Practica 2 segundo semestre 2022

LAB IPC1

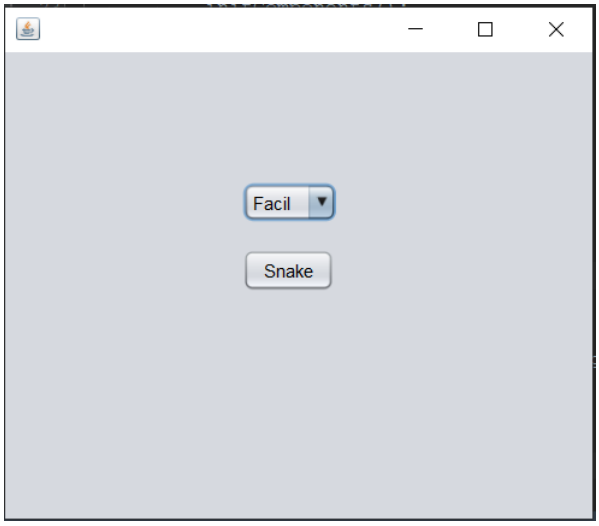
Manual de Usuario

Creado por: Juan Pablo Samayoa Ruiz

202109705

Apache NetBeans IDE 12.6

- **Ventana inicio**



-1 jComboBox
-1 boton

Variables globales

```
public static int intervalo;
public static String dificultad;
```

Se crean 2 tipos de variables publicas, una int y una string que se usaran en varias instancias.

Funcionamiento jComboBox:

```
public void SelecDif(){
    dificultad = (String)jComboBox1.getSelectedItem();
    switch(dificultad){
        case "Facil":
            intervalo = 856;
            break;
        case "Medio":
            intervalo = 830;
            break;
        case "Dificil":
            intervalo = 803;
            break;
    }
    JOptionPane.showMessageDialog(null, "Intervalo: " + String.valueOf(intervalo));
}
```

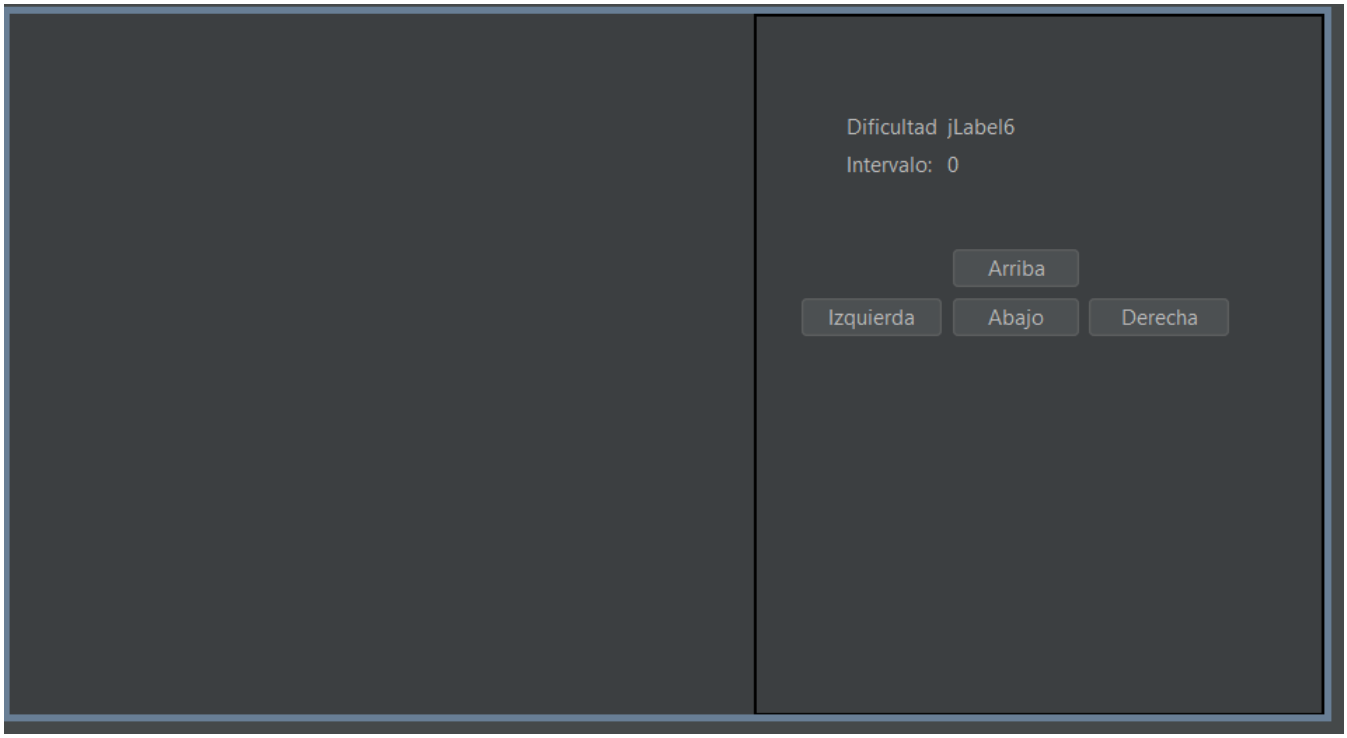
Se crea un switch and case para poder seleccionar la dificultad y el intervalo de esa dificultad

