**Android Tutorial**

**-Using Hardware Sensors**

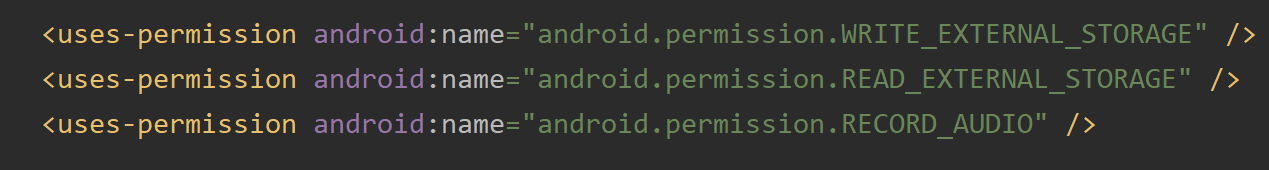
**-Voice Recognition using Google API  
  
  
  
by Shams Ahsanullah**

**DMIT2504**

**Topic 1: Using Hardware Sensors**

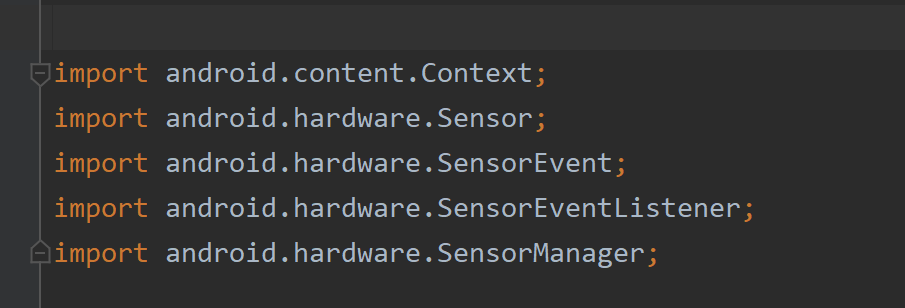
Android has a number of hardware sensors built in that can be utilized programmatically for different purposes. I have worked on the following sensors:

* Gyroscope
* Proximity Sensor
* Accelerometer
* Gravity Sensor

Before starting any section, be sure to add the following permissions to your manifest-  


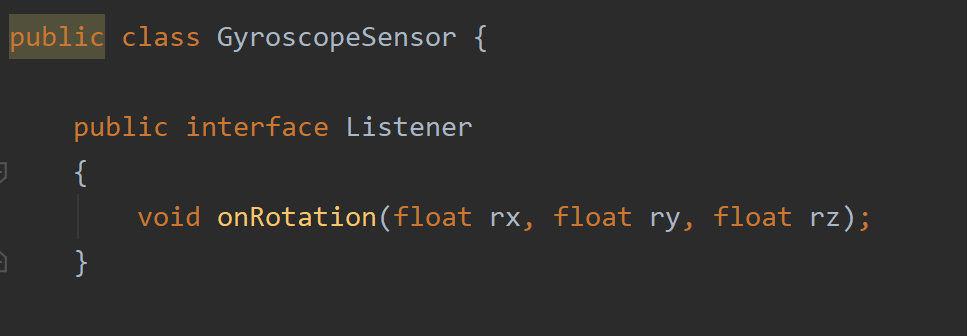
**Gyroscope – Voice Recorder**

I have made an app that can record voice. If a switch is enabled, then whenever the app tilts to a certain position and crosses a threshold, the recording enables. Again when it comes back to the previous position, the recording stops. In this tutorial, I will show you how to use the gyroscope. Here are the steps:

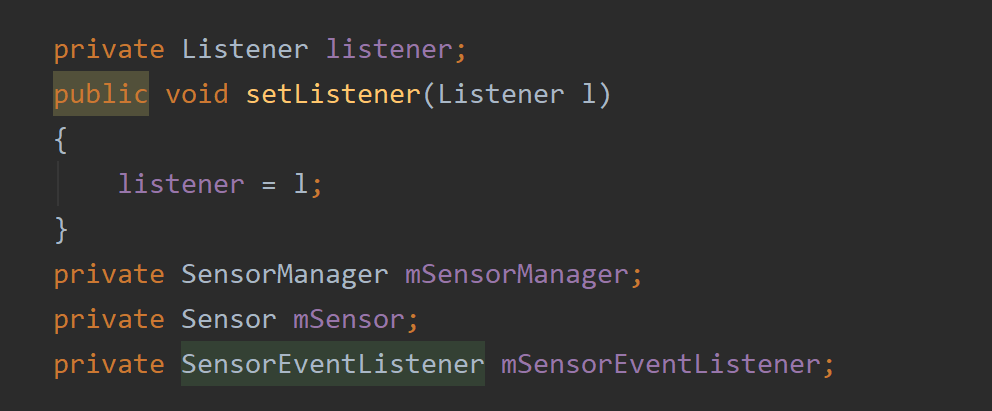
\*\*Packages imported for this class-  


1. Create an activity where you will use the gyroscope. Let’s call it “AudioRecorderActivity”.

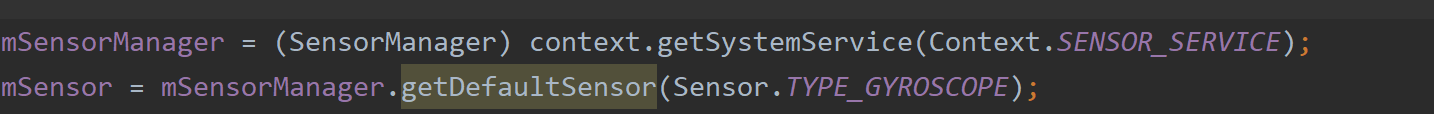
2. Create a class for the gyroscope sensor. It is efficient to use a class for sensor control. Call it “GyroscopeSensor”.

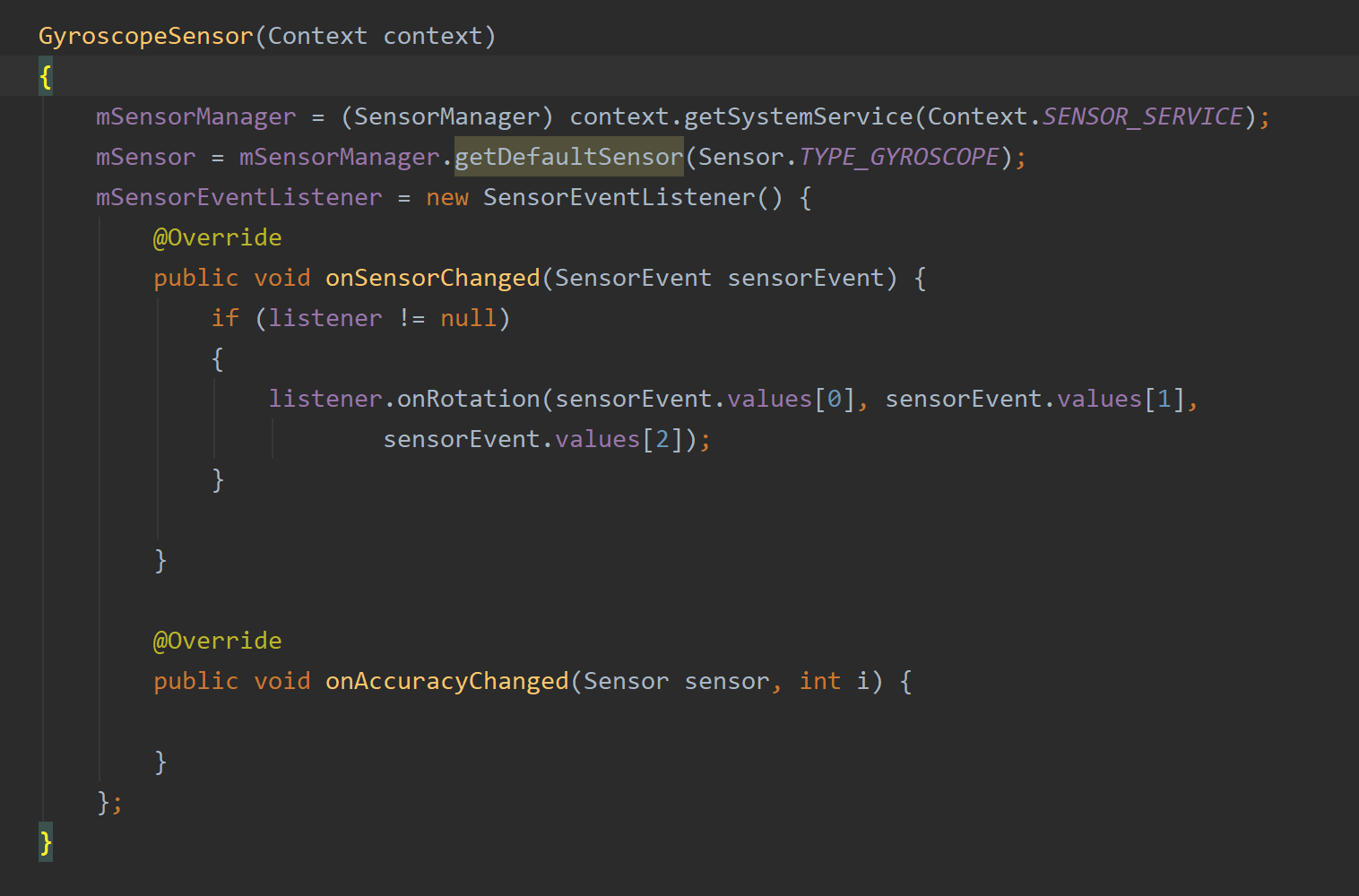
3. Create an interface and name it “Listener”. Inside it, we can add the onRotation method. It will have the x, y and z axis for parameters.  


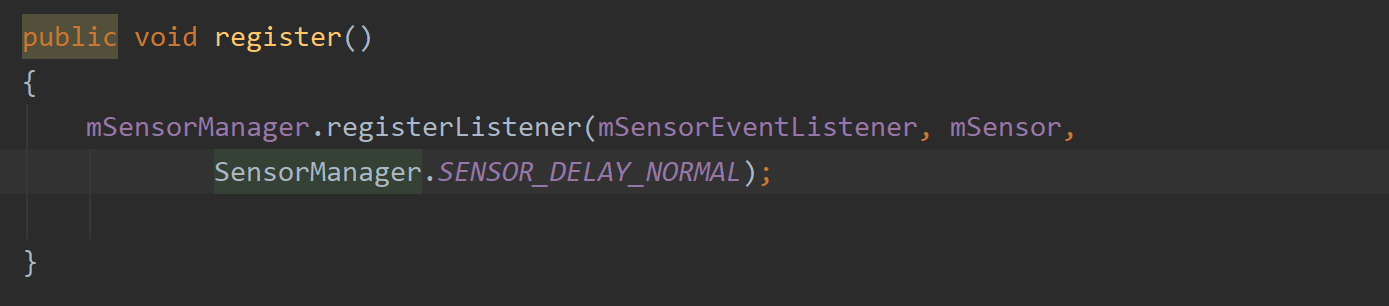
4. Add Listener, SensorManager, Sensor and SensorEventListener objects as shown. Then, add a setListener method that takes in a listener parameter-

