Android Studio Tutorial - Part 2

Welcome!

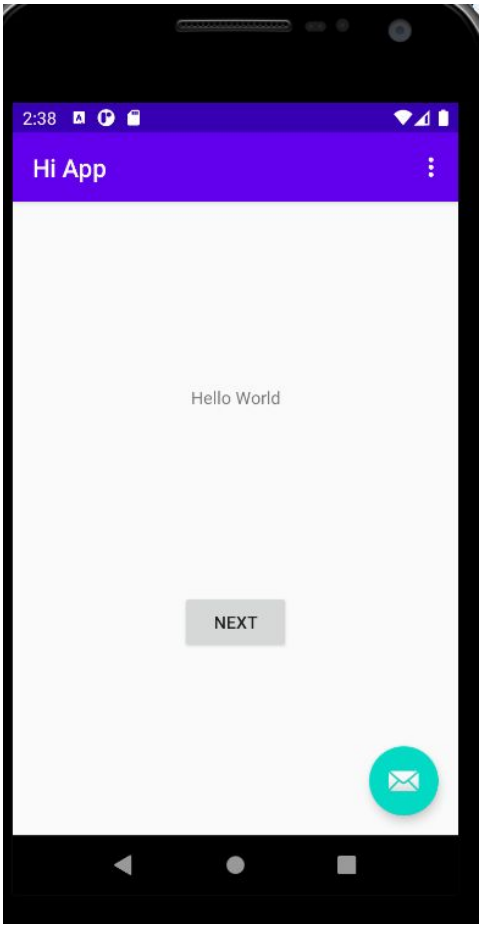
In this tutorial, you will learn how to build and run an interactive Android App in the Java programming language.

You will learn:

- How to set change layouts.
- How to add interactive buttons.
- How to display a second screen when a button is pressed.

Install Android Studio

1. Follow the tutorial part 1 - Hello World App
2. Run your app and you will see a screen like in below:

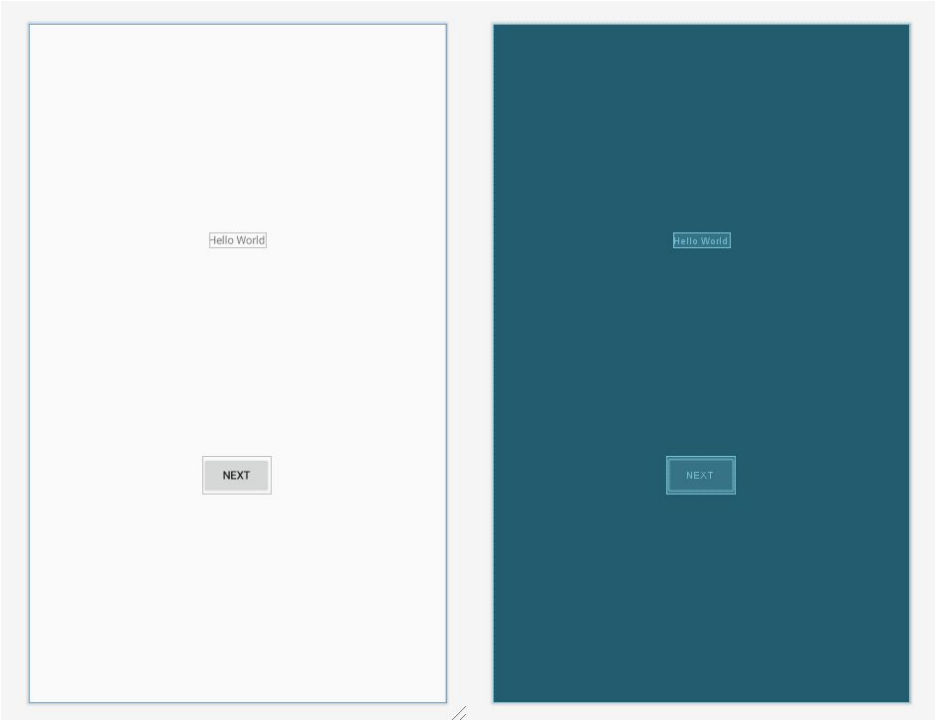


Set the layout

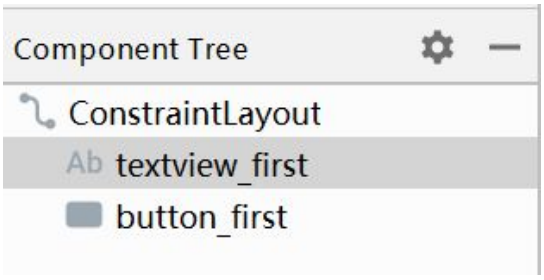
Each visible fragment in an Android app has a layout that defines the user interface for the fragment. Android Studio has a layout editor where you can create and define layouts.

1. Find and open the **Layout** folder (app>res>layout) on the left side in the Project panel
2. Open `fragment_first.xml` file, click **Design** on the upper right corner to pull up the design interface. Now you will see the *Design layout* on the left shows how your app

