Laboratorio de Introducción a la programación y computación 1, Sección D.

PRACTICA 2 – TORRES DE HANOI

MANUAL TECNICO

FECHA: 07/10/2021

Juan Josue Zuleta Beb

Carné: 202006353

Introducción

El presente documento describe los aspectos técnicos informáticos del Juego torres de Hanoi deseñado a través de la interfaz gráfica de Jaca Swing. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

Objetivos

Instruir el uso adecuado del de la instalación y comprensión del código y de la implementación de métodos, para el acceso oportuno y adecuado en la inicialización de este, mostrando los pasos a seguir en el proceso de inicialización, así como la descripción de los archivos relevantes del sistema los cuales nos orienten en la configuración.

Requisitos del Sistema

Sistema operativo 64 bits

- Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit)
- macOS 10.5 o superior
- Linux GNOME o KDE desktop

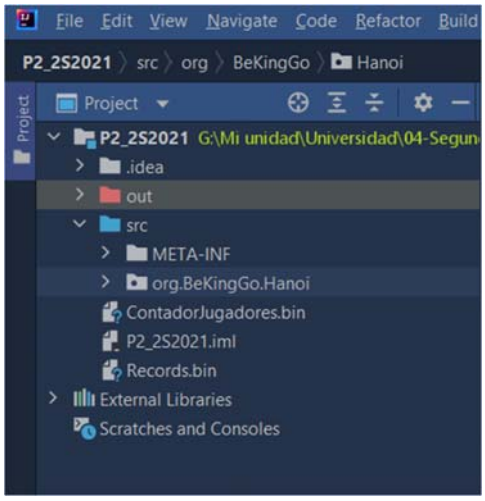
RAM: 1gb como mínimo.
Disco duro: 300mb como mínimo + 1gb para cache.
IDE: Intellij Idea
JDK: Versión 16.0.2
Resolución de pantalla: 1024×768 resolución mínima de pantalla

Paradigma de programación

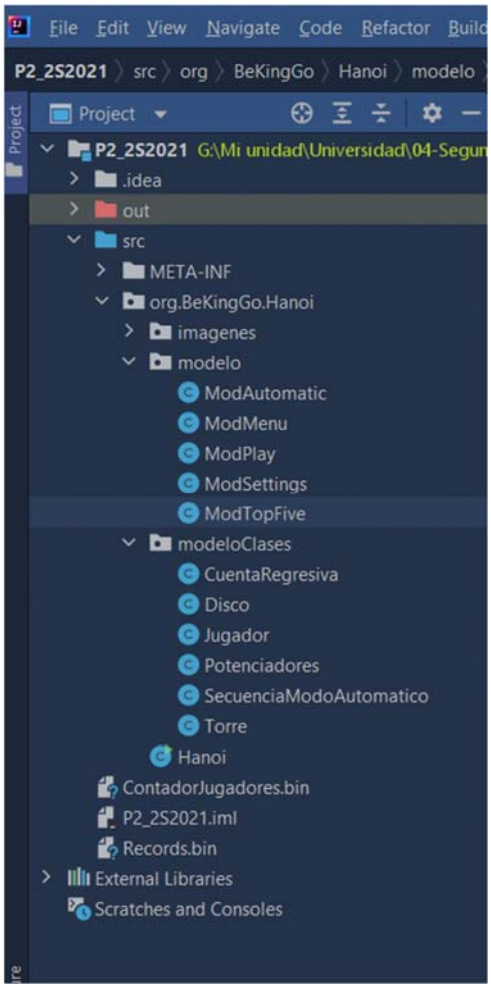
Programación orientada a objetos (Interfaces gráficas, Hilos y Recursividad)

Estructura raíz:

El proyecto tiene la siguiente estructura de directorios:



Directorio de la app:



La clase Hanoi:

Contiene el método principal de nuestra aplicación y controlara todos los métodos y la implementación de la interfaz gráfica.

El código fuente es el siguiente:

```
public static void main(String[] args) {

    java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            try {
                File validarContador = new File( pathname: "ContadorJugadores.bin");
                File validarRecords = new File( pathname: "Records.bin");

                if (validarContador.exists() && validarRecords.exists()) {

                    Hanoi.contadorJugadores = (int) Hanoi.desSerializacion( path: "contadorJugadores.bin");
                    Hanoi.Records = (Jugador[]) Hanoi.desSerializacion( path: "Records.bin");

                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Bienvenido a Torres de Hanoi - FIUSAC");
                    new ModMenu();

                } else {

                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Bienvenido a Torres de Hanoi - FIUSAC");
                    new ModMenu();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

        }
    });
```

Modelo:

Contiene las clases que representan la interfaz gráfica y la modulación de hilos y la interacción con cada una de las clases.

