

Tutorial 2

- Building adaptive layout for calculator
- **Due 27 August Sunday 23:59 AWST, 20 Marks**
- **It is recommended to submit early and get marked. Do not wait for the due date. Tutorial 1,2,3,4 have the same due date, i.e., 27 August Sunday, 23:59 AWST**

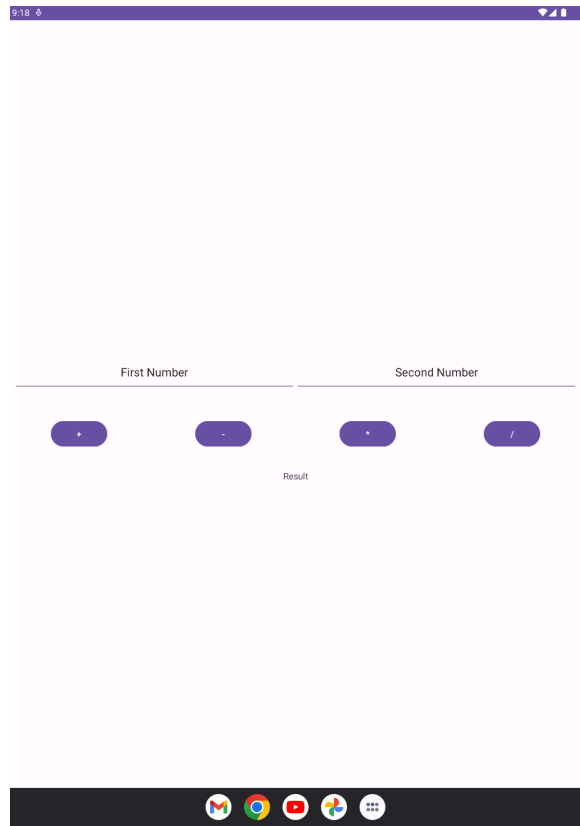


What we did last week

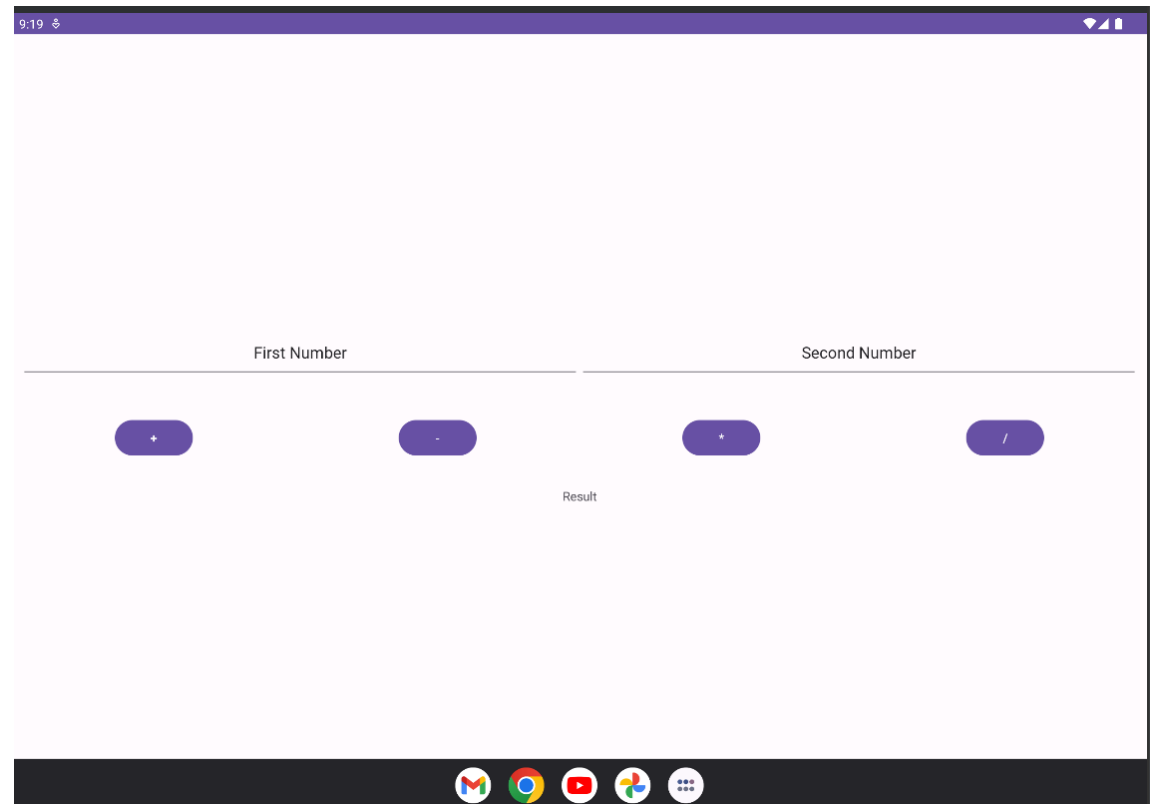
- We have built a simple calculator with two editTexts, four buttons, and a textView to show the results.
- Try running the app on different screen sizes and orientations, landscape, portrait, phone, tablet etc.
- You will find that the view is not consistent, i.e., it does not adapt the screen properly.
- This tutorial is all about building an adaptive layout for the calculator.

