Using Camera in Android

Via MediaStore Activity API or Direct Camera API

Two Levels for Camera Access

- Access Camera via MediaStore Activity API
  - android.provider.MediaStore

- Use Camera via more direct Camera object API
  - android.hardware.Camera.*

Using MediaStore Activity

- This is an android activity which provides basic ACTIONS on camera hardwares
- Use this activity in standard way of starting activities via intents:

```
Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
```

```
startActivityForResult(intent,0)
```

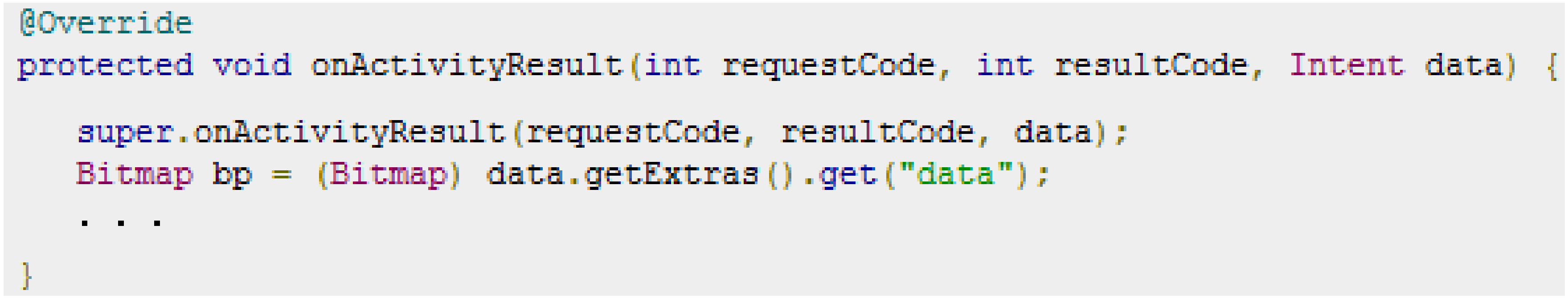
Using MediaStore Activity

```
Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
```

Sr.No	Intent type and description
1	**ACTION_IMAGE_CAPTURE_SECURE** It returns the image captured from the camera , when the device is secured
2	**ACTION_VIDEO_CAPTURE** It calls the existing video application in android to capture video
3	**EXTRA_SCREEN_ORIENTATION** It is used to set the orientation of the screen to vertical or landscape
4	**EXTRA_FULL_SCREEN** It is used to control the user interface of the ViewImage
5	**INTENT_ACTION_VIDEO_CAMERA** This intent is used to launch the camea in the video mode
6	**EXTRA_SIZE_LIMIT** It is used to specify the size limit of video or image capture size

Using MediaStore Activity
Getting the Result from Activity

- You must implement the **onActivityResult()** method:



```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

   super.onActivityResult(requestCode, resultCode, data);
   Bitmap bp = (Bitmap) data.getExtras().get("data");
   . . .
}
```

Using MediaStore Activity: Simple Application

```java
public class MainActivitv extends Activity {
   ImageView imgFavorite;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      imgFavorite = (ImageView)findViewById(R.id.imageView1);
      imgFavorite.setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View v) {
            open();
         }
      });
   }
   public void open(){
      Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
      startActivityForResult(intent, 0);
   }
   @Override
   protected void onActivityResult(int requestCode, int resultCode, Intent data) {
      // TODO Auto-generated method stub
      super.onActivityResult(requestCode, resultCode, data);
      Bitmap bp = (Bitmap) data.getExtras().get("data");
      imgFavorite.setImageBitmap(bp);
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
}
```

Using MediaStore Activity: Simple Application

content of **res/layout/activity_main.xml file:**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity">

   <ImageView
   android:id="@+id/imageView1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_marginLeft="34dp"
   android:layout_marginTop="36dp"
   android:contentDescription="@string/hello_world"
   android:src="@drawable/ic_launcher" />

