Beginner’s Guide To Android Studio

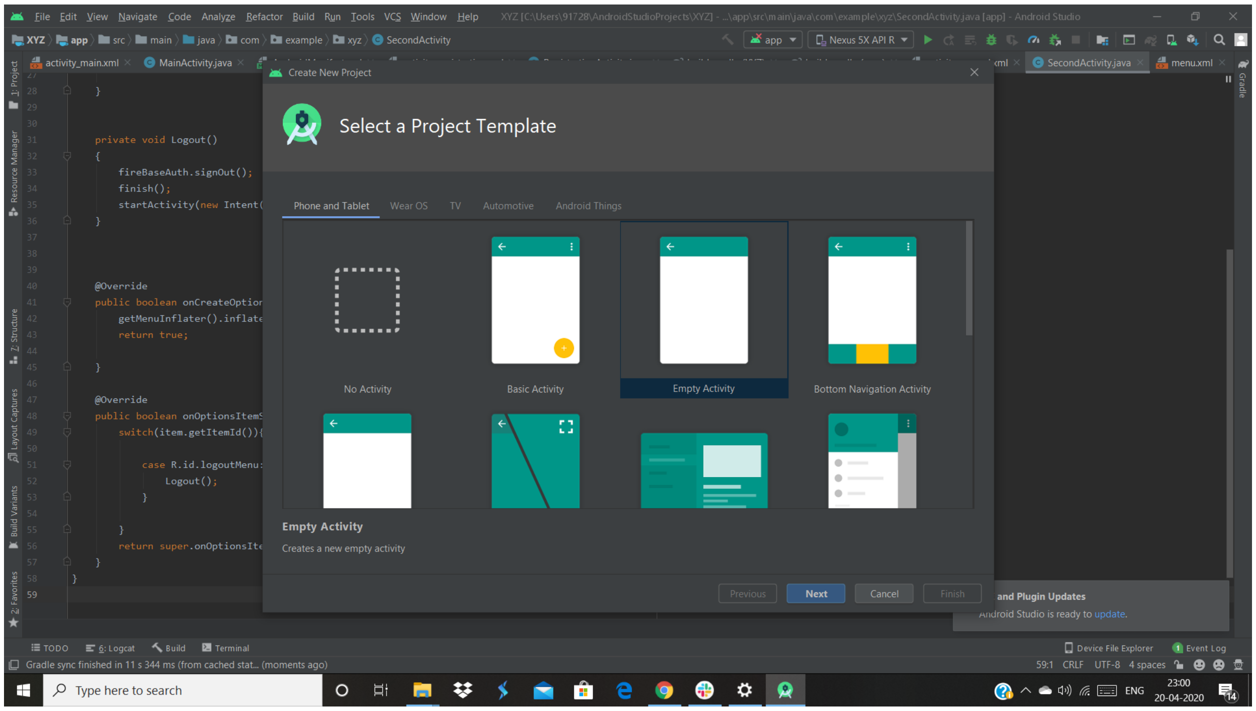
Android studio is a framework that includes every tool necessary to develop Android apps and games. Technically, it is an official IDE (Integrated Development Environment) for facilitating app and game development

Let us begin this journey of developing Apps and Games:

Step 1: *Downloading Android Studio: Android Studio*

Step 2: *Starting a Project(Creating an Activity):*

- File→ New→ New Project


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
XYZ [C:\Users\91728\AndroidStudioProjects\XYZ] - ...\app\src\main\java\com\example\xyz\SecondActivity.java [app] - Android Studio
XYZ app src main java com example xyz SecondActivity
app
Nexus 5X API R
activity_main.xml
MainActivity.java
SecondActivity.java
menu.xml
Create New Project
Select a Project Template
Phone and Tablet
Wear OS
TV
Automotive
Android Things
No Activity
Basic Activity
Empty Activity
Bottom Navigation Activity
Previous
Next
Cancel
Finish
TODO
6: Logcat
Build
Terminal
Device File Explorer
Event Log
Type here to search

```
    }


    private void Logout()
    {
        fireBaseAuth.signOut();
        finish();
        startActivity(new Intent(
    }



    @Override
    public boolean onCreateOptior
        getMenuInflater().inflate
        return true;

    }

    @Override
    public boolean onOptionsItemS
        switch(item.getItemId()){

            case R.id.logoutMenu:
                Logout();
            }

        }
        return super.onOptionsIte
    }
}
```

Empty Activity

Creates a new empty activity

...and Plugin Updates

Android Studio is ready to update.

Gradle sync finished in 11 s 344 ms (from cached stat... (moments ago)

Choose an activity as desired. For beginners, let us start by creating an empty activity.

Next, you assign a name to your application, and choose a language.
Generally, codes are written in two languages: Java and Kotlin.
You can choose any of the two. For this tutorial we will follow Java.

Minimum SDK: Here is where you decide to make a trade off. Choosing the version of API vs the target devices.

