**NIT6120 MOBILE APPLICATION**



Assignment 1- Application with Multiple Application

Submitted to : John Denison

Submitted By: Surbhi Dhawan (s4608520)

**Table of Contents**

[**1.** **Introduction** 3](#_Toc40533591)

[**2.** **Task 1** 3](#_Toc40533592)

[**Layout** 4](#_Toc40533593)

[**3.** **Screenshots** 5](#_Toc40533594)

[**Task 2** 9](#_Toc40533595)

[**Test your app** 10](#_Toc40533596)

# **Introduction**

This application is based on Android Studio Application Version 3.6.2

The task is explained as follows:

Create an app. that will offer two features:

1. An employee’s total weekly pay equals the hourly wage multiplied by the total number of regular hours plus any overtime pay. Overtime pay equals the total overtime hours multiplied by 1.5 times the hourly wage. Write an app that takes as inputs the hourly wage, total regular hours, and total overtime hours and displays an employee’s total weekly pay.
2. a program is written that accepts the lengths of three sides of a triangle as inputs. The program output should indicate the type of a triangle – Equilateral triangle, Isosceles triangle or Scalene Triangle.

# **Task 1**

There are three activities as given in document…

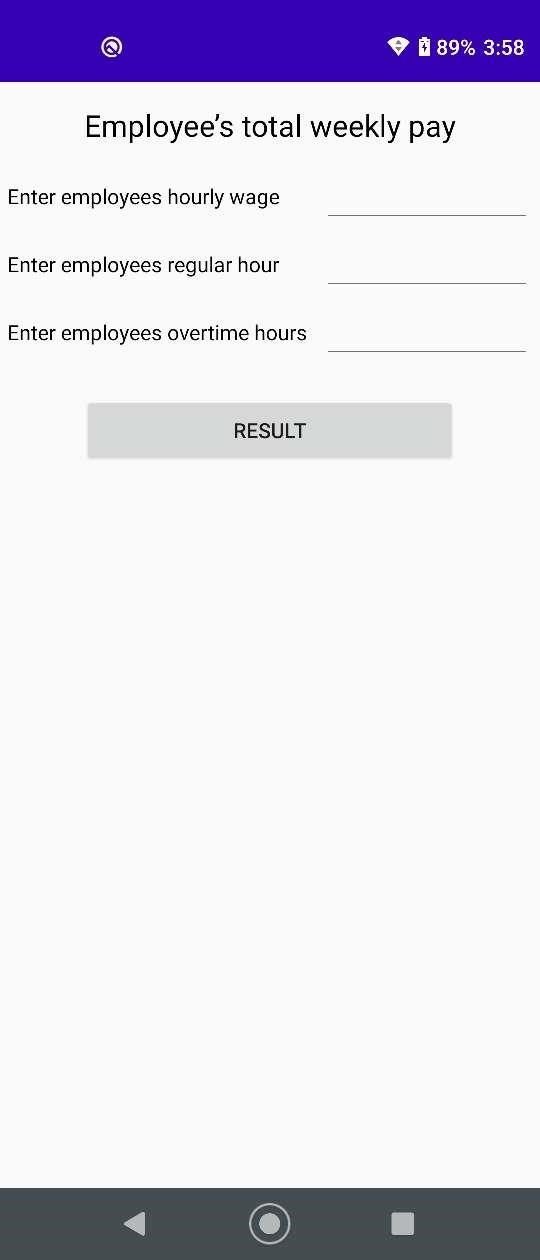
MainActivity.class(we can go on next classes from buttons click)

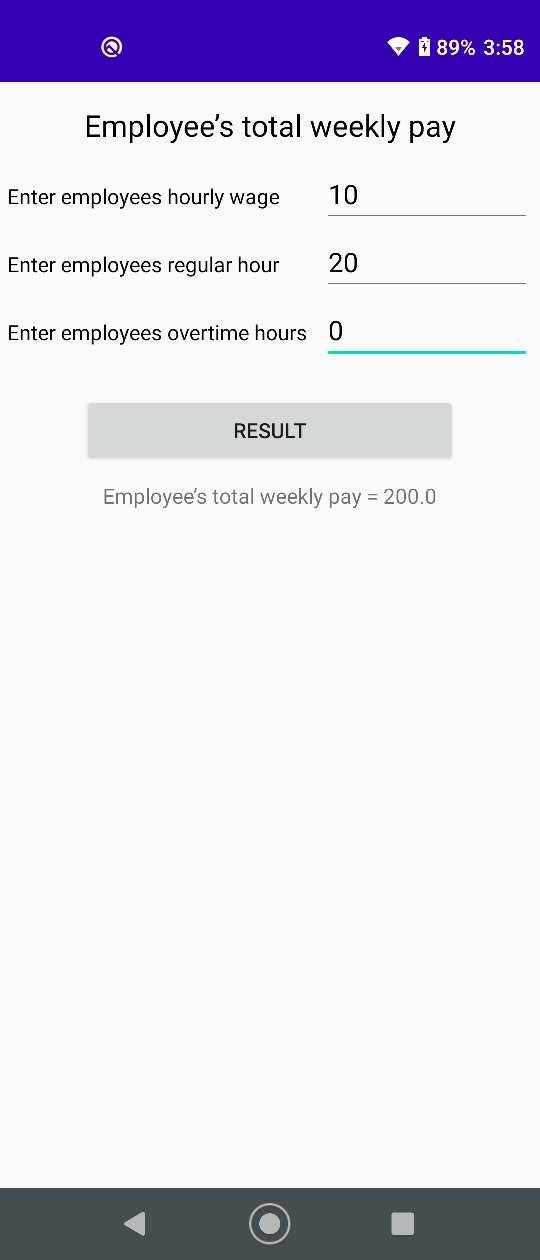
PayActivity.class(To find out weekly pay) //All fields are manadetory