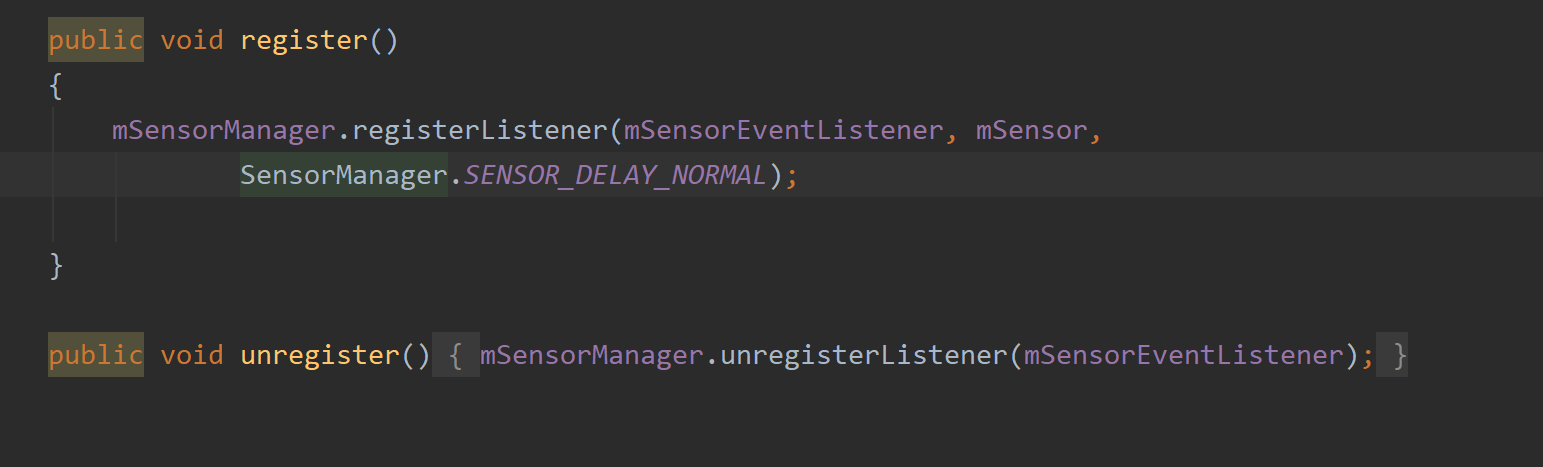


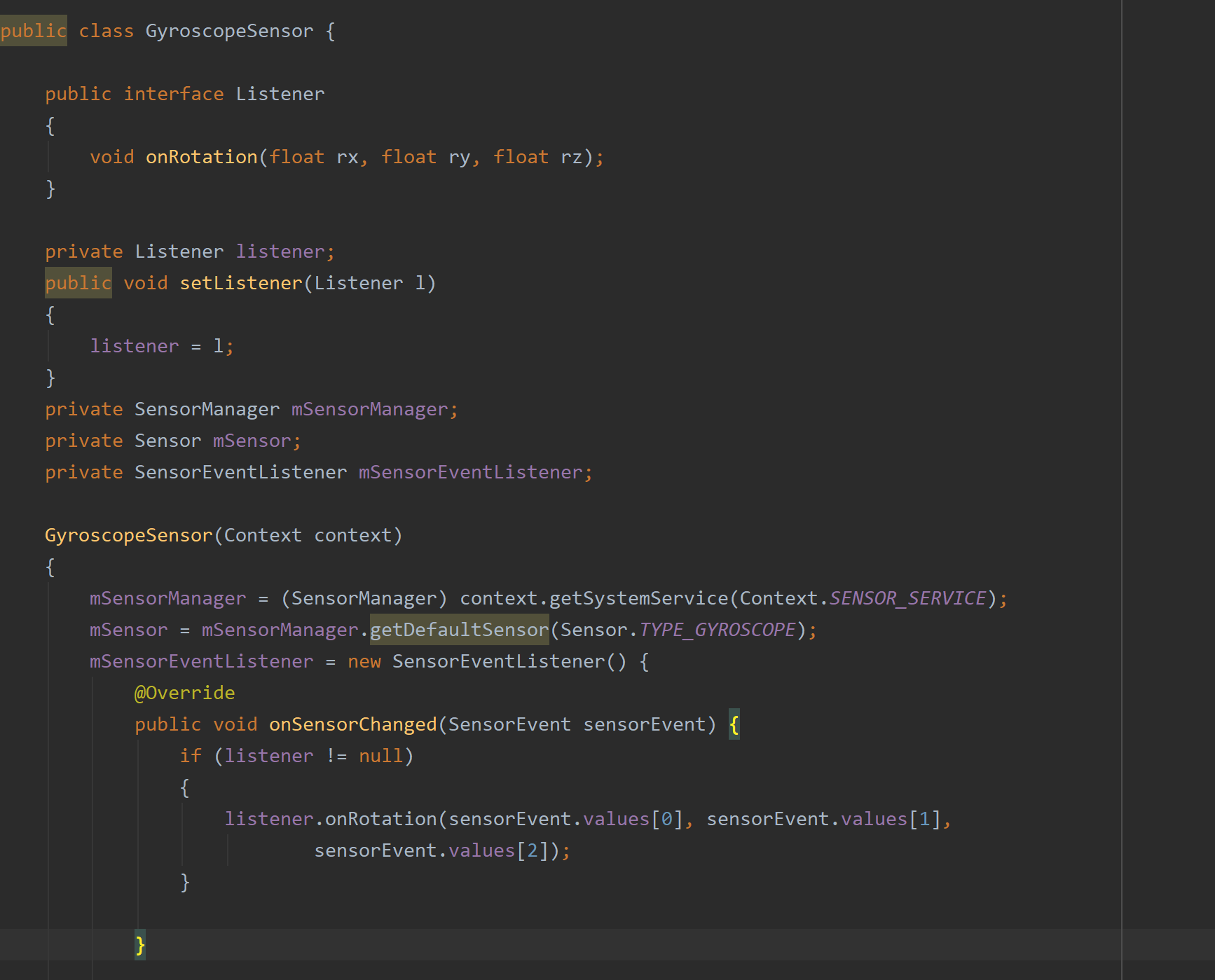
5. Add a constructor that takes in a context. We need to initiate the sensor manager here. The sensor manager calls the system service for sensors. Then, for the sensor object, we initiate it with the sensor we are interested in. With getDefaultSensor, we add the sensor type. Here, the type is TYPE\_GYROSCOPE.

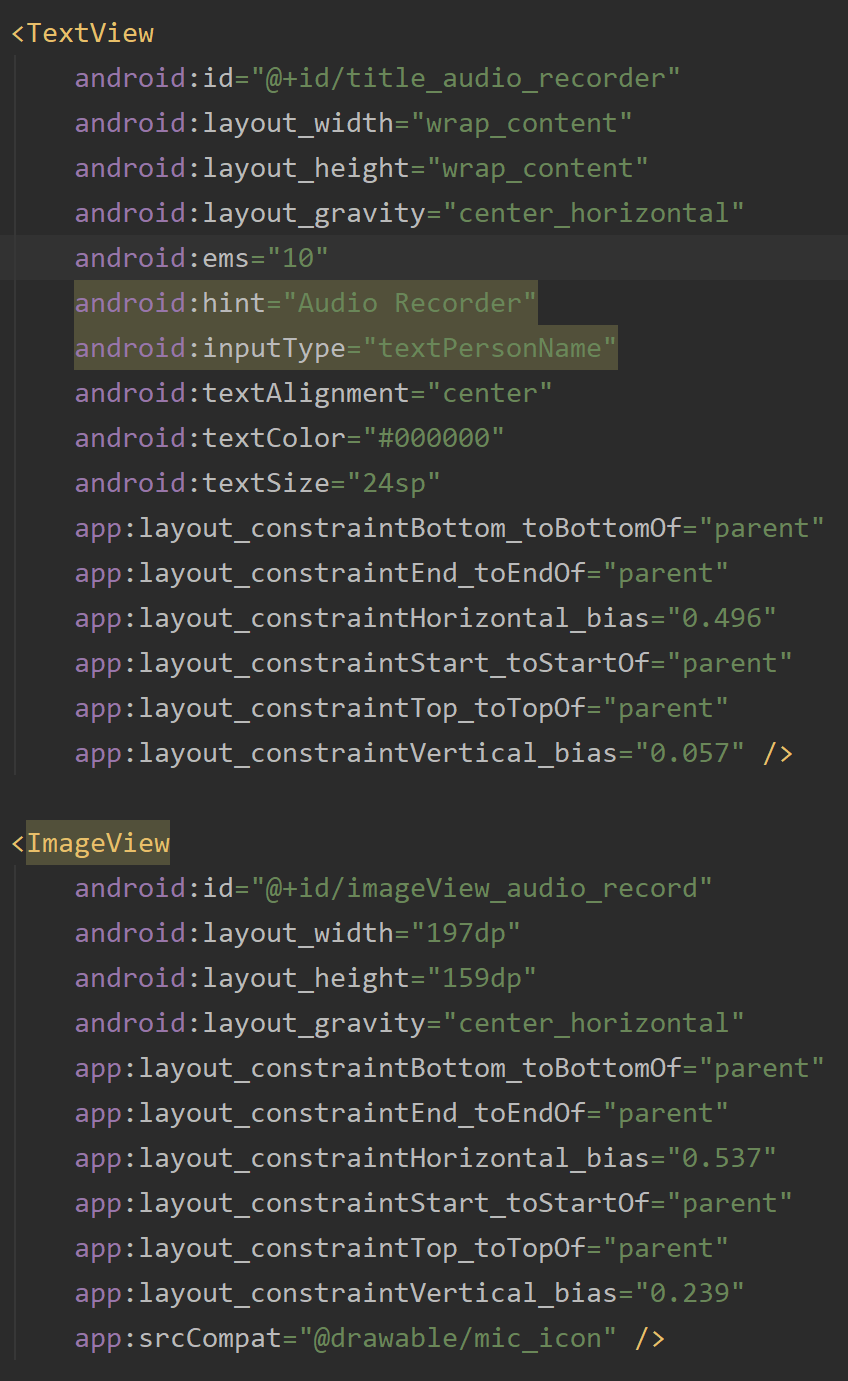
  
  
This link describes the gyroscope type-  
<https://developer.android.com/reference/android/hardware/Sensor#TYPE_GYROSCOPE>

6. Now, we need an event listener for the sensor. It will have onSensorChanged. When the listener is not null, then it takes in the rotation values. Here is the what the constructor includes-

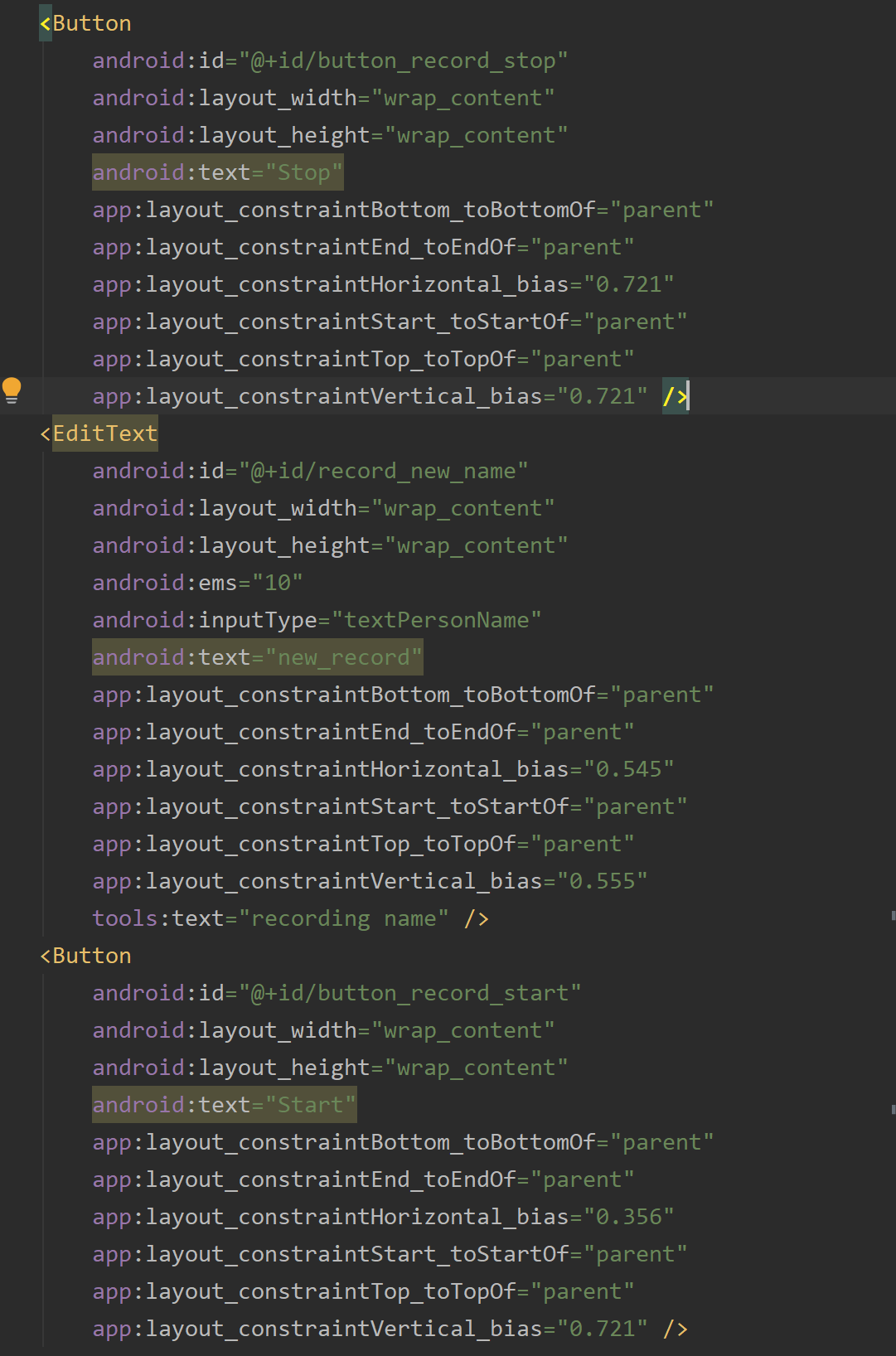
7. For all sensors, we need to register them. Here, we can call a method that will register for us when invoked. For the registerListener, we pass in the event listener, the sensor object and the delay to normal.  
  


