Wicaksa Munajat
CST 338 - Software Design Fall ‘21 (CSUMB)
Module 8 - Final Project Spec
04/22/2020

Simple Android Application: Calculator and Number Generator

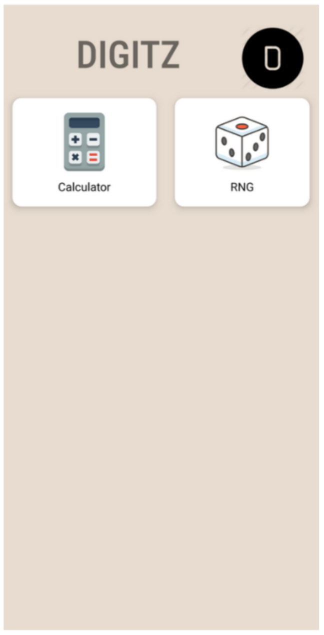
Understanding the assignment: In this assignment, we will be creating an **Android application** called **Digitz** that will contain a **simple calculator** and a **random number generator.** We will be creating a home directory containing two icons that can be clicked and opened into a new page for each section. In order to do this assignment we will need the latest version of **Android Studio.**

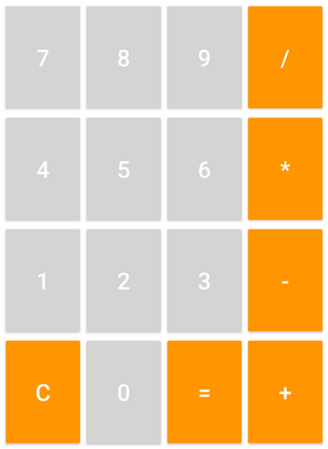
This assignment will be divided into different phases:

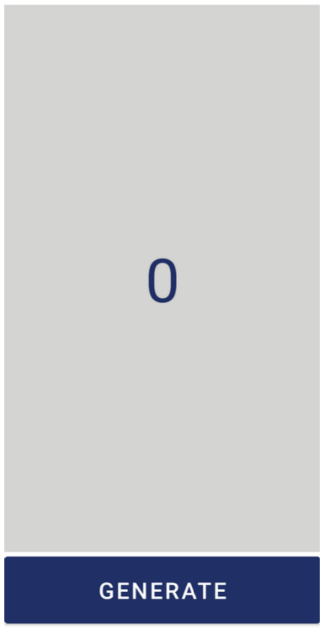
1.0 Setting up the home screen (xml)
1.1 Setting up the home screen (.java)
2.0 Setting up the calculator (xml)
2.1 Setting up the calculator (.java)
3.0 Setting up the random number generator (xml)
3.1 Setting up the random number generator (.java)
4.0 Running the program
4.1 Debugging

Here’s the final product of the application for each screen that we are going to do.

