Tema: La Clase `Camera`

CONCEPTOS

Antecedentes.

La clase original de clase `Camera`, que ahora es obsoleta, es la siguiente:

```
public class Camera extends Object
java.lang.Object -> android.hardware.Camera
```

Android incluye soporte para varias cámaras y funciones de la cámara disponibles en los dispositivos, lo que permite capturar imágenes y vídeos en las aplicaciones.

Lo básico.

Android permite la captura de imágenes y vídeo a través de la API `android.hardware.camera2` o con el `Intent` de la cámara. Las clases principales son las siguientes:

`android.hardware.camera2`	API principal para el control de las cámaras del móvil. Para tomar fotos o videos.
`Camera`	API obsoleta para el control de las cámaras del dispositivo.
`SurfaceView`	Se utiliza para presentar una vista previa de cámara en directo para el usuario.
`MediaRecorder`	Para grabar vídeo de la cámara.
`Intent`	Un tipo de intento de `MediaStore.ACTION_IMAGE_CAPTURE` o `MediaStore.ACTION_VIDEO_CAPTURE` que se utiliza para capturar imágenes o vídeos sin utilizar el objeto `Camera`.

Las declaraciones en el Manifest

Antes de iniciar el desarrollo de la aplicación con la API de la cámara, se debe asegurar que el manifiesto tiene las declaraciones adecuadas para el uso de hardware de la cámara y otras funciones relacionadas.

Permisos de la cámara. La aplicación debe solicitar permiso para utilizar una cámara del dispositivo.

```
<uses-permission android:name="android.permission.CAMERA" />
```

Nota: Si se utiliza la cámara, vía un `Intent`, la aplicación no necesita solicitar este permiso.

Características de la cámara. La aplicación también debe declarar el uso de funciones de la cámara, por ejemplo:

```
<uses-feature android:name="android.hardware.camera" />
```

La adición de funciones de la cámara al manifiesto ocasiona que Google Play evite que la aplicación se instale en dispositivos que no incluyan una cámara o no sean compatibles con las funciones de la cámara que se especifica.

Si la aplicación utiliza una cámara o una característica de la cámara para una función particular, pero no lo requiere, se debe especificar esto en el manifiesto mediante la inclusión del atributo `android:required`, y asignarlo en `false`:

```
<uses-feature android:name="android.hardware.camera" android:required="false" />
```

Permiso de almacenamiento. Si la aplicación guarda las imágenes o videos en un almacenamiento externo del dispositivo (tarjeta SD), también debe especificar en el manifiesto.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Permiso de grabación de audio. Para grabar audio con captura de video, se debe solicitar el permiso de captura de audio.

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

Permiso de ubicación. Si la aplicación etiqueta imágenes con información de ubicación GPS, se debe solicitar permiso de ubicación:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Uso de las aplicaciones de la cámara.
Una forma rápida para la toma de imágenes o vídeos en la aplicación, sin necesidad de una gran cantidad de código adicional, es el uso de un `Intent` para invocar una aplicación de la cámara existente. Un intento a la cámara hace una petición para capturar una imagen o un videoclip a través de una aplicación de la cámara y luego devuelve el control a la aplicación. El procedimiento para invocar un intento cámara sigue estos pasos generales:

1. Crear un Camera `Intent`. Crea un `Intent` que solicite una imagen o video, utilizando uno de los siguientes intentos:

`MediaStore.ACTION_IMAGE_CAPTURE`. Tipo de `Intent` para solicitar la imagen a la aplicación con la cámara.

`MediaStore.ACTION_VIDEO_CAPTURE`. Tipo de `Intent` para solicitar un video a la aplicación con la cámara.

2. Iniciar el `Camera Intent`. Utilizar el método `startActivityForResult`() para ejecutar el intento de la cámara. Después de iniciar el intento, la aplicación se muestra la interface de usuario en la pantalla y el usuario puede tomar fotos o video.
3. Recepción del `Intent Result`. Configurar el método `onActivityResult`() de la aplicación para recibir la respuesta y los datos del intento de la cámara. Cuando el usuario termina, o cancela la operación, el sistema invoca a este método.

Construcción de una aplicación con la cámara.
Algunos desarrolladores pueden requerir una interfaz de usuario de la cámara que se adapte a la apariencia de su aplicación o proporcione características especiales. La creación de una actividad personalizada de la cámara requiere más código que utilizando un intento, pero se puede tener una experiencia más atractiva para los usuarios.

