Manual Técnico – Torres de Hanoi

1. Descripción y Detalles de la Aplicación:

Para el desarrollo de la aplicación se hizo uso de las siguientes herramientas:

- Lenguaje de Programación: Java
- Versión de Java SE: 11
- IDE: Apache Netbeans 12.0
- Paradigma de Programación: Orientado a Objetos

2. Descripción de Clases:

Dado que el paradigma de programación utilizado fue el paradigma orientado a objetos, para el desarrollo de la aplicación se hizo uso de las siguientes clases:

- Main
- ventanaPrincipal
- torre

2.1. Clase Main:

Es la clase principal de la aplicación, desde la cual se iniciará la ejecución del programa. Esta clase contiene los siguientes atributos:

- **ventana:** Atributo de tipo "ventanaPrincipal()", el cual sirve para generar un objeto de tipo ventanaPrincipal.

2.2. Clase ventanaPrincipal:

Es la clase que contiene tanto el diseño de la interfaz gráfica de la aplicación como los aspectos que permiten a la aplicación ser dinámica. Esta clase contiene los siguientes atributos:

- **tr:** Atributo de tipo "torre", el cual nos sirve para generar un objeto de tipo torre, el cual nos servirá para poder interactuar con las propiedades y métodos de una torre.
- **movimientos:** Atributo de tipo "int" (entero), el cual nos sirve para poder llevar la cuenta de movimientos realizados por el usuario.
- **movMax:** Atributo que nos sirve para estableces el límite de movimientos que el usuario podrá realizar. El límite de movimientos depende de la cantidad de discos que el usuario utilice para jugar.
- **bandera:** Atributo de tipo "boolean" (booleano), el cual nos sirve para identificar si el juego ha terminado o no. Si el juego aún no termina, bandera debe adoptar un valor verdadero, de lo contrario, bandera debe adoptar un valor falso.

Esa clase contiene los siguientes métodos:

- **unoAdos:** método que nos permite desplazar un disco de la torre 1 a la torre dos.
- **unoAtres:** método que nos permite desplazar un disco de la torre uno a la torre tres.
- **dosAuno:** método que nos permite desplazar un disco de la torre dos a la torre uno.
- **dosAtres:** método que nos permite desplazar un disco de la torre dos a la torre tres.
- **tresAuno:** método que nos permite desplazar un disco de la torre tres a la torre uno.
- **tresAdos:** método que nos permite desplazar un disco de la torre tres a la torre dos.
- **Iniciar(String):** método que nos permite dar inicio al juego. Este método recibe un parámetro de tipo "String" (cadena de caracteres), el cuál representa la cantidad de discos que se utilizarán en el juego.
- **centrar:** método que nos permite centrar los discos de cada torre en su respectivo espacio.
- **verificarGanador:** clase que nos permite identificar si el usuario ha completado el juego con éxito, si no lo ha completado o si no logró completar el juego con éxito.