Adaptive Calculator application in tablet

Portrait mode

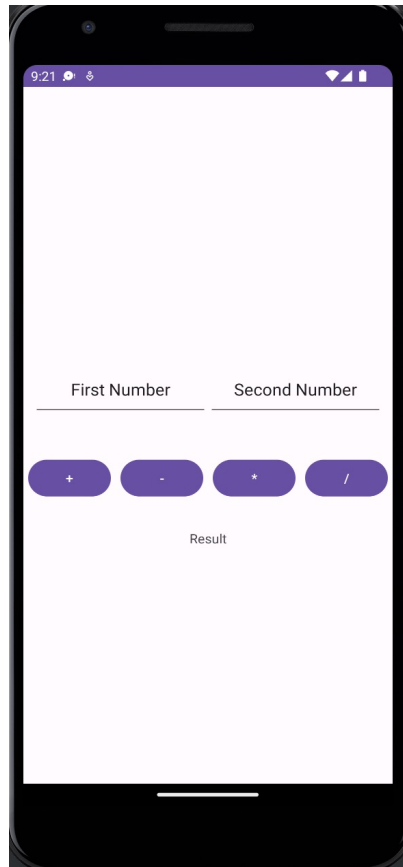


Landscape mode mode

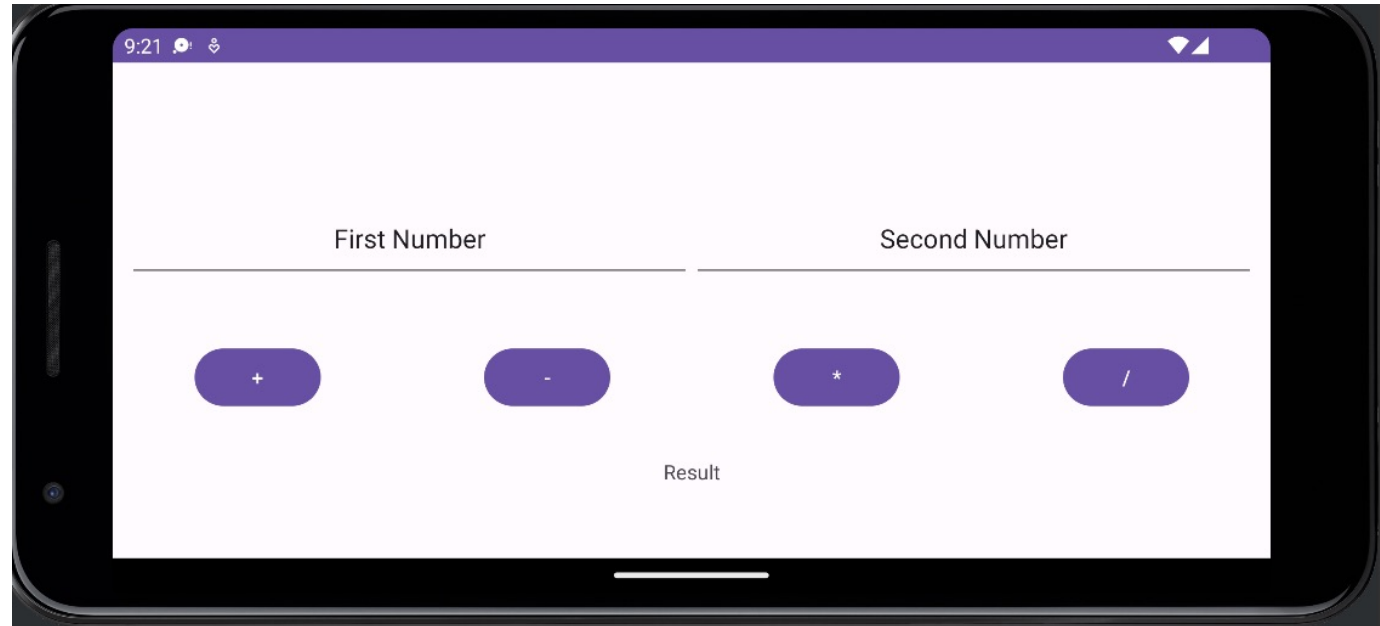


Adaptive Calculator application in phone

Portrait mode



Landscape mode mode



Adaptive calculator properties

All elements automatically adjust to the centre of the screen.

Each editText is 50% of the screen's width, and they fill the screen horizontally together. The text in the editText is centre-aligned

The four buttons also fit in a row. However, their spacing is uniform and proportional to the screen's width.

The result text view is at the centre, its top relative to the position of the buttons.

Task 1 (Cont.): Building the adaptive layout [10 Marks]

Step 1: Create a new Android project

- Open Android Studio and create a new Android project with an empty activity. Name it as per your preference.



Step 2: Locate activity_main.xml

- In the 'res' folder, locate the 'layout' folder, and open the 'activity_main.xml' file.

Task 1 (Cont..)

- Step 3: Set up a ConstraintLayout inside the Root Constraint layout.
- The ConstraintLayout's height should be "wrap-content." and it should be constrained with the root at 4 sides to be always in the center.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns: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">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent">

        <LinearLayout...>
            <androidx.constraintlayout.widget.ConstraintLayout...>
                <TextView...>
            </androidx.constraintlayout.widget.ConstraintLayout>
        </LinearLayout>
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
<LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:padding="10dp"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <EditText
        android:id="@+id/firstNumber"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="text"
        android:text="First Number"
        android:textAlignment="center"/>