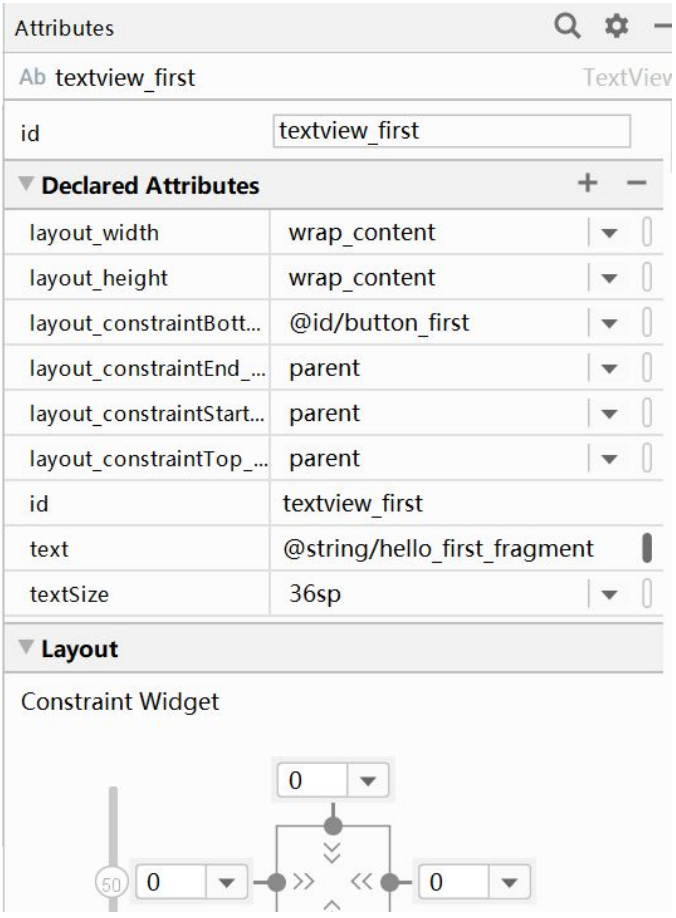
appears on the device. The *Blueprint layout*, shown on the right, is a schematic view of the layout.



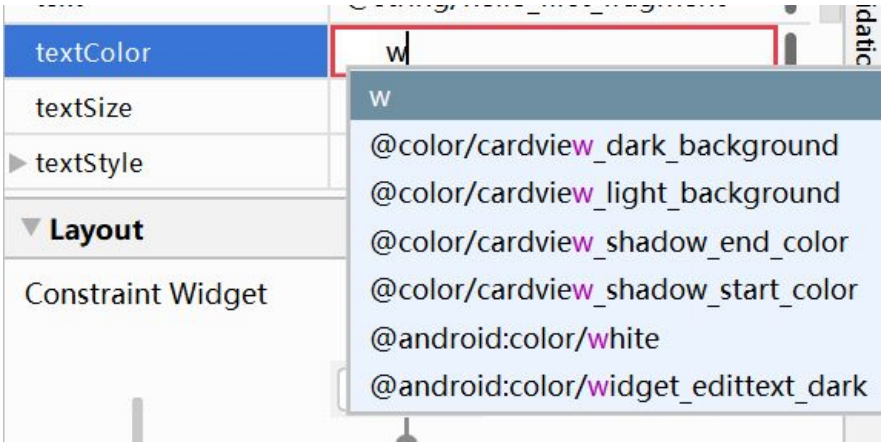
3. Select textview_first in the Component tree.



4. Look at the Attributes panel on the right, and open the Declared Attributes section if needed.



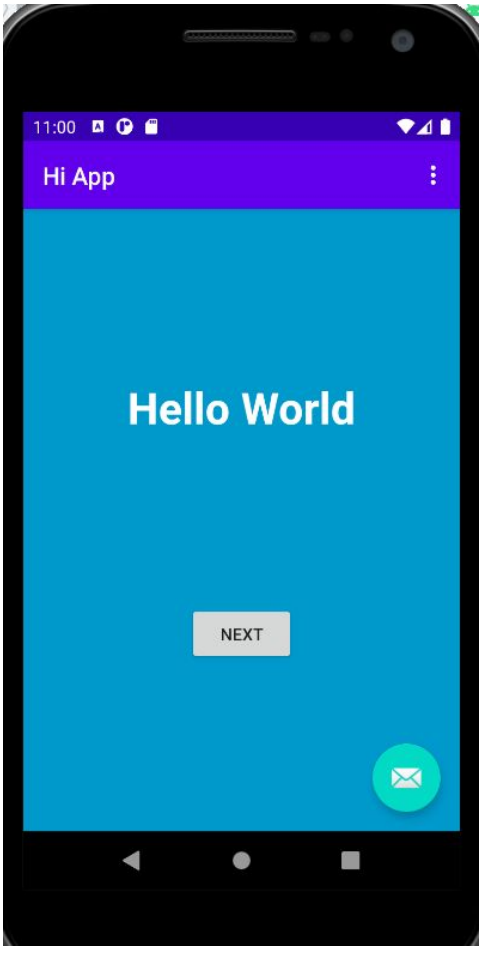
5. You can change the value of string without having to change any other code.
6. Find the textSize field and change it to 36sp.
7. Find the textStyle and select bold
8. Click in the textColor field, and enter “w”. A menu pops up with possible completion values containing the letter w. This list includes predefined colors.
9. Select @android:color/white and press Enter.



