



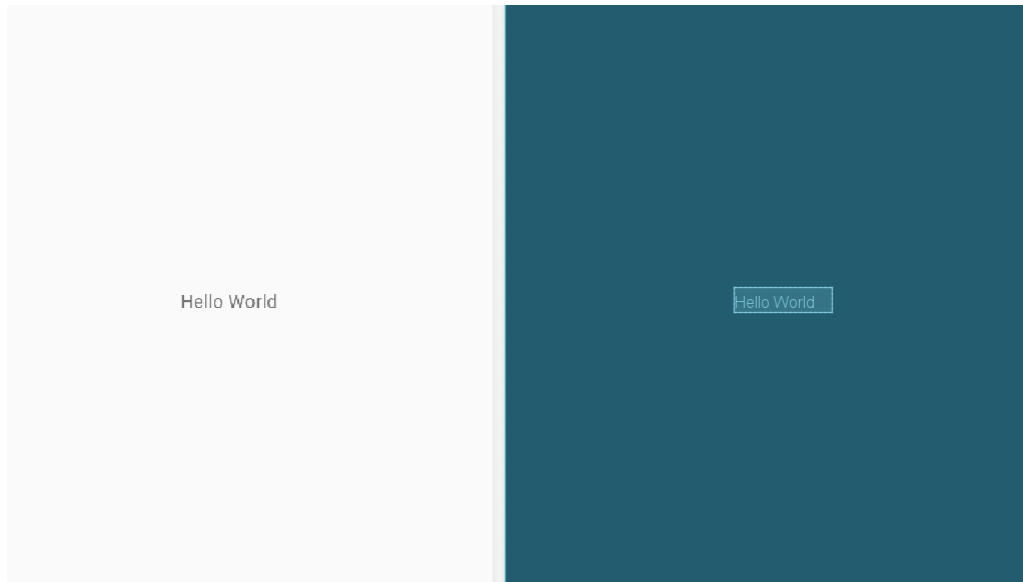
LECTURE 14: UI PROGRAMMING PART I

BY LINA HAMMAD & AHMAD BARGHASH

This Lecture, which will serve as your introduction to UI programming. After this class you will be familiar with: Buttons, Image Buttons, Image Views, Text Views, and Edit Text.

MODIFYING THE USER INTERFACE

- As we mentioned previously, the user interface design for our activity is stored in a file named *activity_main.xml* which, in turn, is located under *app -> res -> layout* in the