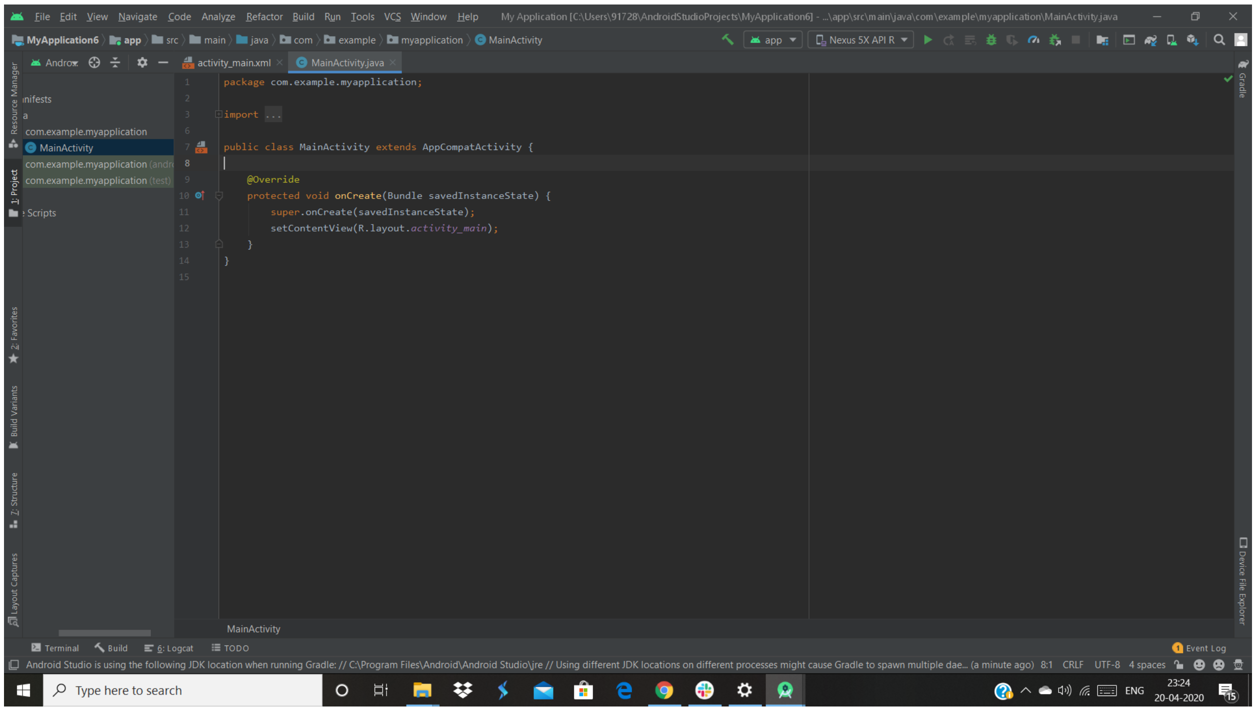
Well,

- API 16(Jelly Bean): Supported by 99.8% devices
- And API 26(Oreo): Supported by 60.8%devices

If you use older versions,your app may miss out on many of the "modern" features and If you use the latest versions , as you might have guessed, your app will not be supported by many devices.

- ➢ Personally, I recommend using an API 23 version but of course it totally depends on what tasks you want your apps to perform.

And Finally The Finish Option. Your Project is Loaded.



My Application [C:\Users\91728\AndroidStudioProjects\MyApplication6] - ...\app\src\main\java\com\example\myapplication\MainActivity.java

```java
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

```
Android Studio is using the following JDK location when running Gradle: // C:\Program Files\Android\Android Studio\jre // Using different JDK locations on different processes might cause Gradle to spawn multiple dae... (a minute ago)
```

This is what the screen looks like(For an empty Activity)

Step 3: *Creating a Virtual Device*

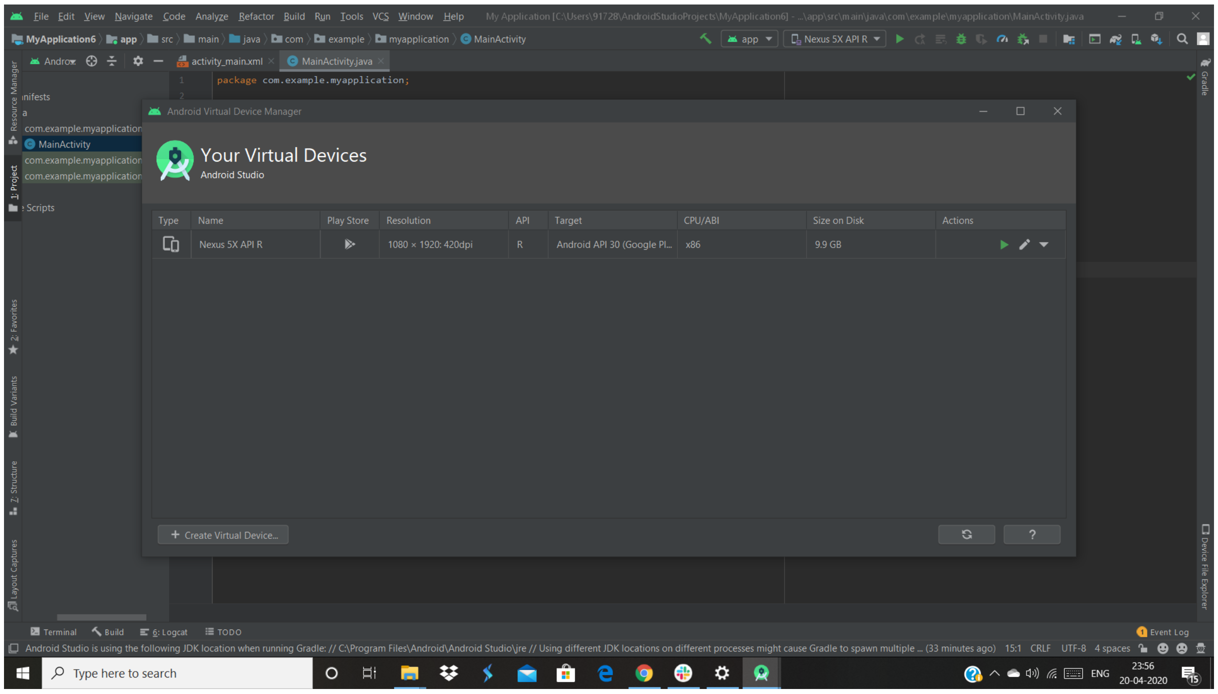
Well, once you start working on a project, you would want to run your app on every step. Android Studio has its built in Emulator wherein you can create virtual devices and run your apps on them.
Alternatively, you can use external emulators like Genymotion Download Genymotion. Also you can connect your own device and run your apps using the Developer Mode.

Here is how you can create a virtual device using Android In-Built Emulator:

- Launch **AVD Manager**(An Option on the Upper Right Corner)

A window similar to this will Pop Up. I have already created a virtual device, yours might be empty.

- Go to the option-" Create Virtual Device"



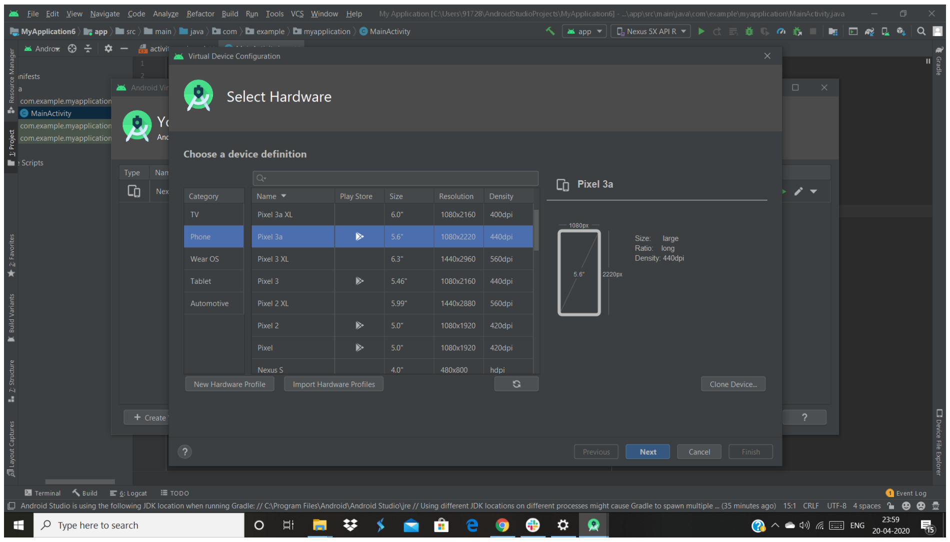
```
package com.example.myapplication;
```

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API R		1080 × 1920: 420dpi	R	Android API 30 (Google Pl...	x86	9.9 GB	