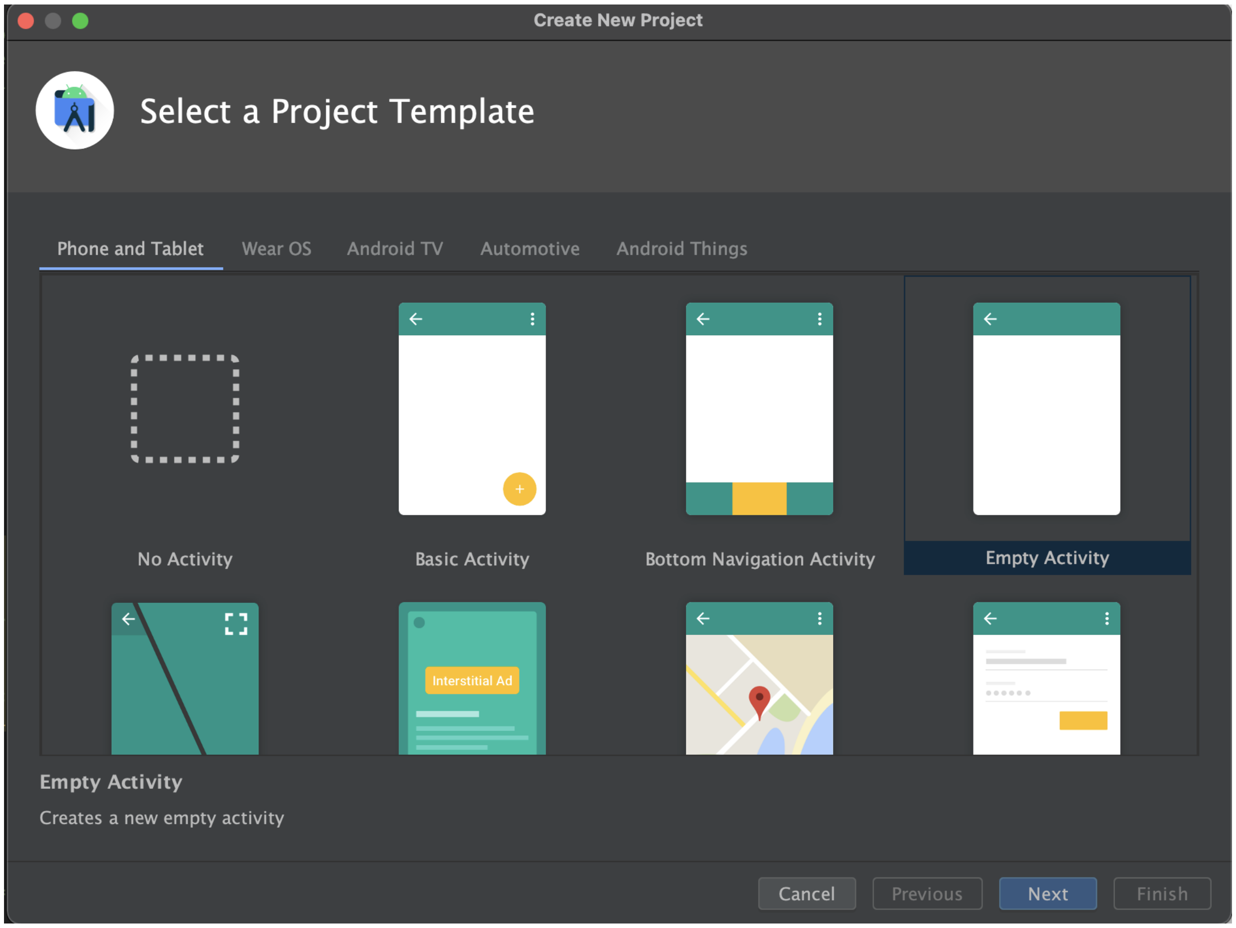
Home Screen **Calculator Screen** **Random Number Generator**



