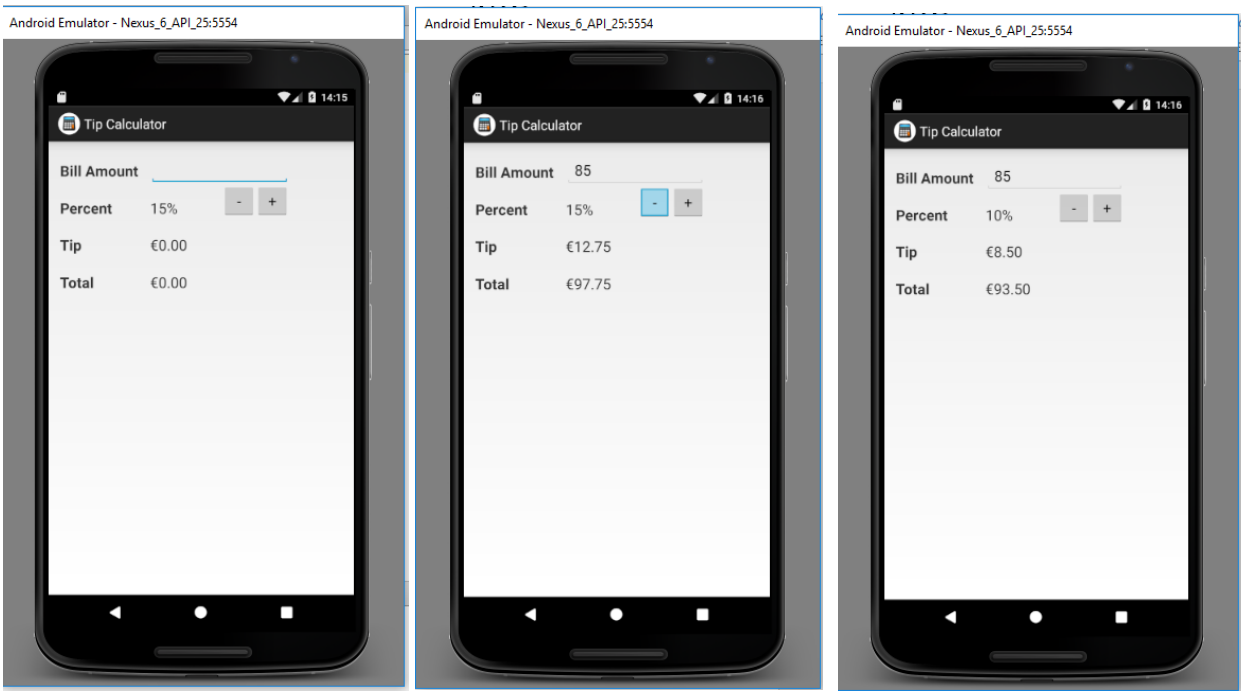
Lab 3: Tip Calculator App
Ref Murach Chapter 3

Learning outcomes:

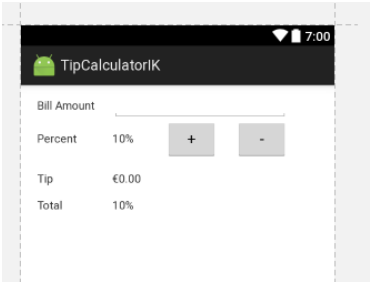
- Be able to use Android Studio to develop a basic GUI driven Android application to enter a bill amount and select a tip percentage.
- Know how to use the Interactive Layout Builder
- Understand basic event handling

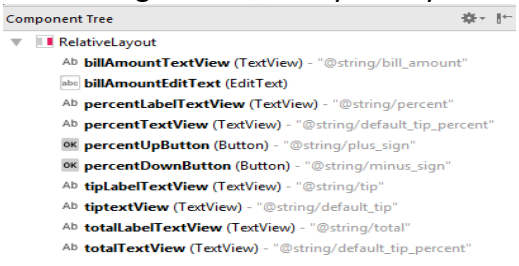


Property	Value
Application Name	TipCalculatorXX
Company Domain	mad.com
Package Name	com.mad.tipcalulatorxx
API (Minimum, Target, Compile with)	Latest (i.e. use the default)
Template	Empty Activity

Task:

1. Create a project for a new Android app named TipCalculatorXX. The project should be stored in a package named com.mad.tipcalculator
2. Add the seven TextView Widgets, one EditText View and two Buttons to the layout shown. Set the id and Text properties of each widget immediately after you add each widget.





