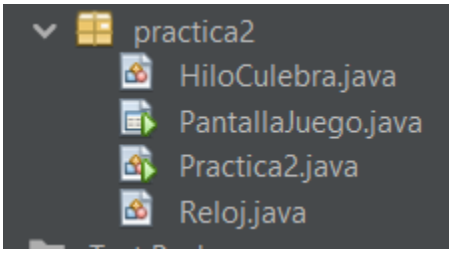
Manual Técnico

Clases:

El programa consiste de 4 clases



Consiste en 2 hilos (HiloCulebra, Reloj) La ventana del juego. (PantallaJuego). Y la clase main (Practica2)

1. PantallaJuego

```
public static String DatosPantallaCulebra[][] = new String[10][10];  // [X][Y]

public static int Num;

public static String RelojParaReporte;

public static String Movimientos="<br>";

public static void PonerTexto(String cod) {
    Reloj.setText(cod);
}

public static String TextoParaPantalla(String[][] DisCul) {...12 lines }
public static String TextoParaPantalla2(String[][] DisCul) {...12 lines }
public static String Dpad = "+x"; // +x, -x, y+, y-
static int PosicionFruta[][] = new int[2][1];
static boolean INICIO = true;
static boolean INICIORELOJ = false;
static int FrutaRandomInicial[] = new int[2];
static double MultiplicadorDificultad;
public static String ReporteInicial = "<h1><center>Práctica 2 | Reporte de resultados   <br>        </center></h1>\n"
        + "<p><b>Alumno: </b>Luis Daniel Castellanos Betancourt<br><b>Carnet: </b> 202103105<br><b>Sección: </b>F<br></p>\n"
        + "<h2><center>************Reporte de resultados************<br>  </center></h2>";
public static String ReporteParaArchivo;
public static String DifEnString;
```

En esta clase se encuentra la mayor parte de las variables usadas para el HiloCulebra, además de procesar la pantalla y botones de la aplicación

`public static String DatosPantallaCulebra[][] = new String[10][10];` Esta variable es ampliamente usada, ya que en este array se almacenan los caracteres (En código HTML), que conformaran el display de 10x10. En cada posición X y Y, el string almacenado puede ser un vacío, una fruta, o el cuerpo de la serpiente. Ya que el cuerpo de la culebra y la fruta usan color, se usa ■, para la culebra y ▣ para la fruta

`public static String TextoParaPantalla(String[][] DisCul)` Este método convierte el array anterior y lo transforma a un solo string, con los parámetros iniciales del html. Esto termina en el Jlabel de la ventana

```
static boolean INICIO = true;
static boolean INICIORELOJ = false;
```

Estas dos variables se encargarán de hacer que los hilos concluyan cuando se vuelvan false

```
static int FrutaRandomInicial[] = new int[2];
static double MultiplicadorDificultad;
```

La primera almacena la posición inicial aleatoria de la fruta, la segunda almacena ya sea 0.03, 0.06, o 0.09, para multiplicar la velocidad de la culebra

```
public static String ReporteInicial = "<h1><center>Práctica 2 | Reporte de resultados   <br>        </center></h1>\n"
        + "<p><b>Alumno: </b>Luis Daniel Castellanos Betancourt<br><b>Carnet: </b> 202103105<br><b>Sección: </b>F<br></p>\n"
        + "<h2><center>************Reporte de resultados************<br>  </center></h2>";
public static String ReporteParaArchivo;
public static String DifEnString;
```

Estas variables servirán mas tarde para hacer el reporte en html. ReporteInicial es el texto que siempre aparecerá al inicio del documento, y ReporteParaArchivo es usada ya para generar el archivo normal. DifEnString es solo para almacenar que dificultad se usó

BotonIniciar

Cuando este botón es iniciado, ocurren esta serie de pasos

- Se llena la matriz principal de vacíos
- Se determina la posición inicial aleatoria de la fruta
- Se pone la cabeza de la culebra en el centro
- Verifica que dificultad está seleccionada en el combobox, y aplica el multiplicador dificultad
- Imprime la pantalla por primera vez
- Y por ultimo desbloquea los botones de dirección

Luego de esto, al presionar los botones en cualquier dirección, se da inicio a los hilos Reloj e HiloCulebra. Estos últimos toman control de la pantalla y el display del reloj

Las variables de esta clase que no se explicaron serán explicadas mas adelante en las demás clases

2. Reloj

La única función del reloj es poner cuanto tiempo ha estado la serpiente en pantalla. El reloj cuenta minutos, segundos, y horas.