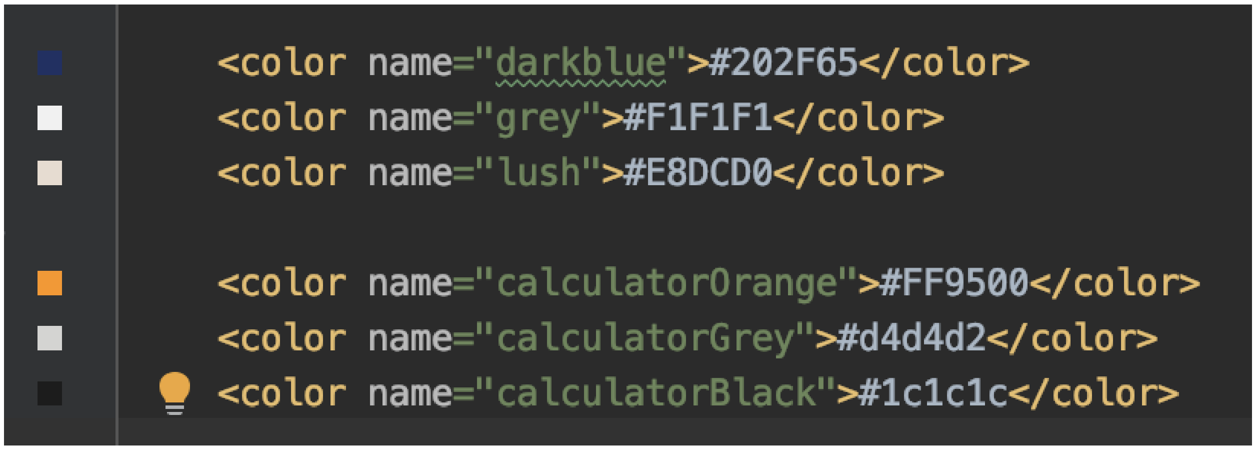




<u>1.0 Setting up the home screen (xml)</u>Create a new Android project called DigitzApp and select **Empty Activity.** Go to your **activity_main.xml** file, which will be the file that will turn into your home screen.



The first thing you want to do is import some colors into the **colors.xml** file so that we have some to choose from when we are creating our application. Here are the colors that you need to add:



```
    <color name="darkblue">#202F65</color>
    <color name="grey">#F1F1F1</color>
    <color name="lush">#E8DCD0</color>

    <color name="calculatorOrange">#FF9500</color>
    <color name="calculatorGrey">#d4d4d2</color>
    <color name="calculatorBlack">#1c1c1c</color>
```

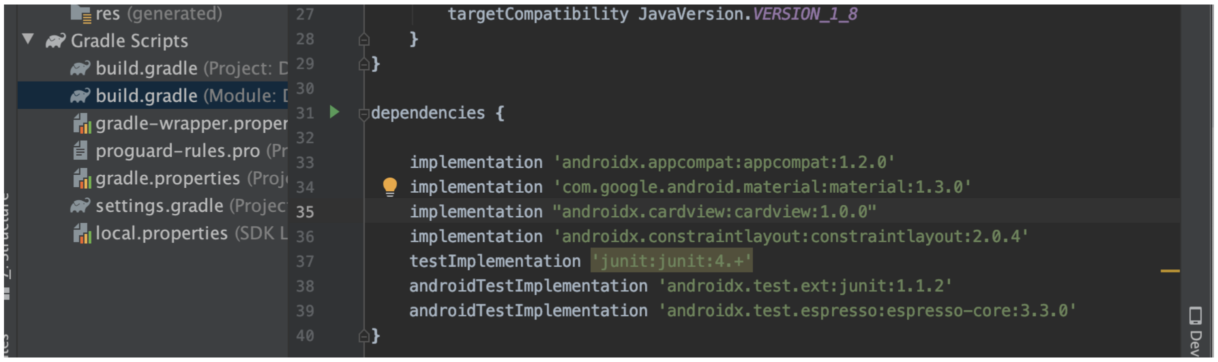
Next, we will be heading into the **activity_main.xml** file in order to start working on our main directory. Here are the things that you need to do:

View

For this project, we will be using the **ScrollView,** which will allow the view hierarchy within it to be scrolled.
Read more :https://developer.android.com/reference/android/widget/ScrollView

Layouts

The layout for this project is a mixture of **CardView, LinearLayout, RelativeLayout, and a GridLayout.** Use this information to create a layout grid that has **2 columns** and **1 row** in order to have the icons side by side in the example. While the last three layouts are available in Android Studio by default, **CardView** will not since it is a **widget** provided by **Android X**. In order to use this widget, you will need to go into your app module's **build.gradle** and add the following dependency:

