ESCUELA POLITÉCNICA NACIONAL

VIDEOJUEGO PAC-MAN

GRUPO N° 2

Integrantes:

Sebastián Aisalla

Stalyn De La Cruz

Cristian López

Emilio Quinga

RESUMEN

El presente proyecto busca organizar de mejor manera y demostrar el dominio y entendimiento de los conocimientos de la programación orientada a objetos, aplicados juntamente con el lenguaje de programación Java.

. Tras una ardua labor de investigación previa a la construcción del programa se adquirió todos los conocimientos faltantes como el del uso de Java Swing para el desarrollo de la interfaz gráfica en el editor de código IntelliJ Idea, además del uso y configuración del JSON que fue primordial para dar las características y comportamiento de cada uno de los fantasmas que pretenden atacar al PacMan.

Luego de obtener todos los conocimientos faltantes, de solidificar los conocimientos adquiridos de la POO y de encontrar diversas fuentes con juegos creados de PacMan, se comenzó a desarrollar el videojuego en colaboración de todos los miembros del equipo.

Tras obtener el código con la mayor parte de la funcionalidad del videojuego se comenzó a desarrollar e implementar la interfaz gráfica con la ayuda de Java Swing y de Canva para la creación del diseño de cada una de las pantallas y del diseño de los botones.

Finalmente se fue controlando errores que se iban presentando a lo largo del desarrollo del código para poder ir puliendo y lograr poco a poco el resultado final, la tan anhelada recreación del videojuego.

Como fruto del trabajo se obtuvo un videojuego que se lo ha denominado Poli PacMan, el cual está modificado visualmente manteniendo como base la dinámica del muy popular videojuego PacMan todo ello organizando los conceptos y aplicaciones de la programación orientada a objetos.

1. INTRODUCCIÓN

1.1 INTRODUCCIÓN GENERAL

La creación de un juego en Java puede parecer una tarea abrumadora al principio, pero con las herramientas y recursos adecuados, es un proceso divertido y gratificante. Java es un lenguaje de programación popular y versátil que se utiliza en muchas aplicaciones, incluyendo el desarrollo de juegos. Es particularmente adecuado para el desarrollo de juegos dado que es un lenguaje orientado a objetos con una sintaxis clara y fácil de leer y entender. Por ejemplo, un juego clásico que se puede crear utilizando Java es Pac-Man.

Pac-Man es un videojuego clásico de arcade creado por el diseñador de videojuegos japonés Toru Iwatani y lanzado por primera vez en 1980 por la empresa Namco. En Pac-Man, el jugador controla un personaje amarillo redondeado que debe comer puntos y frutas mientras evita a los fantasmas que lo persiguen. El objetivo es completar cada nivel mientras se acumula la mayor cantidad de puntos posible. El diseño de personajes coloridos y el simple pero adictivo estilo de juego de Pac-Man han hecho que el juego sea un icono cultural en todo el mundo.

1.2 PRESENTACIÓN DEL PROYECTO

El presente proyecto está orientado a simular el juego de Pac-Man en Java con interfaz gráfica de usuario. Dicho juego poseerá los personajes y sus comportamientos correspondientes. Por otro lado, se utilizará un diseño de software adecuado, es decir, crear y ubicar los objetos con sus métodos en el lugar más idóneo.

2. OBJETIVOS Y METODOLOGÍA

2.1 OBJETIVOS

Objetivo general:

Desarrollar un juego funcional de Pac-Man en el lenguaje de Java, utilizando diversas herramientas y conceptos de programación orientada a objetos para consolidar y aplicar los conocimientos adquiridos en la asignatura.

Objetivos específicos:

Organizar de mejor manera los conceptos de programación orientada a objetos tales como, clases, objetos, atributos, interfaces y herencia, con la finalidad de elaborar un código legible.

Investigar e implementar el funcionamiento de Java Swing y JSON para desarrollar una interfaz de usuario amigable e interactiva.

2.2 METODOLOGÍA

Análisis de requerimientos: Se identifican los requerimientos funcionales y no funcionales del juego, como la mecánica de movimiento del personaje, la generación de laberintos, la interacción con los objetos del juego y la interfaz de usuario.

