

## Lab 3: Extending our PD Patch and App

In this weeks lab we will extend the app that we worked with last week so that we can control two oscillators. Obviously we'll need to edit the PD patch. However we will also need to add some new GUI widgets to our app to control the new parts of our patch.

### Setup

Download the example from Github:

[https://github.com/Hydroxate/Lab03\\_Template](https://github.com/Hydroxate/Lab03_Template)

Open the provided template in android studio.

### Edit The PD Patch

Lets create a more interesting PD patch.

- 1) Locate the raw folder in Android Studio.
- 2) Right click and locate the synth.zip in finder.
- 3) Unzip the archive.
- 4) Delete the archive.
- 5) Open the synth.pd file in pure data.
- 6) Edit the patch. Make the following changes:

Create a new oscillator with a default frequency of 220Hz.

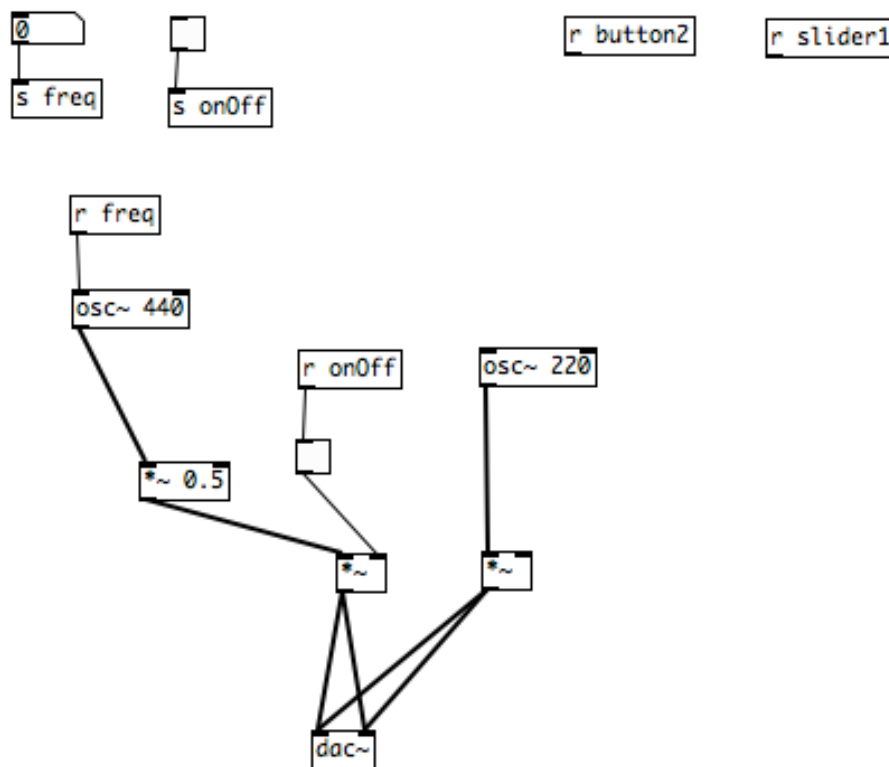
Create a new [ $\ast\sim$ ] to control its amplitude.

Connect the output of the [ $\ast\sim$ ] to the existing [dac~]

Your patch should be similar to the following figure.

Test your patch in pd using toggles and number boxes.

The patch should look similar to that in the next figure.



### Include the new patch in the project

This is the same process as last week. The patch must be zipped to be included in the ndroid app.

- 1) Save the patch
- 2) Close the patch
- 3) Right click the synth patch, and click "Compress synth.pd".
- 4) This will create a zip file called "synth.pd.zip". Rename it "synth.zip".
- 5) Delete synth.pd

### Creating a GUI in Android Studio and Connecting it to a PD Patch.

Now we need to create some new GUI widgets in our app that will communicate with the new [receive ] objects in our PD patch. This is a three step process

- Step 1) First we place the object graphically on the screen of the app and set its properties such as location, color and most importantly, we give it a unique id.
- Step 2) Then we create an instance of the widget type (e.g. button, or slider or switch) in our code with a unique name and we link this to the graphical object we made in step 1.