8. Similar to registering, we also need to unregister the sensor, or it will be always active. We use another method for it.  


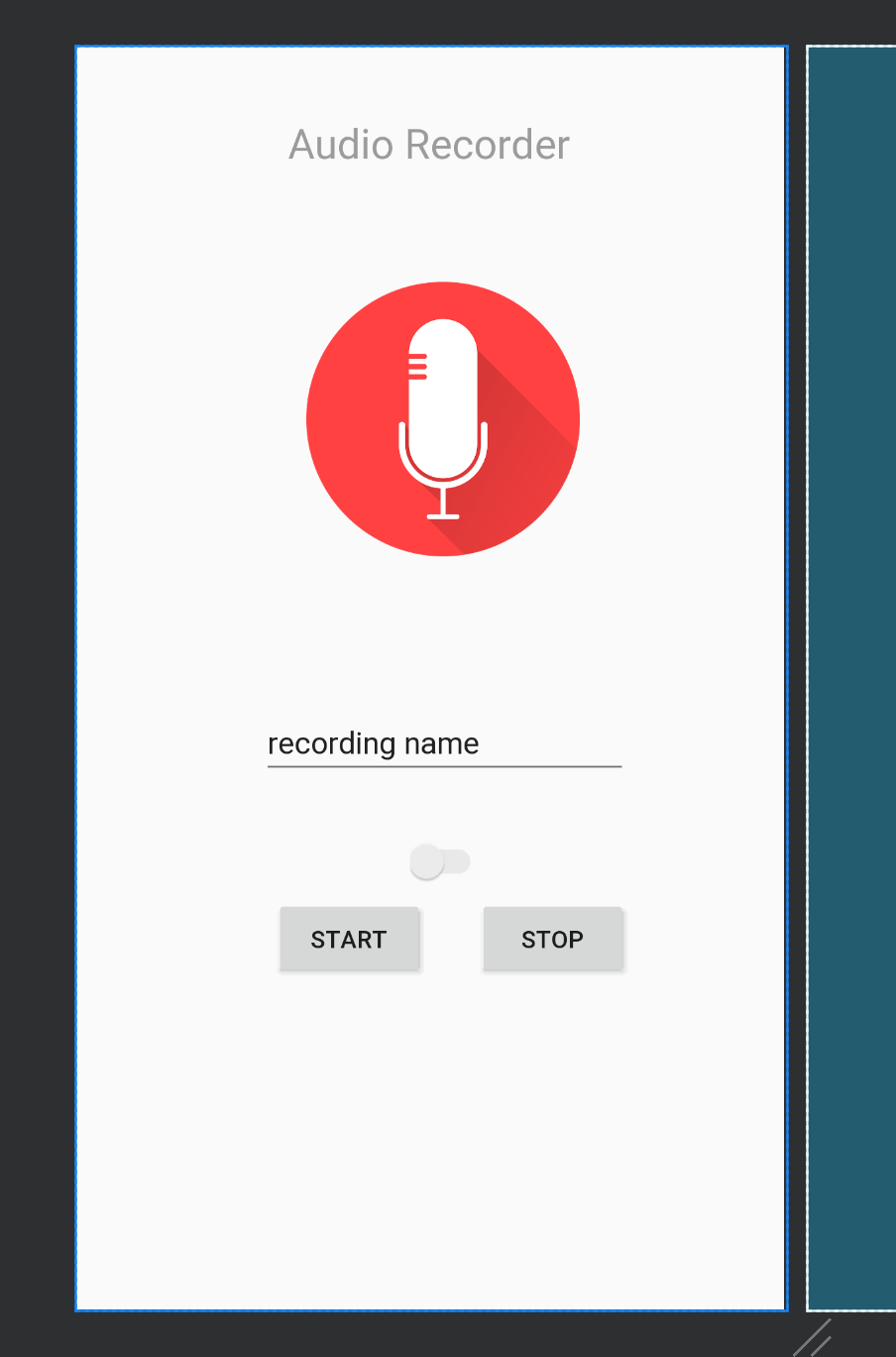
9. Here is a final look at the gyroscope sensor class that we made-  
  

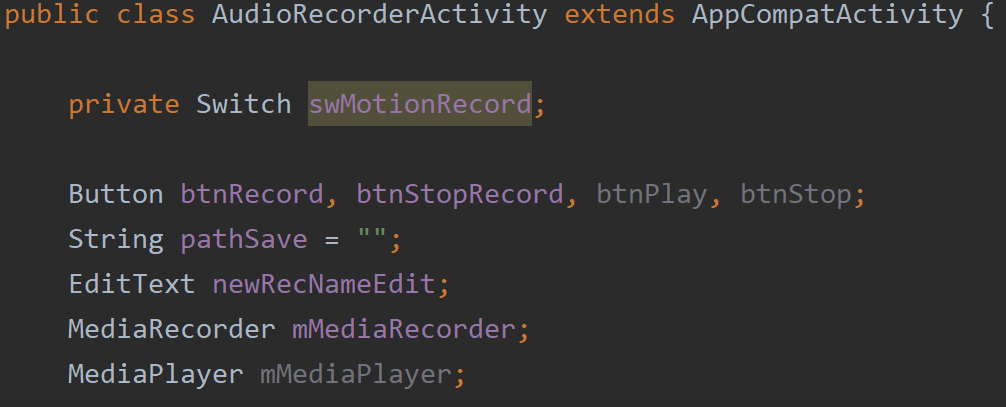

10. Now, let us make the UI for the voice recorder. It is a simple UI just to show the voice is being recorded and the motion is enabled or not. I have an image view and a text view for design purposes.  
  


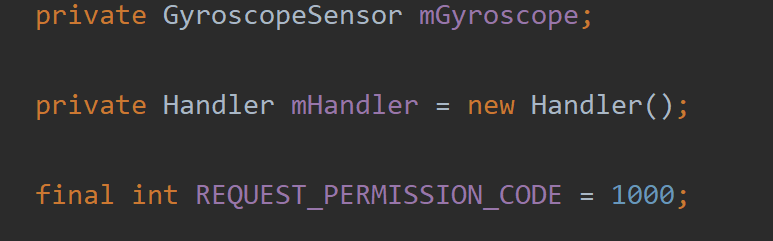
11. I have included a switch. Once flicked, this enables the gyroscope to record audio.  


12. There are two buttons, one for starting to record and stopping. These are just to show that the voice recorder works. There is also an edit text which is used to name the new recordings.  
  


