Android Calculator

SEG2105 - Introduction to Software Engineering – Fall 2025

Lab Goals

- Give you exposure to the Android Studio tool.
- Learn about event-driven systems, such as clicking a button and having an event occur.
- Become familiar with creating user interfaces and connecting code to various elements.

- Learn to troubleshoot (if necessary).

- Guides for installation of Android Studio can be found in Tutorial 2 – Android on Brightspace.

Lab Objectives

- Create a simple calculator app
- Include the following operations:
  - addition (+)
  - subtraction (-)
  - multiplication (*)
  - division (/)
- Include:
  - decimal (32.5)
  - clear (clears the screen)
  - equal (displays the result)
- Consider a few edge cases:
  - some examples are given on the next slide

Considerations

- You might want to consider:
  - What happens when the screen is cleared and user clicks the "=" button? The app should not crash.
  - What happens when the user enters a negative number? (ex. -5+3) Does your code understand the input?
  - What if the user is going float operations (ex. 5.3+1)? Does the result display "6.3" instead of "6"?
  - What if the user is doing integer operations (ex. 3+6)? Does the result display "9" instead of "9.0"?

What needs to be done

- After creating your application, you need to push the code to a Git repository.
  - Use what you learned in Lab 0.
  - Additional Help: https://stackoverflow.com/questions/37093723/how-to-add-an-android-studio-project-to-github

- Have one member in your team create the repository and add other members to it for collaboration.
  - Additional Help: https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/managing-repository-settings/managing-teams-and-people-with-access-to-your-repository
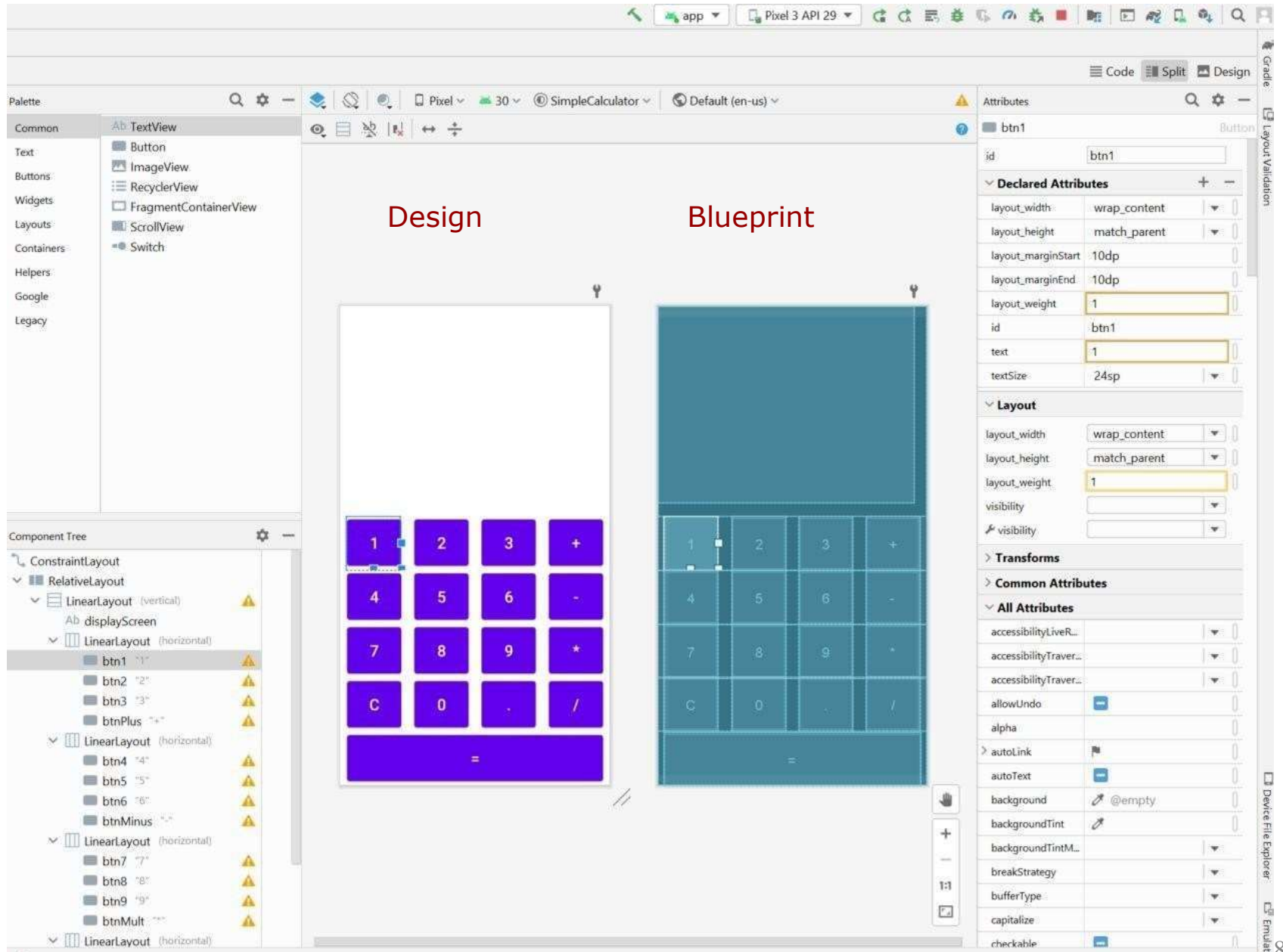
Submission