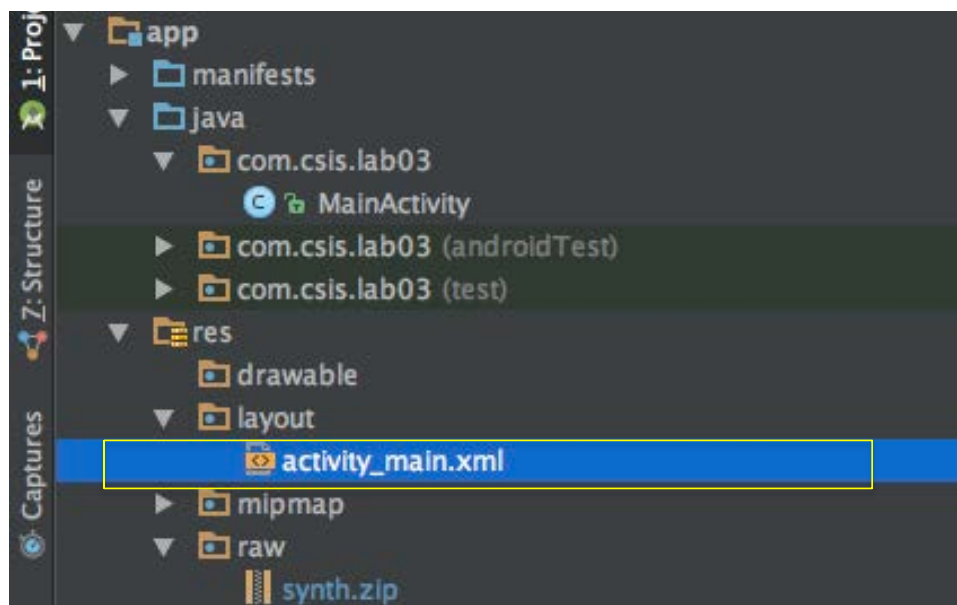
Step 3) Finally we specify what we want to happen when the object is interacted with. We do this by creating a piece of code linked to the instance that allows us to specify for example, “when the button is clicked send a number to pd”. This piece of code is called a listener.

### Add a Button

Lets follow the three step process and add a button to our apps GUI. When this button is clicked it will send a value of “1” to the pd patch.

### Step 1: Add the widget graphically

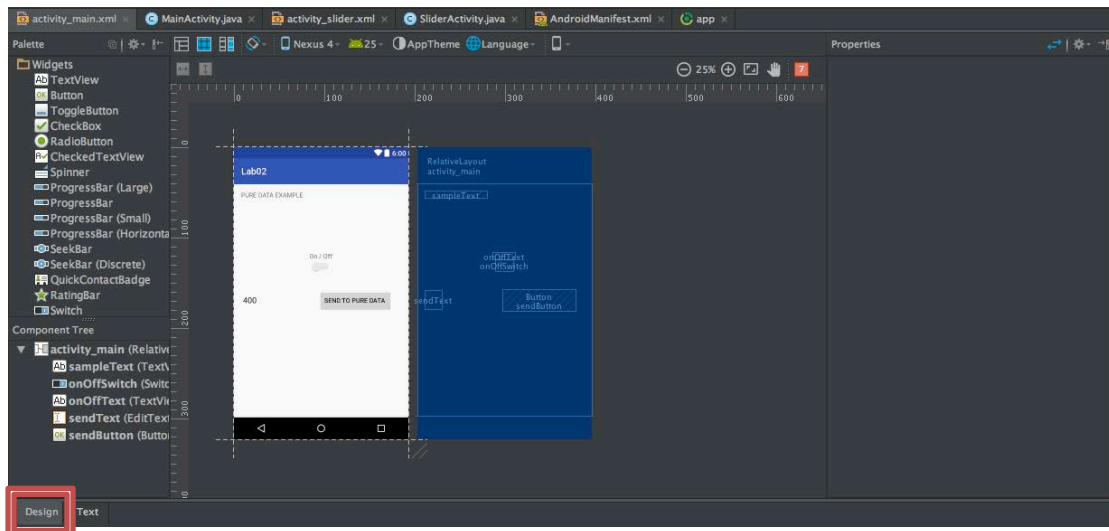
- 1) Click on activity\_main.xml in the project pane, it is located under res/layout.