3. In your MainActivity class define class member variables for the widgets you will need to access:

```
// define variables for the widgets
private EditText billAmountEditText;
private TextView percentTextView;
private Button   percentUpButton;
private Button   percentDownButton;
private TextView tipTextView;
private TextView totalTextView;
```

4. In your oncreate() method get your variables to reference the widgets created in your activity_main.xml file

```
// get references to the widgets
billAmountEditText = (EditText) findViewById(R.id.billAmountEditText);
percentTextView = (TextView) findViewById(R.id.percentTextView);
percentUpButton = (Button) findViewById(R.id.percentUpButton);
percentDownButton = (Button) findViewById(R.id.percentDownButton);
tipTextView = (TextView) findViewById(R.id.tipTextView);
totalTextView = (TextView) findViewById(R.id.totalTextView);
```

5.
```
// define class instance variables that should be saved and used in calculations
private String billAmountString = "";
private float tipPercent = .15f;
```

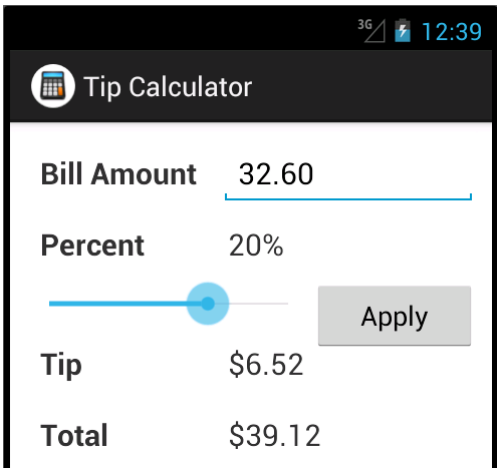
6.
```
// set the listeners
billAmountEditText.setOnEditorActionListener(this);
```

7.
```
public void calculateAndDisplay() {

    // get the bill amount
    billAmountString = billAmountEditText.getText().toString();
    float billAmount;
    if (billAmountString.equals("")) {
       billAmount = 0;
    }
    else {
       billAmount = Float.parseFloat(billAmountString);
    }

    // calculate tip and total
    float tipAmount = billAmount * tipPercent;
    float totalAmount = billAmount + tipAmount;

    // display the other results with formatting
    NumberFormat currency = NumberFormat.getCurrencyInstance();
    tipTextView.setText(currency.format(tipAmount));
    totalTextView.setText(currency.format(totalAmount));
```

```
        NumberFormat percent = NumberFormat.getPercentInstance();
        percentTextView.setText(percent.format(tipPercent));
    }
```

8.
```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.percentDownButton:
            tipPercent = tipPercent - .01f;
            calculateAndDisplay();
            break;
        case R.id.percentUpButton:
            tipPercent = tipPercent + .01f;
            calculateAndDisplay();
            break;
    }
}
```

9.
```
@Override
public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
    if (actionId == EditorInfo.IME_ACTION_DONE ||
            actionId == EditorInfo.IME_ACTION_UNSPECIFIED) {
        calculateAndDisplay();
    }
    return false;
}
```



Add a seek bar

1. Open the XML file for the activity and delete the two buttons that set the tip percent.
2. Create a new row below the "Percent" row that consists of a seek bar followed by the Apply button as shown above.
3. The seek bar should have a maximum value of 30 and a default progress of 15.
4. Open the .java file for the activity and delete all code related two the two buttons that have been deleted.

5. Modify the calculateAndDisplay method so it gets the correct tip percent from the SeekBar widget and displays the correct tip percent on the related TextView widget.
6. Test this change to make sure it works correctly. Note that you must click the Apply button to display the new tip percent that has been set by the seek bar. Although this seek bar should work, it doesn't provide a responsive user interface. That's why the next chapter shows how to handle the events of a seek bar to provide a more responsive user interface.

Write the Java code:

1. Delete all methods except the onCreate() method
2. Use the onCreate() method to get references to the EditText Widget and the three TextView widgets that display data
3. Create a method named calculateAndDisplay(). This method should get the subtotal value. Then it should calculate the discount amount and total. It should give a 20% discount if the subtotal is greater than or equal to 200, a 10% discount if the subtotal is greater than or equal to 100 and no discount if the subtotal is less than 100
4. Add code to the end of calculateAndDisplay() method that displays the results of the calculation on the widgets for the discount percent, discount amount and total
5. Override the onResume() method to call the calculateAndDisplay()method
6. Handle the EditorAction event for the EditText widget so that it executes the calculateAndDisplay method when the Done key is pressed
7. Test the app. Change the orientation. The app should retain its data
8. Override the onPause() method so that it saves the string for subtotal. Then modify the onResume method so that it gets the string for the subtotal.
9. To get these methods to work correctly you need to set up instance variables for the subtotal string and for a SharedPreferences object that you can use to save and get this String.
10. Test the app again. This time the app should always remember the last subtotal that you entered even if you navigate away from the app and return to it.

Deliverable: Upload your compressed project folder to Moodle