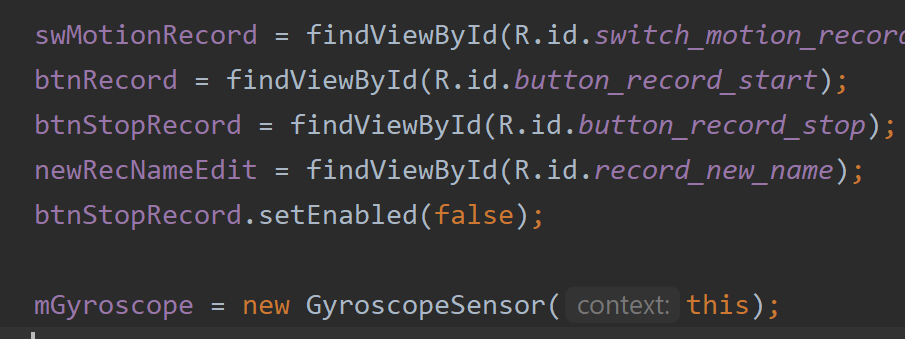
13. here’s the look of the UI screen-

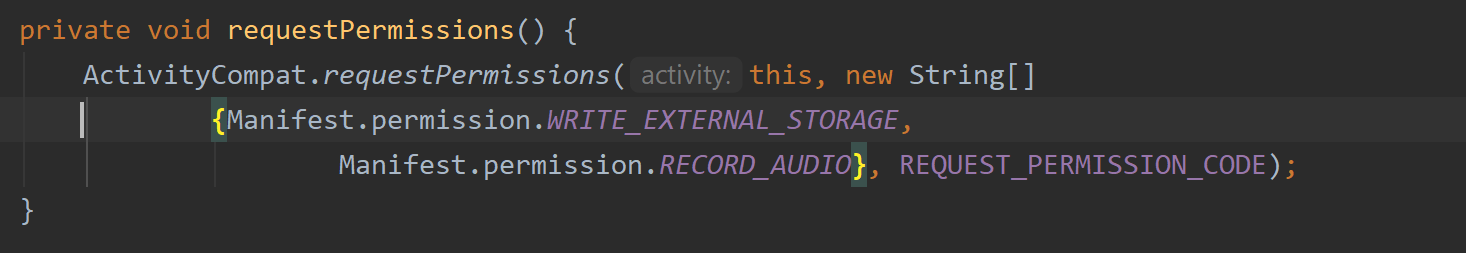


14. Now, let us go to the AudioRecorder activity. Add the following objects-  
  


15. We need a gyrsoscope object, a handler object and an int variable to handle permission code-

16. Inside our onCreate method, find the views and initiate the gyroscope object-



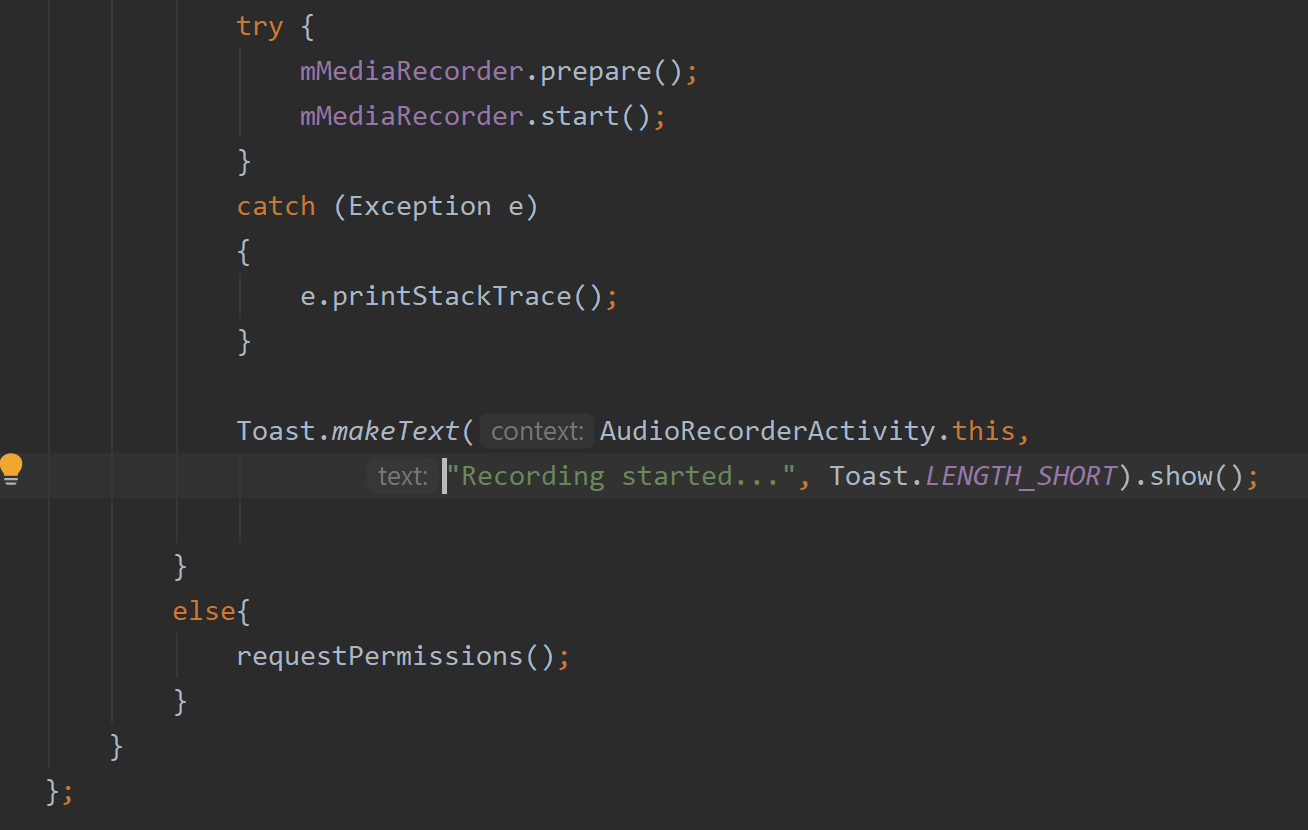
17. Now we will need some methods out side onCreate to handle permissions. Call a method to request permissions-  
  


18. Implement onRequestPermisisonsResult method-  
  


19. Lastly, the check permissions from device method-

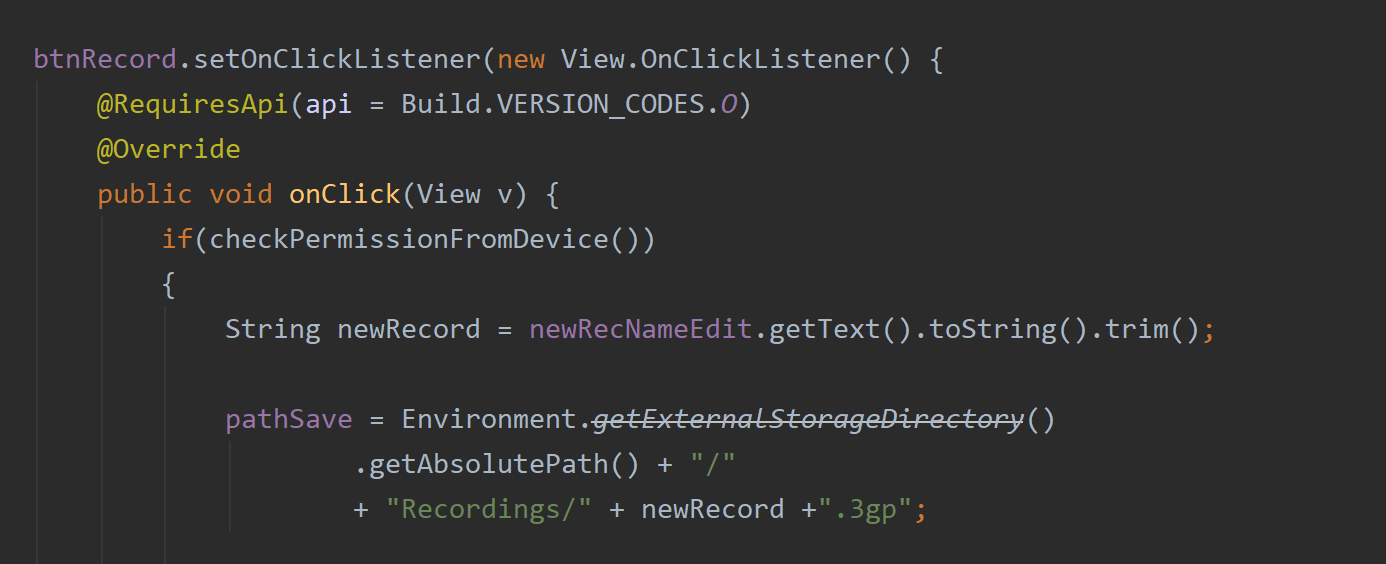


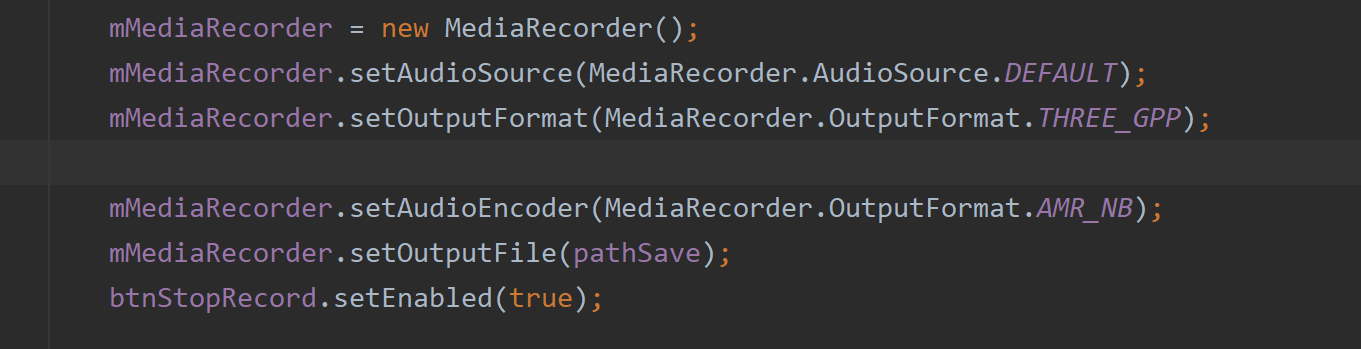
20. We will use runnable to start the voice record and save it on the device-  

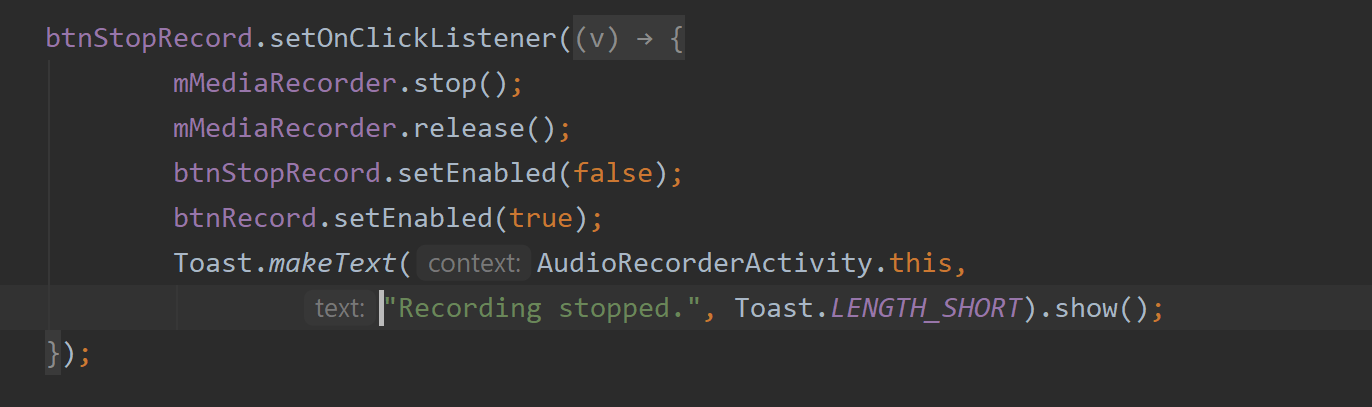
The recording will be saved in that path above. We will start the recording here after we check the permissions using the methods. Its best to put the prepare() method of the media player inside a try catch.

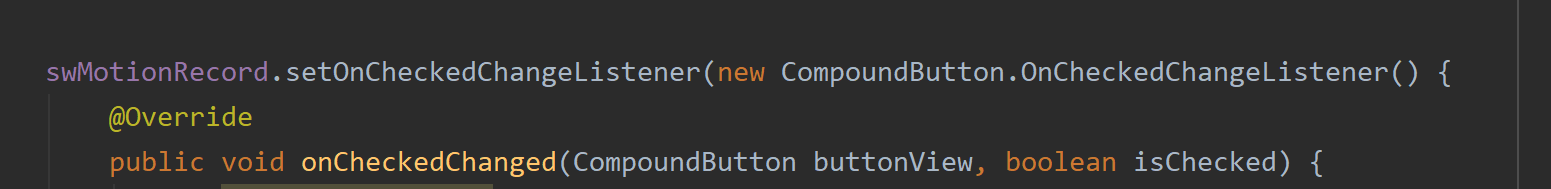
21. Let us set the buttons to for the recorder start and stop. Inside onCreate, set the onCLick Listener for btnRecord. Clicking this button will start the recording. We check permisisons with an if statement. If there is permission, then we get the name of the recording from the user. Then we set the path where it will be recorded.



22. We initiate the media recorder object from the media recorder class. And set the audio source to default and format to 3gp. The output format would be AMR.NB and set the path. The button to stop record ws enabled here so that we can stop the recording-  
  


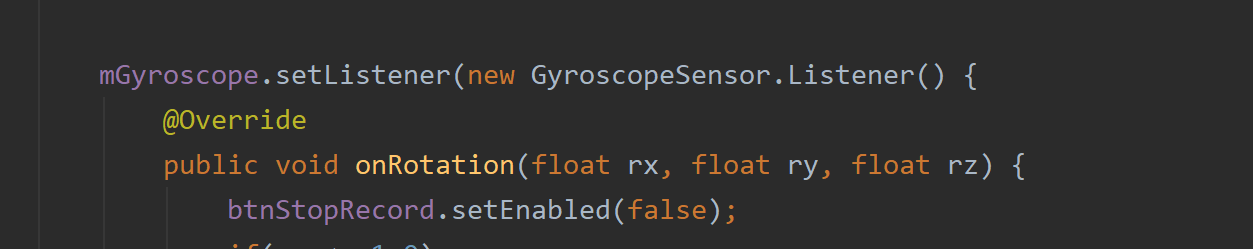
23. We put the prepare and start methods inside a try catch. You can show a toast message when the recording starts. Lastly, the else statement is there in case permission is not given-  


24. Here we code for the stop button-  


25. Now, our recorder works and is able to save audio in storage. We now need to add the gyroscope function. First, we use the switch button and set the OnCheckedChangeListener-  
  


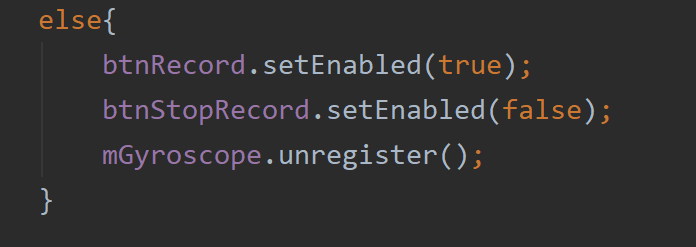
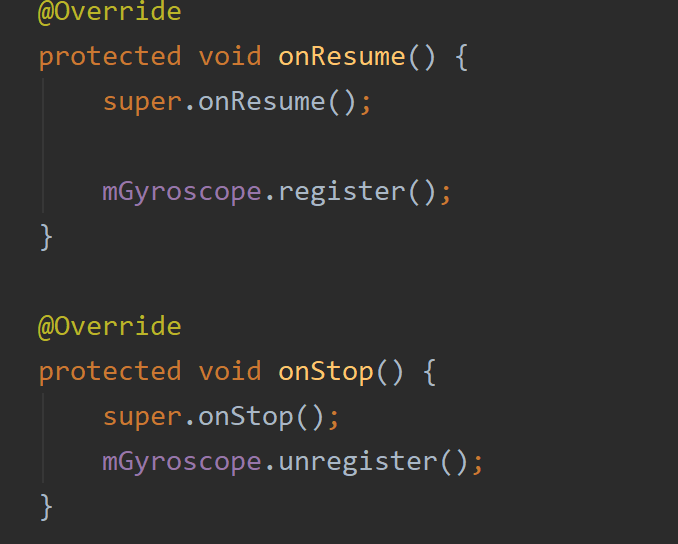
26. Inside onCHecked Change, we need to check if the switch is flicked. If it is true, then we set the start and stop buttons to false because we will only use the gyroscope sensor to start recording in this state. We will need to initiate the gyroscope sensor and register it here.