//Condition **weeklyPay** =((hourlyWage\*regularHour)+(1.5\*overTimeHour\*hourlyWage)); Triangle.class (To find out triangle type) //All fields are manadetory

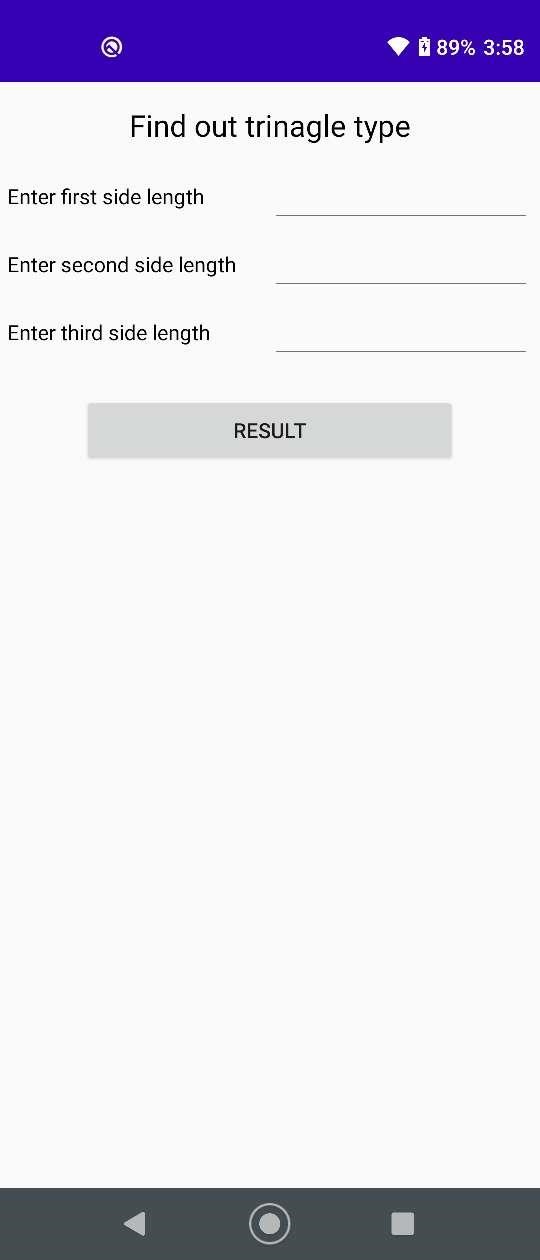
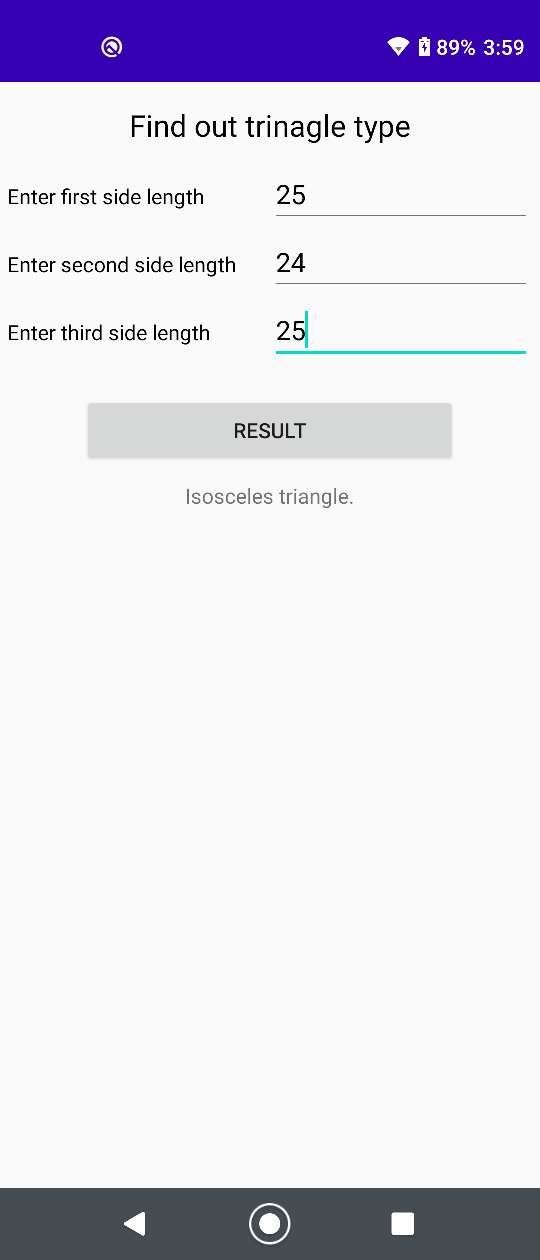
# **Layout**

