explicación del codigo

class MainActivity : AppCompatActivity() {: Esta es la declaración de la clase MainActivity que hereda de AppCompatActivity. AppCompatActivity es una clase de compatibilidad para actividades en Android.

val fd by lazy{...}: Aquí se está creando una propiedad delegada que se inicializa de manera perezosa (lazy). Esto significa que el valor se calcula y se asigna la primera vez que se accede a la propiedad. En este caso, se está abriendo un archivo de música.

val mp by lazy {...}: Similar a la anterior, esta es otra propiedad delegada que se inicializa de manera perezosa. Aquí se está creando un objeto MediaPlayer y se está configurando su fuente de datos para que sea el archivo de música que se abrió anteriormente.

val controllers by lazy{...}: Esta propiedad delegada contiene una lista de los botones de control de la aplicación (anterior, detener, reproducir, siguiente).

object ci {...}: Este es un objeto singleton que contiene las constantes para los índices de los botones de control en la lista de controladores.

val Cancion by lazy{...}: Esta propiedad delegada contiene una referencia al TextView que muestra el nombre de la canción actual.

val canciones by lazy{...}: Esta propiedad delegada contiene una lista de todas las canciones (archivos .mp3) en los assets de la aplicación.

var cancionActualIndex = 0 {...}: Esta es una propiedad mutable que mantiene el índice de la canción actual. Cuando se establece un nuevo valor, también se actualiza la canción actual.

lateinit var cancionActual:String: Esta es una propiedad que se inicializará más tarde con el nombre de la canción actual.

override fun onCreate(savedInstanceState: Bundle?) {...}: Este es el método onCreate que se llama cuando se crea la actividad. Aquí se establece el contenido de la vista, se configuran los oyentes de clics para los botones de control y se establece la canción actual.

fun playClicked(){...}: Este método se llama cuando se hace clic en el botón de reproducción. Comienza o pausa la reproducción de la música.

fun stopClicked(){...}: Este método se llama cuando se hace clic en el botón de detener. Detiene la reproducción de la música y restablece el reproductor a su estado inicial.

fun sigClick(){...} y fun antClick(){...}: Estos métodos se llaman cuando se hacen clic en los botones siguiente y anterior, respectivamente. Cambian la canción actual al siguiente o al anterior en la lista.

fun refrescarmusica(){...}: Este método se llama para actualizar la música que se está reproduciendo en el reproductor. Carga la canción actual en el reproductor y comienza a reproducirla.

Explicación del código de la marquesina.

Este es un archivo de recursos XML en Android que define un estilo personalizado llamado "Marqueetext". Los estilos en Android son una forma de aplicar un conjunto de atributos estéticos a múltiples vistas sin tener que definirlos individualmente para cada vista. En este caso, el estilo "Marqueetext" se ha definido para crear un efecto de desplazamiento de texto, similar a un letrero de marquesina.

Transcribo el significado de cada línea:

<style name="Marqueetext">: Define un nuevo estilo llamado "Marqueetext".

<item name="android:singleLine">true</item>: Este atributo asegura que el texto se muestre en una sola línea.

<item name="android:ellipsize">marquee</item>: Este atributo indica que si el texto es más largo que la vista, se recortará y se mostrará como un letrero de marquesina (es decir, se desplazará horizontalmente).

<item name="android:marqueeRepeatLimit">marquee_forever</item>: Este atributo define cuántas veces se repetirá el efecto de marquesina. En este caso, se repetirá indefinidamente.

<item name="android:focusable">true</item> y <item name="android:focusableInTouchMode">true</item>:
Estos atributos aseguran que la vista pueda obtener el foco, lo cual es necesario para que el efecto de marquesina funcione.

<item name="android:scrollHorizontally">true</item>: Este atributo permite que la vista se desplace horizontalmente.

</style>: Cierra la etiqueta del estilo.

</resources>: Cierra la etiqueta de los recursos.

Para aplicar este estilo a una vista, puedes hacerlo en tu archivo XML de la siguiente manera:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/Marqueetext"
    android:text="Texto largo que se desplaza como una marquesina..."
/>
```

En este caso, el texto dentro del TextView se desplazará horizontalmente como una marquesina si es más largo que la anchura de la vista.

```
class MainActivity : AppCompatActivity() {
    val fd by lazy{
        //assets.openFd("la sirena.mp3")
        assets.openFd(cancionActual)
    }

    val mp by lazy {
        val m = MediaPlayer()
        m.setDataSource(
            fd.fileDescriptor,
            fd.startOffset,
            fd.length
        )
        fd.close()
        m.prepare()
        m
    }

    val controllers by lazy{
        listOf(R.id.btnAnt, R.id.btnStop, R.id.btnPlay,
R.id.btnSig).map{findViewById<MaterialButton>(it)}
    }
 // se crea un objeto CI para que en caso que se necesite cambiar de pos ya
este listo

    object  ci{
        val ant  = 0
        val stop = 1
        val play = 2
        val sig  = 3
    }

    val Cancion by lazy{
        findViewById<TextView>(R.id.NombCancion)
    }

    // se crea una opcion / lazy
    val canciones  by lazy{
      val nombArchivos = assets.list("")?.toList()?: listOf()
       nombArchivos.filter { it.contains(".mp3") }
    }

    var cancionActualIndex = 0
        set(value){
            var v = if(value ==-1){
                canciones.size-1
            }
            else {
                value % canciones.size
            }
            field = v
            cancionActual = canciones[v]
        }

    lateinit var cancionActual:String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        controllers[ci.play].setOnClickListener { playClicked() }
        controllers[ci.stop].setOnClickListener { stopClicked() }
        controllers[ci.ant].setOnClickListener { antClick() }
        controllers[ci.sig].setOnClickListener {sigClick() }

       // Cancion.text = "la sirena.mp3"
        cancionActual = canciones[cancionActualIndex]
        Cancion.text = cancionActual
    }
```

```
    fun playClicked(){
        if (!mp.isPlaying){
            mp.start()
            controllers[ci.play].setIconResource(R.drawable.baseline_pause_48)
            Cancion.visibility = View.VISIBLE
        }
        else {
            mp.pause()

controllers[ci.play].setIconResource(R.drawable.baseline_play_arrow_48)
        }
    }
    fun stopClicked()
    {
        if(mp.isPlaying){
            mp.pause()

controllers[ci.play].setIconResource(R.drawable.baseline_play_arrow_48)
            Cancion.visibility = View.INVISIBLE
        }
        mp.seekTo(0)
    }

 // boton siguiente  y boton previo

    fun  sigClick()
    {
        cancionActualIndex++
        refrescarmusica()
    }

    fun  antClick()
    {
        cancionActualIndex--
        refrescarmusica()
    }

    fun refrescarmusica(){
        mp.reset()
        val fd = assets.openFd(cancionActual)
        mp.setDataSource(
            fd.fileDescriptor,
            fd.startOffset,
            fd.length
        )
        //fd.close()
        mp.prepare()
        playClicked()
        Cancion.text = cancionActual
    }
}
```