27. We are still within the onCheckedChange. Now, we need to call the setListener for the gyroscope. Here, the onRotation is implemented that takes in the x, y and z values from the gyroscope inside the device.  
  


28. We now check for rotations. The two values that determine rotation are 1.0f and -1.0f. We use the x values (float rx) here for the tilts we need. You can use y and z values also but you need to ensure which kind of tilt alters these values.  
  
When the rx < -1.0f, then the recording will start. It will generate a beep letting the user know that the gyroscope has tilted and recording will start-  

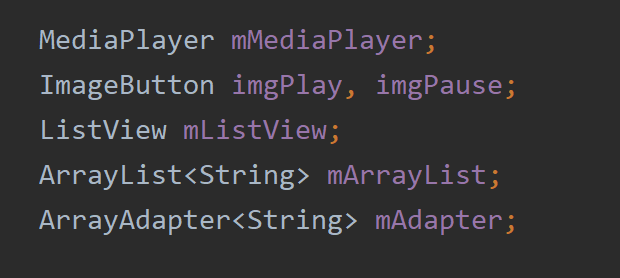

29. Again, when the phone gets back to the previous position (rx > 1.0f), then the recording will stop. This means that the gyroscope has sensed that the phone tilt has changed again.  

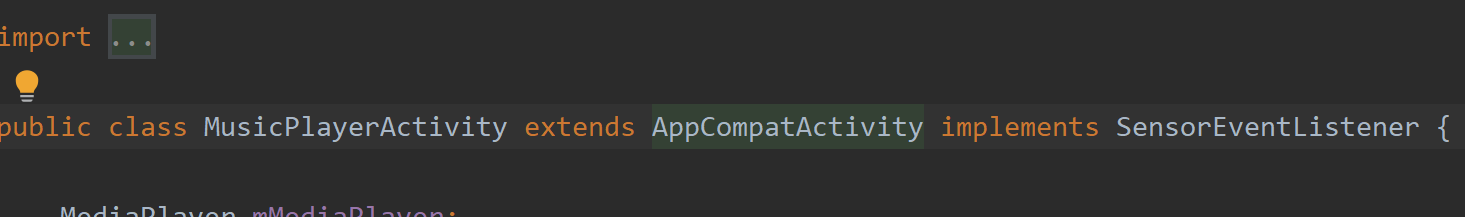

30. This is the else statement of the switch flick. We unregister the sensor so that it is no longer active when we don’t want it to be-  
  
  
31. You can se the demo to be clear which positions activate the gyroscope based on this tutorial. We also need to unregister the sensor in onStop and register in onResume.  


32. Make sure to use proper button handling when using sensors to prevent exceptions.  
  
  
**Proximity Sensor – Controlling Audio Play/Pause**

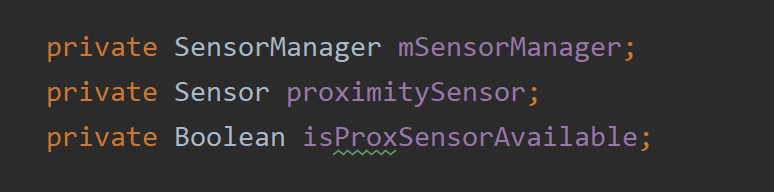
The proximity sensor is an infrared sensor usually placed on the topmost part of the phone’s front side. It measures obstructions at a certain distance and this distance is different for each phone. I tested the app on a Samsung S9 and the value is 8 when there is nothing in front of it. When something is in front of the phone within this range, the value becomes 0.0, meaning there is something in front. We will use this sensor to control audio play and pause.

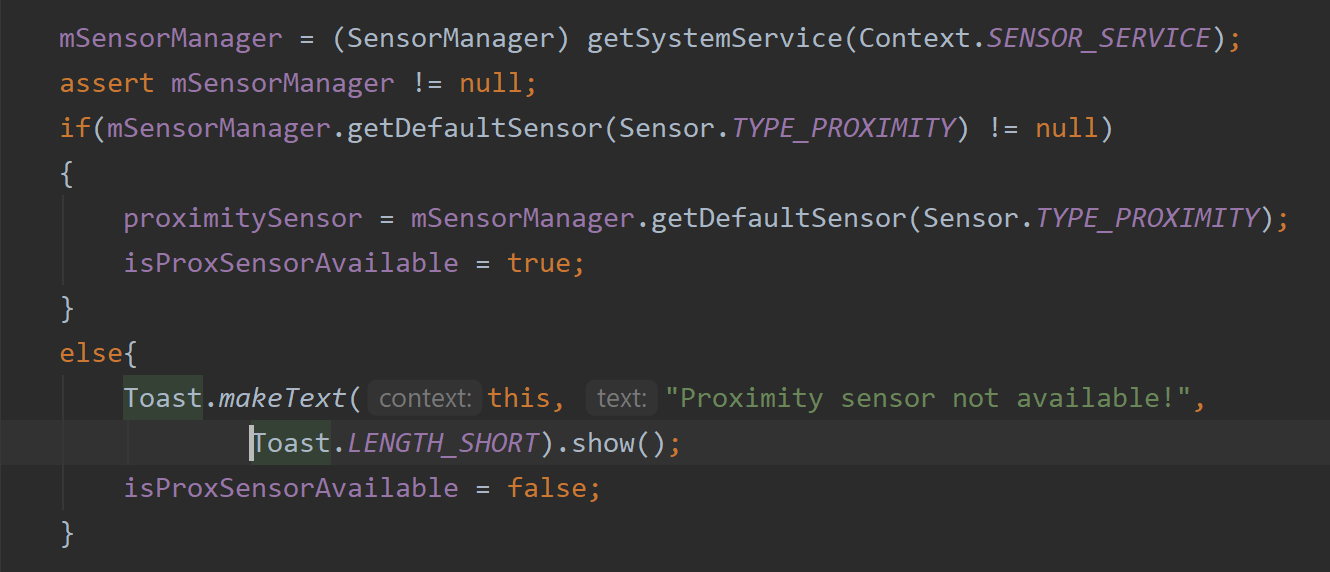
\*\*\*Works on APK 29 and on Android 9. I used a Samsung S9.

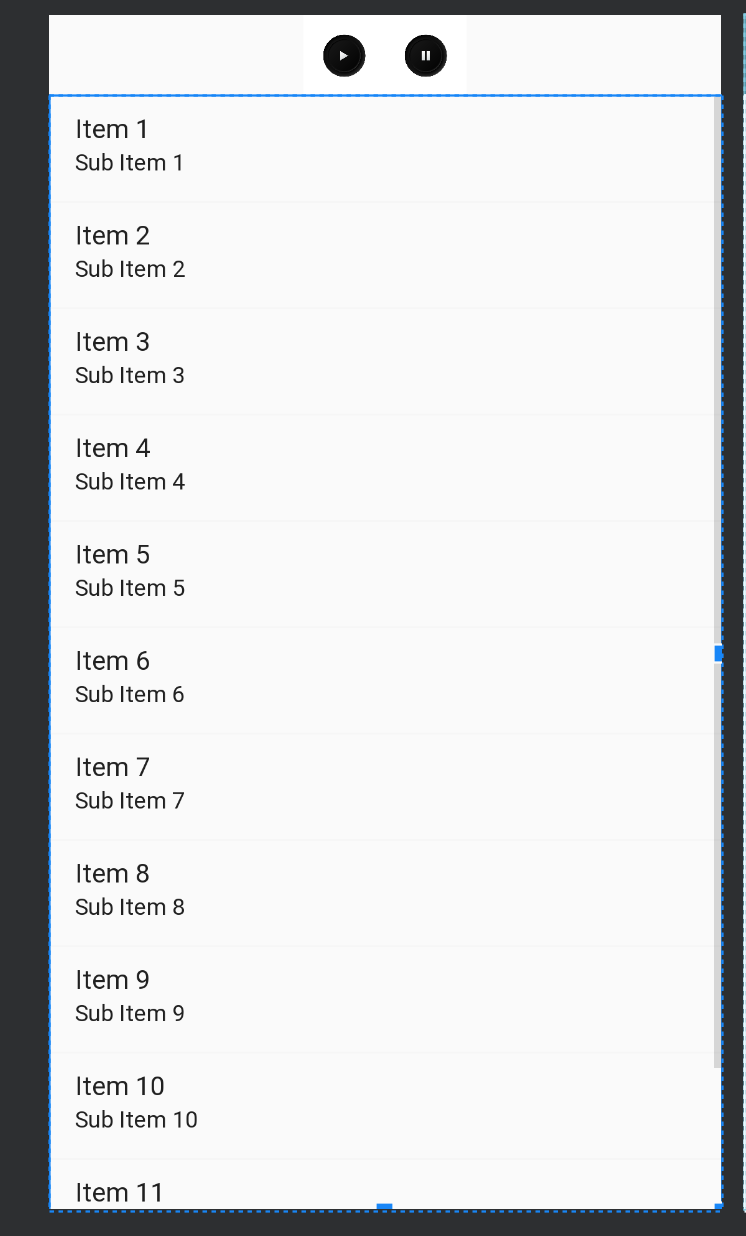
1. This time, we will not create a class for this sensor. We will use the mediaplayer class to build a media player. Declare the objects needed as shown-  


2. The name of this activity will be “MediaPlayerActivity”. This should implement SesnorEventListener-  
  


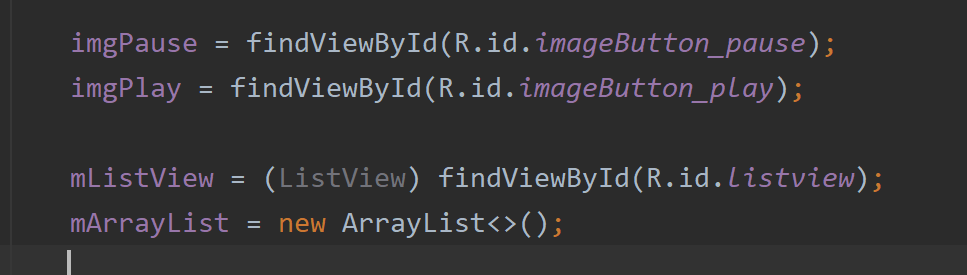
3. We will also need a sensor object. Let’s call it proximitySensor. Declare a sensor manager object and a Boolean variable to check for this sensor’s availability in the device-



