**INSTITUTO POLITÉCNICO NACIONAL**
**ESCUELA SUPERIOR DE CÓMPUTO**


## PROYECTO FINAL: “FLAPPY BIRD”

# MANUAL TÉCNICO

**ALUMNO:** Meza Vargas Brandon David.

**GRUPO:** 2CM1

**MATERIA:** Programación Orientada a Objetos

**FECHA:** 22-01-21

# CONTENIDO

**OBJETIVOS** ..... **3**

**INTRODUCCIÓN** ..... **3**

**FUNCIONAMIENTO** ..... **3**

**DIAGRAMAS** ..... **11**

## OBJETIVOS

- Representar la funcionalidad técnica de la estructura y diseño del programa del proyecto final de la materia programación orientada a objetos.
- Definir claramente como funcionan las clases usadas.

## INTROTUCCIÓN

Este manual técnico describe como funciona mi programa flappy bird para que una persona que tenga ciertas bases de programación pueda entender como funciona, incluso hacerle algunas mejoras o correcciones en caso de encontrar problemas.

Es importante tener en cuenta que es necesario contar con un visor de texto para ver el contenido de las clases.

## FUNCIONAMIENTO

En primer lugar, tenemos la clase que se encargar de mostrar la venta de nuestro juego, uno de sus parámetros es un objeto de tipo ProyectoFlappy que coloca elementos a la ventana, cuando se corre utiliza un serverSocket:

```
public class frame  extends JFrame{
    JLabel bien;
    JTextField n;
    String nombre;
    public frame(int ancho, int alto, String titulo, ProyectoFlappy flappy){
        try{
            flappy.serverSocket = new ServerSocket(7777);
        }catch(IOException e){
            System.out.println("Ya esta corriendo el juego");
            System.exit(0);
        }



        setTitle(titulo);
        pack(); //hace que la ventana coja el tamaño más pequeño posible que permita ver todos los componentes
        setSize(ancho + getInsets().left + getInsets().right, alto + getInsets().top + getInsets().bottom);

        setLocationRelativeTo(null);

        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);

        add(flappy);
        flappy.start();
```

Nuestra clase principal se llama ProyectoFlappy, esta hace que se inicie el juego, nuestra clase implementa la interfaz Runnable para trabajar con hilos en una parte,

esta clase consiste en un método sincronizado en donde inicializamos un hilo y lo ponemos en su estado listo con start(), posteriormente llamamos al metodo run()

```
public synchronized void start(){ //una parte de este codigo esta sincronizado
    activo=true;
    hilo = new Thread();
    hilo.start();
    nombre="jugador";
    run();

}
```

también tenemos el método tick() que hace el cambio de estado de la tierra y de nuestra ave;

```
public void tick(){
    if(!muerto && ini){
        controlaObjeto.tick();
        ti.tick();
    }
}
```

nuestro método init() inicializa todos los elementos que requerimos en nuestro juego, como el fondo, nuestro pájaro, además de los botones para iniciar el juego, de igual forma insertamos jugadores en una base de datos, así como agregar un keyListener y MouseListener que nos servirá para controlar a nuestra ave.

```
public void init(){
        try {
            cone= DriverManager.getConnection("jdbc:mysql://localhost:3306/flappy", "root", "");
           st=cone.createStatement();
           st.executeUpdate("Insert into jugador (usuario, puntos) Values ('"+nombre+"'"+", '"+0+"')");

       } catch (Exception a) {
        a.printStackTrace();
       }
    addKeyListener(new teclas());
    addMouseListener(new mouse());


    try {
        fondo=ImageIO.read(new File("C:\\Users\\PC\\Documents\\NetBeansProjects\\ProyectoFlappy\\images\\fondo0.png"));
    }catch(IOException e){  System.out.println("Image not found"); }
    try {
       perdio=ImageIO.read(new File("C:\\Users\\PC\\Documents\\NetBeansProjects\\ProyectoFlappy\\images\\gameo.png"));
    }catch(IOException e){  System.out.println("Image not found"); }
    try {
       boton=ImageIO.read(new File("C:\\Users\\PC\\Documents\\NetBeansProjects\\ProyectoFlappy\\images\\boton.png"));
    }catch(IOException e){  System.out.println("Image not found"); }
     try {
       listo=ImageIO.read(new File("C:\\Users\\PC\\Documents\\NetBeansProjects\\ProyectoFlappy\\images\\listo.png"));
    }catch(IOException e){  System.out.println("Image not found"); }

    ti = new tierra();
    bird = new Bird(50,50, 51, 36);
    inic=new botones(ProyectoFlappy.ANCHO/2-156/2,320,156,97,boton);
    res=new botones(ProyectoFlappy.ANCHO/2-156/2,320,156,97,boton);


}
```

