UNIVERSIDAD SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE

INTRODUCCIÓN A LA PROGRAMACIÓN Y LA COMPUTACIÓN
SECCIÓN 1 A

CATEDRÁTICO:
ING. JOSE MOISES GRANADOS GUEVARA

ESTUDIANTE:
JORGE ANIBAL BRAVO RODRÍGUEZ
CARNÉ: 202131782

PRÁCTICA 1
ESCAPE DEL LABERINTO
MANUAL TÉCNICO

1. INFORMACIÓN GENERAL DEL DESARROLLO

El desarrollo del juego Escape del Laberinto originalmente inició en un entorno Linux, específicamente KDE NEÓN edición 25 aniversario basado en Ubuntu 20.04. Sin embargo, por problemas de la propia distribución tras una actualización se corrompió el gestor de inicio de sesiones. Esto llevó a la reinstalación de la distribución y a utilizar originalmente el IDE IntelliJ IDEA Community 2021.3.2 por aparentes problemas de dependencias.
Debido a problemas posteriores el desarrollo se migró a Windows 11 y al IDE NetBeans, aunque debido al propio lenguaje de programación utilizado, Java permite la ejecución del programa en múltiples sistemas operativos.

Se da garantía de que las funciones utilizadas en el programa, como limpiar pantalla, funcionan correctamente en Windows y en Linux con Bash como intérprete de línea de comandos. Me es imposible realizar estas pruebas en otros sistemas operativos alternativos como Mac OS, BSD, Solaris, etc.

1.1 NetBeans:

NetBeans es un IDE desarrollado y mantenido por Apache, siendo compatible de entrada con el lenguaje de programación JAVA. Este necesita que exista un SDK instalado previamente. Este IDE es multiplataforma, y en este proyecto fue usado tanto en Linux como en Windows.

Como gestor-constructor de proyectos se utilizó Maven.

1.2 KDE Neon y Windows 11

KDE Neon es una distribución de Linux basada en Ubuntu 20.04 (en la fecha de publicación de este proyecto) cuya principal diferencia de este es que incluye la última versión del entorno de escritorio Plasma en lugar de GNOME. KDE Neon hereda los repositorios propios de Ubuntu mantenidos por Canonical, la compañía que mantiene el proyecto base.

Windows 11 por su parte es un sistema operativo independiente desarrollado por Microsoft, incompatible con sistemas operativos de tipo UNIX.

Tanto en KDE Neon como en Windows 11 se hizo uso de los JDK 15 (openJDK en el caso de Linux, Oracle JDK en el caso de Windows).

1.3 Visual Studio Code

En ciertas ocasiones fue más cómodo utilizar un editor con resaltado de texto para JAVA (utilizando el plugin de soporte de escritura de código JAVA por Red Hat) para lograr una escritura de código ágil. Esto se debe a que el editor de texto es más ligero que un IDE completo, aunque solamente permite la escritura de código.

VS Code fue utilizado tanto en Windows como en Linux.

1.4 REQUISITOS
Al ser un programa desarrollado en Java, una herramienta multiplataforma, se extiende su compatibilidad a la compatibilidad de Java en cuanto a sistema operativo y a requisitos mínimos para ejecutar Java.
El juego no es demandante en cuanto a recursos, por lo que los requisitos mínimos son, en realidad, los requisitos mínimos de un dispositivo que sea capaz de ejecutar Java, sea arquitectura ARM, x86, RISC, etc.

Sucede lo mismo con el sistema operativo, se requiere de un sistema operativo que sea soportado por Java para ejecutar JVM en modo consola. Puede funcionar con Windows, Mac OS, distribuciones GNU/Linux, distribuciones basadas en BSD, Solaris, entre otros.

Se recomienda usar Java 15 o superior, puesto que es en esta versión del Java Development Kit que se realizó el desarrollo.

2. FUNCIONAMIENTO DE LOS MAPAS

2.1 Tipos de Mapas

En el código se especifican dos tipos de mapas, siendo estos:

- Mapas Inmutables: estos son los mapas originales sobre los que se trabajarán. Estos mapas son los mapas definidos como mapa principal, mapa ranura A y B (respectivamente). En el caso de los mapas en ranura A y B ya no podrán ser modificados luego de ser ingresados y cumplir con los requisitos especificados al momento de ser definidos por el usuario.
- Mapa En Uso: Son los mapas que contienen la información de los mapas originales para ser utilizados durante el juego. Estos mapas copian mediante una función la información albergada en los mapas originales, dejando estos intactos.

2.2 Funcionamiento de los Mapas

Los mapas son arrays de Strings de tres dimensiones. Las dimensiones están definidas de la siguiente manera:
MapaEnUso [coordenada en X] [coordenada en Y] [Nivel del mapa]

Las primeras dos dimensiones definen explícitamente una coordenada en el plano establecido. En el nivel cero del mapa se encuentran todos los elementos dispuestos de manera literal como monedas, jugador, bot, paredes, espacios vacíos y salidas.

En el segundo nivel no se guardan los elementos de los elementos del primer nivel, sino sus propiedades. Las propiedades son las siguientes:

0. Casilla normal
1. Casilla de moneda
> 3. Casilla de Salida

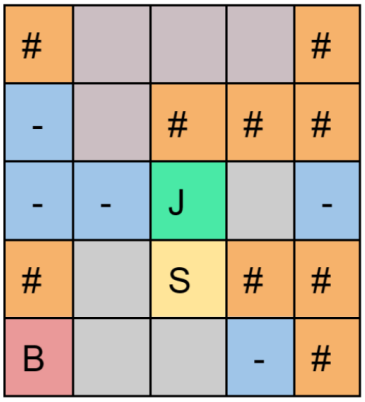
Estas propiedades son usadas por el método revelar monedas. Las monedas son transparentes para el bot y el jugador, pero para que estos no las destruyan, el método mencionado anteriormente comprueba si en una casilla de oro (de propiedad 1) está ocupada en su nivel inferior por el jugador o el bot. En caso de ser cierto el método devuelve la misma entrada, es decir, el bot o el jugador que ya ocupaban esa casilla.

Cuando estos se desplacen a otra casilla el método revelar Monedas buscará nuevamente en el nivel inferior de las propiedades si una coordenada de propiedad 1 está siendo ocupada. Si no es ocupada (es decir, si existe un espacio vacío producto del desplazamiento de uno de los dos jugadores), devuelve el valor del oro en esa coordenada específica.

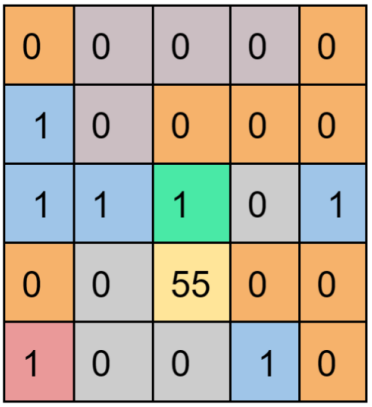
En caso de que el jugador tome el oro disponible, la propiedad 1 de esa coordenada se destruye, reemplazandola por un valor cero. Si el jugador ingresa un comando inexistente entonces se genera aleatoriamente una coordenada de espacio vacío que se reemplaza por una propiedad de oro (propiedad 1) para que esta sea tratada como tal.