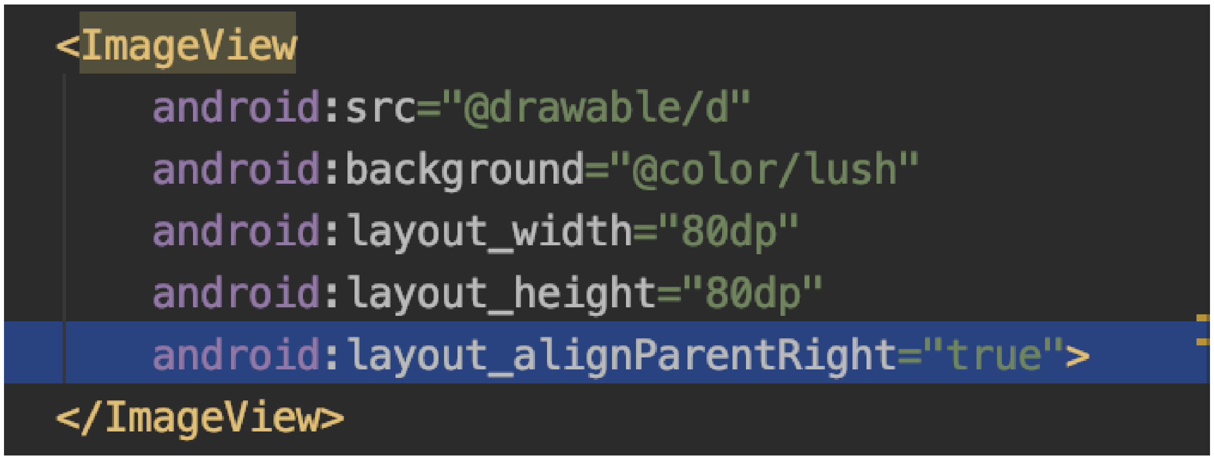


```
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation "androidx.cardview:cardview:1.0.0"
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

Read more: https://developer.android.com/guide/topics/ui/layout/cardview

ImageView

You will have to create **three (3) ImageView** objects to represent each icon on the page. You will need to import **calculator.png, d.png, and dice.png** into the **drawable** folder. When creating your **ImageView** to show the icons, use **android:src="source"** tag to add the icon file name. Here is an example for the "d" image on top of the page.



```
<ImageView
    android:src="@drawable/d"
    android:background="@color/lush"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_alignParentRight="true">
