Lab assignment 1

Due date: Tue 5/31

40 points

Team of 3 to 4 students

Divide the team into two subgroups with two team members per group.

The group works on the lab assignment and can discuss the code with another group. Each group demonstrates the lab to each other. The team then upload the solution from one of the subgroups.

Write the contribution to the lab assignment, the contribution percentage (maximum of 100%) on the comment box for each team member.

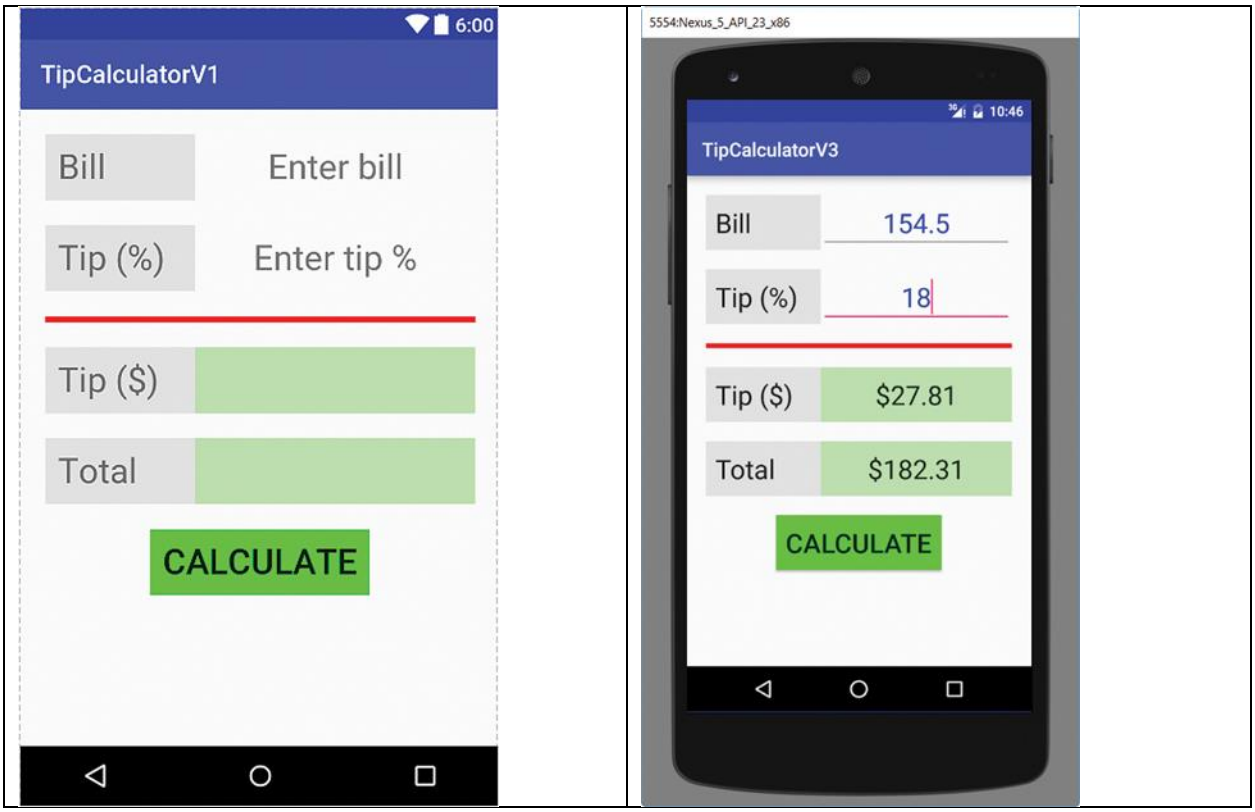
All the team members must agree on the percentage. If the team members have 50% and the lab assignment grade is 40/40, they get only 20 points. The instructor will review the paper summarizing and validate the percentage of each team member. If you have problems with team members, please let me know.

Problem

In this lab assignment you will demonstrate how to implement a model view controller (MVC), UI components, and events.

Design

Version 1 – UI Components, Styles and Themes.



The team will make a simple Tip Calculator app using the Model View Controller architecture; in order to do that, we code three important files:

- TipCalculator.java: the TipCalculator class encapsulates the functionality of a tip calculator; this class is the Model of the app,
- activity_main.xml: this file defines the View of the app, and
- MainActivity.java: this class is the Controller of the app.

You start a new Android Studio project, you call it TipCalculatorv1, and you choose the Empty Activity template.