project file hierarchy. Once located in the Project tool window, double-click on the file to load it into the user interface Layout Editor tool which will appear in the center panel of the Android Studio main window:




```
package com.example.myapp6

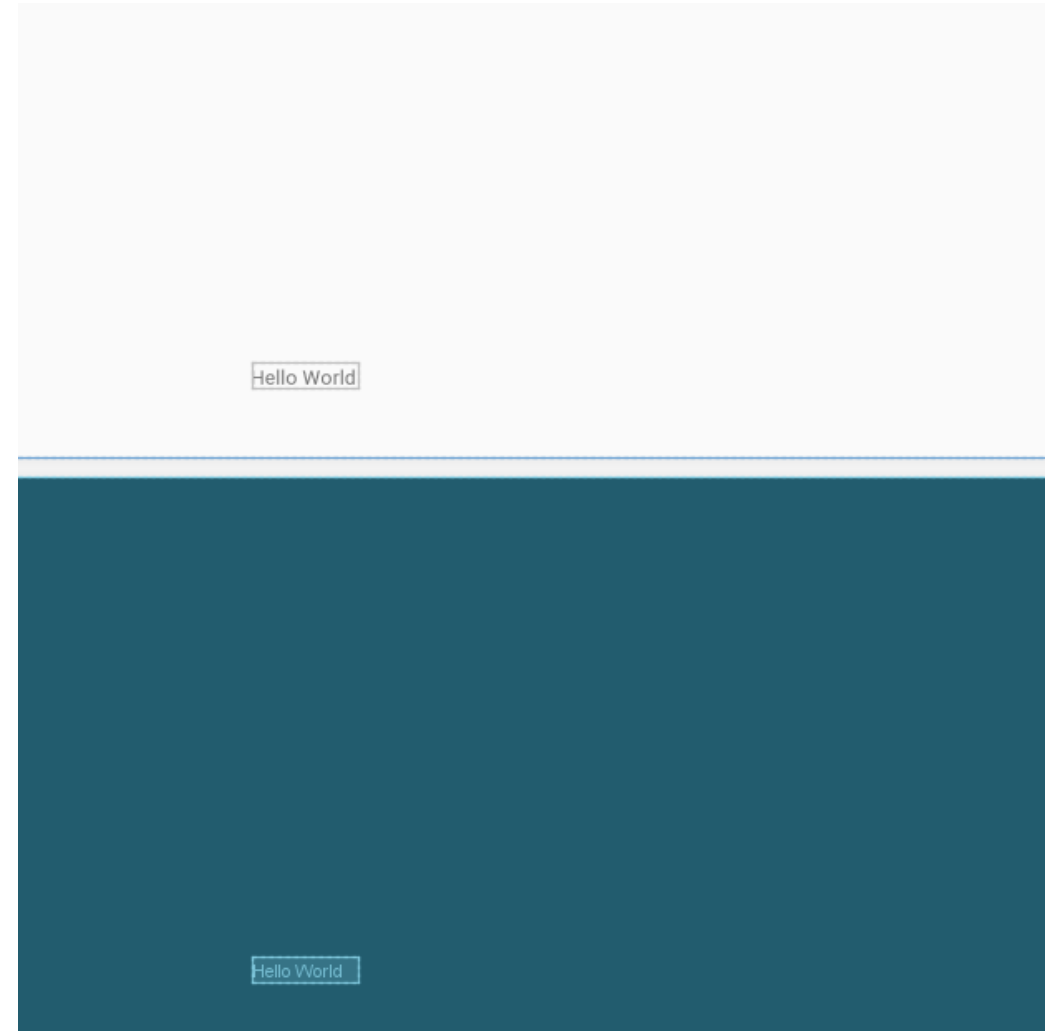
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

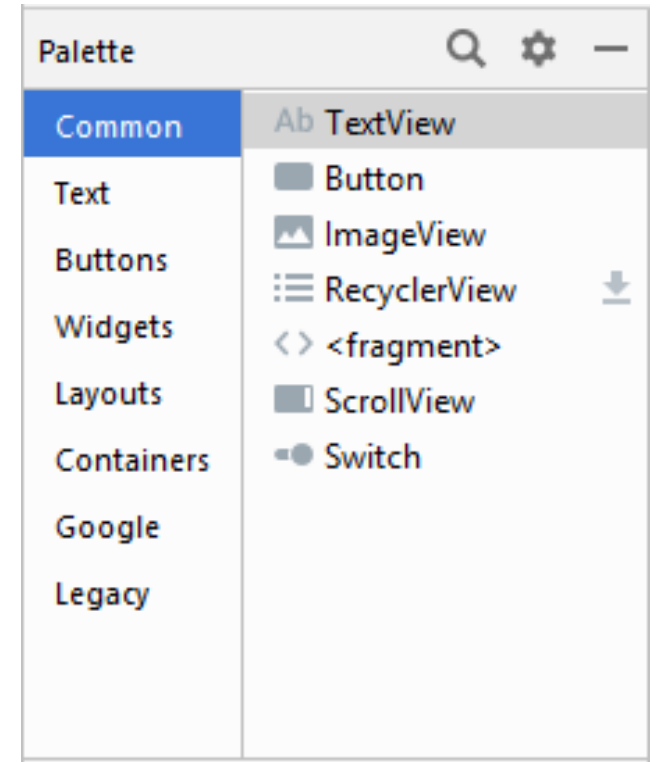
LANDSCAPE OR PORTRAIT

- To change the orientation of the device representation between landscape and portrait simply