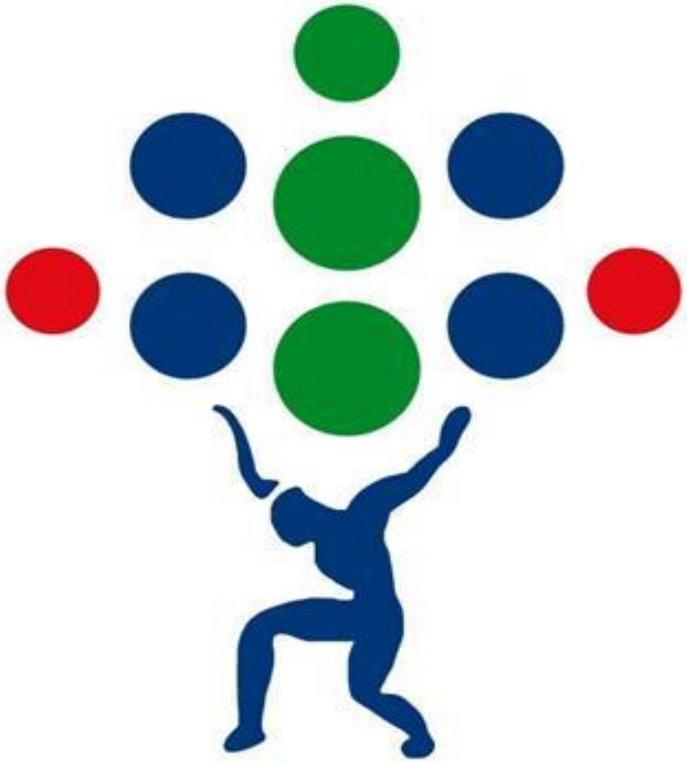
Programación I



UNIVERSIDAD PRIVADA
DOMINGO SAVIO

Integrantes: Leonardo Vargas Aguilera ***Código:*** 6259145

Jorge Enrique Velasquez Cuellar ***Código:***12389312

Docente: Ing. Gustavo Tantani Mamani

Link del Proyecto Completo: https://upds-my.sharepoint.com/:u:/g/personal/sc_leonardo_vargas_a_upds_net_bo/EXJ4V27BqutBurzxLzZ5jrkBU8Tm2jcRi8n2AZirCqO1iw?e=hpQiAj

Primer Proyecto "Pong"

Historia

Pong fue un videojuego de la primera generación de videoconsolas publicado por Atari, fue creado por Nolan Bushnell y lanzado el 29 de noviembre de 1972. Pong está basado en el deporte de tenis de mesa (Ping Pong). La palabra genérica “Pong” es usada para describir el género de videojuegos “bate y bola”

¿De qué trata Pong originalmente?

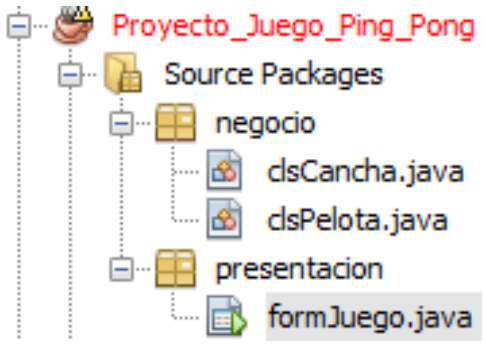
Pong es un juego de deportes en dos dimensiones que simula tenis de mesa. El jugador controla en el juego una paleta moviéndola verticalmente en la parte izquierda de la pantalla, y puede competir tanto contra otro oponente controlado por la computadora, como con otro jugador humano que controla una segunda paleta en la parte opuesta. Los jugadores pueden usar las paletas para pegarle a la pelota hacia un lado u otro. El objetivo consiste en que uno de los jugadores consiga más puntos que el oponente al finalizar el juego. Estos puntos se obtienen cuando el jugador adversario falla al devolver la pelota.

¿De qué trata nuestro Proyecto?

El siguiente proyecto trata de un juego (Pong) con un solo jugador donde el participante o jugador interactúa o juega Pong con el teclado, utilizando las flechas de izquierda y derecha para mover la “paleta” o “tablita” , una vez el jugador da clic en el menú de inicio de juego la pelota cae de manera aleatoria, el objetivo es que el jugador llegue a tocar o hacer interceptar la pelotita con la paleta, en caso de que no sea así, se le restara una vida, en caso de que el jugador llegue a quedarse sin vidas, el juego se termina. Si el jugador quisiera pausar el juego solo tendría que presionar la tecla “espacio” del teclado y para reanudar el juego solo presionaría la tecla “R” del teclado. En la parte superior derecha se encuentran las vidas que tiene el jugador y los puntos que va marcando una vez hace la intercepción tabla y pelota. Nuestro juego contiene un apartado de “EXTRAS” donde se encuentran también las instrucciones de cómo se juega y también los créditos o autores del juego.

A continuación, mostraremos las capturas de pantalla de los códigos desarrollados para nuestro proyecto, así mismo se podrá ver las clases creadas y el juego en acción.

Nuestro proyecto contiene dos clases y JFrame, las clases son la clase Cancha y la clase Pelota:



En la clase cancha posee

```
public class clsCancha extends Thread {

    //Cancha, color, tablero puntos, pelota, tablita
    private Graphics pintor;
    private Color color;
    private formJuego form;

    //Datos para la tablita
    //Posición de la tablita
    private int tx, ty, th, tw;

    //Pelota "Declarar la variable privada"
    private clsPelota objPelota;

    //Puntos
    public int puntos;

    //Vidas
    public int vidas;

    //CONSTRUCTOR
    public clsCancha(Graphics pintor, Color color, formJuego form) {
        this.pintor = pintor;
```

En constructor definimos el tamaño de la tablita y la posición donde se encuentra, también se encuentra la variable para los puntos y vidas que se muestran

```
    //CONSTRUCTOR
    public clsCancha(Graphics pintor, Color color, formJuego form) {
        this.pintor = pintor;
        this.color = color;
        //Tamaño de la tablita
        this.th = 15;
        this.tw = 95;
        //Posición de la tablita
        this.tx = 400 - tw / 2;
        this.ty = 350;
        //Puntos
        this.puntos = 0;
        //Vidas
        this.vidas = 3;
        //Form
        this.form = form;
        //Darle vida a la variable privada "Pelota"
        this.objPelota = new clsPelota(pintor, color);
    }

    //Método que se encarga de iniciar la pelota (Juego)
    public void IniciarPelota() {
        this.objPelota.start();
    }
```