```
Android Studio is using the following JDK location when running Gradle: // C:\Program Files\Android\Android Studio\jre // Using different JDK locations on different processes might cause Gradle to spawn multiple ... (33 minutes ago)
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
My Application [C:\Users\91728\AndroidStudioProjects\MyApplication6] - ...\app\src\main\java\com\example\myapplication\MainActivity.java
MyApplication6 app src main java com example myapplication MainActivity
app
Nexus 5X API R
activity_main.xml
MainActivity.java
Android Virtual Device Manager
+ Create Virtual Device...
?
Terminal
Build
6: Logcat
TODO
Event Log
15:1 CRLF UTF-8 4 spaces
Type here to search
ENG
23:56
20-04-2020



Category	Name	Play Store	Size	Resolution	Density
TV	Pixel 3a XL		6.0"	1080x2160	400dpi
Phone	Pixel 3a	▷	5.6"	1080x2220	440dpi
Wear OS	Pixel 3 XL		6.3"	1440x2960	560dpi
Tablet	Pixel 3	▷	5.46"	1080x2160	440dpi
Automotive	Pixel 2 XL		5.99"	1440x2880	560dpi
	Pixel 2	▷	5.0"	1080x1920	420dpi
	Pixel	▷	5.0"	1080x1920	420dpi
	Nexus S		4.0"	480x800	hdpi

Size: large
Ratio: long
Density: 440dpi

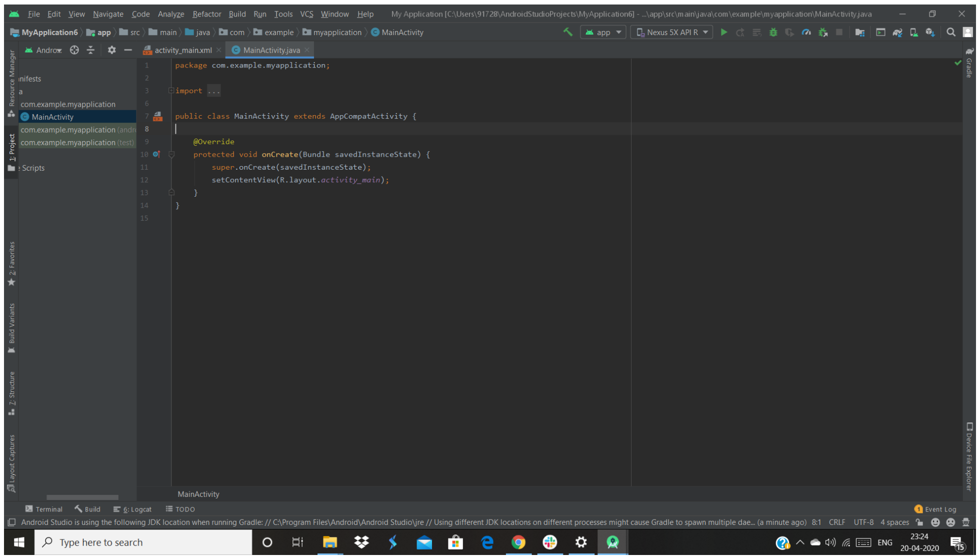
❖ Select a suitable device→ Next → Check your API level → Next → Name your Device→ Finish.

Your Virtual Device is All Set.