</ImageView>
```

TextView

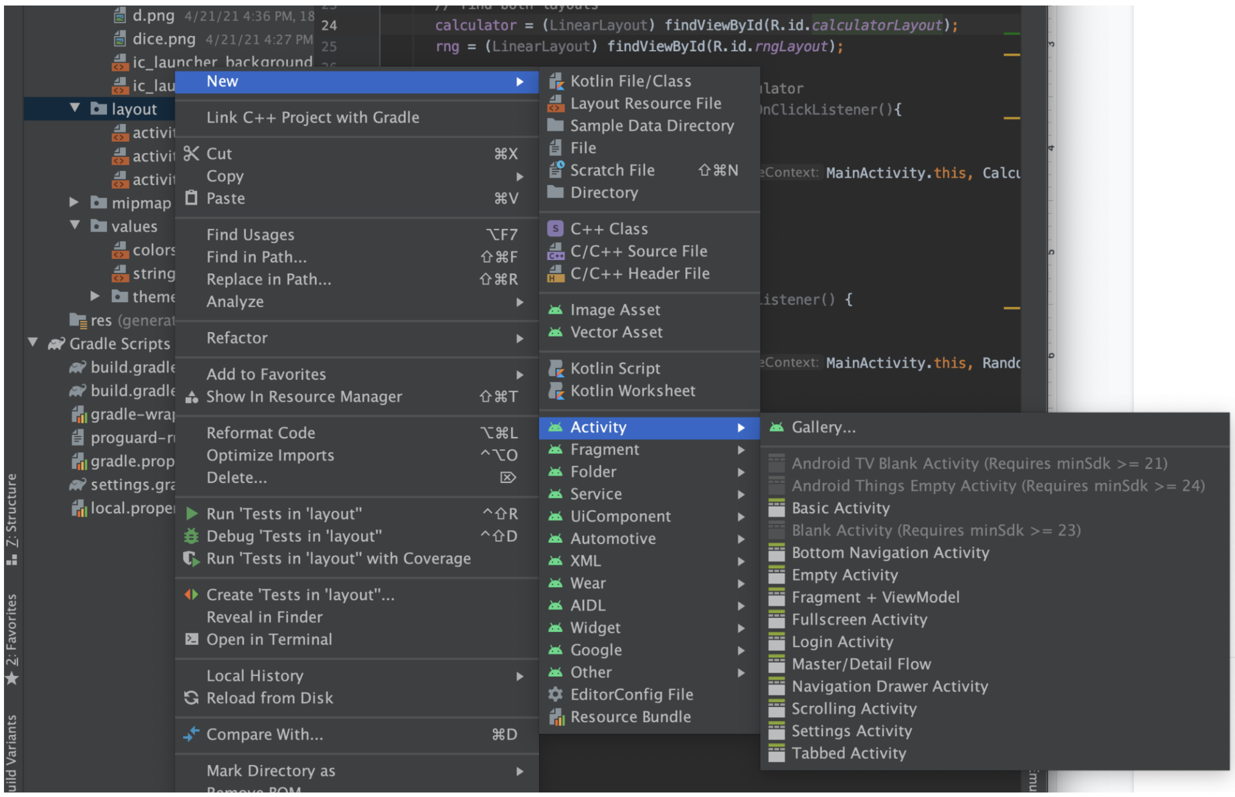
Create **three (3) TextView** objects that will be the **title of your page located on top of the page**, and the **title of the icons located underneath the ImageView.**

Background and Text Colors

The **background color of the page** should be the **"lush"** color that we imported. The **title of the icons** should be **black** while the **title** should be "**grey**".

Other

Add **id** tags to each of the items you've created so that you can reference them later. Create new pages for your calculator and random number generator. To do this, right click on layout -> New -> Activity -> Empty Activity. Create one called **Calculator** and one called **RandomNumberGenerator.**