Figure1 : Main Page **Layout Figure 2: Weekly page activity**



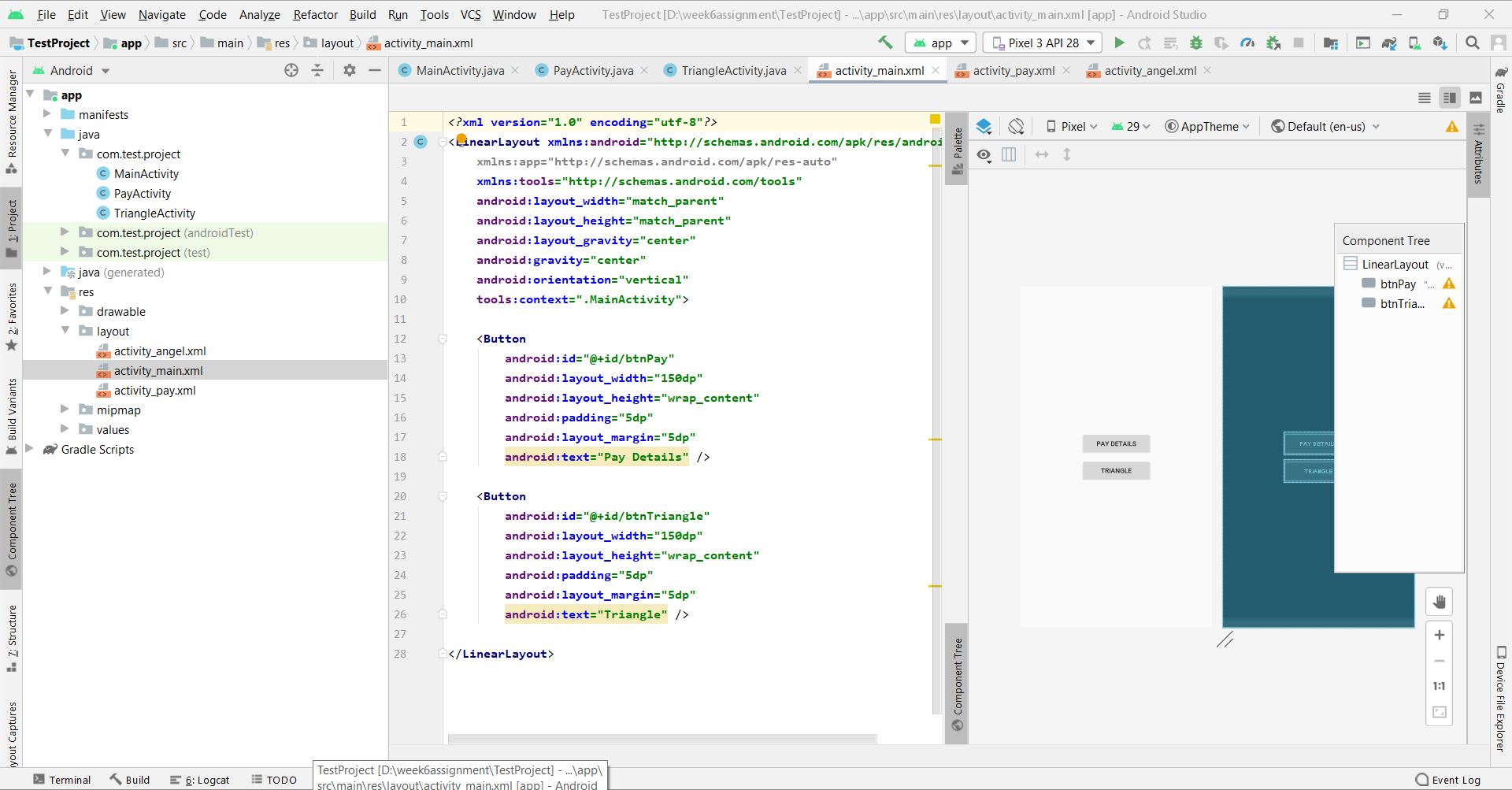


**Figure 3: Result of pay**