posteriormente tenemos nuestro método render(), ayudándonos de la clase Graphics podemos dibujar imágenes en nuestra ventana, también dibujamos el texto que corresponderá al puntaje del jugador, con un if identificamos cuando un jugador muere y mostramos el puntaje mas alto que ha logrado en su partida, si apenas vamos a empezar a jugar se muestra el botón para jugar;

```
public void render(){
    BufferStrategy bs= this.getBufferStrategy();

    if(bs==null){
        createBufferStrategy(3);
        return;
    }

   Graphics g = bs.getDrawGraphics();
   g.drawImage(fondo,0,0,null);

   ti.render(g);

   controlaObjeto.render(g);
    g.setFont(new Font("Arial", Font.BOLD, 50));
    g.setColor(Color.WHITE);
    String s=Integer.toString(puntos);
    g.drawString(s,200, 40);

   if(muerto){
       g.drawImage(perdio,110,130,null);
       inic.render(g);
       g.setFont(new Font("Arial", Font.BOLD, 20));
       g.setColor(Color.WHITE);
       if(ProyectoFlappy.puntos>Bird.p)
       {
           g.drawString("Tu puntuación mas alta es: "+ProyectoFlappy.puntos,70, 330);
       }else if(Bird.p>ProyectoFlappy.puntos){
           g.drawString("Tu puntuación mas alta es: "+Bird.p,70, 330);
       }


   }
    if(!ini){

       g.drawImage(listo,110,130,null);
       res.render(g);
   }

    g.dispose();
    bs.show();
}
```

Finalmente tenemos nuestro método run, este se encarga de correr nuestro juego como su nombre lo dice, hará que se muestren los cambios como el aleteo de nuestra ave y el movimiento de la escena, esto lo hace gracias a que tenemos un ciclo while infinito que esta llamando constantemente a los métodos tick y render;

```
public void run() {
    init();
    this.requestFocus();
    long pasTiem=System.nanoTime();
    double cantTicks= 60.0;
    double ns = 1000000000/cantTicks;
    double delta=0;
    long timer = System.currentTimeMillis();

    while(activo){
        long now = System.nanoTime();
        delta+=(now - pasTiem) / ns;
        pasTiem=now;

        while(delta> 0){
            tick();

            render();

            delta--;
        }

        if(System.currentTimeMillis() - timer > 1000){
            timer+=1000;
            aparTubo.tick();
        }
    }
```

Otra de las clases es la clase abstracta objetos, esta declara dos métodos abstractos, tick() y render() para que aquella clase que extienda de esta los pueda implementar.