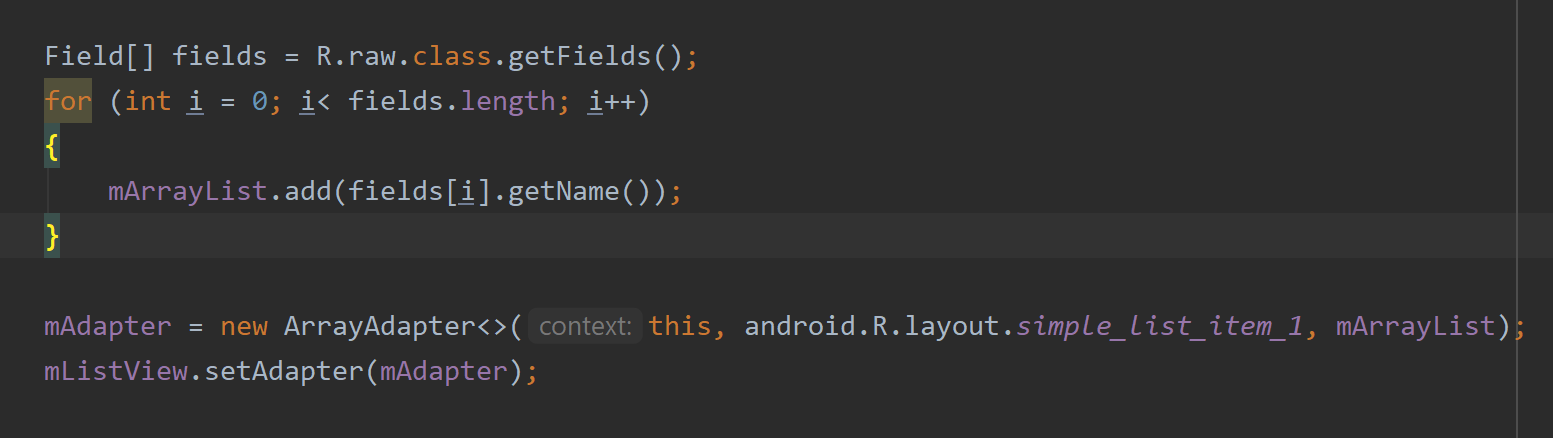
4. Inside our onCreate, we first do a check for this sensor’s availability in our device-  


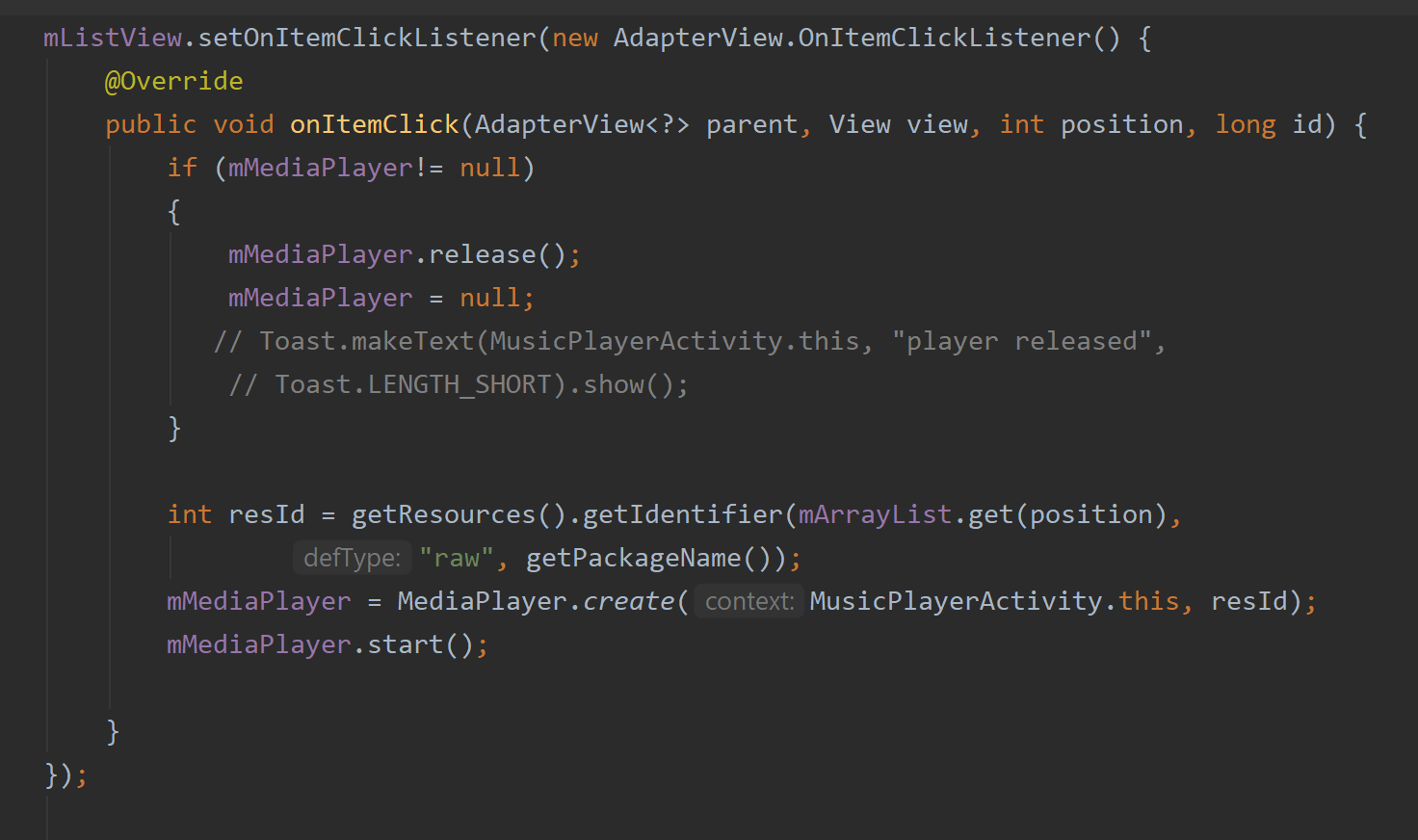
5. We will now make the media player that will play / pause songs. I made a simple UI that displays the songs in a listview and has 2 buttons-  
  


6. The buttons are imageButtons. Find the views-



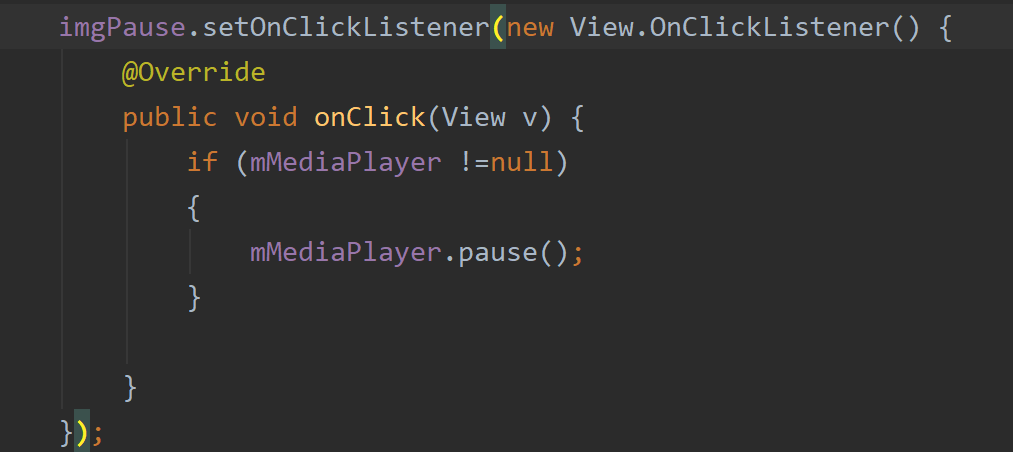
7. I stored few songs in the ‘raw’ folder of the resources. If it’s not there, then you will have to create the raw folder in the res section and copy paste your audio. Make sure to name them in small letters with no spaces. You can also try to get songs from the storage. But for the purposes of this demo, I used a simple method. We will now get the files from the raw folder and populate the listview-

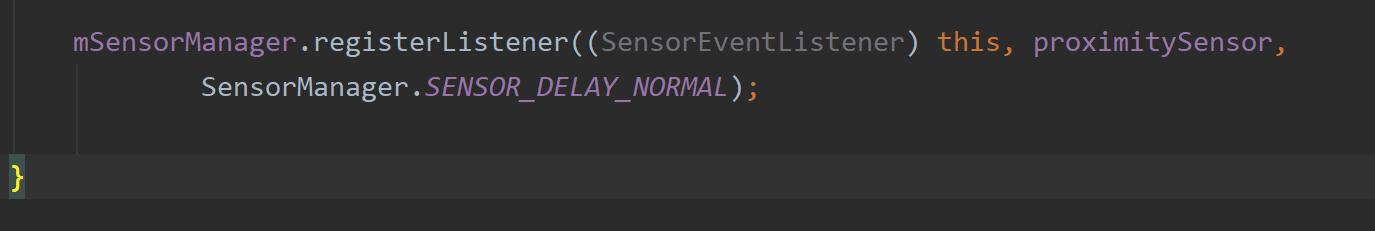


8. We will now set the list item click and once a song is clicked, it will start playing. We get the files from raw and pass them through a variable (resId) to the media player class. Its important to check for null for the media player object.   
   


9. Now, to code the imgPlay and imgPause buttons so that play and pause works-





We will need to register the sensor in the onCreate. We pass the context, the proximity sensor object and set the delay to normal-  


10. Our method of concern is the onSensorChanged. We can now manipulate pause and play through this method. When play button is clicked, the song can be paused using the proximity sensor and then resumed again by it. It is put inside a try-catch block. I put a toast message here to show the values changing when something close is placed in front of the phone. The onSesnorCHanged method requires SensorEvent object. It listens for any change in the value. For the device, my values were 8.0 (nothing on front) and 0.0 (obstruction detected). We can manipulate actions, by using these changes in values. So, here, when object is detected, media player will be paused if playing. Again, if object is detected during pause, the player will resume-



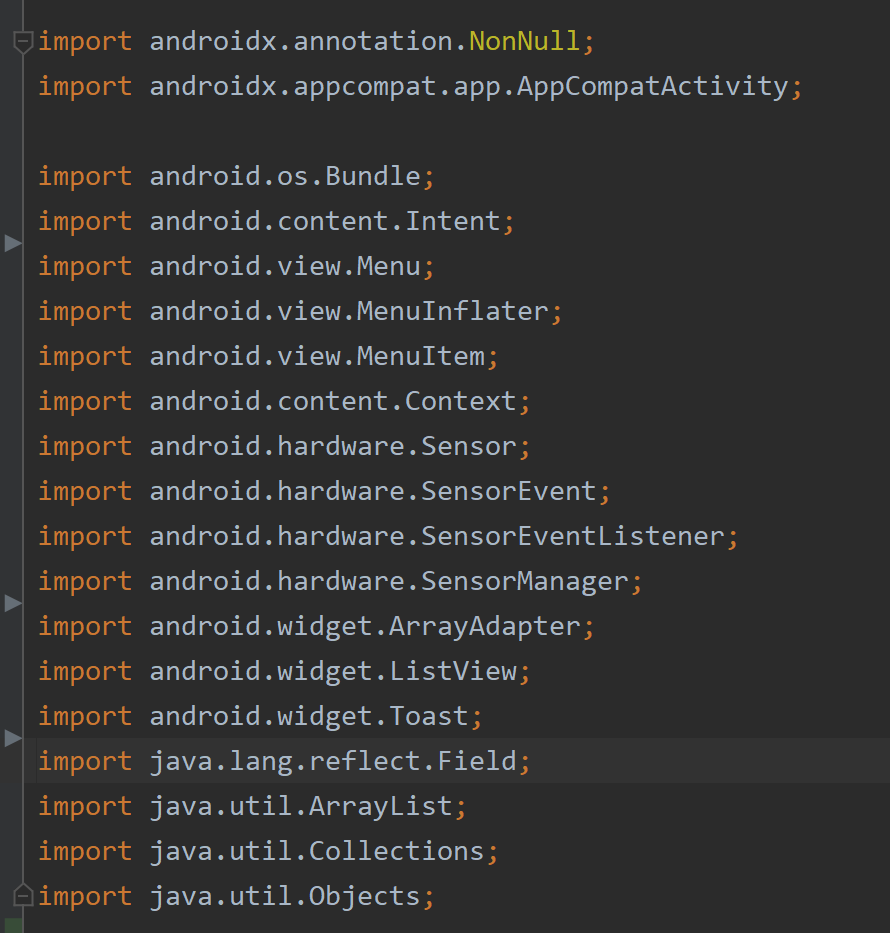
11. We then register at onResume and unregister at onStop-

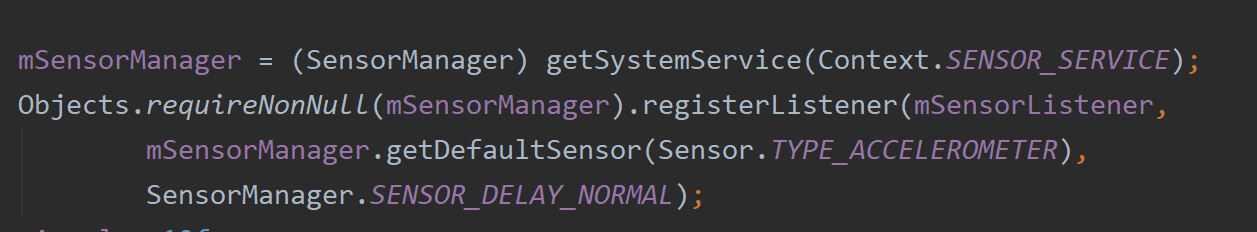


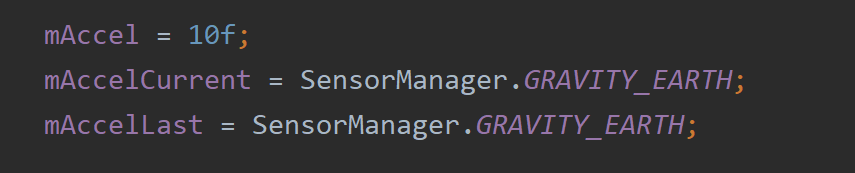
12. Ensure what the values are for your own device and if it has a proximity sensor.