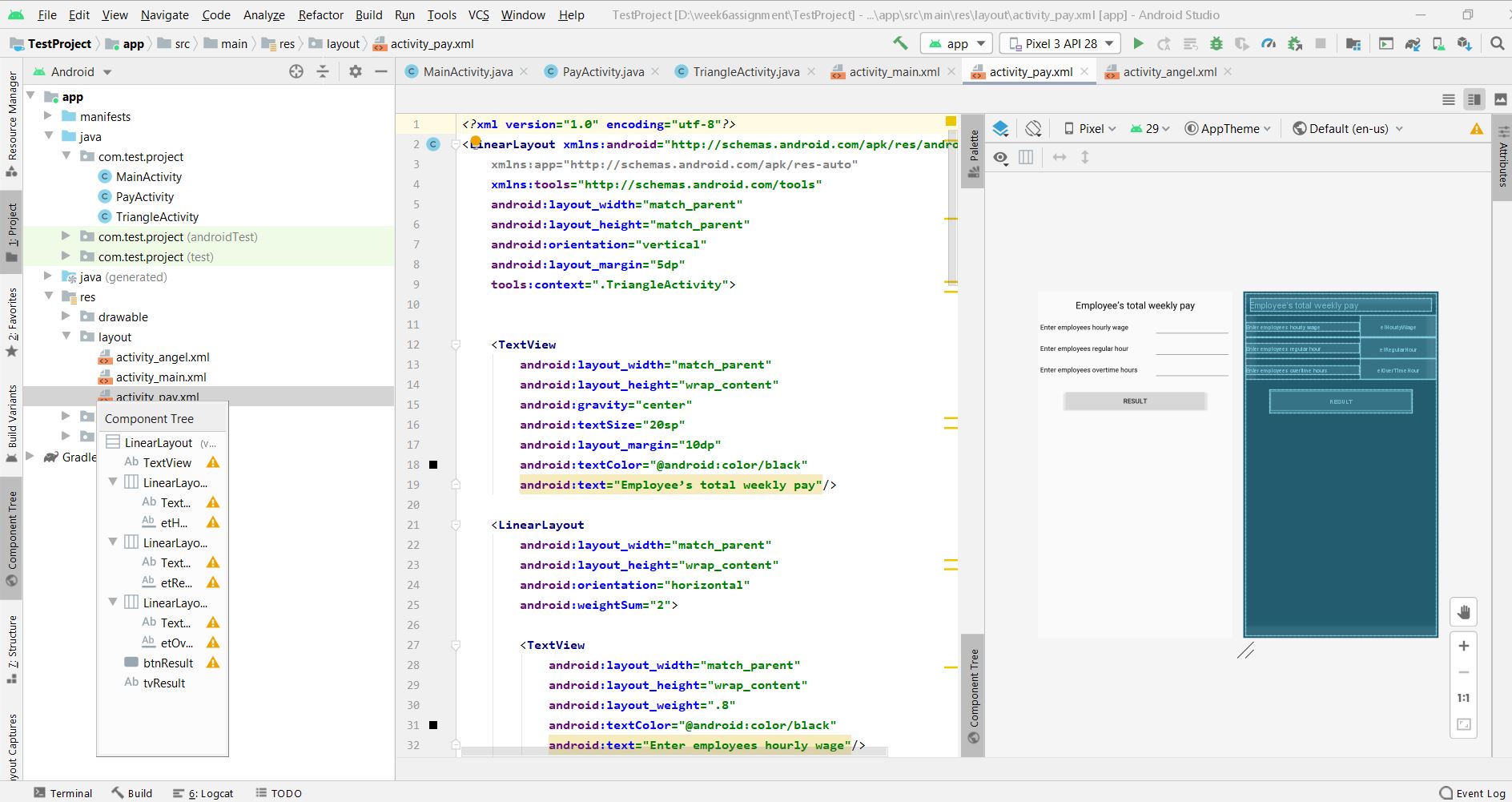


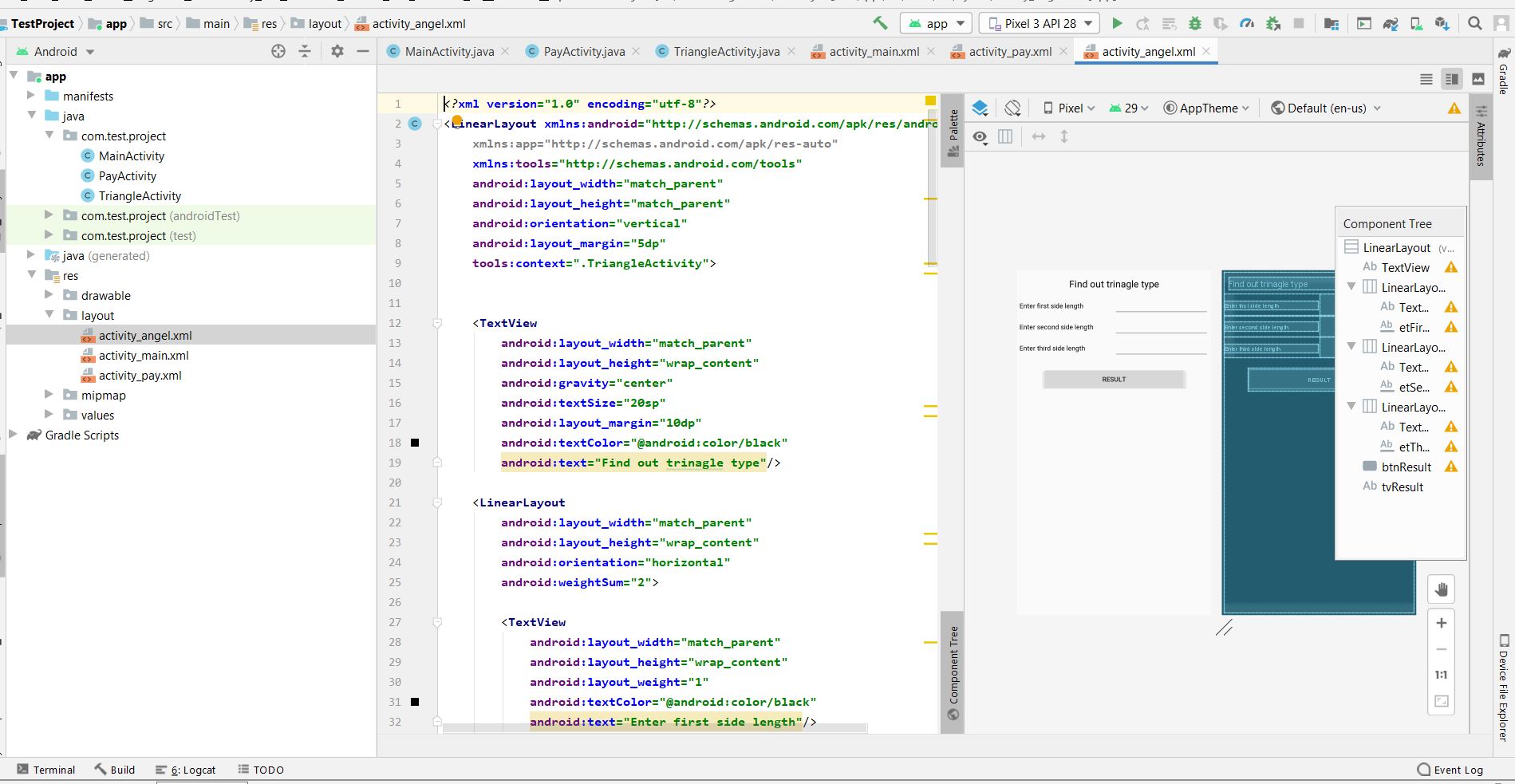
**Figure 4 : Triangle Activity Figure 5 : Result of Triangle**