Explicación gráfica:

Nivel 0:



Nivel 1:



3. ALGORITMOS

El juego fue desarrollado utilizando métodos y funciones, las cuales son llamados en los puntos donde son requeridos, promoviendo la recursividad y modularidad del código. La única hoja de código pertenece al paquete com.jbravo.esdapedellaberinto. Como clase principal se utilizó escapeDelLaberinto, quedando la referencia de la clase main como com.jbravo.escapedellaberinto.escapeDelLaberinto.
Por comodidad de la calificación existe un archivo llamado Algoritmos.txt que contiene los algoritmos en un archivo de texto plano.

```
//Jorge Anibal Bravo Rodríguez  Carné: 202131782
paquete com.jbravo.escapedellaberinto;

public class escapeDelLaber{

        var mapaPrincipal[][][]; // [x][y][cantidad] cantidad -> usada como atributo de
oro y salida
        var mapaRanuraA   [][][]; //por defecto los mapas creados s por usuarios son de
tamaño fijo 15 x 15
      var mapaRanuraB  [][][];
      var      nombreRanuraA;                          //variables para mapas
      var      nombreRanuraB;
      var monedasRequeridasA = 0;
      var monedasRequeridasB = 0;
      var contRanura         = 0;
      var contTurnoJugador   = 0;
      var conTurnoBot        = 0;


        const var pared     = "p"; //nombres dSi erentes a las variables globales|| uso:
impr instrucciones y validación de la entrada
       const var moneda    = "g";
       const var vacio     = "e";
       const var salida    = "s";

         var [1][3] contMapaPrincipal; //0,0 veces que se juega el mapa; 0, 1 veces
ganadas; 0,2 veces perdidas
      var [1][3] contMapaA        ;
      var [1][3] contMapaB        ;

      var mapaEnUso[30][30][2]; //variables de juego

       var            confGrafica = 1;//1 sin vista por turnos, 2 vista parcial, 3 vista
total
      var              contOro = 0; //cuenta la cantidad de oro recolectahacer
      var         cantPlayerWin = 0;
      var            cantBotWin = 0;
      var     cantBotViewPlayer = 0;
      var           contPartida = 0;
      var          contOroTotal = 0;
```

```
      var          contMovTotal = 0;
       var         findPlayer = false; //indica si el jugador se encuentra en rango 5x5
desde el bot
      var                dirMovBot; //0 arriba; 1 derecha; 2 abajo; 3 izquierda
      var          finjuego = false;

   // CONTROLES (declaradas como constantes por comodidad de testeo)
       var arriba     = "w";
       var abajo      = "s";
       var izquierda  = "a";
       var derecha    = "d";
       var coger      = "c";
       var mirMapa    = "o";
       var salir      = "e";
       var oroDisp    = "i";

   //----------- variables usadas para construir los mapas ----------
      var wall =  " M "; //en el hacer cumento se especSi ica -> #
      var esVa =  "   "; //en el hacer cumento se especSi ica -> O
      var coin =  " - "; //en el hacer cumento se especSi ica -> G
      var play =  " J "; //en el hacer cumento se especSi ica -> J
      var bot  =  " B "; //en el hacer cumento se especSi ica -> B
      var exit =  " S "; //en el hacer cumento se especSi ica -> S

      var cooPlay[2]; //coordenadas jugador, 0 es X; 1 es Y
      var cooBot [2]; //coordenadas bot      0 es X; 1 es Y
      var monedasRequeridas = 0;
       var idMapa        = 0; //mapa con el que se trabaja; 0 -> Mapa Default; 1 -> mapa
A; 2 -> mapa B
        var limitador      = 0; //limita a 15 o 30 las casillas disponibles según se
requiera
      var esperaBot =  2000;

   public   void main ( var[] args){

       rellenarMapaPrincipal();
       inicializarMapas();
       inicializarContadores;
        var opMenuPrincipal;
       hacer {
           Escribir(" - BIENVENIDO  AL JUEGO ESCAPE DEL LABERITNO -");
           Escribir("Por favor ingrese la opción que desee realizar");
           Escribir("1 - - - - - - - - - - - - - - - - - - -Jugar");
           Escribir("2 - - - - - - - - - - - - - - - - - - Crear mapa");
           Escribir("3 - - - - - - - - - - - - - - - - - - Reportes");
           Escribir("4 - - - - - - - - - - - -Previsualizar mapas");
           Escribir("5 - - - - - - - - - -Configuraciones generales");
           Escribir("6 - - - - - - - - - - - - - - - - - - -Salir");
           opMenuPrincipal = leer();

           Evaluar (opMenuPrincipal){
               caso 1:
                   limpiarPantalla();
                    var opcion;
                   Escribir("Por favor, ingrese el mapa que desea jugar");
                   evaluar (contRanura){
                       caso 0:
                           Escribir(" 1 - Mapa por defecto");
                           opcion = leer();

                           evaluar  (opcion){
                               caso 1:
                                   idMapa = 0;
                                   llamadasJugar();
                                   alto;
                               por defecto:
                                   Escribir("El dato ingresado  no es válido ");
                                   alto;
                           }
                           alto;
                       caso 1:
                           Escribir("1 - Mapa por defecto");
```

```
                    Escribir("2 - " + nombreRanuraA);
                    opcion = leer();

                    evaluar  (opcion){
                        caso 1:
                            idMapa = 0;
                            llamadasJugar();
                            alto;
                        caso 2:
                            idMapa = 1;
                            llamadasJugar();
                            alto;
                        por defecto:
                            Escribir("El dato ingresado  no es válido ");
                            alto;
                    }
                    alto;
                caso 2:
                    Escribir("1 - Mapa por defecto");
                    Escribir("2 - " + nombreRanuraA);
                    Escribir("3 - " + nombreRanuraB);
                    opcion = leer();
                    evaluar  (opcion){
                        caso 1:
                            idMapa = 0;
                            llamadasJugar();
                            alto;
                        caso 2:
                            idMapa = 1;
                            llamadasJugar();
                            alto;
                        caso 3:
                            idMapa = 2;
                            llamadasJugar();
                            alto;
                        por defecto:
                            Escribir("El dato ingresado  no es válido ");
                            alto;
                    }
                    alto;
            }
            alto;
        caso 2:
            Si (contRanura < 3){
                crearMapa();
            }Sino {
                Escribir("Solamente puede crear hacer s mapas, opción no válida");
            }
            alto;
        caso 3:
            reporteGeneral();
            alto;
        caso 4:
            limpiarPantalla();
            Escribir("Estos son los mapas que puede consultar");
            evaluar (contRanura){
                caso 0:
                    Escribir(" 1 - Mapa por defecto");
                    opcion = leer();

                    evaluar  (opcion){
                        caso 1:
                            idMapa = 0;
                            dibMapa();
                            alto;
                        por defecto:
                            Escribir("El dato ingresado  no es válido ");
                            alto;