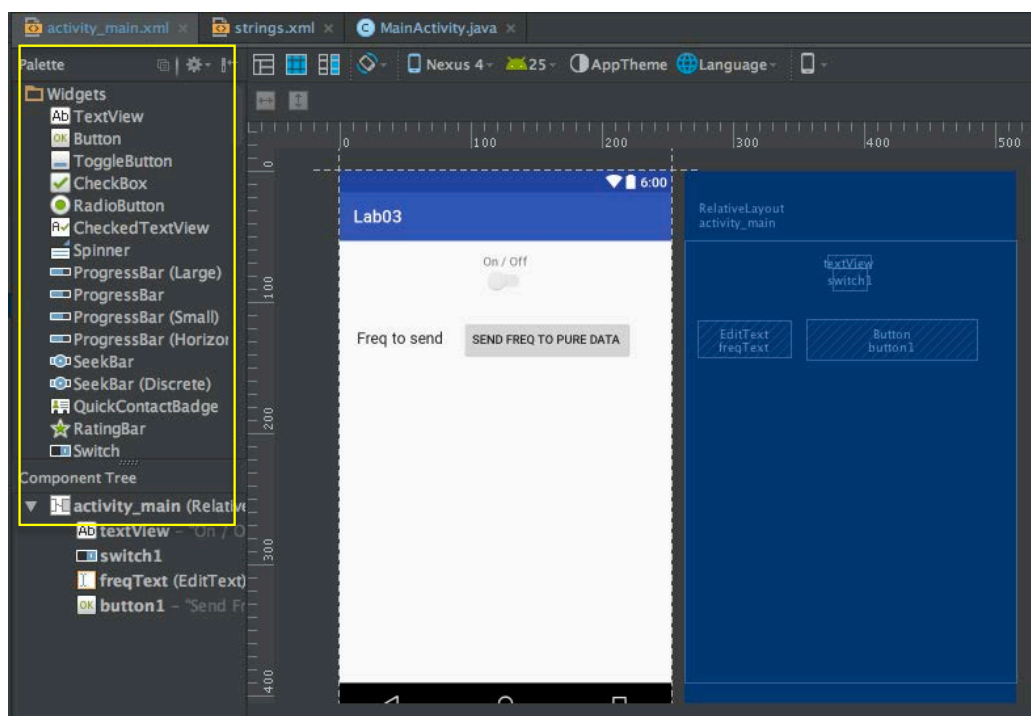
There are two different views possible when working with the xml files. One is a graphical drag and drop view called “design view”. The other is a text view where we can edit the actual xml directly. Its called “text view”. We are going to work with design view for now.

You can switch between text and design view by pressing the highlighted button in the above figure.

For now make sure you are in design view. It should look like the below figure.