Funcionamiento jButton1

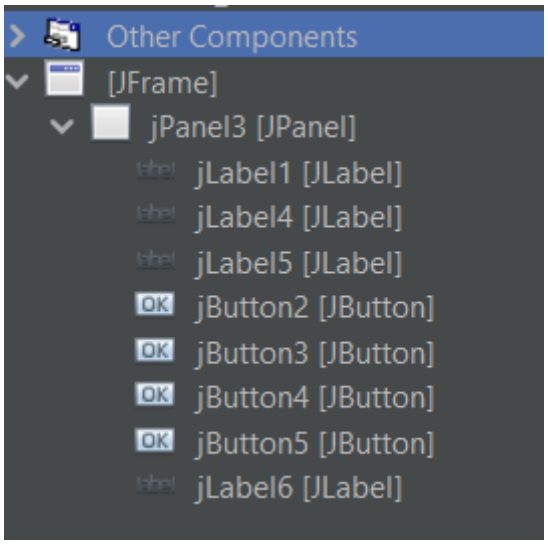
```
SelecDif();
Snake_ventana vent1 = new Snake_ventana();
        vent1.setVisible(true);
        this.hide();
```

Se llama al metodo utilizado para el switch and case de primero, así este podrá iniciar con los valores seleccionados en la parte del jComboBox y al momento de cambiar de ventana esta seguira almacenando los datos seleccionados.

Ventana Snake_ventana



Contenido:



Funcionamiento:

PanelSnake Snake

```
PanelSnake Snake;
```

Se crea una variable para poder utilizar la clase panel snake

Public Snake_ventana

```
public Snake_ventana() {
    this.setLocation(390,200);
    initComponents();

    jLabel6.setText("" + dificultad);
    jLabel5.setText("" + intervalo);

    Snake = new PanelSnake();
    this.add(Snake);
    Snake.setBounds(10, 10, 413, 396);
    Snake.setOpaque(false);

    PanelJuego tablero = new PanelJuego();
    this.add(tablero);
    tablero.setBounds(10, 10, 413, 396);

    this.setFocusable(true);

}
```

Se llaman varias clases con metodos de dibujado para que se sobrepongan al JFrame, además de que se cambia el texto de algunos jLabel dandoles los datos almacenados dentro de un variable ya predefinida, además de hacer que la ventana sea fijada.

formKeyPressed

```
switch(evt.getKeyCode()){
        case KeyEvent.VK_LEFT:
            if(direccion != 'R'){
                direccion = 'L';
            }
            break;
        case KeyEvent.VK_RIGHT:
            if(direccion != 'L'){
                direccion = 'R';
            }
            break;
```

```
            break;
        case KeyEvent.VK_DOWN:
            if(direccion != 'D'){
                direccion = 'U';
            }
            break;
        case KeyEvent.VK_UP:
            if(direccion != 'U'){
                direccion = 'D';
            }
            break;
```

Acciones para poder controlar la dirección a la que se dirije la serpiente

- **PanelJuego**

Imports

```
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JPanel;
```

Imports a utilizar en la clase

Variables a utilizar

```
Color colorFondo= Color.gray;
static final int tamax = 400;
static final int cant = 10;
static final int tam = (int)tamax/cant;
```

Estas variables son las que se utilizan para dar las dimensiones y el color del tablero a crear

Graficadora

```
public void paint(Graphics pintor){
    super.paint(pintor);
    pintor.setColor(colorFondo);
    for(int i= 0; i<cant; i++){
        for(int j= 0; j<cant; j++){
            pintor.fillRect(i*tam, j*tam, tam-1, tam-1);
        }
    }
}
```

En esta parte se realizaran ciclos para poder graficar un tablero simetrico de 10x10 según los valores de las variables anteriormente declaradas

- **PanelSnake**

Imports

```
import java.awt.Color;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.PrintWriter;
```

```
import javax.swing.JPanel;
import java.util.Random;
import javax.swing.JOptionPane;
```

```
import static practica2.ipc.pkg2s.Inicio.dificultad;
import static practica2.ipc.pkg2s.Inicio.intervalo;
import static practica2.ipc.pkg2s.PanelJuego.cant;
import static practica2.ipc.pkg2s.PanelJuego.tam;
```