# **Screenshots**

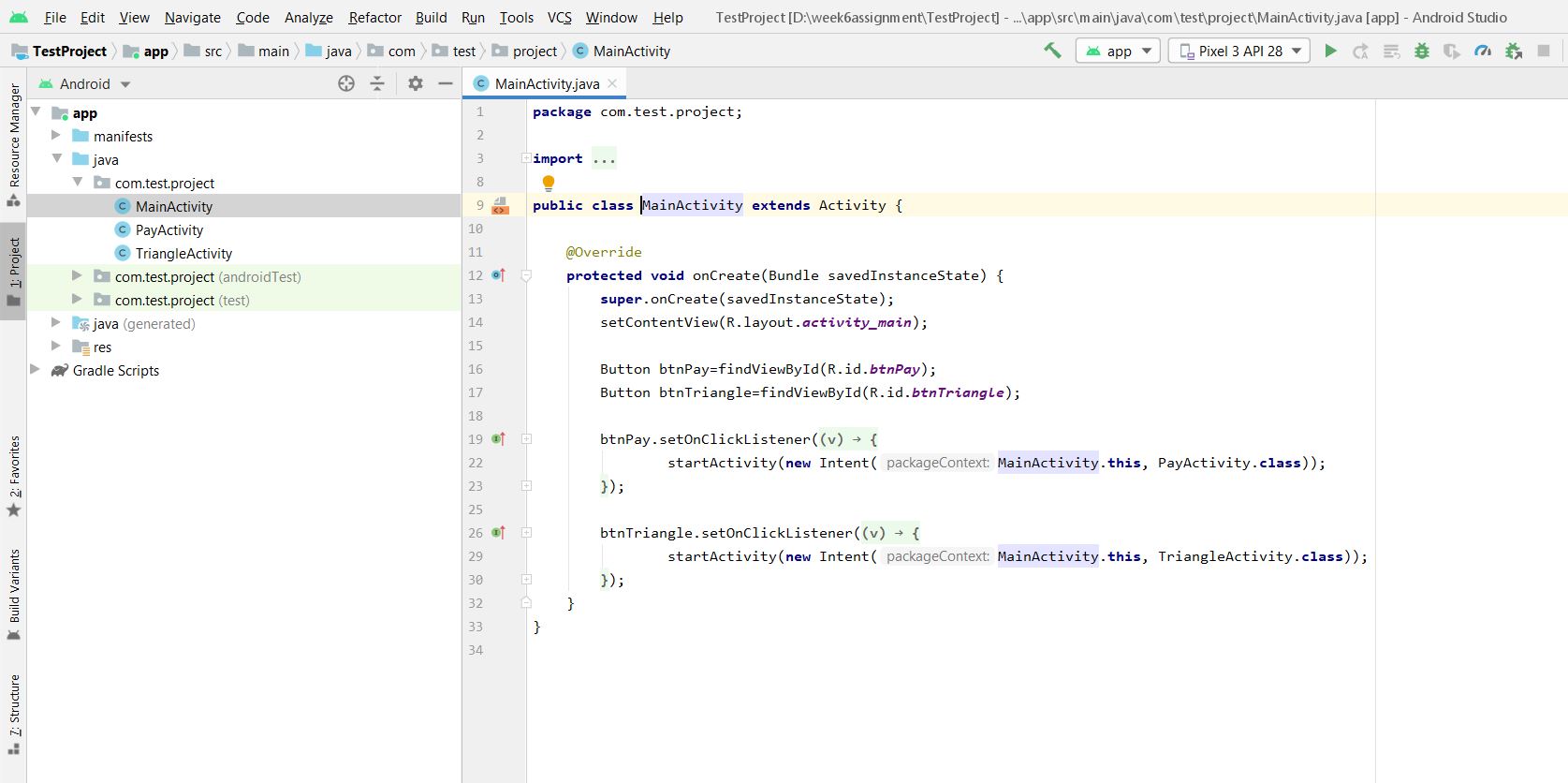


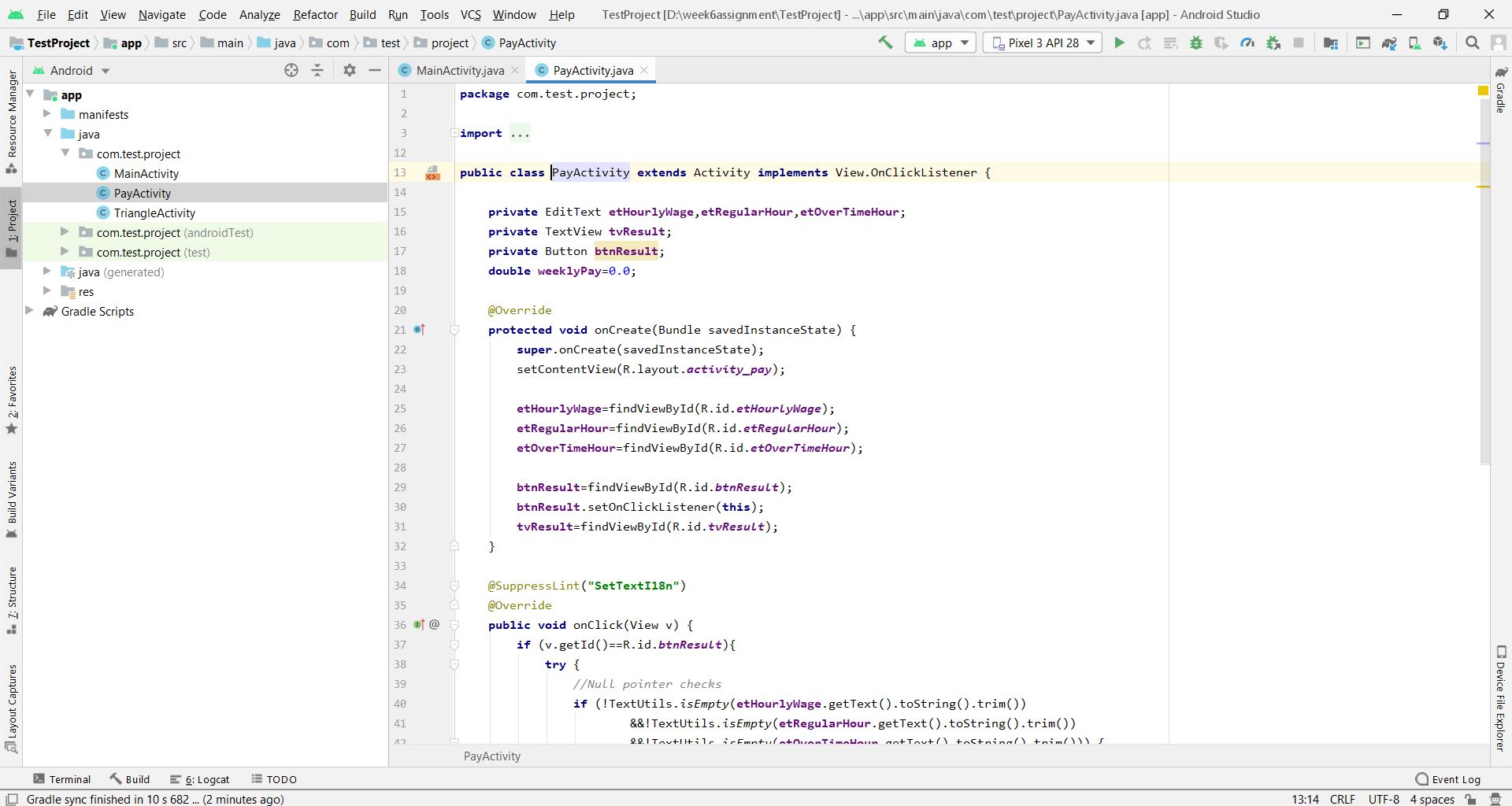