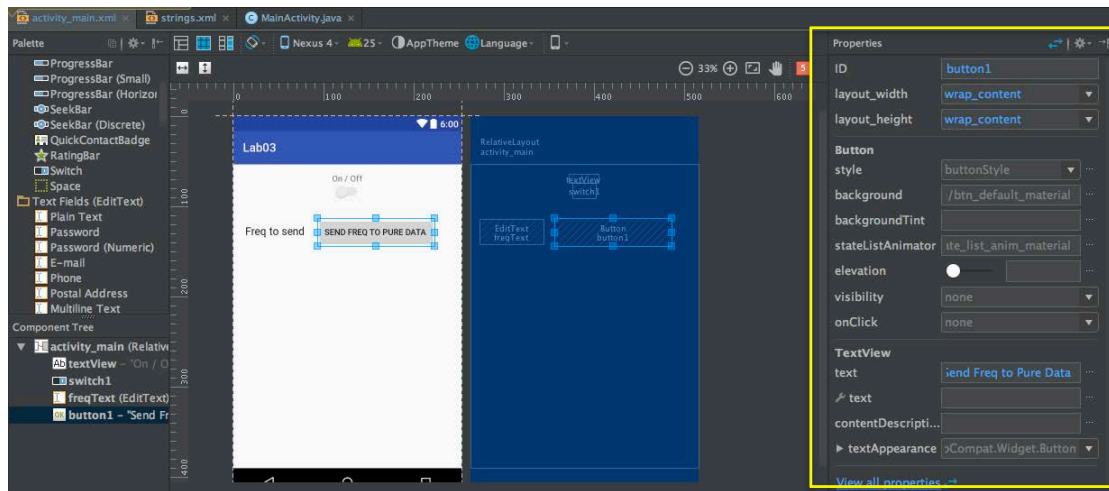


This is the design tab. This is where you can drag and drop items to create your GUI. The widgets (individual GUI objects like buttons, switches, sliders etc) are listed on the left in the palette, these can be dragged and dropped onto the screen.

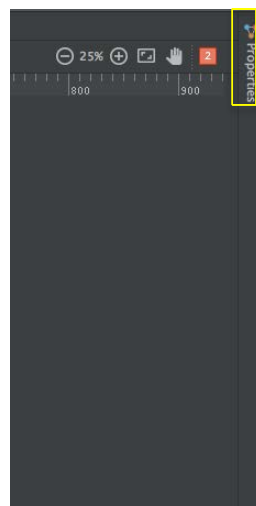


Before we add a new button lets have a closer look at the existing button.

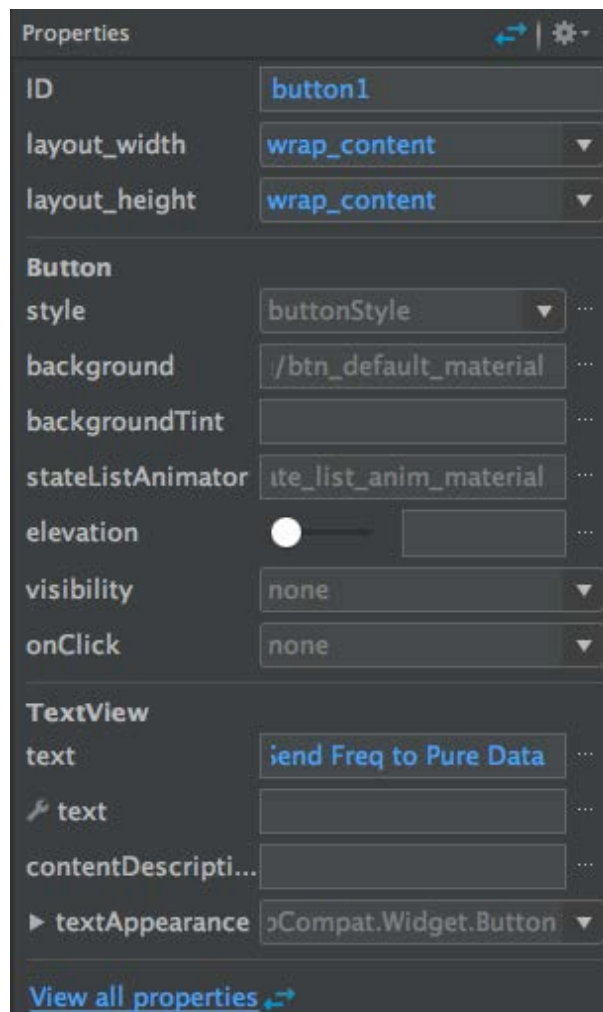
Click on the existing “send freq to pure data” button widget that is already in place . Its properties should now be shown in the properties tab to the right hand side.



HINT: If the properties tab is hidden, click the button to the right to display the tab (see figure below).