                    }
                    alto;
                caso 1:
                    Escribir("1 - Mapa por defecto");
```

```
                        Escribir("2 - " + nombreRanuraA);
                        opcion = leer();

                        evaluar  (opcion){
                            caso 1:
                                idMapa = 0;
                                dibMapa();
                                alto;
                            caso 2:
                                idMapa = 1;
                                dibMapa();
                                alto;
                            por defecto:
                                Escribir("El dato ingresado  no es válido ");
                                alto;
                        }
                        alto;
                    caso 2:
                        Escribir("1 - Mapa por defecto");
                        Escribir("2 - " + nombreRanuraA);
                        Escribir("3 - " + nombreRanuraB);
                        opcion = leer();

                        evaluar  (opcion){
                            caso 1:
                                idMapa = 0;
                                dibMapa();
                                alto;
                            caso 2:
                                idMapa = 1;
                                dibMapa();
                                alto;
                            caso 3:
                                idMapa = 2;
                                dibMapa();
                                alto;
                            por defecto:
                                Escribir("El dato ingresado  no es válido ");
                                alto;
                        }
                        alto;
                }
                alto;
            caso 5:
                 var opConf = 0;
                Escribir("------------ ¿Qué desea configurar? -----------");
                Escribir("1. - - - - - - - - - - Vista de mapa por turnos");
                Escribir("2. - - - - - - Cambiar tiempo de espera del bot");
                Escribir("3. - -Conservar la configuración actual y salir");
                opConf = leer();

                evaluar (opConf){
                    caso 1:
                        confMapaView();
                        alto;
                    caso 2:
                        confTiempoBot();
                        alto;
                    por defecto:
                        Escribir("Se mantendrán las configuraciones actuales");
                        alto;
                }
                alto;
            caso 6:
                Escribir("Gracias por jugar Escape del Laberinto   ");
                alto;
            por defecto:
                   Escribir("El dato ingresado  no se encuentra dentro de las opciones
disponibles");
                Escribir("Por favor  varentelo de nuevo ");
                alto;
        }
```

```
        }mientras(opMenuPrincipal != 6);
    }

    public   void limpiarPantalla() {
            var OS = System.obtener("os.name");
            Si (OS.contiene("Windows")) { //Si es winhacer ws ejecutar cls
                escribir.por.consola("cls")
            } Sino  {
                escribir.por.consola("clear"); //Si no es winhacer ws ejecutar clear
            }
    } //Método  para limpiar pantalla

    public   void inicializarMapas(){
        for ( var y = 0; y < 30; y++){
            for ( var x = 0; x < 30; x++){
                mapaPrincipal[x][y][1] = "0";
                mapaRanuraA[x][y][1] = "0";//3 identSi icahacer r de casilla normal
                mapaRanuraB[x][y][1] = "0";
            }
        }
         mapaPrincipal[ 0][ 1][ 1] = "15";  //oro requerido en salida especificada en el
documento
        mapaPrincipal[ 8][ 0][ 1] = "30";
        mapaPrincipal[13][29][ 1] = "43";
        mapaPrincipal[29][ 1][ 1] = "51";
        mapaPrincipal[29][21][ 1] = "55";
    }//para que no inicien en null
    public   void rellenarMapaPrincipal(){
         /*este proceso asigna el valor de paredes, monedas, espacios y salidas al mapa
especificado en el documento
             Solamente se dejan líneas de ejemplo, en el código original son 900
declaraciones
           En la dimensión [x][y][0] se almacena el caracter que representa (espacio,
moneda, pared) mientras que
          en la dimensión [x][y][1] se almacena propiedades de la casilla, como si es oro
o si es una salida indicando
        la cantidad necesaria de esta*/
        mapaPrincipal[ 0][ 0][ 0] = wall;
        mapaPrincipal[ 0][ 1][ 1] = "15";
        mapaPrincipal[ 8][ 0][ 1] = "30";
        mapaPrincipal[13][29][ 1] = "43";
        mapaPrincipal[29][ 1][ 1] = "51";
        mapaPrincipal[29][21][ 1] = "55";

    } //llena el mapa por defecto con el mapa especificado en el documento

    public   void dibMapa() { //Función para dibujar el mapa en la terminal
        evaluar  (idMapa){
                for ( var y = 0; y < 30; y++) {
                    for ( var x = 0; x < 30; x++) {
                        evaluar(idMapa){
                            caso 0:
                            Escribir (mapaPrincipal[x][y][0]); // i = x; j = y
                            alto;
                            caso 1:
                            Escribir (mapaRanuraA[x][y][0]); // i = x; j = y
                            alto;
                            caso 2:
                            Escribir (mapaRanuraB[x][y][0]); // i = x; j = y
                            alto;
                        }
                        Escribir(" ");
                    }

                }
    } //dibuja los mapas originales en pantalla, sin elementos agregahacer s
    public   void inicializarContadores{
        for( var i = 0; i < 3; i++){
            contMapaA           [0][i] = 0;
            contMapaB           [0][i] = 0;
            contMapaPrincipal   [0][i] = 0;
        }
```

```
    }

    //Funciones para crear mapas
    public   void crearMapa(){
        limpiarPantalla();
        Escribir("BIENVENIDO  a la herramienta de creación de mapas");
             Escribir("A continuación deberá ingresar los parámetros conforme se vayan
pidienhacer ");
         Escribir("Al finalizar se le notSi icará de posibles inconvenientes para realizar
cambios");
           Escribir("Si ingresa un carácter inválido  ese espacio se reemplazará con un
elemento aleatorio");
         Escribir("Este elemento puede ser una moneda, un espacio vacío, una salida o una
pared");
        Escribir("Tome en cuenta que los mapas son de 15 x 15 casillas");
        Escribir(" ");
        contRanura = contRanura + 1;
        evaluar (contRanura){
            caso 1://escribe en ranura A
                Escribir("¿Qué nombre tendrá su mapa?");
                nombreRanuraA = leer();
                hacer {
                        Escribir("Por favor ingrese la cantidad de monedas requeridas para
ganar en este mapa");
                    Escribir("Tome en cuenta que deberá ser un número positivo");
                    monedasRequeridasA = leer();
                }mientras(monedasRequeridasA < 1);
                escribirMapaA();
                alto;
            caso 2:
                Escribir("¿Qué nombre tendrá su mapa?");
                nombreRanuraB =
                hacer {
                        Escribir("Por favor ingrese la cantidad de monedas requeridas para
ganar en este mapa");
                    Escribir("Tome en cuenta que deberá ser un número positivo");
                    monedasRequeridasB = leer();
                }mientras(monedasRequeridasB < 1);
