**455-555 Week 2 Notes**

**This week’s Chapter Readings:**

**Chap. 2** - More on getting started

**Chap. 3-4** - APPLICATIONS, ACTIVITIES

**Categories to overview this week:**

Android Updates ([Gradle](https://developer.android.com/studio/releases/gradle-plugin#updating-gradle), SDK!)

AVD “tweaks”

App project designs (working various text views)

Activity Life Cycles (see ppt in PowerPoints module, check also Week 2 Demo Module)

**Applications:**

Payroll app sample (UI design/Coding of the app, see Week 2 Demo folder)

Lab. 1 overview

------------

Note always consider Android updates!

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a phone

Description automatically generated**

[ **Application** creation in Android Studio]

Create a payroll application. Allow for three views (**2** EditText views and **1** TextView)

Each EditView (shown as Number in your Text section of your Palette) should either represent **hours** as an id and one as **rate** as an id. Your TextView (Plain Text element) should represent **result** as an id.

View modes

Design Surface toggle modes

Here is setup so far

A screenshot of a cell phone

Description automatically generated

Layout zoom

Various text views dragged in order from pallet in design view mode

Here is xml code (in Code mode) thus far

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.ConstraintLayout   
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**EditText  
 android:id="@+id/hours"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="151dp"  
 android:ems="10"  
 android:inputType="textPersonName"  
 android:text="Hours"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />  
 <**EditText  
 android:id="@+id/rate"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="53dp"  
 android:ems="10"  
 android:inputType="textPersonName"  
 android:text="Rate"  
 app:layout\_constraintStart\_toStartOf="@+id/hours"  
 app:layout\_constraintTop\_toBottomOf="@+id/hours"** />  
 <**TextView  
 android:id="@+id/result"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="60dp"  
 android:text="TextView"  
 app:layout\_constraintEnd\_toEndOf="parent"**

**app:layout\_constraintHorizontal\_bias="0.0"**

**app:layout\_constraintStart\_toStartOf="@+id/rate"  
 app:layout\_constraintTop\_toBottomOf="@+id/rate"** />  
  
</**androidx.constraintlayout.widget.ConstraintLayout**>

Here is the UI at run time

Graphical user interface, application

Description automatically generated

Now it doesn’t look like much, we will need to add a button in for an action to occur namely calculate the payroll (**hours\*rate**) when the user clicks on the button and have the result appear in the TextView result field.

Run your app and you should see a nice UI as depicted before with all your constraints set!

Now to finish up your UI for a real professional look and feel. Mod your **TextView** tag as follows

1. Remove the tag attribute that states

android:text = “TextView”

2. For your first two text views (id- hours, id-rate) add in the following tag attributes in for the appropriate controls

a. For first EditText view (with **id** as hours) add in the attribute/value pair as follows:

android:hint=”Enter hours”

b. For the second EditText view (with **id** as rate) add in the attribute/value pair as follows:

android:hint=”Enter rate”

c. For your finalized work, add in a button at the bottom of the UI and make sure it seats itself well in the layout and add in the following attribute/value pairs

android:text = “Calculate”

android:onClick="Calculate"

[ Note you may see the bulb on the left side of your attribute (shown below) which will allow you to create a function in MainActivity automatically, leaving you just to implement your source code. Go ahead if you like and check out the added stub placed into your MainActivity.java file!]

Text

Description automatically generated

For your MainActivity file you want to have a separate function to calculate the payroll for the user.

Code your function (you can always just hard code the name if the method stub wasn’t already created for you), called Calculate as follows:

**void Calculate(View v)**

**{**

**EditText h = findViewById(R.id.hours);**

**EditText r = findViewById(R.id.rate);**

**Double gross\_pay = Double.parseDouble(h.getText().toString()) \***

**Double.parseDouble(r.getText().toString());**

**TextView result = findViewById(R.id.result);**

**String formatted = String.format("$%, .2f", gross\_pay);**

**result.setText("Your payment is => " + formatted);**

**}**

Run your app and test away!

Sample runs follow:

Before

A picture containing table

Description automatically generated

After

Graphical user interface, application

Description automatically generated

Note if your text result is not wide enough to display the full information, then widen your width attribute as follows:

<TextView  
 android:id="@+id/result"  
  **android:layout\_width="200dp"**  
 android:layout\_height="64dp"

::

The following code shows a modded MainActivity:

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

public void Calculate(View view) {

EditText h = findViewById(R.id.hours);

EditText r = findViewById(R.id.rate);

Double gross\_pay = Double.parseDouble(h.getText().toString()) **\***

Double.parseDouble(r.getText().toString());

TextView result = findViewById(R.id.result);

String formatted = String.format("$%, .2f", gross\_pay);

result.setText("Your payment is => " + formatted);

}

}

See if you can further mod your app by including a class that will calculate the payroll and have a class method that returns the payroll result to your TextView!

Lab 1 overview

- Observances

- Use of Import statements, widgets etc.

- **Use of instance variables (visibility important!)**

- XML Files

- Layout file (activity\_main), attributes, working with const. layout

- Code vs Design view modes

- Palette (for choice elements)

- Component tree

- strings.xml (gen. use purposes?)

Grad requirement=>Working with images.

A close up of a logo

Description automatically generated

Note constraint settings ht/wid for image?

Handling errors? Troubleshooting tips?

A screenshot of a cell phone

Description automatically generated

**Build > Clean Project**,a good idea always!!

Regenerate APK (from **Build** menu)

**Other**:

Adding a title to your app

You may "tweak" your UI themes.xml file for Hedgehog under res/values/themes to show your title at runtime. Change the style theme base for the parent attribute.

<resources xmlns:tools="http://schemas.android.com/tools">  
 <!-- Base application theme. -->  
 <style name="Base.Theme.TempConverter" parent="Theme.AppCompat.Light">  
 <!-- Customize your light theme here. -->  
 <!-- <item name="colorPrimary">@color/my\_light\_primary</item> -->  
 </style>  
  
 <style name="Theme.TempConverter" parent="Base.Theme.TempConverter" />  
</resources>

Adding Hints for a given text view

<EditText

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:hint="Your hint text here" />

Consider doing this logic programmatically

TextView textView = findViewById(R.id.yourTextViewId);

textView.setHint("Your hint");

Weblinks of interest

<https://www.wired.com/search/?q=android%2012&page=1&sort=score>