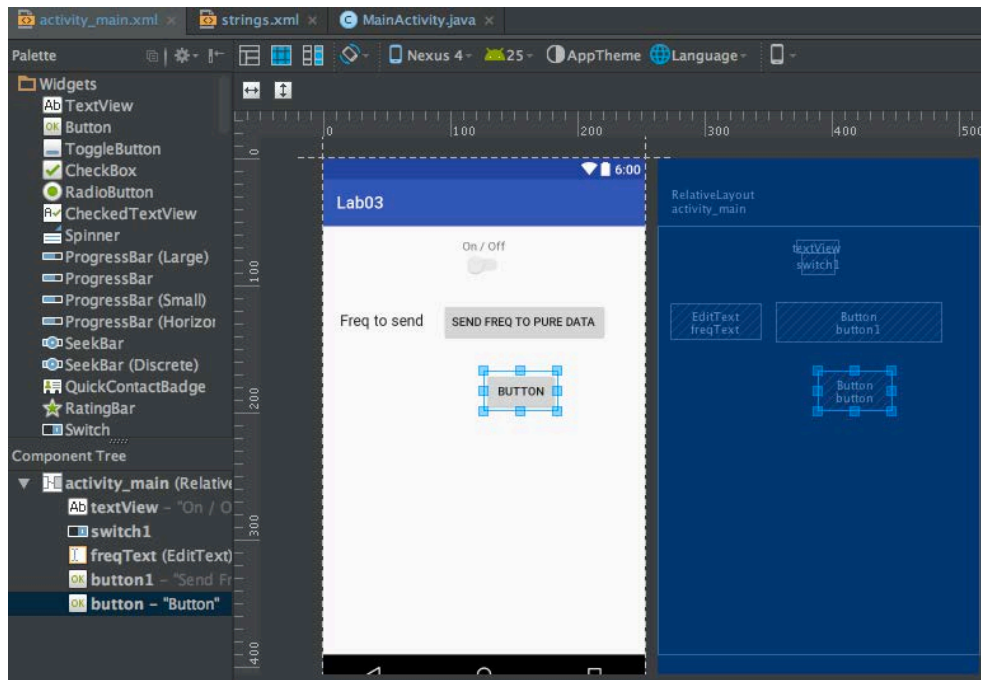
The most important property to note is ID. This ID is what we use in steps 2 and 3 to refer to the widget when we are creating behaviours in our code. There are also properties to set the text of the buttons, their width, and height.



QUESTION: What is the id of the existing “send freq to pure data” button? Note it down on a piece of paper.

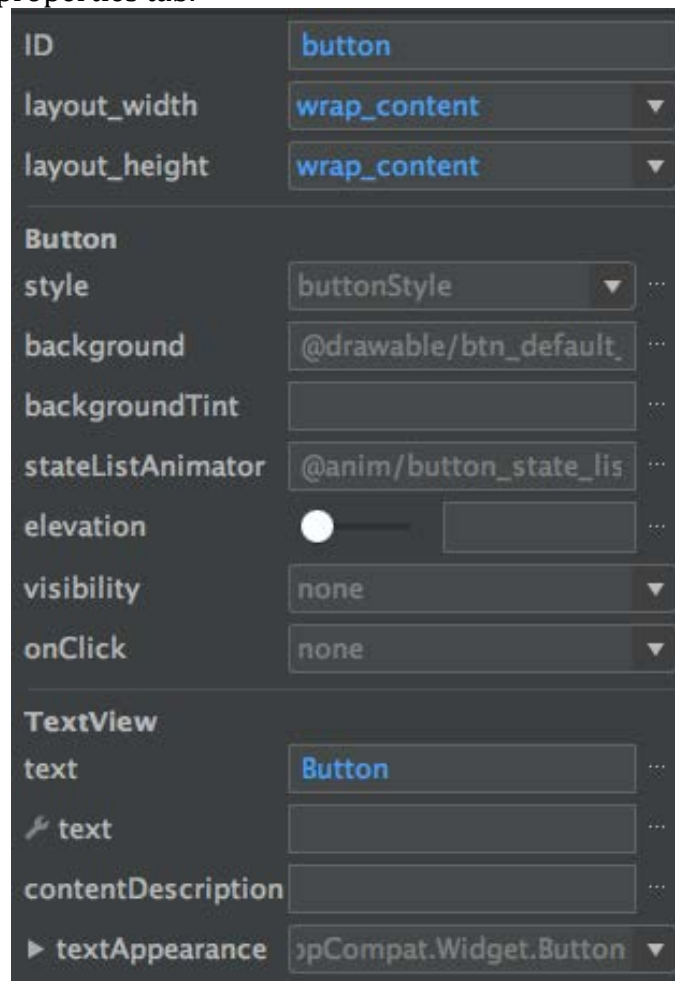
OK. Lets add the new button to our GUI

- 2) Drag a button widget from the Widget palette onto the screen and place it just under the “send freq to pure data” button.



