CSC 120: Using APIs

Alarm System

February 23, 2012

1 Introduction

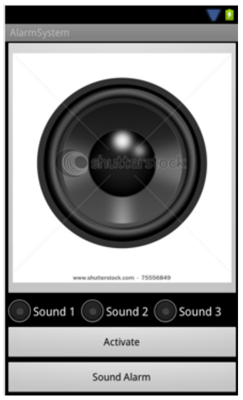
In this lab, you will be creating an alarm system, which will detect sound above a certain amplitude. If sound is detected, an alarm will be activated. In addition to listening for sound, the alarm can also be sounded at any point for testing by the user. In order to accomplish this on Android, you must utilize its onboard microphone and analyze the amplitude of the incoming audio data. This will be done by using classes and methods from APIs.

2 Objective

The purpose of this lab is to introduce and learn the concept of APIs. API stands for Application Programming Interface and is defined as a set of routines, protocols, and tools for building software applications. For this particular lab, you will be using the Android SDK API.

For this lab, you will be using the MediaPlayer and MediaRecorder classes. These classes contain methods that utilize the Android's microphone and speakers to play and record media. Before you begin the activity, visit *http://developer.android.com/reference/android/media/MediaPlayer.html* and *http://developer.android.com/reference/android/media/MediaRecorder.html*. Familiarize yourself with both of these classes and their methods.

The alarm system app should be able to listen for sound and, if detected, sound an alarm. There will be three different choices for the alarm sound. The following picture shows the GUI (graphical user interface) of the app:



Both the speaker picture and the sound alarm buttons will be programmed to sound the alarm when clicked. The activate button will be programmed to begin listening for sound when clicked.

3 Activity

3.1 Research

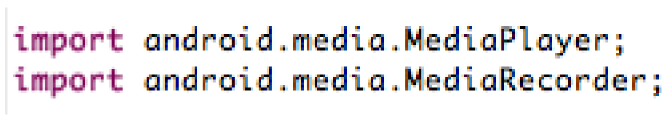
Research the the MediaPlayer and MediaRecorder classes and methods. List which methods you believe will be crucial in programming the alarm system.

3.2 Explore

Open the AlarmSystemActivity.java file. Explore the file. Read the comments. Note the parts of the code that you will be required to fill in. They will be labeled.

3.3 Import

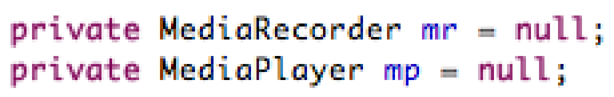
In order to use the classes and methods from APIs, you have to import the required libraries. Import the MediaPlayer and MediaRecorder libraries by inserting the following code at the beginning of the code below the other imports:



```
import android.media.MediaPlayer;
import android.media.MediaRecorder;
```

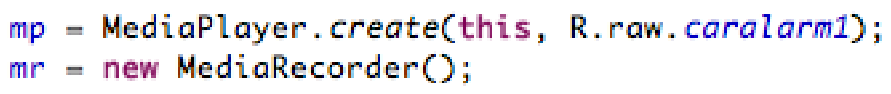
3.4 Create

Now we will create the media class instances and set them to null:



```
private MediaRecorder mr = null;
private MediaPlayer mp = null;
```

And then instantiate them in the onCreate method:

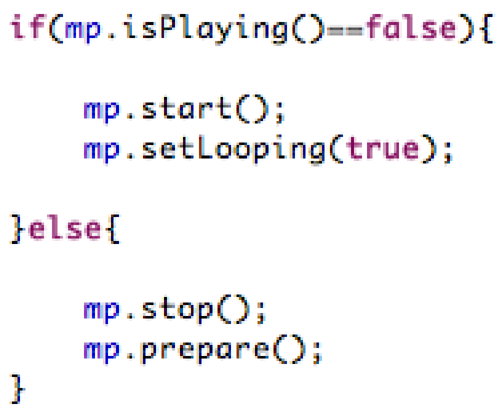


```
mp = MediaPlayer.create(this, R.raw.caralarm1);
mr = new MediaRecorder();
```

The MediaPlayer instance (mp) is created with its context set to this and its loaded sound set to Sound 1 (R.raw.caralarm1).

3.5 Sound Alarm Programming

Now, let's program the buttons to sound the alarm. When the speaker button or sound alarm button is clicked, we want to play or stop the sound depending on if the alarm is already sounding. To do this, we will use the isPlaying() method in the MediaPlayer class to determine if (mp) is currently playing.



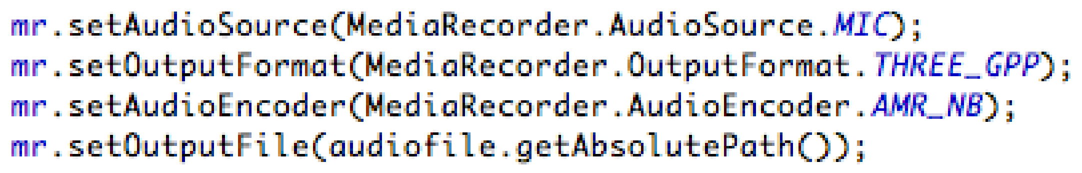
```
if(mp.isPlaying()==false){

    mp.start();
    mp.setLooping(true);

}else{

    mp.stop();
    mp.prepare();
}
```

3.6 Activate Research

Next, we will program the activate button. This will be done in the listen method of the code. There are 4 functions in the MediaRecorder instance that must run before it can start recording. Research how MediaRecorder works and determine what these 4 functions are. Provide the link to the site where you found the information.

3.7 Activate Programming

Now, that you know what these 4 functions are and what they do, let's code:



```
mr.setAudioSource(MediaRecorder.AudioSource.MIC);
mr.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
mr.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
mr.setOutputFile(audiofile.getAbsolutePath());
```

The output file uses a file declared earlier in the code. You do not need to worry about this file. It has already been coded.

3.8 Record

With the MediaRecorder (mr) ready to record, now let's start recording:

```
mr.prepare();
mr.start();
```

Research the prepare method. Why does it have to be called before start the recorder?

3.9 Analyze

Next, we will analyze the recorded audio to determine if the amplitude has reached a certain point. Which method in the MediaRecorder class do you think would be helpful in achieving this?

In order to accomplish this, we will set up a loop that continually checks the maximum amplitude reached, and then check it against the threshold we set up:

```
int i = 0;
while(i==0){

    if(mr.getMaxAmplitude() > ????){

        mr.stop();
        mp.start();
        mp.setLooping(true);
        i=1;

    }
}
```

Everything is now coded except for the amplitude threshold to check for. Test this number and determine which amplitude will be the best. You can research the range this value can be.

3.10 Summary List

The alarm system should now be working properly. Create a list of all the methods used from the MediaPlayer and MediaRecorder libraries with a description of what the method does and what it accomplished in the program.

4 Conclusion

In this lab, you have learned the concept of using APIs. This was done through extensively researching the MediaPlayer and MediaRecorder libraries, and then using their methods to implement an alarm system. In the future, you should be able to research other libraries in an API to accomplish other programming projects.

5 Deliverables

The deliverables for this project are as follows:

AlarmSystem.pkg - with your version of AlarmSystemActivity.java

Lab Report