<u>Step 4:</u> *Working On the Project*

When you create an activity, you see two files on the screen-

- A Java File(Main Activity.Java)
- An XML File(activty_main.Xml)



```java
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

- Java File is where you work on the Backend Part.
    - Here, you write codes and methods for every single feature.

- XML File is basically the Frontend Part.
    - You have two options:
        - Either you can add features manually by writing codes.
        - Alternatively you can drag and drop features and change their attributes.

Let us build our first app: ***A CALCULATOR***

a) Building the interface:

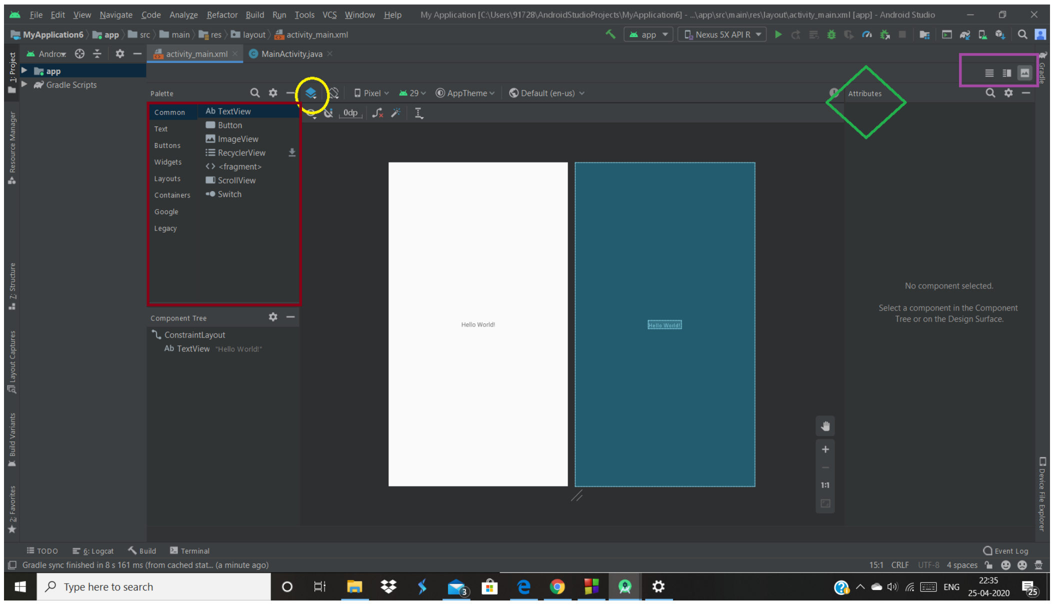
Here is the XML File:(Refer to picture below)

- *The yellow circle:* The option gives you a choice to view your design and/or your blueprint.
    - A blueprint is basically the wireframe of your design.

- *The purple rectangle:* This is the section from where you can switch to design and/or text of an XML file. As mentioned, you can either add features by writing codes or drag and drop. For writing codes, we use the text section and for drag/drop, the design section.

- *The red Palette:* List of options from which you can pick up features

- *The green rhombus:* Attributes corresponding to each feature will be displayed here.



No component selected.

Select a component in the Component Tree or on the Design Surface.

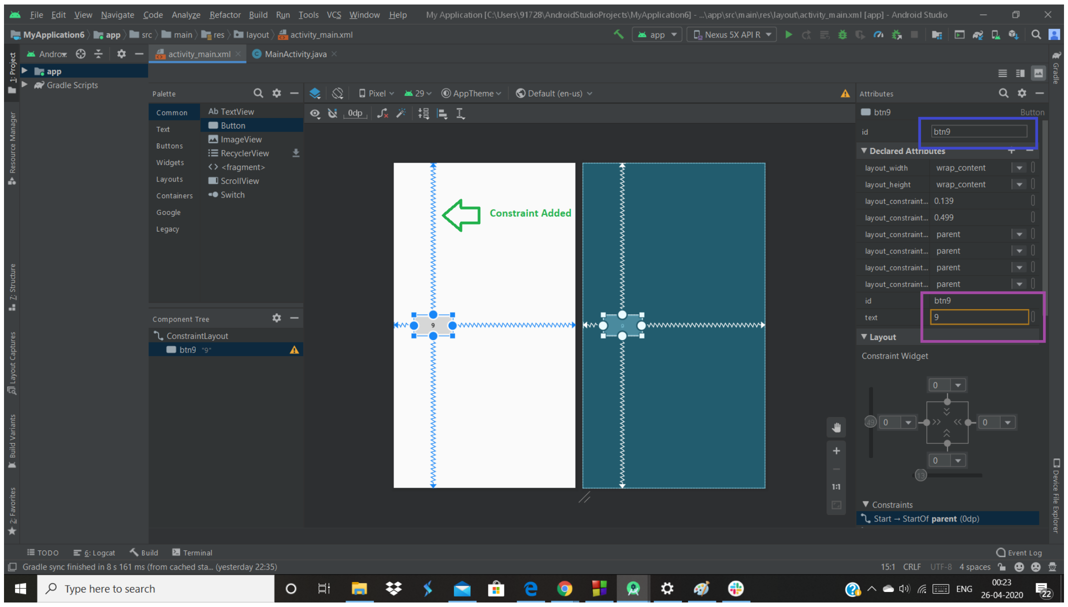
```
Gradle sync finished in 8 s 161 ms (from cached stat... (a minute ago)
```

In XML, you can use various layouts. The most common ones are: RELATIVE and CONSTRAINT(Default)

We will use the constraint layout for our app.

- ❖ Go to the XML Design File
- ❖ Drag and drop a button from the Palette.

We will have to constrain it from all the four sides to fix its position.



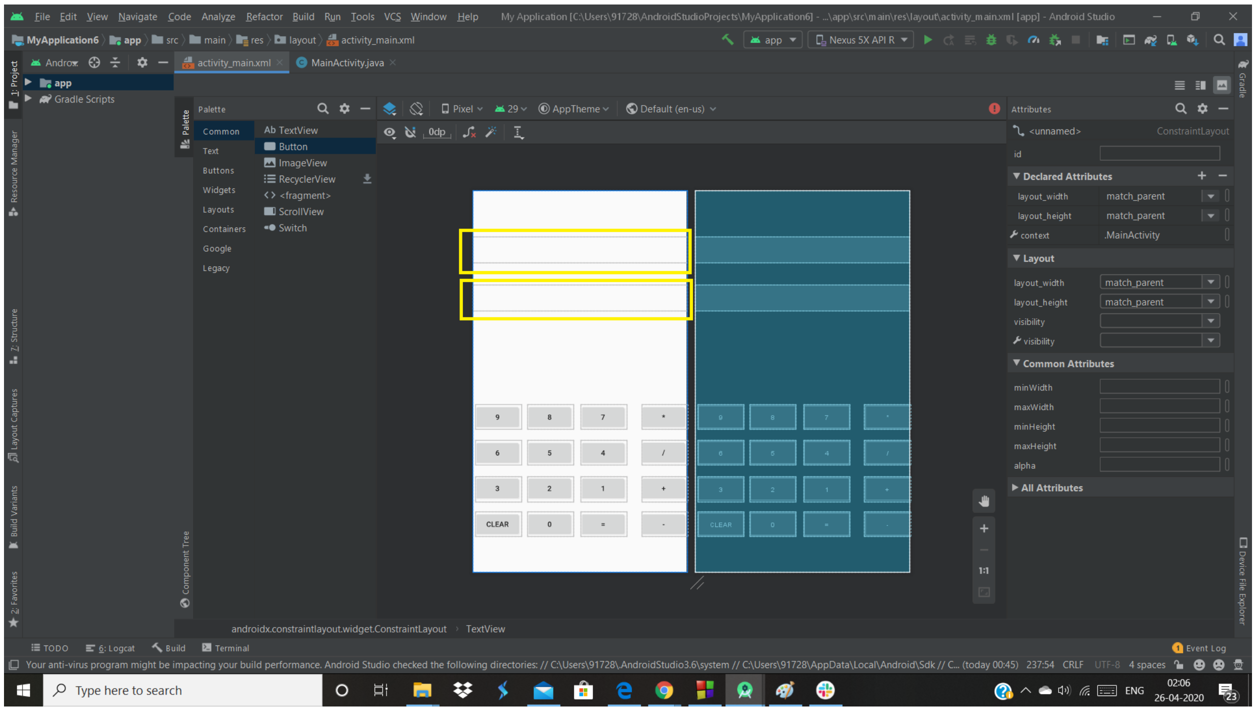
id	btn9
layout_width	wrap_content
layout_height	wrap_content
layout_constraint...	0.139
layout_constraint...	0.499
layout_constraint...	parent
layout_constraint...	parent
layout_constraint...	parent
layout_constraint...	parent
id	btn9
text	9

Start → StartOf parent (0dp)