Nota: La siguiente guía es para la API obsoleta de la cámara. Para las aplicaciones nuevas o avanzadas de la cámara, se recomienda la API `android.hardware.camera2` más reciente.

Los pasos para crear una interfaz personalizada de la cámara son los siguientes:

- **Detectar y Acceder a la Cámara**. Crear código para comprobar la existencia de las cámaras y la petición de acceso.
- **Crear una clase de vista previa**. Crear una clase de vista previa de la cámara que herede de `SurfaceView` e implante la interfaz `SurfaceHolder`. Esta clase previsualiza las imágenes en directo desde la cámara.
- **Crear una plantilla de vista previa**. Una vez que se tenga la clase de vista previa de lacámara, crear una plantilla que incorpore la vista previa y los controles de interfaz de usuario que se desee.
- **Configurar los escuchas para la captura**. Conectar los escuchas de los controles de la interfaz para iniciar las capturas de imágenes o vídeo en respuesta a las acciones del usuario, como digitar un botón.
- **Capturar y guardar archivos**. Configurar el código para capturar imágenes o vídeos y guardar la salida.
- **Liberar la cámara**. Después de usar la cámara, la aplicación debe liberarla adecuadamente para su uso por otras aplicaciones.

El hardware de la cámara es un recurso compartido que debe ser manejado con cuidado para que la aplicación no choque con otras aplicaciones que también deseen usarla.

Precaución: Recordar que se debe liberar el objeto de la cámara con la llamada a `Camera.release`() cuando la aplicación termina su uso. Si la aplicación no libera correctamente la cámara, todos los intentos posteriores para acceder a la cámara, incluyendo los de la propia aplicación, producirán un error y pueden causar que la aplicación, u otra, se cierren.

Detección del hardware de la cámara.
Si la aplicación no requiere específicamente una cámara usando la declaración en el manifiesto, se debe comprobar si la cámara está disponible en tiempo de ejecución. Para realizar esta comprobación, utilizar el método `PackageManager.hasSystemFeature`(), como se muestra en el código de ejemplo a continuación:

```
/** Check if this device has a camera */
private boolean checkCameraHardware(Context context) {
    if (context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA)){
        // this device has a camera
```

```
        return true;
    } else {
        // no camera on this device
        return false;
    }
}
```

Los dispositivos Android pueden tener múltiples cámaras, por ejemplo, una cámara de frente para fotografía y una cámara frontal para video. Android 2.3 (API Nivel 9), o superiores, permiten comprobar el número de cámaras disponibles en un dispositivo utilizando el método `Camera.getNumberOfCameras`().

Acceso a las cámaras.
Si se ha detectado que el dispositivo tiene una cámara, se debe solicitar su acceso para conseguir una instancia de la cámara (a menos que se utilice un intento para acceder a la cámara).

Para acceder a la cámara principal, se utiliza el método Camera.open() y asegurarse de atrapar alguna excepción, como se muestra en el siguiente código:

```
/** A safe way to get an instance of the Camera object. */
public static Camera getCameraInstance(){
    Camera c = null;
    try {
        c = Camera.open(); // attempt to get a Camera instance
    }
    catch (Exception e){
        // Camera is not available (in use or does not exist)
    }
    return c; // returns null if camera is unavailable
}
```

Precaución: Comprobar si hay excepciones cuando se utiliza `Camera.open`(). El no comprobar las excepciones si la cámara está en uso, o no existe, provocará que la solicitud se cierre por el sistema.

Captura de vídeos.
La captura de vídeo utilizando la referencia de Android requiere un manejo cuidadoso del objeto de la cámara y la coordinación con la clase `MediaRecorder`. Durante la grabación de vídeo con la cámara, se debe administrar el `Camera.lock`() y `Camera.unlock`() para permitir que `MediaRecorder` acceda al hardware de la cámara, además a `Camera.open`() y `Camera.release`().

Nota: A partir de Android 4.0 (API de nivel 14), las llamadas a `Camera.lock`() y `Camera.unlock`()se gestionan de forma automática.

A diferencia de la toma de fotografías con la cámara del dispositivo, la captura de vídeo requiere una llamada muy particular. Se debe seguir un orden específico de ejecución para preparar con éxito la captura de vídeo, como se detalla a continuación.

