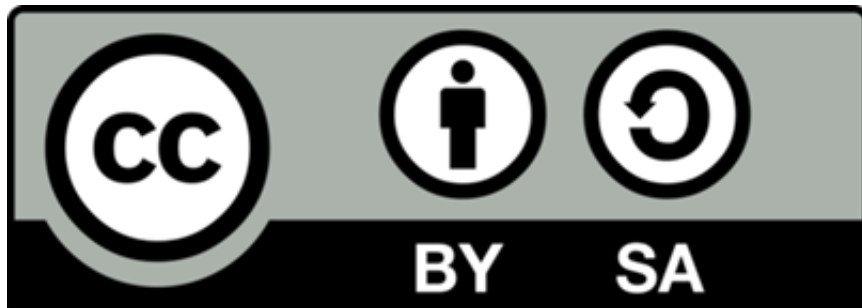


Desarrollo de aplicaciones Android (161811FP07)

Marzo-Abril de 2016. CEP de Granada

José Antonio Vacas @javacasm

javacasm@gmail.com



Comenzando a programar

Estructura del código

Intenta que sea lo más ligera posible

Clases

- Es programación orientada a objetos
- Tipo
- Herencia
- Estilo de java
 - Nombres
 - set y get

Eventos

- Con programación dirigida por eventos
- Listener
- Ciclo de vida de una aplicación

Acceso a los controles

- (Casting)
- Comentarios
- Ejemplo persona en la calle, entra al CEP, entra a la oficina. Le pido cosas(funciones) distintas

Activación depuración

- Info tablet
- nº compilación

Excepciones, globales, casting

Pseudocódigo de la Calculadora

```
introducimos operando1 -> teclanumerica
    esperamos teclas numericas
    añadimos a las cifras anteriores
introduce operador -> teclaOperador
    guardar operando1 fOperando1=convertir(visor)
    guardar operador iOperador=R.id.boton
    borramos pantalla -> teclaC
introducimos operando2 -> teclanumerica
    esperamos teclas numericas
    añadimos a las cifras anteriores
pulamos igual -> teclaIgual
    guardar operando2
    hacer la operacion
    mostrar resultado
borramos último numero -> teclaC
    borramos numero en el visor
reiniciamos -> teclaAC
    inicializar
        borrar operandos y operador -> Inicializacion
        borrar visor ->teclaC
```

Programación 2

Vamos a ver algunas técnicas más avanzadas de programación, como pueden ser crear arrays de id o de atributos.

Para ello vamos a hacer una programación TopDown, es decir, empezaremos creando la estructura del código a partir del pseudocódigo.

Para ello creamos las funciones:

- inicializacion
- clickImagen
- botonStart
- botonStop

Añadiremos a las funciones pseudocódigo para indicar las tareas que se deben realizar en cada uno de ellos, utilizando comentarios de tipo TODO a los que podremos seguir la pista en la ventana TODO del entorno

```
void inicializacion()
{
    // TODO: rellenar aleatoriamente (rellenar imagenBoton )
    // TODO: cargar sonidos
    // TODO: puntuacion a 0
    // TODO: tiempo a 0
}

public void botonStart(View v)
{
    inicializacion(); // Ya sabemos cómo hacerlo

    // TODO: hacer visible Las cartas (tablelayout.setVisibility(Visible) );
```

```

}
public void clickImagen(View v)
{
    // TODO: ¿Y si el boton ya esta pulsado?
    // TODO: Buscar boton para ver su imagen
    // TODO: Mostrar imagen (iv.setImageResource(R.drawable.IMAGEN); )

    // TODO: ¿EsLaPrimera?
    // TODO:     SI Volteamos
    // TODO:     NO
    // TODO: Si imágenes iguales
    // TODO:     actualizar visor puntuacion
    // TODO:     sonido victoria
    // TODO:     No cliceable la 2ª
    // TODO: No son iguales
    // TODO:     sonido fracaso total
    // TODO:     Volteamos

}

```

Estructura de datos

Ahora vamos a ir declarando las variables necesarias para realizar las tareas que hemos indicado en las funciones

