**package** com.test02.illid.audio\_capture02;  
  
**import** android.media.AudioFormat;  
**import** android.media.AudioRecord;  
**import** android.media.MediaRecorder;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.os.AsyncTask;  
**import** android.widget.TextView;  
**import** org.jtransforms.fft.\*;  
**import** java.util.ArrayList;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 **private int sample\_rate** = 16000;  
 **private int fft\_size** = 4096;  
 **private int buffer\_size** = **fft\_size** / 2;  
 **private float mf** = 16384f;  
 *//private AudioRecord ard = new AudioRecord(MediaRecorder.AudioSource.MIC, sample\_rate, AudioFormat.CHANNEL\_IN\_MONO, AudioFormat.ENCODING\_PCM\_16BIT, buffer\_size);* **private** AudioRecord **ard** = **new** AudioRecord(MediaRecorder.AudioSource.***VOICE\_RECOGNITION***, **sample\_rate**, AudioFormat.***CHANNEL\_IN\_MONO***, AudioFormat.***ENCODING\_PCM\_16BIT***, **buffer\_size**);  
 **private boolean isrecording** = **false**;  
 **private** ArrayList<Integer> **freq\_final** = **new** ArrayList<Integer>();  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **final** Button button1 = (Button) findViewById(R.id.***button1***);  
 **final** Button button2 = (Button) findViewById(R.id.***button2***);  
 **final** Button button3 = (Button) findViewById(R.id.***button3***);  
 **final** TextView text1 = (TextView) findViewById(R.id.***textView1***);  
  
 *//buffer\_size = AudioRecord.getMinBufferSize(sample\_rate, AudioFormat.CHANNEL\_IN\_MONO, AudioFormat.ENCODING\_PCM\_16BIT);  
 //Log.d("ACCG", "buffer size = " + buffer\_size);  
 //buffer\_size = fft\_size / 2;* button1.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if** (**ard**.getState() == AudioRecord.***STATE\_INITIALIZED***) {  
 Log.*d*(**"DebugMainActivity"**, **"start"**);  
 }**else**{  
 *//ard = new AudioRecord(MediaRecorder.AudioSource.MIC, sample\_rate, AudioFormat.CHANNEL\_IN\_MONO, AudioFormat.ENCODING\_PCM\_16BIT, buffer\_size);* **ard** = **new** AudioRecord(MediaRecorder.AudioSource.***VOICE\_RECOGNITION***, **sample\_rate**, AudioFormat.***CHANNEL\_IN\_MONO***, AudioFormat.***ENCODING\_PCM\_16BIT***, **buffer\_size**);  
 Log.*d*(**"DebugMainActivity"**, **"start\_no\_ard"**);  
 }  
 **isrecording** = **true**;  
 **ard**.startRecording();  
  
 **new** backAStask().execute();  
 Log.*d*(**"DebugMainActivity"**, **"AsyncTask"**);  
 }  
 });  
 button2.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if** (**ard**.getState() == AudioRecord.***STATE\_INITIALIZED*** && **isrecording**) {  
 Log.*d*(**"DebugMainActivity"**, **"stop"**);  
 **ard**.stop();  
 **ard**.release();  
 *//new backAStask().cancel(true);* **isrecording** = **false**;  
 Integer sum = 0;  
 **for** (**int** i = 0; i < 5; i++) {  
 **freq\_final**.remove(0);  
 }  
 **for** (Integer i : **freq\_final**) {  
 sum += i;  
 }  
 Log.*d*(**"DebugMainActivity"**, **"sum : "** + sum);  
 sum = sum / **freq\_final**.size();  
 Log.*d*(**"DebugMainActivity"**, **"freq\_final size: "** + **freq\_final**.size());  
 Log.*d*(**"DebugMainActivity"**, **"stop final freq: "** + sum);  
 **freq\_final**.clear();  
 text1.setText(sum.toString());  
 } **else** {  
 Log.*d*(**"DebugMainActivity"**, **"stop\_no\_ard"**);  
 **freq\_final**.clear();  
 }  
 }  
 });  
 button3.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 text1.setText(**"Frequency"**);  
 }  
 });  
 }  
  
  
 **private class** backAStask **extends** AsyncTask<Void, **short**[], Void> {  
 @Override  
 **protected** Void doInBackground(Void... params) {  
 *//Log.d("debugdoInBackground", "beforewhile");* **while** (**isrecording**) {  
 **try** {  
 Thread.*sleep*(100);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **short**[] buffer = **new short**[**buffer\_size**];  
 *//Log.d("debugdoInBackground", "insidewhile");* **ard**.read(buffer, 0, **buffer\_size**);  
 publishProgress(buffer);  
 }  
 **return null**;  
 }  
  
 @Override  
 **protected void** onProgressUpdate(**short**[]... buffer){  
 *//System.out.println(Arrays.toString(buffer));  
 //Log.d("debugOnProgressUpdate", "audio data: " + buffer.toString());* **super**.onProgressUpdate(buffer);  
 *//for (Short var: buffer[0]){  
 //Log.d("debugOnProgressUpdate", "audio data: " + var.toString());  
 //}* **double**[] trans = **new double**[**fft\_size**];  
 **double** temp = **mf** \* **mf** \* **fft\_size** \* **fft\_size** / 2d;  
 **double**[] mag = **new double**[1024];  
 **for** (**int** i = 0; i < **fft\_size** / 2; i++){  
 trans[2\*i] = (**double**)buffer[0][i];  
 trans[2\*i+1] = 0;  
 }  
 DoubleFFT\_1D fft = **new** DoubleFFT\_1D(**fft\_size** / 2);  
 fft.complexForward(trans);  
 *// convert to db* **for** (**int** i = 0; i < mag.**length**; i++){  
 mag[i] = 5.0\*Math.*log10*((trans[i] \* trans[i] + trans[i+1] \* trans[i+1]) / temp);  
 }  
  
 **double** max = -100;  
 **int** max\_index = 0;  
 *//int start = (int)Math.round(200\*fft\_size/sample\_rate);* **for** (**int** i = 0; i < mag.**length**; i++){  
 *//trans[i] = Math.abs(trans[i]);* **if** (max < mag[i]){  
 max = mag[i];  
 max\_index = i;  
 }  
 }  
 **int** freq = max\_index \* **sample\_rate** / **fft\_size**;  
 *//Log.d("debugOnProgressUpdate", "first freq: " + freq);* **if** (**sample\_rate** / **fft\_size** < freq && freq < **sample\_rate**/2 - **sample\_rate** / **fft\_size**) {  
 **int** id = max\_index;  
 **double** x1 = trans[id-1];  
 **double** x2 = trans[id];  
 **double** x3 = trans[id+1];  
 **double** c = x2;  
 **double** a = (x3+x1)/2 - x2;  
 **double** b = (x3-x1)/2;  
 **if** (a < 0) {  
 **double** xPeak = -b/(2\*a);  
 **if** (Math.*abs*(xPeak) < 1) {  
 freq += xPeak \* **sample\_rate** / **fft\_size**;  
 max = (4\*a\*c - b\*b)/(4\*a);  
 }  
 }  
 }  
 freq = Math.*round*(freq);  
 *//Log.d("debugOnProgressUpdate", "maxDB: " + max);* Log.*d*(**"debugOnProgressUpdate"**, **"frequency: "** + freq);  
  
 **freq\_final**.add(freq);  
 }  
 }  
  
}