En la siguiente captura se observa los métodos que utilizamos para poder pausar y reanudar el juego y también para terminar el mismo y está la lógica que utilizamos para el juego

```
//Método que se encarga de pausar la pelota (Juego)
public void SuspenderPelota() {
    this.objPelota.suspend();
}

//Método que se encarga de reanudar la pelota (Juego)
public void ContinuarPelota() {
    this.objPelota.resume();
}

//Método que se encarga de terminar el juego
public void TerminarJuego() {
    this.objPelota.stop();
}

@Override
public void run() {
    // Lógica para el juego
    while (true) {
        Rectangle tablita = new Rectangle(tx, ty, tw, th);
        Rectangle pelota = new Rectangle(objPelota.getPx(), objPelota.getPy(), 20, 20 + 10);
        boolean sw = tablita.intersects(pelota);
        if (sw) {
            //System.out.println("Chocaron...");
```

En la siguiente captura con boolean determinamos o vemos si la pelota intercepta con la tablita, si en caso chocan o interceptan, aumentaría un punto y en caso de que la pelotita no pueda interceptar con la tabla y choque con la pared de abajo, perdería una vida. Podemos ver el código que utilizamos para poder aumentar la dificultad de nuestro juego.

```
boolean sw = tablita.intersects(pelota);
if (sw) {
    //System.out.println("Chocaron...");
    //Para que aumente los puntos
    this.objPelota.NuevaDirección(Golpeo.TABLITA);
    this.puntos++;
    this.form.AumentarPuntos(this.puntos);
} else //Para que disminuya la vida
//System.out.println(objPelota.getPy());
if (objPelota.getPy() >= 400) {
    this.vidas--;
    this.form.ReducirVidas(this.vidas);
}
if (vidas == 0) {
    JOptionPane.showMessageDialog(form, "TE QUEDASTE SIN VIDAS");
    this.TerminarJuego();
}
//Para que aumente la velocidad de la pelota "Nivel 2"
if (this.puntos >= 10) {
    int Vpy = this.objPelota.getVariaciónEnY();
    if (Vpy == 5) {
        Vpy = Vpy + 5;
    }
    if (Vpy == -5) {
        Vpy = Vpy - 5;
```

Esta captura es la continuación del código para que aumente la dificultad del juego.

```
            if (this.puntos >= 10) {
                int Vpy = this.objPelota.getVariaciónEnY();
                if (Vpy == 5) {
                    Vpy = Vpy + 5;
                }
                if (Vpy == -5) {
                    Vpy = Vpy - 5;
                }
                objPelota.setVariaciónEnY(Vpy);
                int Vpx = this.objPelota.getVariaciónEnX();
                if (Vpx == 5) {
                    Vpx = Vpx + 5;
                }
                if (Vpx == -5) {
                    Vpx = Vpx - 5;
                }
                objPelota.setVariaciónEnX(Vpx);
            }
            try {
                sleep(50);
            } catch (Exception e) {
            }

        }
    }
```

Aquí tenemos los métodos para dibujar, borrar y para mover la tablita a derecha sin dejar rastros.

```
//Método que se encarga de dibujar la tablita
public void DibujarTablita() {
    this.pintor.setColor(Color.BLACK);
    this.pintor.fillRect(tx, ty, tw, th);
}

//Método que se encarga de limpiar la tablita
public void LimpiarTablita() {
    this.pintor.setColor(this.color);
    this.pintor.fillRect(tx, ty, tw, th);
}

//Método para mover a la derecha la tablita
public void moverDerechaTablita() {
    if (this.tx < 700) {
        LimpiarTablita();
        this.tx = this.tx + 20;
        DibujarTablita();
    }
}

//Método para mover a la izquierda la tablita
public void moverIzquierdaTablita() {
    if (this.tx > 10) {
```

Aquí vemos el código para hacer que la tabla se mueva hacia la izquierda sin dejar rastros.

```
//Método para mover a la izquierda la tablita
public void moverIzquierdaTablita() {
    if (this.tx > 10) {
        LimpiarTablita();
        this.tx = this.tx - 10;
        DibujarTablita();
    }
}
```

A continuación, se presentará la segunda clase utilizada “CLASE PELOTA”:

Aquí importamos los paquetes gráficos y de color

```
package negocio;
//Importamos los paquetes COLOR

import java.awt.Color;
//Importamos los paquetes GRÁFICOS
import java.awt.Graphics;
import java.util.Random;
import javax.swing.JOptionPane;
//import javax.swing.JOptionPane;
```

Aquí colocamos un Hilo y declaramos las variables:

```
 * @author Leonardo Vargas A. --- Jorge Enrique Velásquez C.
 */
//Colocar un HILO
public class clsPelota extends Thread {
//Posición de la pelota

    private int px, py, w, h;
    private int VariaciónEnX, VariaciónEnY;
    private Graphics pintor;
    private Color color;
```

Aquí se encuentran el constructor, el tamaño de la pelota y la posición donde saldrá una vez iniciado el juego, también contiene las variables para modificar el comportamiento de choque de la pelota. También se puede ver que declaramos las constantes para saber hacia donde o en donde choca la pelota.

```
//CONSTRUCTOR

public clsPelota(Graphics pintor, Color color) {
    //Tamaño de la pelota
    this.w = 20;
    this.h = 20;
    //Para que salga aleatoriamente desde arriba entre 1 y 800 (Ancho)
    this.px = (int) (Math.random() * 680) - 30;
    this.py = 100;
    this.pintor = pintor;
    this.color = color;
    //Cree estas variables para poder modificar su comportamiento de choque
    this.VariaciónEnX = -5;
    this.VariaciónEnY = 5;
}

//Aquí declaramos constantes, tienen un valor fijo
public enum Golpeo {
    ARRIBA, IZQUIERDA, DERECHA, TABLITA, ABAJO
```

Aquí utilizamos el Override que es el método para sobre escribir y programamos la dirección de la pelota para que pueda rebotar y no se pase del formulario.

```
//Método para sobrescribir

    @Override
    public void run() {
        while (true) {
            // Aqui hay que programar la direccion de la pelota..
            borrarP();
            this.py = this.py + this.VariaciónEnY;
            this.px = this.px + this.VariaciónEnX;
            pintarP();
            //Para que no pase la pared de la derecha
            if (px > 680) {
                //System.out.println("Right Crash");
                //this.NuevaDirección(Golpeo.DERECHA);
                this.VariaciónEnX = -5;
                //this.VariaciónEnY = -5;
            }
            //Para que no pase la pared de la izquierda
            if (px < 10) {
                //System.out.println("Left Crash");
                //this.NuevaDirección(Golpeo.IZQUIERDA);
                this.VariaciónEnX = 5;
                //this.VariaciónEnY = -5;
            }
```