   <TextView
   android:id="@+id/textView1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentTop="true"
   android:layout_alignRight="@+id/imageView1"
   android:text="@string/tap"
   android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

Using MediaStore Activity: Simple Application

content of **AndroidManifest.xml:**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.camera"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="17" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
         <activity
            android:name="com.example.camera.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
               <action android:name="android.intent.action.MAIN" />

               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
       </activity>
   </application>

</manifest>
```

Using Camera Hardware API

- Use android.hardware.Camera(.*)

```
Camera object = null;
object = Camera.open();
```

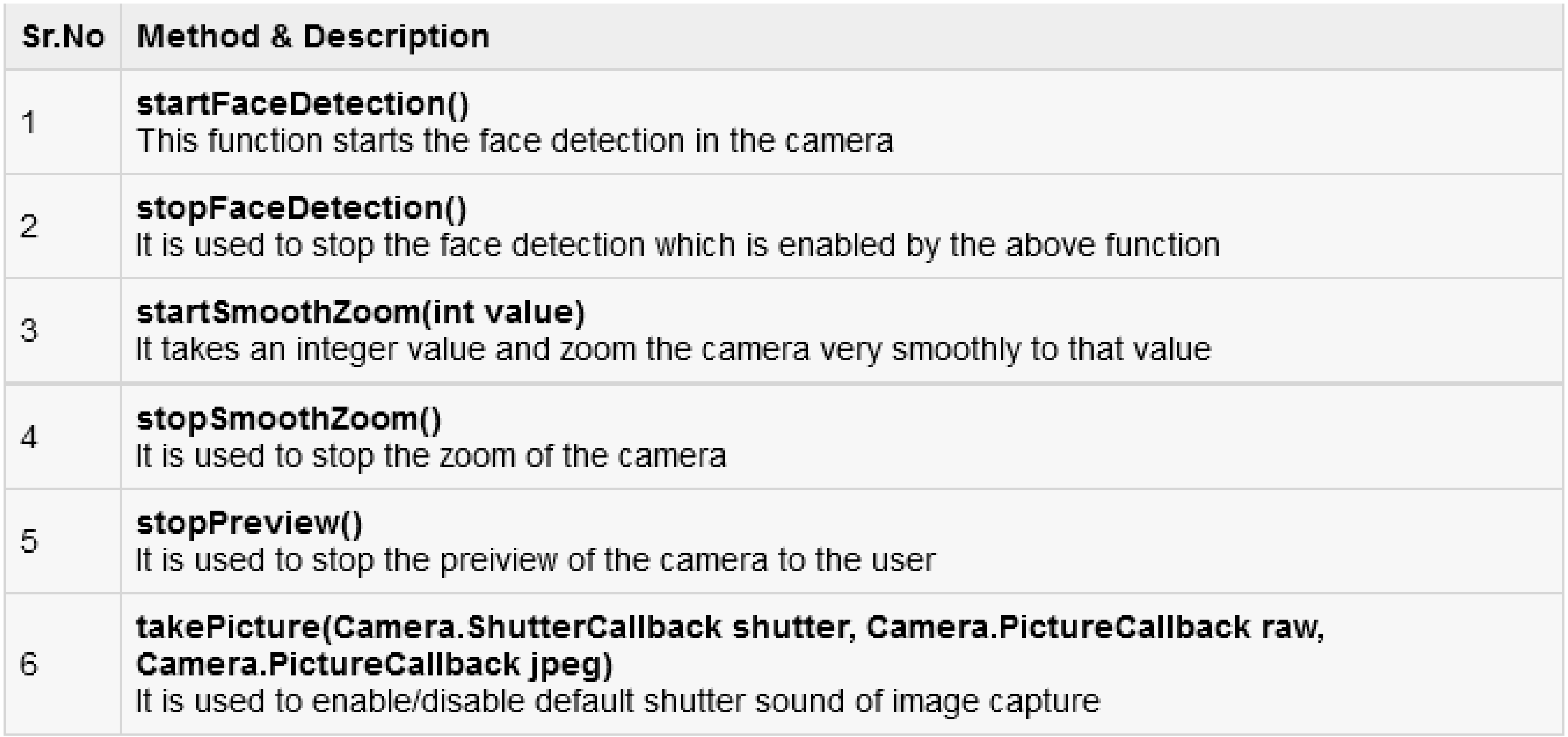
