MOBILE COMPUTING-PROGRESS FILE

Submitted to:

Sir Haq Nawaz

Submitted by:

Amber Shafique (Bsef18a019)

PUCIT, University of the Punjab, Lahore.

LECTURE#1

❖ VCS (Version Control System):

A **version control system** is a kind of software that helps the developer team to efficiently communicate and manage track of all the changes that have been made to the source code along with the information like who made and what change has been made.

- Version control software keeps track of every modification to the code in a special kind of database.
- Using a VCS also generally means that if you screw things up or lose files, you can easily recover

Benefits:-

1. **Code Synchronizing:** Same code available to all members on different repositories connected to a central repository. All the changes in the files are tracked under the central repository. The central repository includes all the information of versioned files, and list of users that check out files from that central place using **VCS.**

2. **Code Testing :VCS** helps to test changes without losing the original version of the application.

3. **Revert: VCS** allows us to revert back to previous versions of the file because it separately maintains each version of file.

❖ Git :

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

Step-0

1.1 GitHub Account:

Sign up to ***github.com*** and create a new account.

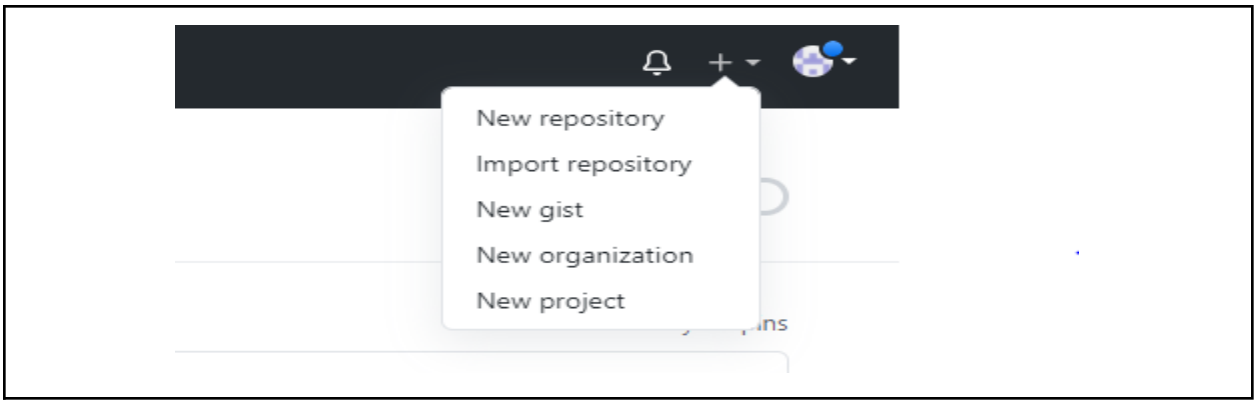
1.2 Git CMD:

Install git cmd and configure it on your PC.

Step-1

1.1 Create a new Repository:

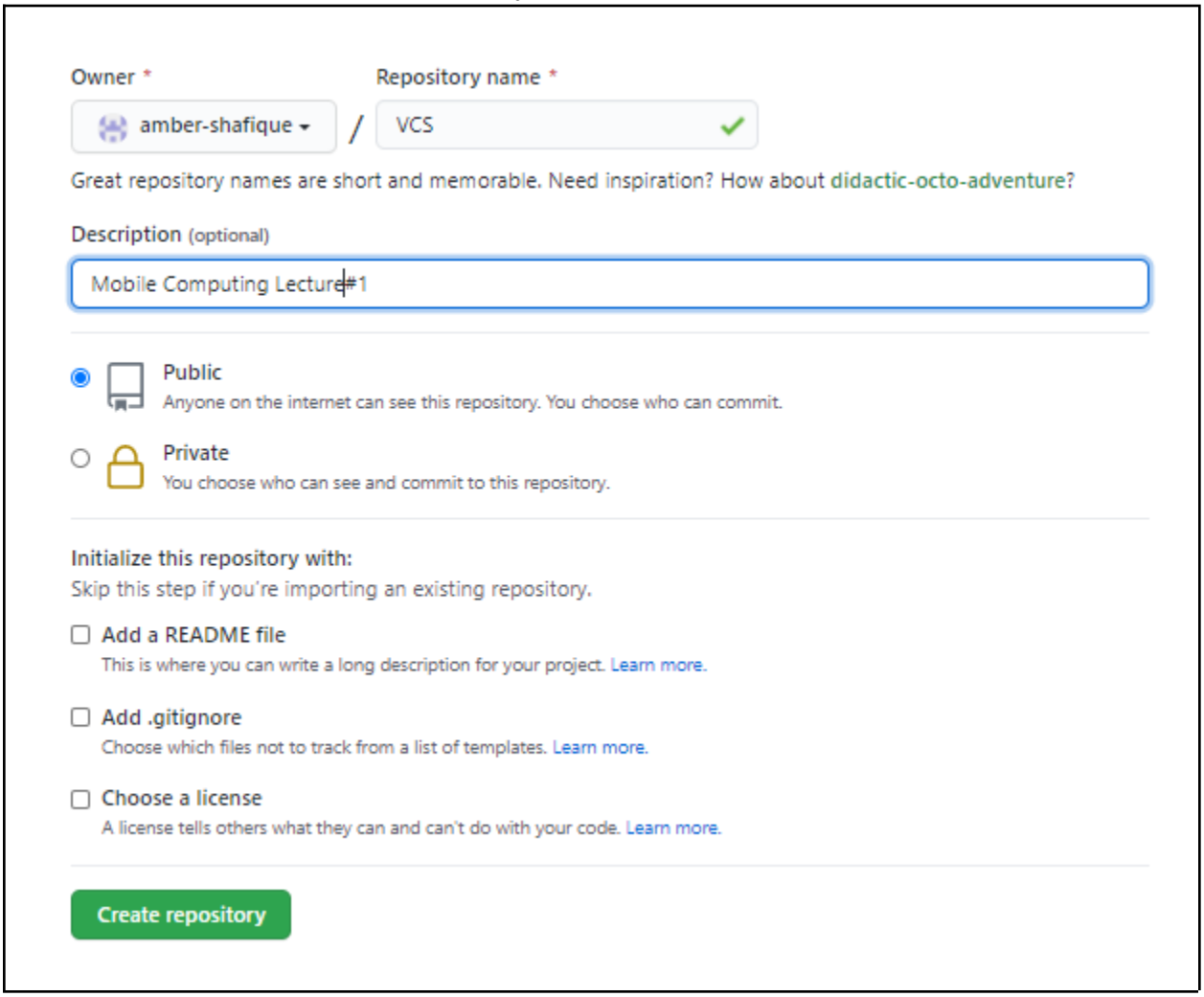
1.1.1 Click on add new repository button on your GitHub account.



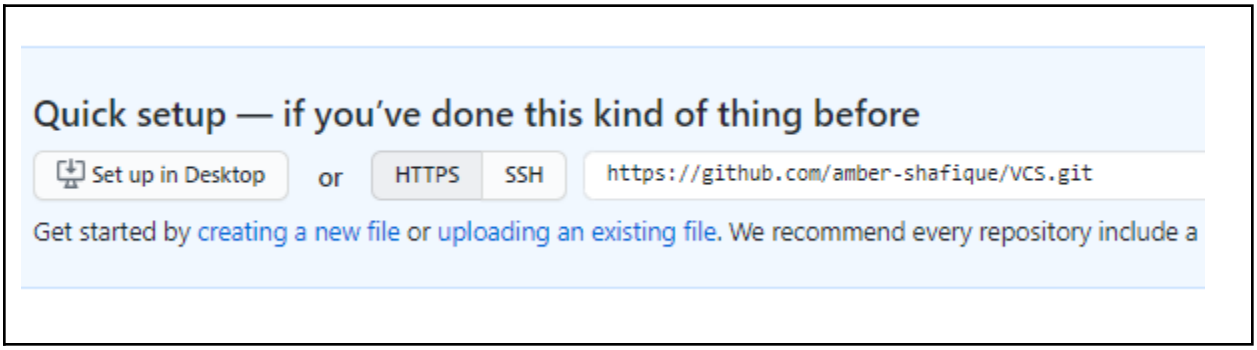
1.1.2 Then provide details of your repository => Name, Description,Type.

Public Repository: It is visible to every user on your GitHub.
Private Repository: It is only available to the repository owner.

1.1.3 Click on the create repository button.



1.2 Repository URL : On creating a new repository GitHub will provide its url that is used to uniquely identify our repository on GitHub.



LECTURE#2

❖ Git Clone:

- It is primarily used to point to an existing repository and make a clone or copy of that repository in a new directory, at another location.
- Makes connection between **local repository(**on PC) and **central repository** (on GitHub).

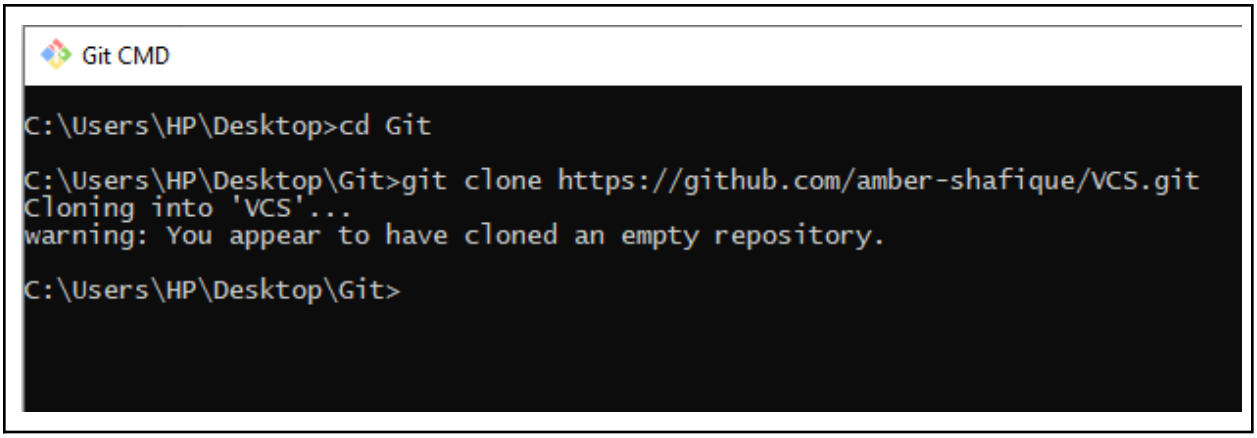
Step-1

Open **git CMD** and go to that folder path (using **cd**) where you want to clone the repository.

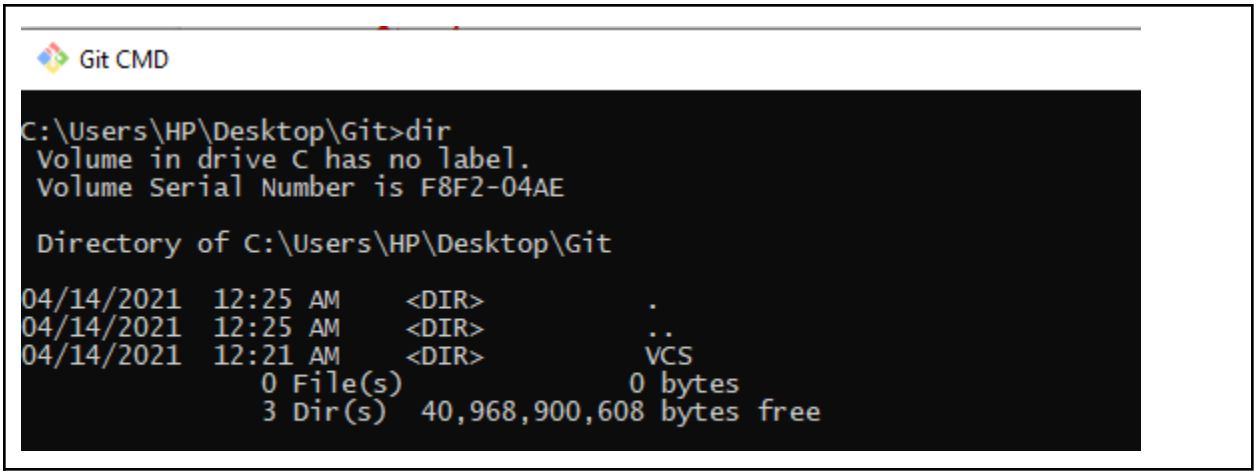
Step-2

Run the following command and give the **git url** of the repository you want to clone locally.

```
git clone URL
```



Repository cloned Locally (see using **dir**)



❖ Add File:

Add locally created file to the central(online) repository.

Step-1

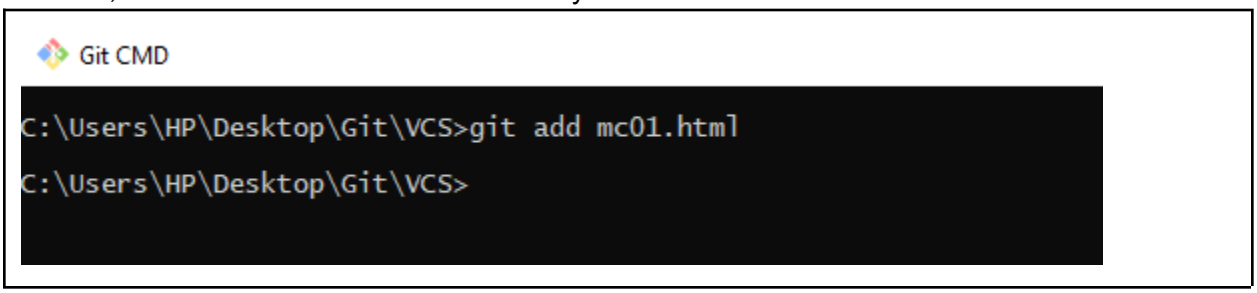
Creating file in local repository.

Step-2

Run the following command and give the **File Name** you want to add on git repository.

```
git add FileName
```

No error, so file has been added successfully.



❖ Add all files :

- **git add --all**
- **git add .**

❖ Add multiple files at same time:

```
git add FileName  FileName  FileName ...
```

❖ Git Commit

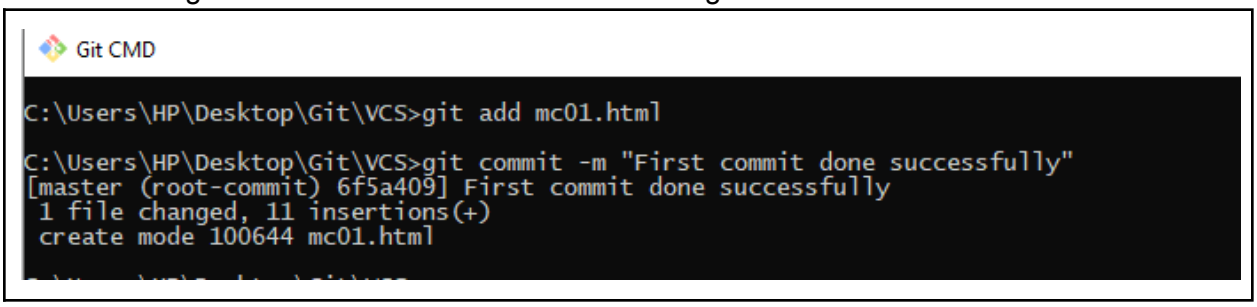
The "commit" command is used to save your changes and track messages.

Step-3

Run the following command and give the **Message** you want to get displayed with added file.

```
git commit -m “message”
```

Message=> to inform the users about the change.



❖ Add and commit simultaneously

```
git commit -am “message”
```

❖ Git Push:

To push the changes made in file locally to server (online repository).

Step-4

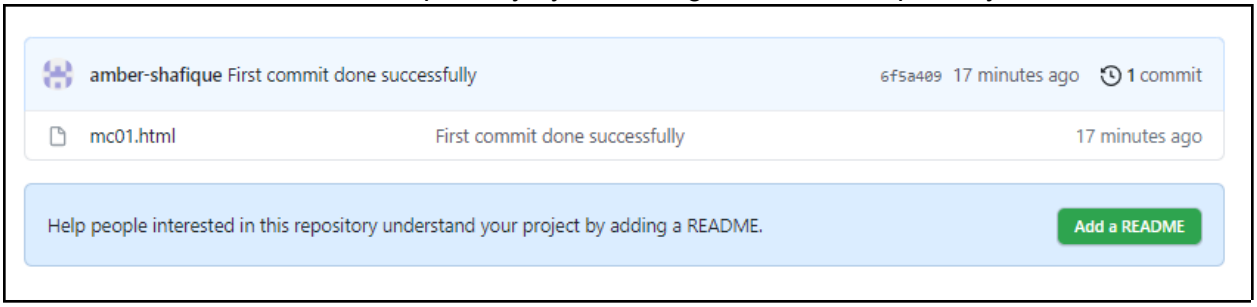
Run the following command to complete the synchronization.

```
git push
```

Now we are synchronized with data available on local disk and online repository.

```
Git CMD

C:\Users\HP\Desktop\Git\VCS>git push
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 345 bytes | 69.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/amber-shafique/VCS.git
 * [new branch]      master -> master
```

Check the status of the online repository by refreshing the GitHub Repository URL.



Content in online and local repository is synchronized.

11 lines (10 sloc) | 146 Bytes

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4          <title>Mobile Computing</title>
5  </head>
6  <body>
7
8      <h1>Lectur-2</h1>
9      <p>Testing for GitHub</p>
10 </body>
11 </html>
```

- **Git Status:**

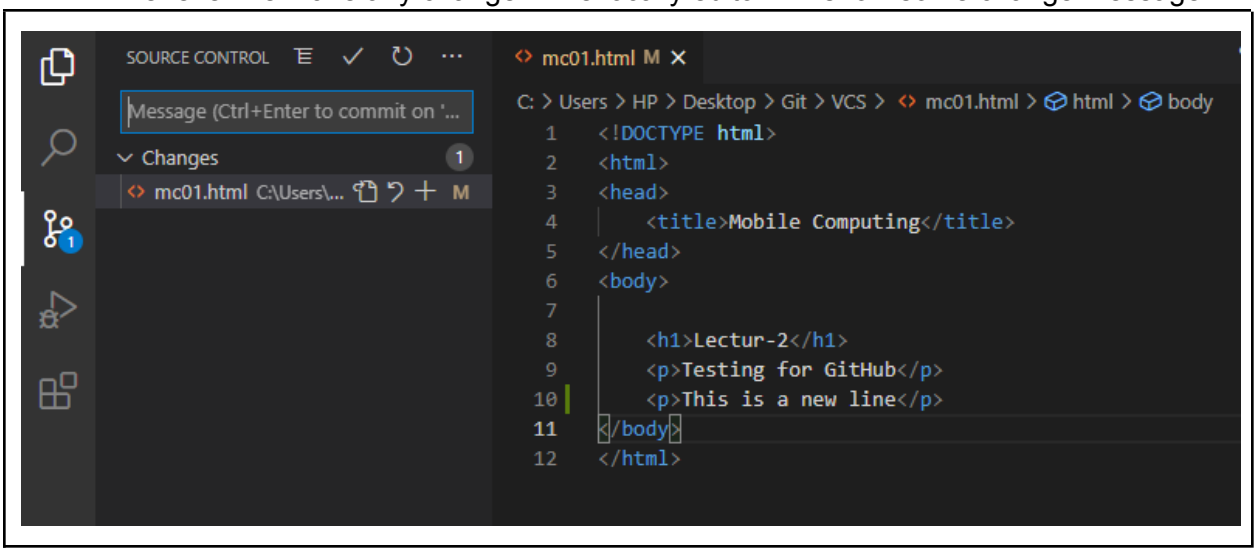
  Gives the status of repository.

```
git status
```

```
Git CMD

C:\Users\HP\Desktop\Git\VCS>git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean

C:\Users\HP\Desktop\Git\VCS>
```

Making change in file locally using IDE:

Whenever we make any change in file locally editor will show some change message



❖ Adding New Commit (To save locally made changes online) :

Run these previously discussed commands on git CMD in sequence.

Step-1

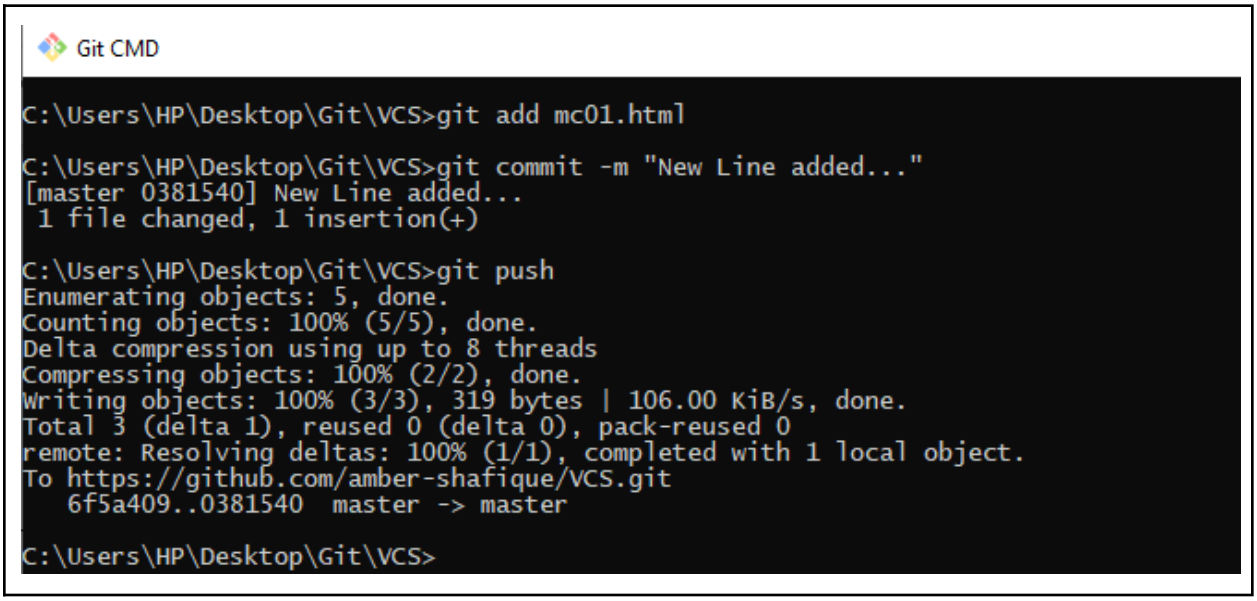
```
git add FileName
```

Step-2

```
git commit -m “message”
```

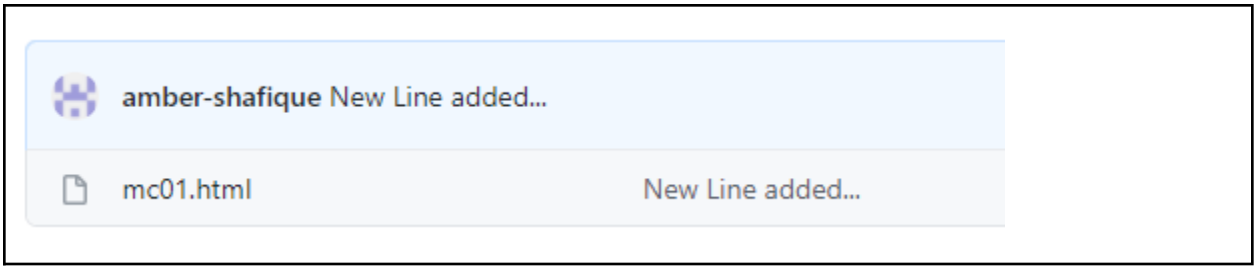
Step-3

```
git push
```



Track of Changes:

To preview change online refresh the repository url and see the changes have been made.



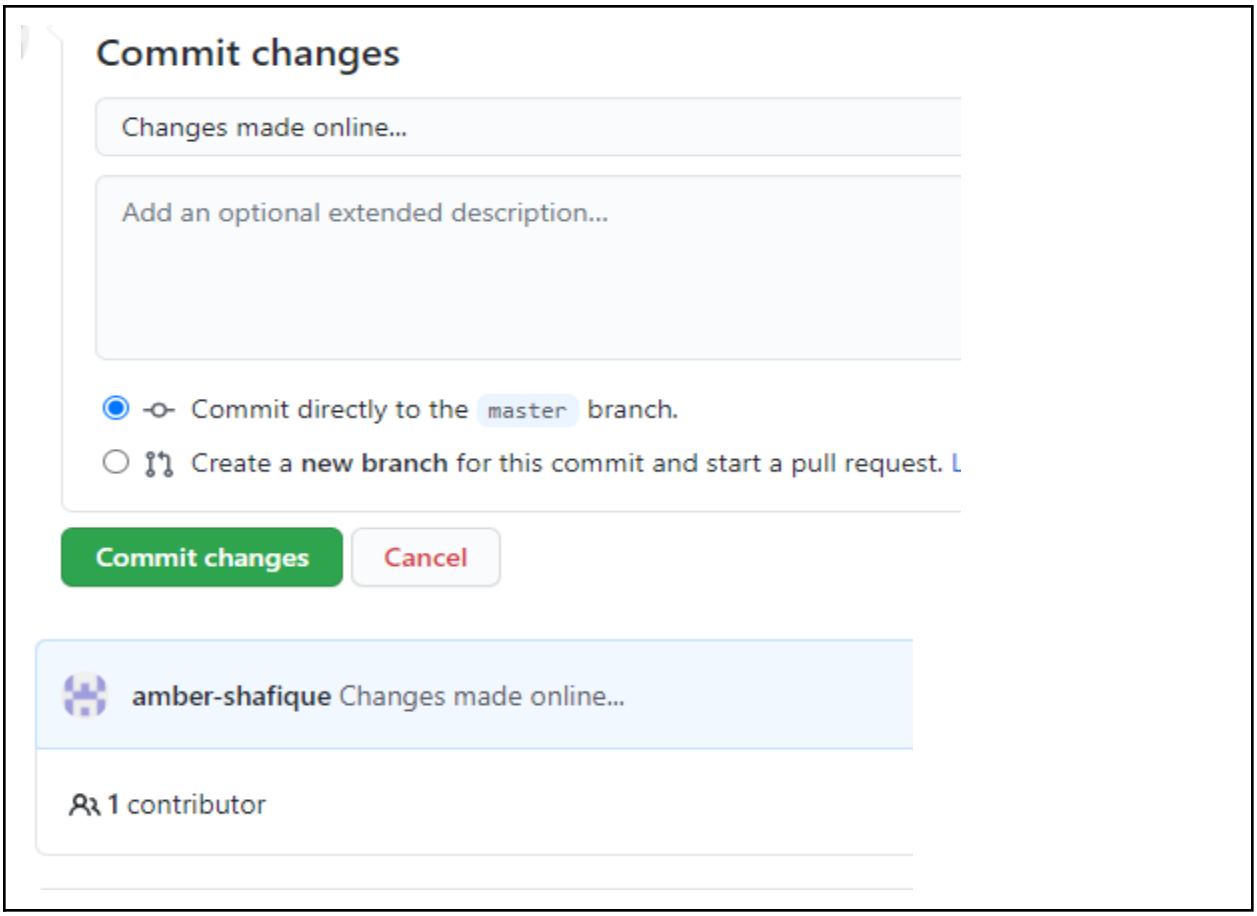
❖ Editing a file online in github repository:

Step-1

Edit the file online using the edit button on file.

Step-2

Save commit changes.



❖ Git Pull:

To get Changes from GitHub to local repository.

Step-1

After making changes online, run the following command on git CMD.