Diseño de la arquitectura: Se define la estructura del código y se decide cómo se organizarán los diferentes componentes del juego, como la representación del laberinto, los personajes del juego y la lógica de la jugabilidad. Se pueden utilizar patrones de diseño para facilitar la escalabilidad y la mantenibilidad del código.

Implementación de las clases principales: Se escriben las clases principales del juego, como la clase Pac-Man, la clase Fantasma y la clase Tablero, y se definen sus atributos y métodos. También se crean las interfaces gráficas de usuario.

Implementación de la lógica del juego: Se escriben los metodos que manejan la lógica del juego, como el movimiento de Pacman y los fantasmas, la detección de colisiones y la actualización de las puntuaciones.

Implementación de los objetos del juego: Se crean las clases para objetos del juego, como los puntos, las frutas, etc.

Implementación JSON a los fantasmas: Se escribe el algoritmo que controla el comportamiento de los fantasmas en el juego, teniendo en cuenta la posición de Pacman y las paredes del laberinto.

Pruebas y depuración: Se realizan pruebas exhaustivas del juego para identificar y corregir errores en el código, así como para verificar que los requerimientos del juego se cumplen.

Mejoras y optimización: Se pueden agregar mejoras adicionales al juego, como efectos de sonido, animaciones, dificultad progresiva, y se pueden hacer ajustes para optimizar el rendimiento.

3. INVESTIGACIÓN

3.1 SOFTWARE Y HERRAMIENTAS PARA LA REALIZACIÓN

En cuanto al código:

- IntelliJ IDEA

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) utilizado para desarrollar software en diferentes lenguajes de programación, incluyendo Java. IntelliJ IDEA es conocido

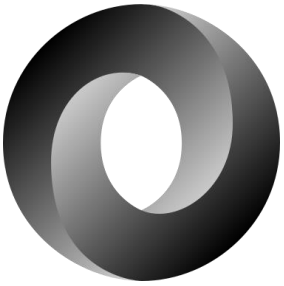
por su amplia gama de características, como autocompletado de código, refactorización de código, pruebas unitarias y depuración. IntelliJ IDEA es altamente compatible con Java, y ofrece herramientas específicas para el desarrollo de aplicaciones Java, como soporte para Java EE, frameworks populares de Java y tecnologías de servidor, y herramientas de análisis de código y corrección de errores.

- Java Swing

Java Swing es una biblioteca gráfica de usuario que se utiliza para desarrollar interfaces gráficas de usuario (GUI) en aplicaciones de escritorio en Java. Swing proporciona una amplia gama de componentes de interfaz de usuario, como botones, menús, etiquetas y campos de entrada, que se pueden personalizar y combinar para crear interfaces de usuario altamente sofisticadas. Además, Swing ofrece una serie de características avanzadas, como la capacidad de crear animaciones y efectos visuales, que permiten a los desarrolladores crear aplicaciones con una interfaz de usuario atractiva y dinámica.

- JSON

JSON (JavaScript Object Notation) es un formato de intercambio de datos ligero y fácil de leer utilizado para transmitir datos entre un cliente y un servidor en aplicaciones web y móviles. En Java, JSON es compatible con bibliotecas como Jackson y Gson, lo que permite a los desarrolladores trabajar con datos JSON en aplicaciones Java y convertirlos en objetos Java.



En cuanto al diseño:

- Canva

Canva es una plataforma de diseño gráfico, que permite al usuario crear sus propios diseños, añadiéndoles imágenes, otros elementos y textos. Ofrece una amplia gama de plantillas, herramientas de edición de imágenes y tipografía.