use the drop-down menu immediately to the left of the device selection menu showing the  icon. If you choose the Landscape option, the layout view will appear as the following:



DRAG AND DROP

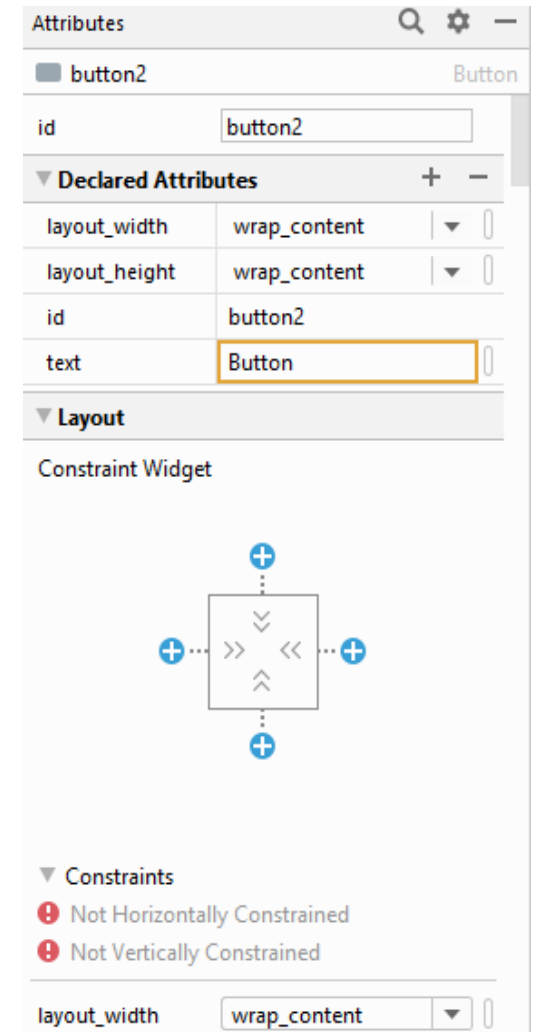
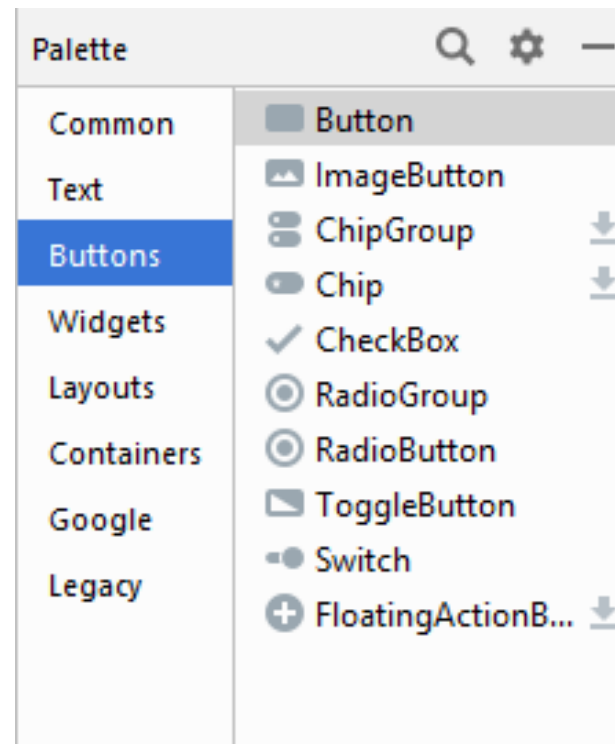
- In the last lecture we learned how to design the desired layout using XML.
- Alternatively, you can drag the needed elements from the Palette.
- Remember to define the horizontal and vertical constraints of every element you use.