Imports a utilizar para toda la clase

Variables a utilizar

```
Color colorSnake= Color.green;
Color colorComida= Color.blue;

static final int CuerpoT_Serpiente = 26;
int[] serpienteX = new int[CuerpoT_Serpiente];
int[] serpienteY = new int[CuerpoT_Serpiente];
public static int cuerpo_snake = 22;
public static int NomManzanas;

int comidaX;
int comidaY;

public static char direccion = 'R';

public static boolean estado = true;
Thread hilo;
Caminante camino;

Random random = new Random();
```

En esta parte se declaran varias variables a utilizar, tales como los colores de la serpiente, tamaño, comida, color de la comida, valores de tipo char para poder controlar el movimiento de la serpiente, variables random para la generación de la comida e incluso hilos que ayudaran a controlar la velocidad de movimiento de la serpiente.

PanelSnake

```
public PanelSnake(){
    this.setFocusable(true);
    iniciarJuego();
}
```

En esta parte se encontrará el llamado al método de iniciar juego, el cual contiene todos los procesos para el funcionamiento del mismo

Metodo paint o graficadora

```
super.paint(pintor);
pintor.setColor(colorComida);
pintor.fillRect(comidaX, comidaY, tam-1, tam-1);
if(estado=true){
for(int i= 0; i<cuerpo_snake; i++){
    if(i==0){
    pintor.setColor(colorSnake);
    pintor.fillRect(serpienteX[i], serpienteY[i], tam-1, tam-1);
    }else{
        pintor.setColor(new Color(45,180,0));
    pintor.fillRect(serpienteX[i], serpienteY[i], tam-1, tam-1);
    }
}
pintor.setColor(Color.red);
pintor.setFont( new Font("Ink Free",Font.BOLD, 40));
FontMetrics metrics = getFontMetrics(pintor.getFont());
pintor.drawString("Score: "+NomManzanas, (416 - metrics.stringWidth("Score: "+NomManzanas))/2, pintor.getFont().getSize())
}else{
}
```

El metodo es lo que termina siendo la graficadora de lo que es las serpiente, su comida y el indicador de puntuación, de forma que termina usando varios arreglos para poder graficar bien lo que se le solicite

Metodo iniciarJuego

```
public void iniciarJuego(){
    generarComida();
    camino = new Caminante(this);
    hilo = new Thread(camino);
    hilo.start();
}
```

Control del uso del hilo y de la generación de la comida, esta es la parte donde se declara que el hilo se iniciará.

Metodo avazar

```
public void avanzar(){
    for(int i=cuerpo_snake; i>0; i--){
        serpienteX[i] = serpienteX[i-1];
        serpienteY[i] = serpienteY[i-1];
    }
    switch(direccion){
        case 'R':
            serpienteX[0]=serpienteX[0]+tam;
            break;
        case 'L':
            serpienteX[0]=serpienteX[0]-tam;
            break;
        case 'U':
            serpienteY[0]=serpienteY[0]+tam;
            break;
        case 'D':
            serpienteY[0]=serpienteY[0]-tam;
            break;
    }
}
```

Se declaran varios parámetros de movimiento para que la serpiente tenga su característico movimiento, esto utilizando arreglos para mantener un margen con respecto al tablero

Metodo colisiones

```
public void Colisiones(){
    for(int i =cuerpo_snake; i>0; i--){
        if((serpienteX[0] == serpienteX[i])&&(serpienteY[0] == serpienteY[i])){
            estado = false;
        }
    }

    if(serpienteX[0]<0){
        estado = false;
    }

    if(serpienteX[0]>396){
        estado = false;
    }

    if(serpienteY[0]<0){
        estado = false;
    }

    if(serpienteY[0] > 396){
        estado = false;
    }


    if(!estado){
        hilo = new Thread(camino);
        hilo.stop();
        JOptionPane.showMessageDialog(null, "Game over");
        System.exit(0);
    }
}
```

Este metodo calcula las coliciones y si la serpiente llega a hacer contacto con una pared o consigo misma este parará el hilo y generará un reporte de las estadisticas de juego

Metodo comerManzana

```
public void ComerManzana(){
    if((serpienteX[0]==comidaX)&&(serpienteY[0]==comidaY)){
        cuerpo_snake++;
        NomManzanas++;
        generarComida();
        if(cuerpo_snake == 25){
            JOptionPane.showMessageDialog(null, "Victoria: "+cuerpo_snake++);
            reporte();
            System.exit(0);
        }
    }
}
```