3.2 CONCLUSIÓN DE LA INVESTIGACIÓN

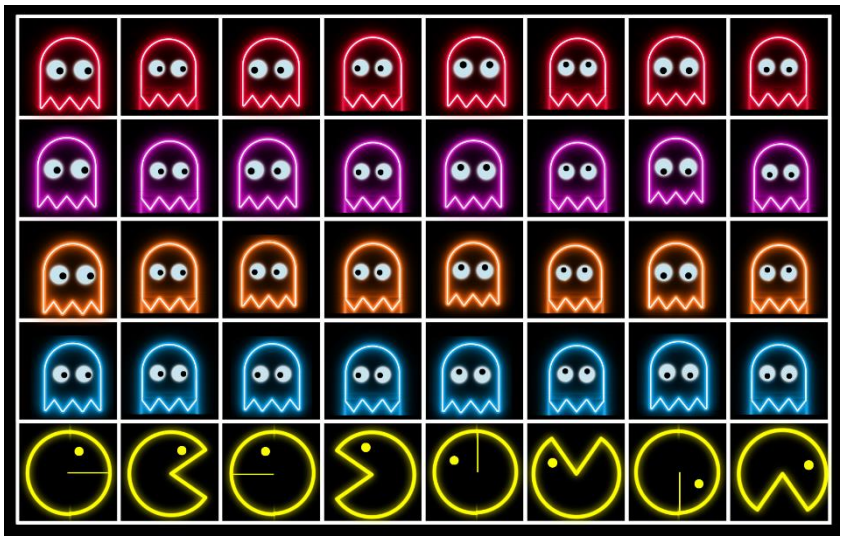
En resumen, el proyecto consistirá en desarrollar un juego de Pac-Man, utilizando los conocimientos adquiridos en la materia de POO, además de implementar otros conocimientos, que permitan la viabilidad del juego.

Los softwares utilizados son Canva para el diseño y tratamiento de imagen del juego, Java Swing para la interfaz de usuario, JSON para el control de los fantasmas, e Intellij IDEA para el código.

3.3 DISEÑO GRÁFICO E ILUSTRACIÓN

3.2.1 PERSONAJE

Los diferentes personajes fueron diseñados en la plataforma Canva, en los cuales se utilizaron iconos preestablecidos que ofrecía el mismo programa. La idea fue darles un concepto neón, con la finalidad que resalten en el fondo oscuro del programa.



3.2.2 PANTALLAS

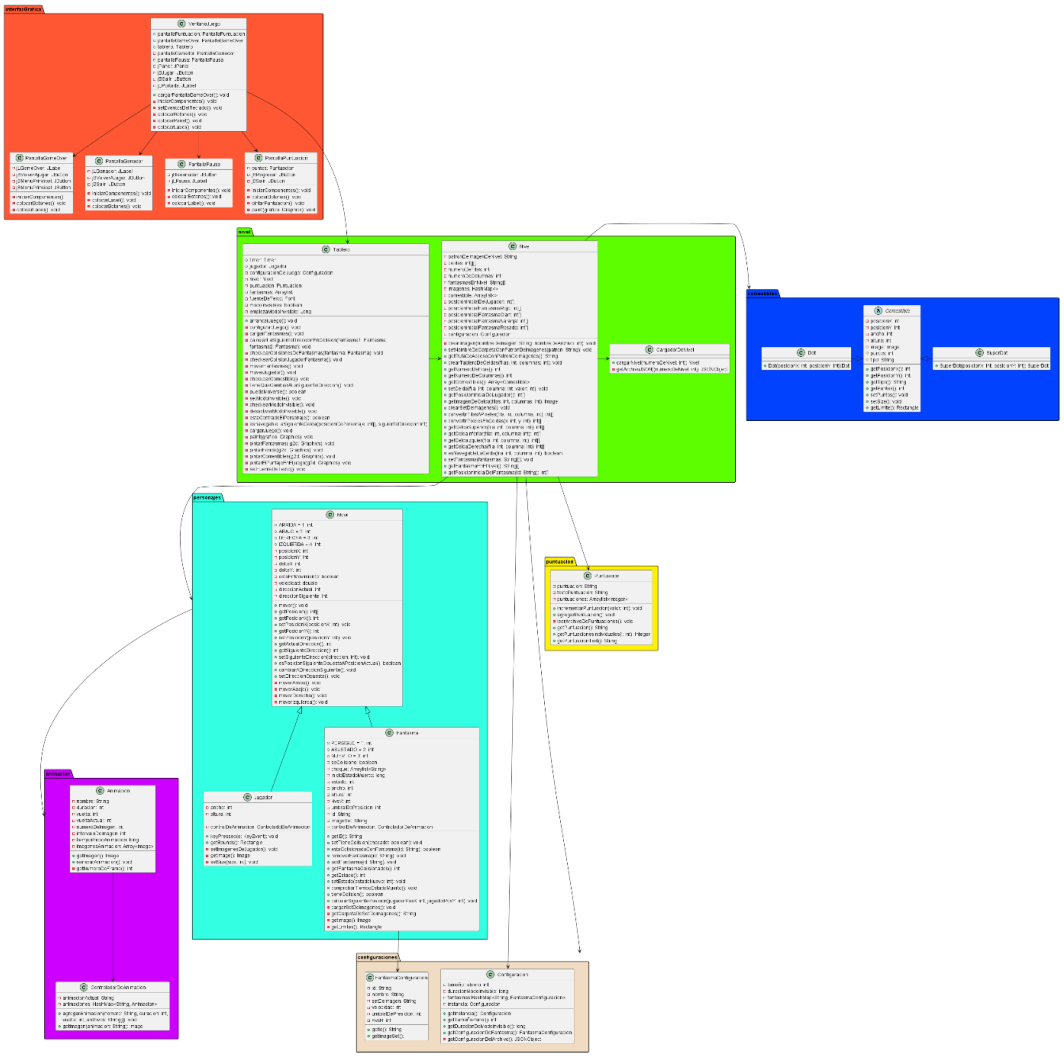
Las diferentes pantallas del juego fueron diseñadas en Canva utilizando imágenes preestablecidas que ofrece la biblioteca del mismo programa. La idea del proyecto en el aspecto visual fue diseñar todo con un toque neón para darle una vista más dinámica y llamativa.