    <EditText
        android:id="@+id/secondNumber"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="text"
        android:text="Second Number"
        android:textAlignment="center"/>
</LinearLayout>
```

Task 1 (Cont..)

- Step 4: Add the LinearLayout with EditTexts
 - Inside the ConstraintLayout, add the a LinearLayout with two EditText views.
 - Check the three constraints of the linear layout.
 - Check the editTexts layout_weight and textAlignment.

Task 1 (Cont..)

- Step 5: Add another inner ConstraintLayout (third one) to hold the Buttons.
- Step 6. Add three guidelines to divide the ConstraintLayout in 25%, 50% and 75%, so four buttons will be placed within the guidelines

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/constraintLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent=".25" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent=".50" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent=".75" />
```


Step 7: Place the four buttons inside of the third constraint layout

```
<Button
    android:id="@+id/mulButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="*"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toEndOf="@+id/guideline2"
    app:layout_constraintEnd_toStartOf="@+id/guideline3"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/divButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="/"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toEndOf="@+id/guideline3"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<Button
    android:id="@+id/addButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="+"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/guideline1"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/subButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="-"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toEndOf="@+id/guideline1"
    app:layout_constraintEnd_toStartOf="@+id/guideline2"
    app:layout_constraintTop_toTopOf="parent" />
```