                escribirMapaB();
                alto;
            por defecto:
                Escribir("No es posible ingresar un nuevo mapa");
                alto;
        }
    } //menú para la creacion de mapas y asignacion de variables emparentadas
    public   void escribirMapaA(){
        idMapa = 1;
          varentrada; //variable transitoria para validar la entrada, asigna valor a la
casilla indicada
          var aleatorio; //usada para asignar un valor aleatorio en caso  de no reconocer
una entrada válida
         var contMonedas = 0; //cuenta las monedas dispuestas en el mapa por el usuario
         var contSalidas = 0; //cuenta la cantidad de salidas dispuestas por el usuario

        for( var y = 0; y < 30; y ++){
            limpiarPantalla();
            Escribir("Deberá ingresar las casillas una por una con los siguientes elementos
sin agregar espacios");
             Escribir("Si ingresa una salida deberá especificar la cantidad de oro que esta
requiere");
                Escribir("Elementos -> Pared: " + pared + " || Moneda: " + moneda + " ||
Espacio: " + vacio + " || Salida: " + salida);
            Escribir("Está ingresando  la fila No. " + y);
            Escribir(" ");
            for ( var x = 0; x < 30; x++){
                entrada =
                evaluar (entrada){
                    caso  pared:
                        mapaRanuraA[x][y][0] = wall;
                        alto;
                    caso moneda:
                        mapaRanuraA[x][y][0] = coin;
```

```
                        contMonedas = contMonedas + 1;
                        alto;
                    caso vacio:
                        mapaRanuraA[x][y][0] = esVa;
                        alto;
                    caso salida:
                        mapaRanuraA[x][y][0] = exit;
                        Escribir("¿Cuánto oro se requiere en esta salida?");
                        mapaRanuraA[x][y][1] =
                        contSalidas = contSalidas +1;
                        Escribir("ahora continua con las casillas");
                        alto;
                    por defecto:
                        aleatorio = NumAleatorio(4);
                        evaluar  (aleatorio){
                            caso 0:
                                mapaRanuraA[x][y][0] = wall;
                                alto;
                            caso 1:
                                mapaRanuraA[x][y][0] = coin;
                                contMonedas = contMonedas + 1;
                                alto;
                            caso 2:
                                mapaRanuraA[x][y][0] = esVa;
                                alto;
                            caso 3:
                                mapaRanuraA[x][y][0] = exit;
                                mapaRanuraA[x][y][1] = NumAleatorio(15)-5;
                                contSalidas = contSalidas +1;
                                alto;
                        }
                }
            }
        }
        Si (contMonedas < monedasRequeridas || contSalidas < 1){
            limpiarPantalla();
             Escribir("No ingresó suficientes monedas o salidas, puede corregirlo usanhacer
el editor");
            editorMapa(contMonedas, contSalidas);
        }
    }
    public   void escribirMapaB(){
         Similar a la función escribirMapaA
    }
    public   void editorMapa( var contMonedas,  var contSalidas){
         var x;
         var y;
        Si (contMonedas < monedasRequeridas){
                 Escribir("Las monedas requeridas son: " + monedasRequeridas + " y las
ingresadas son: " + contMonedas);
            Escribir("El mapa ingresado  es el siguiente");
            Escribir(" ");
            Escribir(" 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14");
            dibMapa();

            Escribir(" ");
              Escribir("Ingrese el número de culumna hacer nde se encuentra el elemento a
cambiar por una moneda ");
            Escribir("Tome en cuenta que columnas y filas inician en cero");
            x = leer();

               Escribir("Ingrese el número de fila hacer nde se encuentra el elemento a
cambiar por una moneda ");
            y = leer();

            evaluar  (idMapa){
                caso 1:
                    mapaRanuraA[x][y][0] = coin;
                    contMonedas = contMonedas + 1;
                    alto;
                caso 2:
                    mapaRanuraB[x][y][0] = coin;
```

```
                    contMonedas = contMonedas + 1;
                    alto;
                }
                limpiarPantalla();
            }

            Si (contSalidas < 1){
                Escribir(" -------------      PRESTE ATENCIÓN       -------------");
                Escribir("Se necesita al menos una salida en cada mapa para ganar");
                Escribir("El mapa ingresado  es el siguiente");
                Escribir(" ");
                Escribir(" 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14");
                dibMapa();
                Escribir(" ");
                  Escribir("Ingrese el número de culumna hacer nde se encuentra el elemento a
cambiar por una salida");
                Escribir("Tome en cuenta que columnas y filas inician en cero");
                x = leer();
                   Escribir("Ingrese el número de fila hacer nde se encuentra el elemento a
cambiar por una salida ");
                y = leer();
                evaluar  (idMapa){
                    caso 1:
                        mapaRanuraA[x][y][0] = exit;
                        Escribir("¿Cuánto oro se requiere en esta salida?");
                        mapaRanuraA[x][y][1] =  leer();
                        contSalidas = contSalidas + 1;

                        alto;
                    caso 2:
                        mapaRanuraB[x][y][0] = exit;
                        Escribir("¿Cuánto oro se requiere en esta salida?");
                        mapaRanuraB[x][y][1] =  leer();
                        contSalidas = contSalidas + 1;
                        alto;
                }
                limpiarPantalla();
            }
            Si (contMonedas < monedasRequeridas || contSalidas < 1){
                editorMapa(contMonedas, contSalidas);//función ciclica hasta que contMonedas
= monedasRequeridas
            }
        }

        //Funciones para jugar
        public   void llamadasJugar(){

            Si (idMapa == 0){
                contMapaPrincipal[0][0] = contMapaPrincipal[0][0] + 1;
            }
            Si (idMapa == 1){
                contMapaA[0][0] = contMapaA[0][0] + 1;
            }
            Si (idMapa == 2){
                contMapaB[0][0] = contMapaB[0][0] + 1;
            }

            contOro                         = 0;
            contTurnoJugador                = 0;
            conTurnoBot                     = 0;
            cantBotViewPlayer               = 0;
            contPartida =   contPartida     + 1;
            finjuego                    = false;
             var everyElements    = true;
            copiarMapaJugar();
            generarElementosRandom m(everyElements);
            revelarMonedas();
                mientras (!finjuego){
                    accionesJugador();
                    contMovTotal = contMovTotal + 1;
                    contTurnoJugador  = contTurnoJugador  + 1;
                    revelarMonedasEnJuego();
```

```
                Escribir("El bot está ejecutanhacer  su turno");
                Esperar(esperaBot);
                Si ((conTurnoBot % 2) == 0){
                    mirarMapaBot();