10. Look at the XML for the TextView. You see that the new properties have been added.

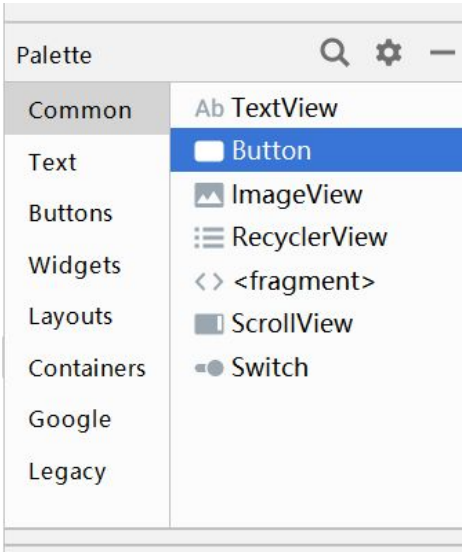
```
<TextView
    android:id="@+id/textview_first"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World"
    android:textColor="@android:color/white"
    android:textSize="36sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@id/button_first"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

11. Select the ConstraintLayout in the Component Tree
12. Go to the Attributes and find the background color file, type "blue". Select @android:color/holo_blue_dark. (you can choose any color that you like to build your own app)
13. Run your app, the screen will be:

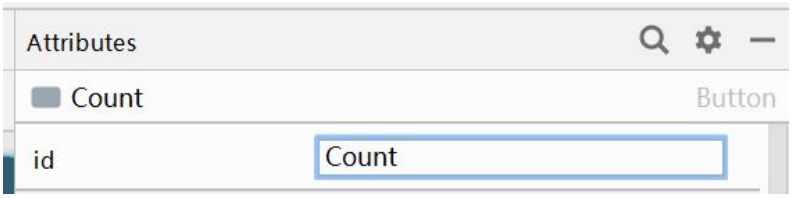


Add new button and constrain their positions

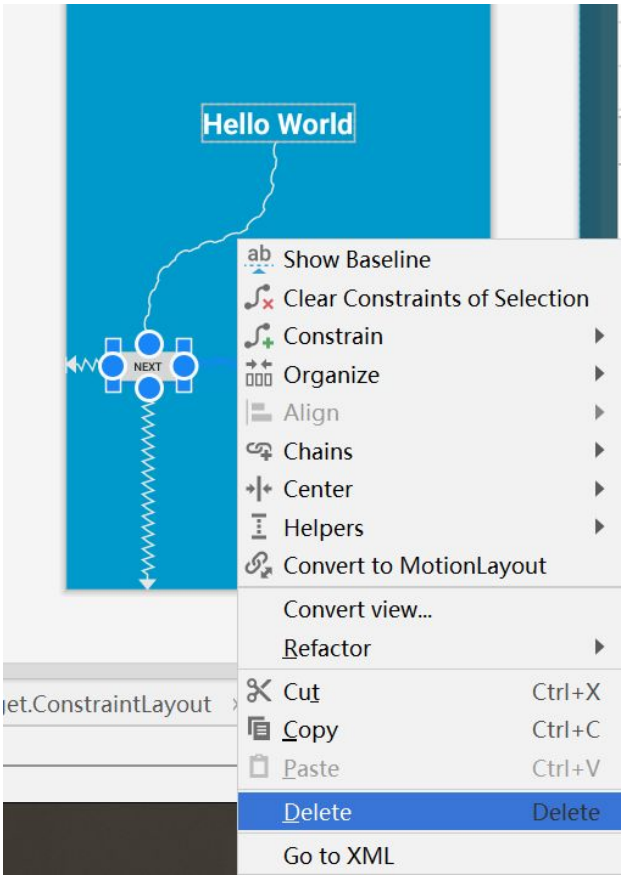
1. Go to the Palette panel, click on Button, drag and drop it onto the design view.



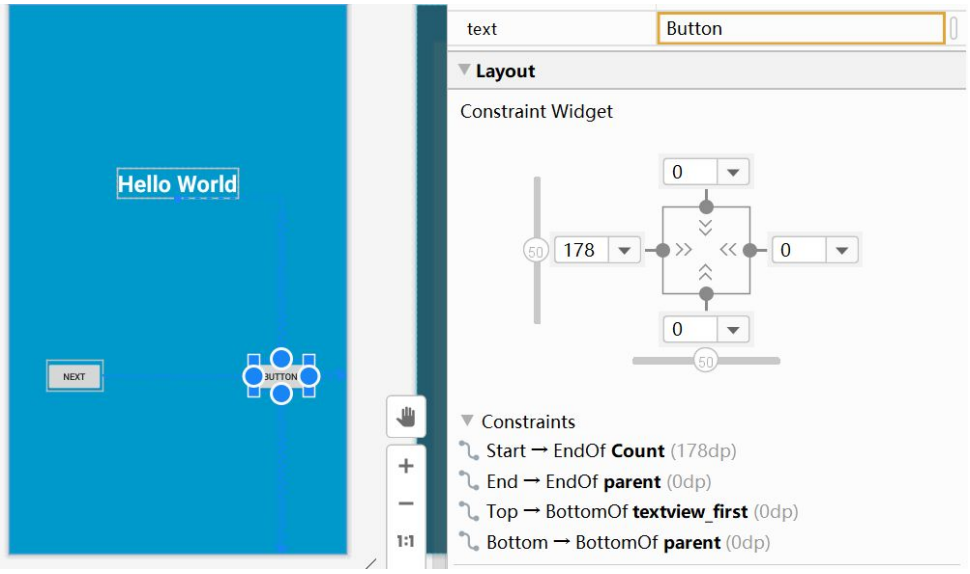
2. Now, let's set their positions.
3. First, click the button_first, go to **Attributes** and change the id to “Count”. A window will pop up asking whether you need to update all usages of button_first, select yes.



4. Same as button_first, change the id for newly added button to “Random”.
5. Select the “Count” button, click its right constraint and delete it.



6. Then click the “Count” button, move the cursor over the circle at the right side of the button, drag it to the left side of the “Random” Button.
7. Click the “Random” button, move the cursor over the circle at the top of the”Random”, click and drag the circle at the top of the “Random” onto the circle at the bottom of the “Textview”.



8. Set the constraint of the other three sides of the “Random” button.

Change string value

1. Go to app>res>values>strings.xml.
2. Change the value of “hello_first_fragment” to be 0 (since we will need to count numbers)
3. Add two new lines shown as following:

```
<string name="hello_first_fragment">0</string>
<string name="count">Count</string>
<string name="Random">Random</string>
```

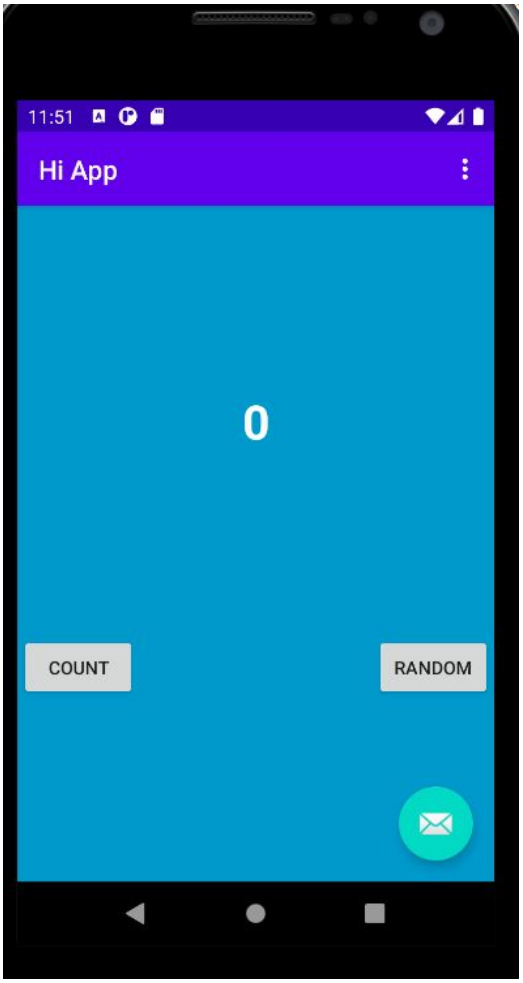
4. Go to fragment_first.xml, change the text value for two buttons. Update the text value for first button as “@string/count”

```
<Button
    android:id="@+id/Count"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/count"
```

Update the text value for the “Random” button as “@string/Random”.

```
android:layout_height="wrap_content"
android:layou  @string/Random
android:layou  Press Enter to insert, Tab to replace
android:text="@string/R"
app:layout_constraintBottom_toBottomOf="parent"
```

5. Run the app and you will get the following:



Set up the count function

1. In the fragment_first.xml layout file, notice the id for the TextView:

```
<TextView
    android:id="@+id/textview_first"
```

2. In FirstFragment.java, add a click listener for the count button below the other click listeners in onViewCreated(). Because it has a little more work to do, have it call a new method, countMe(). (delete the selected function and replace it with counMe())

```
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    view.findViewById(R.id.Count).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            NavHostFragment.findNavController( fragment: FirstFragment.this)
                    .navigate(R.id.action_FirstFragment_to_SecondFragment);
        }
```

→

```
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    view.findViewById(R.id.Count).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            countMe(view);
        }
    });
```

3. In the FirstFragment class, add the method countMe() that takes a single View argument. This method will be invoked when the Count button is clicked and the click listener called.

   `private void countMe(View view){}`

   - Get the value of the showCountTextView.

     `String countString = showCountTextView.getText().toString();`

   - Convert the value to a number, and increment it.

     `Integer count = Integer.parseInt(countString);`

     `count++;`

   - Display the new value in the TextView by programmatically setting the text property of the TextView.

     `showCountTextView.setText(count.toString());`

4. Here is the whole method:

```
private void countMe(View view){
    // Get the value of the text view
    String countString = showCountTextView.getText().toString();
    // Convert value to a number and increment it
    Integer count = Integer.parseInt(countString);
    count++;
    // Display the new value in the text view.
    showCountTextView.setText(count.toString());
}
```

Cache the TextView for repeated use

1. In the FirstFragment class before any methods, add a member variable for showCountTextView of type TextView.

   `TextView showCountTextView;`

2. In onCreateView(), you will call findViewById() to get the TextView that shows the count. The findViewById() method must be called on a View where the search for the requested ID should start, so assign the layout view that is currently returned to a new variable, fragmentFirstLayout, instead.

   `View fragmentFirstLayout = inflater.inflate(R.layout.fragment_first, container, false);`

3. Call findViewById() on fragmentFirstLayout, and specify the id of the view to find, textview_first. Cache that value in showCountTextView.

   showCountTextView = fragmentFirstLayout.findViewById(R.id.*textview_first*);

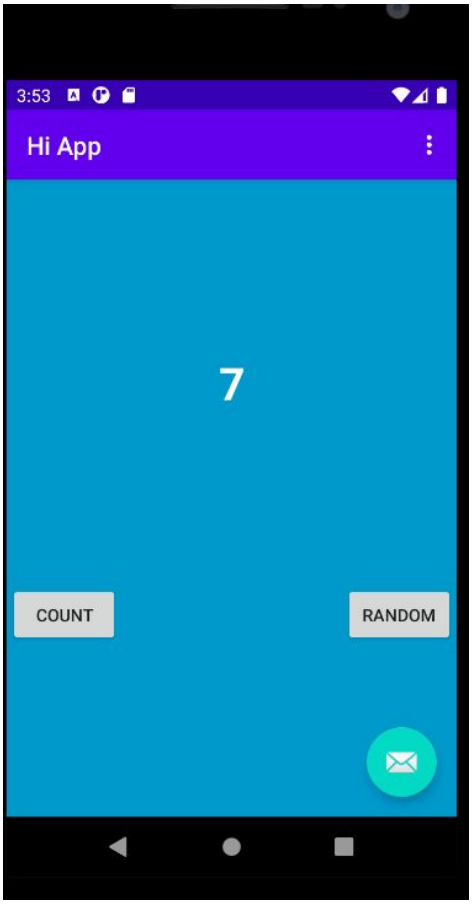
4. Return fragmentFirstLayout from onCreateView().

   return fragmentFirstLayout;

5. Here is the whole method and the declaration of showCountTextView:

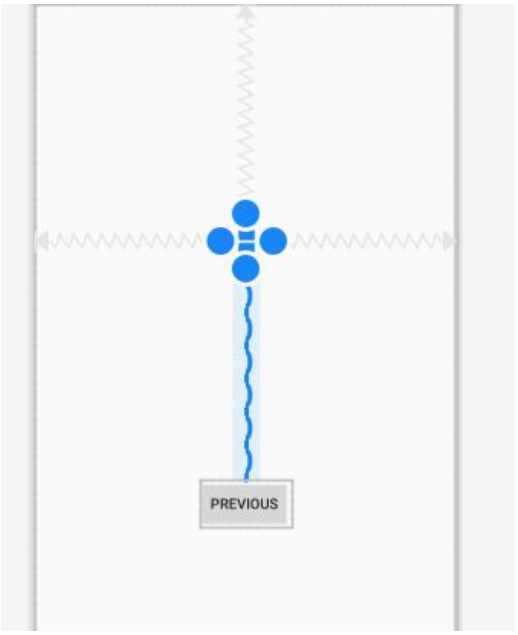
```
public class FirstFragment extends Fragment {