```
git push
```

Changes have been made to local repository.

Git CMD

```
C:\Users\HP\Desktop\Git\VCS>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 789 bytes | 39.00 KiB/s, done.
From https://github.com/amber-shafique/VCS
   0381540..80df6ba  master     -> origin/master
Updating 0381540..80df6ba
Fast-forward
 mc01.html | 5 ++++-
 1 file changed, 4 insertions(+), 1 deletion(-)

C:\Users\HP\Desktop\Git\VCS>
```

❖ Version Hierarchy:

File showing with latest commit.

- **Red Highlighted lines:** The lines that have been deleted.
- **Green Highlighted lines:** The lines that have been changed.

```
5   5     </head>
6   6     <body>
7   7
8       -     <h1>Lectur-2</h1>
        8     <h1>Lectur-2 changed by git</h1>
9   9         <p>Testing for GitHub</p>
10  10
11  11        <p>This is a new line</p>
```

Parents=> previous versions of file.

1 parent 80df6ba commit

LECTURE#3

❖ Merge Conflicts:

A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

Step-1

Changes in lin-8 online.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4          <title>Mobile Computing</title>
5  </head>
6  <body>
7
8      <h1>Lectur-2 changed by git</h1>
```

Step-2

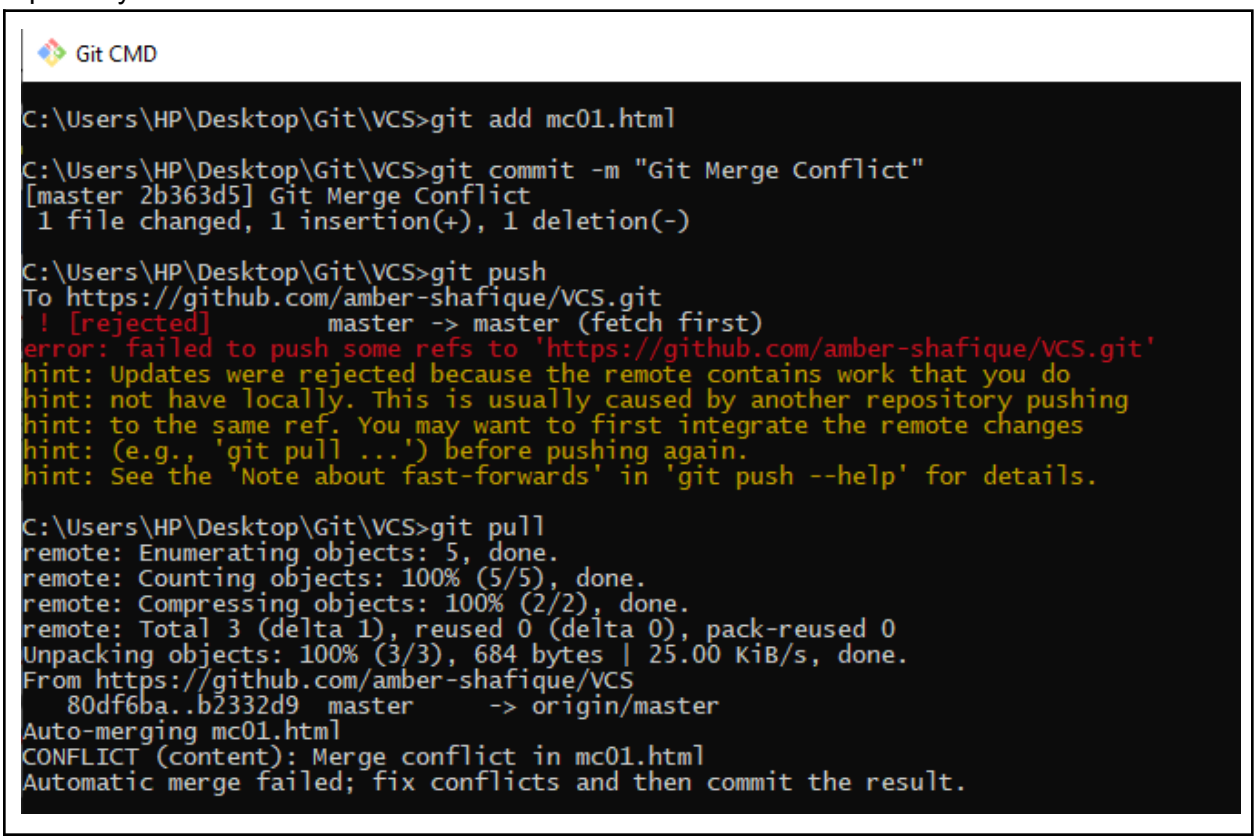
Changes in same line-8 locally.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Mobile Computing</title>
5  </head>
6  <body>
7
8      <h1>Lectur-2 chaged locally</h1>
```

Step-3

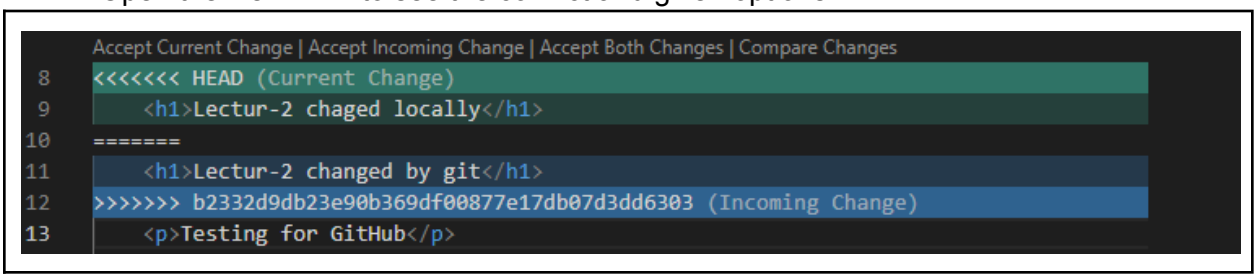
Run commands step by step to save file changes in git CMD.

Merge Conflict has occurred because we are trying to change the same line locally and in online repository.



Step-4

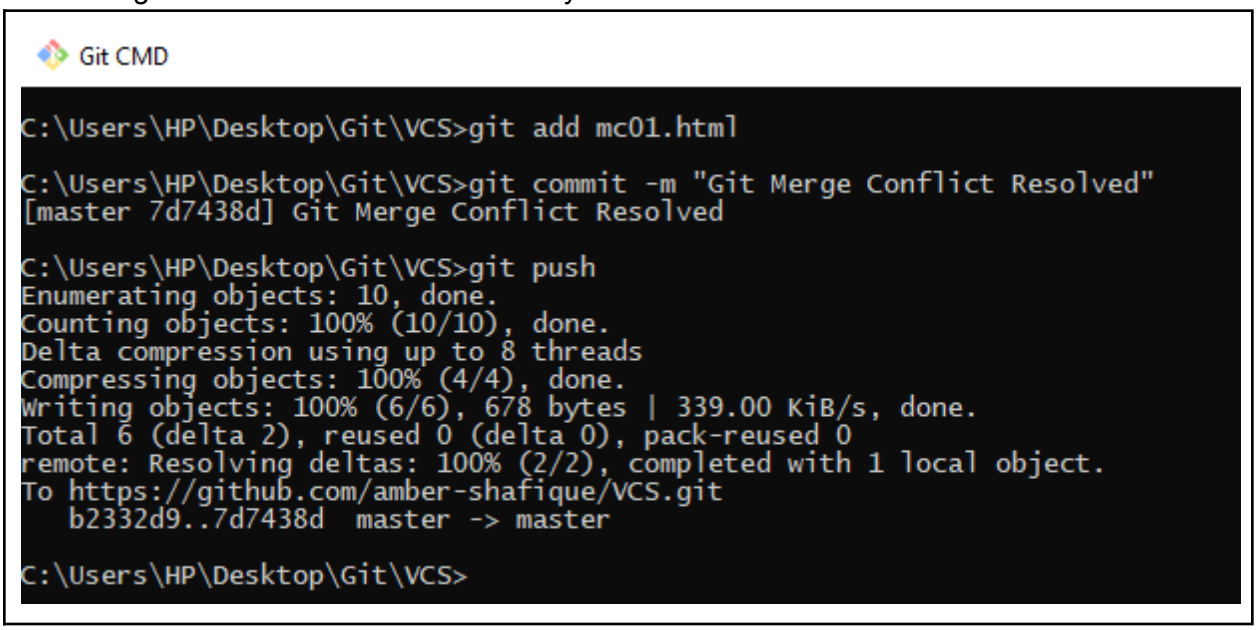
Open the file in IDE to see the conflict and given options.



Step-5

Accept Change and resolve the merge conflict and **again** Run commands step by step to save file changes in git CMD.

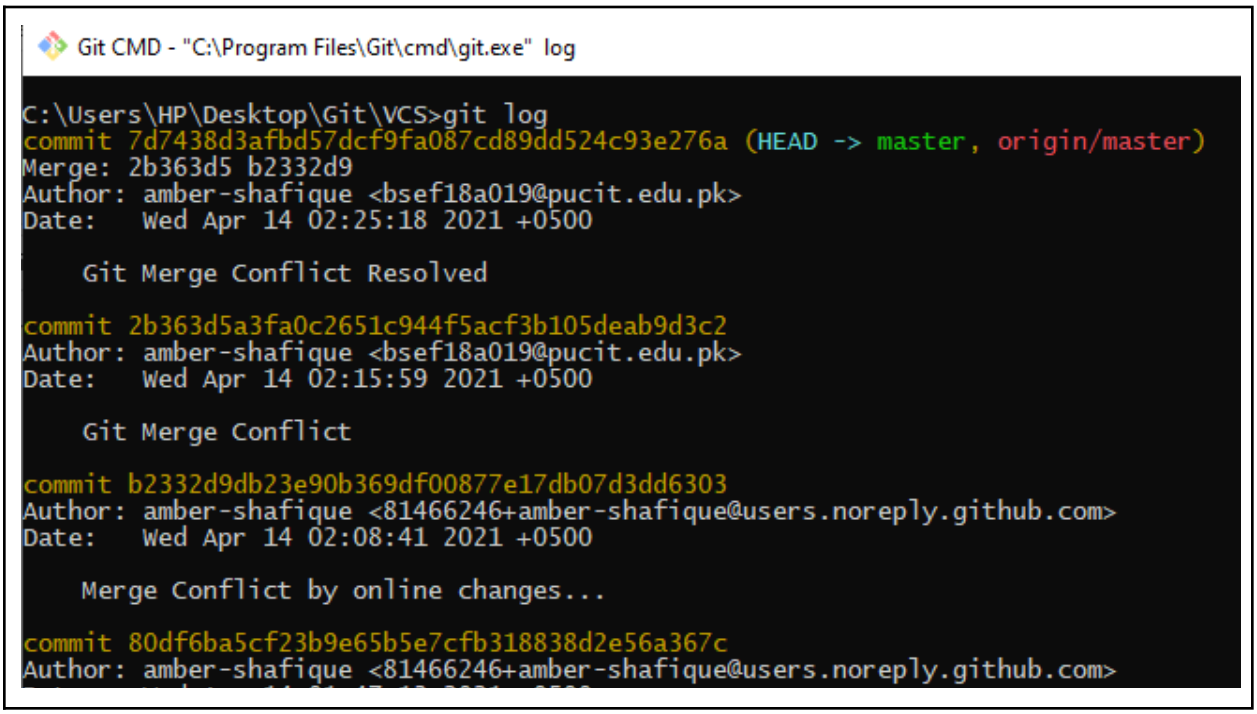
Now changes have been saved successfully as conflict is resolved.



❖ Git Logs:

To see the details of all commits.

```
git log
```



❖ Remove a File:

Files already present

```
C:\Users\HP\Desktop\Git\MC-1>dir
 Volume in drive C has no label.
 Volume Serial Number is F8F2-04AE

 Directory of C:\Users\HP\Desktop\Git\MC-1

04/14/2021  03:44 PM    <DIR>          .
04/14/2021  03:44 PM    <DIR>          ..
04/14/2021  02:42 AM                 6 file1.txt
04/14/2021  03:44 PM                 6 file2.txt
04/14/2021  03:06 AM                 7 file3.txt
               3 File(s)             19 bytes
               2 Dir(s)  41,993,297,920 bytes free
```

Step-1

To remove file type the following command on git CMD, and give the name of file you want to delete.

git rm FileName

File deleted locally

Git CMD

```
C:\Users\HP\Desktop\Git\MC-1>git rm file1.txt
rm 'file1.txt'

C:\Users\HP\Desktop\Git\MC-1>
```

Now to update delete changes to git:

Step-2

Git add .

Step-3

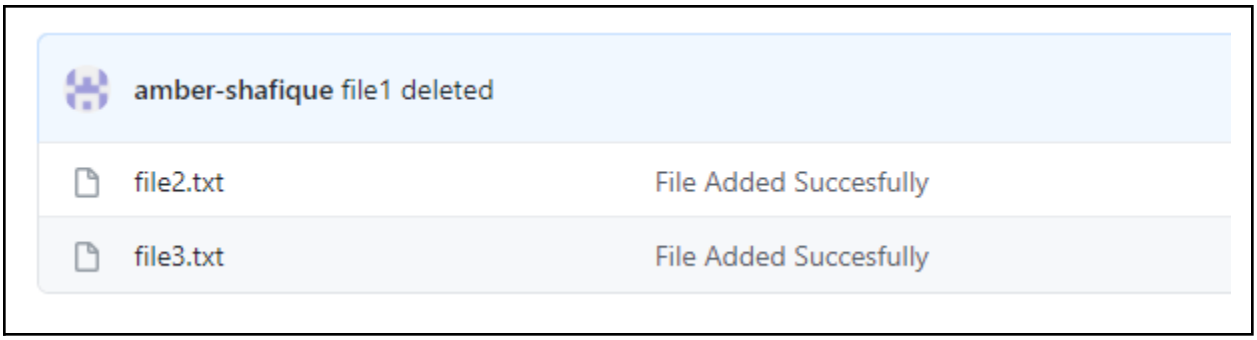
git commit -m “message”

Step-4

git push

```
C:\Users\HP\Desktop\Git\MC-1>git add .

C:\Users\HP\Desktop\Git\MC-1>git commit -m "file1 deleted"
[master 37a39ee] file1 deleted
 1 file changed, 1 deletion(-)
 delete mode 100644 file1.txt

C:\Users\HP\Desktop\Git\MC-1>git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 264 bytes | 88.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/amber-shafique/MC-1.git
   f837866..37a39ee  master -> master

C:\Users\HP\Desktop\Git\MC-1>
```

Preview delete changes on GitHub



❖ Branching:

Branching is the practice of creating copies of programs or objects in development to work in parallel versions, retaining the original and working on the branch or making different changes to each.

Step-1

Git Branch:

Run the following command on cmd to check available branches.

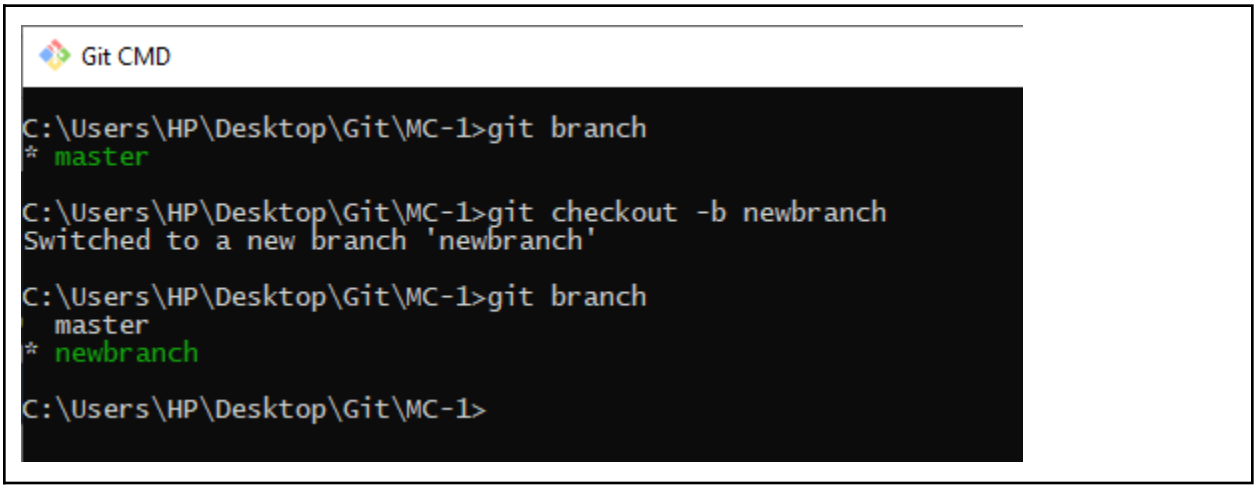
```
git branch
```

Step-2

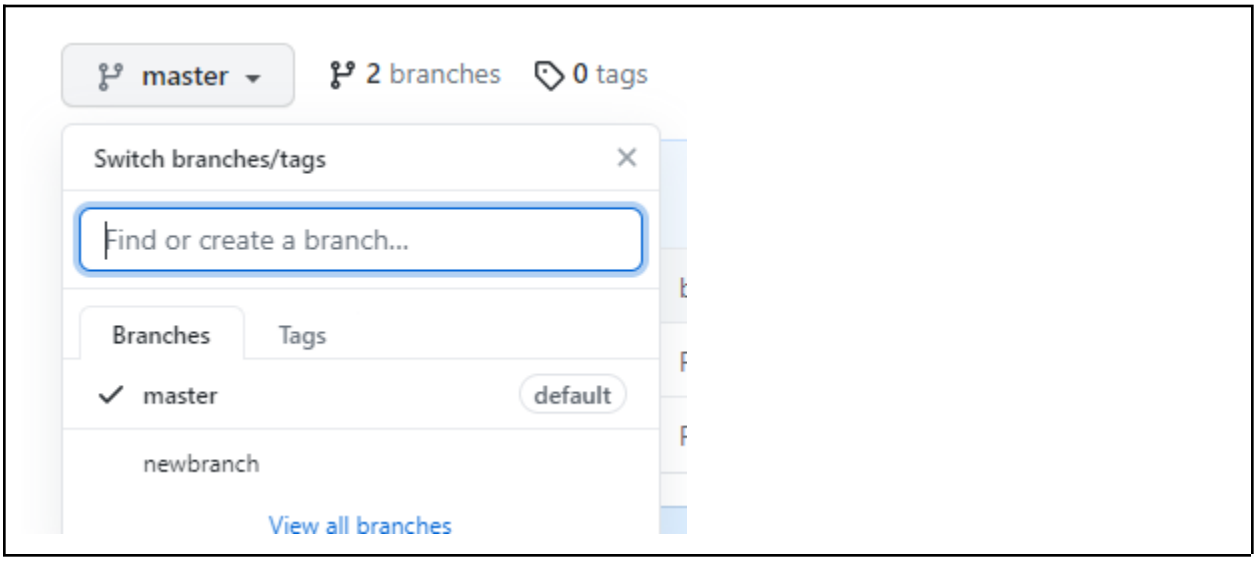
Git Checkout:

To create a new branch, type the following command on git CMD and give the name of branch.

```
git checkout -b newbranch
```



Branch Added on GitHub

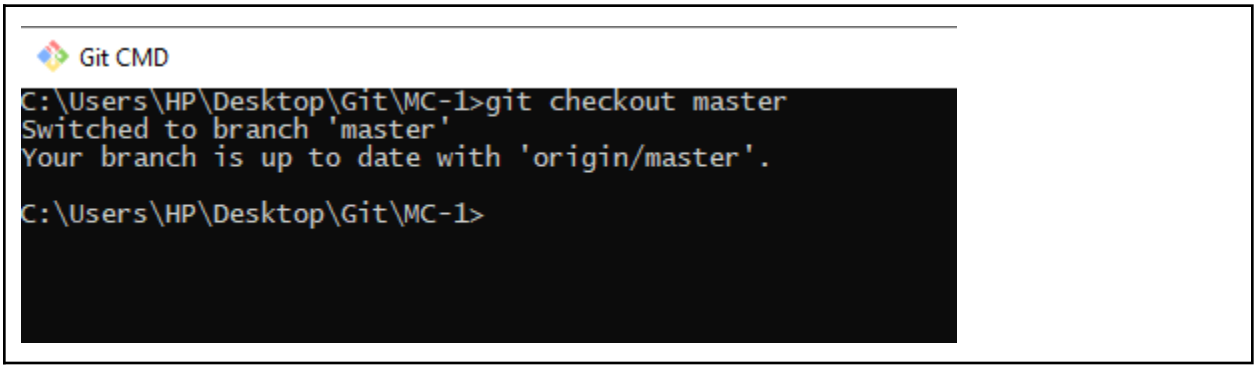


Step-3

Git Checkout Master:

To navigate from one branch to another. (master=>branch name)

git checkout master



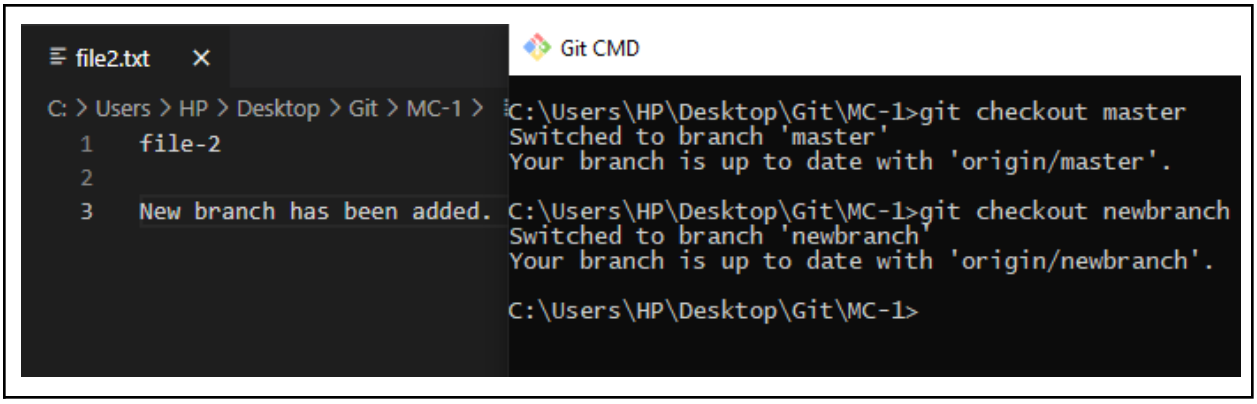
To Check Branching:

Step-1 Add changes to a file on master branch.

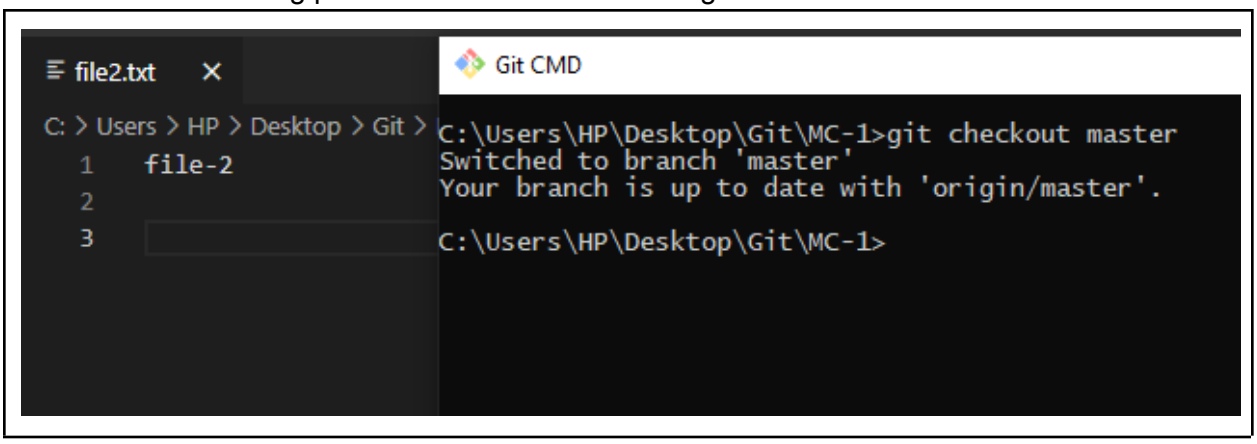
Step-2 Move to new branch and add file on the new branch.

Step-3 Checkout file on both branches and see the changes accordingly on IDE.

New branch showing changes...



Master branch showing previous version without changes...



❖ Merging:

Merging is Git's way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.

Merge Branch:

To merge the changes in branches.

Git merge branchname

```
C:\Users\HP\Desktop\Git\MC-1>git merge newbranch
Removing file3.txt
Auto-merging file2.txt
CONFLICT (content): Merge conflict in file2.txt
CONFLICT (add/add): Merge conflict in BranchingPractice.html
Auto-merging BranchingPractice.html
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\HP\Desktop\Git\MC-1>
```

❖ Delete Branch:

To delete a branch.NewBranch=>you want to delete.

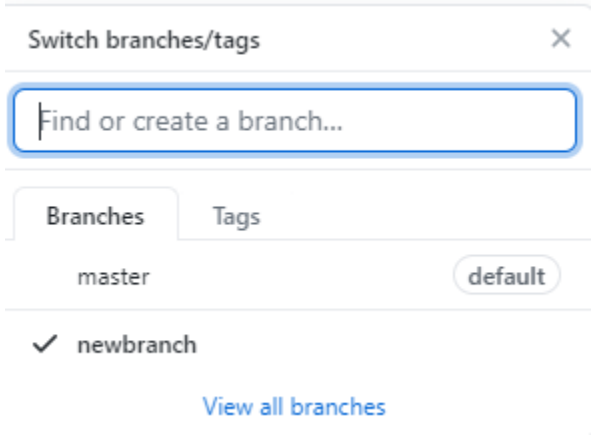
Git branch -D NewBranch

Checking out branches before and after deletion...

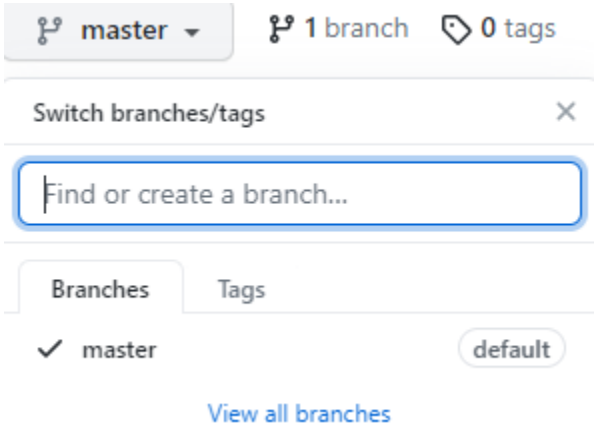
Git CMD

```
C:\Users\HP\Desktop\Git\MC-1>git branch
* master
  newbranch

C:\Users\HP\Desktop\Git\MC-1>git branch -D newbranch
Deleted branch newbranch (was ba2149a).