1. **Abrir la cámara**. Utilizar `Camera.open`() para obtener una instancia del objeto de cámara.
2. **Conectar la vista previa**. Preparar una vista previa de la imagen de la cámara en vivo conectando un `SurfaceView` a la cámara mediante `Camera.setPreviewDisplay`().
3. **Iniciar vista previa.** Invocar a `Camera.startPreview`() para comenzar la presentación de las imágenes de la cámara en vivo.
4. **Iniciar la grabación de vídeo**. Los siguientes pasos deben ser en orden para grabar vídeo con éxito:
    a. **Desbloquear la cámara**. Desbloquear cámara para uso de `MediaRecorder` invocando a `Camera.unlock`().
    b. **Configurar el MediaRecorder**. Llamar a los siguientes métodos `MediaRecorder` en este orden.

1. `setCamera`(). Configurar la cámara para utilizar la captura de vídeo, utilizar la instancia actual de `Camera`.
2. `setAudioSource`(). Establecer la fuente de audio, utilizar `MediaRecorder.AudioSource.CAMCORDER`.
3. `setVideoSource`(). Establecer la fuente de vídeo, utilizar `MediaRecorder.VideoSource.CAMERA`.
4. Ajustar el formato de salida de vídeo y codificación. Para Android 2.2 (API Nivel 8) y superior, utilizar el método `MediaRecorder.setProfile`, y obtener un perfil de instancia utilizando `CamcorderProfile.get`(). Para las versiones de Android anteriores a 2.2, configurar los parámetros de formato y codificación de salida de vídeo:
   i. `setOutputFormat`(). Ajustar el formato de salida, especificar la configuración predeterminada o `MediaRecorder.OutputFormat.MPEG_4`.
   ii. `setAudioEncoder`(). Establecer el tipo de codificación de sonido, especificar la configuración predeterminada o `MediaRecorder.AudioEncoder.AMR_NB`.
   iii. `setVideoEncoder`(). Establecer el tipo de codificación de vídeo, especificar la configuración predeterminada o `MediaRecorder.VideoEncoder.MPEG_4_SP`.
5. `setOutputFile`(). Establecer el archivo de salida, utilizar `getOutputMediaFile(MEDIA_TYPE_VIDEO).toString()` del método de ejemplo de la sección guardar los archivos multimedia.
6. `setPreviewDisplay`(). Especificar el elemento de la plantilla de vista previa `SurfaceView` para la aplicación. Utilizar el mismo objeto que especificado para `Connect Preview`.
   Precaución: Se debe llamar a estos métodos de configuración `MediaRecorder` en este orden, de lo contrario la aplicación encontrará errores y la grabación fallará.

c. **Preparar MediaRecorder**. Preparar el `MediaRecorder` con valores de configuración proporcionados por llamar `MediaRecorder.prepare` ().
d. **Comience MediaRecorder**. Iniciar la grabación de vídeo llamando `MediaRecorder.start` ().

5. **Detener la grabación de vídeo**. Llame a los métodos siguientes en orden, para completar con éxito una grabación de vídeo:
   a. **Detener `MediaRecorder`**. Detener la grabación de vídeo llamando a `MediaRecorder.stop`().
   b. **Restablecer `MediaRecorder`**. De forma opcional, retire los ajustes de configuración de la grabadora llamando `MediaRecorder.reset`().
   c. **Liberar `MediaRecorder`**. Soltar el `MediaRecorder` llamando `MediaRecorder.release` ().
   d. **Bloquear la cámara**. Bloquear la cámara para que las futuras sesiones de `MediaRecorder` pueden utilizarla llamando `Camera.lock`(). A partir de Android 4.0 (API de nivel 14), no se requiere esta llamada a menos que el `MediaRecorder.prepare`() falle.
6. **Detener la vista previa**. Cuando su actividad ha terminado de usar la cámara, detener la vista previa utilizando `Camera.stopPreview`().
7. **Liberar la cámara**. Liberar la cámara para que otras aplicaciones puedan utilizarla llamando `Camera.release`().

Nota: Es posible utilizar `MediaRecorder` sin crear primero una vista previa de la cámara y saltar los primeros pasos de este proceso.
Sugerencia: Si la aplicación se utiliza típicamente para la grabación de vídeo, ajustar `setRecordingHint(booleano)` a `true` antes de comenzar la previsualización. Esta configuración ayuda a reducir el tiempo que se necesita para iniciar la grabación.

Características de la cámara.
Android es compatible con una amplia gama de funciones de la cámara que se pueden controlar con la aplicación de cámara, como el formato de imagen, modo de flash, ajustes de enfoque, y muchos más. La mayoría de las funciones de la cámara se

acceden y establecen con el objeto `Camera.Parameters`. Sin embargo, hay varias características importantes que requieren más que simples ajustes en `Camera.Parameters`.