```
Gradle sync finished in 8 s 161 ms (from cached sta... (yesterday 22:35)
```

We have constrained the button from all the four sides(you just need to drag the button to a side from the center) , Also I have changed the button text to 9 (refer to pink rectangle) and assigned an id to the button The green rhombus:(Here btn9).

Similarly add other buttons, rename them and assign an ID.



Attributes

<unnamed> ConstraintLayout

id

Declared Attributes

layout_width	match_parent
layout_height	match_parent
context	.MainActivity

Layout

layout_width	match_parent
layout_height	match_parent
visibility	
visibility	

Common Attributes

minWidth	
maxWidth	
minHeight	
maxHeight	
alpha	

All Attributes

androidx.constraintlayout.widget.ConstraintLayout › TextView

```
Your anti-virus program might be impacting your build performance. Android Studio checked the following directories: // C:\Users\91728\.AndroidStudio3.6\system // C:\Users\91728\AppData\Local\Android\Sdk // C... (today 00:45)   237:54  CRLF  UTF-8  4 spaces
```

Sometimes, there is this warning which says “Hard-coded Text”. You may ignore it for now.
(Incase you want to resolve this, you need to define your string in the string.xml file(res→ values→ string.xml)

- ➢ Read about it here [Hard-coded string]

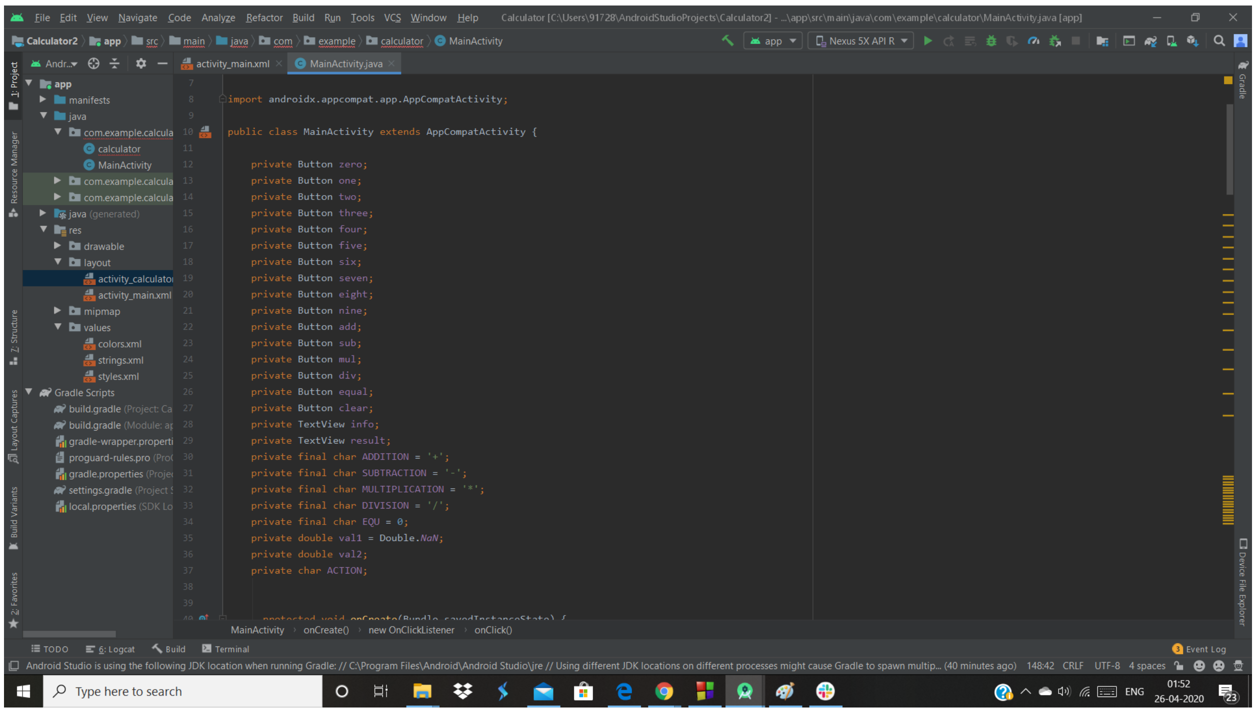
Also as you can see,there are two large yellow rectangles in the design view. Those are the two text views I have added.
One for the control : (Id= tvcontrol) and another for result(Id= tvresult)

- ➔ tvcontol: basically to show the computation
- ➔ tvresult: to show the computed result.

Our XML file is complete.

b) Working on the Java file:

First of all, we declare all the variables that we need.

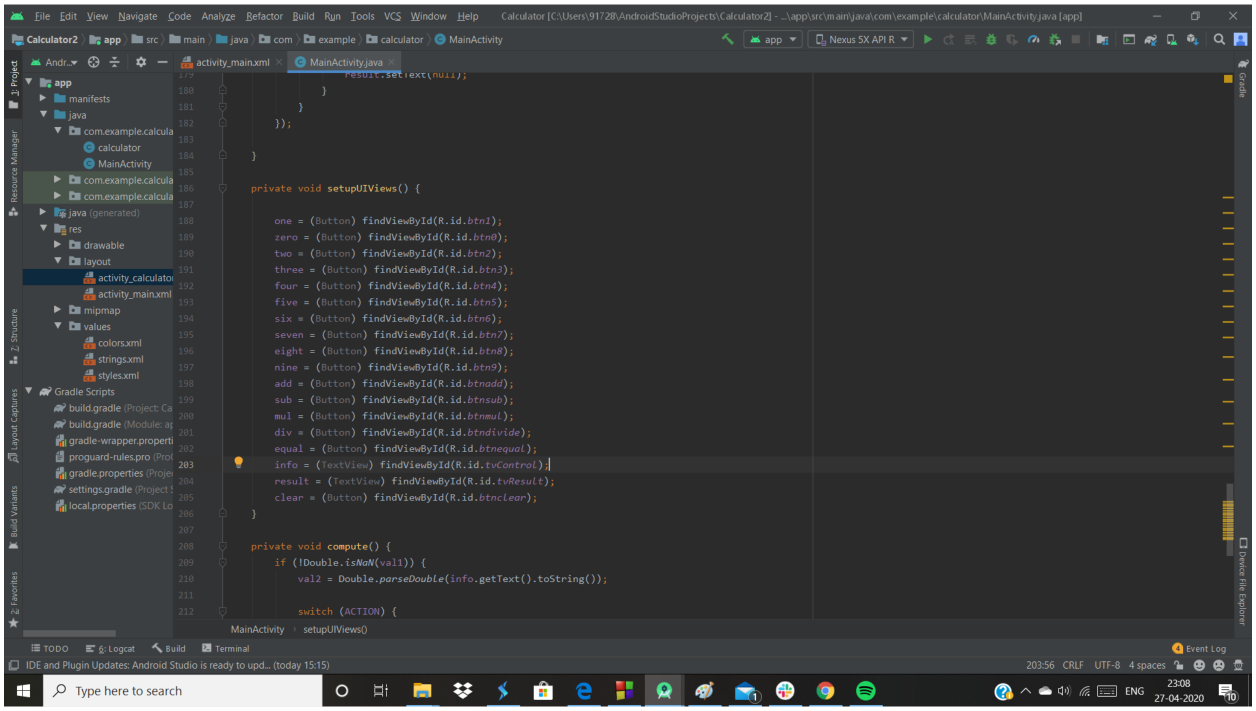