- Due 29th September on Brightspace.
- Submit a brief video showcasing the calculator. Show all buttons working as expected. Evaluate the following expressions:
  - 1+1
  - 2*3
  - 4/5
  - 10-3

- Share the link to your repository on Brightspace. Not submitting the link to the repository will count as an incomplete submission.

- You will find the same information on Brightspace.!

Layout

- You can use a RelativeLayout with LinearLayouts, as show in the screenshot.

- You can use the ConstraintLayout to create different constraints and build a responsive UI.

  - Ex. constrain a button to the left side of another button.
  - https://developer.android.com/training/basics/firstapp/building-ui

- Sometimes it is easier to drag and drop elements from the palette to the Component Tree.

- Make sure to give all important elements an id.

Component Tree

```
ConstraintLayout
  RelativeLayout
    LinearLayout (vertical)
      Ab displayScreen
      LinearLayout (horizontal)
        btn1 "1"
        btn2 "2"
        btn3 "3"
        btnPlus "+"
      LinearLayout (horizontal)
        btn4 "4"
        btn5 "5"
        btn6 "6"
        btnMinus "-"
      LinearLayout (horizontal)
        btn7 "7"
        btn8 "8"
        btn9 "9"
        btnMult "*"
```

Design

Blueprint

- Sometimes you will want to declare attributes using the .xml code instead.
- Get comfortable with both the Code and Design view that Android Studio offers.