TODO: ¿usar Tag?

```

int []imagenBoton={R.drawable.conejo,R.drawable.oveja,R.drawable.pollo,
    R.drawable.conejo,R.drawable.oveja,R.drawable.pollo,
    R.drawable.rinoceronte,R.drawable.serpiente,R.drawable.tiburon,

```

```

        R.drawable.rinoceronte,R.drawable.serpiente,R.drawable.tiburon
    }; // Guarda la imagen de cada boton
    int []idBoton={R.id.iv11,R.id.iv12,R.id.iv13,
        R.id.iv21,R.id.iv22,R.id.iv23,
        R.id.iv31,R.id.iv32,R.id.iv33,
        R.id.iv41,R.id.iv42,R.id.iv43}; // Guarda el id de los botones

    int iPuntuacion=0;
    int iTiempo=0;

    boolean bEsLaPrimera=true; // Para saber si es la primera o 2ª carta
    int idBotonPrimeroPulsado=0;
    int idBotonSegundoPulsado=0;
    int idImagenPrimerBotonPulsado=0;

```

Código

Ahora vamos a completar el código. La mayor parte tiene una traducción directa. Sólo vamos a comentar el método que busca un control en toda la lista disponible

```

    int iNumeroBotonPulsado=0;
    for(int i=0;i<12;i=i+1)
    {
        if(v.getId()==idBoton[i])
        {
            iNumeroBotonPulsado=i;
            break;
        }
    }

```

donde vemos que iteramos por medio de un bucle a lo largo de todos los ids hasta encontrar la del control pulsado. En ese caso guardamos la posición en a variable iNumeroBotonPulsado

Sonido

Eventos en diferido

Una característica de Android es ...

TODO: Handler

Tiempo

Podemos usar:

- Una medida de tiempos "a mano"
- Un control del interface de tiempos Chronometer

Para poner a 0 el tiempo, lo que tenemos que hacer es establecer el tiempo base del cronómetro (algo así como poner en hora)

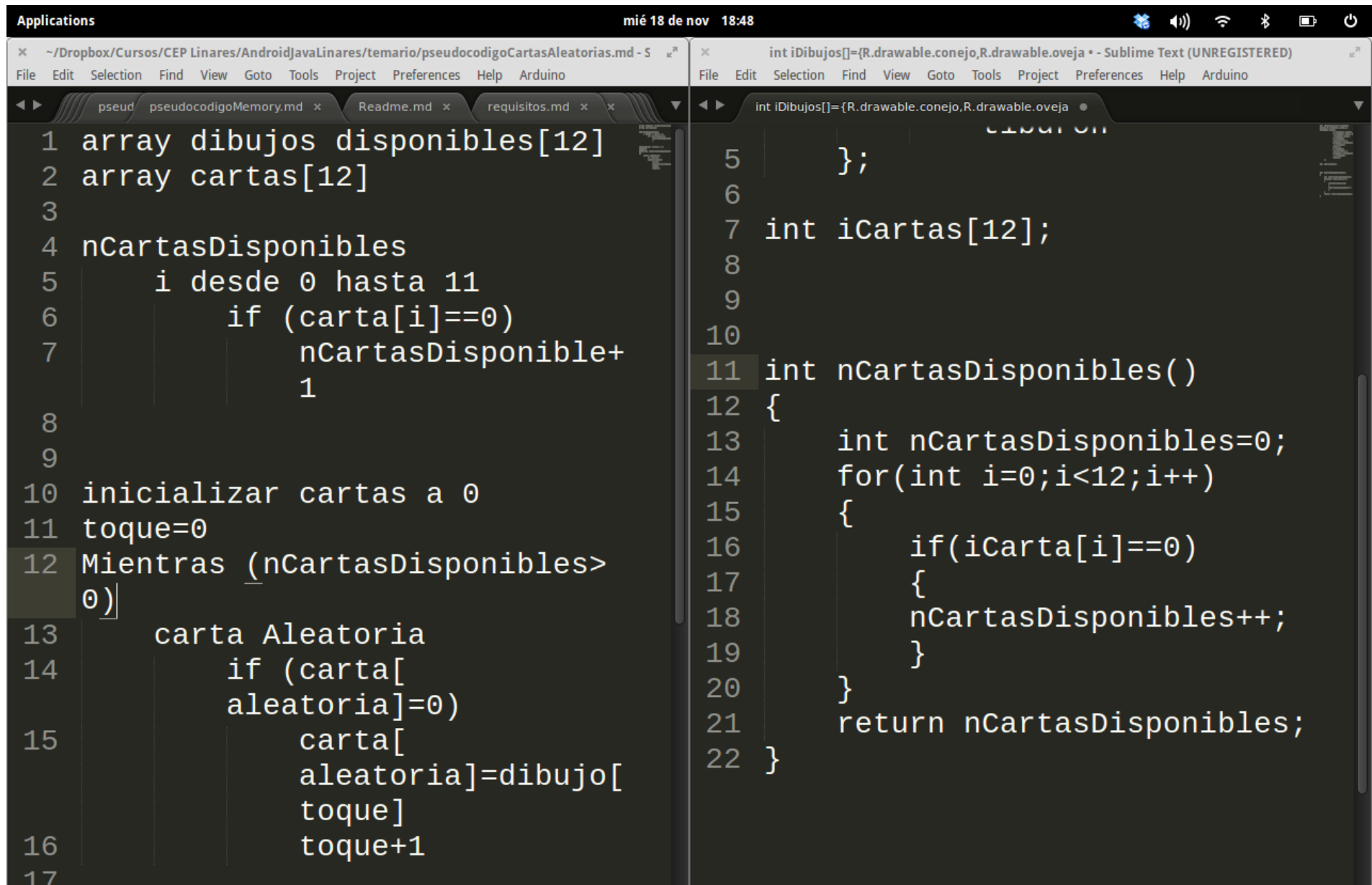
```
mChronometer.setBase(SystemClock.elapsedRealtime());
```