2.3. Clase torre:

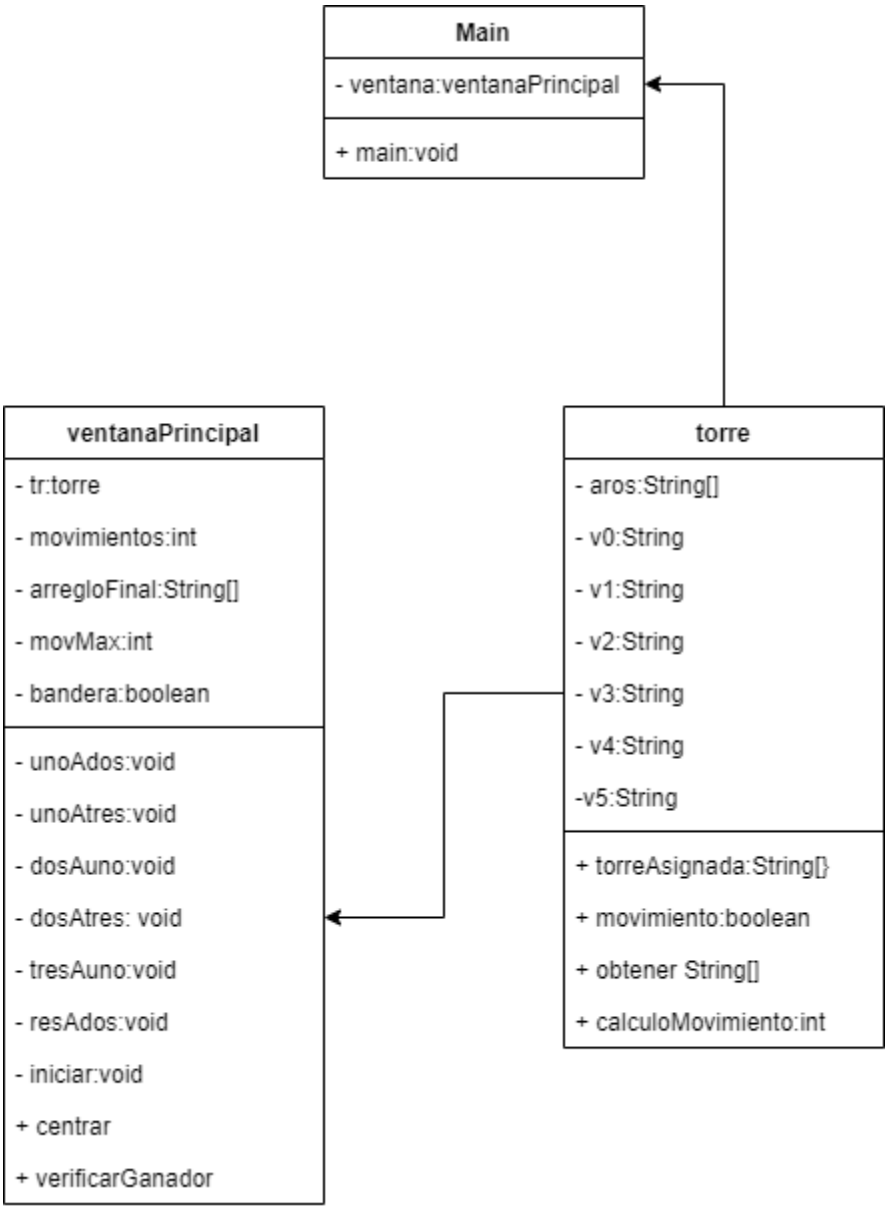
Es la clase que nos permite identificar el comportamiento y propiedades que un torre debe tener. Esta clase contiene los siguientes atributos:

- **v0:** atributo de tipo "String" (cadena de caracteres), el cual dibuja un espacio en el que no se encuentra ningún disco de la torre de hanoi.
- **v1:** atributo de tipo "String" (cadena de caracteres), el cual dibuja un disco de longitud 1 (el disco más pequeño).
- **v2:** atributo de tipo "String" (cadena de caracteres), el cual dibuja un disco de longitud 2.
- **v3:** atributo de tipo "String" (cadena de caracteres), el cual dibuja un disco de longitud 3.
- **v4:** atributo de tipo "String" (cadena de caracteres), el cual dibuja un disco de longitud 4.
- **v5:** atributo de tipo "String" (cadena de caracteres), el cual dibuja un disco de longitud 5 (el disco más grande).