**Figure 6: Activity Main Xml**

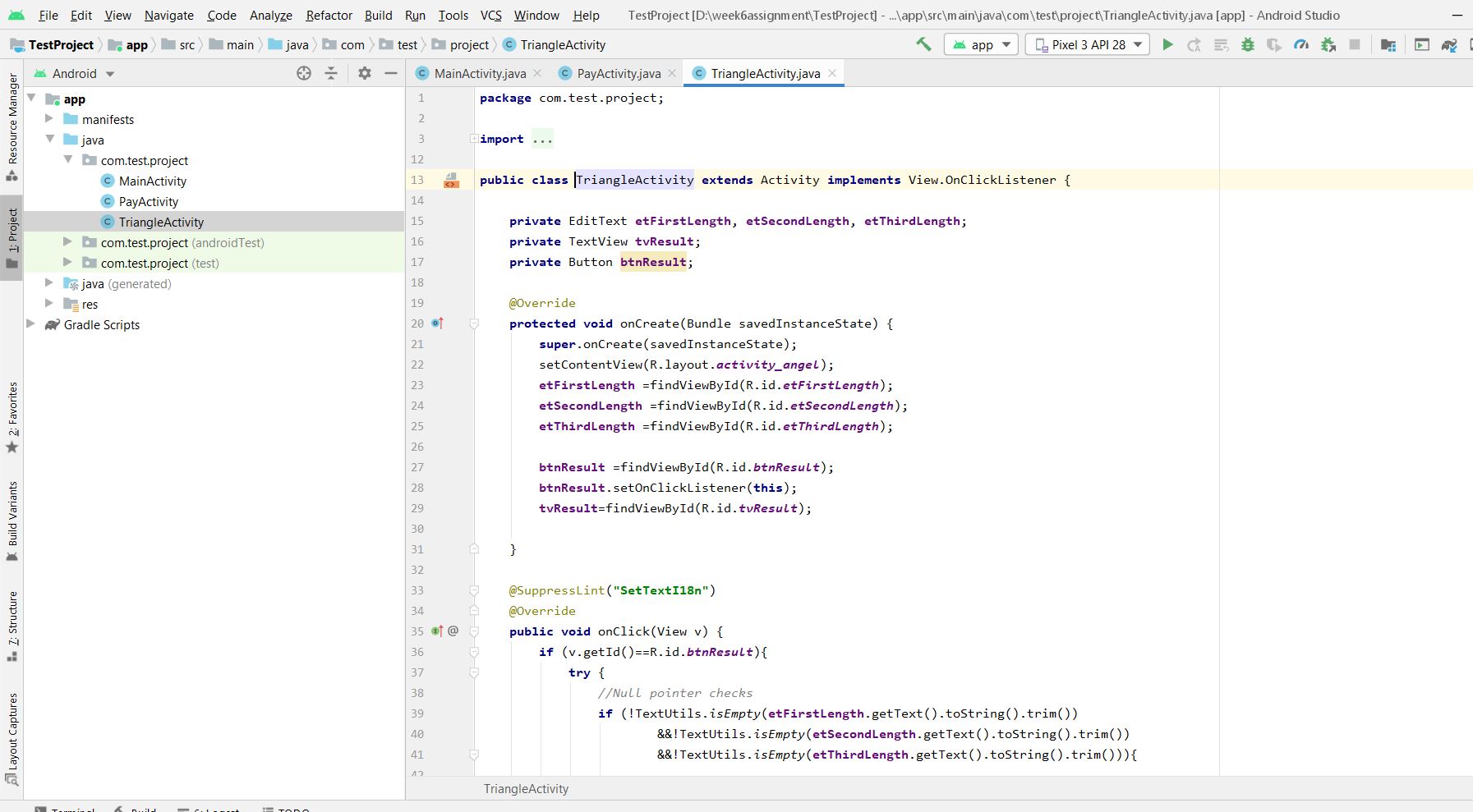
**Figure 7 : Activity