Model

For this app, the Model is simple and is only composed of one class that encapsulates a tip calculator, the TipCalculator class. It is a regular Java class.

```
package com.example.tipcalculator;

public class TipCalculator {
    private float tip;
    private float bill;

    public TipCalculator(float newTip, float newBill ) {
```

```
        setTip( newTip );
        setBill( newBill );
    }

    public float getTip( ) {
        return tip;
    }

    public float getBill( ) {
        return bill;
    }

    public void setTip( float newTip ) {
        if( newTip > 0 )
            tip = newTip;
    }

    public void setBill( float newBill ) {
        if( newBill > 0 )
            bill = newBill;
    }

    public float tipAmount( ) {
        return bill * tip;
    }

    public float totalAmount( ) {
        return bill + tipAmount( );
    }
}
```

AndroidManifest.xml

In order to keep things simple, we only allow the app to work in vertical orientation, which we can specify in the AndroidManifest.xml file. We do that by assigning the value portrait to the android:screenOrientation attribute of the activity tag.

```
<activity android:name=".MainActivity"
    ____________________________________>
    <intent-filter>
. . . . . .
```

themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.TipCalculator"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
```

```
    <style name="TextStyle" parent="@android:style/TextAppearance">
        <item name = "android:layout_width">wrap_content</item>
        <item name = "android:layout_height">wrap_content</item>
        <item name = "android:textSize">28sp</item>
        <item name = "android:padding">10dp</item>
    </style>

    <style name="LabelStyle" parent="TextStyle">
        <item name = "android:background">@color/lightGray</item>
    </style>

    <style name="CenteredTextStyle" parent="TextStyle">
        <item name = "android:gravity">center</item>
    </style>

    <style name="InputStyle" parent="CenteredTextStyle">
        <item name = "android:textColor">@color/darkBlue</item>
    </style>

    <style name="OutputStyle" parent="CenteredTextStyle">
        <item name = "android:background">@color/lightGreen</item>
    </style>

    <style name="ButtonStyle" parent="TextStyle">
        <item name = "android:background">@color/darkGreen</item>
    </style>

</resources>
```

Actvivity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout tools:context="com.example.tipcalculator.MainActivity"
android:layout_height="match_parent" android:layout_width="match_parent"
xmlns:tools="http://schemas.android.com/tools"
xmlns:android="http://schemas.android.com/apk/res/android">
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/label_bill"
        style="@style/LabelStyle"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="20dp"
        android:minWidth="120dp"
        android:text="@string/label_bill"/>

    <EditText
        android:id="@+id/amount_bill"
        style="@style/InputStyle"
        android:layout_marginRight="20dp"
        android:layout_toRightOf="@+id/label_bill"
        android:layout_alignBottom="@+id/label_bill"
        android:layout_alignParentRight="true"
        android:hint="@string/amount_bill_hint"
        android:inputType="numberDecimal" />
```

```
<TextView
    android:id="@+id/label_tip_percent"
    style="@style/LabelStyle"
    android:layout_marginTop="20dp"
    android:layout_below="@+id/label_bill"
    android:layout_alignLeft="@+id/label_bill"
    android:layout_alignRight="@+id/label_bill"
    android:text="@string/label_tip_percent"/>

<EditText
    android:id="@+id/amount_tip_percent"
    style="@style/InputStyle"
    android:layout_toRightOf="@+id/label_tip_percent"
    android:layout_alignBottom="@+id/label_tip_percent"
    android:layout_alignRight="@id/amount_bill"
    android:hint="@string/amount_tip_percent_hint"
    android:inputType="number" />

<!-- red line -->
<View
    android:id="@+id/red_line"
    android:layout_below="@+id/label_tip_percent"
    android:layout_marginTop="20dp"
    android:layout_height="5dip"
    android:layout_width="match_parent"
    android:layout_alignLeft="@id/label_bill"
    android:layout_alignRight="@id/amount_bill"
    android:background="#FF00" />

<TextView
    android:id="@+id/label_tip"
    style="@style/LabelStyle"
    android:layout_marginTop="20dp"
    android:layout_below="@id/red_line"
    android:layout_alignLeft="@+id/label_bill"
    android:layout_alignRight="@+id/label_bill"
    android:text="@string/label_tip" />

<TextView
    android:id="@+id/amount_tip"
    style="@style/OutputStyle"
    android:layout_toRightOf="@+id/label_tip"
    android:layout_alignBottom="@+id/label_tip"
    android:layout_alignRight="@id/amount_bill" />

<TextView
    android:id="@+id/label_total"
    style="@style/LabelStyle"
    android:layout_marginTop="20dp"
    android:layout_below="@id/label_tip"
    android:layout_alignLeft="@+id/label_bill"
    android:layout_alignRight="@+id/label_bill"
    android:text="@string/label_total" />
```

```
    <TextView
        android:id="@+id/amount_total"
        style="@style/OutputStyle"
        android:layout_toRightOf="@+id/label_total"
        android:layout_alignBottom="@+id/label_total"
        android:layout_alignRight="@id/amount_bill" />
    <Button android:text="@string/button_calculate"
        android:layout_marginTop="20dp"
        style="@style/ButtonStyle"
        android:layout_below="@+id/amount_total"
        android:layout_centerHorizontal="true"
        android:onClick="calculate"/>

</RelativeLayout>
```