Sr.No	Method & Description
1	**getCameraInfo(int cameraId, Camera.CameraInfo cameraInfo)** It returns the information about a particular camera
2	**getNumberOfCameras()** It returns an integer number defining of cameras availaible on device
3	**lock()** It is used to lock the camera , so no other application can access it
4	**release()** It is used to release the lock on camera , so other applications can access it
5	**open(int cameraId)** It is used to open particular camera when multiple cameras are supported
6	**enableShutterSound(boolean enabled)** It is used to enable/disable default shutter sound of image capture

Using Camera Hardware API
Important Classes

Class	Description
Camera	It is used to control the camera and take images or capture video from the camera
SurfaceView	This class is used to present a live camera preview to the user.

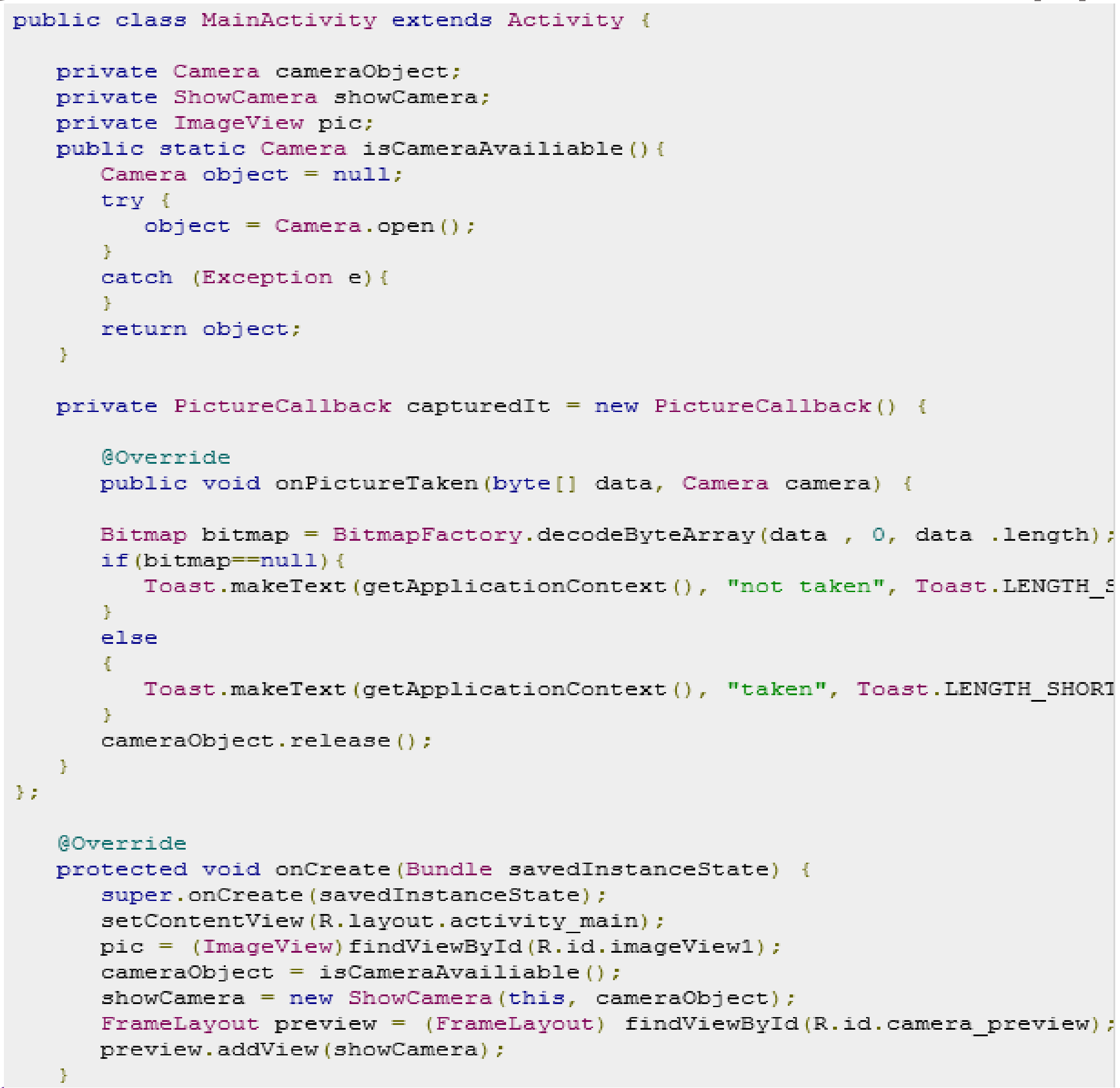
```
public class ShowCamera extends SurfaceView implements SurfaceHolder.Callback {