```java

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private Button zero;
    private Button one;
    private Button two;
    private Button three;
    private Button four;
    private Button five;
    private Button six;
    private Button seven;
    private Button eight;
    private Button nine;
    private Button add;
    private Button sub;
    private Button mul;
    private Button div;
    private Button equal;
    private Button clear;
    private TextView info;
    private TextView result;
    private final char ADDITION = '+';
    private final char SUBTRACTION = '-';
    private final char MULTIPLICATION = '*';
    private final char DIVISION = '/';
    private final char EQU = 0;
    private double val1 = Double.NaN;
    private double val2;
    private char ACTION;


      protected void onCreate(Bundle savedInstanceState) {
```

For the buttons part, you can also use arrays. But here for simplicity we will define them separately. We will have to define actions(add/sub…) and two values(the first has been initialized with null) and ofcourse the two text views(info and result).

Next,
As the user clicks on a button we need to read that button and update info.

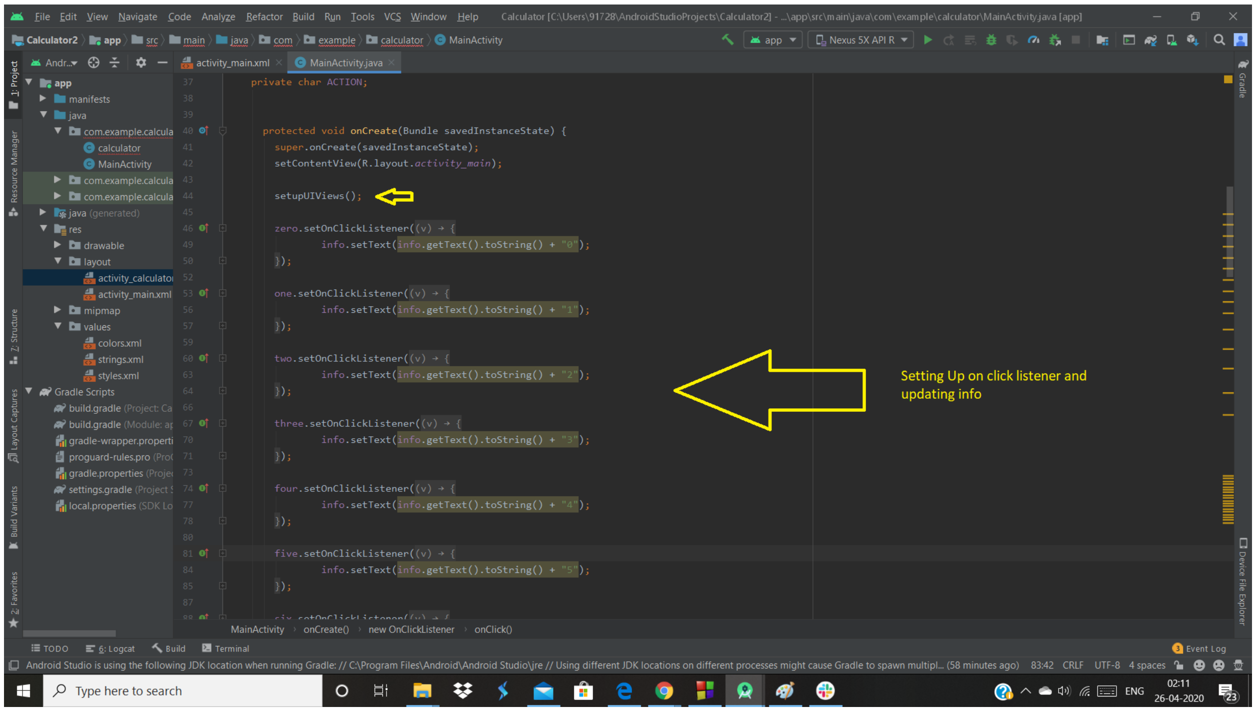
First of all, we will have to set up ids. I have declared a function and made a function call to the same



```java
                    }
                }
            });

        }

        private void setupUIViews() {

            one = (Button) findViewById(R.id.btn1);
            zero = (Button) findViewById(R.id.btn0);
            two = (Button) findViewById(R.id.btn2);
            three = (Button) findViewById(R.id.btn3);
            four = (Button) findViewById(R.id.btn4);
            five = (Button) findViewById(R.id.btn5);
            six = (Button) findViewById(R.id.btn6);
            seven = (Button) findViewById(R.id.btn7);
            eight = (Button) findViewById(R.id.btn8);
            nine = (Button) findViewById(R.id.btn9);
            add = (Button) findViewById(R.id.btnadd);
            sub = (Button) findViewById(R.id.btnsub);
            mul = (Button) findViewById(R.id.btnmul);
            div = (Button) findViewById(R.id.btndivide);
            equal = (Button) findViewById(R.id.btnequal);
            info = (TextView) findViewById(R.id.tvControl);
            result = (TextView) findViewById(R.id.tvResult);
            clear = (Button) findViewById(R.id.btnclear);
        }

        private void compute() {
            if (!Double.isNaN(val1)) {
                val2 = Double.parseDouble(info.getText().toString());

                switch (ACTION) {
```

Declared function: setupUIViews

Again, an array would have been a savior to such ugly codes.