```
calculator = (LinearLayout) findViewById(R.id.calculatorLayout);
rng = (LinearLayout) findViewById(R.id.rngLayout);
```

1.1 Setting up the home screen (.java)

In this portion, we are going to make our icons open up a new page when we click on them. To do this, we are going to apply an **onClickListener** to one the **LinearLayout** that houses both the calculator and dice icons. You will have to know the concept of **starting activity** which is explained below. Open your **MainActivity.java** to do this.

Read More: https://developer.android.com/training/basics/firstapp/starting-activity

2.0 Setting up the calculator (xml)

After you have completed the previous sections, you will now work on building your calculator. This simple calculator will only have two components: a TextView and Button. You must have these components in the project:

TextView:
A screen to show the numbers that have been pressed and the answer to the calculation performed. This will be placed on top of the screen. Make sure it doesn't take up too much (more than one third) of the page. Use your discretion to make it look nice.

Buttons:

- **Numerical buttons** (0-9)
- A **"Clear"** button to clear the screen,
- **Operation** buttons (+ - * /)
- **Equal** button that will perform your calculations when pressed.

Layout

This calculator will use a **LinearLayout** that will place each button in a 4x4 grid. You will have to use some thinking and **nesting** in your xml code in order to align these buttons properly. Here's an article to read on.

Read more: https://developer.android.com/guide/topics/ui/layout/linear

Colors

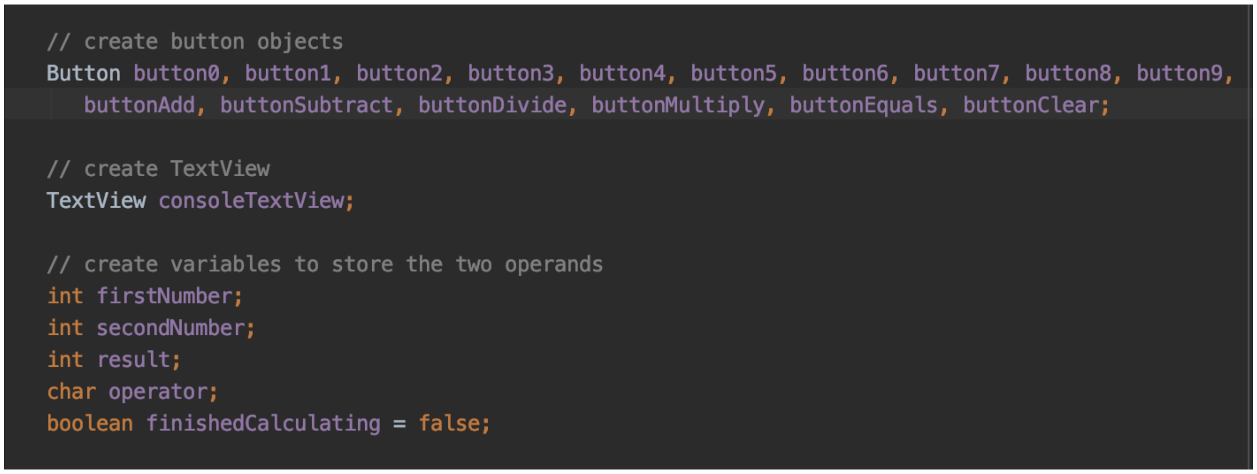
Color the **number** buttons **calculatorGrey**.
Color the **operations**, **clear**, and **equal** buttons **calculatorOrange**.

Naming

Add **id** tags to each of your items that you have created above to be referenced easily in the next portion.

2.1 Setting up the calculator (.java)

Declare these variables before starting.



