Java Extra Credit 2: Android Development

**Files to Submit: MainActivity.java and activity\_main.xml**

**Prerequisites: Extra Credit Java Assignment 1: Java Sundaes! Link here**

**Points: \_\_**

Android uses Java to develop its applications. In this assignment, you will create a tip calculator to gain some basic experience with Android development.

Start by downloading and installing Android Studio, following the instructions at the links below.

<https://developer.android.com/studio/index.html>

<https://developer.android.com/studio/install.html>

This video will familiarize you a bit with the Android Studios platform and how to link the display to your code. <https://www.youtube.com/watch?v=aE5f1tV5nU4&t=947s> There are also some helpful pointers for basic app building here:

<https://developer.android.com/training/basics/firstapp/index.html>

* Note: The video mentions an important concept called “casting” when you declare a new object.

Button btn = (Button) findViewById(R.id.btnDoMagic);

(Button) is called a cast, and it is a promise that the object, which in this case is of type View, is actually a Button object. If this doesn’t make sense, don’t worry, just follow this format when you create other objects!

**Creating Your Tip Calculator**

Open Android Studios and follow along with the beginning of the video to get yourself started. In the Application name box, call your project Tip Calculator and leave the Activity Name as MainActivity.

When you run your project for the first time, you will not have any available virtual devices listed. You can choose to either run the app on an emulator (Nexus 5 works well) or on a real Phone, following the instructions here: <https://developer.android.com/training/basics/firstapp/running-app.html>

The emulator will take a while to load when you first open it. Subsequent runs will not take as long to load if you leave the emulator open.

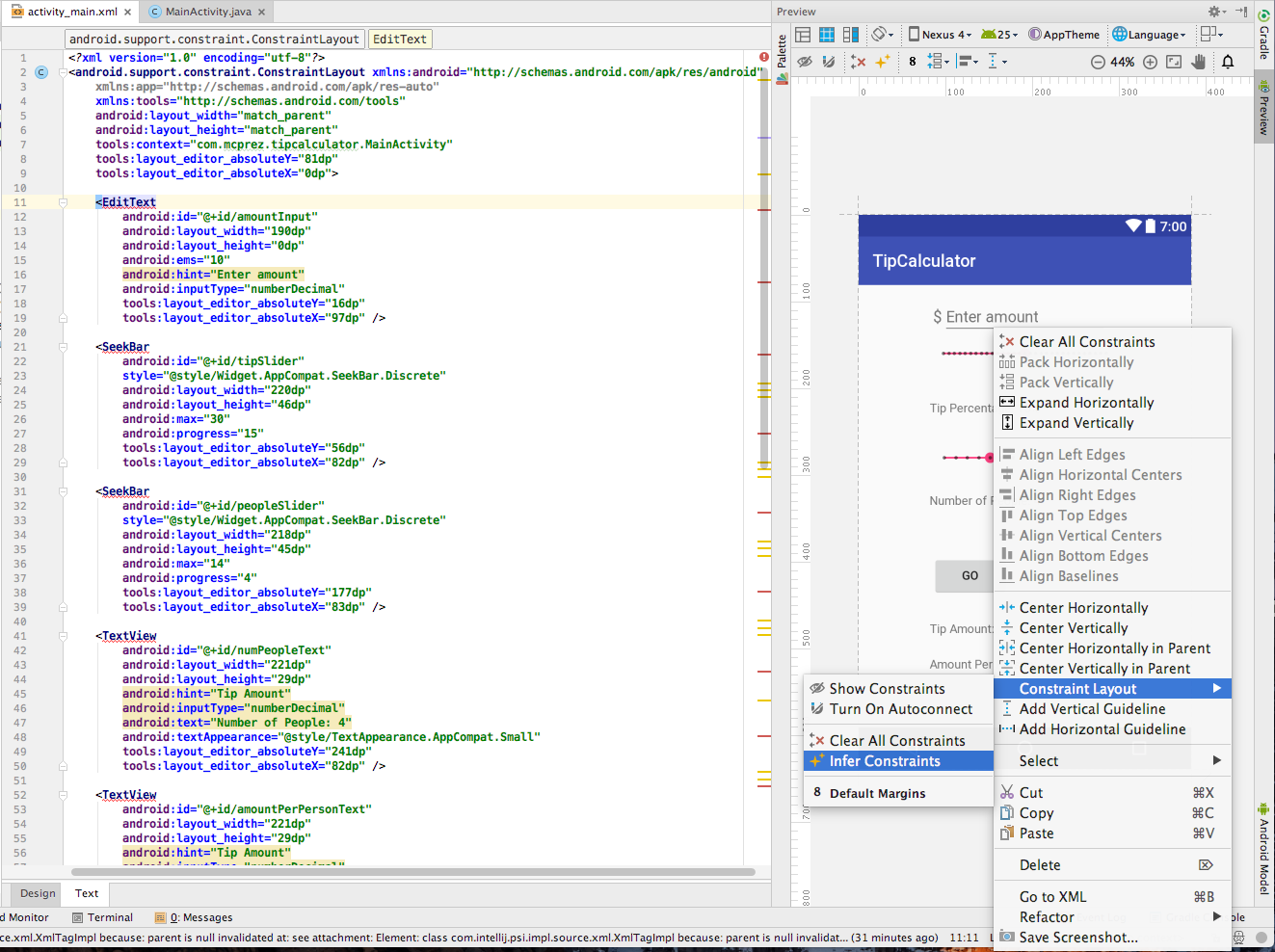
Navigate to activity\_main.xml as shown in the video. At the bottom of the window, click the Design tab. This is where we will begin building the display. The top half of the window shows your Palette, and contains various objects you can put into your display.