Esta captura es la continuación de la anterior, en esta imagen se ve para que la pelota pueda rebotar en la parte superior del formulario y en caso de que la pelota choque la parte de abajo del formulario ,está el código para vuelva a salir de su lugar de origen en este caso de la parte superior.

```
        }
        //Para que no se pase arriba
        if (py < 70) {
            //System.out.println("Top Crash");
            //this.NuevaDirección(Golpeo.ARRIBA);
            this.VariaciónEnY = 5;
            //this.VariaciónEnX = 5;
        }
        //Para que la pelota regrese al lugar de origen
        if (this.py > 400) {
            //System.out.println("Hi Again");
            this.NuevaDirección(Golpeo.ABAJO);
            this.px = (int) (Math.random() * 680) - 30;
            this.py = 100;
            //this.VariaciónEnY = -5;
        }
        try {
            sleep(50);
        } catch (Exception e) {
        }

    }
}
```

A continuación, vemos los métodos para dibujar la pelotita y borrar, para que se mueva sin dejar rastros.

```
    //Para pintar la pelota
    private void pintarP() {
        this.pintor.setColor(Color.ORANGE);
        this.pintor.fillOval(px, py, w, h);
    }

    //Para borrar la pelota
    private void borrarP() {
        this.pintor.setColor(this.color);
        this.pintor.fillOval(px, py, w, h);
    }
```

Aquí vemos todos los GET y SET de la posición en “x” y “y” de la pelota , también los GET y SET del tamaño que tiene.

```
//Todos los GET y SET de las variables (px, py, w, h)
public int getPx() {
    return px;
}

public void setPx(int px) {
    this.px = px;
}

public int getPy() {
    return py;
}

public void setPy(int py) {
    this.py = py;
}

public int getW() {
    return w;
}

public void setW(int w) {
    this.w = w;
}

public int getH() {
    return h;
}

public void setH(int h) {
    this.h = h;
}

public int getVariaciónEnX() {
    return VariaciónEnX;
}

public void setVariaciónEnX(int VariaciónEnX) {
    this.VariaciónEnX = VariaciónEnX;
}

public int getVariaciónEnY() {
    return VariaciónEnY;
}

public void setVariaciónEnY(int VariaciónEnY) {
    this.VariaciónEnY = VariaciónEnY;
}
```

aquí vemos líneas de códigos que al principio utilizamos, pero luego la simplificamos y solo dejamos el código para cuando choque la tabla con la pelota y cambie su direccion.

```
//Líneas de código relacionado con el comportamiento de choque
    public void NuevaDirección(Golpeo objectCrash) {
        //"Random" Clase para sortear objetos (Entre 0 y 1)
        //int randomValue = (int) new Random().nextInt(positivo - negativo) + ne
        int Velocidad = 5;
        switch (objectCrash) {
//Para cambiar la dirección cuando toque la pared de arriba
            //case ARRIBA:
            //this.VariaciónEnY = Velocidad;
            //this.VariaciónEnX = Velocidad * arrayOfRandoms[randomValue];
            //break;
//Para cambiar la dirección cuando toque la pared de la derecha
            //case DERECHA:
            //this.VariaciónEnX = -Velocidad;
            //this.VariaciónEnY = Velocidad * arrayOfRandoms[randomValue];
            //break;
//Para que vuelva a caer la pelota cuando pase el límite
            //case ABAJO:
            //System.out.println("PERDISTE");
            //this.px = (int)(Math.random()*700)-30;
            //this.py = 100;
            //break;
//Para cambiar la dirección cuando toque la pared de la izquierda
            //case IZQUIERDA:
            //this.VariaciónEnX = Velocidad;
            //this.VariaciónEnY = Velocidad * arrayOfRandoms[randomValue];
            //break;
//Para cambiar la dirección cuando toque la tablita
            case TABLITA:
                this.VariaciónEnY = -Velocidad;
                //this.VariaciónEnX = Velocidad * arrayOfRandoms[randomValue];
                break;

        }

    }

}
```

En la parte del formulario utilizamos lo siguientes codigos:

Importamos los siguientes paquetes:

```
package presentacion;
//Importar los paquetes de gráficos de JAVA

import java.awt.Graphics;
//Importar los paquetes de COLOR
import java.awt.Color;
import javax.swing.JOptionPane;
//import javax.swing.JOptionPane;
//Importar la clase CANCHA
import negocio.clsCancha;
```

```
* @author Leonardo Vargas A. --- Jorge Enrique Velásquez C.
*/
public class formJuego extends javax.swing.JFrame {

    /**
     * Creates new form formJuego
     */
    //Hacer visible los gráficos
    private Graphics pintor;
    private clsCancha objCancha;

    public formJuego() {
        initComponents();
        this.pintor = this.getGraphics();
        //Llamar a la clase CANCHA
        this.objCancha = new clsCancha(this.getGraphics(), this.getBackground(), this);
    }

    //PUNTOS
    public void AumentarPuntos(int puntos) {
        this.jLabel2.setText(Integer.toString(puntos));
    }
```

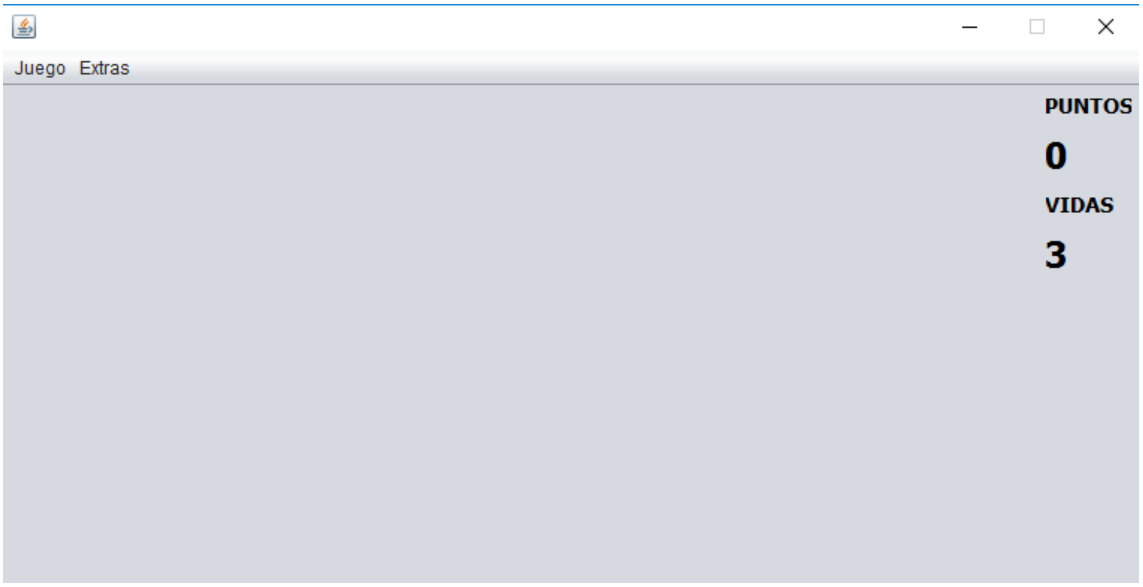
Aquí tenemos el label para que nos muestre las vidas que poseemos y también tenemos los eventos para que podamos utilizar el teclado para mover la tablita de izquierda a derecha.

```
    //VIDAS
    public void ReducirVidas(int vidas) {
        this.jLabel3.setText(Integer.toString(vidas));
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

//Todo lo relacionado con las teclas que serán para mover la tablita
    private void formKeyPressed(java.awt.event.KeyEvent evt) {
        //JOptionPane.showMessageDialog(this, evt.getKeyCode());
        //Creando variable para saber el código de la tecla que se presiona
        int cod = evt.getKeyCode();
        if (cod == 37) {
            this.objCancha.moverIzquierdaTablita();
        }
        if (cod == 39) {
            this.objCancha.moverDerechaTablita();
```