Tabla 1. Características comunes de la cámara según el nivel de la API de Android en el que se introdujeron.

Característica de API	Level	Descripción
`Face Detection`	14	Identifica caras humanas en la imagen y las usa para el foco, medición y balance de blancos.
`Metering Areas`	14	Especifica una o más áreas en una imagen para cálculo de balance de blancos.
`Focus Areas`	14	Asigna una o más áreas de una imagen para el uso del foco.
`White Balance Lock`	14	Detiene o inicia el ajuste automático del balance de blancos.
`Exposure Lock`	14	Detiene o inicia los ajustes automáticos de exposición.
`Video Snapshot`	14	Toma una foto mientras toma video (`frame grab`).
`Time Lapse Video`	11	Graba tramas con retardos para grabar un video con lapso de tiempo.
`Multiple Cameras`	9	Permite más de una cámara, incluyendo la cámara frontal y trasera.
`Focus Distance`	9	Reporta distancias entre la cámara y los objetos que parecen estar enfocados.
`Zoom`	8	Configura la ampliación de la imagen.
`Exposure Compensation`	8	Incrementa o decrementa el nivel de exposición de luz.
`GPS Data`	5	Incluye u omite los datos de la posición geográfica con la imagen.
`White Balance`	5	Configura el balance blanco, que afecta los valores del color en la imagen capturada.
`Focus Mode`	5	Establecer cómo la cámara se centra en un tema como automática, fija, macro o infinito.
`Scene Mode`	5	Aplicar un modo predeterminado para tipos específicos de la fotografía nocturna, playa, nieve o escenas de luz de velas.
`JPEG Quality`	5	Ajusta el nivel de compresión de una imagen JPEG, lo que aumenta o disminuye la calidad y tamaño del archivo de la imagen.
`Flash Mode`	5	Apaga el flash, lo apaga o utiliza la configuración automática.
`Color Effects`	5	Aplica efectos de color a la imagen capturada, blanco y negro, tono sepia o negativo.
`Anti-Banding`	5	Reduce el efecto de bandas en gradientes de color debido a la compresión JPEG.
`Picture Format`	1	Especificar el formato de archivo de la imagen.
`Picture Size`	1	Especificar las dimensiones en píxeles de la imagen guardada.

Nota: Estas características no son compatibles con todos los dispositivos debido a las diferencias de hardware y software de aplicación.

DESARROLLO

EJEMPLO 1.

Paso 1. Crear un nuevo proyecto Camara en Android Studio. En la carpeta java/com.example.mipaquete, abrir y modificar el archivo predeterminado MainActivity.java con el siguiente código:

```
import java.io.*;
import java.util.Date;
import java.text.SimpleDateFormat;
import android.os.*;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity {
    private final String ruta =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES) +
"/misfotos/";
    private File    f = new File(ruta);
    private Button  jbn;
    @Override
    protected void onCreate(Bundle b) {
        super.onCreate(b);
        setContentView(R.layout.activity_main);
        jbn = (Button) findViewById(R.id.xbn);
        f.mkdirs();
        jbn.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                String s = ruta + getCode() + ".jpg";
                File f1 = new File(s);
                try{
                    f1.createNewFile();
                }catch(IOException ex){
                    Log.e("Error", "Error:" + ex);
                }
```

```
                Uri u = Uri.fromFile(f1);
                Intent in = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                in.putExtra(MediaStore.EXTRA_OUTPUT, u);
                startActivityForResult(in, 0);
            }
        });
    }
    private String getCode(){
        SimpleDateFormat sdf = new SimpleDateFormat("yyyymmddhhmmss");
        String n = "pic_" + sdf.format(new Date());;
        return n;
    }
}
```

Paso 2. En la carpeta res/layout, abrir y modificar el archivo activity_main.xml con el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main"
    tools:context=".CamaraActivity">
    <Button
        android:id="@+id/xbn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/foto"/>
</RelativeLayout>
```

Paso 3. En la carpeta res/values, abrir y modificar el archivo strings.xml con el siguiente código:

```
<?xml versión="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Camara</string>
    <string name="action_settings">Settings</string>
    <string name="foto">Tomar foto</string>
</resources>
```

Paso 4. En la carpeta app/manifests, abrir y modificar el archivo AndroidManifest.xml para agregar las etiquetas de permisos de uso de la cámara y almacenamiento, entre las etiquetas `<manifest>` y `<application>`, como se muestra enseguida con **letras negritas**:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.escom.camara">
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        :    >
            <activity
                :    >
        :
```

```
            </activity>
    </application>
</manifest>
```

Paso 5. Por último, ejecutar la aplicación. La plantilla solamente contiene un botón para tomar la foto. Al digitar el botón se muestra una animación predeterminada de Android Studio, para indicar que se captura una foto. Si se digita el botón inferior X, para cerrar la imagen de la foto, se regresa a la plantilla principal para tomar otra foto.