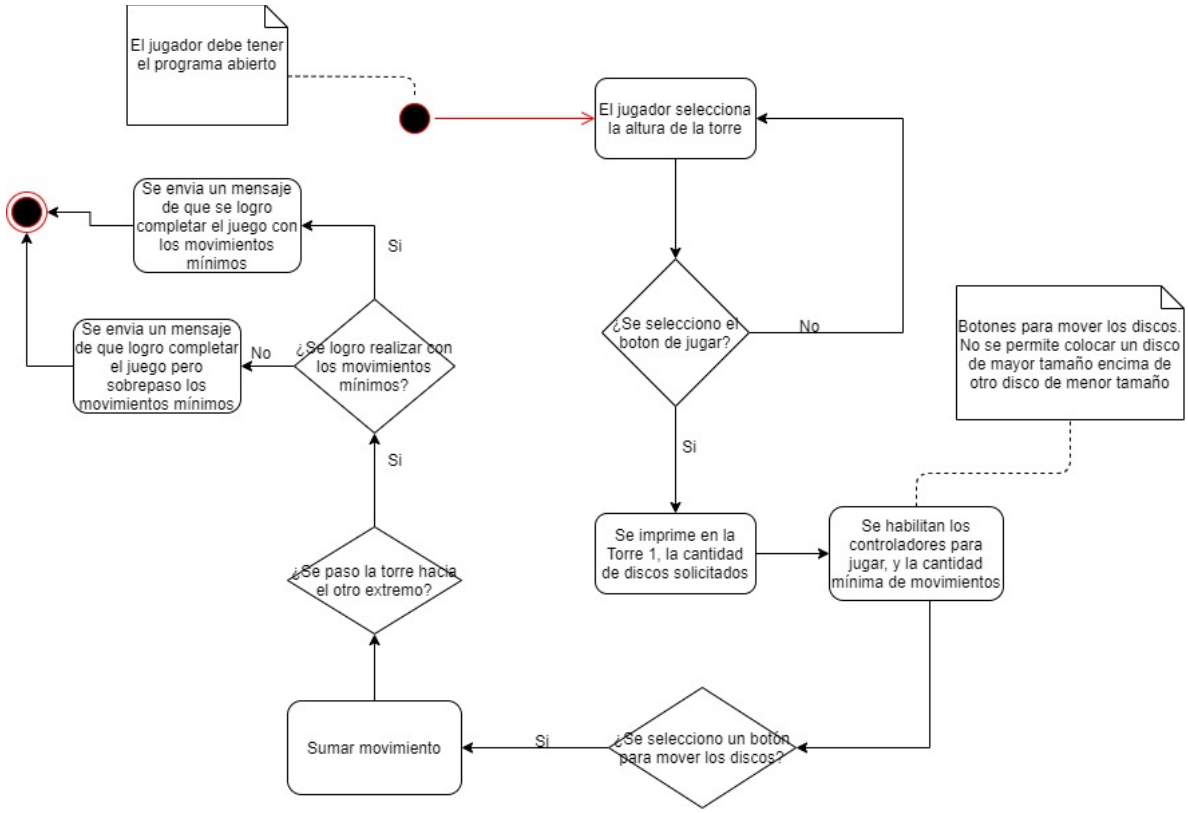
Esta clase contiene los siguientes métodos:

- **torreAsignada:** método que devuelve una torre con la cantidad de discos especificados por el usuario. Esta torre es la torre con la que se inicia el juego.
- **movimiento:** método de tipo "boolean" (booleano), el cual nos sirve para identificar si un movimiento es válido o no. El método devuelve un valor verdadero si un disco es colocado sobre otro más grande o si es colocado en un espacio donde no hay más discos. El método devuelve un calor falso si un disco es colocado sobre otro más pequeño.
- **obtener:** método que devuelve una torre modificada, es decir, cualquier torre omitiendo la torre inicial.
- **calculoMovimientos:** método de tipo "int" (entero), el cuál devuelve la cantidad de movimientos en que los discos podrán desplazarse. La cantidad de movimientos depende de la relación de recursividad que establece la cantidad de discos que el usuario estableció.

3. Diagrama de Clases:



3.2 Diagrama de Actividad:



3.3 Formula de Recurrencia Utilizada:

$$a_n = 2^n - 1$$