Interface

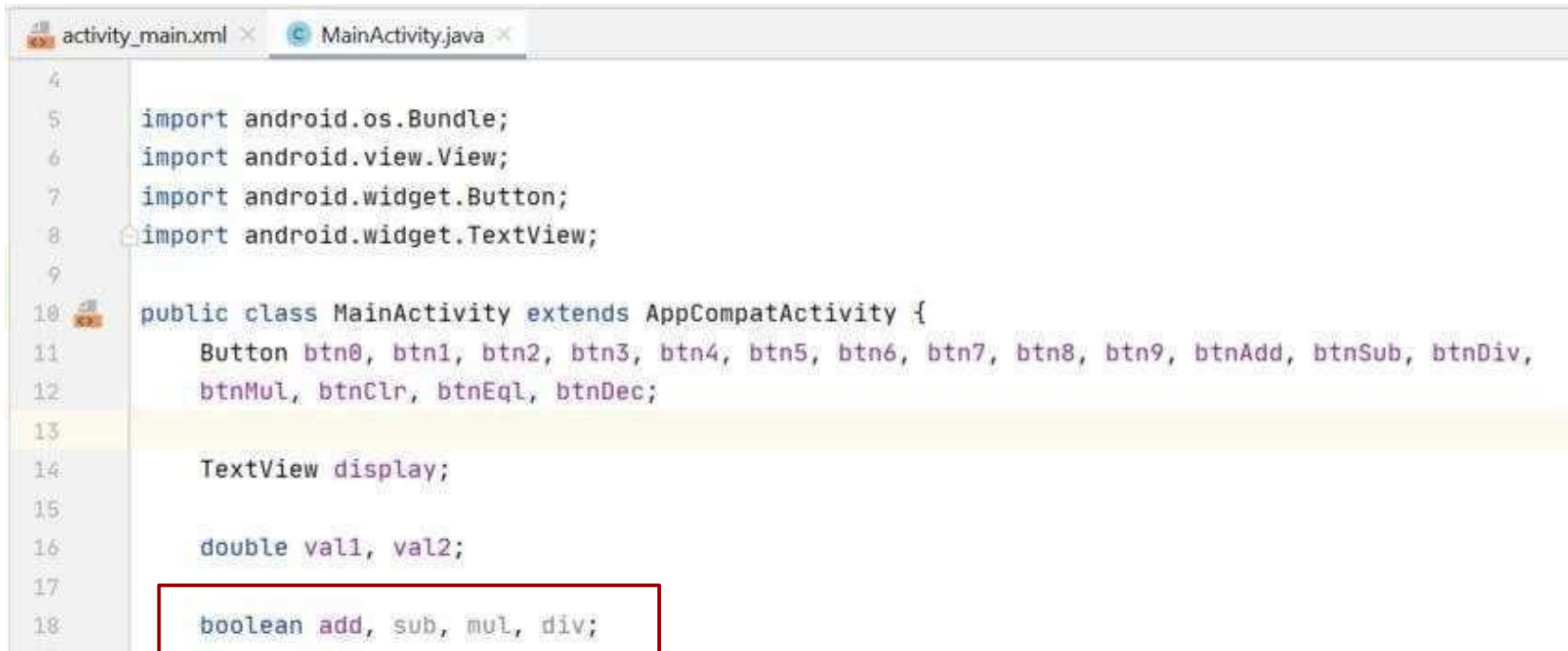- You have full flexibility to experiment with the layout and design of you application.

Source: Apple   Source: Blackberry

10

Variable declaration

- This example shows a boolean approach for the calculator logic.
- The double and boolean variables will be used for the mathematical operations.

```java
activity_main.xml    MainActivity.java

4
5    import android.os.Bundle;
6    import android.view.View;
7    import android.widget.Button;
8    import android.widget.TextView;
9
10   public class MainActivity extends AppCompatActivity {
11       Button btn0, btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9, btnAdd, btnSub, btnDiv,
12       btnMul, btnClr, btnEql, btnDec;
13
14       TextView display;
15
16       double val1, val2;
17
18       boolean add, sub, mul, div;
```

- Alternatively, you can use the enum approach in your application.
- You are also free to use any other logic.

```java
public class MainActivity extends AppCompatActivity {
    Button btn0, btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9, btnAdd, btnSub, btnDiv,
    btnMul, btnClr, btnEql, btnDec;

    TextView display;

    double val1, val2;

    enum Operator{none, add, minus, multiply, divide}
    Operator optr = Operator.none;
```

Variable initialization

- Still in the MainActivity, initialize the buttons under the onCreate method.
- findViewById() uses the id's you gave to the elements in the layout. Make sure all buttons are given proper id's.

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    btn0 = findViewById(R.id.btn0);
    btn1 = findViewById(R.id.btn1);
    btn2 = findViewById(R.id.btn2);
    btn3 = findViewById(R.id.btn3);
    btn4 = findViewById(R.id.btn4);
    btn5 = findViewById(R.id.btn5);
```

button.setOnClickListener()

- Write the code for setOnClickListener() for each type of button.
- You want the number to be shown in the TextView, therefore we use setText().
- You also want to be able to enter large values (ex. 123), therefore we use getText() and then add our string.
- An example for "1" is seen below.
- Do this for each calculator button.

```
btn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        display.setText(display.getText() + "1");
    }
});
```

- Build the setOnClickListener() for each of the different operations (+, -, /, *) and decimal, clear, equal.
- You can use whatever logic you think is best. There is no one answer.

```
btnEql.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {

        //....

    }
```