Pasando con la clase animación encontramos que esta se encarga de la animación de los distintos componentes que tiene nuestro juego, el método dentro de esta clase que se encarga de hacer esto es tick() que mientras estemos jugando hará el cambio de las imágenes para el aleteo del ave.

```
public void tick(){
    long pasTiem = (System.nanoTime() - tIni) / 1000000; //nanoTime devuelve eñ valor actual del temporizador de la maquina

    if(pasTiem >= delay && activo){
        imgAct++;
        tIni = System.nanoTime();
    }

    if(imgAct == imgs.length){
        imgAct=0;
        if(!activo)
            stop();
    }
}
```

Tenemos otra clase llamada controlaObjeto, esta se encarga de añadir los distintos objetos de nuestro juego a una lista vinculada, esto para llevar un control de aquellos

objetos que están en nuestro juego y en caso de no necesitar uno, retirarlo, esta clase también cuenta con una clase render() que se encarga de dibujar en pantalla los elementos que van ingresando.

```
public class controlaObjeto {

    public static LinkedList<Objetos> list = new LinkedList<Objetos>();
    public static void añadir(Objetos o){
        list.add(o);
    }
    public static void quitar(Objetos o){
        list.remove(o);
    }
    public static void render(Graphics g){
        Objetos aux=null;
        for(int i=0;i<list.size();i++){
            aux=list.get(i);
            aux.render(g);
        }
    }
```

Una clase muy importante es la llamada bird, esta se encarga de añadir el ave a nuestro juego, además de que aquí establecemos una velocidad con la que caerá nuestra ave si no estamos volando:

```
public Bird(int x, int y, int ancho, int largo) {
    super(x, y, ancho, largo);
    gravedad=0.3f;
    maxVel=12f;



    for(int i=0;i<imgs.length;i++)
     try {
        imgs[i]=ImageIO.read(new File("C:\\Users\\PC\\Documents\\NetBeansProjects\\ProyectoFlappy\\images\\bird"+(i)+".png"));
    }catch(IOException e){  System.out.println("Image not found"); }

    anima = new Animacion(this, 100, true, imgs);
    anima.start();
    controlaObjeto.añadir(this);
}
```

Esta clase también tiene un método llamado tick() aquí hacemos varias cosas, hacemos que el ave siempre este en el rango de nuestra ventana para que no desaparezca de esta, además de checar cuando colisiona con un tubo y así parar el juego, además de almacenar en una base de datos el puntaje obtenido por el jugador.

```
if(temp instanceof tubo){
    if(this.getBounds().intersects(temp.getBounds())){
        ProyectoFlappy.muerto=true;
try {
    cone= DriverManager.getConnection("jdbc:mysql://localhost:3306/flappy", "root", "");
   String query="update jugador set puntos= ? where usuario= ?";
   PreparedStatement pS=cone.prepareStatement(query);
   st = cone.createStatement();
   rs=st.executeQuery("SELECT puntos FROM jugador WHERE usuario='"+ProyectoFlappy.nombre+"'");

   if(rs.next()){
       p=rs.getInt("puntos");
       if(ProyectoFlappy.puntos>p){
        pS.setInt(1, ProyectoFlappy.puntos);
        pS.setString(2,ProyectoFlappy.nombre);
        pS.executeUpdate();
   }
```

Otro método que tenemos es el de la tierra con el que cargamos la tierra que va moviéndose en nuestro juego, esta animación también se realiza en esta clase con las clases tick() y render();

```
public tierra(){
    x1=0;
    x2= ProyectoFlappy.ANCHO;
    velX=3;
     try {
        tierra=ImageIO.read(new File("C:\\Users\\PC\\Documents\\NetBeansProjects\\ProyectoFlappy\\images\\tie.png"));
    }catch(IOException e){  System.out.println("Image not found"); }


public void tick(){
    x1-=velX;
    x2-=velX;

    if(x1+ProyectoFlappy.ANCHO<0)
            x1=ProyectoFlappy.ANCHO;
    if(x2+ProyectoFlappy.ANCHO<0)
            x2=ProyectoFlappy.ANCHO;
}

public void render(Graphics g){
    g.drawImage(tierra, x1,ProyectoFlappy.LARGO-168,null);
    g.drawImage(tierra,x2,ProyectoFlappy.LARGO-168,null);
}
```

Con las clases tipoTubo, aparTubo y tubo controlamos la aparición de los tubos.