    TextView showCountTextView;

    @Override
    public View onCreateView(
            LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState
    ) {
        // Inflate the layout for this fragment
        View fragmentFirstLayout = inflater.inflate(R.layout.fragment_first, container,  attachToRoot: false);
        showCountTextView = fragmentFirstLayout.findViewById(R.id.textview_first);
        return fragmentFirstLayout;
    }
```

6. Run your app. Press the Count button and watch the count update.



Implement the second fragment

1. Open fragment_second.xml (app > res > layout > fragment_second.xml) and switch to Design View if needed. Notice that it has a ConstraintLayout that contains a TextView and a Button.
2. Remove the chain constraints between the TextView and the Button.



3. Add another TextView from the palette and drop it near the middle of the screen. This TextView will be used to display a random number between 0 and the current count from the first Fragment.
4. Set the id to @+id/textview_random (textview_random in the Attributes panel.)
5. Constrain the top edge of the new TextView to the bottom of the first TextView, the left edge to the left of the screen, and the right edge to the right of the screen, and the bottom to the top of the Previous button.
6. Set both width and height to wrap_content.
7. Set the textColor to @android:color/white, set the textSize to 36sp, and the textStyle to bold.
8. Set the text to "R". This text is just a placeholder until the random number is generated.
9. Here is the XML code for the TextView that displays the random number:

```
<TextView
    android:id="@+id/textview_random"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="R"
    android:textColor="@android:color/white"
    android:textSize="36sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/button_second"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textview_second" />
```

10. Update the TextView to display the header
    - In the fragment_second.xml, select textview_second, change the id to textview_header in the Attributes panel
    - In strings.xml, change hello_second_fragment to be "Here is a random number between 0 and %d."

```
<string name="hello_second_fragment">Here is a random number between 0 and %d.</string>
```

    - Go back to the fragment_second.xml, add a text attribute to be the hello_second_fragment.

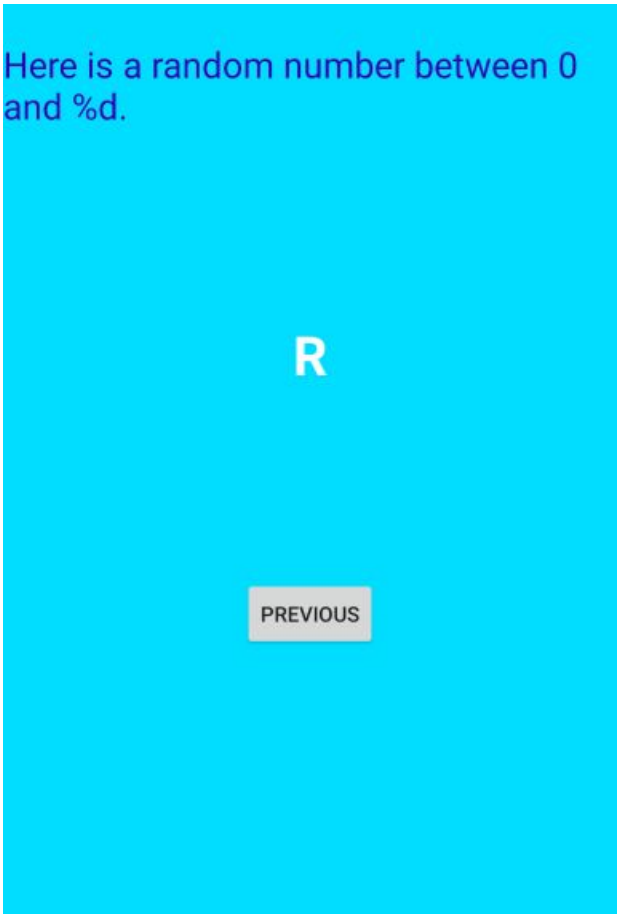
```
<TextView
    android:id="@+id/textview_header"
    android:layout_width="wrap_content"
    android:text="@string/he"
    android:layou
    app:layout_co
    app:layout_co
    app:layout_constraintTop_toTopOf="parent" />
```

```
@string/hello_second_fragment
@string/hello_first_fragment
Press Enter to insert, Tab to replace
```

    - In the Attribute panel, Set the width to match_constraint, but set the height to wrap_content, so the height will change as needed to match the height of the content.
    - Set top, left and right margins to 24dp. Left and right margins may also be referred to as "start" and "end" to support localization for right to left languages.
    - Remove any bottom constraint.
    - Set the text color to @color/colorPrimaryDark and the text size to 24sp.
    - Check the code, you will have:

```
<TextView
    android:id="@+id/textview_header"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="24dp"
    android:layout_marginEnd="24dp"
    android:layout_marginRight="24dp"
    android:text="@string/hello_second_fragment"
    android:textColor="@color/colorPrimaryDark"
    android:textSize="24sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