```java
// create button objects
Button button0, button1, button2, button3, button4, button5, button6, button7, button8, button9,
    buttonAdd, buttonSubtract, buttonDivide, buttonMultiply, buttonEquals, buttonClear;

// create TextView
TextView consoleTextView;

// create variables to store the two operands
int firstNumber;
int secondNumber;
int result;
char operator;
boolean finishedCalculating = false;
```

Button button0 … buttonClear: You are going to create variables to represent all of the button objects you have made in the xml file.
TextView consoleView: This variable will store the TextView object you created to show the results.
int firstNumber: This variable will represent the first operand.
int secondNumber: This variable will represent the second operand.
int result: This variable will store the result of the calculation.
char operator: This variable will store the operation that will be performed.
boolean finishedCalculating: This variable will be true after you have done the calculation.

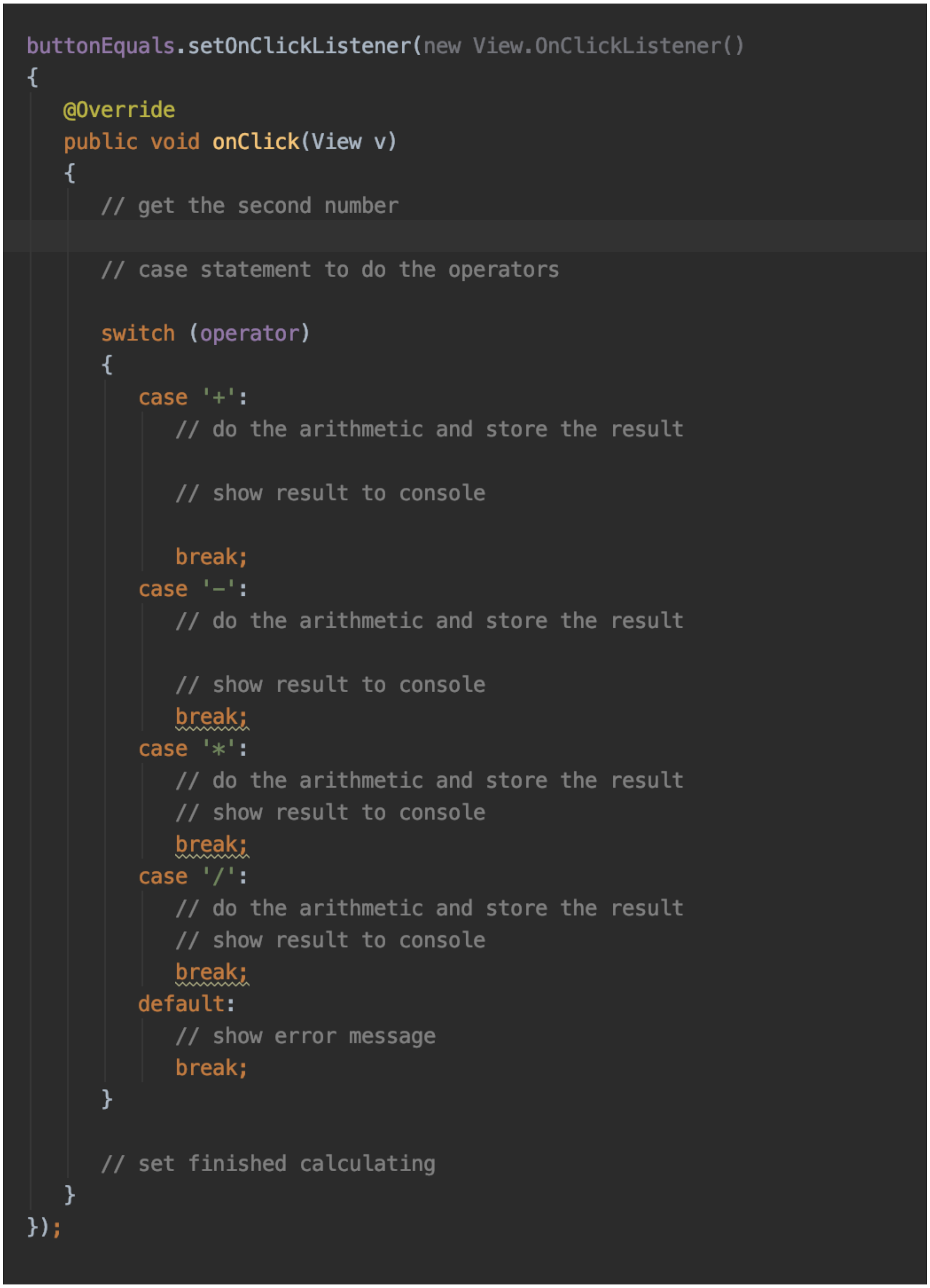
After setting up the variables you need to do the following. First, find each button id using **findViewById( )** to find each button in the calculator xml file and save them into the variable. Do the same for the **TextView**.

Next, create **onClickListeners( )** for **EACH** button. You will have to think about how you will be able to save each set of numbers to be added into the variables. Here's a couple of hints.

When the **operators** are pressed, they need to set the value of the **operator** to a char value such as **'+'**. In addition, they will need to store the value in the text box to **firstNumber**.

After typing the next number, you will need to hit **equals (**this app for now only supports two values being added hence a simple calculator).

The **equals** button will need to 1). store the second value into **secondNumber,** 2). perform the calculation, and 3). show the answer on the **TextView** Here's an example of the code.



```
buttonEquals.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        // get the second number

        // case statement to do the operators

        switch (operator)
        {
            case '+':
                // do the arithmetic and store the result

                // show result to console

                break;
            case '-':
                // do the arithmetic and store the result

                // show result to console
                break;
            case '*':
                // do the arithmetic and store the result
                // show result to console
                break;
            case '/':
                // do the arithmetic and store the result
                // show result to console
                break;
            default:
                // show error message
                break;
        }