La clase tipoTubo nos dice que un tubo puede estar arriba o abajo, esto lo indicamos poniendo la palabra reservada enum:

```
public enum tipoTubo { //el tubo solo puede estar abajo o arriba, por eso usamos enum
    ABAJO, ARRIBA;
}
```

Con la clase aparTubo vamos apareciendo los tubos arriba o debajo de una forma aleatoria, así como con una altura aleatoria para que no siempre tengan un mismo tamaño:

```
  private static Random rand=new Random();
  public static int tierraTam=168;
  public static int area=ProyectoFlappy.LARGO - tierraTam;
  public static int espacio=140;
  public static int minTam=40;
  public static int maxTam=area-espacio-minTam;
  public static int delay=1;
  public static int n;

  public static void aparecerTubo(){
      int altoUp=rand.nextInt(maxTam)+1;
      while(altoUp <minTam){
          altoUp = rand.nextInt(maxTam)+1;
      }

      int altoDown=area-espacio-altoUp;
      tubo tuboArri=new tubo(500,0,78,altoUp,tipoTubo.ARRIBA);
      tubo tuboAba=new tubo(500,altoUp+espacio,78,altoDown,tipoTubo.ABAJO);

      controlaObjeto.añadir(tuboArri);
      controlaObjeto.añadir(tuboAba);
  }
  public static void tick(){
      if(n <delay){
          n++;
      }else{
          aparecerTubo();
          n=0;
      }
  }
```

La clase tubo le asigna la textura al tubo, además de realizar la controlar la animación en el eje x para que vayan apareciendo desde la derecha y se desplacen a la izquierda:

```
    try {
        tub=ImageIO.read(new File("C:\\Users\\PC\\Documents\\NetBeansProjects\\ProyectoFlappy\\images\\tubo.png"));
    }catch(IOException e){  System.out.println("Image not found"); }
```

```
@Override
public void tick() {
    x-=velX;
    if(x+ancho<0){
        controlaObjeto.quitar(this);
        if(tipo==tipoTubo.ABAJO)
            ProyectoFlappy.puntos+=1;
    }
```

La clase botones la usamos para verificar cuando el jugador presiona el botón de empezar para que el juego empiece a correr o en caso de haber muerto el jugador volver a empezar a jugar, esto se verifica con el método click()

```
public static boolean click(int mX, int mY, botones btn){
    if( mX>= btn.x && mX<= btn.x+btn.ancho && mY>=btn.y && mY <=btn.y+btn.alto){
        return true;
    }else{
        return false;
    }
}
```

Si coincide que el click que damos esta en la misma posición que la imagen se devuelve true, si no regresa false.

Tenemos una clase que implementa la interfaz MouseListener, esta nos es de utilidad para saber en donde hace click el usuario, esto lo usamos en la clase botones explicada anteriormente:

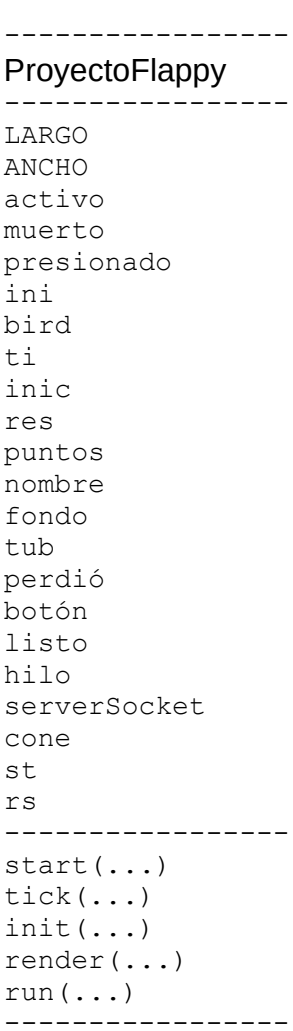
```
if(botones.click(e.getX(), e.getY(), ProyectoFlappy.inic)){
    if(ProyectoFlappy.muerto){
        ProyectoFlappy.presionado=true;
        controlaObjeto.list.clear();
        controlaObjeto.añadir(ProyectoFlappy.bird);

        ProyectoFlappy.puntos=0;
        ProyectoFlappy.muerto=false;
        ProyectoFlappy.ini=false;
        ProyectoFlappy.inic.presionado =false;

    }

}
 if(botones.click(e.getX(), e.getY(), ProyectoFlappy.res)){
    if(!ProyectoFlappy.ini){
        ProyectoFlappy.presionado=true;

       controlaObjeto.list.clear();
        controlaObjeto.añadir(ProyectoFlappy.bird);
        ProyectoFlappy.ini=true;
        ProyectoFlappy.muerto=false;
        ProyectoFlappy.res.presionado =false;

    }
```

