombre del archivo a reproducir, en el método nosotros llamaremos a ese archivo añadiéndole la ruta *Sonidos* en este caso dado que así hemos llamado al paquete, seguido del mismo nombre del archivo, seguido de la extensión .wav que es la extensión de nuestros clips de sonido.

Bibliotecas utilizadas:

- **Java awt** → Para el apartado gráfico.
- **Java util** → Para trabajar con aleatorios y arraylist
- **Java applet** → Para poder acceder a los clips de sonido y reproducirlos.

Clase Panel:

1. **paintComponent(Graphics p)** → El único método de esta clase, nos permitirá elegir los colores de los elementos de nuestro juego, así como añadir una interfaz y notificaciones en caso de la pausa o del fin del juego.

Bibliotecas utilizadas:

- **Java awt y Java swing** → Apartado gráfico y ventana.