ModeloClases:

Contiene las clases que representan el paradigma de la programación orientada a objetos y contiene los atributos y métodos de cada una de las funcionalidades de nuestra aplicación y su interacción con cada una de las clases.

ModAutomatic:

Es la clase del modo de juego automático que contiene el código fuente recursivo para que la maquina virtual pueda resolver el algoritmo de forma automática.

El código fuente es el siguiente:

```java
//----------ALGORITMO DE RECURSIVIDAD PARA TORRES DE HANOI----------

public void TorresHanoi(int Discos, int TorreA, int TorreB, int TorreC) throws IOException, InterruptedException {

    //CASO BASE
    if   (Discos == 1) {

        ModAutomatic.txtAreaI.append("       Mover Disco #"+ Discos + "    de la Torre #"+TorreA+ "    haca la Torre #"+TorreC+"\n"+"\n");
        contador++;
        ModAutomatic.lblt5.setText(String.valueOf(contador));
        Thread.sleep( millis: 1500);

    } else {

        //CASO DOMINIO
        TorresHanoi( Discos: Discos - 1, TorreA, TorreC, TorreB);
        ModAutomatic.txtAreaI.append("       Mover Disco #"+ Discos + "    de la Torre #"+TorreA+ "    haca la Torre #"+TorreC+"\n"+"\n");
        contador++;
        ModAutomatic.lblt5.setText(String.valueOf(contador));

        Thread.sleep( millis: 1500);
        TorresHanoi( Discos: Discos - 1, TorreB, TorreA, TorreC);

    }
  }
}
```

CuentaRegresiva:

Es la clase del modo de juego principal que contiene el código fuente implementando hilos para poder crear la cuenta regresiva que hará que el tiempo de partida se disminuya constantemente.

El código fuente es el siguiente:

```java
import java.io.IOException;

public class CuentaRegresiva extends Thread{

    @Override
    public void run(){
           try {
               while ( ModPlay.contTiempoPartida != 0){
                   sleep( millis: 1000);
                   ModPlay.contTiempoPartida--;
                   ModPlay.lblt2.setText(String.valueOf(ModPlay.contTiempoPartida));

                   //VERIFICA SI EL JUGADOR SE HA QUEDADO SIN TIEMPO
                   if (ModPlay.contTiempoPartida == 0){

                       ModPlay.moduloDeJuego.setVisible(false);

                       JOptionPane.showMessageDialog( parentComponent: null, message: "Te has quedado sin tiempo, Has Perdido ! \n " +
                               "Intenta configurar una mayor cantidad de tiempo! :) ");

                       ModPlay.contTiempoPartida = 120;
                       new ModMenu();
                       ModPlay.contReversa.stop();
                       Potenciadores.hilo.stop();

                   }
               }
           } catch (InterruptedException | IOException e) {
               e.printStackTrace();
           }
    }
}
```

Potenciadores:

Es la clase del modo de juego principal que contiene el código fuente implementando hilos para poder crear los potenciadores los cuales tendrán la característica de poder modificar el tiempo de partida en tiempo real lo cual significa que pueden disminuir o aumentar el tiempo de partida durante la ejecución del juego.

El código fuente es el siguiente:

```
@Override
public void run() {
    try {
        while (true) {
            while (movEnY < 345) {

                Thread.sleep( millis: 1000);
                Decrementar.setVisible(false);
                Incrementar.setVisible(true);
                movEnY += 20;
                //Incrementar.setIcon(aumetar);
                Incrementar.setBounds( x: 50, movEnY,  width: 35,  height: 35);
                Incrementar.setBackground(aux2);

                Thread.sleep( millis: 1000);
                Incrementar.setVisible(false);
                Decrementar.setVisible(true);
                movEnY += 20;
                //Incrementar.setIcon(disminuir);
                Decrementar.setBounds( x: 50, movEnY,  width: 35,  height: 35);
                Decrementar.setBackground(aux1);

                if (movEnY >= 275) {
                    movEnY = 0;
                }
            }
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```