                }Sino {
                    moverBot();
                    revelarMonedasEnJuego();
                }
                Escribir("El bot ha completahacer  su turno");
                limpiarPantalla();
                mostrarMapa();
                compBotGana();
                conTurnoBot = conTurnoBot + 1;
            }
        }
        contOroTotal = contOroTotal + contOro;
        reporteFinPartida();
    }

    public   void accionesJugador(){

         var accion; //guarda la accion realizada en el turno
         var contErr = 0;         // cuenta la cantidad de comandos mal ingresados
          var cantComandToEnd = 3; // indica la cantidad de comandos mal ingresados para
finalizar el juego
         var finTurno = false;

        hacer  {
             Escribir("Desplazamiento: Arriba: " + arriba + " || Abajo: " + abajo + " ||
Derecha: " + derecha + "|| Izquierda: " + izquierda);
            Escribir(" vareracción: Mirar mapa : " + mirMapa + " || Coger oro: " + coger +
" || Oro disponible : "+ oroDisp + " || Salir : " + salir);
            accion =  leer();

                limitador  = 29;
                finjuego = false;
                evaluar (accion){
                    caso arriba:
                        Si ((cooPlay[1] -1) <0 ){ //limitar al borde del mapa
                            Escribir("Jugada imposible");
                        } Sino  {
                                Si ( mapaEnUso[cooPlay[0]][cooPlay[1]-1][0].equals(esVa) ||
mapaEnUso[cooPlay[0]][cooPlay[1]-1][0].equals(coin)){
                                mapaEnUso[cooPlay[0]][cooPlay[1]-1][0] = play;
                                mapaEnUso[cooPlay[0]][cooPlay[1]][0] = esVa;
                                cooPlay[1] = cooPlay[1]-1; //solo se mueve en y
                            }Sino {
                                Si (mapaEnUso[cooPlay[0]][cooPlay[1]-1][0].equals(exit)){
                                                                        Si  (contOro  >=
(mapaEnUso[cooPlay[0]][cooPlay[1]-1][1])){
                                        Escribir("¡Lograste completar el Laberinto   !");
                                        Escribir("¡Felicidades!");
                                        cantPlayerWin = cantPlayerWin + 1;
                                        contMapaPrincipal[0][1] = contMapaPrincipal[0][1] +
1;
                                        finjuego = true;
                                    }Sino {
                                        Escribir("Jugada imposible");
                                                Escribir("Esta salida necesita " +
mapaEnUso[cooPlay[0]][cooPlay[1]-1][1] + " de oro");
                                     }}Sino {Escribir("Jugada imposible, la casilla ya está
ocupada");}}}
                        finTurno = true;
                        alto;
                    caso abajo:
                        Si ((cooPlay[1] + 1) > limitador  ){ //limitar al borde del mapa
                            Escribir("Jugada imposible");
                            contErr = contErr +1;
                        } Sino  {
                                Si ( mapaEnUso[cooPlay[0]][cooPlay[1]+1][0].equals(esVa) ||
mapaEnUso[cooPlay[0]][cooPlay[1]+1][0].equals(coin)){
                                mapaEnUso[cooPlay[0]][cooPlay[1]+1][0] = play;
```

```
                                mapaEnUso[cooPlay[0]][cooPlay[1]][0] = esVa;
                                cooPlay[1] = cooPlay[1]+1; //solo se mueve en y
                            }Sino {
                                Si (mapaEnUso[cooPlay[0]][cooPlay[1]+1][0].equals(exit)){
                                                                                    Si  (contOro  >=
(mapaEnUso[cooPlay[0]][cooPlay[1]+1][1])){
                                        Escribir("¡Lograste completar el Laberinto   !");
                                        Escribir("¡Felicidades!");
                                        cantPlayerWin = cantPlayerWin + 1;
                                        contMapaPrincipal[0][1] = contMapaPrincipal[0][1] +
1;
                                        finjuego = true;
                                    }Sino {
                                        Escribir("Jugada imposible");
                                                    Escribir("Esta salida necesita " +
mapaEnUso[cooPlay[0]][cooPlay[1]+1][1] + " de oro");
                                     }}Sino {Escribir("Jugada imposible, la casilla ya está
ocupada");}}}
                        finTurno = true;
                        alto;
                    caso izquierda:
                        Si ((cooPlay[0] - 1) < 0 ){ //limitar al borde del mapa
                            Escribir("Jugada imposible");
                            contErr = contErr +1;
                        } Sino  {
                                Si ( mapaEnUso[cooPlay[0]-1][cooPlay[1]][0].equals(esVa) ||
mapaEnUso[cooPlay[0]-1][cooPlay[1]][0].equals(coin)){
                                mapaEnUso[cooPlay[0]-1][cooPlay[1]][0] = play;
                                mapaEnUso[cooPlay[0]][cooPlay[1]][0] = esVa;
                                cooPlay[0] = cooPlay[0]-1; //solo se mueve en x
                            }Sino {
                                Si (mapaEnUso[cooPlay[0]-1][cooPlay[1]][0].equals(exit)){
                                                                                    Si  (contOro  >=
(mapaEnUso[cooPlay[0]-1][cooPlay[1]][1])){
                                        Escribir("¡Lograste completar el Laberinto   !");
                                        Escribir("¡Felicidades!");
                                        cantPlayerWin = cantPlayerWin + 1;
                                        contMapaPrincipal[0][1] = contMapaPrincipal[0][1] +
1;
                                        finjuego = true;
                                    }Sino {
                                        Escribir("Jugada imposible");
                                                    Escribir("Esta salida necesita " +
mapaEnUso[cooPlay[0]-1][cooPlay[1]][1] + " de oro");
                                     }}Sino {Escribir("Jugada imposible, la casilla ya está
ocupada");}}}
                        finTurno = true;
                        alto;
                    caso derecha:
                        Si ((cooPlay[0] + 1) > limitador  ){ //limitar al borde del mapa
                            Escribir("Jugada imposible");
                        } Sino  {
                                Si ( mapaEnUso[cooPlay[0]+1][cooPlay[1]][0].equals(esVa) ||
mapaEnUso[cooPlay[0]+1][cooPlay[1]][0].equals(coin)){
                                mapaEnUso[cooPlay[0]+1][cooPlay[1]][0] = play;
                                mapaEnUso[cooPlay[0]][cooPlay[1]][0] = esVa;
                                cooPlay[0] = cooPlay[0]+1; //solo se mueve en x
                            }Sino {
                                Si (mapaEnUso[cooPlay[0]+1][cooPlay[1]][0].equals(exit)){
                                                                                    Si  (contOro  >=
(mapaEnUso[cooPlay[0]+1][cooPlay[1]][1])){
                                        Escribir("¡Lograste completar el Laberinto   !");
                                        Escribir("¡Felicidades!");
                                        cantPlayerWin = cantPlayerWin + 1;
                                        contMapaPrincipal[0][1] = contMapaPrincipal[0][1] +
1;
                                        finjuego = true;
                                    }Sino {
                                        Escribir("Jugada imposible");