C:\Users\HP\Desktop\Git\MC-1>git branch
* master

C:\Users\HP\Desktop\Git\MC-1>
```

Before Deletion



After Deletion



LECTURE#4

❖ Android Studio:

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. A unified environment where you can develop for all Android devices. Apply Changes to push code and resource changes to your running app without restarting your app.

Step-1

Installation:

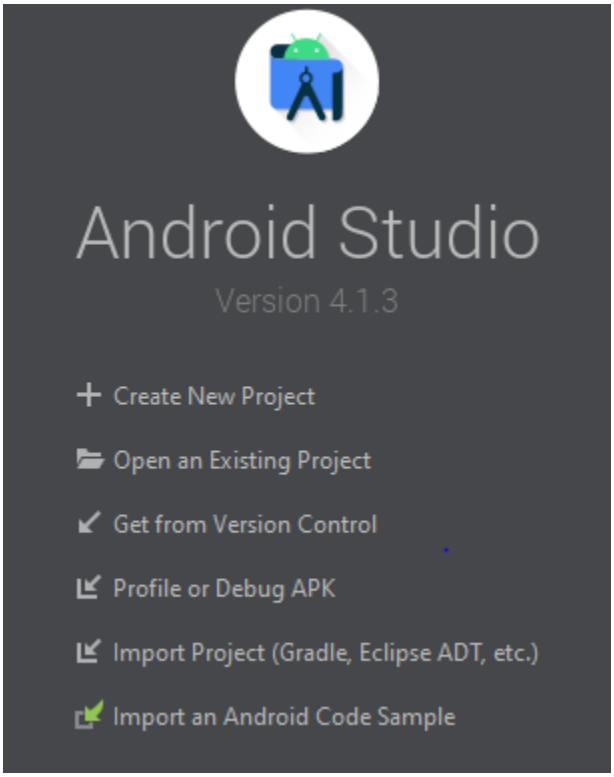
Download any latest version of **Android Studio** and configure it on your PC.

(https://developer.android.com/studio)

Step-2

Create a New Project:

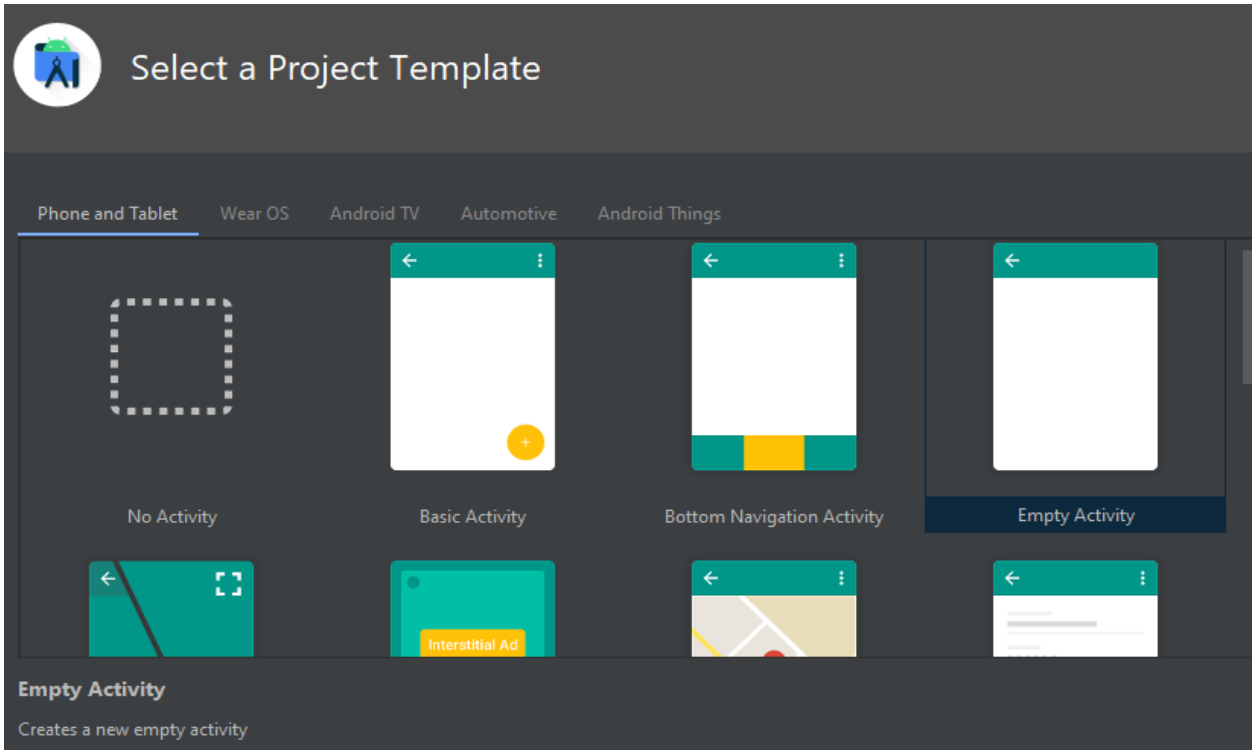
Open Android Studio and select the option to create a new project.



Step-3

Template Selection:

From templates choose Empty Activity.



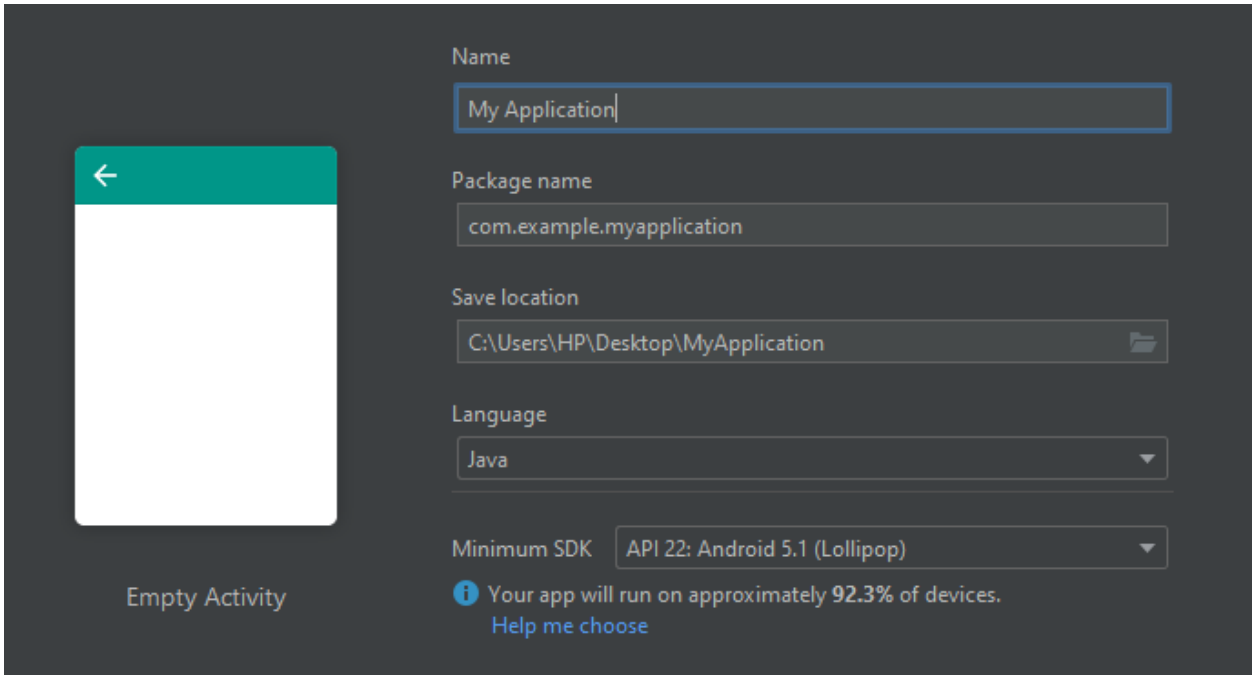
Step-4

Project Name:

Select a name and location for project for project.

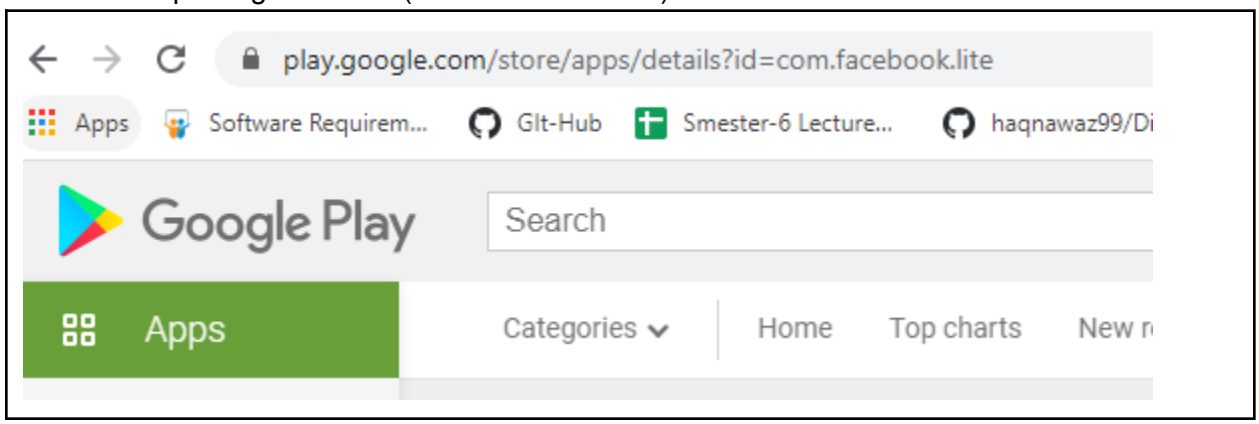
Package Name:

It is used to uniquely identify the APK file of our application and it should must be unique to publish the App on play store.

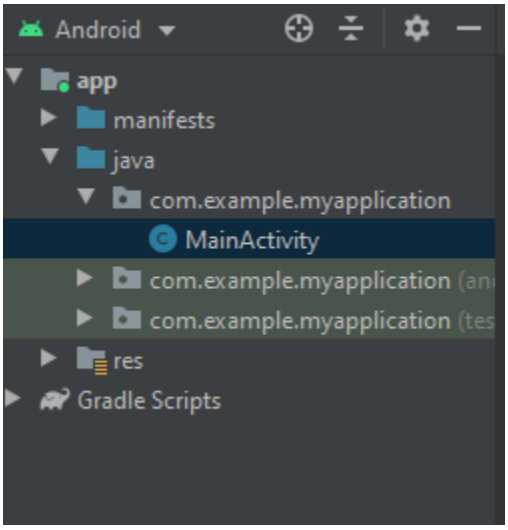
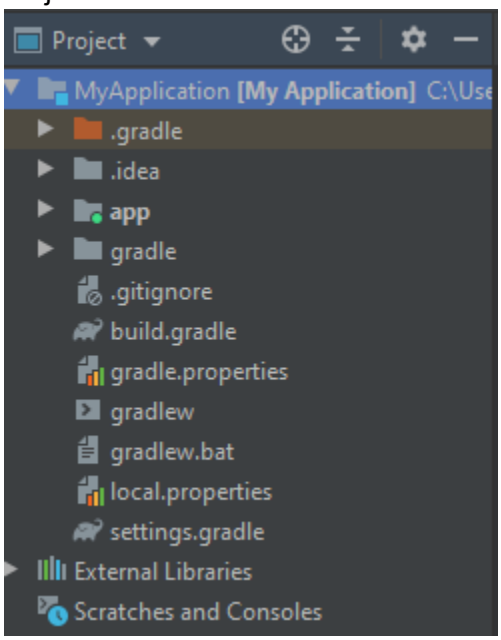


Package Name Example:

Here package name is (**com.facebook.lite**)

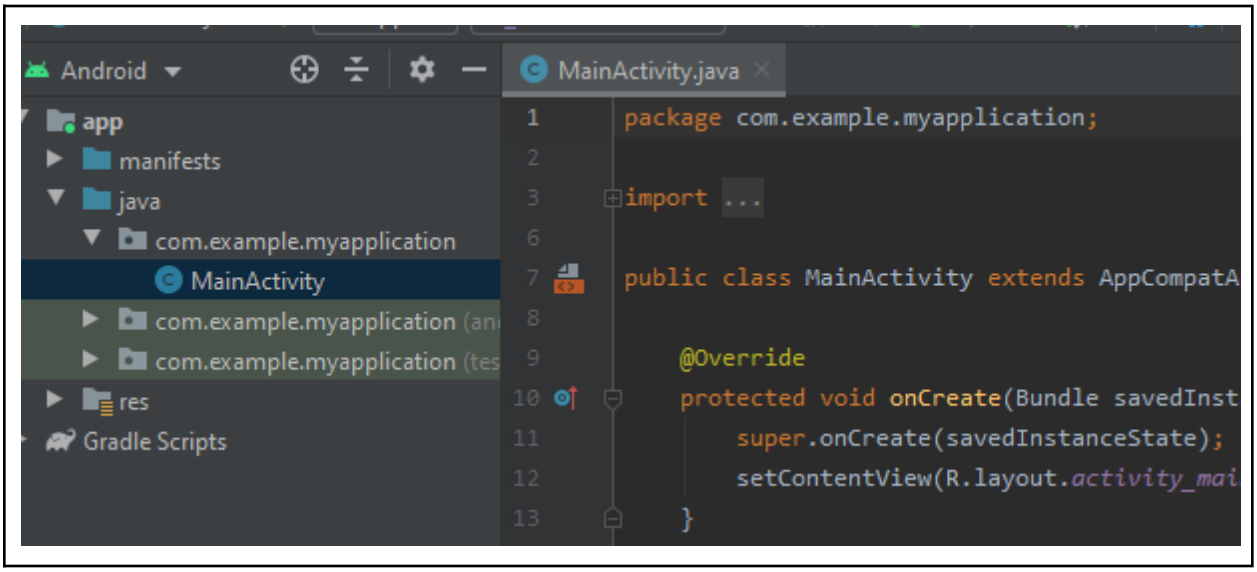


❖ Android Studio File Structure:

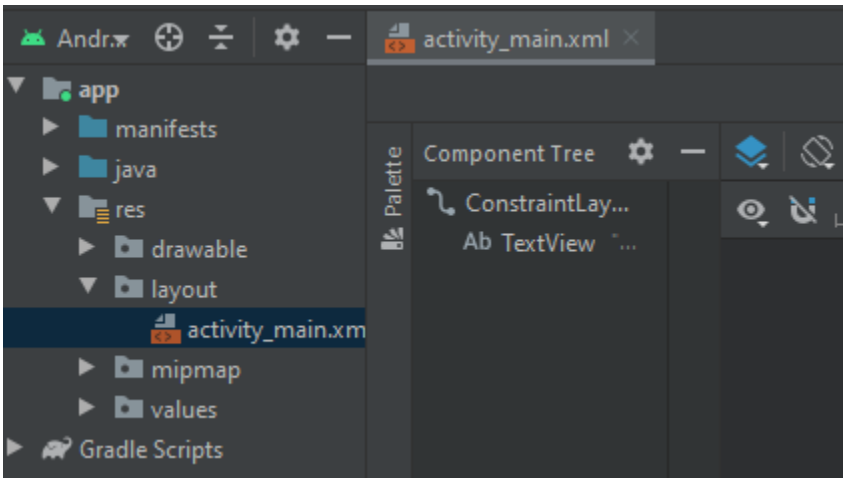
Android view (used by default)	Project view
	

Files for coding:

- **MainActivity.java** file for coding.



Activity_main.xml file for Graphical user interface(**GUI**).



Step-4

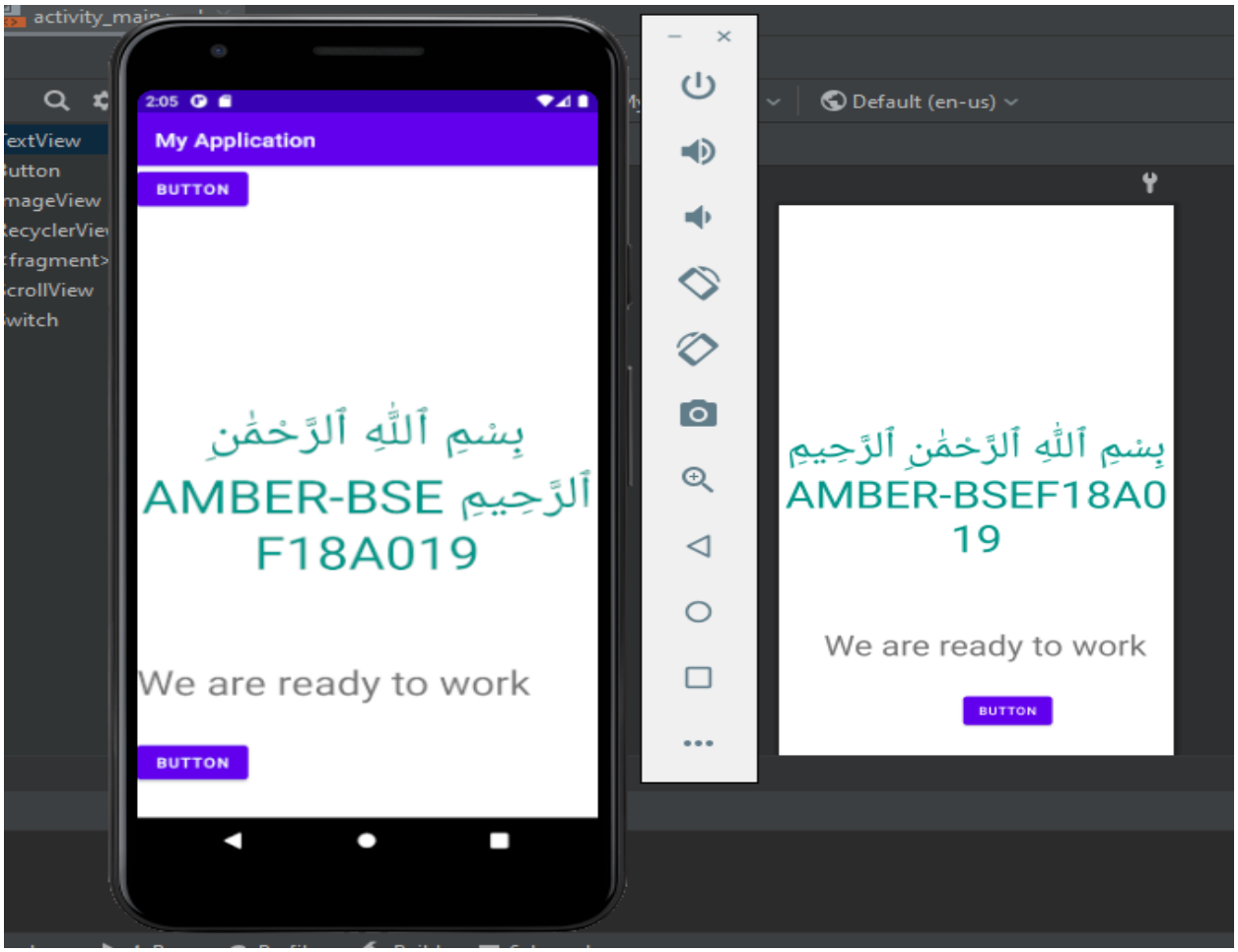
Select Virtual Device:

Select **Virtual Devices** from AVD(Android Virtual Device) Manager to view output.

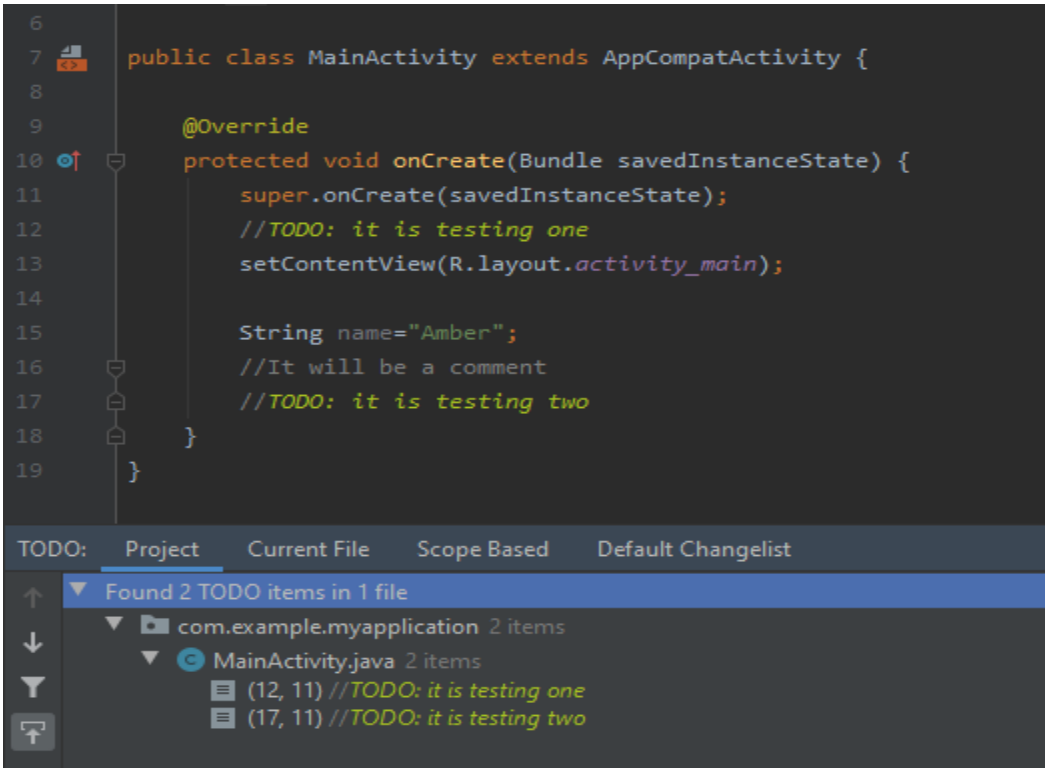
Step-4

To view output:

Run the **Emulator** choosing any virtual device and see the output.

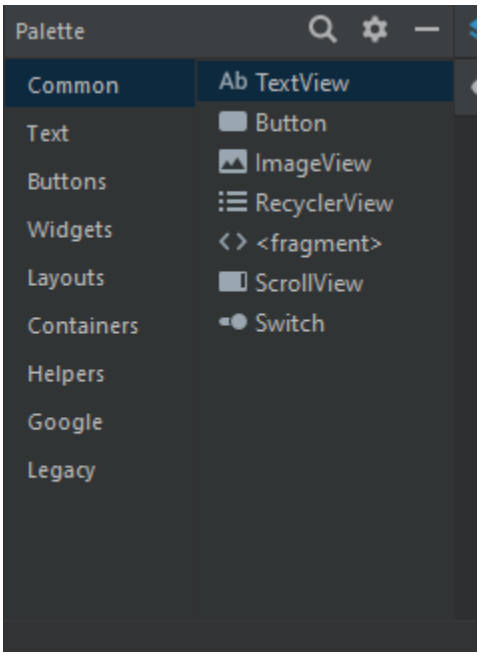

activity_main
Default (en-us)
2:05
My Application
BUTTON
بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ
AMBER-BSE ٱلرَّحِيمِ
F18A019
We are ready to work
BUTTON
بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ
AMBER-BSEF18A0
19
We are ready to work
BUTTON

To Do list:



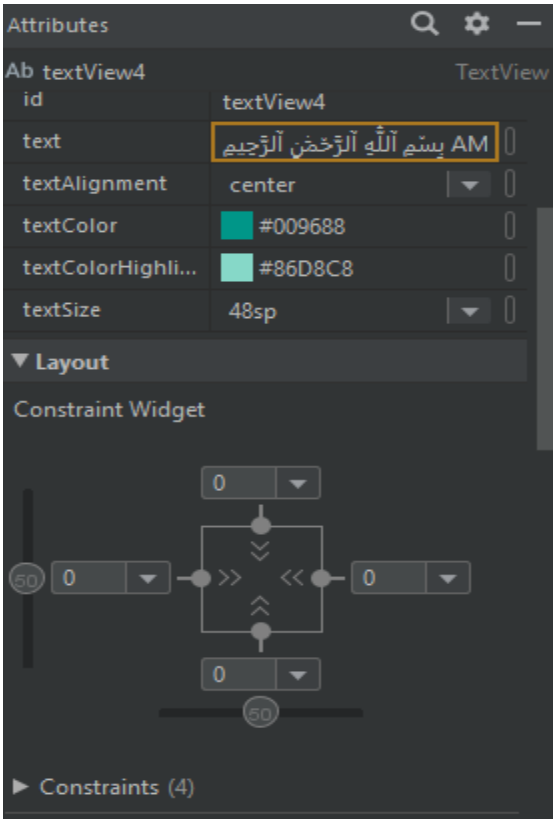
Design Palette:

We can add text,Button,image and other controls by using drag and drop from this palette.



Attributes:

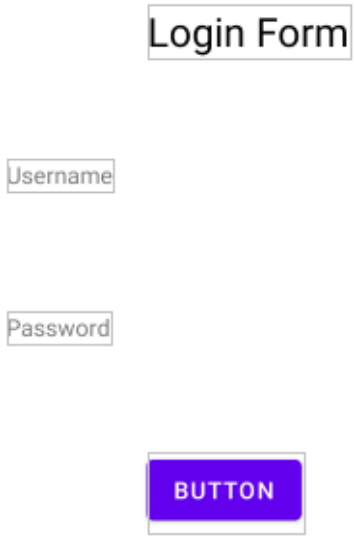
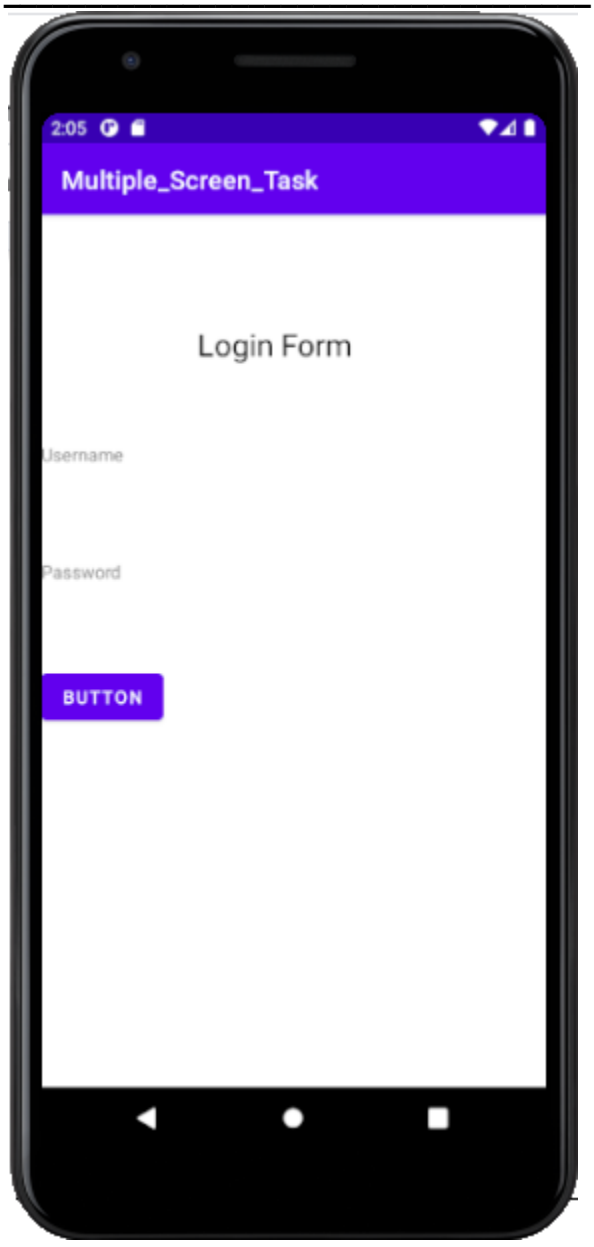
To set the values of elements added in xml.



❖ Multiple Screen Task:

Screen-1

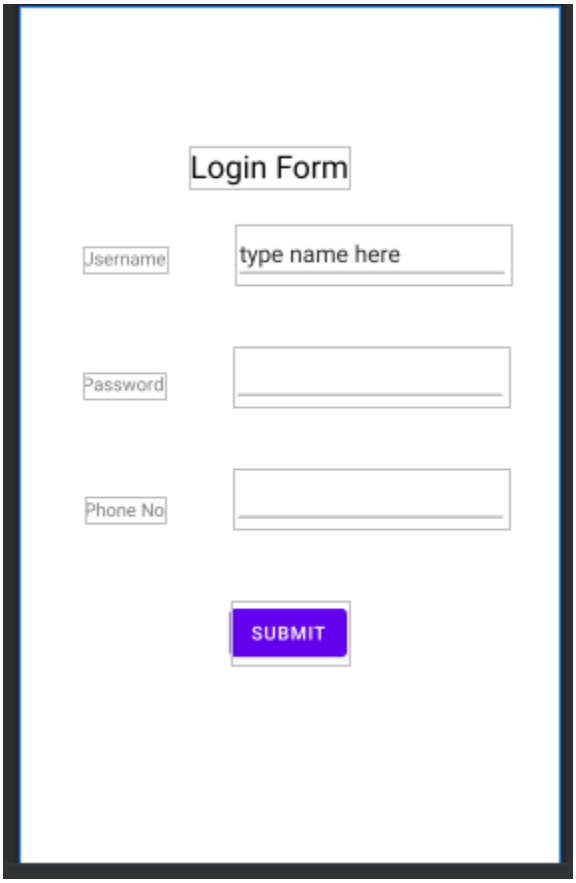
Step-1 Add elements of text box and buttons on the screen from palette by drag and drop.

XML View	Emulator View
	

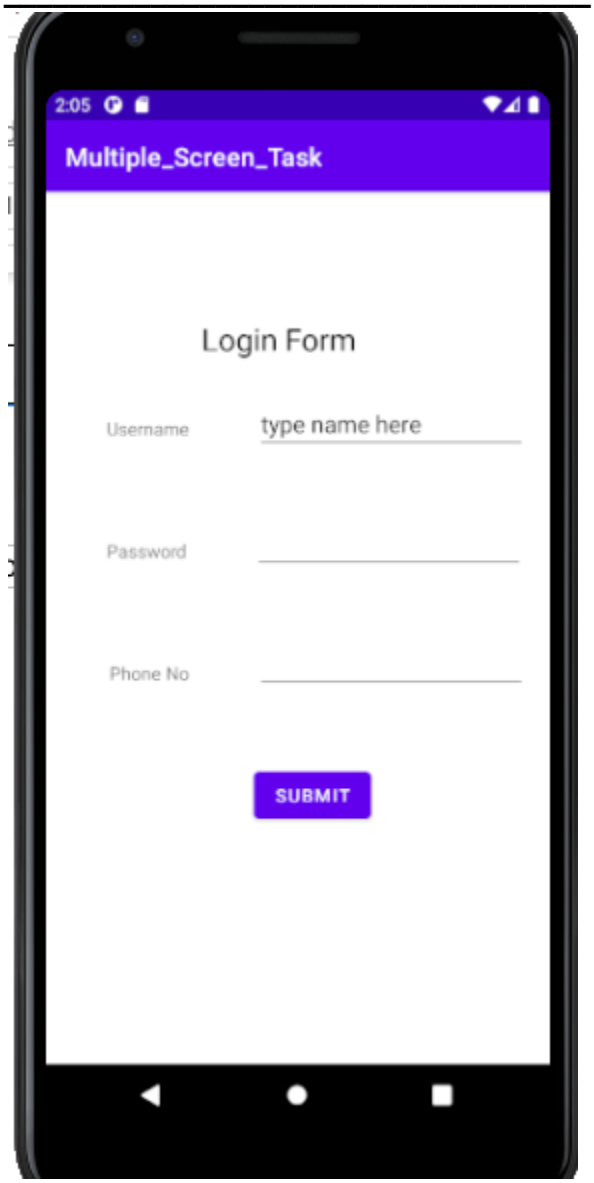
Screen-2

Step-2 Adding more elements of plain text and password to take input.

XML View

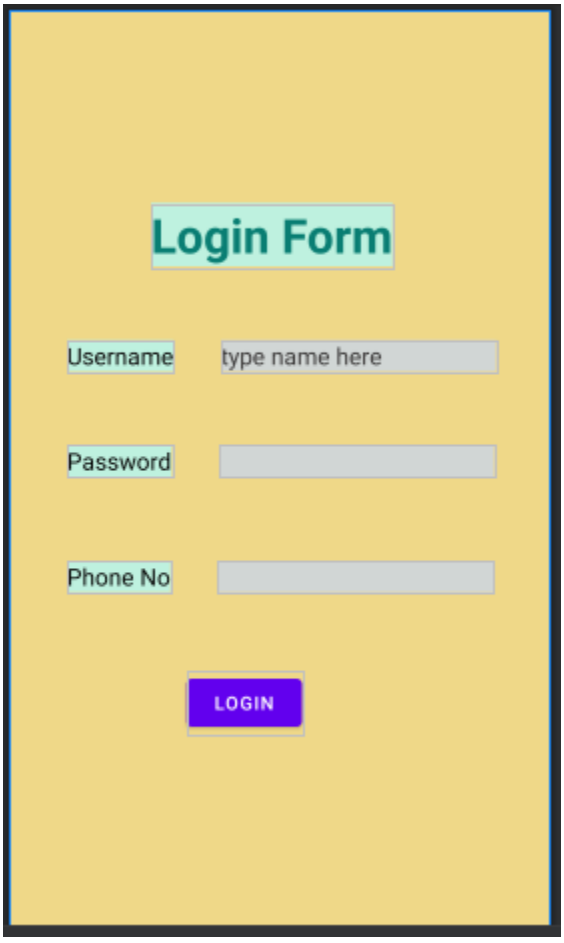
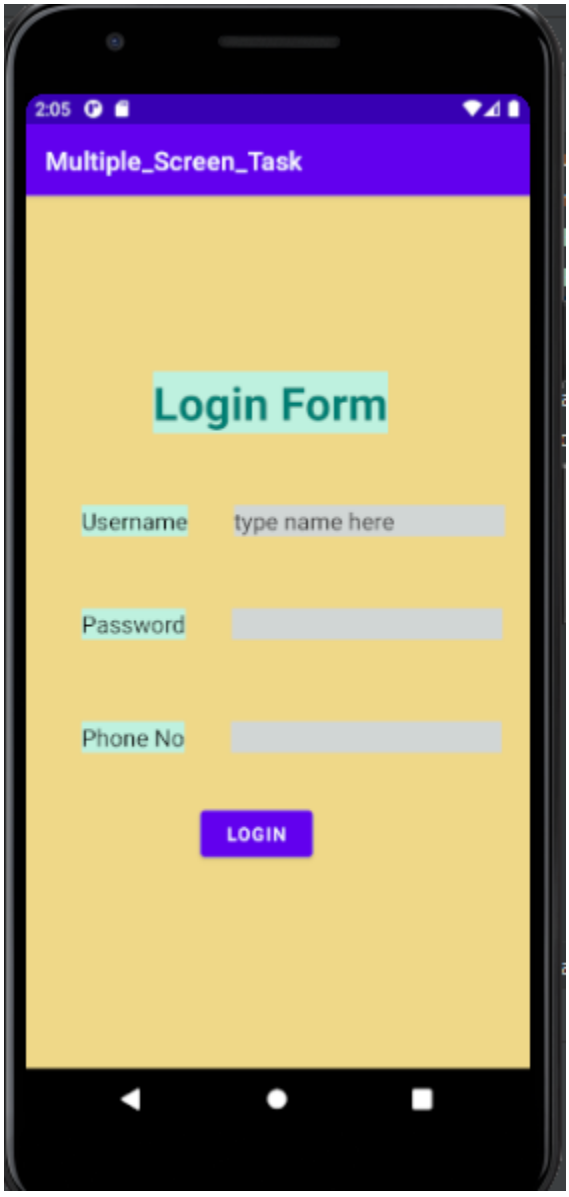


Emulator View



Screen-3

Step-3 Setting constraints and others attributes to each element to set the screen presentation.

XML View	Emulator View
	

Final Screen view taking input:

LECTURE#5

❖ View Group:

A ViewGroup is an **invisible container**. It is a special view that can contain other views (called children.)

- **Layouts:**

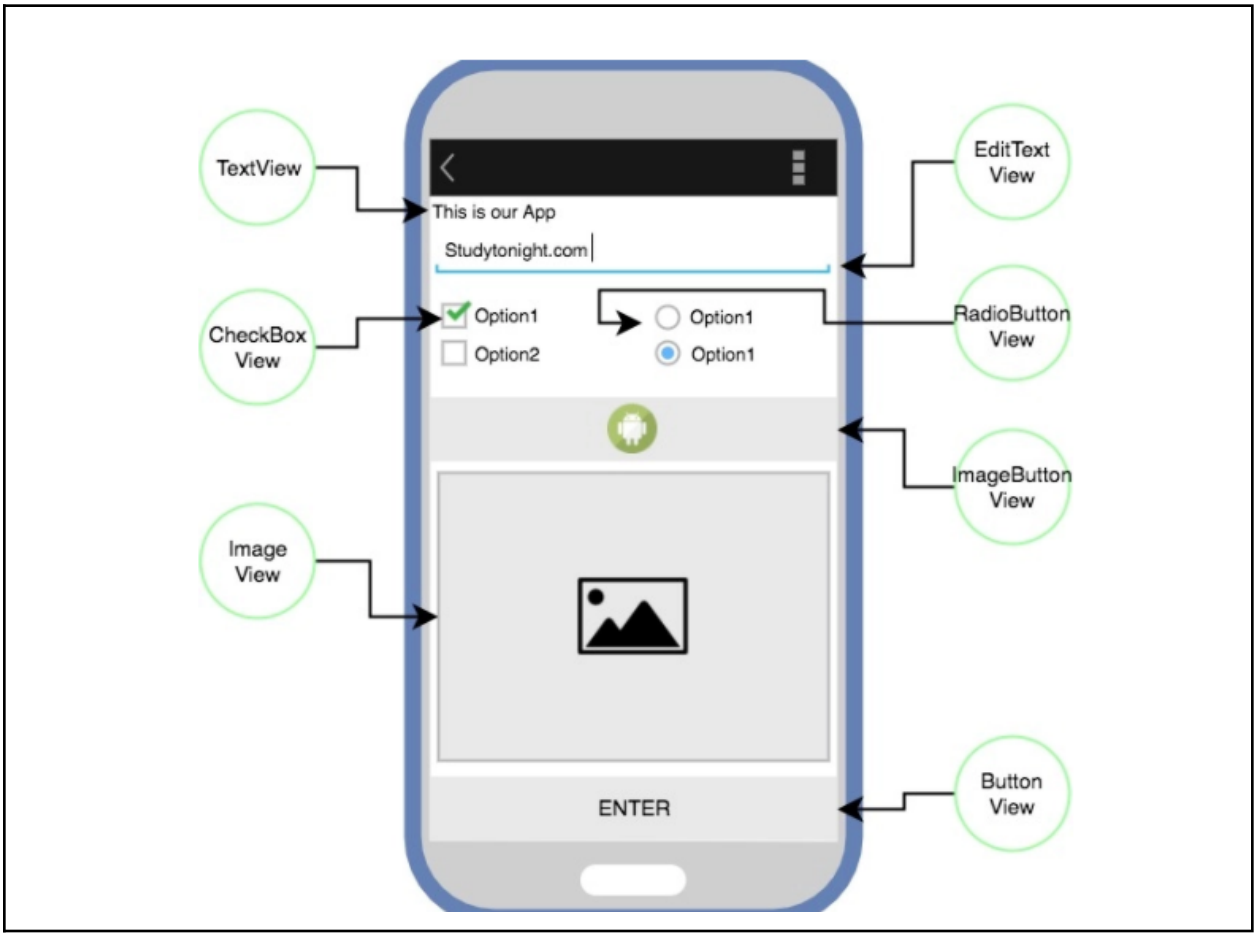
  View group objects are called **“layouts”** can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout .

❖ View:

View is the basic building block of **UI**(User Interface) in android. View refers to the android. view. View class, which is the super class for all the GUI components like TextView, ImageView, Button etc. It draws something that user can see and interact with.