Android Studio palette

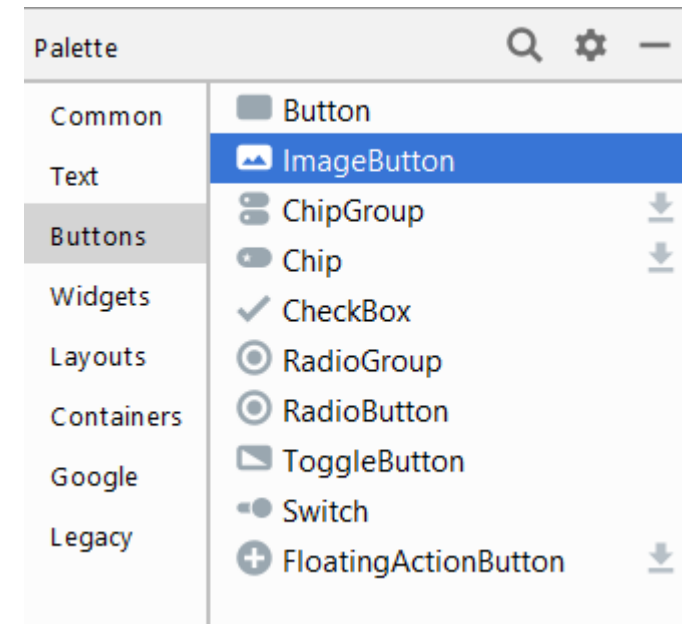
BUTTONS -- BUTTON

- Represents a push-button widget that registers when the screen is touched within its bounds.
- We can either use XML or the Attributes menu to modify the features of the button.



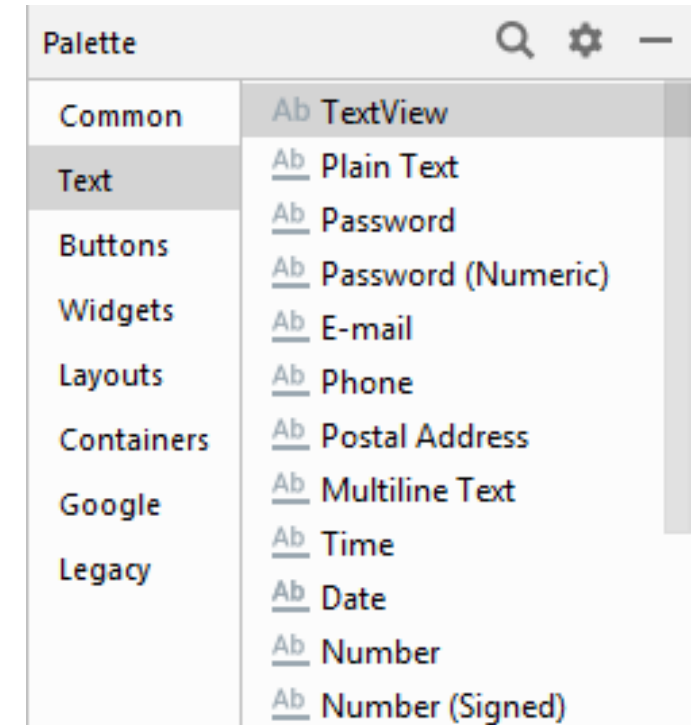
WIDGETS – IMAGE BUTTON

- `ImageButton` works similar to the `Button`, except that it also displays an image. The `ImageButton` class is a subclass of the `ImageView` class.
- To import image for the button, you should follow the next steps:
 1. Right click on `res` -> `drawable` folder and click show in explore.
 2. Open the `drawable` folder then drag and drop your image in the `drawable` folder.
 3. The image should now appear in the `res` -> `drawable` folder.
 4. Drag and drop the `ImageButton` element from the palette to the layout, then choose the image you imported from `drawable` then `project` where you will find the image.