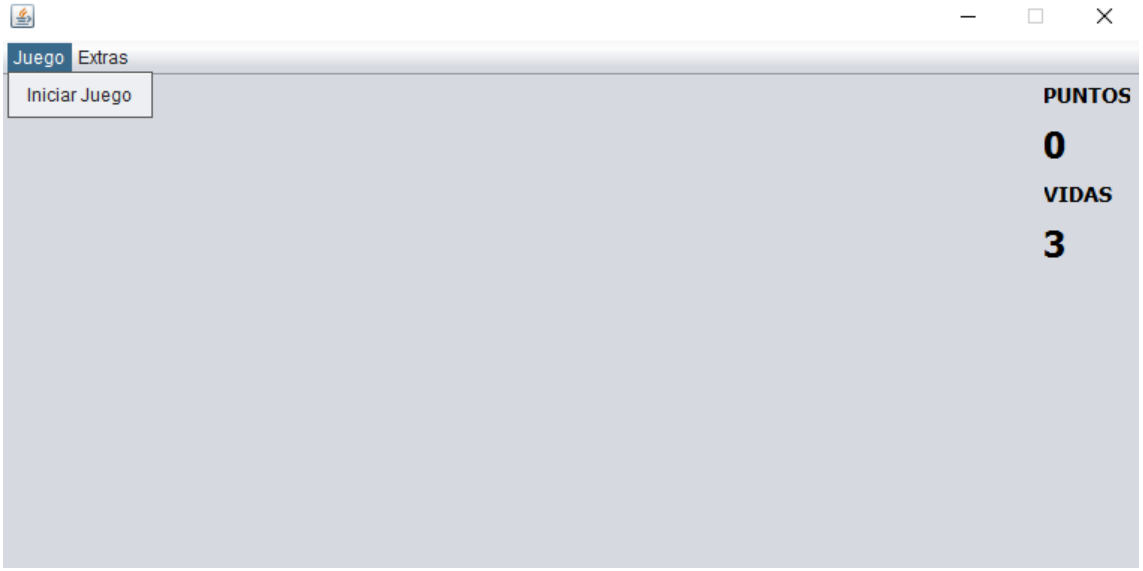
Aquí vemos los métodos que utilizamos para poder poner el juego en pausa y también para reanudarlo, así mismo se encuentra en método que hace iniciar el juego, los créditos y las instrucciones del juego.

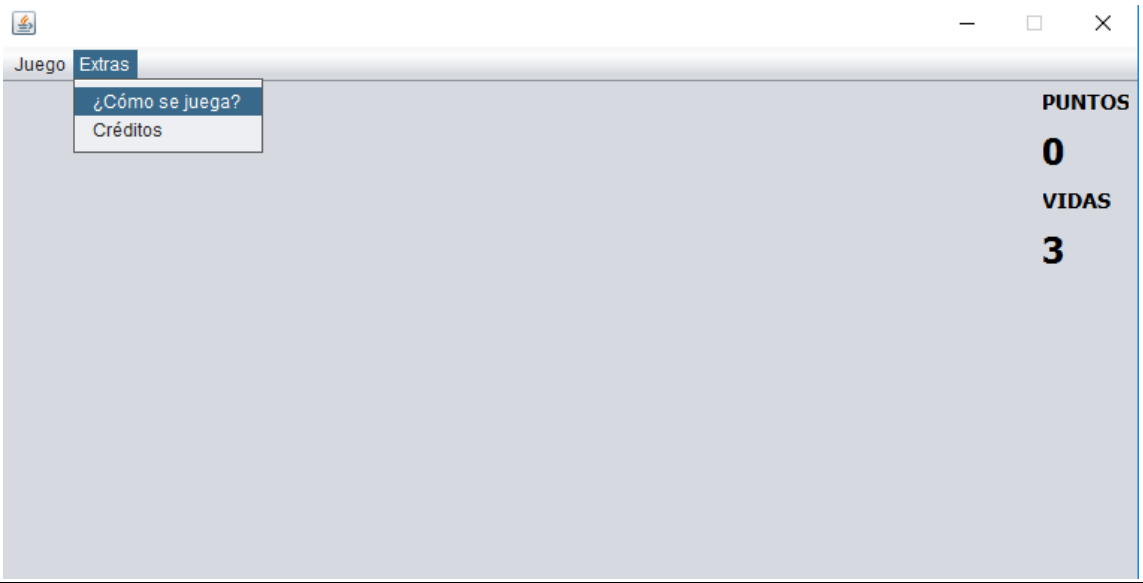
```
//Método para suspender la pelota (TECLA ESPACIO)
        if (cod == 32) {
            this.objCancha.SuspenderPelota();
        }
//Método para reanudar la pelota (TECLA R)
        if (cod == 82) {
            this.objCancha.ContinuarPelota();
        }

    }
//"Menú" Iniciar juego
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        objCancha.start();
        this.objCancha.IniciarPelota();
    }
//"Menú" Créditos
    private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
        JOptionPane.showMessageDialog(this, "Leonardo Vargas A. --- Jorge E. Velásquez C. --- Ing.Gustav
    }
//"Menú" ¿Cómo se juega?
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        JOptionPane.showMessageDialog(this, "(Iniciar Juego) para que caiga la pelota y mover con las fl
    }
```

A continuación, veremos el juego en acción:

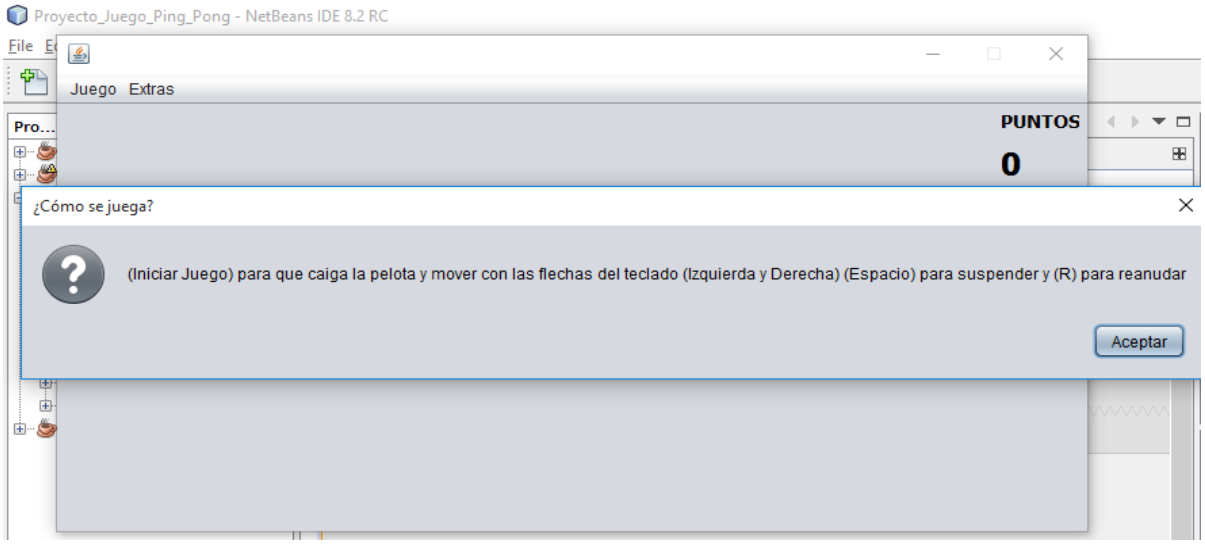


A continuación, vemos las opciones que tiene:

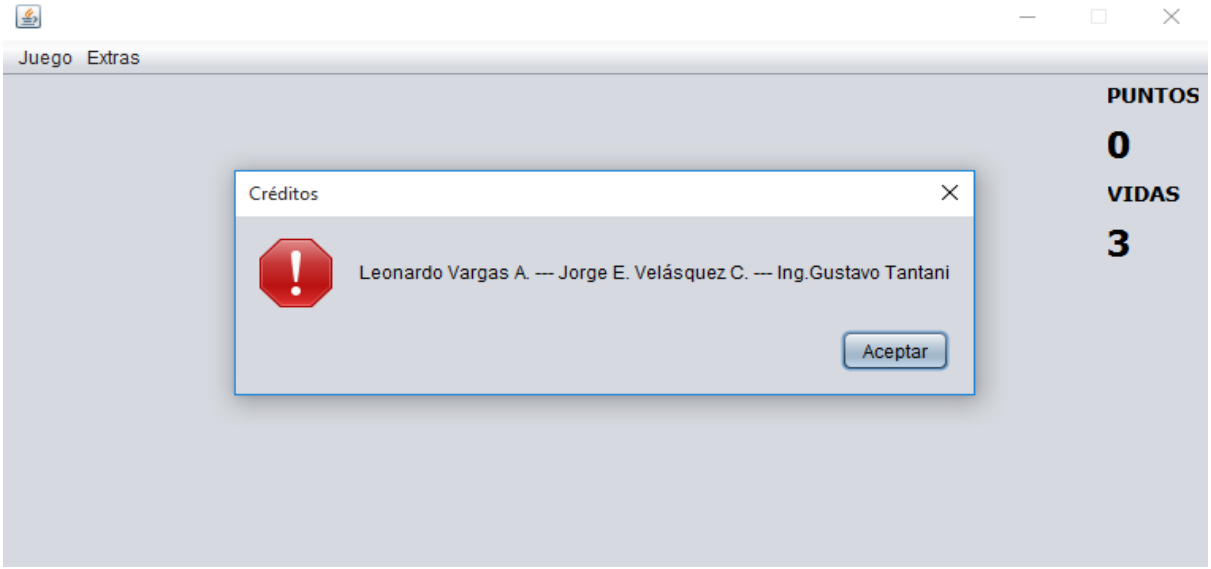




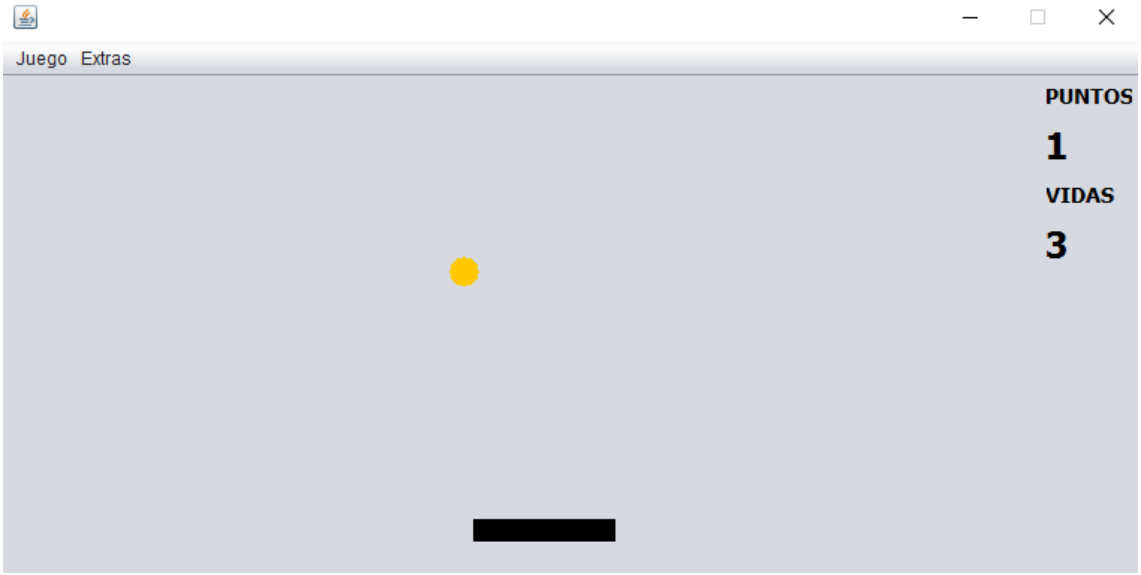
A continuación, vemos las instrucciones:



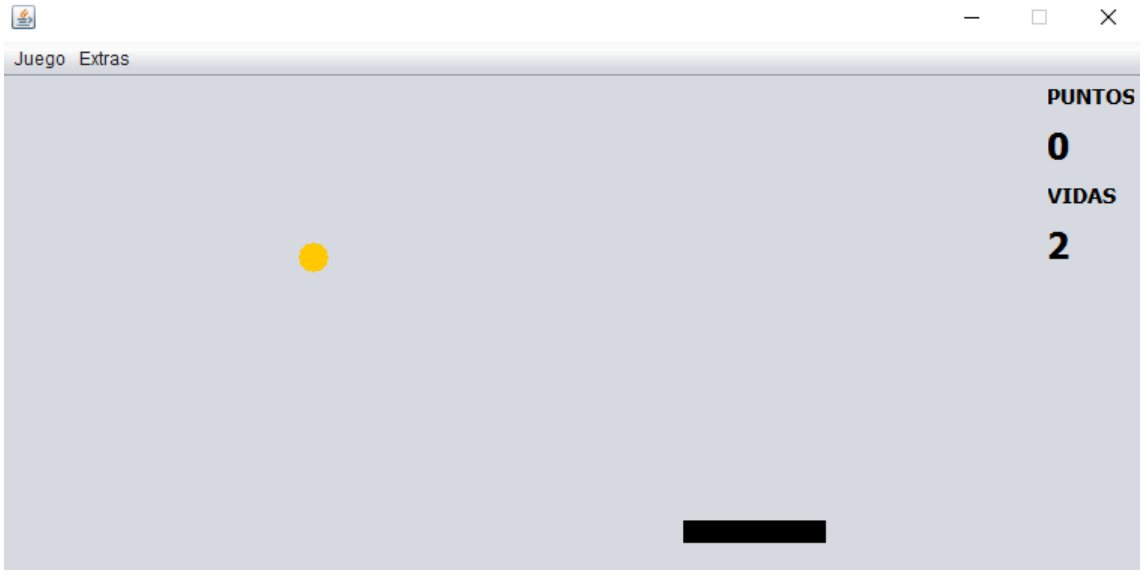
A continuación, vemos los créditos:



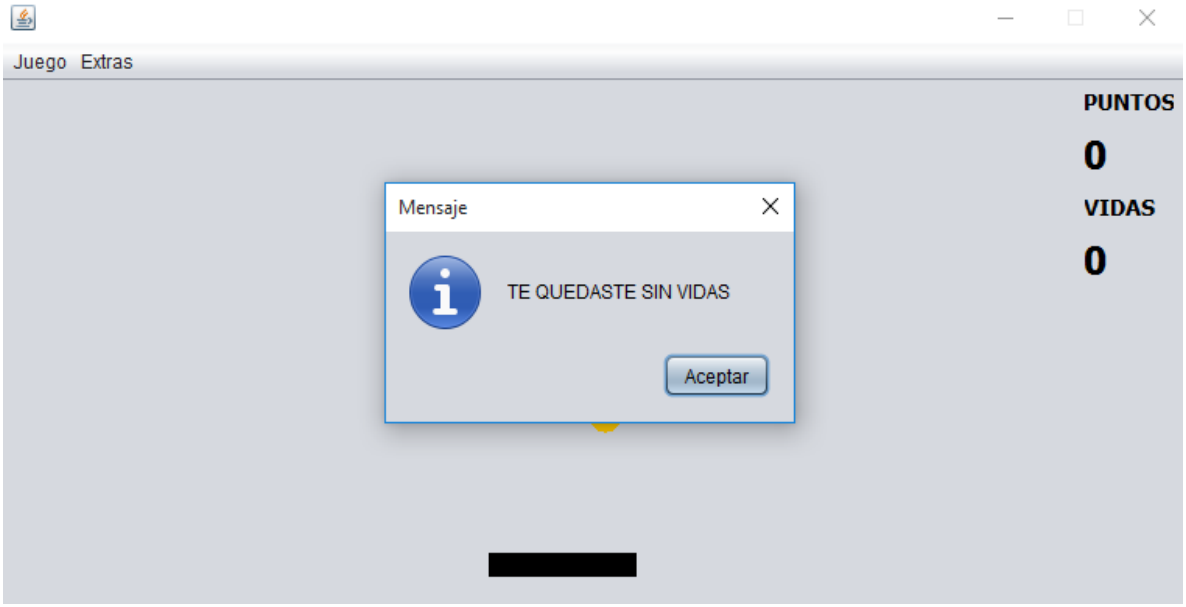
En la siguiente captura vemos el programa o juego en funcionamiento y aumentando los puntos cuando interceptan la pelota y la tabla:



En esta captura vemos como disminuye la vida cuando no lo logran interceptar la pelota y tabla y la pelota choca con la parte inferior del formulario:



El juego terminara cuando ya tengamos más vidas o una vez que las vidas sean igual a 0.



Proyecto Mejorado

Lo que hicimos para poder mejorar nuestro proyecto de acuerdo a lo que nos pedía, fue aumentarle o implementar dos vectores para crear las filas de los bloques.

A continuación, mostraremos las capturas de las variables y los códigos que utilizamos:

Declaramos las siguientes variables de la primera y segunda fila de bloques en la clase cancha también declaramos variables que determinan si es bloque, representaría con el numero1, en caso de que no fuera bloque lo representamos con número 0, también declaramos que cuando sea el primer bloque se represente con el numero 0, y cuando sea el segundo bloque se represente con el 1.

```
public final class clsCancha extends Thread {

    //Cancha, color, tablero puntos, pelota, tablita
    private Graphics pintor;
    private Color color;
    private formJuego form;
    //Datos para la tablita
    //Posición de la tablita
    private int tx, ty, th, tw;
    //Pelota "Declarar la variable privada"
    private clsPelota objPelota;
    //Puntos
    public int puntos;
    //Vidas
    public int vidas;

    public int[] PrimeraFilaDeBloques, SegundaFilaDeBloques;

    private final int ES_BLOQUE = 1;
    private final int NO_ES_BLOQUE = 0;

    private final int PRIMER_BLOQUE = 0;
    private final int SEGUNDO_BLOQUE = 1;
```

En la siguiente captura vemos que a cada uno de los vectores les asignamos 10 valores, que se convierten en bloques porque los pinta.

```
        this.PrimeraFilaDeBloques = new int[]{1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
        this.SegundaFilaDeBloques = new int[]{1, 1, 1, 1, 1, 1, 1, 1, 1, 1};

        this.pintarBloques();
    }
```

A través de este código hacemos que se pinten los bloques, El for es para recorrer el vector.

Si [i] = 1 pinta de color rojo el bloque, si [i] = 0 pinta de color de fondo.

El método que se utilizó para realizar la posición de cada uno de los bloques fue una formula rústica que multiplica la posición del vector por una constante definida (65).