   private Camera theCamera;

   public void surfaceCreated(SurfaceHolder holder) {
      theCamera.setPreviewDisplay(holder);
      theCamera.startPreview();
   }
   public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3){
   }
   public void surfaceDestroyed(SurfaceHolder arg0) {
   }
}
```

Using Camera Hardware API
Other Camera Options



Sr.No	Method & Description
1	**startFaceDetection()** This function starts the face detection in the camera
2	**stopFaceDetection()** It is used to stop the face detection which is enabled by the above function
3	**startSmoothZoom(int value)** It takes an integer value and zoom the camera very smoothly to that value
4	**stopSmoothZoom()** It is used to stop the zoom of the camera
5	**stopPreview()** It is used to stop the preivew of the camera to the user
6	**takePicture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback jpeg)** It is used to enable/disable default shutter sound of image capture

Using Camera Hardware API: Simple App



```
public class MainActivity extends Activity {

   private Camera cameraObject;
   private ShowCamera showCamera;
   private ImageView pic;
   public static Camera isCameraAvailiable(){
      Camera object = null;
      try {
         object = Camera.open();
      }
      catch (Exception e){
      }
      return object;
   }

   private PictureCallback capturedIt = new PictureCallback() {

      @Override
      public void onPictureTaken(byte[] data, Camera camera) {

      Bitmap bitmap = BitmapFactory.decodeByteArray(data , 0, data .length);
      if(bitmap==null){
         Toast.makeText(getApplicationContext(), "not taken", Toast.LENGTH_S
      }
      else
      {
         Toast.makeText(getApplicationContext(), "taken", Toast.LENGTH_SHORT
      }
      cameraObject.release();
   }
};

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      pic = (ImageView)findViewById(R.id.imageView1);
      cameraObject = isCameraAvailiable();
      showCamera = new ShowCamera(this, cameraObject);
      FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
      preview.addView(showCamera);
   }
```

Using Camera Hardware API: Simple App

```
public class ShowCamera extends SurfaceView implements SurfaceHolder.Callback {

   private SurfaceHolder holdMe;
   private Camera theCamera;

   public ShowCamera(Context context,Camera camera) {
      super(context);
      theCamera = camera;
      holdMe = getHolder();
      holdMe.addCallback(this);
   }

   @Override
   public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
   }

   @Override
   public void surfaceCreated(SurfaceHolder holder) {
      try   {
         theCamera.setPreviewDisplay(holder);
         theCamera.startPreview();
      } catch (IOException e) {
      }
   }

   @Override
   public void surfaceDestroyed(SurfaceHolder arg0) {
   }

}
```