The code above is used the relative layout. **You will redesign UI with ConstraintLayout**.

Version 2 – Events and Simple Event Handling: Coding the Controller

Modify activity_main.xml file by adding a method android:onClick:"methodName" called calculate to execute when the user clicks on the Button.

```
<Button android:text="@string/button_calculate"
    android:layout_marginTop="20dp"
    style="@style/ButtonStyle"
    android:layout_below="@+id/amount_total"
    android:layout_centerHorizontal="true"
    _____________________________/>
```

Modify MainActivity class

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import java.text.NumberFormat;

public class MainActivity extends AppCompatActivity {
    private TipCalculator tipCalc;
    public NumberFormat money = NumberFormat.getCurrencyInstance();
    private EditText billEditText;
    private EditText tipEditText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        tipCalc = new TipCalculator( 0.17f, 100.0f );
        setContentView(_____________________________);
```

```
    }

    public void calculate(View v) {
        billEditText = (EditText) findViewById(R.id.amount_bill);
        tipEditText = _____________________________________________
        String billString = billEditText.getText().toString();
        String tipString = ________________________________________

        TextView tipTextView =
                (TextView) findViewById(R.id.amount_tip);
        TextView totalTextView =
                ___________________________________________
        try {
            // convert billString and tipString to floats
            float billAmount = Float.parseFloat(billString);
            int tipPercent = ___________________________________
            // update the Model
            tipCalc.setBill(_____________________);
            tipCalc.setTip(.01f * tipPercent);
            // ask Model to calculate tip and total amounts
            float tip = _____________________________
            float total = ____________________________
            // update the View with formatted tip and total amounts
            tipTextView.setText(money.format(tip));
            ___________________________________________
        } catch (NumberFormatException nfe) {
            // pop up an alert view here (optional)
        }
    }