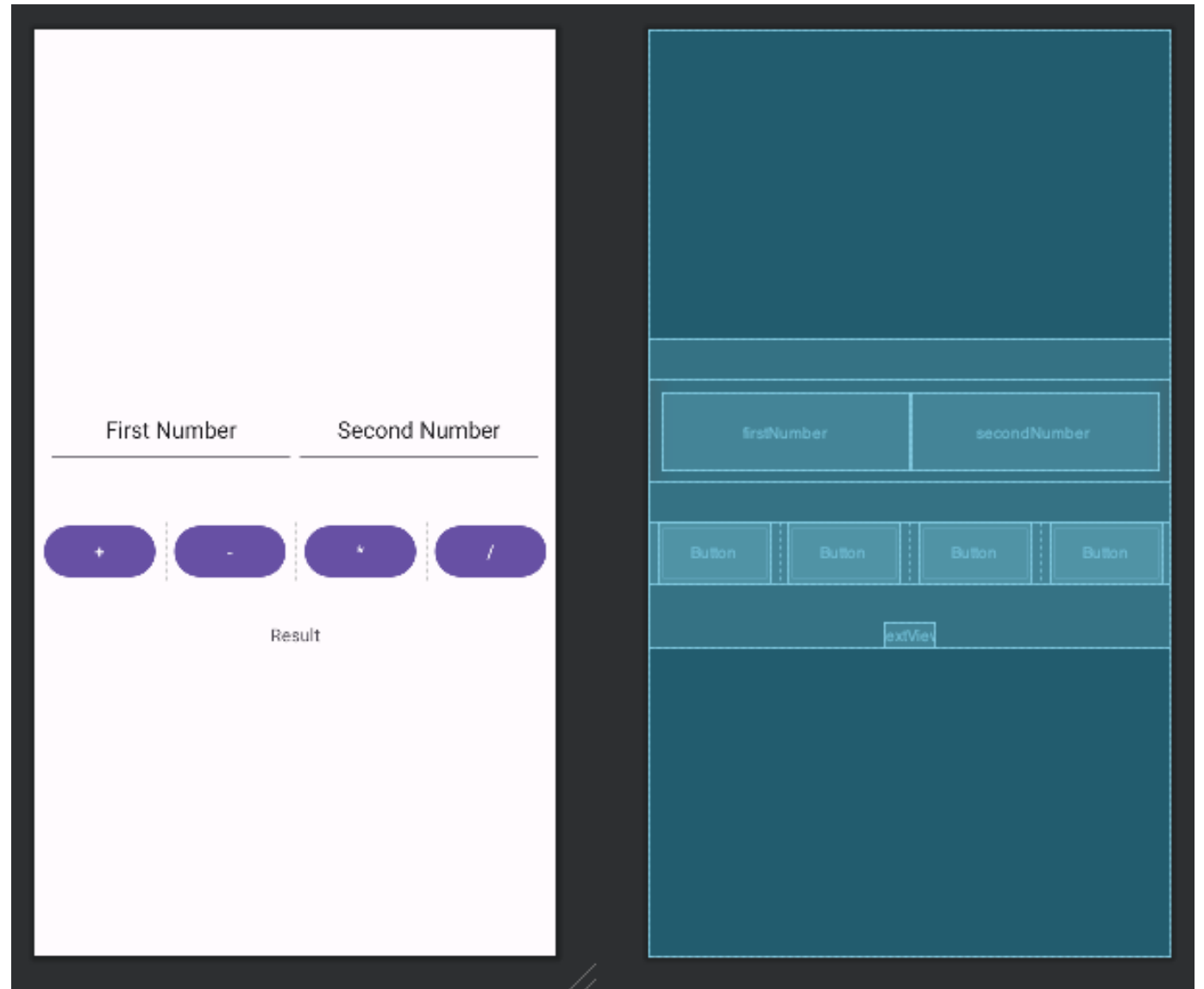


Step 8: Place the
textView inside the
second constraintLayout.
Make sure it is placed
under the third
constraintLayout

```
<TextView
    android:id="@+id/resultView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:text="Result"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/constraintLayout1" />
```

The final blueprint of Task 1

Run the App and check if you get the desired output



Task 2: Updating the layout [5 marks]



1

Check the lecture slide two and code about drawable.



2

Now change the buttons in Task 1 into circles.



3

Add different background colours in the linear layout, buttons and text view to make it attractive. You can choose your colours.

Task 3: Updating the layout [5 Marks]

- Add three images at the bottom of the screen. They will act as an adaptive menu bar, i.e., always be at the bottom. You can use any images you want. Just make sure each image takes about 33% of the screen's width and they fill the screen horizontally together at the bottom.



END