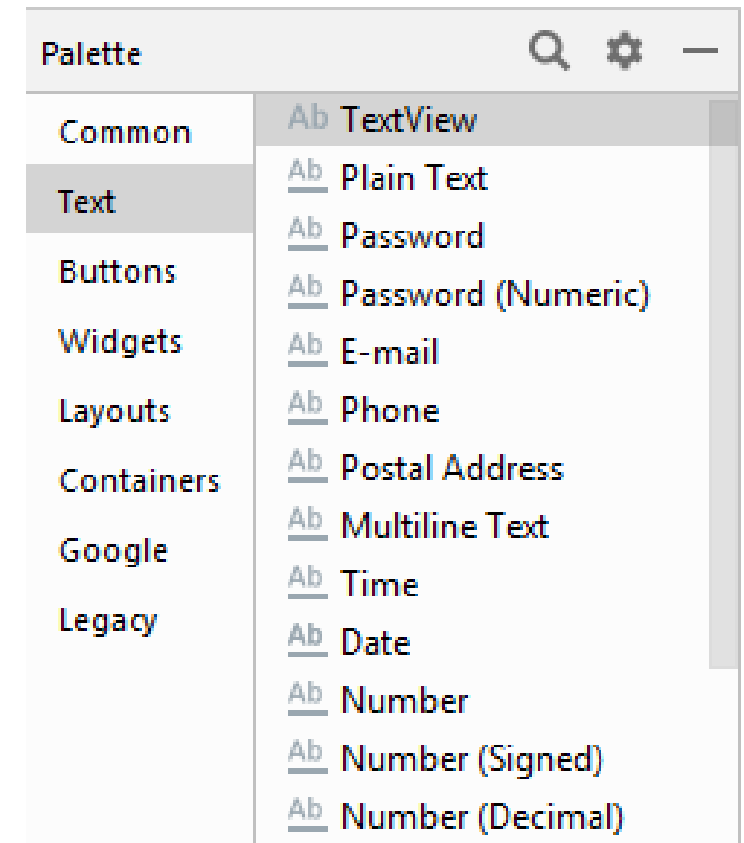
TEXTS -- TEXTVIEW

- Used either to input the text or to view it.
- Need to be converted to text after input or before output.
- We can either use XML or the Attributes menu to modify the features of the button.
- The TextView is used to display text to the users and this text information can't be edited by the users.



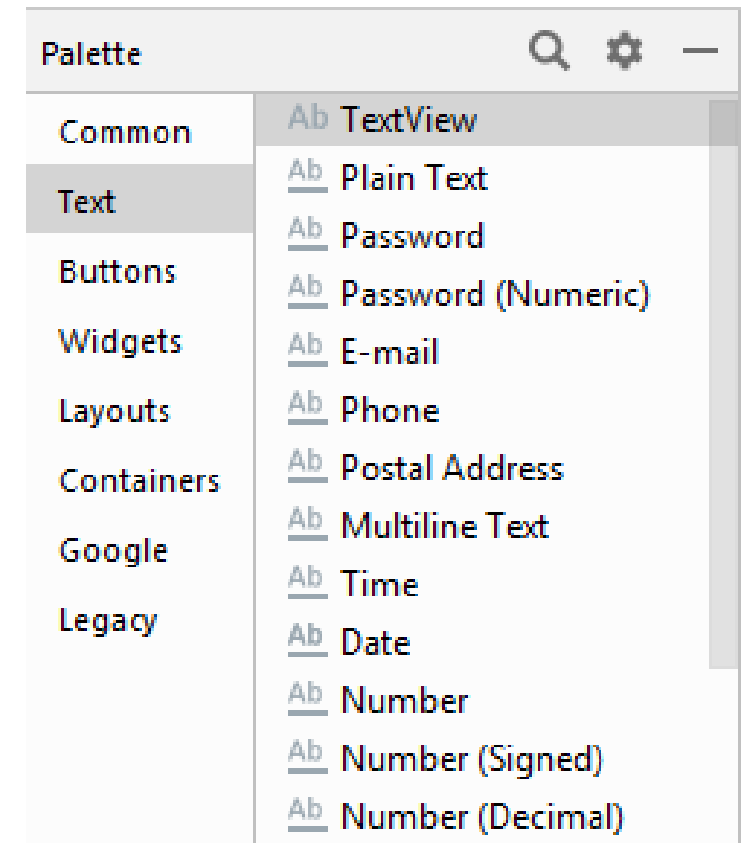
TEXTS -- EDITTEXT

- Following are the pre-built EditText views available in Palette pane:
 - a) Plain Text
 - Shows the standard text keyboard.
 - b) Password
 - Shows the standard text keyboard and masks the text that is entered for privacy.
 - c) Password (Numeric)
 - Shows a numeric keyboard and masks the text that is entered for privacy.
 - d) E-mail
 - Shows the standard text keyboard with the addition of the @ character.
 - e) Phone
 - Shows the phone-style keyboard.
 - f) Postal Address
 - Shows the standard text keyboard.
