INDICE

1.1. Introducción

- Un control **Chronometer** implementa en Android un cronómetro simple.
  - ✓ Comienza con el método start().
  - ✓ Se para con el método stop(). Pero ... sigue contando desde atrás. Es decir, si ejecuta start () nuevamente sin parámetros, configurará el temporizador al mismo tiempo que si no se detuviera.
- Cuando se llama a start(), el control toma el tiempo al que ha accedido el móvil (elapsedRealtime()) como tiempo base y cuenta desde allí.

  Si no se le da un tiempo base, toma el tiempo en el que se llama a start ().

- El método setBase () se usa para establecer un tiempo base.
- De forma predeterminada, muestra la hora en formato "MM:SS". El método setFormat (String) se puede utilizar para cambiar el formato.
- La clase **Chronometer** hereda directamente da clase **View**.
- Para establecer los valores de tiempo base, confiamos en la clase SystemClock.
- **Referencias:**
  - ✓ Clase **Chronometer**: https://developer.android.com/reference/android/widget/Chronometer.html
  - ✓ Clase **SystemClock**: https://developer.android.com/reference/android/os/SystemClock.html

1.2. Caso práctico

- Partimos que ya tenemos creado el proyecto inicial.

  Si no lo tenemos creado antes, crear un paquete de nome **UI** como un subpaquete de tu paquete principal.

  Dentro do paquete UI, crearemos un nuevo paquete de nombre: **Chonometers**.

- Dentro del paquete **Chonometers** crear una nueva 'Empty Activity' de nombre: **UD02_01_ Chonometers** de tipo Launcher.

Modificar el archivo **AndroidManifiest.xml** y añade un label a la activity como ya vimos en la creación del proyecto base.

Al iniciar la aplicación el crono está parado. Iniciémoslo.

El crono está en funcionamiento y si no se para se autodestruirá la aplicación en X seg. El valor de partida para la autodestrucción se establece una variable en el código.

Se paró el crono. Si se pulsa en *Start*, es como comenzar de nuevo.

1.2.1. XML do Layout

- Observar cómo se crea el control Chronometer y los métodos a los que llaman los botones.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tvAutodestr_UD02_01_Chonometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Autodestrucción en: ?? seg" />

    <Chronometer
        android:id="@+id/chrCronometro_UD02_01_Chonometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textSize="20sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btStart_UD02_01_Chonometer"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Start" />
```

```
            <Button
                android:id="@+id/btStop_UD02_01_Chonometer"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Stop" />
        </LinearLayout>


    </LinearLayout>
```

1.2.2. Código Java

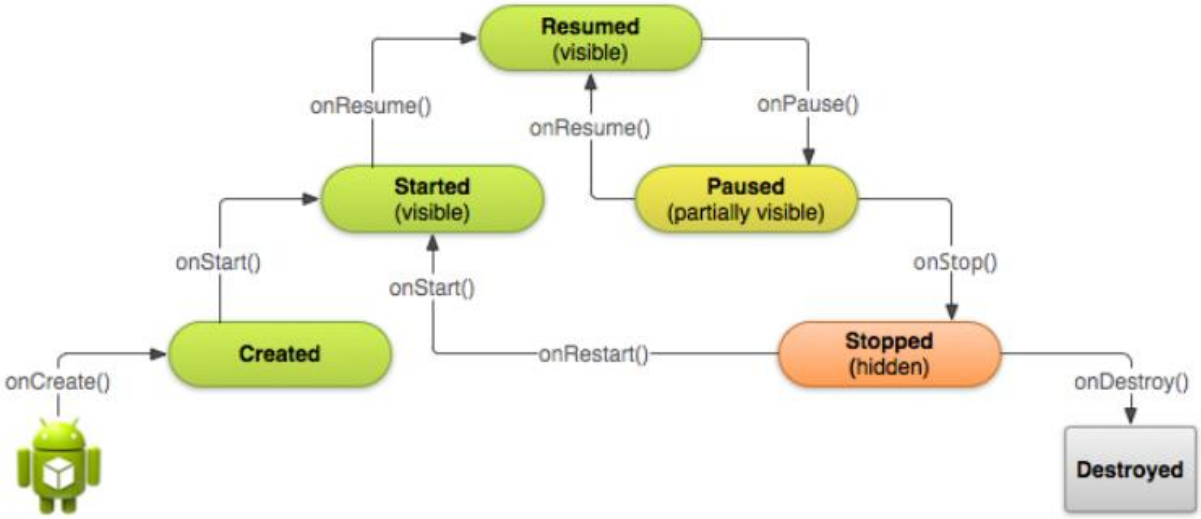
```
package com.example.ud02_01_chronometer;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.SystemClock;
import android.view.View;
import android.widget.Button;
import android.widget.Chronometer;
import android.widget.TextView;

