

RectLab: A Rectangle Perimeter Calculator made on Kotlin using Android Studio

M. Ahabb Sheraz, Reg. no. 2021327, CS121 B Semester Project

Problem Statement:

The main aim was to create an application, to be made on Kotlin using Android Studio, that can calculate the perimeter of a rectangle.

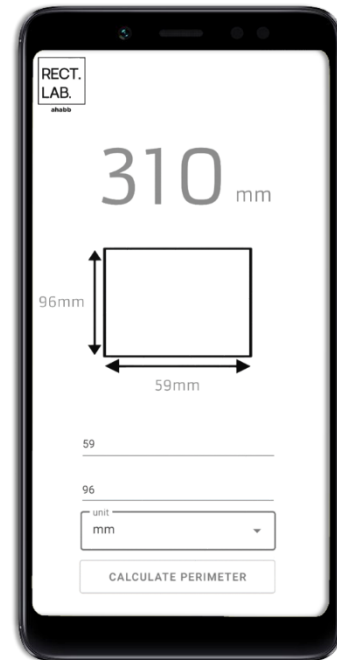
Introduction:

Rectangle Laboratory, RectLab for short, is a simple application that can calculate the perimeter of almost any rectangle. The naming convention was inspired by the much capable MATLAB.

Algorithm Design and Code:

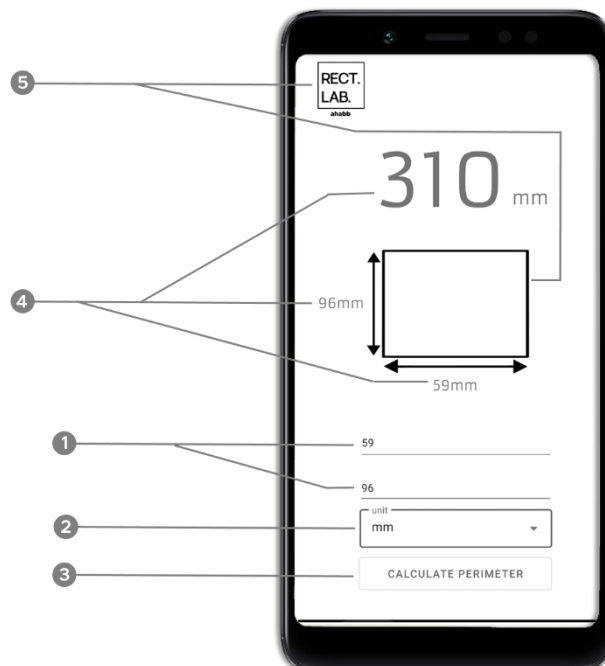
This section has been divided into four main parts:

- Essential Libraries and Defining Variables
- Button Call Function
- Exposed Drop Down Menu Implementation



RectLab app running on Redmi Note 5 Pro.

- 1 Number-only TextFields
- 2 Exposed Drop-Down Menu
- 3 Button
- 4 TextViews
- 5 Scalable Vector Graphics



Features of the Application.

1. Importing Essential Libraries and Defining Variables

```
1 package com.example.rectlab
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.widget.*
6 import androidx.core.content.ContentProviderCompat.requireContext
7 import androidx.core.splashscreen.SplashScreen.Companion.installSplashScreen
8 import kotlin.math.abs
9
```