1. Click Text, and drag a Number (Decimal) textbox to the top of the display. This allows the user to input a decimal number when using the app. Once the text box is in the display, click on it. Then, find the Properties window on the right and type “Enter Amount” in the Hint box. This is what the text box will show before the user begins typing numbers into it.
2. Next go to All in the Palette, and put a Seekbar (Discrete) underneath the textbox. This will be a slider for your tip percentage. In the Properties window, set the max to 30 (the min is permanently set to 0) and set the progress (the default value when you open the app) to 15.
3. Back in the palette, go to Text and insert a TextView under the seekbar. This will be a label for the seekbar. Change the Text to “Tip Percentage: 15%” to correspond with our default value
4. Underneath, add another slider with a max of 15 and a progress of 4. Add a TextView label that says “Number of People: 4”. This will allow you to split the bill evenly after the total amount is calculated.Add a “GO” button, and two more TextViews that say “Tip Amount:” and “Amount Per Person:” below. Your display should now look like this:  
   
5. Next, click on the Text tab at the bottom of the window. Here you will see the code generated when you added objects to the display. Give each of your objects an informative id by changing the text after **android:id="@+id/**. For example:

**android:id="@+id/editText"** to **android:id="@+id/inputAmount"**.

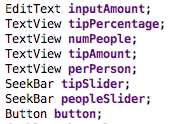
Do this for each object.

1. Notice at this point, there is an error on each object heading. This is because the view is not constrained, so the objects will not be positioned correctly when you run the app. To fix this, right-click on an object in the Preview window and select Constraint Layout -> Infer Constraints. This automatically creates constraints for the objects based on the your design display.



Great! Now let’s add some functionality to each component.

1. Go to MainActivity.java by clicking on the tab at the top. This is where you will write your code that controls each object.
2. Start by declaring variables to represent each of your objects, using the same format as with the button in the video. Just as in the video, you will have to import a library each time you introduce a new type of object Don’t forget to cast! For example: 



These are the variable names I will be using throughout this tutorial:

1. First, let’s connect each of the seekbars to their corresponding TextView label. Similar to the button example in the video, set the tip seekbar to listen for any changes using   
     
   **tipSlider**.setOnSeekBarChangeListener(**new** SeekBar.OnSeekBarChangeListener() {}  
     
   Add the methods used to implement the interface with Alt-Enter. Three methods will appear: onProgressChanged(), onStartTracking(), and onStopTracking(). For our tip calculator, we are concerned about what happens when the number changes, so we will only add code within the first method. When we adjust the seekbar, we want the label beneath to change based on the progress (aka value) of the seekbar. To do this, we must use



Do the same for the second seekbar that adjusts the number of people.

1. Finally, we’ll add the functionality of the “Go” button. When we hit the button, it should calculate the tip and the amount each person has to pay and display the results in the last two TextView labels. Following the video guidelines, create an on-click listener for the button.

Now we can add code within the listener. First, we need to take the user input and turn it into a Double so that we can perform calculations with it. We use the method parseDouble() to convert the input text String into a Double.

**Double subtotal** = new Double (Double.*parseDouble*(**inputAmount**.getText().toString()));

Next, we calculate the tip by multiplying the subtotal and the tip percentage. You can retrieve the value of the tip seekbar using the method getProgress().

**Double tip** = new Double (subtotal \* **tipSlider**.getProgress()/100);

Calculate the amount per person in a new variable by dividing the total amount (subtotal + tip) by the number of people (retrieved from the value of the slider).

Lastly, set the text of each of the last two labels to display the values you just calculated.

**tipAmount**.setText(**"Tip Amount: $ "** + String.*format*(**"%.2f"**, **tip**));

**perPerson**.setText(**"Amount Per Person: $"** + String.*format*(**"%.2f"**, **amountPerPerson**));

Because the number is a monetary value, we use “%.2f” to force the String to display two decimal points.

1. You’re all finished! Run your app and test it out! To submit, close your project and navigate to the location you saved it in. Open the folder, and search for MainActivity.java and activity\_main.xml. Submit these two files.