Por último, pero muy importante, tenemos la clase teclas que implementa la interfaz KeyLsitener, gracias a esta clase establecemos que tecla debe presionar el jugador para que se eleve nuestra ave, cada que sea presionada esta tecla se reducirán 6 unidades a las coordenadas en y de nuestra ave, de esta forma su posición cambiara a mas arriba de nuestra ventana;

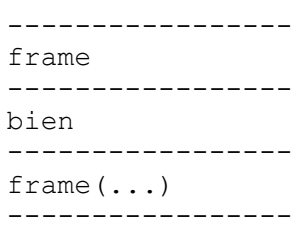
```
public class teclas implements KeyListener{

    @Override
    public void keyTyped(KeyEvent e) {

    }

    @Override
    public void keyPressed(KeyEvent e) {
        if(e.getKeyCode()==KeyEvent.VK_SPACE)
            ProyectoFlappy.bird.setVelY(-6);

    }
```

A continuación, veremos los diagramas de las clases usadas:

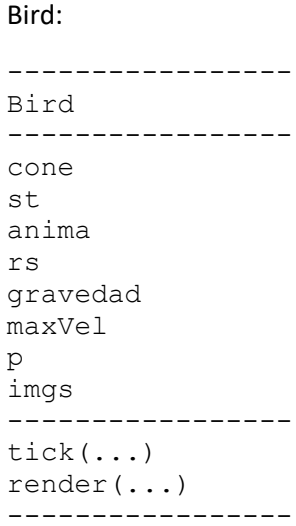
ProyectoFlappy:




frame:




Bird:



Animacion:

```
-----------------
Animacion
-----------------
x
y
imgAct
active
ciclo
objetivo
imgs
-----------------
Animacion(...)
Animacion(...)
Animacion(...)
render(...)
tick(...)
start(...)
stop(...)
-----------------
```

Objetos:

```
-----------------
Objetos
-----------------
x
y
ancho
alto
velX
velY
-----------------
tick(...)
render(...)
-----------------
```

controlaObjeto:

```
-----------------
controlaObjeto
-----------------
list
-----------------
añadir(...)
quitar(...)
render(...)
tick(...)
-----------------
```

Tierra

```
-----------------
tierra
-----------------
tierra
x1
x2
velX
-----------------
render(...)
tick(...)
-----------------
```

aparTubo:

```
-----------------
aparTubo
-----------------
rand
tierraTam
area
espacio
minTam
maxTam
delay
n
-----------------
aparecerTubo(...)
render(...)
-----------------
```

tubo

```
-----------------
tubo
-----------------
tipo
tub
-----------------
render(...)
tick(...)
-----------------
```

teclas

```
-----------------
teclas
-----------------
-----------------
KeyTyped(...)
tKeyPressed(...)
KeyReleased(...)
-----------------
```

mouse:

```
-----------------
mouse
-----------------
-----------------
mouseClicked(...)
mousePressed (...)
mouseReleased(...)
mouseEntered (...)
mouseExited(...)

-----------------
```

botones:

```
-----------------
botones
-----------------
x
y
ancho
alto
bot
presionado
-----------------
click(...)
render(...)
-----------------
```