de <http://stackoverflow.com/questions/5345697/chronometer-reset> y

[http://developer.android.com/reference/android/widget/Chronometer.html#setBase\(long\)](http://developer.android.com/reference/android/widget/Chronometer.html#setBase(long))

Des-Ordenación de cartas

¿Funciona shuffle? <http://www.vogella.com/tutorials/JavaAlgorithmsShuffle/article.html>



```
Applications
mié 18 de nov 18:48

~/Dropbox/Cursos/CEP Linares/AndroidJavaLinares/temario/pseudocodigoCartasAleatorias.md - S
File Edit Selection Find View Goto Tools Project Preferences Help Arduino

pseud pseudocodigoMemory.md x Readme.md x requisitos.md x x
1 array dibujos disponibles[12]
2 array cartas[12]
3
4 nCartasDisponibles
5     i desde 0 hasta 11
6     if (carta[i]==0)
7         nCartasDisponibles+
8         1
9
10 inicializar cartas a 0
11 toque=0
12 Mientras (nCartasDisponibles>
13     0)
14     carta Aleatoria
15     if (carta[
16         aleatoria]=0)
17         carta[
18         aleatoria]=dibujo[
19         toque]
20         toque+1
21
22

int iDibujos[]={R.drawable.conejo,R.drawable.oveja} - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help Arduino

int iDibujos[]={R.drawable.conejo,R.drawable.oveja}
5     };
6
7     int iCartas[12];
8
9
10
11 int nCartasDisponibles()
12 {
13     int nCartasDisponibles=0;
14     for(int i=0;i<12;i++)
15     {
16         if(iCarta[i]==0)
17         {
18             nCartasDisponibles++;
19         }
20     }
21     return nCartasDisponibles;
22 }
```



```
17
18
19
20
```

On master* in temario, Line 12, Column 32

Tab Size: 4

Markdown

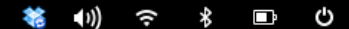
Line 11, Column 5

Tab Size: 4

Plain Text

Applications

mié 18 de nov 18:57



× ~/Dropbox/Cursos/CEP Linares/AndroidJavaLinares/temario/pseudocodigoCartasAleatorias.md - 5

