NAME: SAKSHI VINAYAK KITTURKAR
CSU ID: 2860273
HOMEWORK 17

Build a sound level detection app
Extension of the audio capture app, do not save audio
Use maximum amplitude, and display the sound level graphically using line chart or bar chart
https://developer.xamarin.com/api/property/Android.Media.MediaRecorder.MaxAmplitude/

= I have built a sound level detection application where extension of the audio capture app do not save audio and uses maximum amplitude and display the sound level graphically using line chart.

In layout: This code appears to be a simple Android app that records audio using the device's microphone, periodically updates the amplitude value, and visualizes it in a real-time graph. The graphing library used here seems to be GraphView.

These are class-level variables. RANGE seems to be a constant value, and there are variables for managing graph data (mSeries1, graph2LastXValue, graph), requesting record permissions (REQUEST_RECORD_PERMISSION), handling media recording (mediaRecorder), and displaying amplitude (amplitudeTextView).

```
private static int RANGE = 4;
private LineGraphSeries mSeries1;
private long graph2LastXValue = 0;
GraphView graph;
private static final int REQUEST_RECORD_PERMISSION = 100;

private MediaRecorder mediaRecorder;
private TextView amplitudeTextView;
```

The onCreate method is called when the activity is first created. It sets the content view to the layout defined in activity_main.xml.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

Checks if the app has the RECORD_AUDIO permission. If not, it requests the permission from the user. If granted, it proceeds to initialize the MediaRecorder.

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO)
        != PackageManager.PERMISSION_GRANTED) {
    // Permission is not granted, request it
    ActivityCompat.requestPermissions(this,
            new String[]{Manifest.permission.RECORD_AUDIO},
            REQUEST_RECORD_PERMISSION);
} else {
    // Permission is already granted, proceed with initialization
    initializeMediaRecorder();
}
```

Initializes the MediaRecorder for audio recording, sets up its configuration, prepares it, and starts recording. It also schedules periodic updates of the amplitude on the UI.

```
void initializeMediaRecorder() {
    mediaRecorder = new MediaRecorder();

    // ... (mediaRecorder configuration)

    try {
        mediaRecorder.prepare();
        mediaRecorder.start();
        // ... (periodic amplitude update on the UI)
    } catch (Exception e) {
        Log.e("MainActivityyyy", "Exception : " + e.getMessage());
    }
}
```

Updates the amplitude on the graph using the MediaRecorder's getMaxAmplitude() method. It appends the new data point to the graph series.

```
void updateAmplitude() {
    if (mediaRecorder != null) {
        int amplitude = mediaRecorder.getMaxAmplitude();
        mSeries1.appendData(new DataPoint(mSeries1.getHighestValueX() + 1, amplitude), true,
100);
    }
}
```

Handles the result of the permission request. If the RECORD_AUDIO permission is granted, it proceeds with the initializeMediaRecorder method; otherwise, it can handle the denial of permission.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    // ... (permission result handling)
}
```

Stops and releases the MediaRecorder when the activity is destroyed to clean up resources.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if (mediaRecorder != null) {
        mediaRecorder.stop();
        mediaRecorder.release();
    }
}
```

In layout: ExampleInstrumentedTest.java This code is an example of an instrumented test for an Android application. It is specifically designed to run on an Android device or emulator, interacting with the Android runtime environment.

This method is a test case named useAppContext. Test cases in JUnit are methods annotated with @Test. The purpose of this specific test is to check whether the app's context has the expected package name.

```
@Test
public void useAppContext() {
```

It retrieves the application context using InstrumentationRegistry. The InstrumentationRegistry is a part of the Android Testing Support Library and provides various methods for accessing instrumentation-related information.

```
Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
```

The assertEquals method checks whether the actual package name obtained from the app's context matches the expected package name ("com.example.hw17"). If they match, the test passes; otherwise, it fails.

```
assertEquals("com.example.hw17", appContext.getPackageName());
```

In layout: ExampleUnitTest This code is an example of a local unit test for a Java class, which is intended to be executed on the development machine (host machine) rather than on an Android device or emulator. Local unit tests are used to test the functionality of individual methods or units of code in isolation from the broader application or Android system.

The code starts with the package declaration and includes necessary imports for JUnit testing. The org.junit.Test annotation is used for marking methods as test cases, and static org.junit.Assert.* includes various assertion methods used for testing.

```
package com.example.hw17;

import org.junit.Test;
import static org.junit.Assert.*;
```

The class is a simple Java class named ExampleUnitTest. It is not an Android-specific test; it's meant for local unit testing.

```
public class ExampleUnitTest {
```

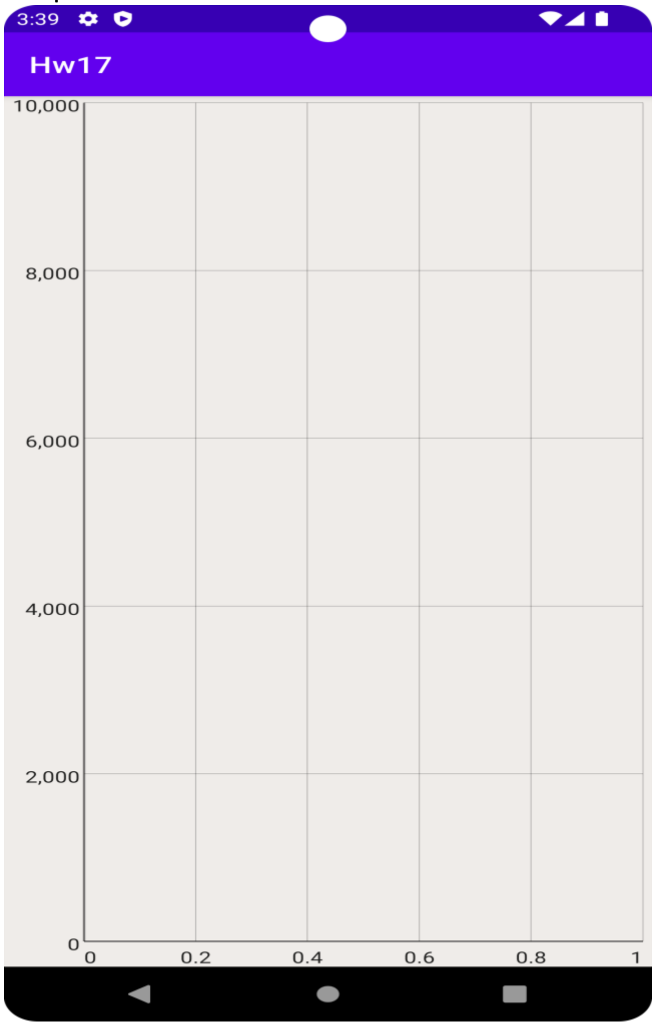
This method is a test case named addition_isCorrect. Test cases in JUnit are methods annotated with @Test. The purpose of this specific test is to check whether addition works correctly.

```
@Test
public void addition_isCorrect() {
```

The assertEquals method checks whether the result of the addition operation (2 + 2) is equal to the expected value (4). If they match, the test passes; otherwise, it fails.

```
assertEquals(4, 2 + 2);
```

Output:


3:39
Hw17
10,000
8,000
6,000
4,000
2,000
0
0
0.2
0.4
0.6
0.8
1