- **Widgets:**

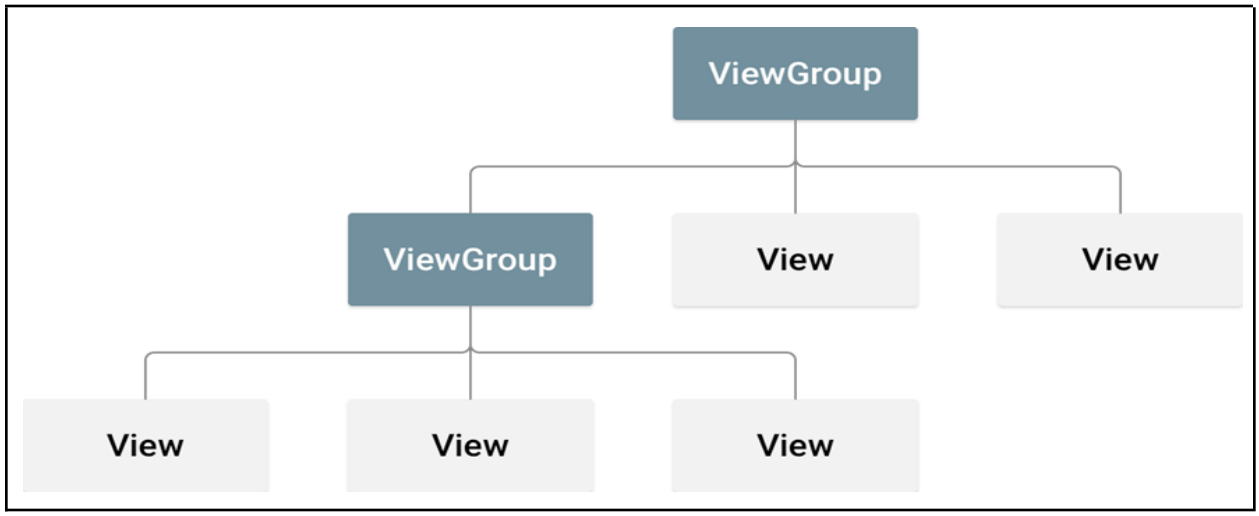
  The View objects are usually called **"widgets"** and can be one of many subclasses, such as Button or TextView.



Root Element:

Each layout file must contain exactly one **root element**, which must be a **View or ViewGroup** object. Once you've defined the root element, you can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout. For example, here's an XML layout that uses a vertical LinearLayout to hold a TextView and a Button.

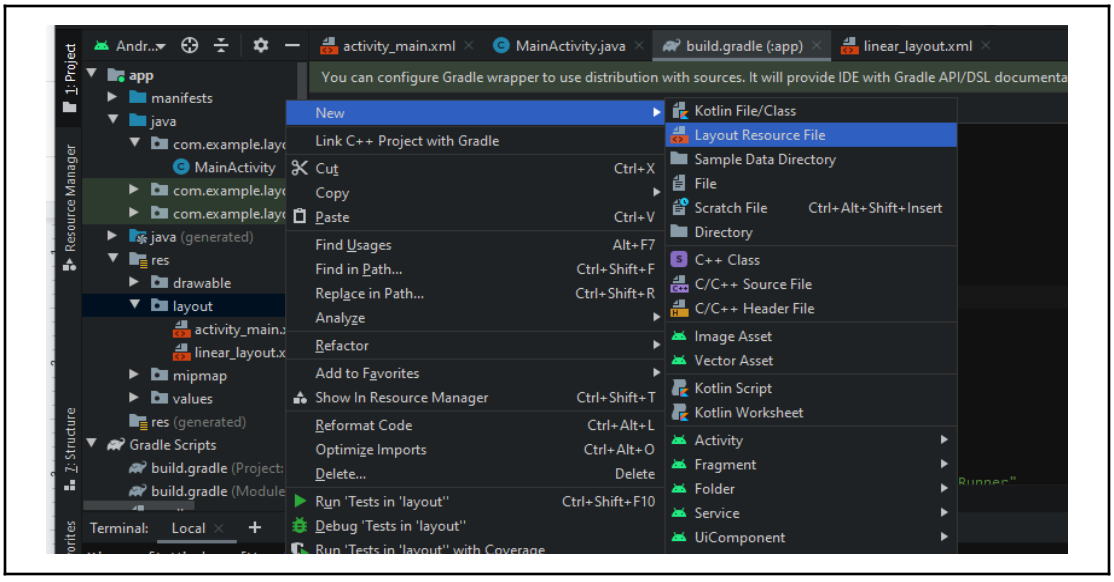
Constraint Layout is the root element by default.



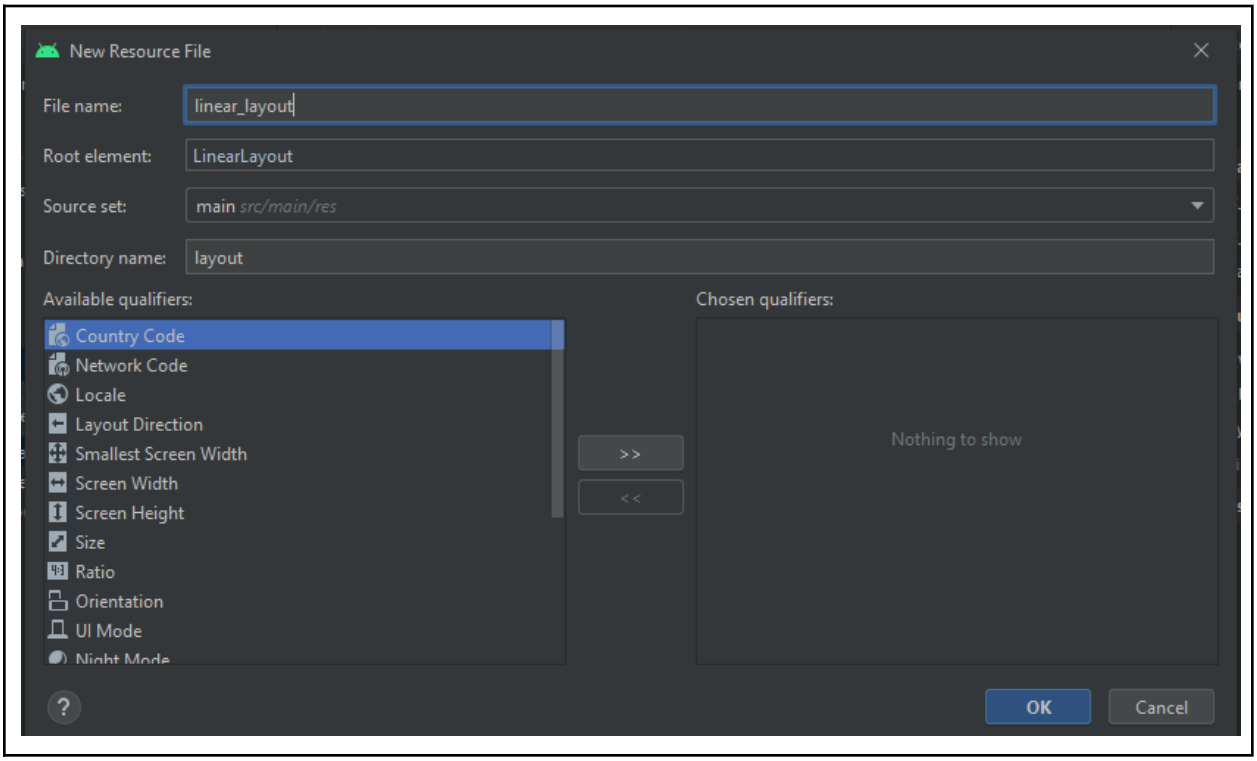
❖ Layouts Practice:

Step-1 Create a new project with empty template.

Step-2 Add a new resource file.



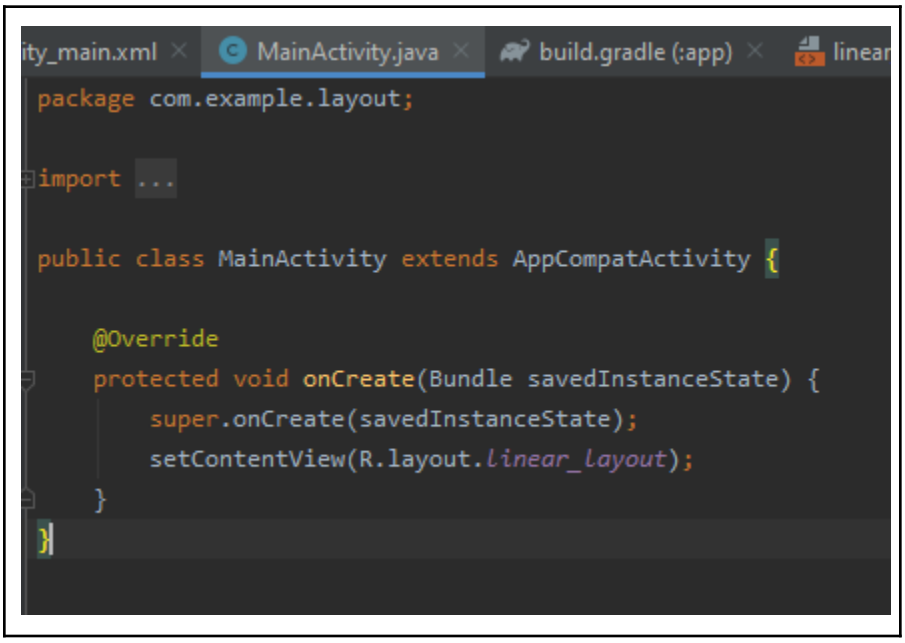
Step-3 Select the root element of **LinearLayout**.



Step-4 Add a text view and button, and you will see that these elements are displayed in a linear layout on xml design view.

❖ Load XML Resources:

Step-5 In MainActivity.java file change the **setContentView(R.layout .___)**, to the layout file you want to run on emulator.



Layout Parametres:

They are used by views to tell their parents how they want to be laid out.

❖ Constraint Layout:

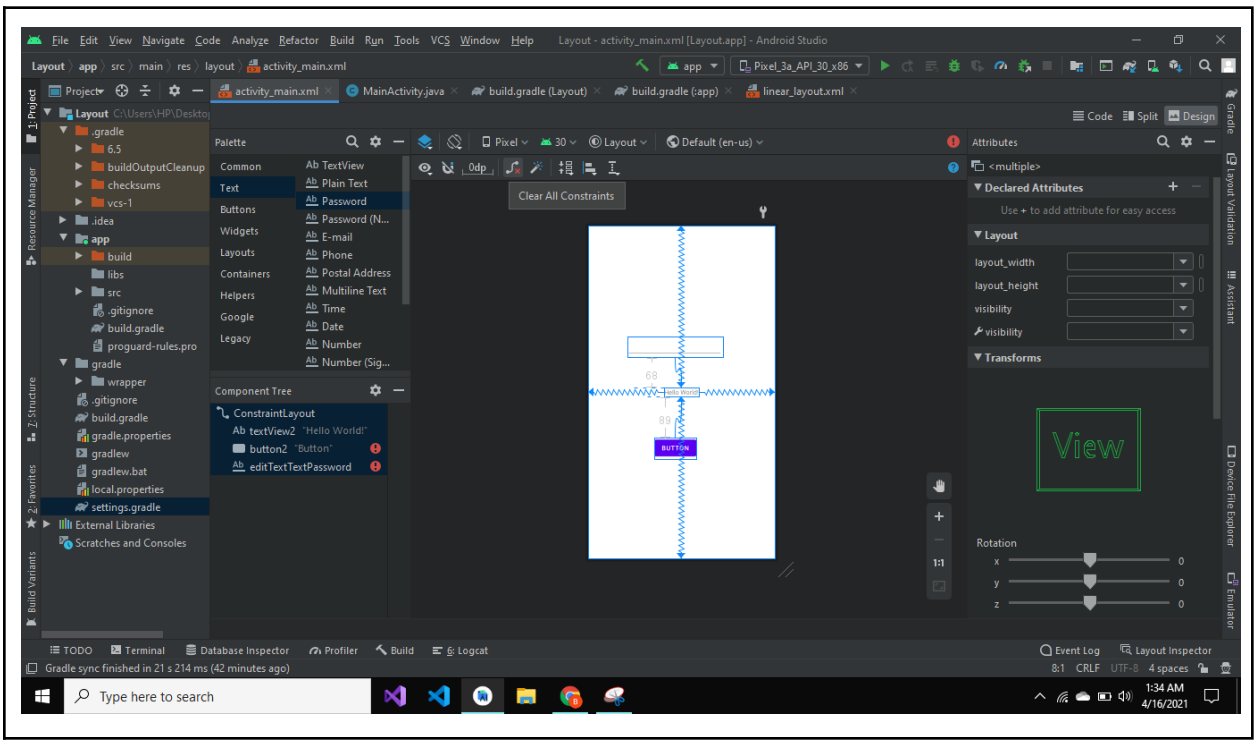
In Constraint Layout, you can connect all the view components to each other and place them on screen.

- To define a **view's position** in ConstraintLayout , you must add at least one horizontal and one vertical constraint for the view. Each constraint represents a connection or alignment to another view, the parent layout, or an invisible guideline.

Clear All constraints:

Step-1 Select all on xml design layout by **ctrl+a**

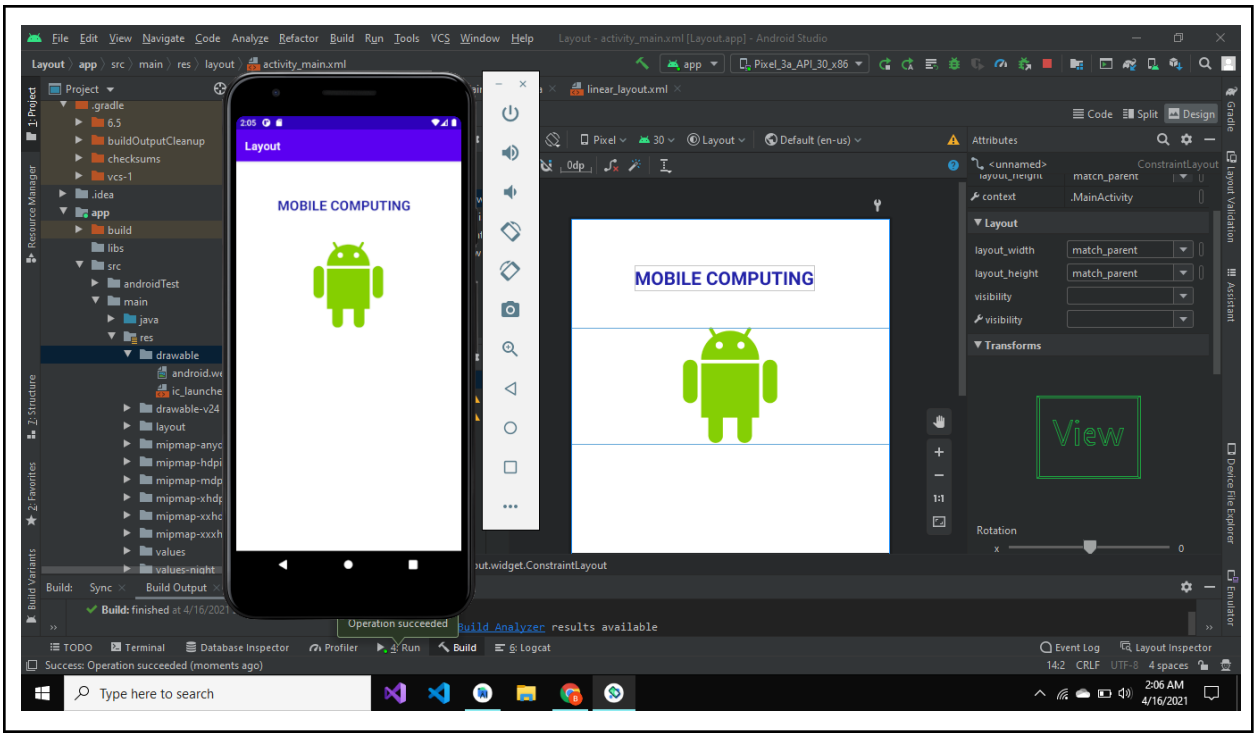
Step-2 Select the option Clear All constraints.

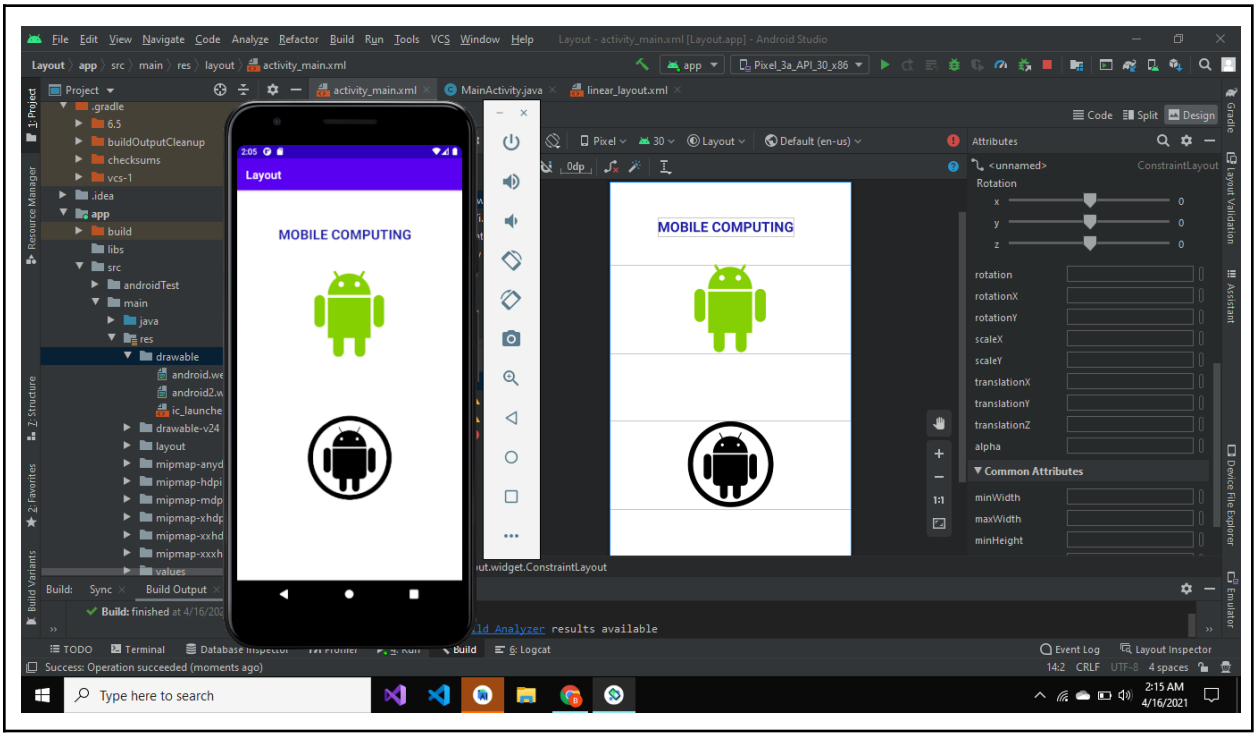


❖ Adding Image in Android Layout:

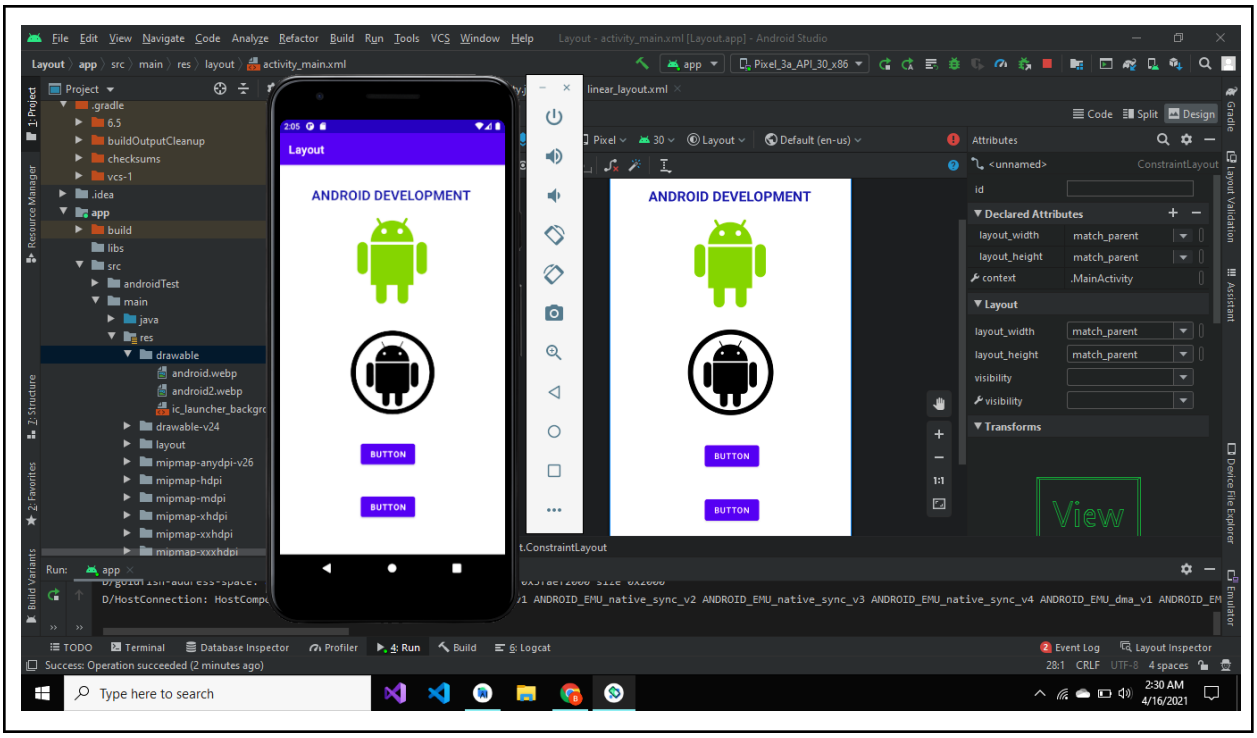
Step-1 In drawable paste any image from your system

Step-2 Add an image view from palette and select the image you want to add.

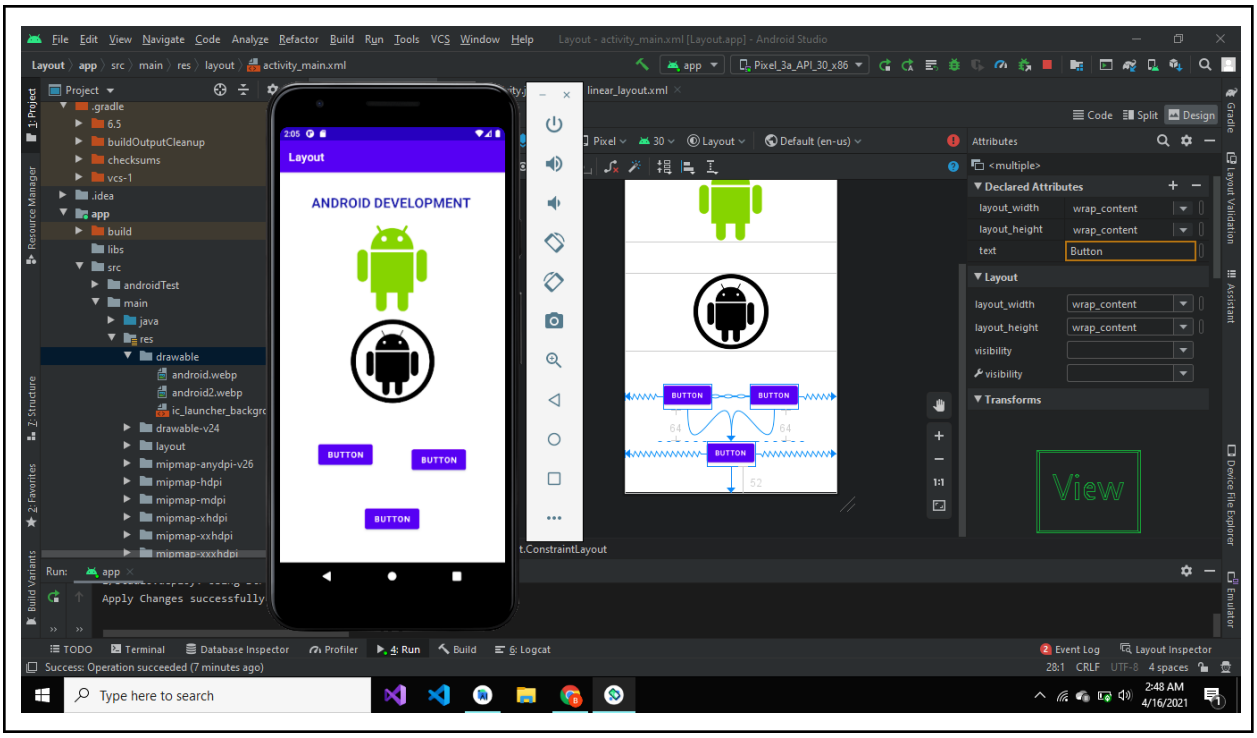




❖ **Adding buttons with alignment:**



❖ Buttons using Layout Constraint Center-Horizontally:



2:05
Layout
ANDROID DEVELOPMENT
BUTTON
BUTTON
Someone is There
BUTTON

2:05
Layout
ANDROID DEVELOPMENT
BUTTON
BUTTON
Some one new is There
BUTTON

❖ Making Instances for buttons:

Reminder place details

LECTURE#6

❖ Activity:

An Activity is an application component that represents one window, one hierarchy of views.

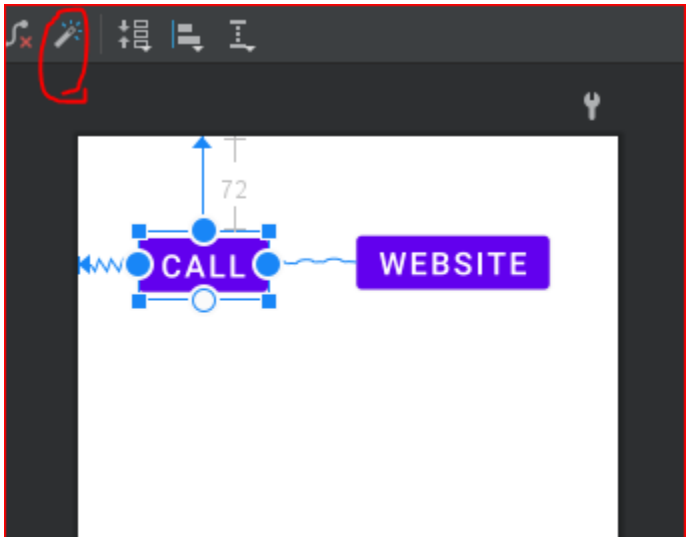
- Typically fills the screen, but can be embedded in other Activity or appear as floating window.
- Java class, typically one Activity in one file.

❖ Intent:

An intent is to perform an action on the screen. It is mostly used to start activity, send broadcast receiver,start services and send message between two activities.

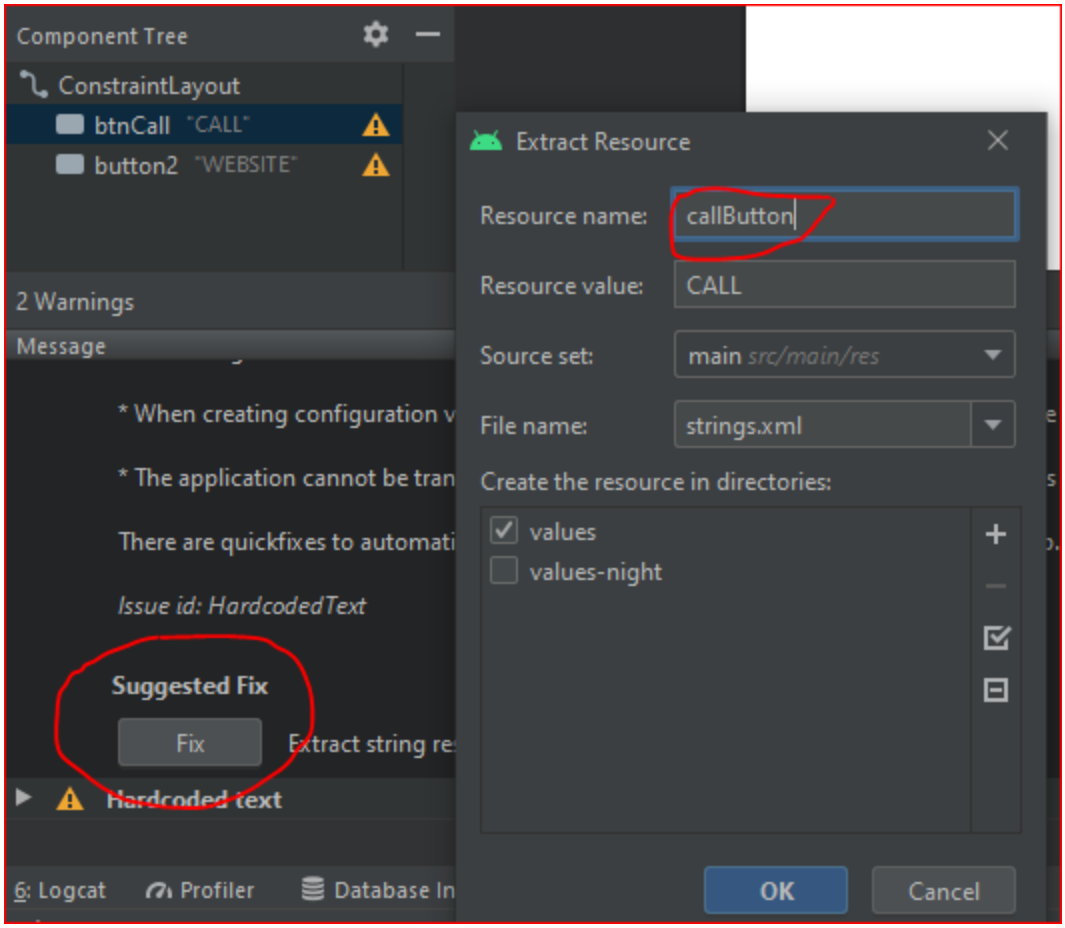
To Set Constraints Automatically:

Step-1 Use the highlighted tool to set constraints automatically.

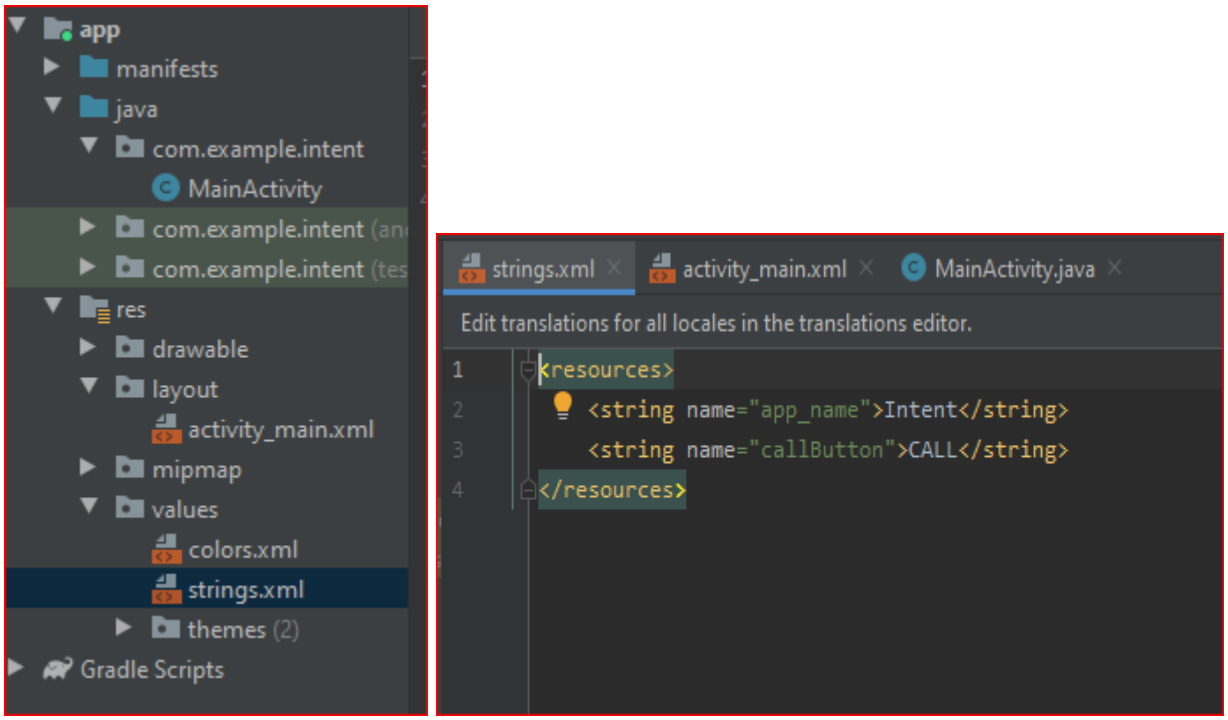


To define String Resources:

Step-2 Click on the fixes button and set a string name for the element.



- **The string name will be defined in string.xml file that is**



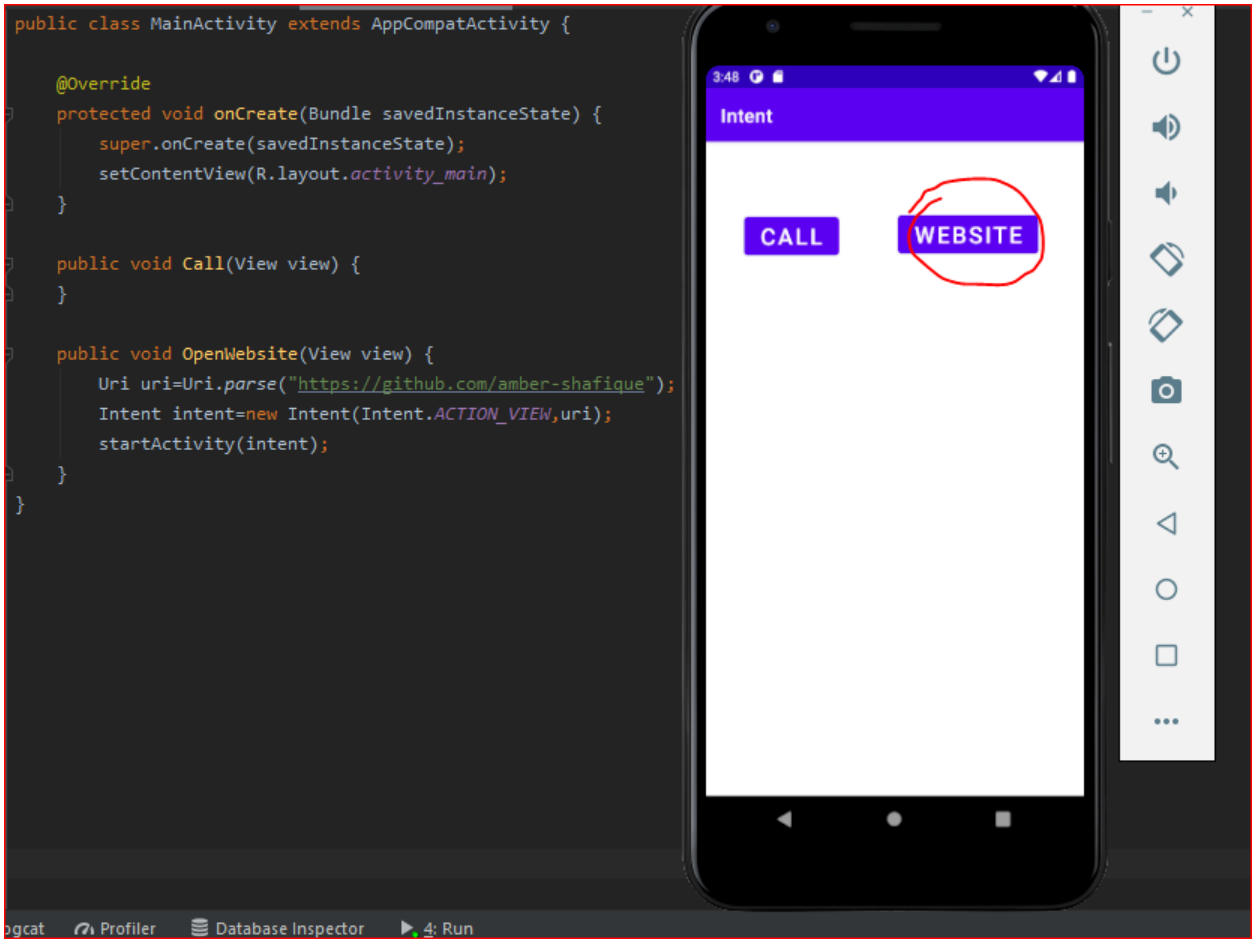
❖ Intent Uses

❖

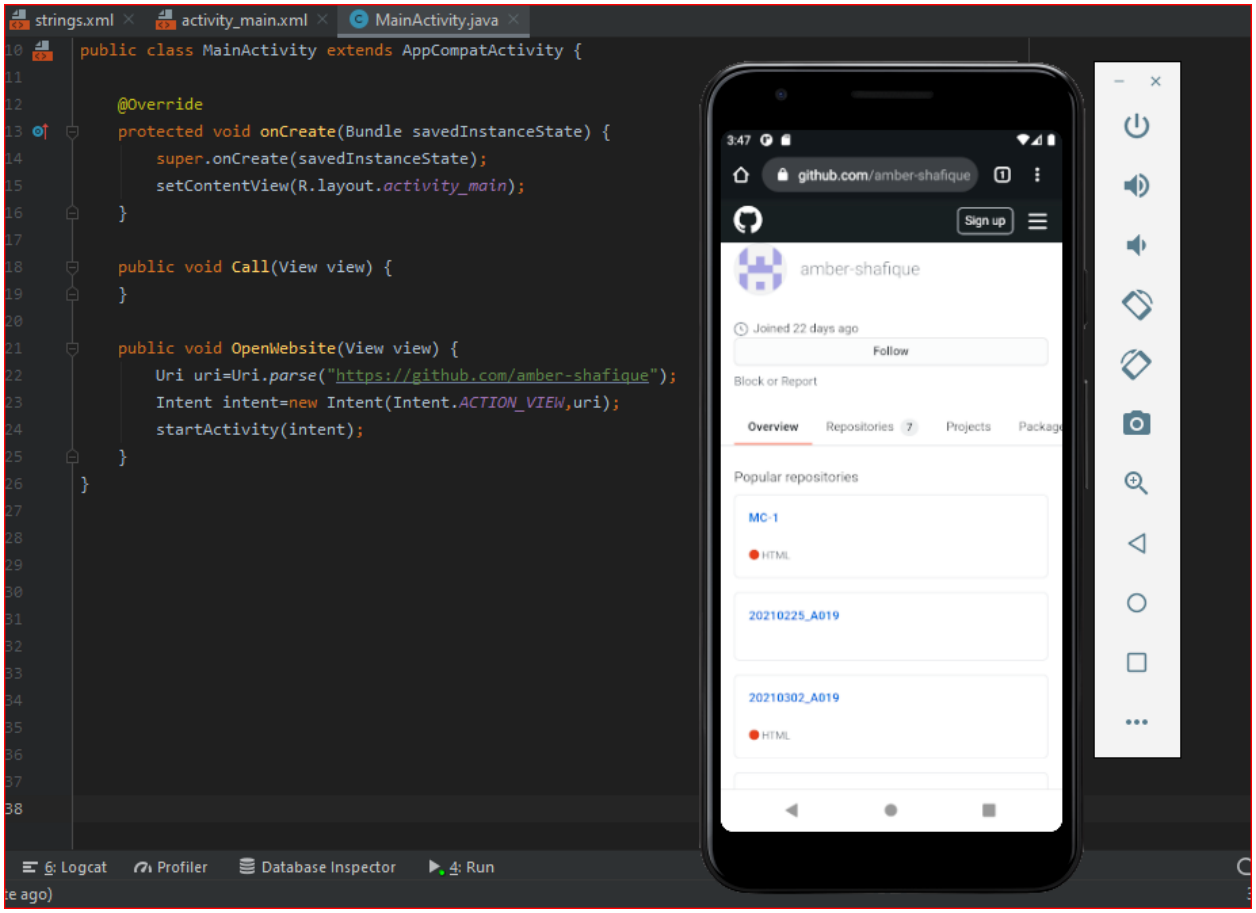
1. To open Website using Intent:

```
Uri uri=Uri.parse("https://github.com/amber-shafique");
Intent intent=new Intent(Intent.ACTION_VIEW,uri);
startActivity(intent);
```

Step-1 Add the above code onClick instance of button.
Step-2 Click on the website button on click of which you have added intent to open website.



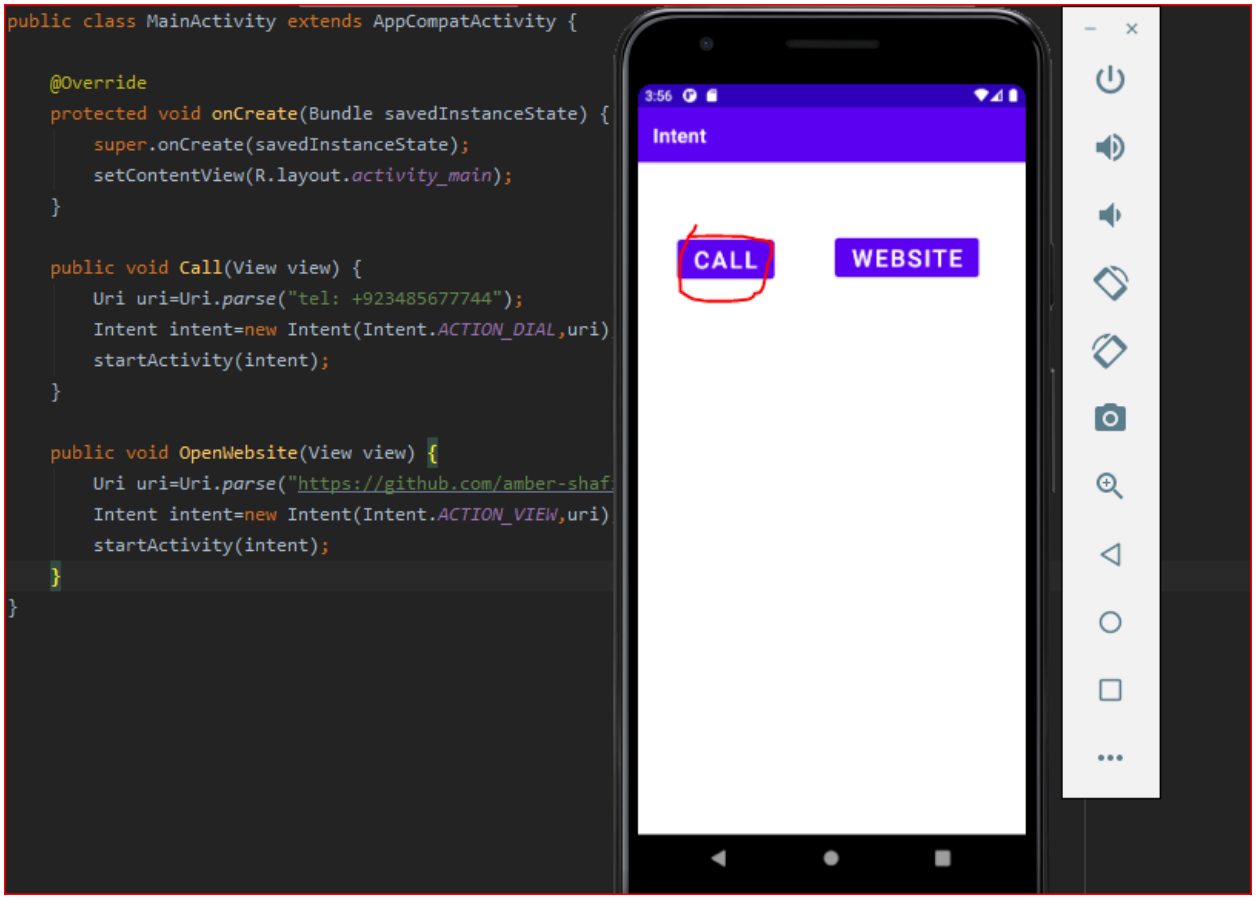
- Just after Clicking the **website** button , the url will open



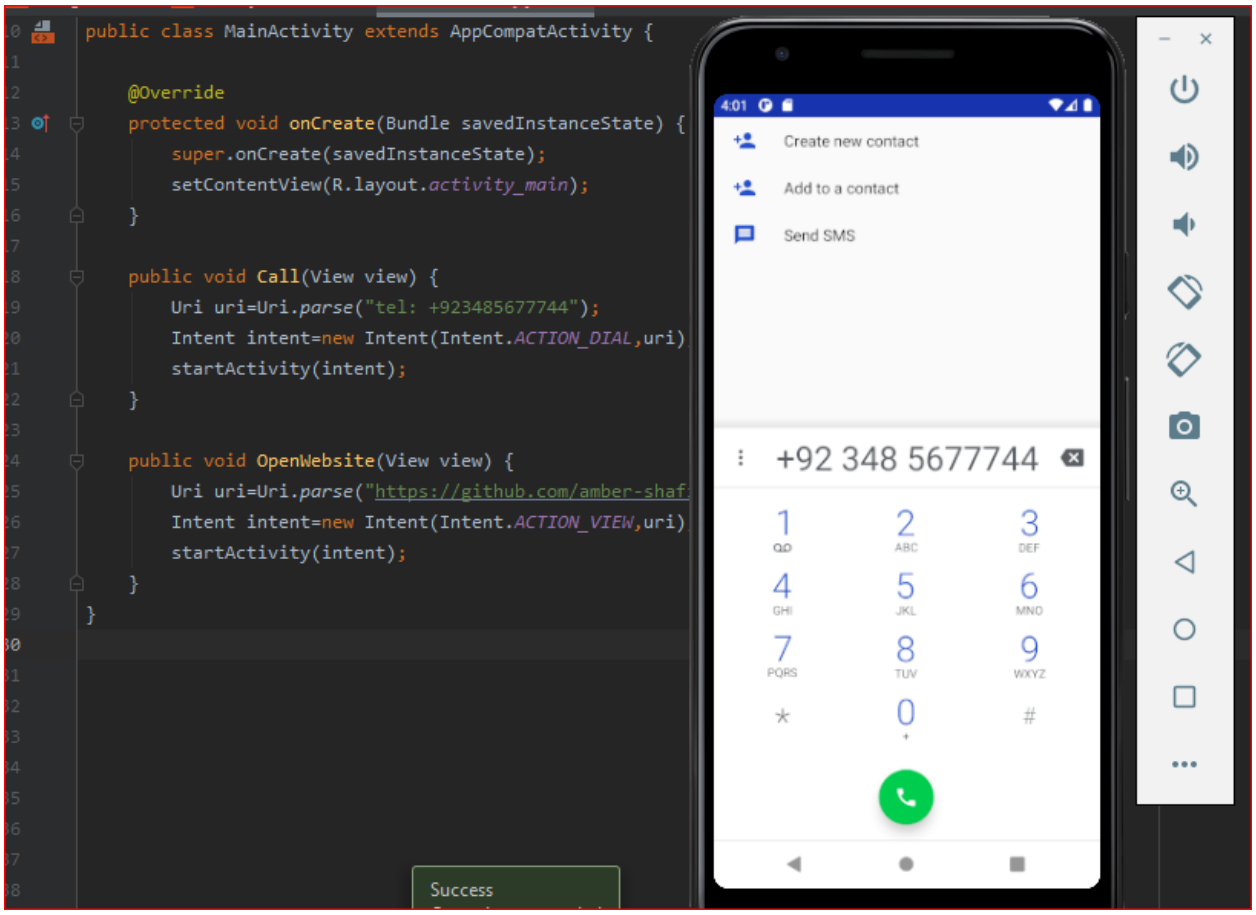
2. To Dial a Call using Intent:

```
Uri uri=Uri.parse("tel: +923485677744");
Intent intent=new Intent(Intent.ACTION_DIAL,uri);
startActivity(intent);
```

Step-1 Add the above code onClick instance of button.
Step-2 Click on the website button on click of which you have added intent to open call dialer.

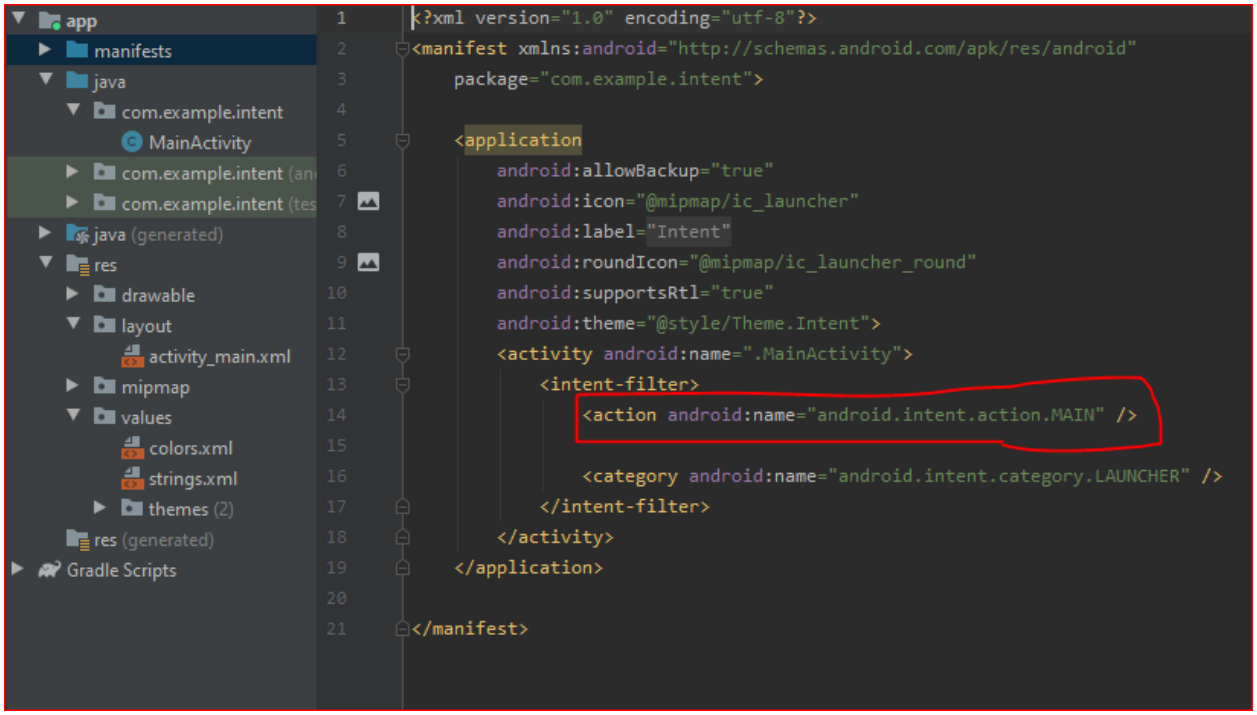


- Just after Clicking the **call** button , the call dialer will open

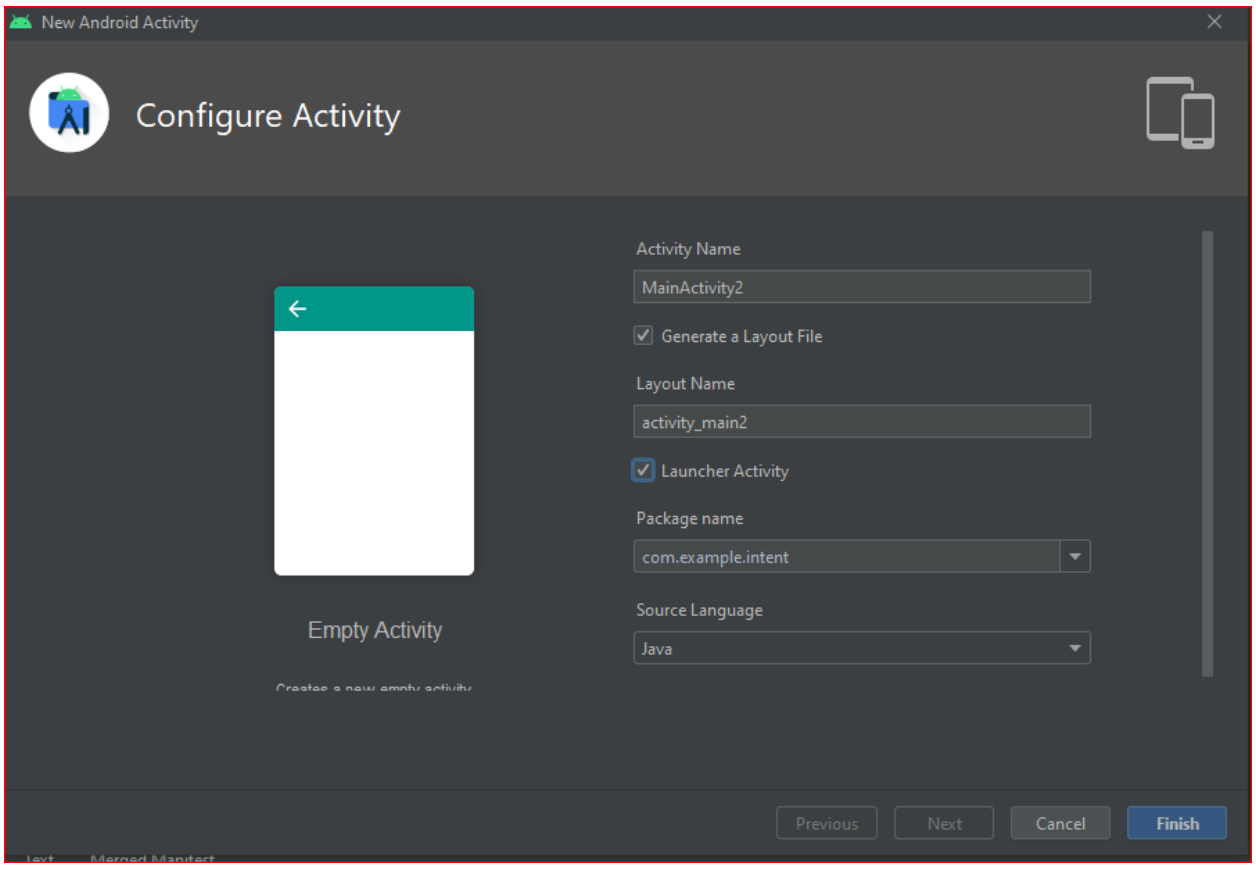


- **Setting another Activity as Launcher Activity:**

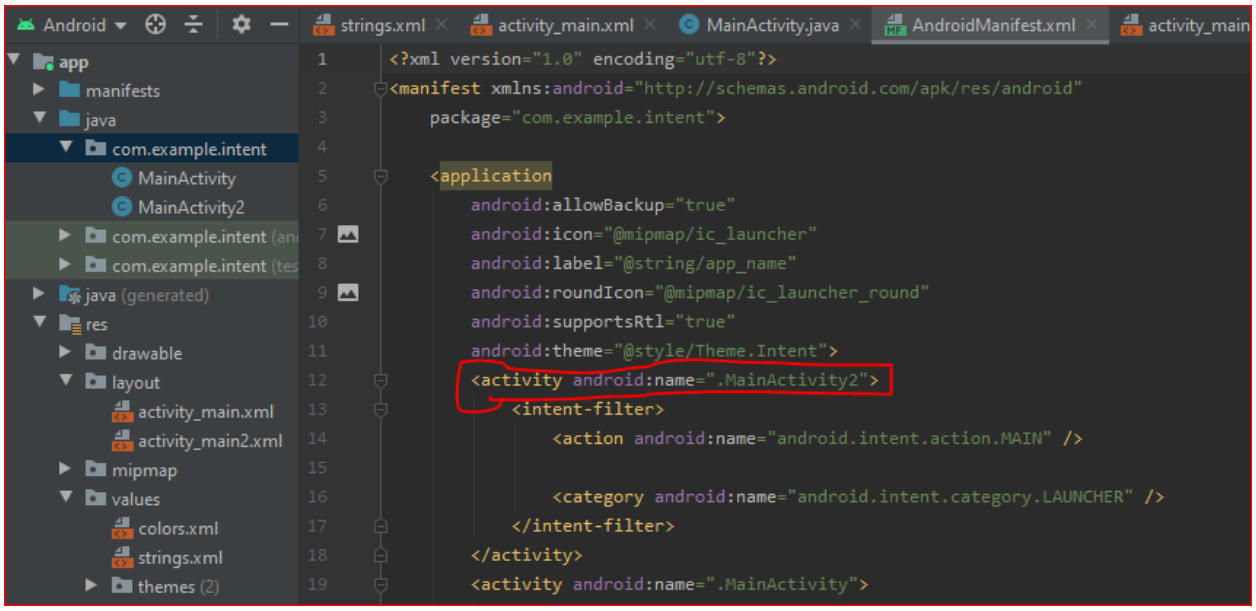
  By default launching activity in **AndoidManifest.xml** file.



Create a new activity and check its attribute of **Launcher Activity.**



Launcher Activity changes in **AndoidManifest.xml** file



3. To move from one activity to another:

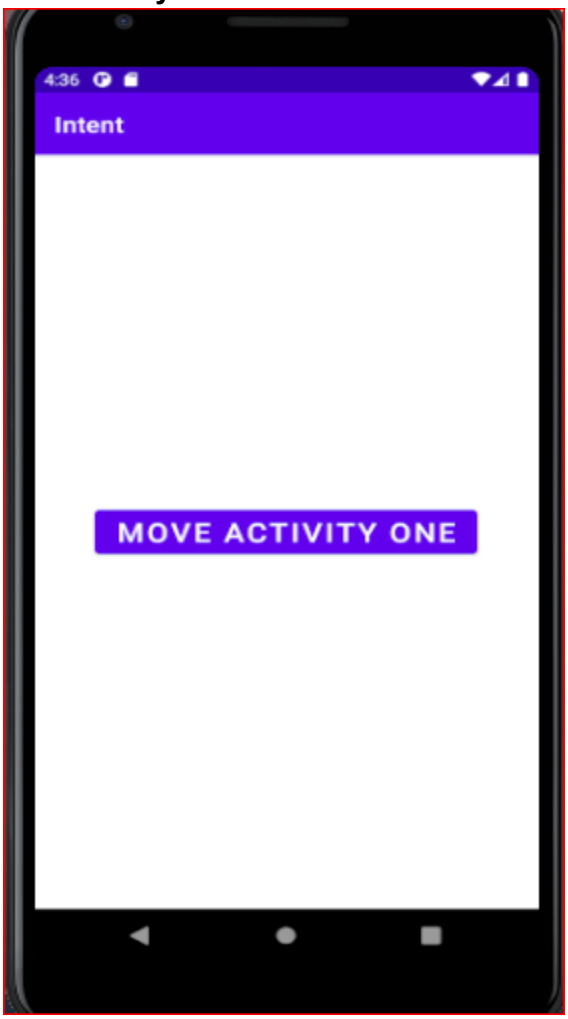
this=> present activity
ActivitName.class=> activity where to move