        // set finished calculating
    }
});
```

After this is done, make sure your code runs properly and that you are showing the correct result into the **TextView**.

Challenge: Use the boolean value to automatically clear the screen when you press a number after the calculator finishes a calculation (ie presses the equal button).

3.0 Setting up the random number generator (xml)

This is the simplest page to create since it doesn't require a lot of objects on the screen. You will create a simple random number generator that will show a **random number between 0-100** appear on the screen after pressing a button. Here's how to set it up.

Layout

For this section, we will be using a **LinearLayout.** Set the **width** and **height** to **match the** parent and the **orientation** to **vertical**.

TextView

Next, create a **TextView** that will show the random number. Set the **text** to **0** when you open the page. Set the **width** to **match the parent**, **height** to **580dp**, and the **weight** to **1.** Set the **gravity** to **center**. Set the **text size** to **80dp**.

Button

Next you will create a button. This button will say "**GENERATE**". It will sit at the **bottom** of the page. The **text size** will be **30dp**, **width** will **match parent**, **height** will be **100dp**.

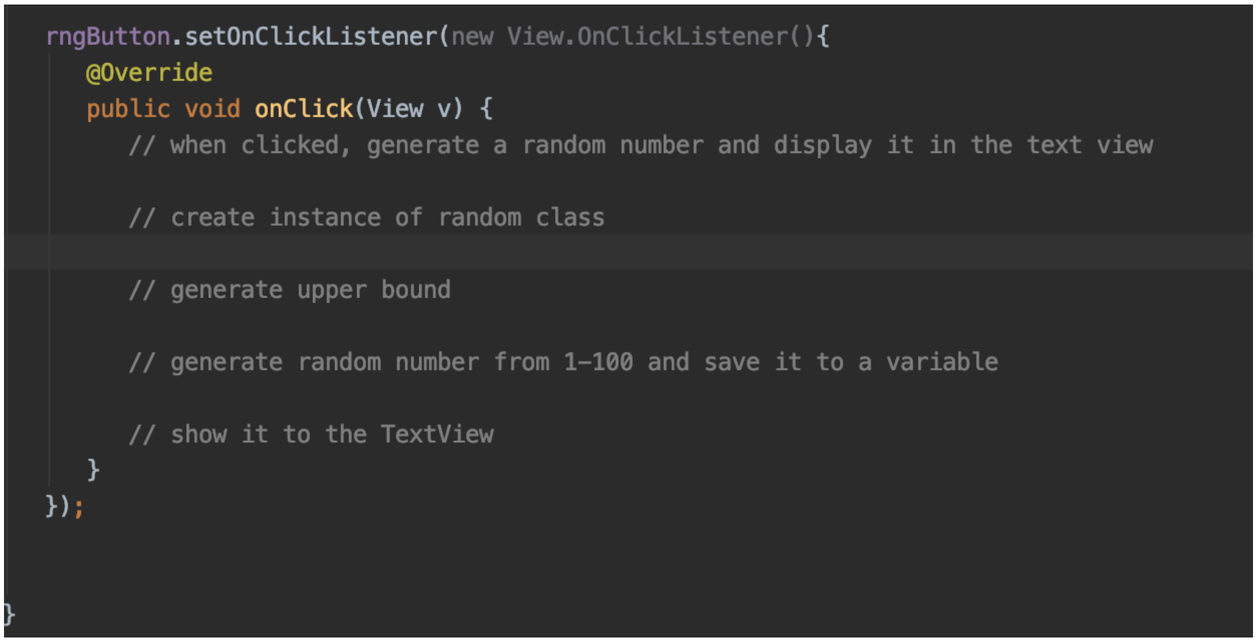
Colors

Change the color of the text in the **TextView** to **darkBlue**, and the **background color** to **calculatorGrey**.

As for the button, change the **text color** to **white**, and the **background** of the button to **darkBlue**.

3.1 Setting up the random number generator (.java)

This part is simple. All you really want to do is show the random number in the TextView when you click GENERATE. Here's a template of it as a hint.



```
rngButton.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        // when clicked, generate a random number and display it in the text view

        // create instance of random class

        // generate upper bound

        // generate random number from 1-100 and save it to a variable

        // show it to the TextView
    }
});


}
```

4.0 Running the program

Your program should now run. When it opens, you should be able to click on the calculator icon to go to the calculator page, and the dice icon to go to the rng page. You can now add more number applications to the main page then be able to click them to go to a different page where that application exists. This could be a future project to work on.

4.1 Debugging

There are some errors that you might run into along the way. Make sure that you name every single component in your .xml file in order to be able to find them in the .java file. If the page doesn't open up when you click, make sure that you have set the correct on click listener to the correct object. If your calculator logic doesn't work, check that your switch statement has a break statement. These are some of the errors that you might encounter when building this project. Keep a list of them to go over and fix. I encourage you to work with your team members to bounce ideas off one another.

Happy Coding!