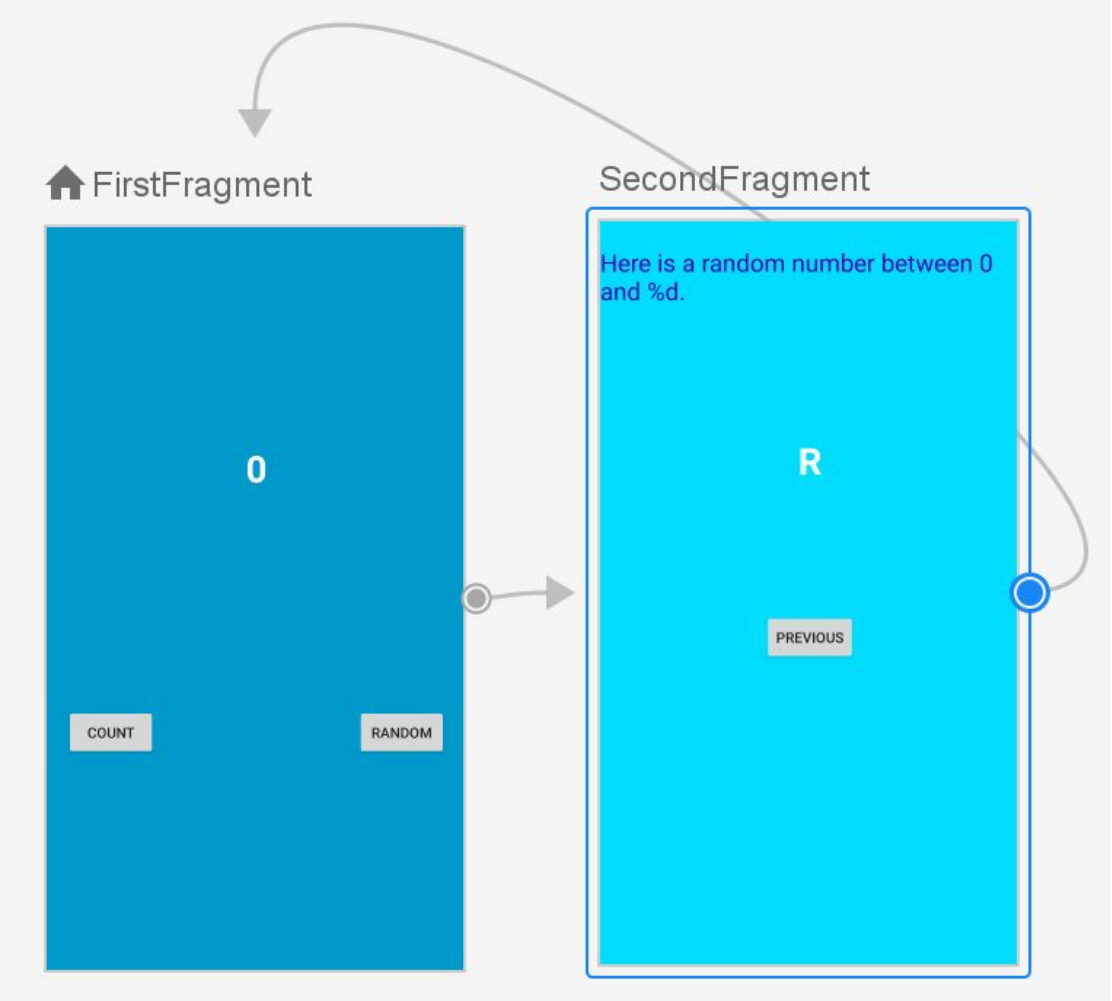
11. In the layout for the second activity, fragment_second.xml, set the background to be "@android:color/holo_blue_bright"
12. Now, you will have your second page displays as following:



Set the navigation graph

When Android Studio uses the Basic Activity template for a new project, it sets up two fragments, and a navigation graph to connect the two. It also set up a button to send a string argument from the first fragment to the second. This is the button you changed into the Random button. And now you want to send a number instead of a string.

1. Open nav_graph.xml (app > res > navigation > nav_graph.xml).
2. You can freely move the elements in the navigation editor. For example, if the fragments appear with SecondFragment to the left, drag FirstFragment to the left of SecondFragment so they appear in the order you work with them.
3. It should be something like the following:

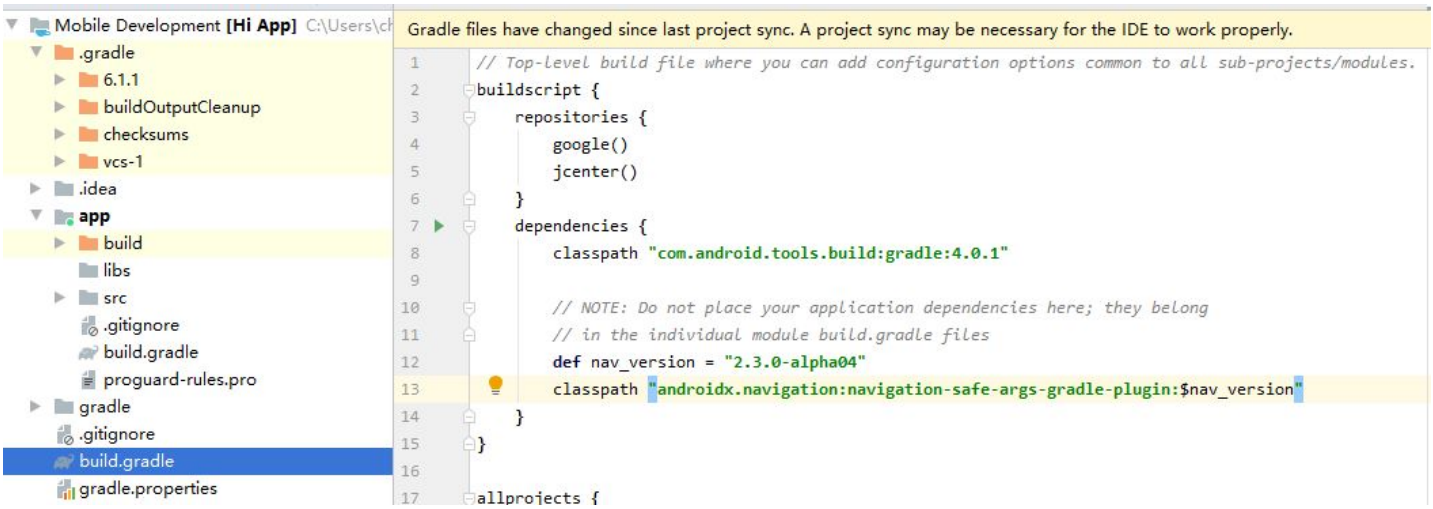


Enable SafeArgs

This will enable SafeArgs in Android Studio.

1. Open Gradle Scripts > build.gradle (Project: Hi App)
2. Find the dependencies section In the buildscript section, and add the following lines after the other classpath entries:

```
def nav_version = "2.3.0-alpha04"

classpath "androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version"
```



3. Open Gradle Scripts > build.gradle (Module: app)
4. Just below the other lines that begin with apply plugin add a line to enable SafeArgs:

apply **plugin**: **'androidx.navigation.safeargs'**

5. Android Studio should display a message about the Gradle files being changed. Click Sync Now on the right hand side.
6. After a few moments, Android Studio should display a message in the Sync tab that it was successful.
7. Choose Build > Make Project. This should rebuild everything so that Android Studio can find FirstFragmentDirections.

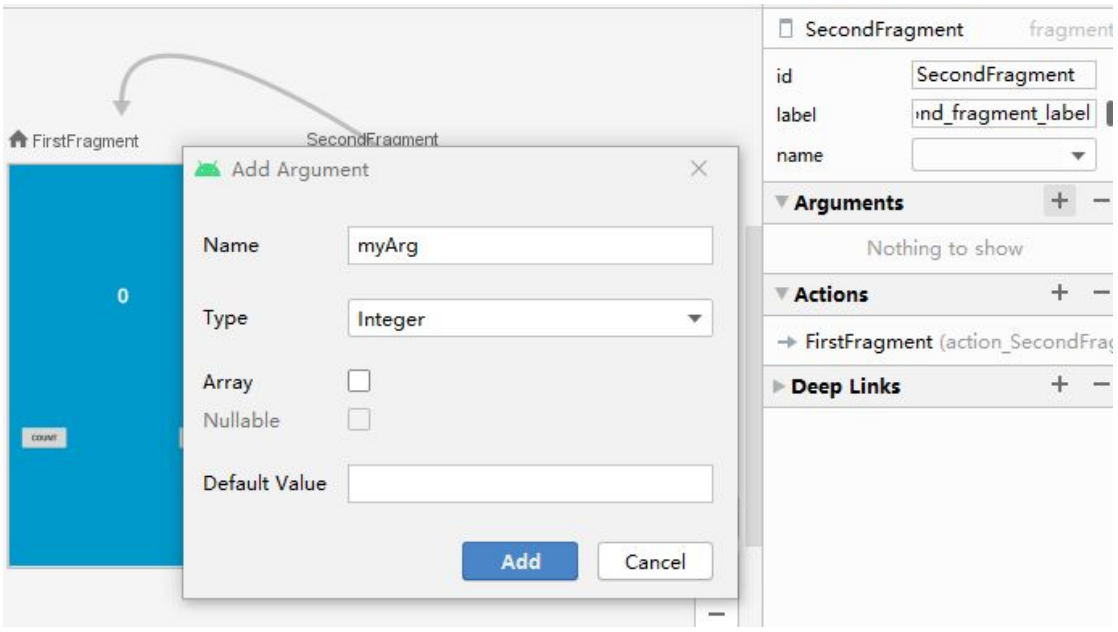
Create the argument for the navigation action

1. In the navigation graph, click on FirstFragment, and look at the **Attributes** panel to the right. (If the panel isn't showing, click on the vertical Attributes label to the right.)
2. In the **Actions** section, it shows what action will happen for navigation, namely going to SecondFragment.

3. Click on SecondFragment, and look at the **Attributes** panel.

The Arguments section shows Nothing to show.

4. Click on the + in the **Arguments** section.
5. In the **Add Argument** dialog, enter myArg for the name and set the type to **Integer**, then click the **Add** button.



Send the count to the second fragment

1. Open FirstFragment.java **(app > java > com.example.myfirstapp > FirstFragment)**
2. Find the method onViewCreated() and notice the code that sets up the click listener to go from the first fragment to the second.
3. Replace the code in that click listener with a line to find the count text view, textview_first.

   ```
   int currentCount = Integer.parseInt(showCountTextView.getText().toString());
   ```

4. Create an action with currentCount as the argument to actionFirstFragmentToSecondFragment().

   ```
   FirstFragmentDirections.ActionFirstFragmentToSecondFragment action = FirstFragmentDirections.actionFirstFragmentToSecondFragment(currentCount);
   ```

5. Add a line to find the nav controller and navigate with the action you created.

   ```
   NavHostFragment.findNavController(FirstFragment.this).navigate(action);
   ```

6. Here is the whole method, including the code you added earlier:

```
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    view.findViewById(R.id.Random).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            int currentCount = Integer.parseInt(showCountTextView.getText().toString());
            FirstFragmentDirections.ActionFirstFragmentToSecondFragment action = FirstFragmentDirections.actionFirst
            NavHostFragment.findNavController( fragment: FirstFragment.this).navigate(action);
        }
    });

    view.findViewById(R.id.Count).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            countMe(view);
        }
    });
}
```

7. Run your app. Click the Count button a few times. Now when you press the Random button, the second screen shows the correct string in the header, but still no count or random number, because you need to write some code to do that.