                                                    Escribir("Esta salida necesita " +
mapaEnUso[cooPlay[0]+1][cooPlay[1]][1] + " de oro");
```

```
                                    }}Sino {Escribir("Jugada imposible, la casilla ya está
ocupada");}}}
                        finTurno = true;
                        alto;
                    caso mirMapa:
                        mirarMapa();
                        finTurno = true;
                        alto;
                    caso coger:
                        Si (mapaEnUso[cooPlay[0]][cooPlay[1]][1].equals("1")){
                            contOro = contOro + NumAleatorio(15 - 5) + 5;
                            mapaEnUso[cooPlay[0]][cooPlay[1]][1] = "0";
                        }Sino {
                            Escribir("Jugada imposible");
                        }
                        finTurno = true;
                        alto;
                    caso salir:
                        finjuego = true;
                        finTurno = true;
                        alto;
                    caso oroDisp:
                        Escribir("El oro dispoible actual es de: " + contOro);
                        finTurno = true;
                        alto;
                    por defecto:
                        Escribir("No se reconoce el comando , se le restará una cantidad de
oro y se ubicará en el mapa");
                         var everyelements = false;
                        generarElementosRandom m(everyelements);
                        contErr = contErr + 1;
                        Si (contErr == cantComandToEnd){
                            finjuego = true;
                            finTurno = true;
                        }
                        alto;
                }
        }mientras (!finTurno);
    }

    public   void copiarMapaJugar(){
        Si (idMapa == 0){
            for ( var y = 0; y < 30; y++){
                for ( var x = 0; x < 30;x++){
                    mapaEnUso[x][y][0] = mapaPrincipal[x][y][0];
                    mapaEnUso[x][y][1] = mapaPrincipal[x][y][1];
                }
            }
        }
        Si (idMapa == 1){
            for ( var y = 0; y < 30; y++){
                for ( var x = 0; x < 30; x++){
                    mapaEnUso[x][y][0] = mapaRanuraA[x][y][0];
                    mapaEnUso[x][y][1] = mapaRanuraA[x][y][1];
                }
            }
        }
        Si (idMapa == 2){
            for ( var y = 0; y < 30; y++){
                for ( var x = 0; x < 30; x++){
                    mapaEnUso[x][y][0] = mapaRanuraB[x][y][0];
                    mapaEnUso[x][y][1] = mapaRanuraB[x][y][1];
                }
            }
        }
    }
    public   void generarElementosRandom ( var everyElements){
         //si everyElements -> true, genera jugador, bot y una salida; everyElements ->
false, genera una moneda por penalización
         var restriccionMapa = 30;
         var distBotJugador = 4; //Determina el area excluyente para generar al bot en el
mapa
```

```
            Si (everyElements){
                 var generado  = false;
                hacer  {
                    generado  = false;
                     var x = NumAleatorio(restriccionMapa);
                     var y = NumAleatorio(restriccionMapa);
                    Si (mapaEnUso[x][y][0].equals(esVa)) {
                        mapaEnUso[x][y][0] = play;
                        cooPlay[0] = x;
                        cooPlay[1] = y;
                        generado  = true;
                    }
                }mientras(!generado ); //Generar jugador

                 var xMen = cooPlay[0] - distBotJugador;
                 var xMay = cooPlay[0] + distBotJugador;
                 var yMen = cooPlay[1] - distBotJugador;
                 var yMay = cooPlay[1] + distBotJugador;
                //limitar bordes del mapa
                Si (xMen < 0) {
                    xMen = 0;
                }
                Si (xMay > restriccionMapa) {
                    xMay = (restriccionMapa - 1);
                }
                Si (yMen < 0) {
                    yMen = 0;
                }
                Si (yMay > restriccionMapa) {
                    yMay = (restriccionMapa - 1);
                }

                     var tokenGen = 0; //aprueba o desaprueba si coordenada aleatoria se
encuentra en rango de exclusión
                hacer  {
                       cooBot[0] = NumAleatorio(restriccionMapa);  //coordenadas X e Y del
bot
                    cooBot[1] = NumAleatorio(restriccionMapa);
                     Si (mapaEnUso[cooBot[0]][cooBot[1]][0].equals(esVa)) { //Si el espacio
aleatorio está vacío verSi icar si está en el rango de exclusión
                         Si (cooBot[0] >= xMen && cooBot[1] <= xMay) {
                             Si (cooBot[0] >= yMen && cooBot[1] <= yMay) {
                                 tokenGen = 0;
                             } Sino  {
                                 tokenGen = 1;
                             }
                         } Sino  {
                             tokenGen = 1;
                         }
                    } Sino  {
                         tokenGen = 0;
                    }
                } mientras (tokenGen != 1);
                mapaEnUso[cooBot[0]][cooBot[1]][0] = bot;
            }Sino {
                Si (contOro > 0){
                    //generar solo una moenda y penalizar
                     var generado  = false;
                    contOro = contOro - NumAleatorio(6)+1;
                    hacer  {
                        generado  = false;
                         var x = NumAleatorio(restriccionMapa);
                         var y = NumAleatorio(restriccionMapa);
                        Si (mapaEnUso[x][y][0].equals(esVa)) {
                            mapaEnUso[x][y][0] = coin;
                            mapaEnUso[x][y][1] = "1";
                            generado  = true;
                        }
                    }mientras(!generado );
                }Sino  {
                    Escribir("No tiene suficiente oro para penalizarlo");
```

```
                }

            }
    }
    public   void revelarMonedas(){
            for ( var y = 0; y < 30; y++){
                for( var x = 0; x < 30; x++){
                    Si (mapaEnUso[x][y][0].equals(coin)){
                        mapaEnUso[x][y][1] = "1"; //1 es moneda visible
                    }
                }
            }
     } //inicia todas las monedas encontradas con el estado  visible antes de ejecutar
primer turno
    public   void revelarMonedasEnJuego(){
            for ( var y = 0; y < 30; y++){
                for( var x = 0; x < 30; x++){
                    Si (mapaEnUso[x][y][1].equals("1")){
                                                                                    Si
(mapaEnUso[x][y][0].equals(play)||mapaEnUso[x][y][0].equals(bot)){
                            mapaEnUso[x][y][0] = mapaEnUso[x][y][0];
                        }Sino {
                            mapaEnUso[x][y][0] = coin;
                        }
                    }
                }
            }

    }
    public   void dibMapaJugar() {
        for ( var y = 0; y < 30; y++) {
            for ( var x = 0; x < 30; x++) {
                Escribir (mapaEnUso[x][y][0]); // i = x; j = y
            }
            Escribir(" ");
        }
    } //dibuja los mapas con elementos agregahacer s
    public   void mirarMapa() {
         var xMen;
         var xMay;
         var yMen;
         var yMay;

            //limitar casillas 5x5