3. HiloCulebra

Esta clase es básicamente el juego en sí. Lo único que hacen las demás clases es controlar la pantalla y el reporte. En cierto modo, lo único que hace es manipular la ya mencionada DatosPantallaCulebra y mandarla a pantalla principal

En el metodo run(), hay un ciclo while, el cual ejecuta el juego. Conforme se explique lo que hay en este ciclo, se irán explicando los métodos usados por la clase

Antes del ciclo

```
System.out.println("inicia culebra");
for (int j = 0; j < 10; j++) { //Asigna timer en 0 para todos los espacios
    for (int k = 0; k < 10; k++) {
        Timer[0][j][k] = 0;
    }
}
```

Primero se inicia una variable llamada “Timer”, el timer es esencial para el tamaño de la culebra y se explicara posteriormente.

Durante el ciclo

Paso 1 moviendo a la serpiente:

Para entender como se ha hecho el juego, se tiene que entender que existen cuatro posibles estados en la matriz DatosPantallaCulebra. Estos son

- Vacío (no hay nada)
- Cuerpo (Manejado por timer)
- Cabeza (Parte que se mueve a cualquier direccion)
- Fruta

Cuando inicia el ciclo, se ejecuta el metodo…

```
public static String[][] MueveCulebra(
```

Esto lo que hace es mover la cabeza al lado que indiquen los botones de la pantalla inicial

```
private void pyActionPerformed(java.awt.event.ActionEvent evt) {...16 lines }
private void mxActionPerformed(java.awt.event.ActionEvent evt) {...16 lines }
private void myActionPerformed(java.awt.event.ActionEvent evt) {...16 lines }
private void pxActionPerformed(java.awt.event.ActionEvent evt) {...16 lines }
```

Cuando la serpiente se ha movido, por ejemplo a la ubicación (x+1,y+1), se almacena en el array Timer el valor del tamaño de la culebra en la posición (x,y). En cada iteración del ciclo, en esa posición del timer se le restará 1. Por ejemplo, si el tamaño de la culebra es 5, entonces, cuando la culebra se haya movido a (x+1,y+1), el timer en la posicion(x,y) será de 5. Si se detecta que en esa posicion el timer es 5, entonces imprimirá su cuerpo. Pero como se va restando el timer en cada iteración, desaparecerá después de 5 iteraciones.

Esto genera un problema. Supongamos que el tamaño de la culebra es de 4. Si en cierto ciclo come una fruta, la serpiente tardará en crecer a tamaño 5, 4 iteraciones. Por eso se implemento un

método que corrige el tamaño de la culebra para que cada vez que coma una fruta, crezca inmediatamente `public static void CreceCulebra()`

Otro problema que encontré, es que cada vez que la culebra chocaba con la pared el programa se detenia y generaba una excepción. Use un Try Catch para que cada vez que arrojara esta excepción terminara el juego. Lo que sucede cuadno el juego acaba se explicara mas tarde.

Paso 2 Detección de frutas:

Cuando la posicion x y de la cabeza = posicion x y de la fruta, se ejecutan una serie de pasos que son los siguientes:

- Se teletransporta la fruta a un nuevo lugar
- Crece el numero del tamaño, es decir, del timer
- Asimismo crece la velocidad

Paso 3 Restando en el Timer:

Aquí se ejecuta un ciclo anidado. Si detecta que el timer en tal posición es mayor a 1, lo resta por 1 y deja el cuerpo de la culebra. Si detecta que el timer es 0, entonces manda un vacío a la matriz datospantallaculebra

Paso 4 Imprimiendo datos en Pantalla:

En este paso se imprime la pantalla principal, el tamaño y el intervalo.

```
PantallaJuego.PantallaPrincipal.setText(TextoParaPantalla(DatosPantallaCulebra)); //imprime en pantalla la culebra
PantallaJuego.LabelTamano.setText(String.valueOf(SiclosDefault));
PantallaJuego.LabelIntervalo.setText(String.valueOf(VelocidadCulebra));
```

Detección de fin del juego:

Si el juego se pierde, ya sea colisionando o alcanzando el tamaño maxino (que se detecta al final del ciclo while) ejecutan los siguientes pasos, en ese orden

- Se mandan invierten variables boolean para detener el reloj
- Se imprime el mensaje de juego terminado
- Se vuelve a habilitar el botón iniciar en pantalla principal

Una vez hecho eso, termina el ciclo y se procede a generar el reporte en html con la sus respectivas clases.