```java
        private char ACTION;


          protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            setupUIViews();

            zero.setOnClickListener((v) → {
                    info.setText(info.getText().toString() + "0");
            });

            one.setOnClickListener((v) → {
                    info.setText(info.getText().toString() + "1");
            });

            two.setOnClickListener((v) → {
                    info.setText(info.getText().toString() + "2");
            });

            three.setOnClickListener((v) → {
                    info.setText(info.getText().toString() + "3");
            });

            four.setOnClickListener((v) → {
                    info.setText(info.getText().toString() + "4");
            });

            five.setOnClickListener((v) → {
                    info.setText(info.getText().toString() + "5");
            });
```

Setting Up on click listener and updating info

Calling the function setupUIViews()

Also, we have written the onSetClickListener function for all our buttons:

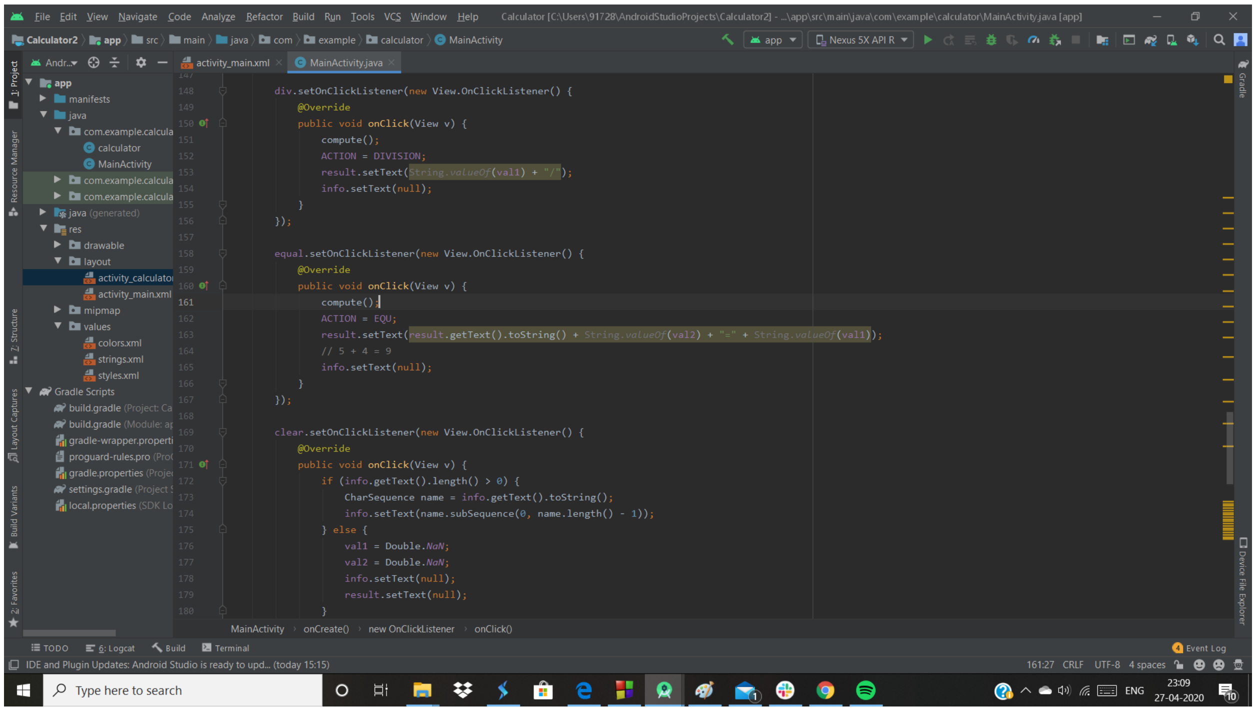
- For the buttons corresponding to numbers 0 to 9:
  When a user clicks on a particular button, this functional code is executed and the info is updated accordingly.

- For the buttons corresponding to the operators(+,-,*,/,=):
  We need to compute according to the chosen operator, update info and result.We will use a compute() function and update the action according to the operator.

Refer to the figure below.

- For the clear button:
  - If our info is not null:
    We need to update info. We will have to delete the last character of the info(or in other words display the string upto (size-1)

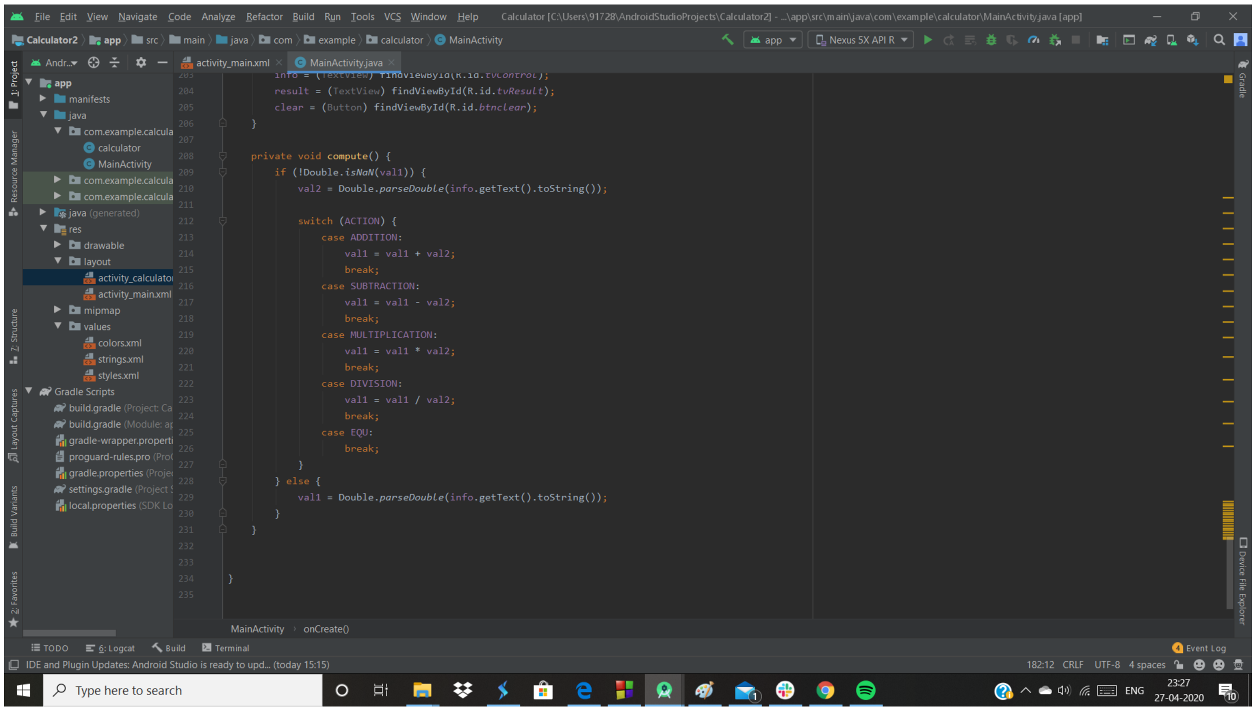
  - If the result has already been computed:
    We need to set text in result and info to be null