Now we need to set the properties for this new button.

3) Go to the properties tab.



4) Rename the button's ID to button2.

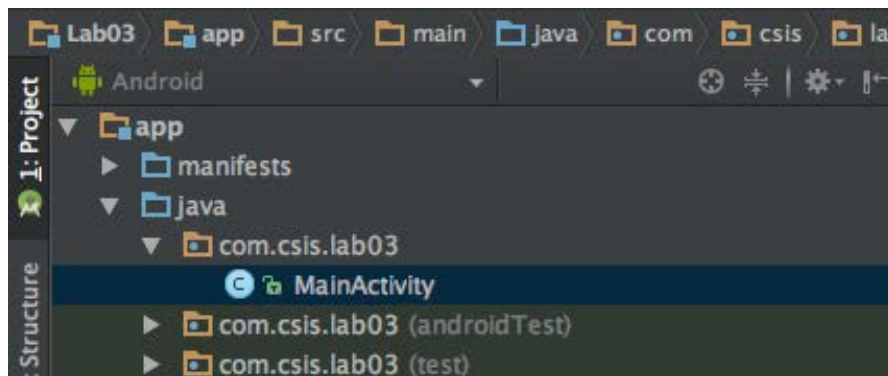
5) Rename the button's text to Button2

We have completed step 1 of our 3 step process for adding a widget to the gui.

### Step 2: Create an instance in code

At this stage our widget only exists as a graphical object, essentially a picture on the canvas. We have finished step 1 of our 3 step process. On to step two, "create an instance of the widget type (e.g. button, or slider or switch) in code with a unique name and link this to the graphical object we just made."

- 1) Double click the MainActivity.java file in the project pane. This is located under the java folder.



Lets have a look at the instance for the existing button. Go to line 39. It looks like the below.

```
Button button1 = (Button) findViewById(R.id.button1);
```

When this line of code is executed it will create an instance of the Button family called "button1" that is connected with the graphical button that has an ID of "button1". Read that line again a few times. It's important. What part of the line of code specifies that the instance will be called "button1"?

Answer: Well, look at the beginning of the line before the = (i.e. Button button1). This means create an instance of the Button family and name it "button1".

Where in the code does it specify what graphical button we are referring to?

Answer: Look at the end of the line where it says "r.id.button1". "button1" is the ID of the "send freq to pure data" button widget that was typed in the properties section in the GUI design tab. By specifying it here we link the Button instance (called button1) to the existing graphical object with id of button1.

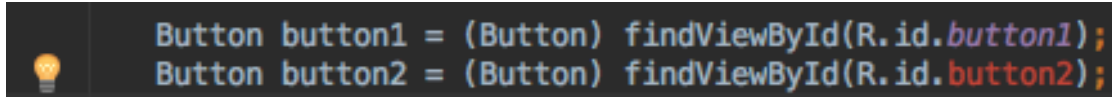


Lets add the code to create an instance of the Button family - which we will call "button2" – and link it to our new button widget (which has an ID of button2).

2) Underneath the above line of code add the following:

```
Button button2 = (Button) findViewById(R.id.button2);
```

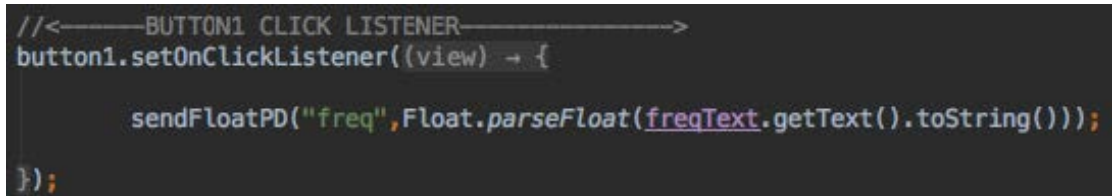
It should look like the below now:



We have now completed step 2 of our three step process. We have created a Button instance with a unique name (button2) and linked it to the graphical button object that we palced on the canvas in step 1. We are ready for the third and final step.