 - g) Multiline Text
 - Shows the standard text keyboard with the addition of the Enter key for adding new line.



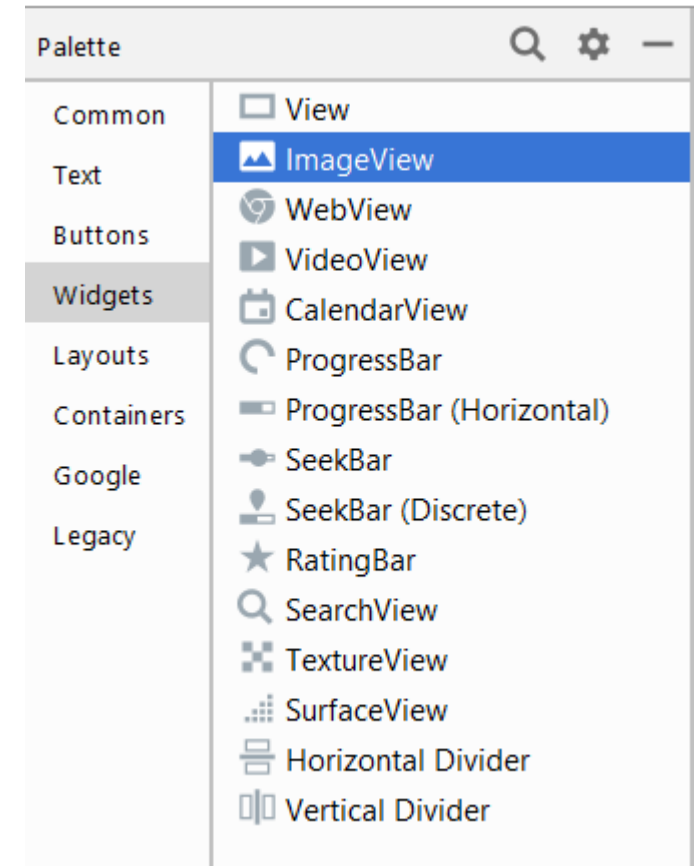
TEXTS -- EDITTEXT

- Following are the pre-built EditText views available in Palette pane:
 - h) Time
 - Shows the numeric keyboard with the addition of the : character.
 - i) Date
 - Shows the numeric keyboard with the addition of the / character.
 - j) Number
 - Shows the basic numeric keyboard.
 - k) Number (Signed)
 - Shows the basic numeric keyboard. Allows a + or - character at the start.
 - l) Number (Decimal)
 - Shows the basic numeric keyboard. Allows a decimal point . to provide fractional values.



WIDGETS – IMAGE VIEW

- The **ImageView** is a view that shows images on the device screen.
- To import image in your layout you should follow the next steps:
 1. Right click on *res -> drawable* folder and click show in explore.
 2. Open the drawable folder then drag and drop your image in the drawable folder.
 3. The image should now appear in the *res -> drawable* folder.
 4. Drag and drop the **ImageView** element from the palette to the layout, then choose the image you imported from drawable then project where you will find the image.



ADDING ACTIONS

- Start by building this layout
 - EditText ID: Radius
 - Button ID:BT1
 - TextView ID: textView
- The application should get the radius from the user through the text box then when the button is clicked the area of the circle should be calculated and displayed in the textView.