**Accelerometer and Gravity Sensors – Shake Detection**

For this next part in the tutorial, I will demonstrate how to use a composite sensor. This means that two or more base sensors will feed data to be used. Here, I have used an Accelerometer and the Gravity sensor. We will use these sensors to detect a shake event. We will then use it to shuffle the playlist from our raw folder.

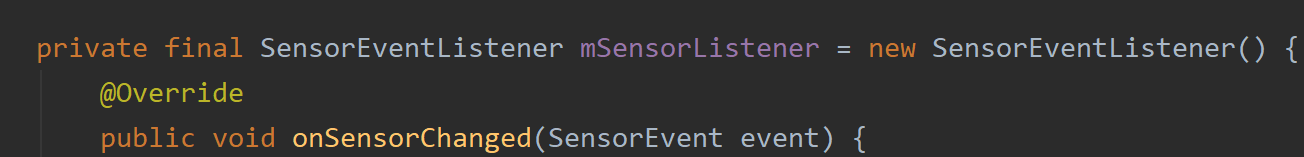
Here are the imports for this activity-  


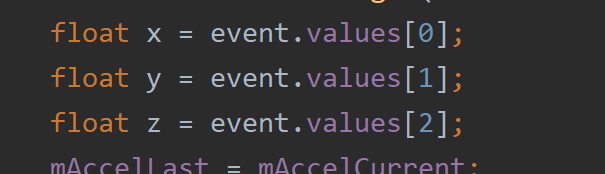
1. We will start a new activity called ShakeDetectionActivity. First, declare the float variables to store sensor data for all axis and a sensormanager object. We need a sensor service for this and define the sensor type, sensor listener and set the delay, as shown-  
  


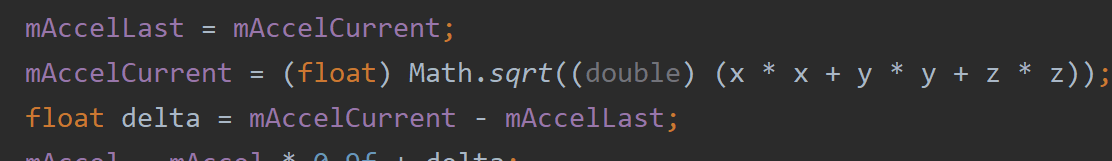
2. We now set the values for the variables for the two sensors. The gravity sensor will take two values, the last one and the current one-  


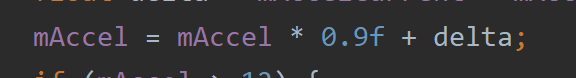
3. We will use a listview to populate from the songs in the raw folder.   
  

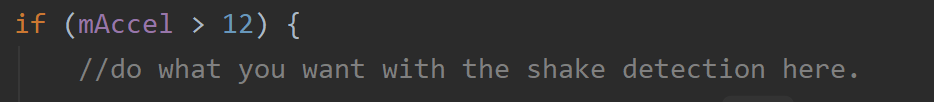

4. Outside the onCreate, we implement the SensorEventListener which overrides the onSensorChanged-

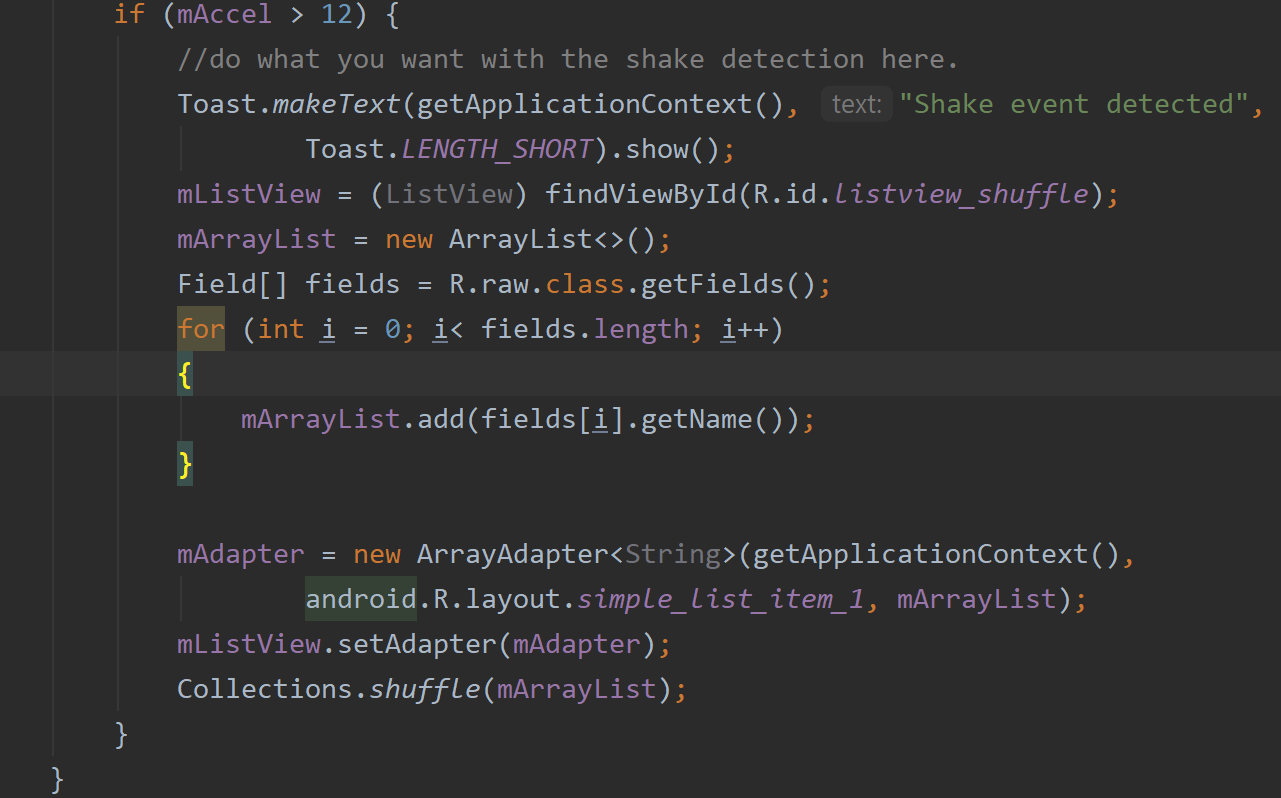


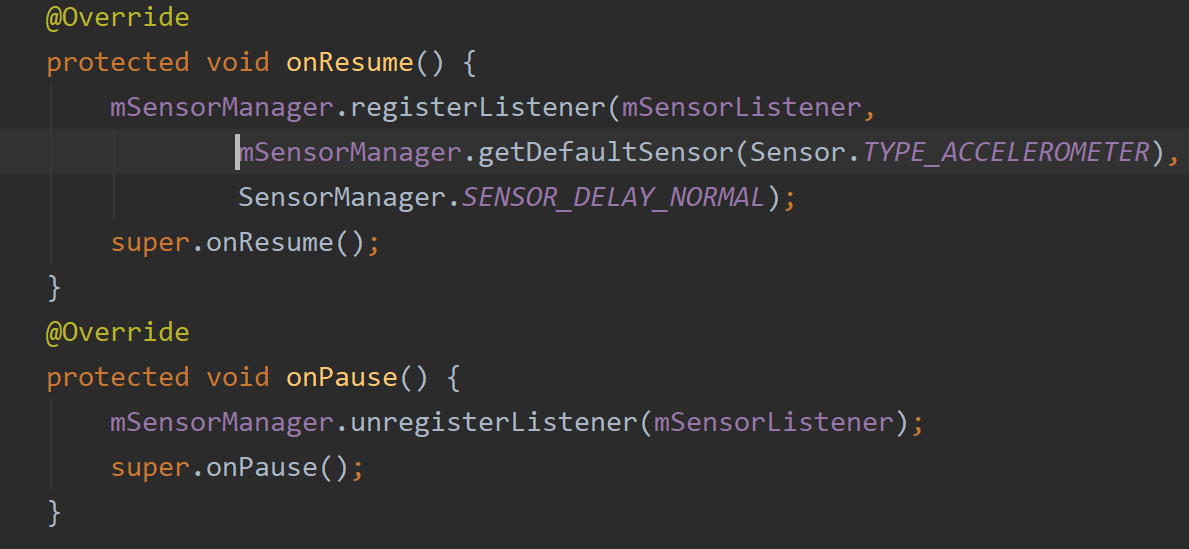
5. Inside onSensorChanged, first we initiate the float values. These are the values of the event along the 3 axes.   
  


6. We now set the last values as the current values at first. The current value will change according to the formula. And we add a floar variable that takes the difference between the first and last values-  


7. We will need to set the values for mAccel. This is important because the change anticipated for this value will determine if a shake even is present-  