Update SecondFragment to compute and display a random number

You have written the code to send the current count to the second fragment. The next step is to add code to SecondFragment.java to retrieve and use the current count.

1. In the onViewCreated() method below the line that starts with super, add code to get the current count, get the string and format it with the count, and then set it for textview_header.

   Integer count = SecondFragmentArgs.*fromBundle*(getArguments()).getMyArg();

   String countText = getString(R.string.***hello_second_fragment***, count);

   TextView headerView = view.getRootView().findViewById(R.id.***textview_header***);

   headerView.setText(countText);

2. Get a random number between 0 and the count.

   Random random = **new** java.util.Random();

   Integer randomNumber = 0;

   if (count > 0) {

     randomNumber = random.nextInt(count + 1);

   }

3. Add code to convert that number into a string and set it as the text for textview_random.

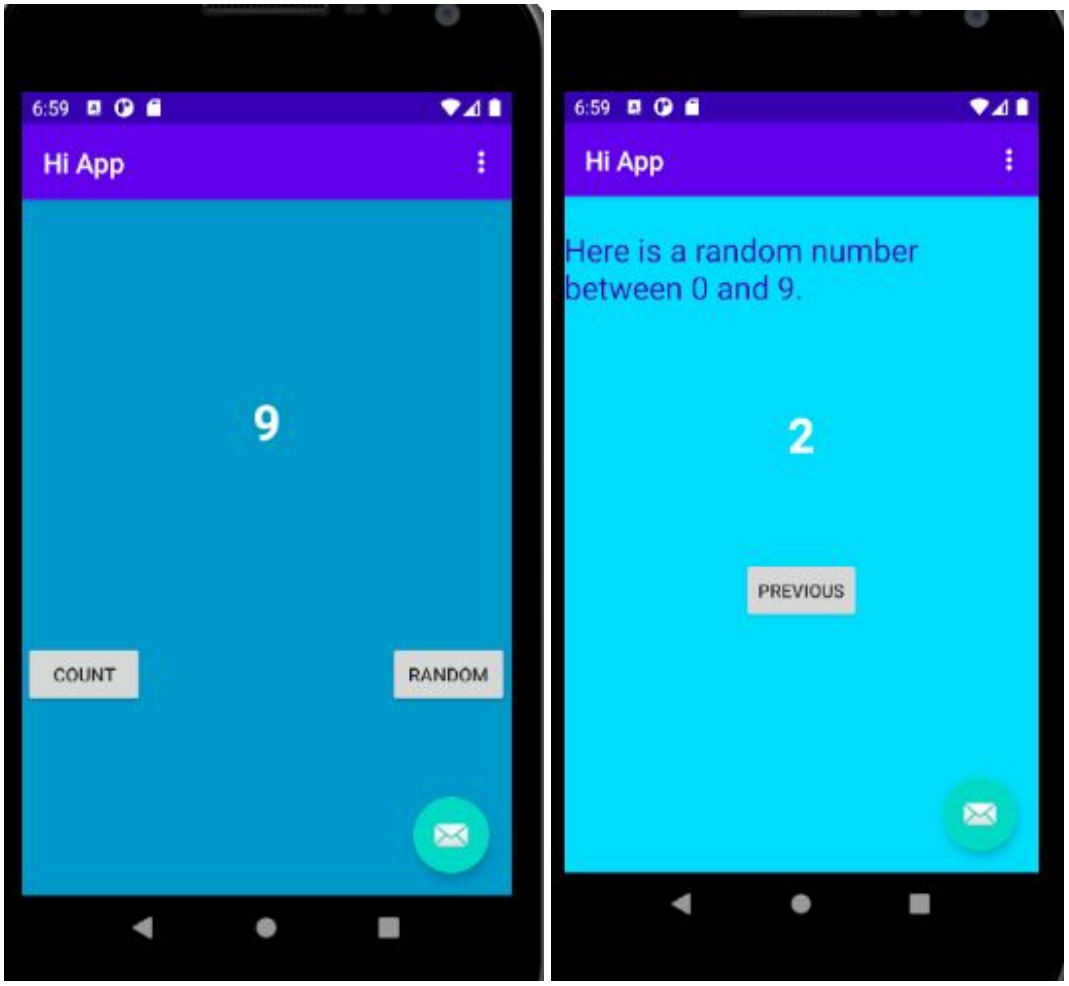
   TextView randomView = view.getRootView().findViewById(R.id.***textview_random***);

   randomView.setText(randomNumber.toString());

4. The whole method is showing below:

```
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    Integer count = SecondFragmentArgs.fromBundle(getArguments()).getMyArg();
    String countText = getString(R.string.hello_second_fragment, count);
    TextView headerView = view.getRootView().findViewById(R.id.textview_header);
    headerView.setText(countText);
    Random random = new java.util.Random();
    Integer randomNumber = 0;
    if (count > 0) {
        randomNumber = random.nextInt( bound: count + 1);
    }
    TextView randomView = view.getRootView().findViewById(R.id.textview_random);
    randomView.setText(randomNumber.toString());

    view.findViewById(R.id.button_second).setOnClickListener((view) -> {
            NavHostFragment.findNavController( fragment: SecondFragment.this)
                    .navigate(R.id.action_SecondFragment_to_FirstFragment);
    });
```

5. Run the app. Press the Count button a few times, then press the Random button.



Congratulations!!!