pay xml**

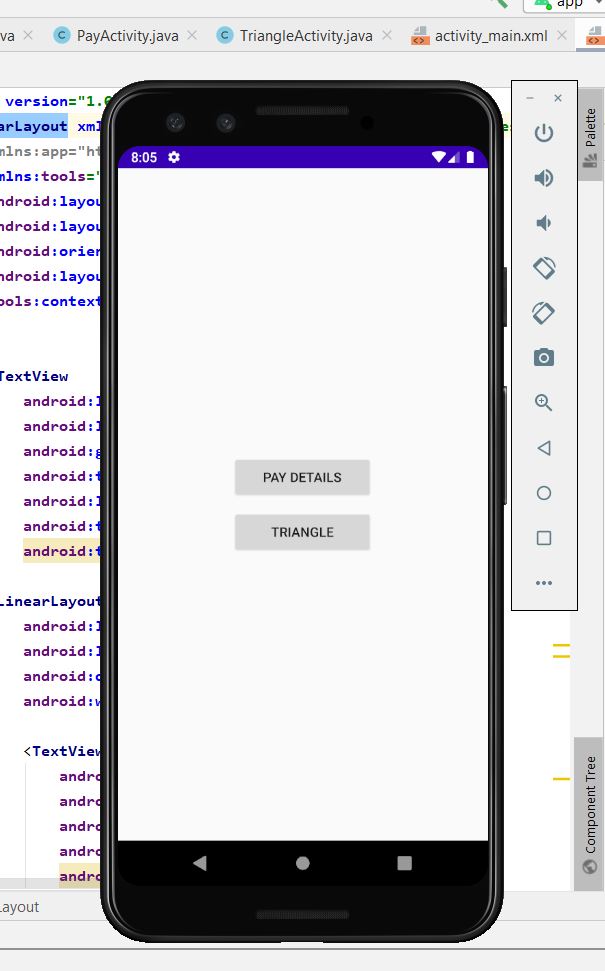
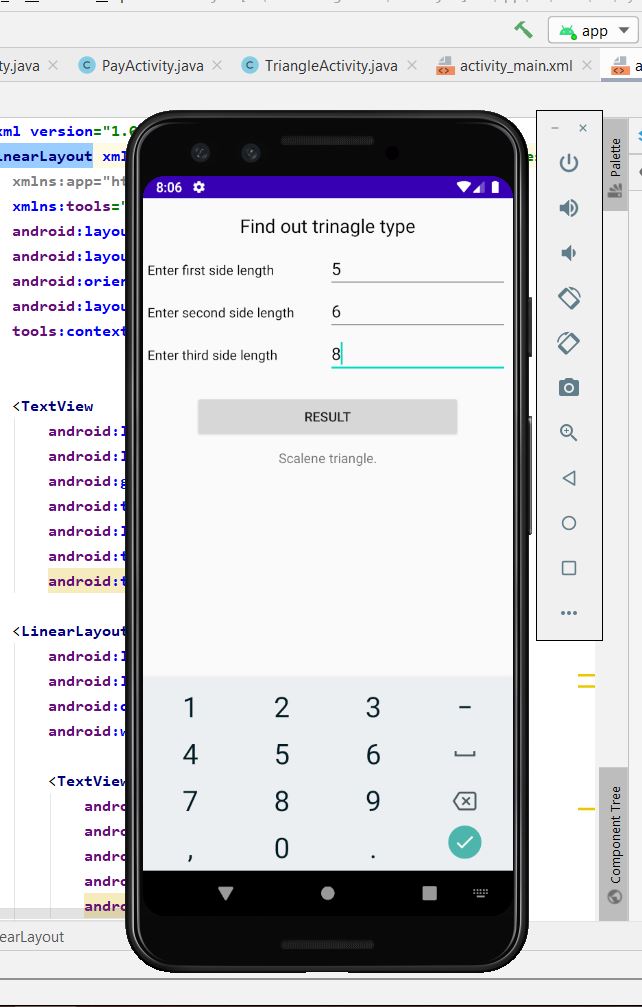


