Problem 1.2.5 Synthesizer

Introduction

This problem will pose several challenges for you to complete within the context of a synthesizer app. The idea is that the synthesizer will play different musical notes depending on the challenge you are completing. Your synthesizer will have buttons for you to use to produce “music”.

Materials

* Computer with Android™ Studio
* Android™ tablet and USB cable, or a device emulator

Problem

Part I: The Challenges

You will be adding functionality to your Synthesizer app in the form of challenges. The challenges are designed to help reinforce your skill with some of the most common Java programming constructs. You may not complete all of the challenges in the time allotted by your instructor. If this is the case, you are encouraged to return to them whenever you finish other activities or projects early as well as outside of class.

As you attempt these challenges, you will find it necessary to cause your app to delay for a time in between playing sounds, so you will set a time delay in your code.

1. Open your Synthesizer project in Android Studio.
2. In SynthesizerActivity, add line 3 as shown.

|  |  |
| --- | --- |
| 1  2  3 | public class SynthesizerActivity extends AppCompatActivity {  private final int WHOLE\_NOTE = 1000; |

WHOLE\_NOTE is a **constant** value, one that never changes after it is created. Here, WHOLE\_NOTE will be used to determine how long to wait between playing one sound and the next.

You will call the method defined in lines 3–9 below whenever you require a pause between playing sounds.

* 1. Add it to your code just before the definition of onButton1Click(View v).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | }  private void delayPlaying(int delay) throws InterruptedException {  try {  Thread.sleep(delay);  } catch (InterruptedException e) {  Log.e("SynthesizerActivity","Audio playback interrupted");  }  }  public void onButton1Click(View v){ |

Below is an example of how a Button identified by mChallenge1 could be set up to play an E, delay for a whole note, and then play an F. As you attempt the challenges, you will replace the code between the try and catch statements (lines 7–9 below) with your own.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | mChallenge1.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Log.e("SynthesizerActivity", "Challenge 0 Button clicked");  try {  mpE.start();  delayPlaying(WHOLE\_NOTE);  mpF.start();  } catch (InterruptedException e) {  e.printStackTrace();  }  }  }); |

Details about the try and catch statements will be covered later in the course. For now, know that you may need to replace their code with your own.

1. Challenge 1: Create a button that plays a scale from low E pitch to high E pitch with a half-note delay in between each note. Use the following sequence of notes.

**E, F Sharp, G, A, B, C Sharp, D, E**

*Hint: a half note can be expressed as* WHOLE\_NOTE/2.

1. Challenge 2: In this challenge you will add functionality so that a user can select a note and the number of times to play that note. When the user presses their Challenge 2 button, the app should play that note, that number of times. To set up for this challenge, add two NumberPickers to your app. You can do this by declaring and initializing them in a similar manner to how you declared and initialized your Buttons. *Hint: you will need to use* NumberPicker’ssetMinValue*,* setMaxValue*, and* getValue *methods, as well as* MediaPlayer’s seekTo*, and* start *methods to complete this challenge.*
2. Challenge 3: Store each of your MediaPlayers in an array and solve Challenge 1 again by iterating through each element in the array.
3. Challenge 4: Solve Challenge 2 again by accessing a new array like the array you created in Challenge 3.
4. Challenge 5: The following sequence of notes is the first line of a version of “Twinkle, Twinkle, Little Star”. Create a Button that plays this tune when pressed.

**A, A, High E, High E, High F Sharp, High F Sharp, High E, D, D, C Sharp, C Sharp, B, B, A**

1. Challenge 6: Complete Challenge 5 without referencing any object of type MediaPlayer within the call to setOnClickListener. *Hint: you may create a helper method that references one or more* MediaPlayers*.*
2. Challenge 7: Complete Challenge 6 using an array to retrieve each note.
3. Challenge 8: The timing of the notes playing is not quite right. Fix it.
4. Challenge 9: Following are the notes that make up the second line of this version of Twinkle, Twinkle, Little Star. The second line repeats itself once. The third line is the same as the first line. Create a Button that plays the entire song when clicked.

**High E, High E, D, D, C Sharp, C Sharp, B**

1. Challenge 10: Complete Challenge 9 and allow the user of the app to select how many times the second line repeats.
2. Challenge 11: Create a CheckBox that toggles whether or not the second line will play.
3. Challenge 12: Create a Button that plays a version of your favorite song.