Libraries that loads and links android studio widgets and APIs

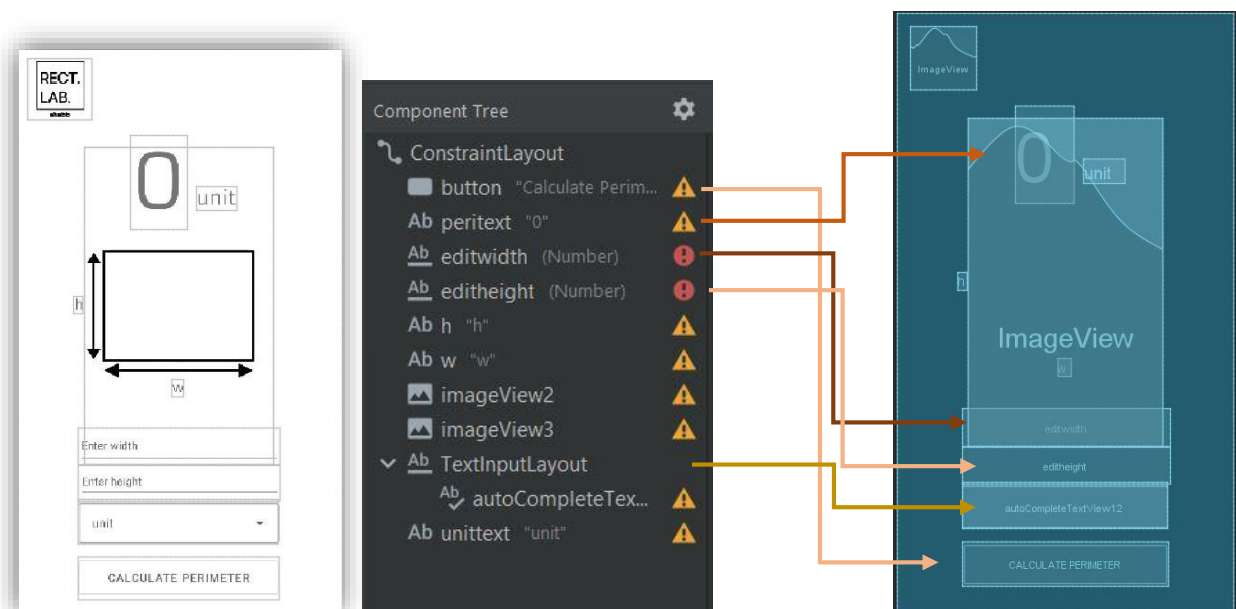
Unused libraries

This library will be used to take absolute value of numbers

Importing essential libraries into MainActivity.kt

```
10 class MainActivity : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_main)
14
15         //declaring variables in the main function
16         val units = resources.getStringArray(R.array.units) //already declared a string array named units in strings.xml
17         val adapter = ArrayAdapter< context: this, R.layout.dropdown_item, units>
18         val autoCompleteTV = findViewById<AutoCompleteTextView>(R.id.autoCompleteTextView12) //declared to change text on drop down menu
19         autoCompleteTV.setAdapter(adapter)
20
21         val calbutton = findViewById<Button>(R.id.button) //declared a button and connect it to app button using id
22         val peritext = findViewById<TextView>(R.id.peritext) //declared a textview and connect it to app textview using id
23         val unittext = findViewById<TextView>(R.id.unittext) //declared a textview and connect it to app textview using id
24         val diswidth = findViewById<TextView>(R.id.w) //declared a textview and connect it to app textview using id
25         val disheight = findViewById<TextView>(R.id.h) //declared a textview and connect it to app textview using id
26         val height = findViewById<TextView>(R.id.editheight) //declared a textfield and connect it to app textfield using id
27         val width = findViewById<TextView>(R.id.editwidth) //declared a textfield and connect it to app textfield using id
28     }
29 }
```

Declaring variables by linking them to their respective widget IDs (MainActivity.kt)



Widgets with their IDs

2. Button Call Function

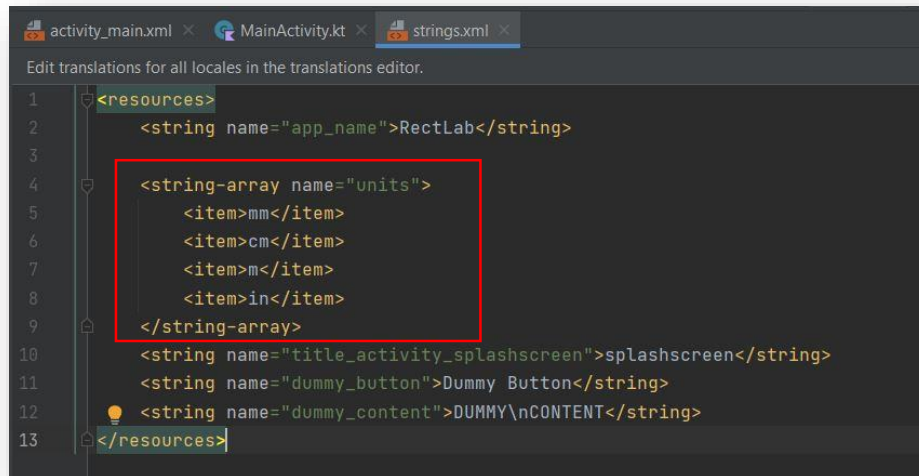
```
calbutton.setOnClickListener {  
    val h = height.text.toString().toInt() //height = findViewById<TextView>(R.id.editheight)  
    val w = width.text.toString().toInt()  //width = findViewById<TextView>(R.id.editwidth)  
    val h1 = abs(h) //makes sure to take absolute value of height  
    val w1 = abs(w) //makes sure to take absolute value of height  
    peritext.text = (2 * (h1 + w1) ).toString() //calculates perimeter and converts it to string to display it  
    diswidth.text = w1.toString() + autocompleteTV.getText().toString()  
    disheight.text = h1.toString() + autocompleteTV.getText().toString()  
    unitext.text = autocompleteTV.getText().toString()  
    //easter eggs  
    if (w1 >= 100000 || h1 >= 100000){  
        Toast.makeText(this, "OH NO! A REKTANGLE!!!", Toast.LENGTH_LONG).show()  
    }  
    if (diswidth.text == disheight.text){  
        Toast.makeText(this, "Rectangle is a Square.", Toast.LENGTH_LONG).show()  
    }  
}
```