File Edit Selection Find View Goto Tools Project Preferences Help Arduino

pseud pseudocodigoMemory.md × Readme.md × requisitos.md × ×

```
1 array dibujos disponibles[12]
2 array cartas[12]
3
4 nCartasDisponibles
5     i desde 0 hasta 11
6     if (carta[i]==0)
7         nCartasDisponibles+1
8
9
10 inicializar cartas a 0
11 toque=0
12 Mientras (nCartasDisponibles>0)
13     carta Aleatoria
14     if (carta[aleatoria]=0)
15         carta[
16             aleatoria]=dibujo[
17             toque]
18         toque+1
19
20
```

× int iDibujos[]={R.drawable.conejo,R.drawable.oveja} - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help Arduino

int iDibujos[]={R.drawable.conejo,R.drawable.oveja} •

```
10 {
11     int nCartasDisponibles=0;
12     for(int i=0;i<12;i++)
13     {
14         if(iCarta[i]==0)
15         {
16             nCartasDisponibles++;
17         }
18     }
19     return nCartasDisponibles;
20 }
21
22 void reparteCartas()
23 {
24     for(int i=0;i<12;i++)
25     { iCarta[0]; }
26     int iDibujoToca=0;
27
28     while(nCartasDisponibles(>)>0)
29     {
30         int iCartaAleatoria=(int)(
31             Math.Random()*12);
```

```
31      if(iCarta[
32      iCartaAleatoria]==0)
33      {
34          iCarta[iCartaAleatoria
35          ]=dibujo[iDibujoToca];
36          iDibujoToca++;
37      }
```

On master* in temario, Line 12, Column 32 Tab Size: 4 Markdown Line 36, Column 1 Tab Size: 4 Plain Text

Start y stop del juego

Sonido

Se puede utilizar casi cualquier formato: wav, midi, mp3. Pero el recomendado es el ogg, formato completa libre y tan versátil como el mp3. Si queremos usar este formato podemos convertir cualquier otro formato a ogg con este conversor online <http://audio.online-convert.com/es/convertir-a-ogg>

Para trabajar en Android con sonidos tenemos 2 opciones:

- MediaPlayer: una librería capaz de reproducir todos los formatos, tanto de vídeo como de audio, que nos proporciona una gran versatilidad pero con el coste de necesitar muchos recursos, lo que a veces la hace lenta.
- SoundPool: una clase pensada para reproducir sonidos de manera rápida (como es necesario en los juegos) y que nos permite preparar con antelación la reproducción de los audios, para que en el momento necesario estos suenen de manera instantánea.

Usaremos este último. Para su uso solo tenemos que crear el objeto SoundPool, cargar los sonidos con el método load, lo que prepara su reproducción y luego utilizar play para hacer que estos se reproduzcan instantáneamente.

En la parte de la creación del objeto, vamos a ver cómo hacer una zona de código condicional, es decir que dependiendo de la versión de Android que usemos se utilice un código u otro ([ejemplo](#))

En concreto podemos ver que a partir del API 21 se ha cambiado la forma de crear los objetos SoundPool, haciendo necesario usar el método Builder

```
SoundPool sp;
if((android.os.Build.VERSION.SDK_INT) >= 21){
    SoundPool.Builder sp21 = new SoundPool.Builder();
    sp21.setMaxStreams(5);
    sp = sp21.build();
}
else{
    sp = new SoundPool(5, AudioManager.STREAM_MUSIC, 0);
}
```

Una vez creado el objeto sólo tenemos que cargar los sonidos con load, lo que nos devolverá un id para cada sonido

```
idAplauso= sp.load(this,R.raw.applause,0);
idMal=sp.load(this,R.raw.evil,0);
idAcierto=sp.load(this,R.raw.sonido_acierto,0);
```

Para reproducirlo sólo haremos play

```
sp.play(idAcierto,1,1,1,0,1);
```

con los argumentos: play(idSonido,volumenIzda,volumenDrcha,repeticion,prioridad,velocidad)

Sonidos (descargados de <http://soundbible.com>) :

Acierto

Risa error

Aplauso