3.3.3 BOTONES

El diseño de los botones fue diseñado en el mismo programa que fueron diseñadas las pantallas, Canva, su diseño fue inspirado en los colores del sello de la Escuela Politécnica Nacional y el animal característico de un Politécnico, el Búho, en base a estos elementos se diseñó cada uno de los botones utilizados en el proyecto.

3.3 DESARROLLO

3.3.1 UML



3.4.2 DESARROLLO EN DIGITAL

Fantasma

Esta clase maneja las instancias de los fantasmas y hereda de la clase Movimiento que gestiona la acción en general del movimiento. En esta clase se gestiona la inteligencia artificial de los fantasmas con base en los parámetros HvsV (Horizontal vs. Vertical) y accuracy_threshold (límite de eficacia).

El ancho y el alto son virtuales. No corresponden con el ancho y alto de la imagen del fantasma. De esta manera podemos permitir cierto solapamiento para dar la sensación de que efectivamente han tocado al jugador, ya que han invadido el espacio del mismo.

Gracias al sistema de configuración vía JSON y al algoritmo de búsqueda mediante los parámetros anteriormente citados, no es necesario crear una clase específica por cada fantasma.

```
90      public Fantasma(int x, int y, int size, String id, String imageset,
91                      int HvsV, int accuracy_threshold, int accuracyThreshold) {
92          super(x, y);
93          estado = PERSIGUE;
94          this.id = id;
95          this.imageset = imageset;
96          this.HvsV = HvsV;
97          this.accuracy_threshold = accuracy_threshold;
98          this.ancho = (int) (((long) size) * 0.95);
99          this.altura = (int) (((long) size) * 0.95);
100         this.seColisiono = false;
101         this.choque = new ArrayList<>();
102         loadImageSet();
103     }
```

Personaje

Esta clase tiene la finalidad de gestionar la animación, posición y movimiento del jugador. Se le transmite la pulsación del teclado desde la clase que la instancia.

El ancho y el alto son virtuales. No corresponden con el ancho y alto de la imagen del fantasma. De esta manera podemos permitir cierto solapamiento para dar la sensación de que efectivamente han tocado al jugador, ya que han invadido el espacio del mismo.

```
38      public Jugador(int x, int y, int size) {
39          super(x, y);
40          setSize(size);
41          animacionControl = new ControladorDeAnimacion();
42          setPlayerImages();
43      }
```

Movimiento

Clase principal que gestiona el movimiento y la posición de los objetos que hereden de ella. La lógica de si puede moverse en una dirección u otra, o la elección de la dirección se delega en otras clases.

```java
public Movimiento(int xInicial, int yInicial) {
    this.x = xInicial;
    this.y = yInicial;
    dy = 1;
    dx = 1;
    estaEnMovimiento = false;
    velocidad = 1.0;
    direccionActual = 0;
    direccionSiguiente = 0;
}
```