```
Intent intent=new Intent(this, MainActivity.class);
startActivity(intent);
```

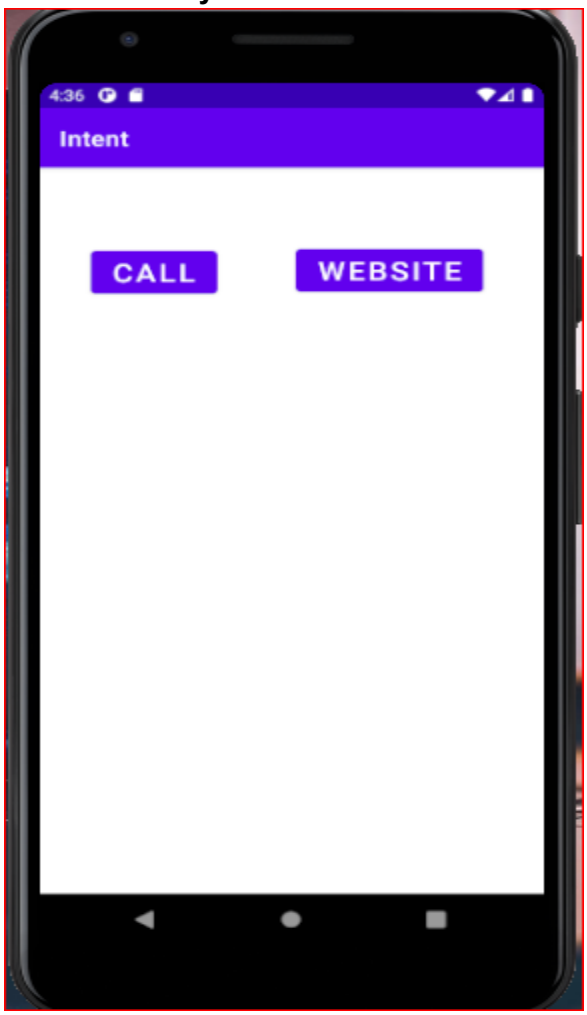
Step-1 Add the above code onClick instance of button.
Step-2 Click on the website button on click of which you have added intent to open another activity..

- Just after Clicking the **Move Activity One** button , the second activity will open.

First activity



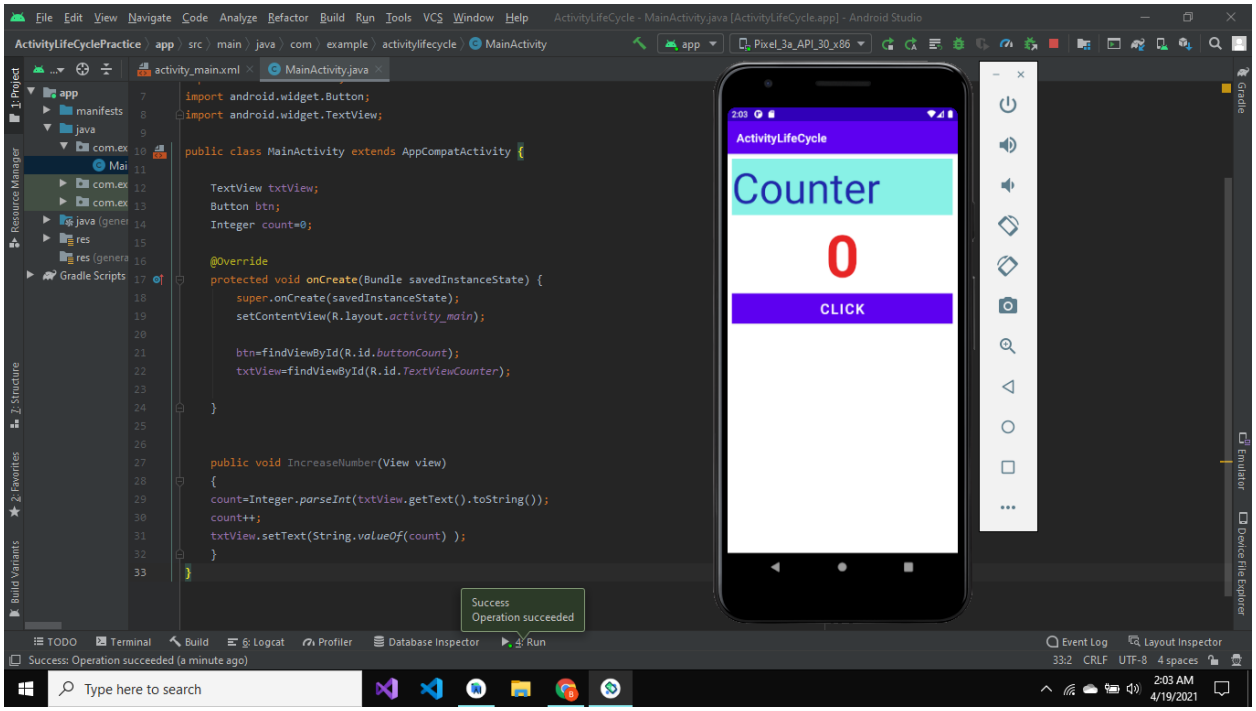
Second activity



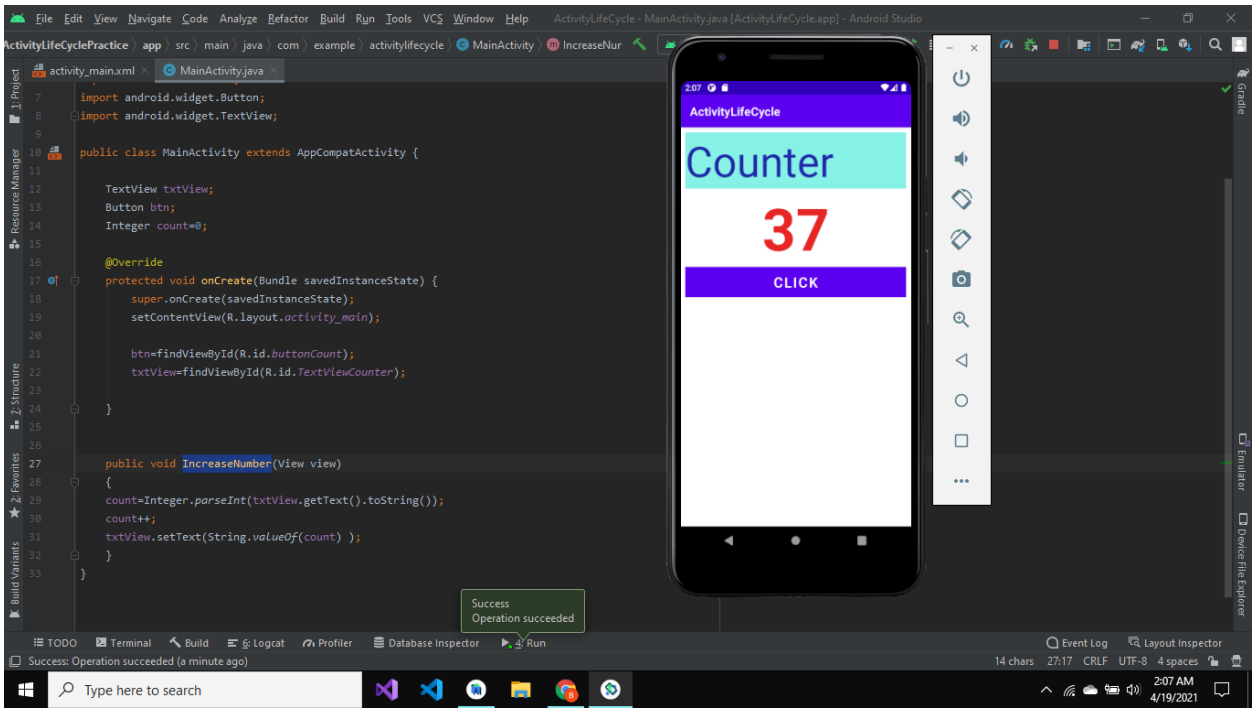
LECTURE#7

❖ Counter:

Adding code to display a counter on screen.

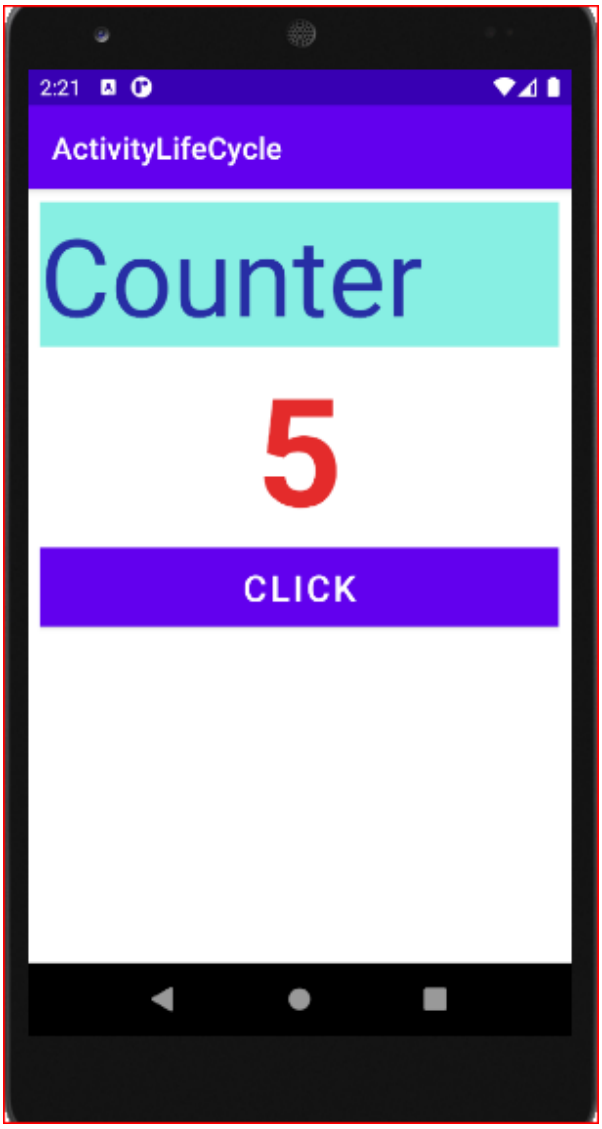


Click on the button to increase value of counter and see the result.



Rotation-Action Problem:

When we rotate the screen counter values goes back to zero

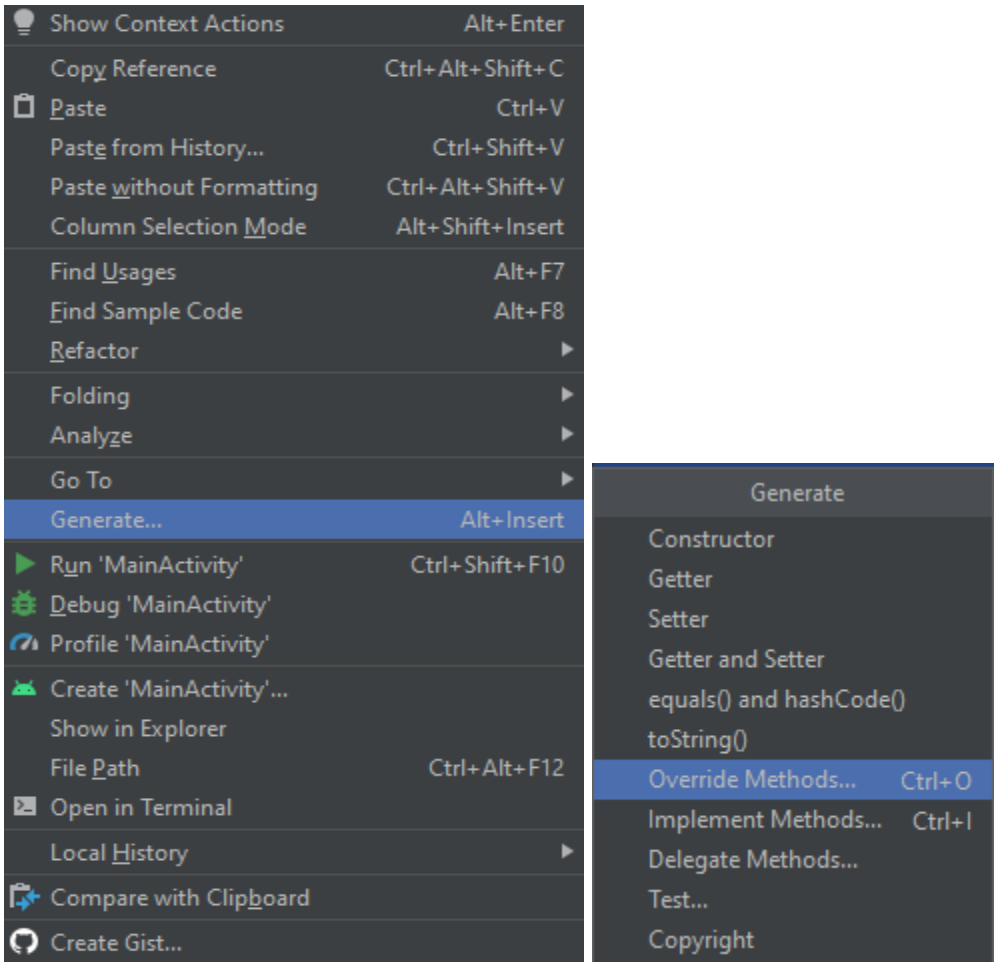
Before Rotation:	**After Rotation:**
	

❖ Activity LifeCycle

- **Create a LOG**

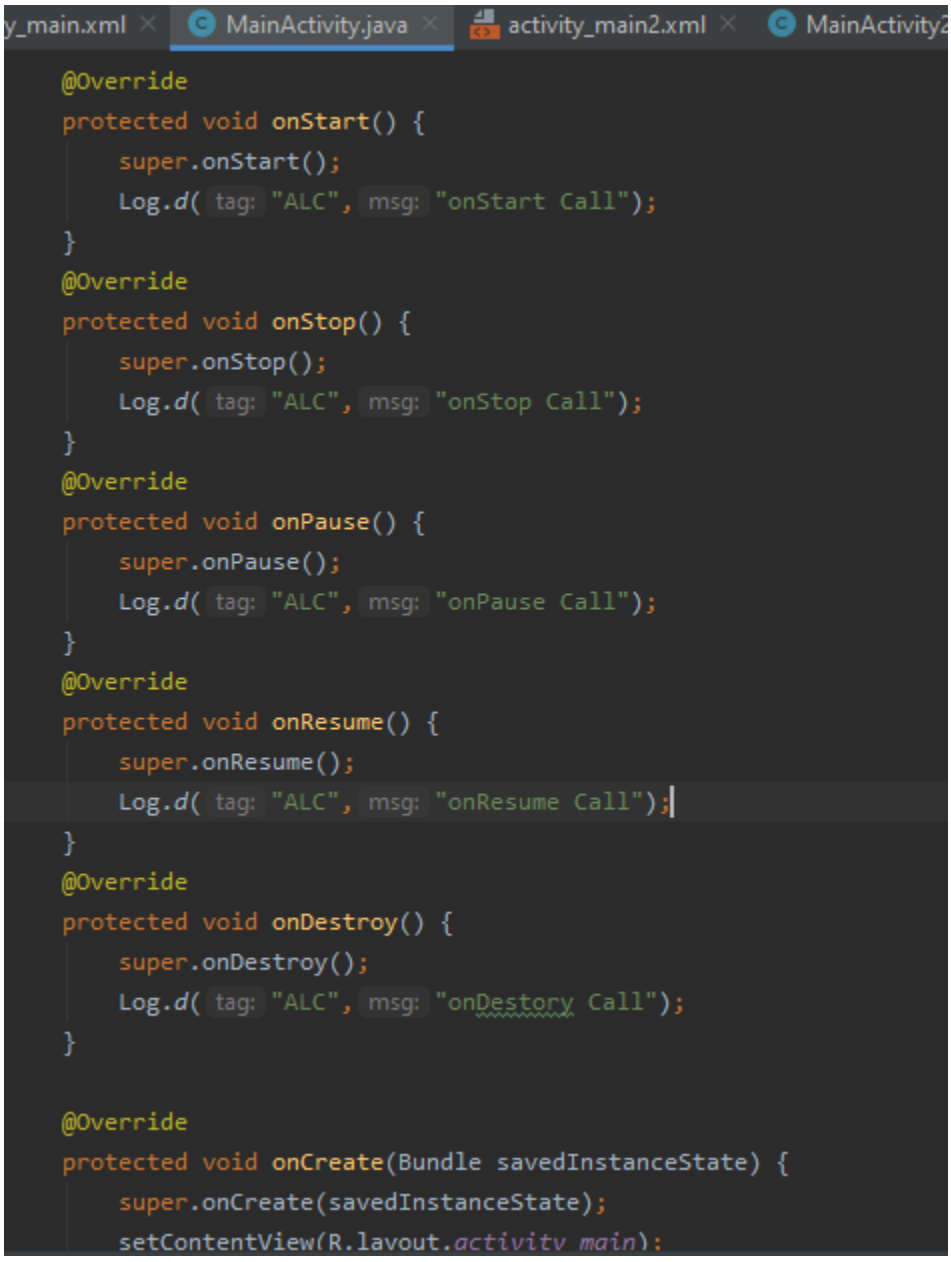
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    btn=findViewById(R.id.buttonCount);
    txtView=findViewById(R.id.TextViewCounter);

    Log.d( tag: "ALC", msg: "onCreate Call");
}
```

- **To Override methods:**



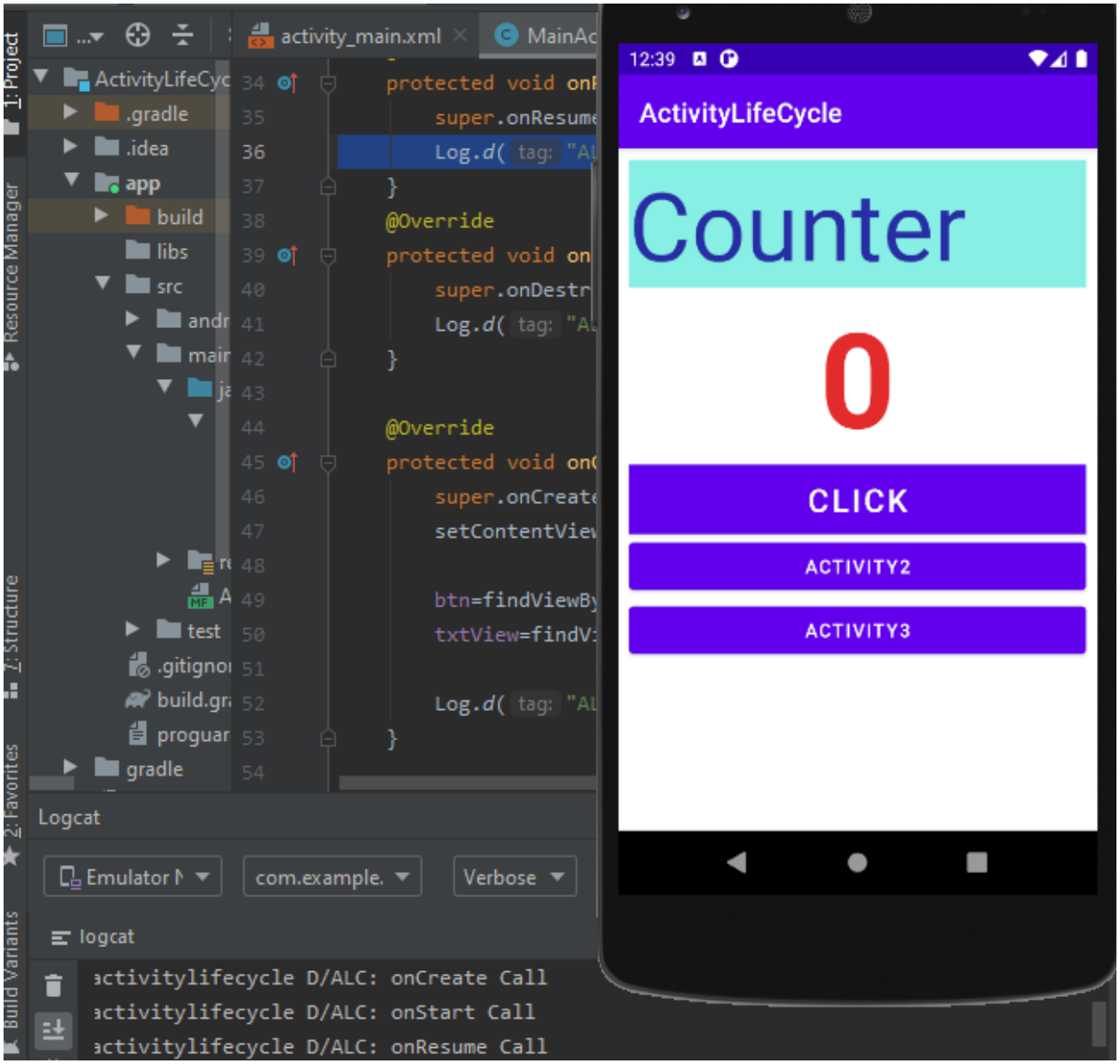
Override the 6 methods in main activity to check their logs in logcat.

- **onCreate()**
- **onStart()**
- **onResume()**
- **on pause()**
- **on Stop()**
- **onDestroy()**

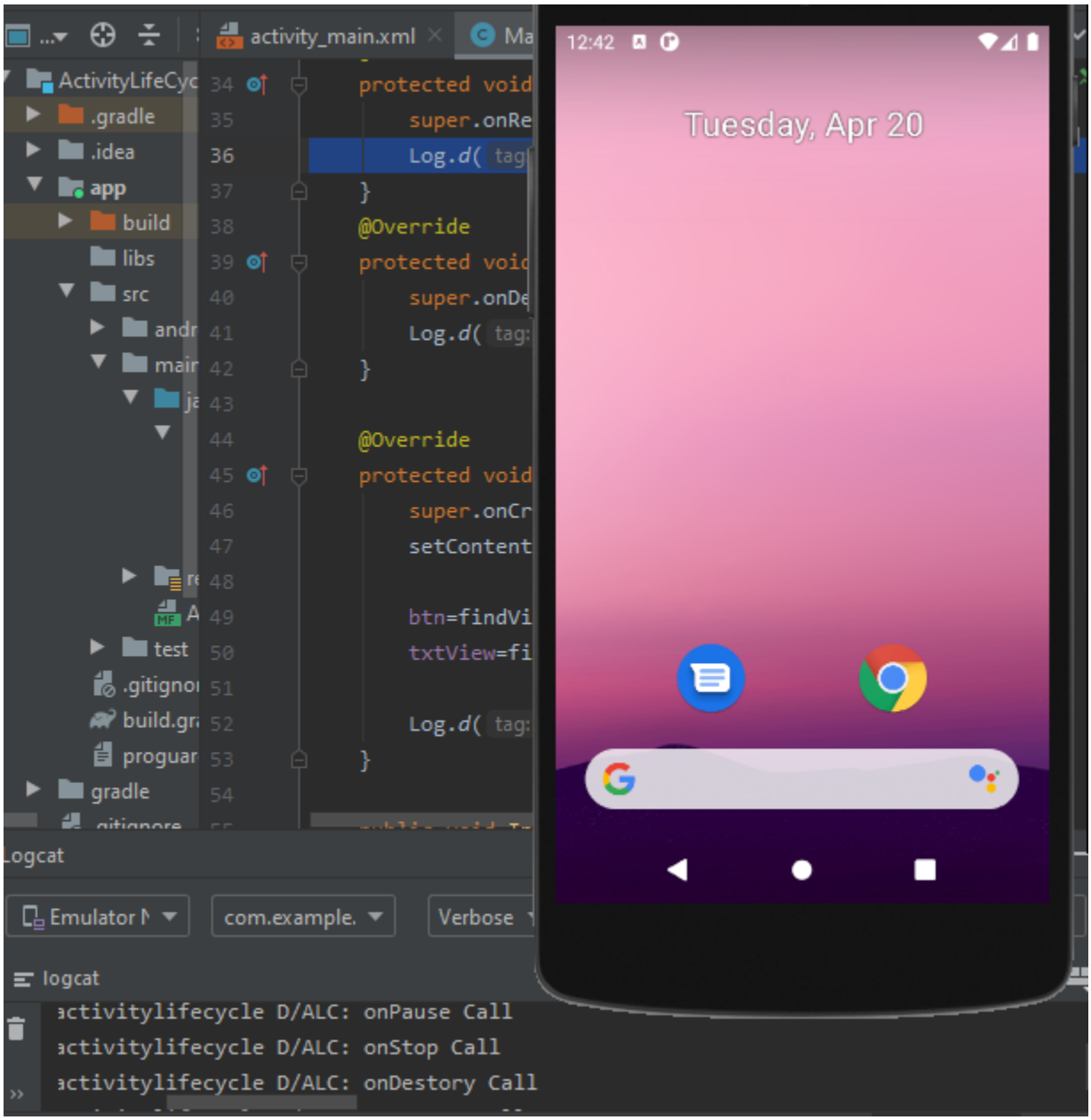


❖ ALC of One Activity:

- Run the emulator and check the logcat.
- On running the application following three funcion will be called as shown in logcat
    1. onCreate()
    2. onStart()
    3. onResume()