Es el metodo el cual almacena el score e incrementa el tamaño de la serpiente

Metodo de generación de reportes

```
String nombre = "Juan Pablo Samayoa Ruiz-202109705";
String Gamemode = dificultad;
int tiempo =0;
int inter = intervalo;
int tam_Snake = cuerpo_snake-1;
int movimientos = 0;

File archivo = new File("reportes\\202109705.html");

FileWriter escribir;

PrintWriter nuevaLinea;

if(!archivo.exists()){
try {
    archivo.createNewFile();
    escribir = new FileWriter(archivo,true);
    nuevaLinea = new PrintWriter(escribir);
    nuevaLinea.println("<!DOCTYPE html>\n" +
                       "<html lang=\"en\">\n" +
                       "<head>\n" +
                       "    <meta charset=\"UTF-8\">\n" +
                       "    <title>Document</title>\n" +
                       "</head>\n" +
                       "<body>");

    nuevaLinea.println("<h1>REPORTE</h1>");
    nuevaLinea.println("<table>");
```

```
nuevaLinea.println("<tr>");
nuevaLinea.println("<th> Jugador </th>");
nuevaLinea.println("<th> Dificultad </th>");
nuevaLinea.println("<th> Tiempo transcurrido </th>");
nuevaLinea.println("<th> Intervalo </th>");
nuevaLinea.println("<th> Tamaño serpiente </th>");
nuevaLinea.println("<th> Historial de moviemientos </th>");
nuevaLinea.println("</tr>");

nuevaLinea.println("<tr>");
nuevaLinea.println("<td>"+nombre+"</td>");
nuevaLinea.println("<td>"+Gamemode+"</td>");
nuevaLinea.println("<td>"+tiempo+"</td>");
nuevaLinea.println("<td>"+inter+"</td>");
nuevaLinea.println("<td>"+tam_Snake+"</td>");
nuevaLinea.println("<td>"+movimientos+"</td>");
nuevaLinea.println("</tr>");

nuevaLinea.println("</table>");

nuevaLinea.println("</body>\n" +
                   "</html>");
escribir.close();
```

```
} catch (Exception e) {

}
lse{
try {
    BufferedWriter bw = new BufferedWriter(new FileWriter("reportes\\202109705.html"));
    bw.write("");


    escribir = new FileWriter(archivo,true);
    nuevaLinea = new PrintWriter(escribir);
    nuevaLinea.println("<!DOCTYPE html>\n" +
                       "<html lang=\"en\">\n" +
                       "<head>\n" +
                       "    <meta charset=\"UTF-8\">\n" +
                       "    <title>Document</title>\n" +
                       "</head>\n" +
                       "<body>");

    nuevaLinea.println("<h1>REPORTE</h1>");
    nuevaLinea.println("<table>");
```

```
            nuevaLinea.println("<tr>");
            nuevaLinea.println("<th> Jugador </th>");
            nuevaLinea.println("<th> Dificultad </th>");
            nuevaLinea.println("<th> Tiempo transcurrido </th>");
            nuevaLinea.println("<th> Intervalo </th>");
            nuevaLinea.println("<th> Tamaño serpiente </th>");
            nuevaLinea.println("<th> Historial de moviemientos </th>");
            nuevaLinea.println("</tr>");


            nuevaLinea.println("<tr>");
            nuevaLinea.println("<td>"+nombre+"</td>");
            nuevaLinea.println("<td>"+Gamemode+"</td>");
            nuevaLinea.println("<td>"+tiempo+"</td>");
            nuevaLinea.println("<td>"+inter+"</td>");
            nuevaLinea.println("<td>"+tam_Snake+"</td>");
            nuevaLinea.println("<td>"+movimientos+"</td>");
            nuevaLinea.println("</tr>");


            nuevaLinea.println("</table>");

            nuevaLinea.println("</body>\n" +
                               "</html>");
            escribir.close();
        } catch (Exception e) {
        }
}
```

Esta sección lo que hace es la creación de un reporte sobre las estadisticas del juego, este se debe crear al final para que obtenga toda la información de las variables

- **Caminante del abismo**

```
PanelSnake panel;
public Caminante(PanelSnake panel){
    this.panel=panel;
}
@Override
public void run() {
    while(estado){
    panel.avanzar();
    panel.Colisiones();
    panel.ComerManzana();
    panel.repaint();
    try {
        //JOptionPane.showMessageDialog(null, "Intervalo: " + String.valueOf(Inicio.intervalo));
            Thread.sleep(Inicio.intervalo);
            //JOptionPane.showMessageDialog(null, dificultad);

    } catch (InterruptedException ex) {
        Logger.getLogger(Caminante.class.getName()).log(Level.SEVERE, null, ex);
    }
    }
}

public void parar(){
    estado = false;
}
```

Este hilo es el que se encarga de que la serpiente se mueva en un lapso de tiempo determinado, además de verificar metodos como lo son el crecimiento de la serpiente o las colisiones, de forma que se mantedrá un timer que depende de las acciones de la serpiente