Comestible

Esta clase abstracta es la clase principal de los objetos recolectables. Establece los atributos y métodos mínimos para cada uno de ellos.

```java
public Comestible(int posicionX, int posicionY) {
    this.posicionX = posicionX;
    this.posicionY = posicionY;
}
```

Dot

Clase que se encarga de generar los dots "puntos" que come el pac man en el nivel y hereda de Comestible.

```java
public Dot(int posx, int posy) {
    super(posx, posy);
    ImageIcon imagen = new ImageIcon(this.getClass().getResource( name: "../resources/images/pickups/dot.png"));
    image = imagen.getImage();
    puntos = 10;
    tipo = "Dot";
}
```

SuperDot

Clase que se encarga de generar los súper dots que permite comer a los fantasmas y hereda de Comestible.

```java
public SuperDot(int x, int y) {
    super(x, y);
    ImageIcon imagen = new ImageIcon(this.getClass().getResource( name: "../resources/images/pickups/bigdot.png"));
    image = imagen.getImage();
    puntos = 15;
    tipo = "SuperDot";
}
```

Puntuación

Clase encargada de manejar la Puntuación del juego.

```
18      public Puntuacion() {
19          textoPuntuacion = "Puntuacion";
20          puntuacion = "0";
21          this.puntuaciones = new ArrayList<>();
22          agregarPuntuacion();
23          leerArchivoDePuntuaciones();
24      }
```

Nivel

Esta clase almacena la información referente al nivel:

- Estructura del nivel
- Los objetos recolectables
- Número de filas y columnas
- Los identificadores de los fantasmas disponibles en el nivel
- Posición inicial de cada fantasma y del jugador.
- El patrón de imágenes del nivel.

```
49      public Nivel() {
50          patronDeImagenDeNivel = "default";
51          imagenes = new HashMap<>();
52          comestibles = new ArrayList<>();
53          posicionInicialDelJugador = new int[2];
54          posicionInicialDelJugador[0] = 0;
55          posicionInicialDelJugador[1] = 0;
56          configuracion = Configuracion.getInstance();
57      }
```

CargadorDeNivel

Esta clase recoge la información de un fichero JSON, y lo carga en un objeto de tipo Nivel. La información que recoge del fichero es:

- La estructura del mapa
- Fantasmas disponibles en el mapa.

```
23      public CargadorDeNivel() {
24          tablero = new Nivel();
25      }
```

Tablero

Clase que lleva la logica principal del juego: render y actualizacion de los objetos. Obtiene la configuración principal de los fantasmas, tamaño de las celdas del objeto Config que accede a un fichero JSON.

```
38      public Tablero() {
39          // Cargamos las opciones del JPanel y añadimos el listener para los eventos de teclado
40          setFocusable(true);
41          setDoubleBuffered(true);
42          setBackground(Color.BLACK);
43          setFuenteDeTexto();
44          arrancarJuego();
45      }
```

Animación

Esta clase tiene por finalidad gestionar la animación.

```java
public Animacion(String nombreAnimacion, int duracionAnimacion, int vueltaAnimacion, String[] rutaImagenes) throws F
    nombre = nombreAnimacion;
    duracion = duracionAnimacion;
    vuelta = vueltaAnimacion;
    vueltaActual = 0;
    imagenesAnimation = new ArrayList<>();
    for (String i : rutaImagenes) {
        ImageIcon imagen = new ImageIcon(this.getClass().getResource(i));
        imagenesAnimation.add(imagen.getImage());
    }
    // Cálculo del número de imágenes y el intervalo de las mismas en la animación.
    numeroDeImagen = rutaImagenes.length;
    intervaloDeImagen = duracion / numeroDeImagen;
    // Inicialización del parámetro a 0 para saber que la animación
    // empieza desde el primer frame.
    tiempoInicioAnimation = 0;
}
```

ControladorDeAnimacion

Esta clase tiene el propósito de gestionar animaciones de los personajes del juego.

```java
public ControladorDeAnimacion() {
    animaciones = new HashMap<>();
}
```

Configuración

Esta clase guarda la configuración general de la aplicación. Guardamos:

- Tamaño en pixeles de las celdas
- Fantasmas disponibles y su configuración. Sé instancia siguiendo un patrón Singleton (Es un patrón de diseño creacional que garantiza que tan solo exista un objeto de su tipo y proporciona un único punto de acceso a él para cualquier otro código), ya que su información es sensible de ser utilizada en cualquier punto de la aplicación, y solo es necesario cargar su información 1 vez. Recoge la información de un fichero JSON situado en la carpeta:
- src/resources/config.json

```java
protected Configuracion() {
    instance = null;
    fantasmas = new HashMap<>();
    JSONObject obj = getConfigFile();
    Long size = (Long) obj.get("level_box_size");
    tamañoTablero = size.intValue();
    duracionModoInvicible = (Long) obj.get("invincible_mode_duration");
    JSONArray fantasmas = (JSONArray) obj.get("fantasmas");
    Iterator<JSONObject> iterator = fantasmas.iterator();
    while (iterator.hasNext()) {
        JSONObject config = iterator.next();
        String id = (String) config.get("id");
        String nombre = (String) config.get("nombre");
        String imageset = (String) config.get("imageset");
        Long HvsV = (Long) config.get("HvsV");
        Long velocidad = (Long) config.get("velocidad");
        Long accuracy = (Long) config.get("accuracy_threshold");
        FantasmaConfiguracion fantasma = new FantasmaConfiguracion(id, nombre, imageset,
                velocidad.intValue(), accuracy.intValue(), HvsV.intValue());
        this.fantasmas.put(id, fantasma);
    }
}
```

FantasmaConfiguracion

En esta clase almacenaremos la configuración de los fantasmas. Para evitar utilizar objetos Json cada vez que se requiere la información de un fantasma complicando la legibilidad y escritura del código, se ha optado por la creación de esta clase de ayuda. Como su finalidad es de solo lectura, solo se crean los "getters" de los diferentes atributos.

```java
public FantasmaConfiguracion(String id, String nombre, String imageset, int velocidad, int accuracy_threshold, int HvsV)
    this.id = id;
    this.nombre = nombre;
    this.imageset = imageset;
    this.velocidad = velocidad;
    this.accuracy_threshold = accuracy_threshold;
    this.HvsV = HvsV;
}
```

VentanaJuego

Esta es una clase que extiende de JFrame, es decir, es la clase que funciona como ventana del juego, también se encarga de administrar e instanciar todas las clases JPanel, en otras palabras, aquí se encuentran los métodos base para poder activar o visualizar a las otras pantallas (Paneles).

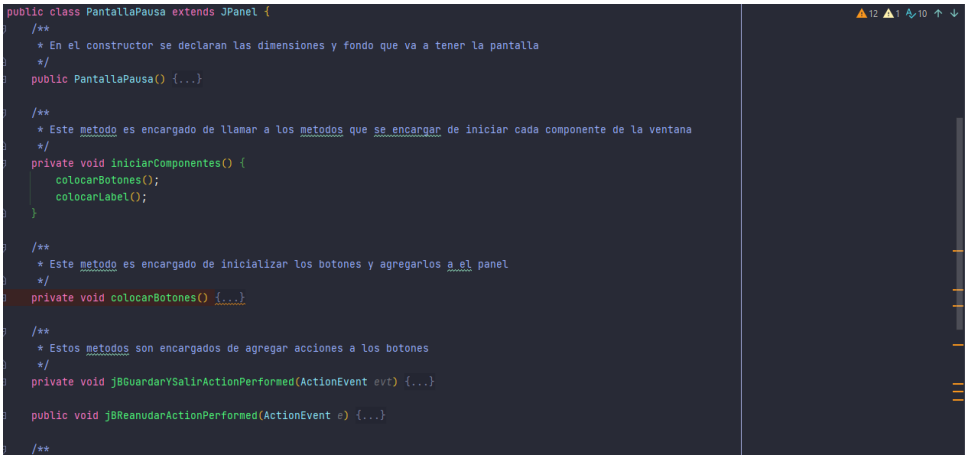
```java
*/

public class VentanaJuego extends JFrame {
    /**
     * En el constructor se declaran las dimensiones, titulo y ubicacion que va a tener la pantalla
     */
    public VentanaJuego() {
        setTitle("POLI PAC-MAN");
        setSize( width: 500,  height: 676);
        setLocationRelativeTo(null);
        iniciarComponentes();
        setMinimumSize(new Dimension( width: 570,  height: 676));
        setMaximumSize(new Dimension( width: 570,  height: 676));
        this.getContentPane().setBackground(Color.BLACK);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    /**
     * Este metodo es encargado de llamar a los metodos que se encargar de iniciar cada componente de la ventana
     */
    private void iniciarComponentes() {
        colocarPanel();
        colocarBotones();
        colocarLabel();
        eventosDelTeclado();
    }
```