            xMen = cooPlay[0] - 2;
            xMay = cooPlay[0] + 2;
            yMen = cooPlay[1] - 2;
            yMay = cooPlay[1] + 2;
            //Limitar a los bordes del mapa
            Si (xMen < 0) {
                xMen = 0;
            }
            Si (yMen < 0) {
                yMen = 0;
            }
            Si (xMay > 29) {
                xMay = 29;
            }
            Si (yMay > 29) {
                yMay = 29;
            }

            for ( var y = yMen; y <= yMay; y++) {
                for ( var x = xMen; x <= xMay; x++) {
                    Escribir (mapaEnUso[x][y][0]);
                }
                Escribir(" ");
            }

    } //imprime el comando  MIRAR para el jugador (5x5 por defecto)
    public   void mirarMapaBot(){
         var xMen = cooBot[0] - 2;
```

```
        var xMay = cooBot[0] + 2;
        var yMen = cooBot[1] - 2;
        var yMay = cooBot[1] + 2;

           Si (xMen < 0) {
               xMen = 0;
           }
           Si (yMen < 0) {
               yMen = 0;
           }
           Si (xMay > 29) {
               xMay = 29;
           }
           Si (yMay > 29) {
               yMay = 29;
           }
               for ( var y = yMen; y <= yMay; y++) { //Bot busca al jugador en rango
definihacer  5 x 5
               for ( var x = xMen; x <= xMay; x++) {
                   Si (mapaEnUso[x][y][0].equals(play)){ //i = x; j = y
                       findPlayer = true;
                       x = xMay;
                       y = yMay;
                   }Sino {
                       findPlayer = false;
                   }
               }
           }
       Si (findPlayer == true){
               Escribir("El bot te ha visto");
               cantBotViewPlayer = cantBotViewPlayer +1;
               dirMovBot = 4;
                           Si  (cooPlay[0]  ==  cooBot[0]){  //si  las  x  son  iguales
-------------------------------------------------------------------------
                   Si (cooPlay[1] < cooBot[1]){ //si jugador está arriba de bot
                                                                                   Si
((mapaEnUso[cooBot[0]][cooBot[1]-1][0].equals(esVa)||mapaEnUso[cooBot[0]][cooBot[1]-1][0].e
quals(play))||mapaEnUso[cooBot[0]][cooBot[1]-1][0].equals(coin)){
                           dirMovBot = 0;
                       }Sino {
                           dirMovBot = 4;//4 aleatorio
                       }
                   }Sino { //si jugador está abajo de bot
                                                                                   Si
((mapaEnUso[cooBot[0]][cooBot[1]+1][0].equals(esVa)||mapaEnUso[cooBot[0]][cooBot[1]+1][0].e
quals(play))||mapaEnUso[cooBot[0]][cooBot[1]+1][0].equals(coin)) {
                           dirMovBot = 2;
                       }Sino {
                           dirMovBot = 4;
                       }
                   }
               }
                           Si  (cooPlay[1]  ==  cooBot[1]){  //Si  las  Y  son  iguales
-------------------------------------------------------------------------
                   Si (cooPlay[0] < cooBot[0]){ //si jugador está a la izquierda de bot
                                                                                   Si
((mapaEnUso[cooBot[0]-1][cooBot[1]][0].equals(esVa)||mapaEnUso[cooBot[0]-1][cooBot[1]][0].e
quals(play))||mapaEnUso[cooBot[0]-1][cooBot[1]][0].equals(coin)){
                           dirMovBot = 3;
                       }Sino {
                           dirMovBot = 4;
                       }
                   }Sino { //si jugador está a la derecha de bot
                                                                                   Si
((mapaEnUso[cooBot[0]+1][cooBot[1]][0].equals(esVa)||mapaEnUso[cooBot[0]+1][cooBot[1]][0].e
quals(play))||mapaEnUso[cooBot[0]+1][cooBot[1]][0].equals(coin)) {
                           dirMovBot = 1;
                       }Sino {
                           dirMovBot = 4;
                       }
                   }
               }
```

```
    }
    public   void moverBot() {
        Si (findPlayer == true) {
            Escribir("El bot se ha movihacer ");
            Si (dirMovBot == 0) {
                mapaEnUso[cooBot[0]][cooBot[1] - 1][0] = bot;
                mapaEnUso[cooBot[0]][cooBot[1]][0] = esVa;
                cooBot[1] = cooBot[1] - 1;
            }
            Si (dirMovBot == 2) {
                mapaEnUso[cooBot[0]][cooBot[1] + 1][0] = bot;
                mapaEnUso[cooBot[0]][cooBot[1]][0] = esVa;
                cooBot[1] = cooBot[1] + 1;
            }
            Si (dirMovBot == 3) {
                mapaEnUso[cooBot[0] - 1][cooBot[1]][0] = bot;
                mapaEnUso[cooBot[0]][cooBot[1]][0] = esVa;
                cooBot[0] = cooBot[0] - 1;
            }
            Si (dirMovBot == 1) {
                mapaEnUso[cooBot[0] + 1][cooBot[1]][0] = bot;
                mapaEnUso[cooBot[0]][cooBot[1]][0] = esVa;
                cooBot[0] = cooBot[0] + 1;
            }
            Si (dirMovBot == 4) {
                movBotAleatorio();
            }
        } Sino  {//movimiento aleatorio
            movBotAleatorio();
        }

    }
    public   void movBotAleatorio(){
          var valido  = 0; //valida si el movimiento del bot es posible (valido  = 1 ->
posible), de lo contrario lo  varenta de nuevo
         var movBot; //0 arriba, 1 derecha, 2 abajo, 3 izquierda
            hacer  {
                movBot = NumAleatorio(4);
                evaluar (movBot){
                    caso 0: //arriba
                        Si ((cooBot[1]) > 0){ //limitar al borde del mapa
                                Si  (mapaEnUso[cooBot[0]][cooBot[1] - 1][0].equals(esVa) ||
mapaEnUso[cooBot[0]][cooBot[1] - 1][0].equals(coin)) {
                                cooBot[1] = cooBot[1] - 1;
                                mapaEnUso[cooBot[0]][cooBot[1]][0] = bot;
                                mapaEnUso[cooBot[0]][cooBot[1]+1][0] = esVa;
                                valido  = 1;
                            } Sino  {
                                valido  = 0;
                            }
                        }Sino {
                            valido  = 0;
                        }
                        alto;
                    caso 1://derecha
                        Si ((cooBot[0]) < 29){ //limitar al borde del mapa
                                    Si  (mapaEnUso[cooBot[0]+1][cooBot[1]][0].equals(esVa) ||
mapaEnUso[cooBot[0]+1][cooBot[1]][0].equals(coin)) {
                                cooBot[0] = cooBot[0] + 1;
                                mapaEnUso[cooBot[0]][cooBot[1]][0] = bot;
                                mapaEnUso[cooBot[0]-1][cooBot[1]][0] = esVa;
                                valido  = 1;
                            } Sino  {
                                valido  = 0;
                            }
                        }Sino {
                            valido  = 0;
                        }
                        alto;
                    caso 2: //abajo
                        Si ((cooBot[1]) < 29){ //limitar al borde del mapa
```

```
                                Si (mapaEnUso[cooBot[0]][cooBot[1]+1][0].equals(esVa) ||
mapaEnUso[cooBot[0]][cooBot[1]+1][0].equals(coin)) {
                                cooBot[1] = cooBot[1] + 1;
                                mapaEnUso[cooBot[0]][cooBot[1]][0] = bot;
                                mapaEnUso[cooBot[0]][cooBot[1]-1][0] = esVa;
                                valido  = 1;
                            } Sino  {
                                valido  = 0;