}
```

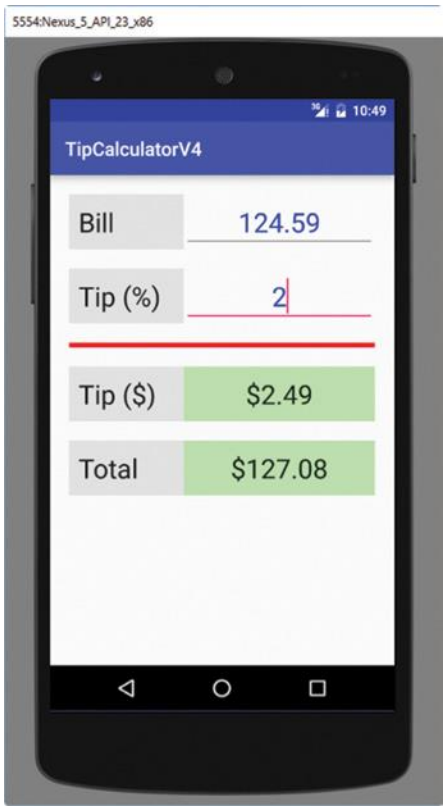
Grading:

1. Submit the project in zip file (extension zip)
2. A pdf document that contains the content of the following files: AndroidMainifest.xml, activity_main.xml, themes.xml, and MainActivity class.
3. A zoom link or YouTube that demonstrate the user of the TipCalculator app (less than 3 minutes). Write the link on the comment section in the Dropbox.
4. Write full names of the team members
5. Write down the contribution of each team members. All the team members must agree to the contribution of each team member.

Note:
App is running correctly and met all the requirements is 30 points, but
Incomplete item 1(-30 points)
Incomplete item 2 (-5 points)
Incomplete item 3 (-10 points)

Version 3 – Event Handling for Keyboard Input [10 points]

We should update the tip and total amount any time the user changes the data (i.e., any time the user is typing on the keyboard) even if the user does not click on the Calculate button. In fact, we do not even need the Calculate button.



Modify the activity_main.xml file to delete the `Button` element.

We have already learned that clicking on a `Button` is an event. Typing inside an `EditText` component, or more generally pressing a key, is also an event. The Android framework provides developers with tools that alert us whenever an event happens or something is changing or has changed inside a GUI component.

Generally, to capture and process an event, we need to do the following, in order:

1. Write an event handler (a class extending a listener interface).
2. Instantiate an object of that class.
3. Register that object on one or more GUI component.

The TextWatcher interface, from the android.text package, provides three methods that are called when the text inside a GUI component (a TextView or an object of a subclass of TextView, such as EditText)

changes, assuming that a TextWatcher object is registered on the TextView.

Methods of the TextWatcher interface

Method	When Is the Method Called?
void afterTextChanged (Editable e)	Somewhere within e, the text has changed
void beforeTextChanged (CharSequence cs, int start, int count, int after)	Within cs, the count characters beginning at start are about to be replaced with after characters
void onTextChanged (CharSequence cs, int start, int before, int count)	Within cs, the count characters beginning at start have just replaced before characters

In order to have a TextWatcher be notified of any change in the text inside a TextView, we need to do the following:

- Code a class (also called a handler) that implements the TextWatcher interface
- Declare and instantiate an object of that class
- Register that object on the TextView (in this app the two EditTexts)

In our handler class, we are only interested in the `afterTextChanged` method. We do not really care how many characters have changed and where within the text. We only care that something has changed within the text. So we will implement the other two methods, `beforeTextChanged` and `onTextChanged,` as "do nothing" methods. Because we need to access the two `EditTexts` and the two `TextViews`, we choose to implement the handler class as a `private`, inner class of `MainActivity`. When coding a handler, we can also implement it as a separate `public` class if we choose to.

MainActivity class

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import java.text.NumberFormat;

public class MainActivity extends AppCompatActivity {
    private TipCalculator tipCalc;
```

```
public NumberFormat money = NumberFormat.getCurrencyInstance();
private EditText billEditText;
private EditText tipEditText;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    tipCalc = new TipCalculator( 0.17f, 100.0f );
    setContentView(_____________________________);
    billEditText = (EditText) findViewById(R.id.amount_bill);
    tipEditText = _____________________________________________);
    TextChangeHandler tch = ___________________________________
    billEditText.addTextChangedListener( tch );
    tipEditText._____________________________________________

public void calculate()
{
    String billString = billEditText.getText().toString();
    String tipString = _________________________________

    TextView tipTextView =
            (TextView) findViewById(R.id.amount_tip);
    TextView totalTextView =

            ____________________________________________
    try {
        // convert billString and tipString to floats
        float billAmount = _____________________________
        int tipPercent = ___________________________________
        // update the Model
        tipCalc._____________________
        tipCalc.setTip(.01f * tipPercent);
        // ask Model to calculate tip and total amounts
        float tip = _____________________
        float total = ____________________
        // update the View with formatted tip and total amounts
        tipTextView.setText(_____________________________);
        totalTextView.setText(_____________________________________);
    } catch (NumberFormatException nfe) {
        // pop up an alert view here
    }
}
private class TextChangeHandler implements TextWatcher {
    ____________________________________ {
        _____________________________
    }

    public void beforeTextChanged( CharSequence s, int start,
                                   int count, int after ) {
    }

    public void onTextChanged( CharSequence s, int start,
                               int before, int after ) {
    }
}
```

```
}
```

Grading:

1. Submit the project in zip file (extension zip)
2. A pdf document that contains the content of the following files: AndroidMainifest.xml, activity_main.xml, themes.xml, and MainActivity class.
3. A zoom link or YouTube that demonstrate the user of the TipCalculator app (less than 3 minutes). Write the link on the comment section in the Dropbox.
4. Write full names of the team members
5. Write down the contribution of each team members. All the team members must agree to the contribution of each team member.

Note:
App is running correctly and met all the requirements is 10 points, but
Incomplete item 1(-10 points)
Incomplete item 2 (-2 points)
Incomplete item 3 (-5 points)