**Figure 8 : Activity Triangle xml**

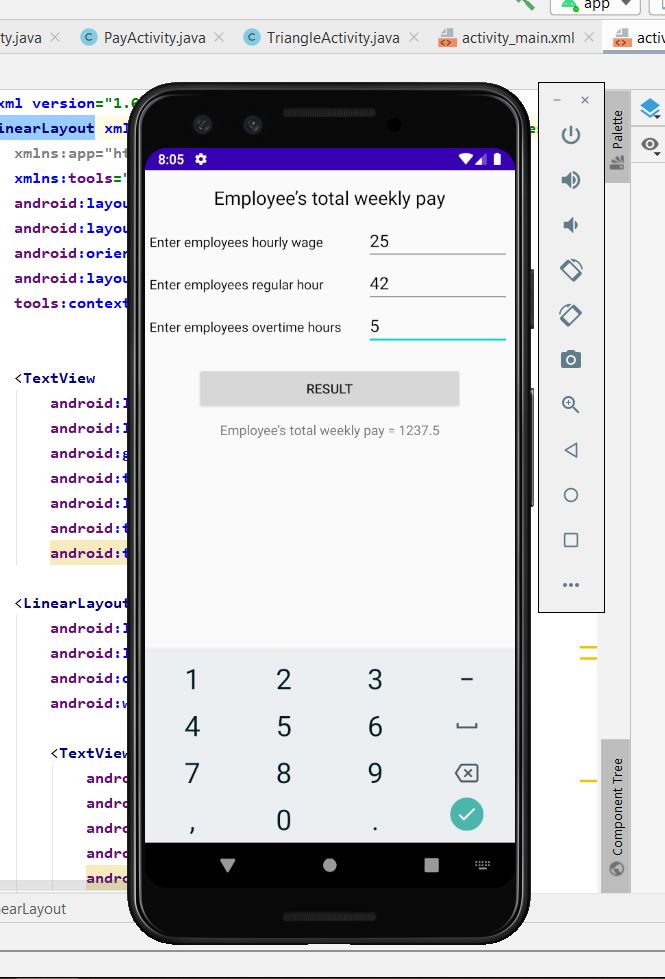
**Figure 9: Main Java Activity**

**Figure 10 : Pay Activity Java**

**Figure 11 : Triangle Activity Java**



**Figure 12 : Emulator Main Figure 13 : Emulator Triangle Result**



**Figure 14: Emulator Pay Activity**

# **Task 2**

**Debug an Android Application**

It is the most part of developers everyday work. To run an application successfully we need to fix every error and every bug from minor to major. Debugging helps us to perform and run error free code by scanning each line, evaluating all variables and method. By debugging it is very easy to find small errors in big code.

**Start Debug Mode**

To start debugging you need to get ready with device setup for debugging, USB connected and an opened project in Android Studio. Choose a device from device window and launch application in the mode of debug. It will automatically on in debug mode or you can select from debug button in bottom of the window. Another approach to begin investigating without restarting your application is by tapping on "Attach debugger to Android process".

After that select the gadget and app to attach to the debugger. This is a special feature if you want to run multiple screen.

**Start Debugging**

* Set breakpoints in the code of application.
* Select the device from the toolbar from drop down menu.
* In toolbar, you can click on debug
* If your app is already running on debugging mode then you can continue withour restaring.
* If you can’t see debug window, select **View > Tool Windows > Debug** (or click **Debug**  in the tool window bar), and then click the **Debugger** tab.
* Click **Attach debugger to Android process**
* In dialog box of choose Process, we need to select the process
* From Debugger you can choose the type of Debugger: Auto, Java, Native based on your project.
* The debug window appears at final stage.

# **Test your app**

Android studio has special feature of simple testing. In few simple steps you can perform unit test or instrumental test.

**Types of Testing**

1. **Local Unit Test:** These test run on the machine called Java virtual Machine (JVM). These tests are used to decrease the execution time when it has no Android framework dependencies.
2. **Instrumental Test :** Testing to be done on genuine gadget or in emulator, clearly the slower system rather than Unit Tests in light of the fact that another test APK is made each time we play out a test on gadget. Instrumented tests can be performed to computerize choice on any catch, composing some content, looking through perspectives, or performing different activities in your application that solitary a real client can perform physically. Most generally, we execute instrumented tests for UI testing.

**Start A Test**

Following is the process to create a new test for a class or method, either by using Unit test or Instrumented Test.

1. Firstly, open a file to test.
2. Click the class or method, and then press **“Ctrl+Shift+T”>** Click **Create New Test.**
3. In the **Create Test** window, select any methods to generate.
4. Choose the destination directory, then select the source set corresponding to the type of test we are creating:

* **Android Test** for an instrumented test.
* **Test** for a local unit test

1. In this way m number of new tests can be created any time while testing the Android Application**.**