                            }
                        }Sino {
                            valido  = 0;
                        }
                        alto;
                    caso 3: //izquierda
                        Si ((cooBot[0]) > 0){ //limitar al borde del mapa
                                Si (mapaEnUso[cooBot[0]-1][cooBot[1]][0].equals(esVa) ||
mapaEnUso[cooBot[0]-1][cooBot[1]][0].equals(coin)) {
                                cooBot[0] = cooBot[0] - 1;
                                mapaEnUso[cooBot[0]][cooBot[1]][0] = bot;
                                mapaEnUso[cooBot[0]+1][cooBot[1]][0] = esVa;
                                valido  = 1;
                            } Sino  {
                                valido  = 0;
                            }
                        }Sino {
                            valido  = 0;
                        }
                        alto;
                }
            }mientras (valido !=1);
        }
    }
    public   void compBotGana(){
        Si (cooBot[0] == cooPlay[0] && cooBot[1]== cooPlay[1]){
            Escribir("Ha sihacer  atrapahacer  por el bot");
            cantBotWin = cantPlayerWin + 1;
            Si (idMapa == 0){
                contMapaPrincipal[0][2] = contMapaPrincipal[0][2] + 1;
            }
            Si (idMapa == 1){
                contMapaA[0][2] = contMapaA[0][2] + 1;
            }
            Si (idMapa == 2){
                contMapaB[0][2] = contMapaB[0][2] +1;
            }
            finjuego = true;
        }
    }//comprueba si el bot gana la partida

    //configuraciones generales
    public   void confTiempoBot(){
          Escribir("Por favor ingrese el tiempo (en segunhacer s) que desea que duren los
turnos del bot");
                                Escribir("    ---------------------------------    ALERTA
------------------------------------ ");
           Escribir("La jugabilidad puede verse afectada negativamente. Mecanicas como el
tiempo de  ");
         Escribir("visualización de monedas actuales y el mapa dependen del valor de espera
del bot");
        Escribir("Por favor ingrese el tiempo en milisegundos s.");
        hacer {
        esperaBot = (leer());
        }mientras(esperaBot < 0);
    }
    public   void confMapaView() {
        limpiarPantalla();
        Escribir("CONFIGURACION DE VISTA DE MAPA");
        Escribir("Por favor ingrese el tipo de vista que desea durante el juego");
        Escribir("Puede cambiar esta vista posteriormente en: Menú Inicial -> Configuración
de vista de mapa");
        Escribir(" ");
        Escribir("1. SIN VISTA POR  TURNOS (FIEL A LOS REQUERIMIENTOS DE LA PRÁCTICA)");
```

```
            Escribir("    Esta opción es una representación fiel de lo requerido  en la práctica.");
        Escribir("   No muestra ninguna parte del mapa a menos de que ingresemos el comando ");
        Escribir("   para observar una parte del mapa, pero gasta un turno.");
        Escribir(" ");
        Escribir("2. VISTA PARCIAL PERMANENTE POR TURNOS");
         Escribir("   Muestra una cuadrícula de 5x5 permanente con el jugador siempre en el centro.");
                Escribir("      Puede seleccionar esta opción por comodidad de juego o calificación.");
        Escribir(" ");
        Escribir("3. VISTA TOTAL PERMANENTE POR TURNOS");
        Escribir("    Muestra la totalidad del mapa después de cada turno.");
              Escribir("       Puede seleccionar esta opción por comodidad de juego o calificación.");
        confGrafica = leer();
        Si (confGrafica < 1 || confGrafica > 3){
            confGrafica = 1;
            Escribir("Ha ingresado  un valor no válido  para la vista de mapa");
                Escribir("Se ha aplicahacer  la configuración por defecto: SIN VISTA POR TURNOS");
        }
    }

    public   void mostrarMapa(){
        evaluar (confGrafica){
             //No hay caso 1, aún así se evalua confGrafica en otra función, por eso 1 es asignable a confGrafica
            caso 2:
                mirarMapa();
                alto;
            caso 3:
                dibMapaJugar();
                alto;
        }
    }

    //Funciones para reportes
    public   void reporteFinPartida(){
        Escribir(" ");
        Escribir("Estos son los Reportes de la partida terminada: ");
        Escribir(" ");
        Escribir("Cantidad de oro recolectahacer : "+ contOro);
                Escribir("Cantidad  de  movimientos  realizahacer  s  por  el  jugador:  " + contTurnoJugador );
        Escribir("Cantidad de movimientos realizahacer s por el bot: " + conTurnoBot);
               Escribir("Cantidad  de  veces  que  se  estuvo  en  la  visión  del  bot:  " + cantBotViewPlayer);

    }
    public   double promOroPartida(){
        Si (contPartida == 0){
            return 0;
        }Sino {
            return (contOroTotal/contPartida);
        }

    }
    public   double promMovimientos(){
        Si (contPartida == 0){
            return 0;
        }Sino {
            return (contMovTotal/contPartida);
        }

    }
    public    var mapaMasJugado (){
        Si (contMapaPrincipal[0][0]>contMapaA[0][0]) {
            Si (contMapaPrincipal[0][0] > contMapaB[0][0]) {
                return ("El mapa más jugado  es el mapa principal");
            } Sino  {
```

```
                return ("El mapa más jugado  es: " + nombreRanuraB);
            }
        }Sino {
            Si (contMapaA[0][0] > contMapaB[0][0]){
                return ("El mapa más jugado  es: " + nombreRanuraA);
            }Sino {
                return ("El mapa más jugado  es: " + nombreRanuraB);
            }
        }
    }
    public    var mapaMasganado (){
        Si (contMapaPrincipal[0][1]>contMapaA[0][1]) {
            Si (contMapaPrincipal[0][1] > contMapaB[0][1]) {
                return ("El mapa más ganado  es el mapa principal");
            } Sino  {
                return ("El mapa más ganado  es: " + nombreRanuraB);
            }
        }Sino {
            Si (contMapaA[0][0] > contMapaB[0][0]){
                return ("El mapa más ganado  es: " + nombreRanuraA);
            }Sino {
                return ("El mapa más ganado  es: " + nombreRanuraB);
            }
        }
    }
    public    var mapaMaserdido (){
        Si (contMapaPrincipal[0][2]>contMapaA[0][1]) {
            Si (contMapaPrincipal[0][2] > contMapaB[0][2]) {
                return ("El mapa más perdido  es el mapa principal");
            } Sino  {
                return ("El mapa más perdido  es: " + nombreRanuraB);
            }
        }Sino {
            Si (contMapaA[0][2] > contMapaB[0][2]){
                return ("El mapa más perdido  es: " + nombreRanuraA);
            }Sino {
                return ("El mapa más perdido  es: " + nombreRanuraB);
            }
        }
    }
    public   void reporteGeneral(){
        Escribir(" ");
        Escribir("Estos son los reportes de la última partida");
        reporteFinPartida();
        Escribir(" ");
        Escribir("Estos son los reportes generales: ");
        Escribir(" ");
        Escribir("Partidas ganadas por el bot: " + cantBotWin);
        Escribir("Partidas ganadas por el jugador: " + cantPlayerWin);
        Escribir("Promedio de oro por partida: " + promOroPartida());
        Escribir("Promedio de movimientos / partida: " +  promMovimientos());
        Escribir("El mapa más jugado  es: " + mapaMasJugado ());
        Escribir("El mapa más ganado  es: " + mapaMasganado ());
        Escribir("El mapa más perdido  es: " + mapaMaserdido ());
        Escribir("Total de mapas creados es: " + contRanura);

    }
}
```