- Repetir el ejercicio en un dispositivo real. Verificar en la carpeta Galería, o una equivalente, la foto real tomada.

EJEMPLO 2.

Paso 1. Crear un nuevo proyecto Camara2 en Android Studio. En la carpeta java/com.example.mipaquete, abrir y modificar el archivo predeterminado MainActivity.java con el siguiente código:

```
import java.io.*;
import android.app.Activity;
import android.content.Intent;
import android.graphics.*;
import android.media.MediaScannerConnection;
import android.media.MediaScannerConnection.MediaScannerConnectionClient;
import android.net.Uri;
import android.os.*;
import android.provider.MediaStore;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;
public class MainActivity extends Activity {
    Button      jbn;
    RadioButton jrb1, jrb2;
    Intent      i;
    int         TAKE_PICTURE = 1;
    int         SELECT_PICTURE = 2;
    String      s = "";
```

```
    @Override
    public void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.activity_main);
        s = Environment.getExternalStorageDirectory() + "/test.jpg";
        jbn = (Button)findViewById(R.id.xbn1);
        jbn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                jrb1 = (RadioButton)findViewById(R.id.xrb1);
                jrb2 = (RadioButton)findViewById(R.id.xrb2);
                i =  new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                int code = TAKE_PICTURE;
                if (jrb1.isChecked()) {
                    Uri output = Uri.fromFile(new File(s));
                    i.putExtra(MediaStore.EXTRA_OUTPUT, output);
                } else if (jrb2.isChecked()){
                    i = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Images.Media.INTERNAL_CONTENT_URI);
                    code = SELECT_PICTURE;
                }
                startActivityForResult(i, code);
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == TAKE_PICTURE) {
            if (data != null) {
                if (data.hasExtra("data")) {
                    ImageView iv = (ImageView)findViewById(R.id.xiv1);
                    iv.setImageBitmap((Bitmap) data.getParcelableExtra("data"));
                }
            } else {
                ImageView iv = (ImageView)findViewById(R.id.xiv1);
                iv.setImageBitmap(BitmapFactory.decodeFile(s));
                new MediaScannerConnectionClient() {
                    private MediaScannerConnection msc = null; {
                        msc = new MediaScannerConnection(getApplicationContext(), this);
msc.connect();
                    }
                    public void onMediaScannerConnected() {
                        msc.scanFile(s, null);
                    }
                    public void onScanCompleted(String path, Uri uri) {
                        msc.disconnect();
                    }
                };
            }
        } else if (requestCode == SELECT_PICTURE){
            Uri image = data.getData();
            InputStream is;
            try {
                is = getContentResolver().openInputStream(image);
                BufferedInputStream bis = new BufferedInputStream(is);
                Bitmap bitmap = BitmapFactory.decodeStream(bis);
```

```
ImageView iv = (ImageView)findViewById(R.id.xiv1);
```

```
                iv.setImageBitmap(bitmap);
            } catch (FileNotFoundException e) {}
        }
    }
}
```

Paso 2. En la carpeta res/layout, abrir y modificar el archivo activity_main.xml con el siguiente código