```java
        div.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                compute();
                ACTION = DIVISION;
                result.setText(String.valueOf(val1) + "/");
                info.setText(null);
            }
        });

        equal.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                compute();
                ACTION = EQU;
                result.setText(result.getText().toString() + String.valueOf(val2) + "=" + String.valueOf(val1));
                // 5 + 4 = 9
                info.setText(null);
            }
        });

        clear.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (info.getText().length() > 0) {
                    CharSequence name = info.getText().toString();
                    info.setText(name.subSequence(0, name.length() - 1));
                } else {
                    val1 = Double.NaN;
                    val2 = Double.NaN;
                    info.setText(null);
                    result.setText(null);
                }
```

Did we miss something? Ofcourse, we haven't declared the COMPUTE function: For the Compute function, we need to calculate our result according to the action, we will use Switch-Case for that.

Wait, but there is a catch. We will have to check whether the first value is not null. If it is null, we will have to accept val1 otherwise we will accept val2 and proceed to calculation.



```java
        info = (TextView) findViewById(R.id.tvControl);
        result = (TextView) findViewById(R.id.tvResult);
        clear = (Button) findViewById(R.id.btnclear);
    }

    private void compute() {
        if (!Double.isNaN(val1)) {
            val2 = Double.parseDouble(info.getText().toString());

            switch (ACTION) {
                case ADDITION:
                    val1 = val1 + val2;
                    break;
                case SUBTRACTION:
                    val1 = val1 - val2;
                    break;
                case MULTIPLICATION:
                    val1 = val1 * val2;
                    break;
                case DIVISION:
                    val1 = val1 / val2;
                    break;
                case EQU:
                    break;
            }
        } else {
            val1 = Double.parseDouble(info.getText().toString());
        }
    }


}
```

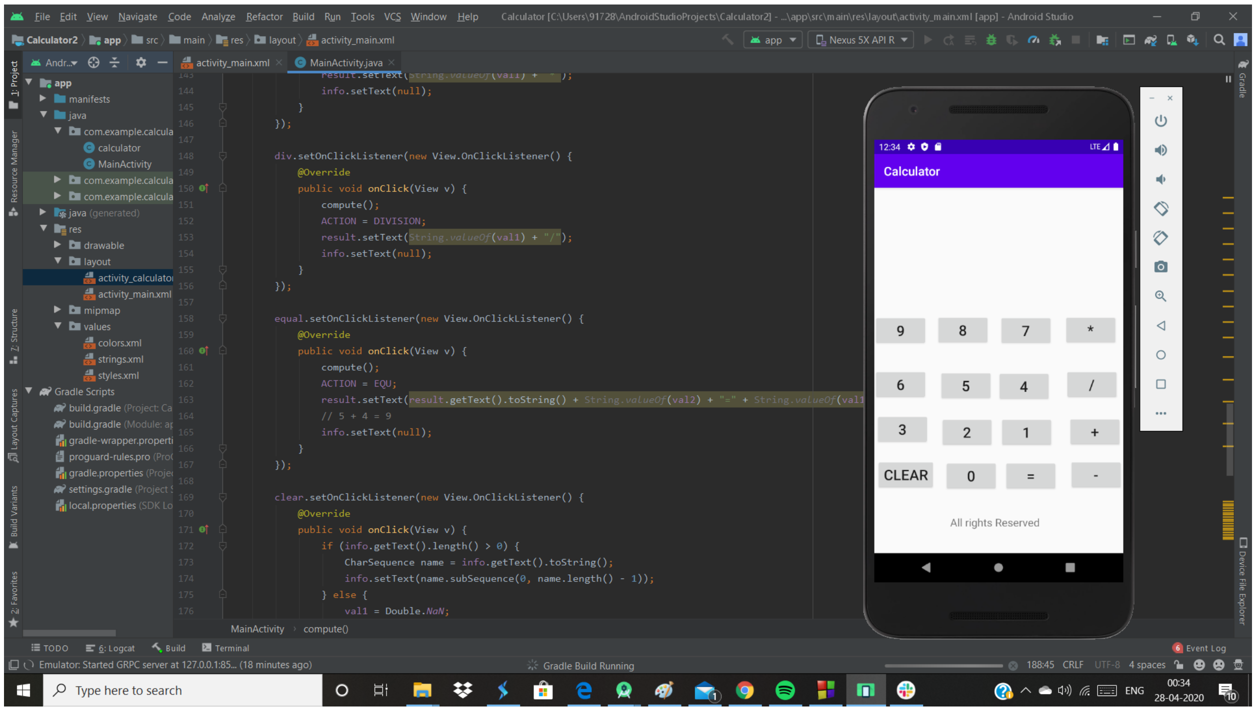
Now, what I have done is updated val1, you might be wondering why? Right?

Suppose you want to calculate : 12*9-79

➔ If you don't update val1, and rather return the result , you won't be able to perform a multi-step calculation, or rather your result will be 12*9 and - 79 will have no significance.

But updating val1 will result in val1 becoming 108 after the first step, and then the next step continues.

Well, our code is ready. You may now run your app on your virtual device.



```java
                info.setText(null);
            }
        });

        div.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                compute();
                ACTION = DIVISION;
                result.setText(String.valueOf(val1) + "/");
                info.setText(null);
            }
        });

        equal.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                compute();
                ACTION = EQU;
                result.setText(result.getText().toString() + String.valueOf(val2) + "=" + String.valueOf(val1
                // 5 + 4 = 9
                info.setText(null);
            }
        });

        clear.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (info.getText().length() > 0) {
                    CharSequence name = info.getText().toString();
                    info.setText(name.subSequence(0, name.length() - 1));
                } else {
                    val1 = Double.NaN;
```

c) Building the APK file:

From the Menu Bar:
Build--> Build Bundle(s)/APK(s)--> Build APK(s)

Your APK file is built and stored in your local disk:

In Android Studio Projects→ Under Your Project Name→ App → Build→ Output→ APK→ Debug→ app-debug.apk

This is your required APK file. You can rename it, install it and send it to your friends.

Congratulations on Building Your First App.

❖ Some Ideas:

Try your hands on some day-to-day applications like:

- ➢ Calendar
- ➢ Random Number Generator.

- Tic-Tac Toe
- A Quiz App