public class UD02_01_Chronometer extends AppCompatActivity {

    private int tiempoAutodestruccion; //No podemos declararla final ya que se modifica su va-
lor. Al tener una referencia a ella de clase anónima
                                       // OnChronometerTickListener, tenemos que declararlo a
nivel de clase
    private Chronometer crono;

    private void gestionarEventos(){
        final TextView tvAutodestruccion;

        //Eventos de crono
        crono = (Chronometer) findViewById(R.id.chrCronometro_UD02_01_Chonometer);
        tvAutodestruccion = (TextView) findViewById(R.id.tvAutodestr_UD02_01_Chonometer); // Lo
ponemos aquí ya que el método onChronometerTick se ejecuta muchas veces
        crono.setOnChronometerTickListener(new Chronometer.OnChronometerTickListener() {
            @Override
            public void onChronometerTick(Chronometer chronometer) {


                long tiempoPasado = SystemClock.elapsedRealtime() - chronometer.getBase();

                int tiempoSeg = (int) tiempoPasado / 1000;
                if (tiempoSeg == tiempoAutodestruccion)
                    finish();

                tvAutodestruccion.setText("Autodestrucción en: " + (tiempoAutodestruccion -
tiempoSeg) + " seg ");

            }
        });

    //Eventos botón start
    Button btStart = findViewById(R.id.btStart_UD02_01_Chonometer);
    btStart.setOnClickListener(new View.OnClickListener(){
```

```
        @Override
         public void onClick (View v){
            tiempoAutodestruccion = 20;
            crono.setBase(SystemClock.elapsedRealtime());
            crono.start();
        }
        });

    //Eventos botón stop (no tenemos por que declarar una variable)
        ((Button) findViewById(R.id.btStop_UD02_01_Chonometer)).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick (View v){
                crono.stop();
            }
        });
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        gestionarEventos();
    }
}
```

- **Líneas 40-49:** Gestionamos el evento Click en el botón start:
  - ✓ **Línea 45:** Definimos el tiempo para la autodestrucción de la Activity. Cuando veamos los menús, veremos que este valor podría ser establecido por el usuario que usa la aplicación, como una opción.
  - ✓ **Línea 46:** Se establece el tiempo base a partir del cual comienza a contar el cronómetro. SystemClock.elapsedRealtime (): Devuelve el tiempo, en milisegundos, que el teléfono ha estado encendido.
  - ✓ **Línea 47:** el cronómetro empieza a contar, no desde cero, sino desde el tiempo que se toma como base (el tiempo que se tarda en acceder al móvil).
- **Líneas 51-57:** Gestionamos el evento Click en el botón de parada:
  - ✓ Línea 55: Detiene el cronómetro.
- **Líneas 21-38:** Gestionamos el Listener del cronometro que se llama cada vez que hay un cambio en su valor:
  - ✓ **Línea 24:** Se establece el Listener del cronometro. Este método se llamará cada vez que el cronómetro cambie de valor.
  - ✓ **Líneas 29 y 30:** El tiempo transcurrido desde que se puso en marcha el cronómetro es la diferencia entre el tiempo que lleva acceso al dispositivo y el tiempo en el cual se inició el cronómetro.
  - ✓ **Línea 31:** Tiempo en segundos.
  - ✓ **Línea 33:** Si llegamos al momento de la autodestrucción debemos matar a la Activity. Sería lo mismo que si pulsamos el botón "Retroceso o Back".

1.3. Ciclo de vida de una actividad

- Profundizaremos más en el ciclo de vida de una actividad.
- Una actividad desde su lanzamiento pasa por muchos estados.
- Cuando una actividad no está en la primera línea de la pantalla, no se destruye sino que se encuentra en un estado de parada esperando en la pila para ser llamada, o si está demasiado baja en la pila y se necesitan sus recursos, el sistema puede destruirla.



- En nuestro ejemplo, la actividad se destruye explícitamente si el temporizador alcanza un cierto tiempo
- El método **finish ()** se utiliza para esto.
- Ver la línea 33 del código.