### Step 3: Add functionality.

Now we need to add the functionality. We want our button to send a value of 1 to our patch when it is clicked. To do this we must add a listener: code that detects and responds to interaction. Lets look at the listener code for the included "send freq to pure data" button. Its just below the line we just added. See the next figure.



Look at the beginning of this code i.e. button1.setOnClickListener. We could read this as "attach an onclick listener to the instance called "button1"".

Now look at the next line that begine with "sendFloatPD". This sendFloatPD method is used to send floats to PD. It needs to be given two arguments, 1, the name of the [receive ] object in the patch you want to send to and 2 a float value to send.

NOTE: With the existing button1 we are grabbing the float we want to sent from a text box widget which has an ID of freqText.

```
//<-----BUTTON1 CLICK LISTENER----->
button1.setOnClickListener((view) -> {

    sendFloatPD("freq", Float.parseFloat(freqText.getText().toString()));

});
```

For now though we just want to send a “1” when our new button is clicked. The listener for this is very simple. Lets create it.

### 1) Create the listener for the button **button2**.

Type this at line 75.

```
button2.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View view)
    {
        // send a float to PD("the name of the [receive ] object in pd to
        send to", the value to send);

        sendFloatPD("button2", 1.0f);

    }
});
```

It should look like this:

```
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // send a float to PD("the name of the receive object in pd to send to", the value to send);
        sendFloatPD("button2", 1.0f);
    }
});
```

We have now completed step 3 of our three step process for adding widgets to the apps gui. This code will be executed whenever button2 is clicked. It will send a 1 to the [receive button2] object in the pure data patch.

Make sure there is a [receive button2] object in your pd patch to handle this float. It should be controlling the amplitude of an oscillator.

### 2) Build and run the app and test! Is the button working?

If the button is not working you should go back over the above steps. Things to check:

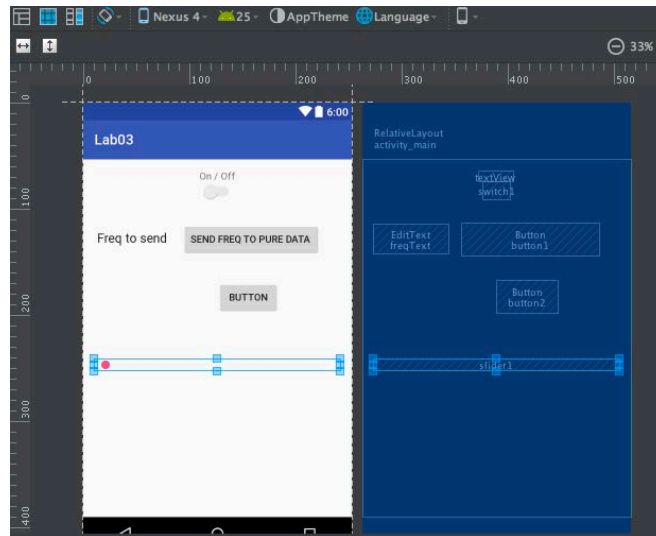
- Does the id you gave your button match the id specified in step 2?
- Is the listener attached to the correct instance?
- Does your patch have a receive button2 object?
- Is the name of the receive object correctly specified in sendFloatPD method ?
- Raise your hand for assistance if you are still stuck having checked these things.

## Part 2: Adding a Slider to the GUI

Lets repeat our three step process but this time we'll add a slider. In android this is called a **seekbar**.