Donde el ancho (width) es fijo y el alto (height) también.

```
public void pintarBloques() {
    for (int i = 0; i < this.PrimeraFilaDeBloques.length; i++) {
        if (this.PrimeraFilaDeBloques[i] == this.ES_BLOQUE) {
            this.pintor.setColor(Color.RED);
        } else {
            this.pintor.setColor(this.color);
        }
        this.pintor.clearRect((i * 65) + 40, 60, 60, 40);
        this.pintor.fillRect((i * 65) + 40, 60, 60, 40);
    }

    for (int i = 0; i < this.SegundaFilaDeBloques.length; i++) {
        if (this.SegundaFilaDeBloques[i] == this.ES_BLOQUE) {
            this.pintor.setColor(Color.RED);
        } else {
            this.pintor.setColor(this.color);
        }
        this.pintor.clearRect((i * 65) + 40, 120, 60, 40);
        this.pintor.fillRect((i * 65) + 40, 120, 60, 40);
    }
}
```

Aquí vemos que cuando hay un choque de pelota con bloque, aumenta un punto, también vemos que hace que cambie el valor de 1 a 0, es decir que cuando choquen pelota con bloque, se elimine el bloque.

```
public void choqueEnBloque(int fila, int columna) {
    this.aumentarPuntos();
    if (fila == this.PRIMER_BLOQUE) {
        this.PrimeraFilaDeBloques[columna] = this.NO_ES_BLOQUE;
    } else if (fila == this.SEGUNDO_BLOQUE) {
        this.SegundaFilaDeBloques[columna] = this.NO_ES_BLOQUE;
    }
}
```

En la clase pelota hicimos que veinte if que preguntan o hacen las intersecciones de la pelota con los bloques, son veinte if porque cada vector contiene diez elementos o bloques y para que la pelota tenga dirección luego de chocar con un bloque. Estos if hace que pregunte si la posición de la pelota está dentro de los 4 puntos de los bloques (x, y), luego se llama a la función “Choque en bloque” y se determina la posición del bloque que está chocando.

Y se cambia la variación en Y.

```
// FILA DE ARRIBA
// BLOQUE 0
if (this.px >= 40 + 10 && this.px <= 100 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[0] == 1) {
        // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(0, 0);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 1
if (this.px >= 105 + 10 && this.px <= 165 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[1] == 1) {
        // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(0, 1);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 2
if (this.px >= 170 + 10 && this.px <= 230 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[2] == 1) {
        // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(0, 2);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 3
if (this.px >= 235 + 10 && this.px <= 295 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[3] == 1) {
        // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(0, 3);
        this.VariaciónEnY = 5;
    }
}
```

```
//BLOQUE 4
if (this.px >= 260 + 10 && this.px <= 320 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[4] == 1) {
       // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(0, 4);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 5
if (this.px >= 325 + 10 && this.px <= 385 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[5] == 1) {
       // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(0, 5);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 6
if (this.px >= 390 + 10 && this.px <= 450 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[6] == 1) {
       // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(0, 6);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 7
if (this.px >= 455 + 10 && this.px <= 515 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[7] == 1) {
        //System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(0, 7);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 8
if (this.px >= 520 + 10 && this.px <= 580 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
    if (this.clCancha.PrimeraFilaDeBloques[8] == 1) {
}
  // BLOQUE 9
  if (this.px >= 585 + 10 && this.px <= 645 + 10 && this.py >= 60 + 10 && this.py <= 100 + 10) {
      if (this.clCancha.PrimeraFilaDeBloques[9] == 1) {
        //  System.out.println("Choco el ladrillo");
          this.clCancha.choqueEnBloque(0, 9);
          this.VariaciónEnY = 5;
      }
  }
  // FILA DE ABAJO
  // BLOQUE 0
  if (this.px >= 40 + 10 && this.px <= 100 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
      if (this.clCancha.SegundaFilaDeBloques[0] == 1) {
         // System.out.println("Choco el ladrillo");
          this.clCancha.choqueEnBloque(1, 0);
          this.VariaciónEnY = 5;
      }
  }
  // BLOQUE 1
  if (this.px >= 105 + 10 && this.px <= 165 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
      if (this.clCancha.SegundaFilaDeBloques[1] == 1) {
          //System.out.println("Choco el ladrillo");
          this.clCancha.choqueEnBloque(1, 1);
          this.VariaciónEnY = 5;
      }
  }
  // BLOQUE 2
  if (this.px >= 170 + 10 && this.px <= 230 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
      if (this.clCancha.SegundaFilaDeBloques[2] == 1) {
          //System.out.println("Choco el ladrillo");
          this.clCancha.choqueEnBloque(1, 2);
          this.VariaciónEnY = 5;
      }
```

```
// BLOQUE 3
if (this.px >= 235 + 10 && this.px <= 295 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
    if (this.clCancha.SegundaFilaDeBloques[3] == 1) {
       // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(1, 3);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 4
if (this.px >= 260 + 10 && this.px <= 320 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
    if (this.clCancha.SegundaFilaDeBloques[4] == 1) {
       // System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(1, 4);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 5
if (this.px >= 325 + 10 && this.px <= 385 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
    if (this.clCancha.SegundaFilaDeBloques[5] == 1) {
        //System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(1, 5);
        this.VariaciónEnY = 5;
    }
}
// BLOQUE 6
if (this.px >= 390 + 10 && this.px <= 450 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
    if (this.clCancha.SegundaFilaDeBloques[6] == 1) {
        //System.out.println("Choco el ladrillo");
        this.clCancha.choqueEnBloque(1, 6);
        this.VariaciónEnY = 5;
    }
}
```

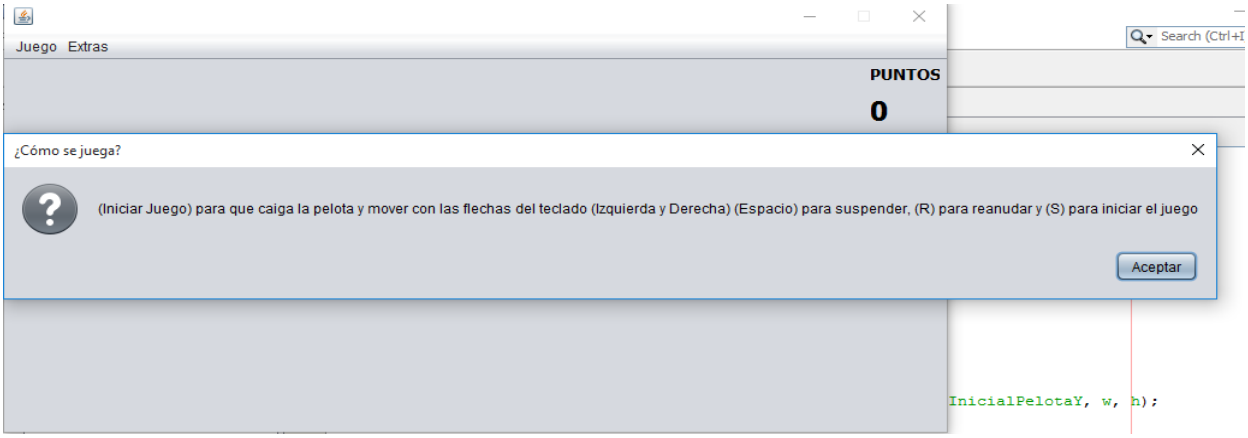
```
    // BLOQUE 7
    if (this.px >= 455 + 10 && this.px <= 515 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
        if (this.clCancha.SegundaFilaDeBloques[7] == 1) {
            //System.out.println("Choco el ladrillo");
            this.clCancha.choqueEnBloque(1, 7);
            this.VariaciónEnY = 5;
        }
    }
    // BLOQUE 8
    if (this.px >= 520 + 10 && this.px <= 580 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
        if (this.clCancha.SegundaFilaDeBloques[8] == 1) {
           // System.out.println("Choco el ladrillo");
            this.clCancha.choqueEnBloque(1, 8);
            this.VariaciónEnY = 5;
        }
    }
    // BLOQUE 9
    if (this.px >= 585 + 10 && this.px <= 645 + 10 && this.py >= 120 + 10 && this.py <= 160 + 10) {
        if (this.clCancha.SegundaFilaDeBloques[9] == 1) {
           // System.out.println("Choco el ladrillo");
            this.clCancha.choqueEnBloque(1, 9);
            this.VariaciónEnY = 5;
        }
    }

    try {
        sleep(50);
    } catch (Exception e) {
    }

}
```