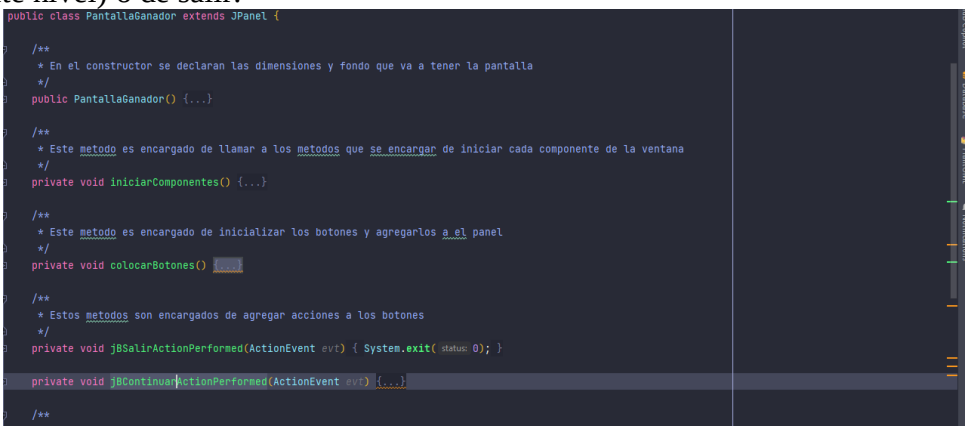
PantallaPausa

Es una clase tipo JPanel, la misma que se visualiza cuando un jugar pausa el juego mediante el teclado, en este caso con la tecla “ENTER”, aquí se encuentran métodos para agregar: Labels que nos servirán para colocar imágenes de fondo, y los botones para reanudar la partida o salir.



PantallaGanador

Es una clase tipo JPanel que se activa cuando el jugador termina con todos los comestibles del nivel, es la encargada de administrar los botones de continuar (continuar al siguiente nivel) o de salir.



PantallaGameOver

Esta clase extiende de JPanel, y aparece cuando un jugador es comido por un fantasma, por ende, cuando pierde, aquí se administra: un Label, que se utiliza para agregar un fondo de pantalla, y los Botones de “volver a jugar” (Instancia Un nuevo tablero), “puntuación” (activa la pantalla puntuación) y “menú principal” (Inicializa nuevamente la ventana del juego).

```java
public class PantallaGameOver extends JPanel {
    /**
     * En el constructor se declaran las dimensiones y fondo que va a tener la pantalla
     */
    public PantallaGameOver() {...}

    /**
     * Este metodo es encargado de llamar a los metodos que se encargar de iniciar cada componente de la ventana
     */
    private void iniciarComponentes() {...}

    /**
     * Este metodo es encargado de inicializar los botones y agregarlos a el panel
     */
    private void colocarBotones() {...}

    /**
     * Estos metodos son encargados de agregar acciones a los botones
     */
    private void jBPuntuacionActionPerformed(ActionEvent evt) {...}

    public static void jBMenuPrincipalActionPerformed(ActionEvent evt) {...}

    private void jBVolverAJugarActionPerformed(ActionEvent evt) {...}

    /**
```

PantallaPuntuacion

Esta es una clase JPanel la cual se activa cuando en la pantalla GameOver se presiona el botón “Puntuacion”, en la misma se lee el archivo encargado de guardar las puntuaciones y las muestra de manera organizada desde la puntuación más alta a la más baja.

```java
     * @param g2d Objeto Graphics2D
     */
    private void pintarPuntuacion(Graphics2D g2d) {
        Rectangle rect = getBounds();
        Color color = Color.WHITE;
        g2d.setColor(color);
        Tablero.pintarString(g2d, text: "Top" + "\nPuntuaciones", x: (rect.width / 2) - 270, y: 100);
        for (int i = 0; i < 5; i++) {
            Tablero.pintarString(g2d, text: String.valueOf( n: i + 1) + ".-    " + String.valueOf(puntos.getPuntuacionesIndividuales(i)
        }
    }

    /**
     * Este metodo es el encargado de configurar la manera en que se va a pintar la pantalla incluyendo
     * la fuente de texto
     *
     * @param grafico Objeto Graphics
     */
    public void paint(Graphics grafico) {
        super.paint(grafico);
        Graphics2D grafico2D = (Graphics2D) grafico;
        grafico2D.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING, RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
        grafico2D.setFont(Tablero.fuenteDeTexto.deriveFont((float) ((float) 100 * 0.50)));
        pintarPuntuacion(grafico2D);
        Toolkit.getDefaultToolkit().sync();
        grafico.dispose();
```