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/widget28"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <Button
        android:text="Toma foto"
        android:id="@+id/xbn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true">
    </Button>
    <RadioGroup
        android:id="@+id/xrg1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true">
        <RadioButton
            android:id="@+id/xrb0"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tomar una vista previa">
        </RadioButton>
        <RadioButton
            android:id="@+id/xrb1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tomar una imagen completa">
        </RadioButton>
        <RadioButton
            android:id="@+id/xrb2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Mostrar una foto de la galería">
        </RadioButton>
    </RadioGroup>
    <ImageView
        android:layout_width="wrap_content"
        android:layout_below="@+id/xrg1"
        android:layout_height="wrap_content"
        android:id="@+id/xiv1">
    </ImageView>
</RelativeLayout>
```

Paso 3. En la carpeta res/values, abrir y modificar el archivo strings.xml con el siguiente código:

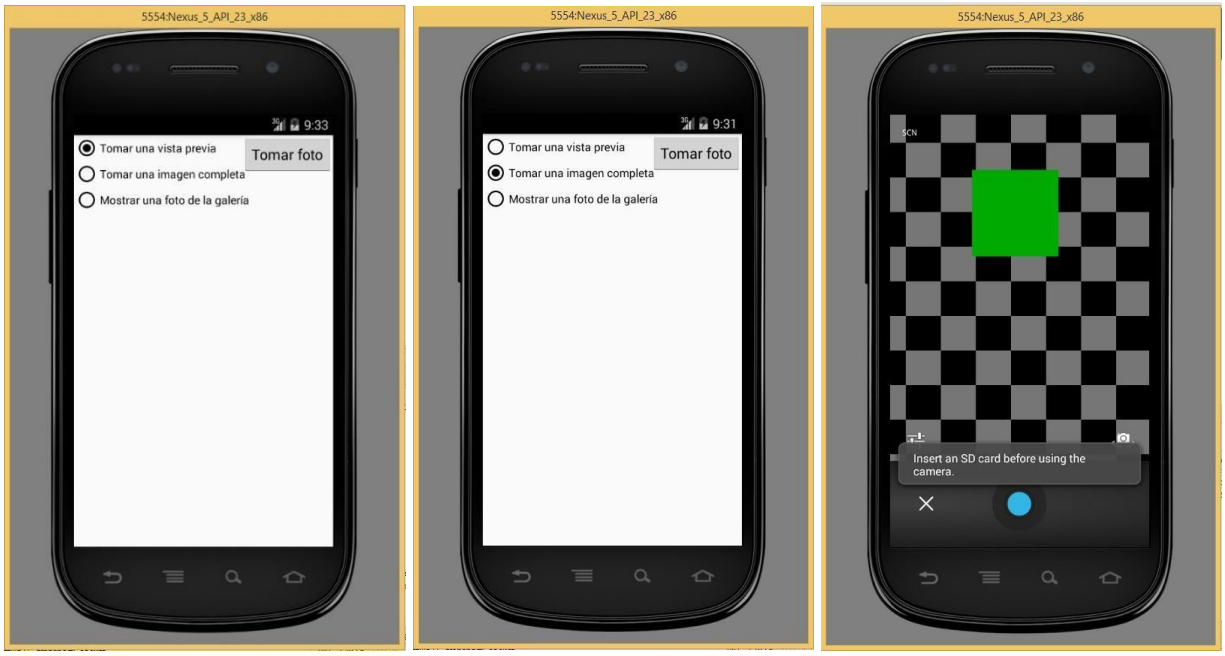
```
<?xml versión=”1.0” encoding=”utf-8”?>
<resources>
    <string name="app_name">Camara</string>
    <string name="action_settings">Settings</string>
    <string name="foto">Tomar foto</string>
</resources>
```

Paso 4. En la carpeta app/manifests, abrir y modificar el archivo AndroidManifest.xml para agregar las etiquetas de permisos de uso de la cámara y almacenamiento, entre las etiquetas `<manifest>` y `<application>`, como se muestra enseguida con **letras negritas**:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.escom.camara">
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        :    >
            <activity
                :    >
        :
            </activity>
    </application>
</manifest>
```

Paso 5. Por último, ejecutar la aplicación. La plantilla contiene tres opciones y un botón para tomar la foto.

a. Si se utiliza el emulador virtual, al digitar la selección Tomar una vista previa o Tomar una imagen completa, se muestra una animación predeterminada de Android Studio, para indicar que se captura una foto, en forma similar al ejemplo anterior. Si se digita el botón inferior **X**, para cerrar la imagen de la foto, se regresa a la plantilla principal para tomar otra foto.



b. Si se digita la selección Mostrar una foto de la galería, se muestra una opción de una carpeta con imágenes:

c. Utilizar el dispositivo real para ejecutar lo siguiente:

- Si se digita la opción Tomar una vista previa, se muestra la imagen actual capturada por el lente de la cámara.
- Si se digita la opción Toma una imagen completa, se captura una foto. Buscar la foto capturada en la galería del dispositivo real.
- Si se digita el botón Mostrar una foto de la galería, se abre la ventana en la cual se selecciona una foto que se mostrará en la pantalla del dispositivo real.

NOTA. Generar un reporte AlumnoCamaraGrupo.pdf con las imágenes obtenidas en la ejecución de los ejercicios y enviarlo al sitio indicado por el profesor.