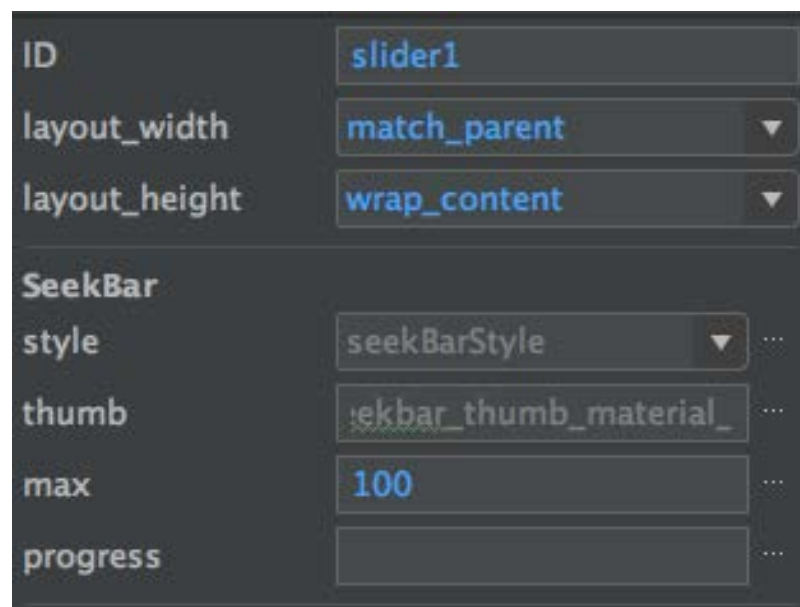
### Step 1: Add the seekbar widget

- 1) Return to the Design view of the GUI by double clicking activity\_main.xml.
- 2) Drag and drop a seekbar under the button. It will look like this:



- 3) Edit the properties as following:  
Rename seekbar ID to slider1.  
Set "max" to 100.

It will look like the following:



### Step 2: Create an instance in code

We now need to add an instance of the slider in the code. But first we must declare the seekbar at the top of the class. This is the difference between seekbars and other GUI widgets.

1. Go to line 25 and type the seekbar declaration. It will look like the below:

```
public class MainActivity extends AppCompatActivity {  
    private PdUiDispatcher dispatcher; //must declare this to use later, used to receive data from sendEvents  
    private SeekBar slider1; //Declaring slider1 here  
    float slide1Value = 0.0f;
```

We now create the instance.

2. Go to line 76 in MainActivity.java and add the below:

```
slider1 = (SeekBar) findViewById(R.id.slider1);
```

We have now completed step 2 of our three step process. We have created a seekBar instance with a unique name (slider1) and linked it to the graphical seekBar object, with an id of slider1, that we placed on the canvas in step 1.

On to step 3. Lets add the listener code.

### Step 3: Add the listener code

Add the listener code as follows:

```
slider1.setOnSeekBarChangeListener(  
    new SeekBar.OnSeekBarChangeListener()  
{  
    @Override  
    public void onProgressChanged(SeekBar seekBar, int progress,  
    boolean fromUser)  
    {  
        slide1Value = progress / 100.0f;  
        sendFloatPD("slider1", slide1Value);  
    }  
  
    @Override  
    public void onStartTrackingTouch(SeekBar seekBar) {  
    }  
  
    @Override  
    public void onStopTrackingTouch(SeekBar seekBar) {  
    }  
  
});
```

It will look like the following:

```

//<-----SLIDER 1 LISTENER----->
slider1 = (SeekBar) findViewById(R.id.slider1);

slider1.setOnSeekBarChangeListener(
    new SeekBar.OnSeekBarChangeListener()
    {
        @Override
        public void onProgressChanged(SearchBar seekBar, int progress, boolean fromUser) {
            slide1Value = progress / 100.0f;
            sendFloatPD("slider1", slide1Value);
        }

        @Override
        public void onStartTrackingTouch(SearchBar seekBar) {
        }

        @Override
        public void onStopTrackingTouch(SearchBar seekBar) {
        }
    }
);

```

5) Check that you have a receive object in the pd patch to get the data from the slider i.e. [receive slider1]. You will be receiving a number between 0.0 and 1.0. Use it to control something.

## Going Onwards

The homework for this lab will be sent out shortly. You'll be required to add lots of gui objects and get them controlling some sound processing in PD.

Use the rest of the lab to practice the three step process for adding GUI widgets and getting them communicating with PD.

You will need to be able to do this quickly in the coming weeks. Practice, practice practice. You will learn best by making mistakes and trying to solve them.

### Bonus:

add a new text field and button. Create a listener for the button that sends the value from the text field to pd when it is clicked. HINT: the text field will need an ID that the buttons listener can refer to. Note that you have an example already, this is how the included button works.