3.4 COMPLEJIDAD DEL CODIGO

Al momento de realizar el código del juego PacMan se nos presentó algunas complicaciones al momento de implementar ciertas funciones al juego que se presentaran a continuación:

- Al momento de que los fantasmas chocaban entre sí, estos se quedaban estancados y no perseguían al jugador en ninguna dirección.
- La escritura de la puntuación del juego en un archivo de texto, puesto que no podíamos hacer que se almacene de forma correcta la puntuación ya que se sobrescribían las mismas y no se mostraba la puntuación obtenida en la partida actual del juego, si no la puntuación de una partida anterior.
- Al verificar cuando un fantasma chocaba contra PacMan ya que el tamaño en pixeles no era el adecuado, por lo que tuvimos que meter un solapamiento de pixeles para que funcione correctamente.
- Al momento en que el PacMan comía los puntos el juego se detenía y no se ejecutaba correctamente, debido a un error con los pixeles.

4. CONCLUSIONES

- La importancia de las clases y los objetos en la programación ha quedado demostrada a lo largo del desarrollo de este proyecto. La correcta utilización de estos elementos permitió organizar de manera efectiva la estructura del programa, lo que se tradujo en un código más eficiente. El uso del UML como guía para el desarrollo del proyecto también fue fundamental, ya que permitió una representación clara y concisa de los objetos y sus relaciones en el código. En conclusión, la comprensión y aplicación adecuada de las clases, objetos y herramientas como el UML son esenciales para el éxito en proyectos de programación orientados a objetos.

- La implementación de Java Swing y JSON permitió crear una interfaz de usuario amigable e interactiva para el juego de Pac-Man. La investigación previa permitió seleccionar las herramientas adecuadas para desarrollar una interfaz gráfica, que mejoró la experiencia del usuario en el juego. Además, la implementación de estas herramientas mostró la capacidad de utilizar estas tecnologías y ampliar los conocimientos en el desarrollo de proyectos de programación.

5. BIBLIOGRAFIA

- (s.f.). Obtenido de Abstract Window Toolkit Overview: https://www.oreilly.com/openbook/javawt/book/ch01.pdf

- *Contapyme*. (2016). Obtenido de https://www.contapyme.com/api/ejemplo/JAVA.html#:~:text=El%20manejo%20de%20JSON%20en%20el%20lenguaje%20de%20JAVA%20se,JSON%20a%20String%20y%20viceversa.

- *FANDOM*. (s.f.). Obtenido de https://pacman.fandom.com/es/wiki/Pac-Man_(videojuego)

- *Jtech*. (s.f.). Obtenido de http://www.jtech.ua.es/historico/paj/restringido/apuntes/sesion09-apuntes.htm

- Marc Loy, R. E. (s.f.). *Google Libros.* Obtenido de https://books.google.es/books?hl=es&lr=&id=fU1K9MxaWp0C&oi=fnd&pg=PT4&dq=java+swing&ots=9RnVmeZKhG&sig=CZrPwLD_apPf9koObRS7q9rIzRM#v=onepage&q&f=false

- *PURE GAMING*. (2 de Octubre de 2017). Obtenido de https://puregaming.es/fantasmas-pacman-personalidad/

- *sc.ehu*. (s.f.). Obtenido de http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/threads/moviendo.htm

- wagnervladimir. (13 de Marzo de 2017). *Youtube*. Obtenido de https://www.youtube.com/watch?v=5LkpnrNO0fY

- Zukowski, J. (Marzo de 1997). *Java AWT Reference*. Obtenido de https://www.oreilly.com/openbook/javawt/book/