En la siguiente captura hicimos un boolean que pregunta si aún existen bloques, en caso de que no hay ninguno, mandamos un mensaje que diga que ganamos el juego.

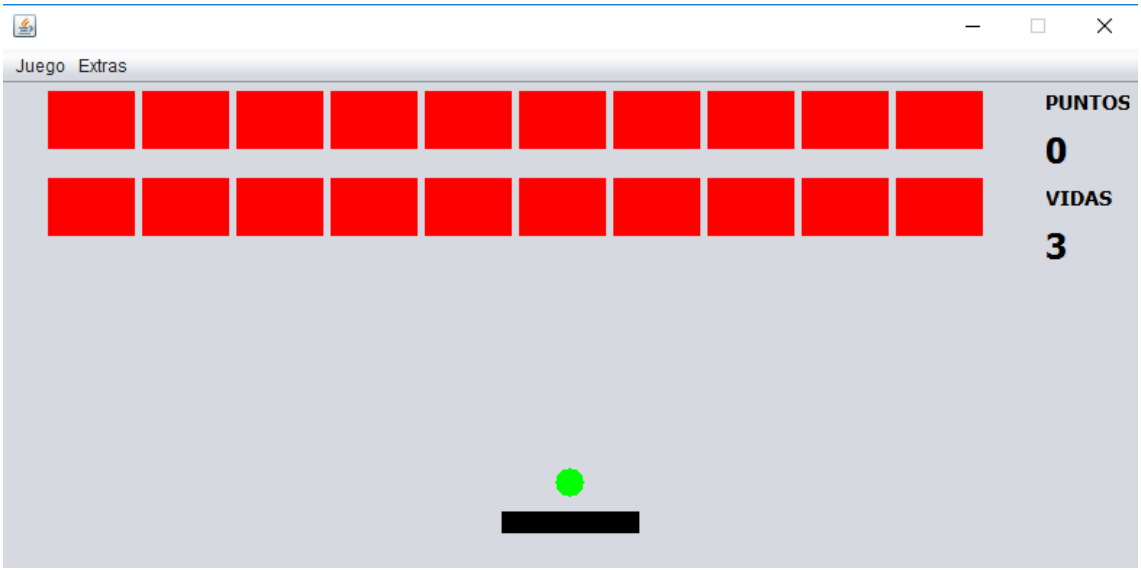
```
private boolean validarSiExistenBloques() {
    for (int i = 0; i < this.PrimeraFilaDeBloques.length; i++) {
        if (this.PrimeraFilaDeBloques[i] == this.ES_BLOQUE) {
            return false;
        }
    }

    for (int i = 0; i < this.SegundaFilaDeBloques.length; i++) {
        if (this.SegundaFilaDeBloques[i] == this.ES_BLOQUE) {
            return false;
        }
    }
    return true;
}
```

A continuación, veremos las capturas del juego ya en funcionamiento:

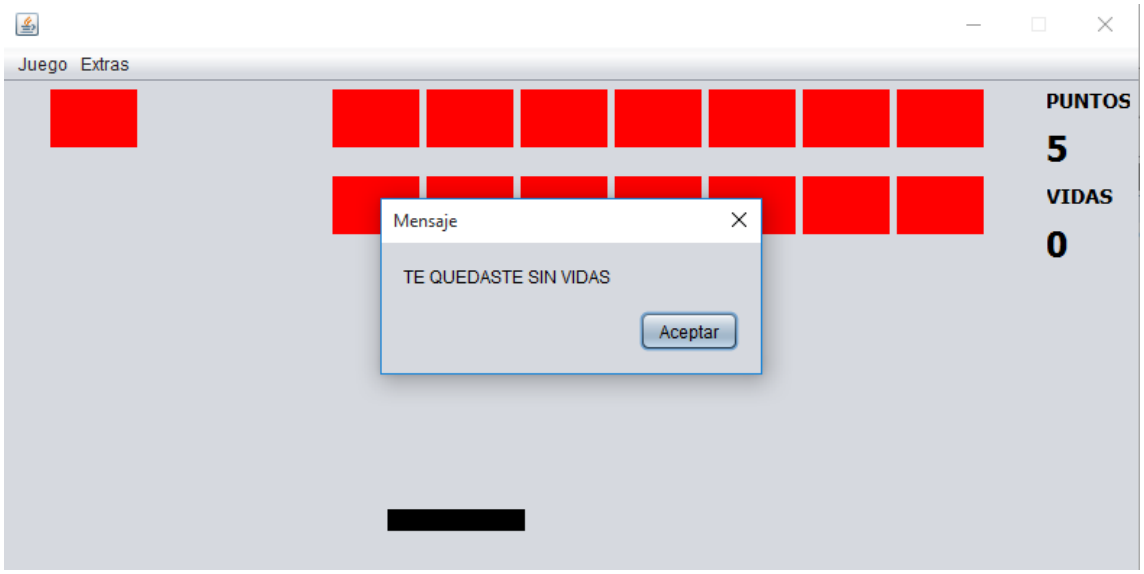
Actualizamos los controles y la guía de juego, la mostramos en el Joptionpane.



Una vez el usuario en iniciar juego, se pintan los bloques y la pelotita comienza casi junto con la tabla, una vez el usuario presiona la letra”S” del su teclado, comienza a moverse la pelotita y por ende comienza el juego contando los puntos cuando se intercepten la pelota y los bloques, en caso de que la pelota caiga sin interceptar la tabla, se disminuye una vida.



Cuando no puedan chocar la pelota con la tabla, la vida disminuye, una vez quede en cero vidas, hicimos que mande el siguiente mensaje:



Una vez no haya bloques, el juego se terminará con un mensaje diciendo que ganamos.