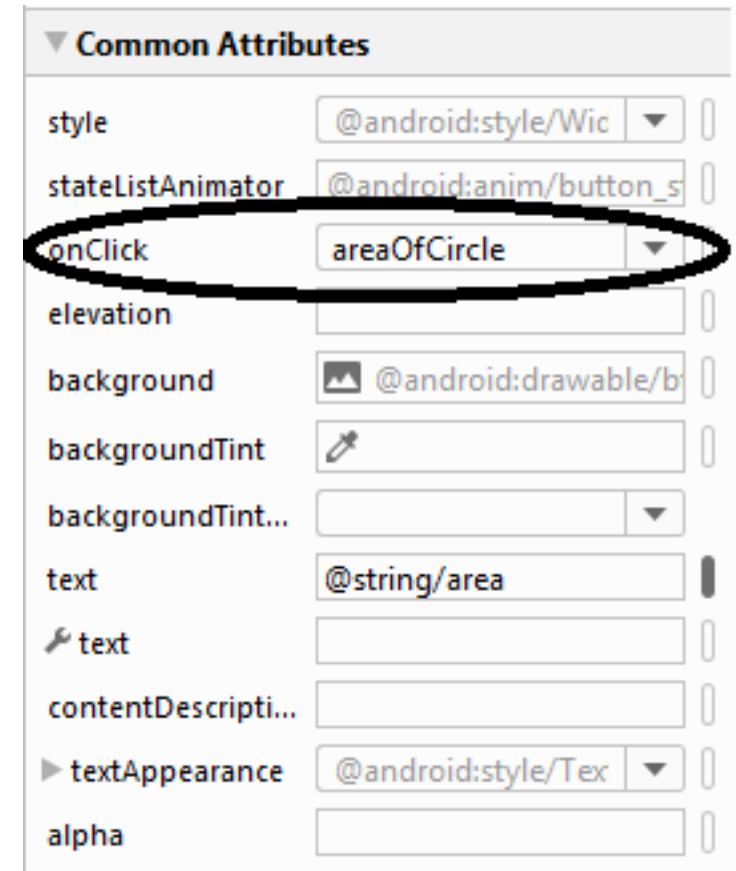
Enter Radius:

**CALCULATE
AREA**

Area of Circle

ADDING ACTIONS

- You can connect an `onClick` action to a method you create. For example, you can use the `onClick` attribute from the attribute menu.
- You need to create a method with the same name in the class file `MainActivity.kt`.
- You can start by creating the method in `MainActivity.kt`, then you will find it available in the `onClick` attribute.



ADDING ACTIONS

- Sample method to calculate the area of a circle

```
fun areaOfCircle(view:View) {  
    val radius = Radius.text.toString().toFloat()  
    val area = radius * radius * 3.14  
    textView.text = area.toString()  
}
```

- You will notice that android studio requires to import a needed class and guides you to the solution by pressing ALT+ENT



The screenshot shows a code editor with a function definition `fun areaOfCircle(view:View) {`. The parameter `view` is of type `View`, which is underlined in red. A blue tooltip box is overlaid on the code, containing a question mark icon and the text `android.view.View? Alt+Enter`. The tooltip is pointing to the `View` type in the function signature. In the background, another line of code `setC...(_main)` is partially visible.

- Will import automatically

```
import android.view.View
```

ANOTHER METHOD TO ADD THE SAME ACTION

- Use `onClickListeners` within the main `onCreate` function
 - `setOnClickListener` – triggered when a button is clicked.
 - `setOnLongClickListener` – triggered when a button is pressed for a longer duration.
- You will need to assign an ID for each element you are using such as the button, textView, and editText in the previous example such as
 - `val bt1= findViewById<Button>(R.id.BT1)`
 - `val tv= findViewById<TextView>(R.id.textView)`
 - `val et= findViewById<EditText>(R.id.Radius)`
- Needed classes will be automatically imported or use ALT+ENT to import them such as
 - `import android.widget.Button`
 - `import android.widget.EditText`
 - `import android.widget.TextView`

CODE WILL LOOK LIKE

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val bt1 = findViewById<Button>(R.id.BT1)  
        val tv = findViewById<TextView>(R.id.textView)  
        val et = findViewById<EditText>(R.id.Radius)  
  
        bt1.setOnClickListener {  
            val radius = et.text.toString().toFloat()  
            val area = radius * radius * 3.14  
            tv.text = area.toString()  
        }  
    }  
}
```

EXTRA CLASSES

- Toast in Android is used to display a piece of text for a short span of time.
- When Toast is made, the piece of text is appears on the screen, stays there on the screen for about 2 or 3 to 5 seconds and disappears.

```
Toast.makeText(this, "Success", Toast.LENGTH_SHORT).show()  
Toast.makeText(this, "Longer Success", Toast.LENGTH_LONG).show()
```

- Random class is used to generate a random number of your choice

```
var i = Random.nextInt(10)
```