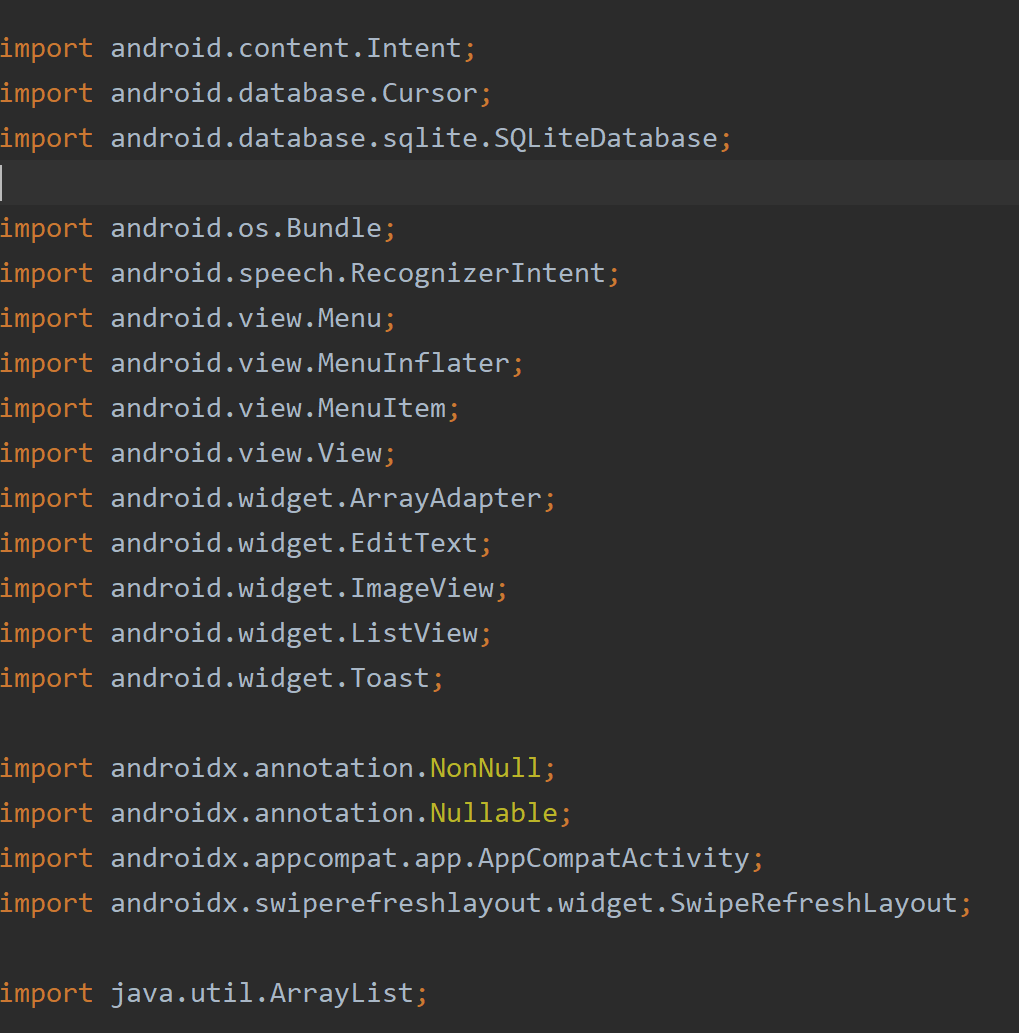
8. When the value becomes greater than 12, it will be regarded as a shake event-  


Inside this condition, we can perform different actions based on this change. In this case, it was the list of songs from the playlist. Using Collections.shuffle() and passing the arraylist will randomize the list after the shake is detected. 

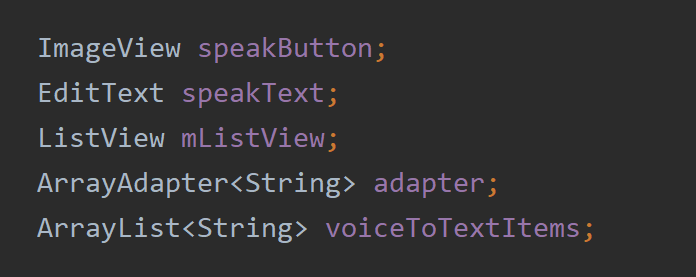
9. We register it on resume and unregister at onPause-  


**Topic 2: Voice Recognition using Google API**

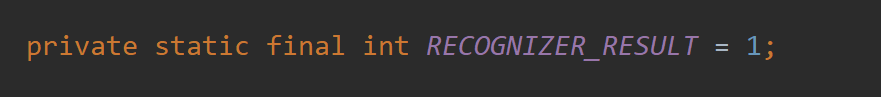
My second topic was voice recognition from a google API. Using this API, we can implement a voice recognition functionality. This can be used to make a voice to text app that takes verbal instructions from the user and performs tasks. For demo purposes, Ithe app I made is a voice to text app that takes notes and turns them into a list.  
  
Here are imports for this part-

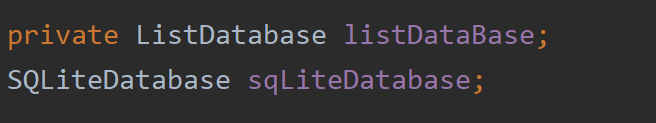


1. I used this functionality in the MainActivity. Declare some objects. We have an edit text view that will display the current words spoken by the user-



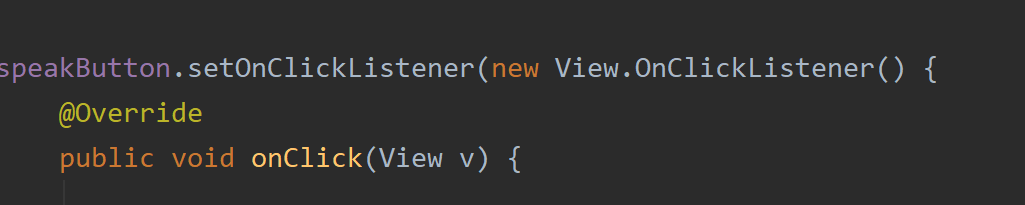
2. We will need a integer constant set to one to be used for the result of the voice match-



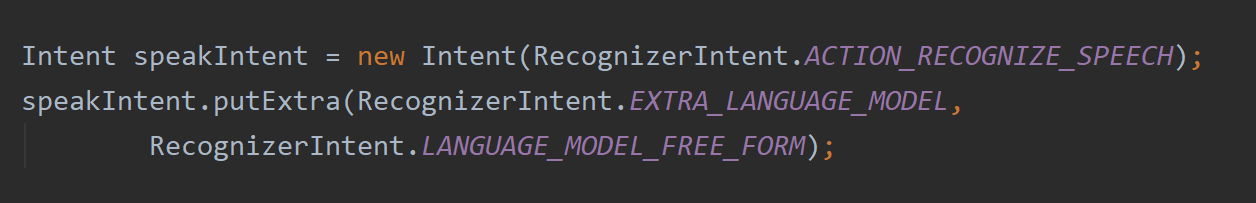
3. We are going to store the lists in SQLite so we need database objects-  


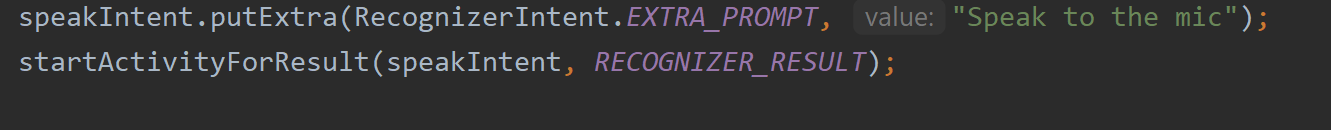
4. I have created an SQLite database class for the list. It has one table, insert and retrieve data methods. The demo for SQLite is not shown in this tutorial.

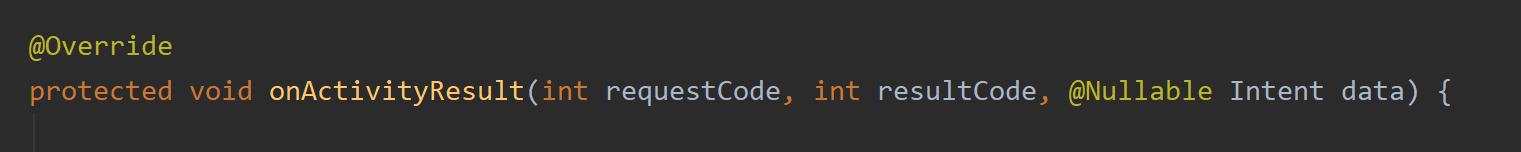
5. Inside onCreate, after finding the vies, we call a cursor object to retrieve the list and add the items to our array list to display data-  


6. When the user presses the speakButton, it will start the google API and a prompt will ask the user to speak to the mic. We need a listener for the button-  


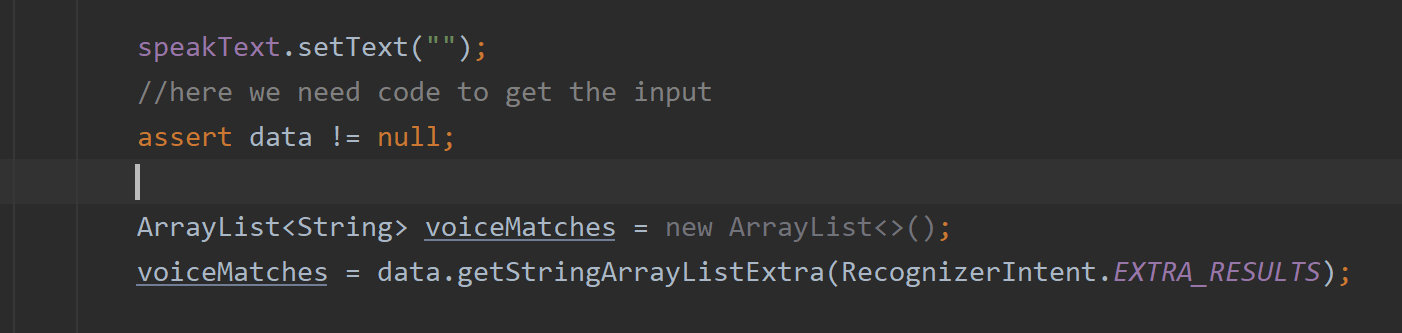
7. Now, inside the listener onCLick, we will use an intent to activate the recognition. It will be a RecognizerIntent that has ACTION\_RECOGNIZE\_SPEECH as parameter. The documentation can be found here-  
<https://developer.android.com/reference/android/speech/RecognizerIntent>

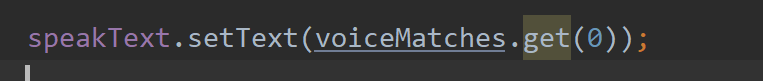
  
  
The extra language model informs the recognizer which speech model to access. Here, it is the free form model for speech recognition.

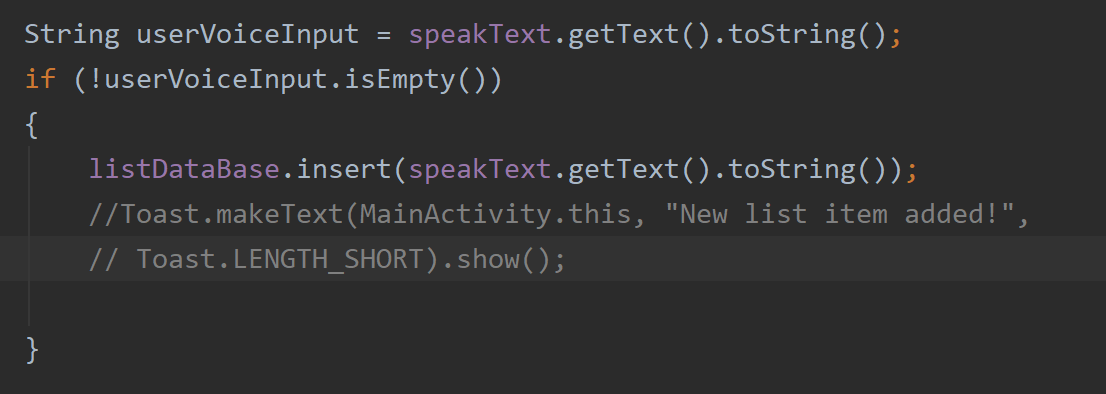
8. We need a prompt to tell the user when to speak-  
  


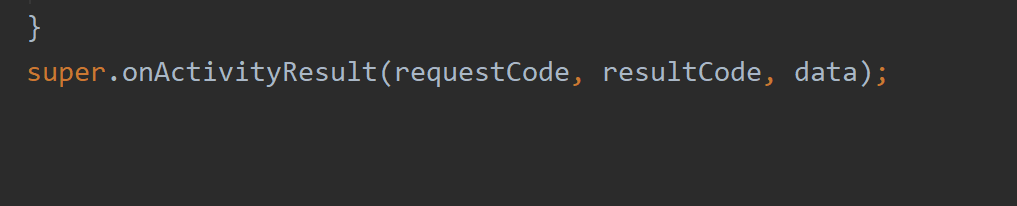
9. Implement onActivityResult outside the onCreate.-  


10. Inside the onActivityResult, we initiate the database object here. Now, we need to check the requestcode. If the condition prevails for the requestcode then perform the retrieval of the text from the voice. We set the speakText edit text to empty to remove any previous input. We then delare an arraylist that will store the retrieved text after voice recognition-



11. Set the recognized text-  


12. We now insert the words into the database-

13. The super call of this method is at the very end.

It should now be able to utilize the voice recognition API from Google and store the texts in a database.