Button Call Function

It is here that the main operation occurs. The user enters both the height and the width in their respective number-only text fields. This input data gets first converted to a string and then to an integer before getting stored in the variables 'h' and 'w' respectively. To avoid subtraction when calculating the perimeter, the absolute values of variable 'h' and 'w' gets stored in variables 'h1' and 'w1' respectively. The perimeter is found by using the formula of perimeter of a rectangle, $(width + height) * 2$. The perimeter is then converted to string from integer so to display it. The values of 'w1' and 'h1' are also converted to string and get displayed on the TextViews.

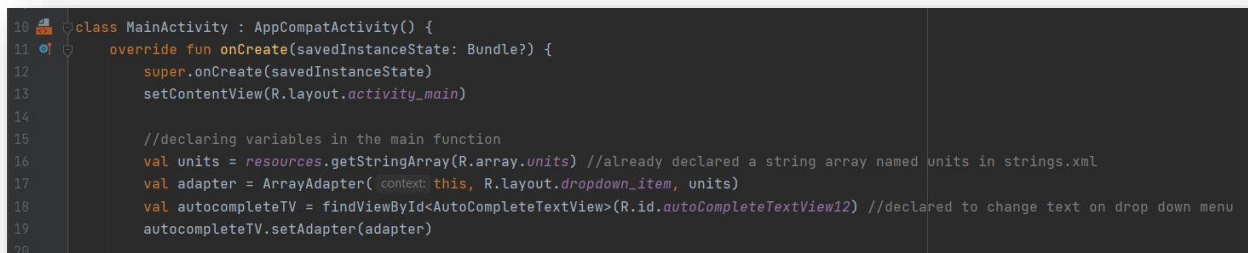
In the function, we have placed two if-statements. One checks whether the rectangle is a square or not by comparing the string values of the width and the height. The other one is more of an Easter egg that displays the message, "OH NO!!! A REKTANGLE", if either the absolute value of height or width is greater than 100000.

3. Implementation of the Exposed Drop Down Menu



```
1 <resources>
2   <string name="app_name">RectLab</string>
3
4   <string-array name="units">
5     <item>mm</item>
6     <item>cm</item>
7     <item>m</item>
8     <item>in</item>
9   </string-array>
10  <string name="title_activity_splashscreen">splashscreen</string>
11  <string name="dummy_button">Dummy Button</string>
12  <string name="dummy_content">DUMMY\nCONTENT</string>
13 </resources>
```

strings.xml



```
10 class MainActivity : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_main)
14
15         //declaring variables in the main function
16         val units = resources.getStringArray(R.array.units) //already declared a string array named units in strings.xml
17         val adapter = ArrayAdapter<String>(context: this, R.layout.dropdown_item, units)
18         val autoCompleteTV = findViewById<AutoCompleteTextView>(R.id.autoCompleteTextView12) //declared to change text on drop down menu
19         autoCompleteTV.setAdapter(adapter)
20     }
21 }
```

MainActivity.kt

With the help of the Exposed Drop-Down Menu, we can quickly and easily select an item from a list. In our case, the user uses the drop-down menu to choose the units for the perimeter output. I used a TextInputLayout that contains an AutoCompleteTextView to use Exposed Drop-Down Menu. I added a string array items which contains the units to the string.xml file. We need this data to inflate the drop-down items. In MainActivity.kt, the variable units gets the reference to the unit string array from strings.xml. Then I created an array adapter and pass the context, my dropdown layout, and string array units. Finally, set adapter to the autoCompleteTextView in the autoCompleteTV.

4. Conclusion

This project introduced me to the world of Android Application Development and gave me the chance to learn a lot of new things. Through this project, I was introduced to Kotlin for Android Studio. As for the project, I am happy with how it turned out to be. In the future, I plan to add more features such as the ability to convert units of both the height and width.