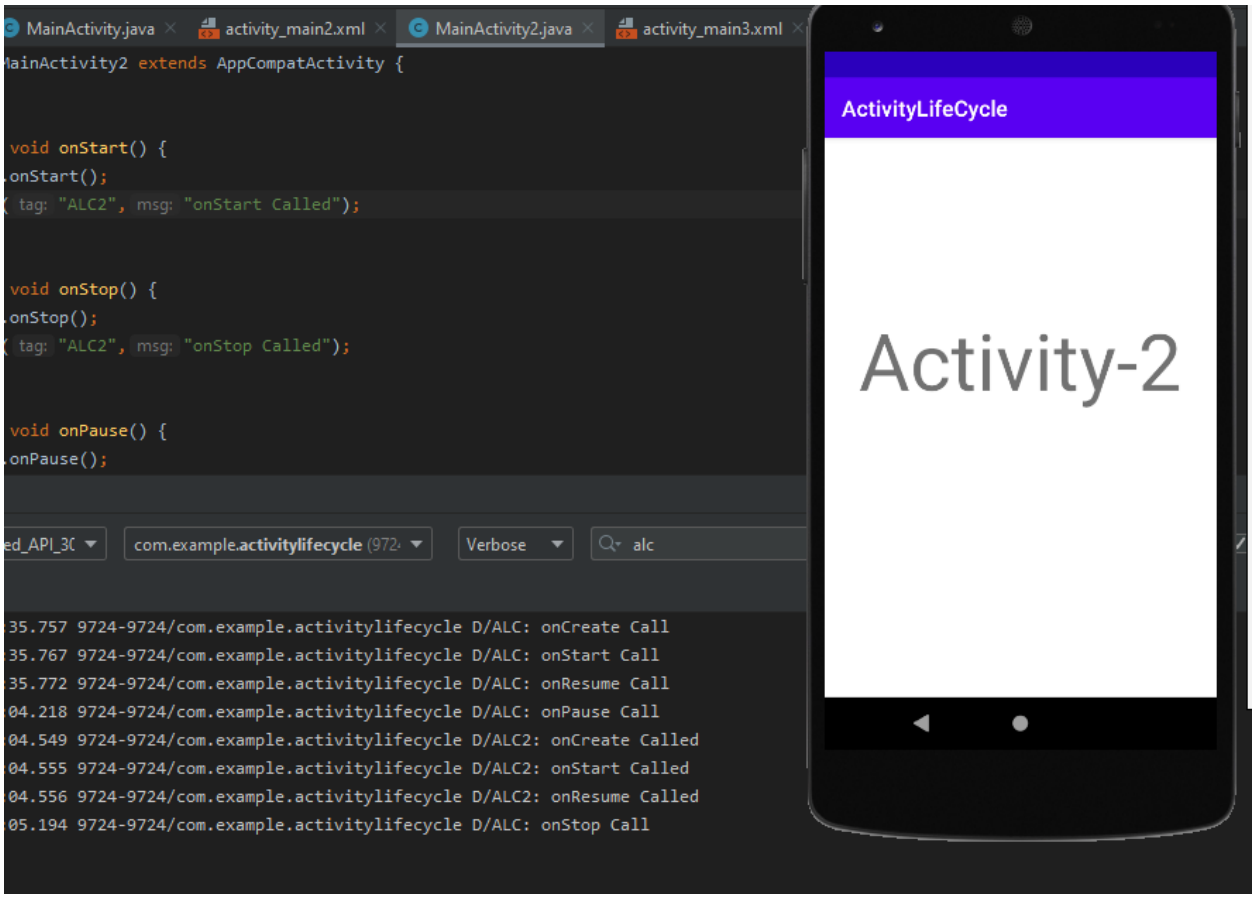


- On just clicking back button on emulator screen following three functions will be called.
    1. on pause()
    2. on Stop()
    3. onDestroy()

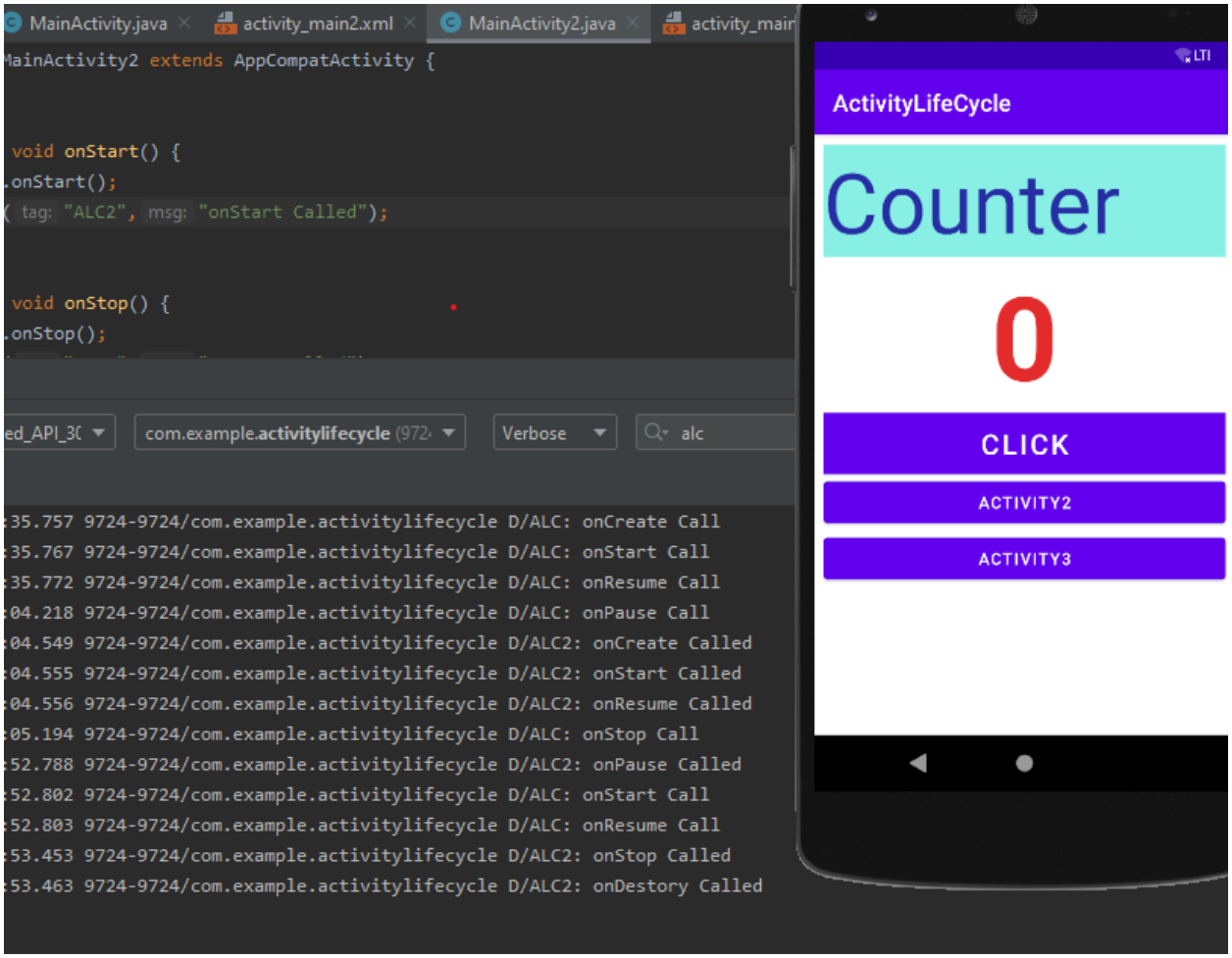


❖ ALC of Main activity to any other activity:

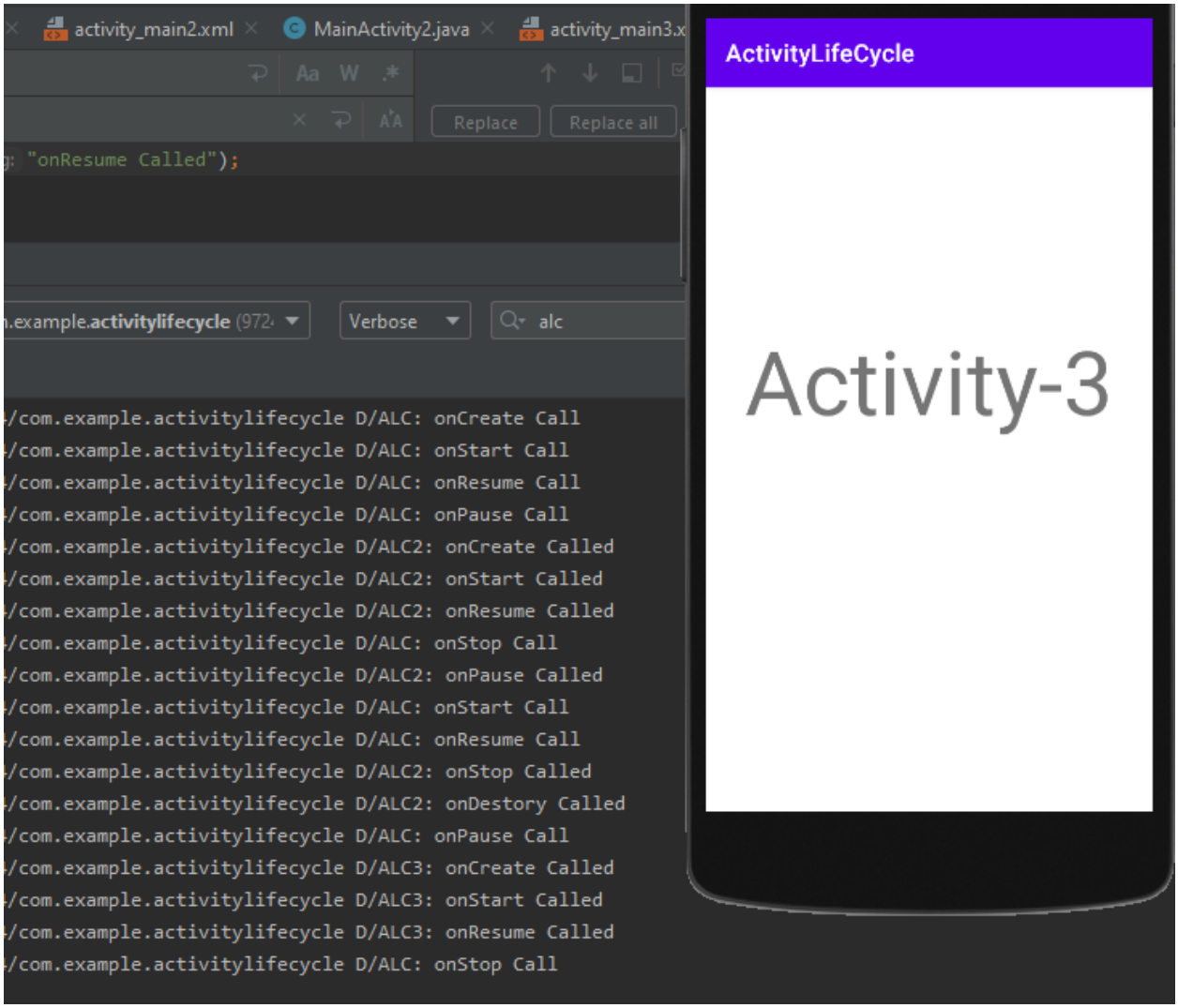
- Override the same 6 functions in activity2.
- Rebuild the application and this time after opening the main activity navigate to second activity and see the logcat.
- Logcat is displaying functions calls by tag names as:
  ALC => for Main Acvtivity
  ALC2 => for second activity when clicked



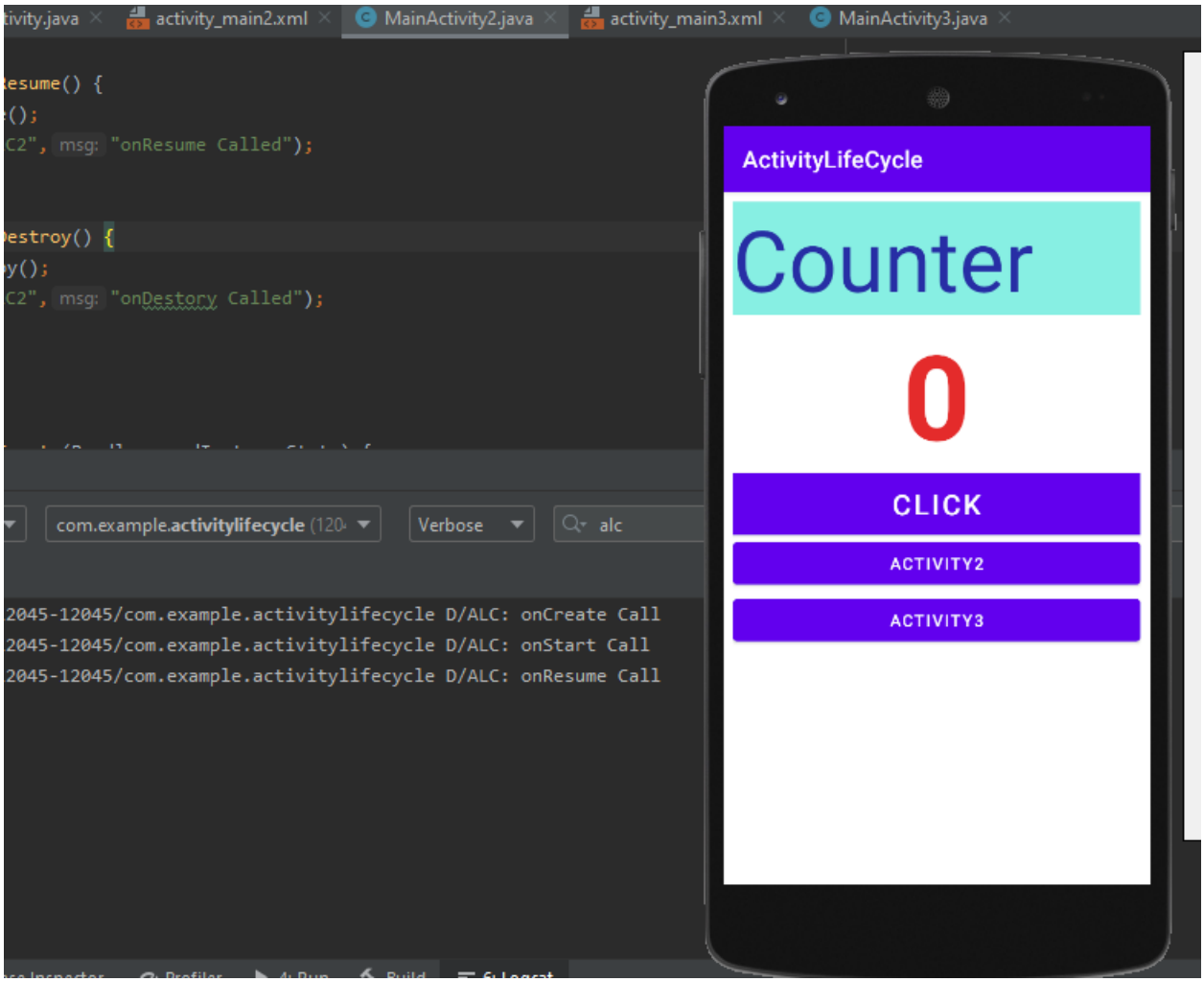
- Now from second activity click back to move back to main activity and see the logcat.

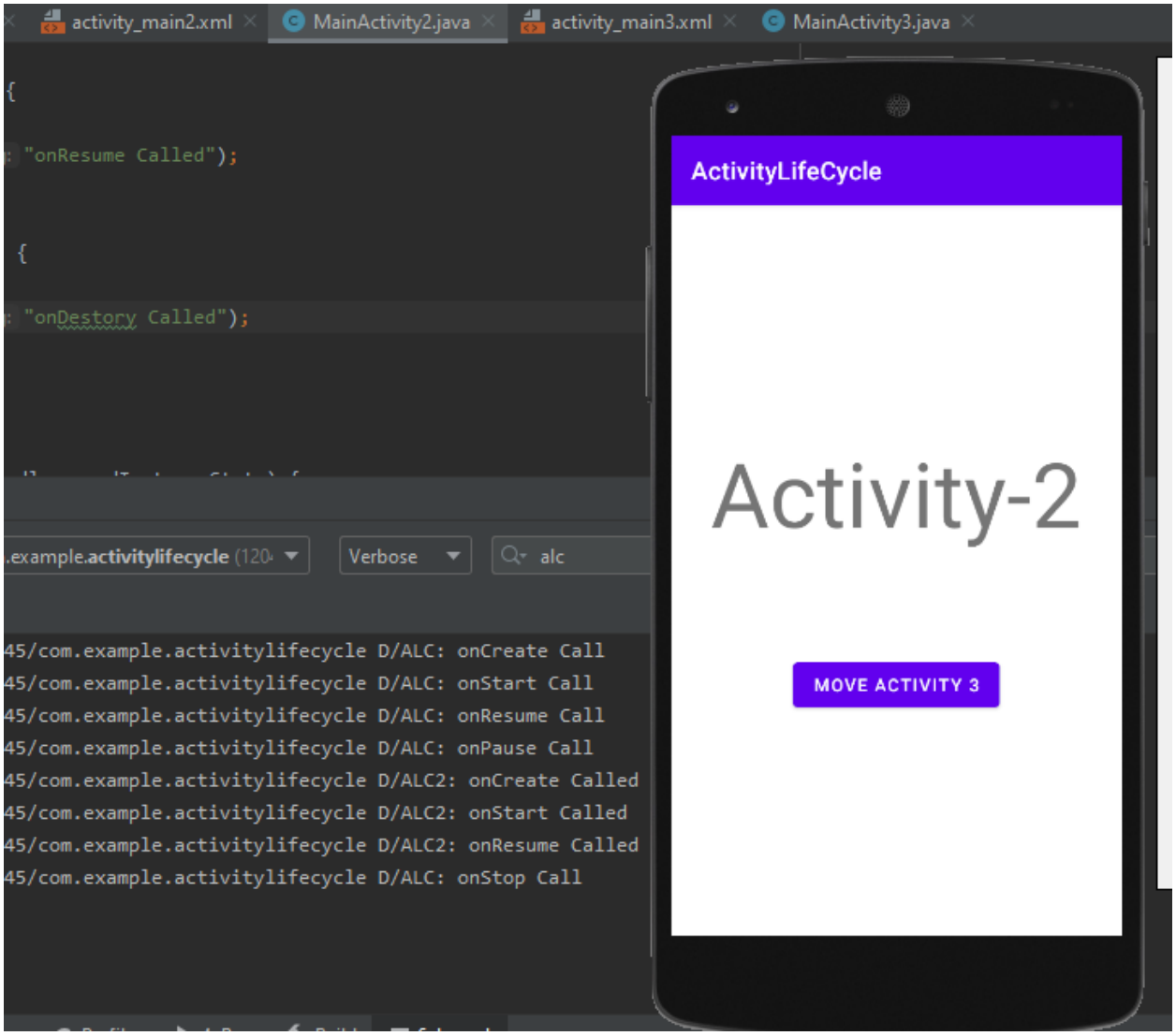


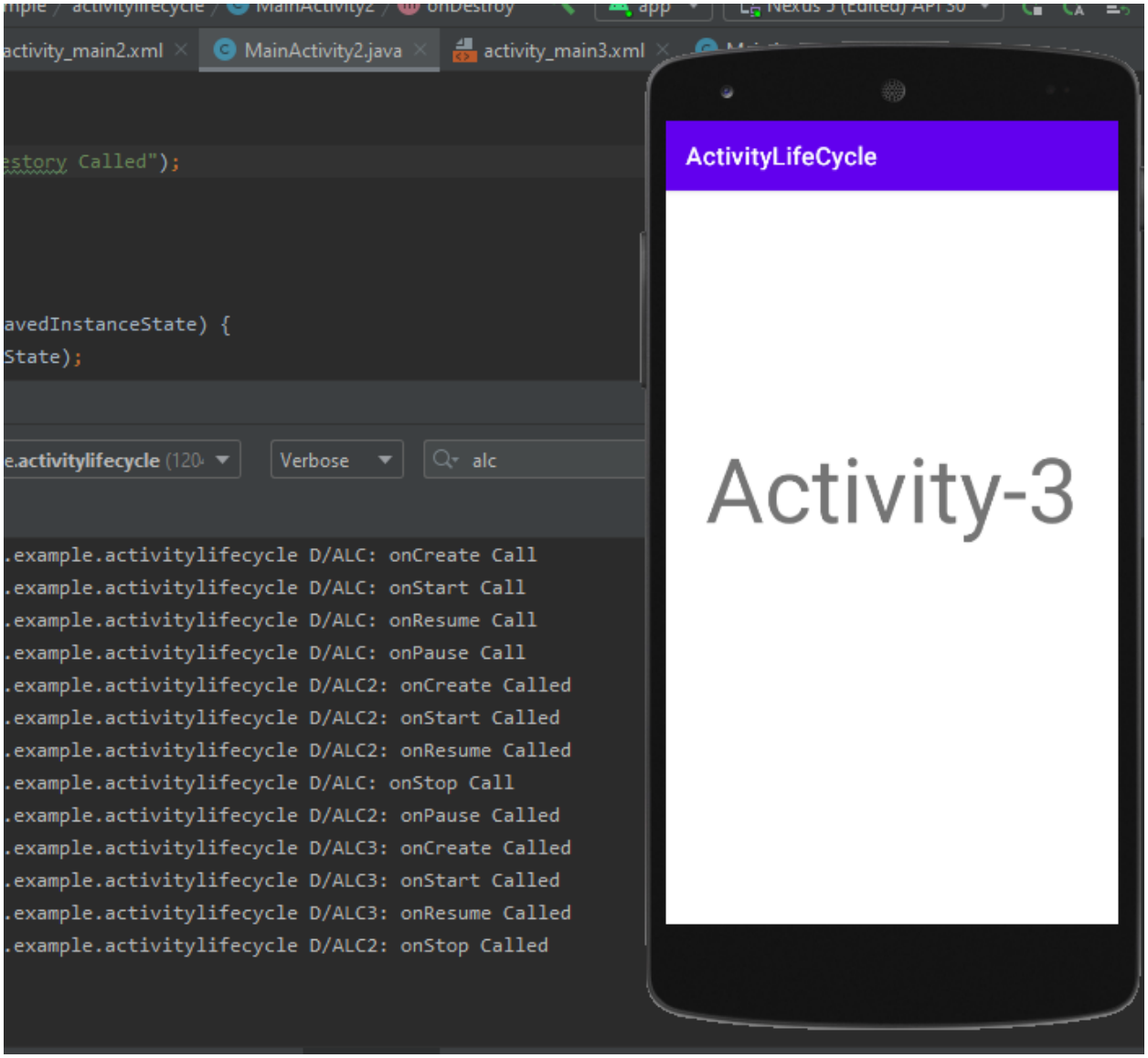
Repeating same steps for activity 3,navigate from main to 2nd activity then back to main and move to Activity-3. Then see the logcat.

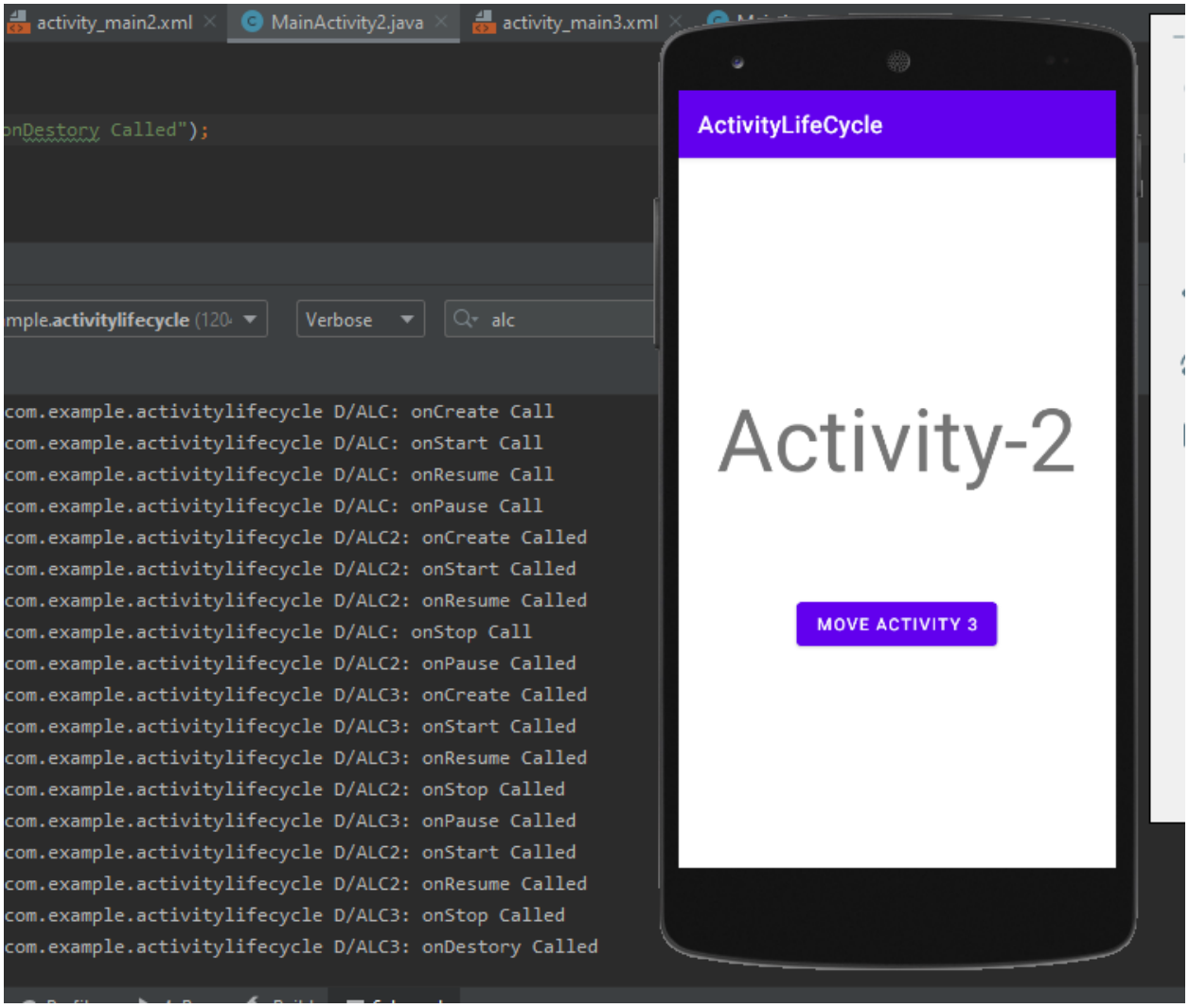


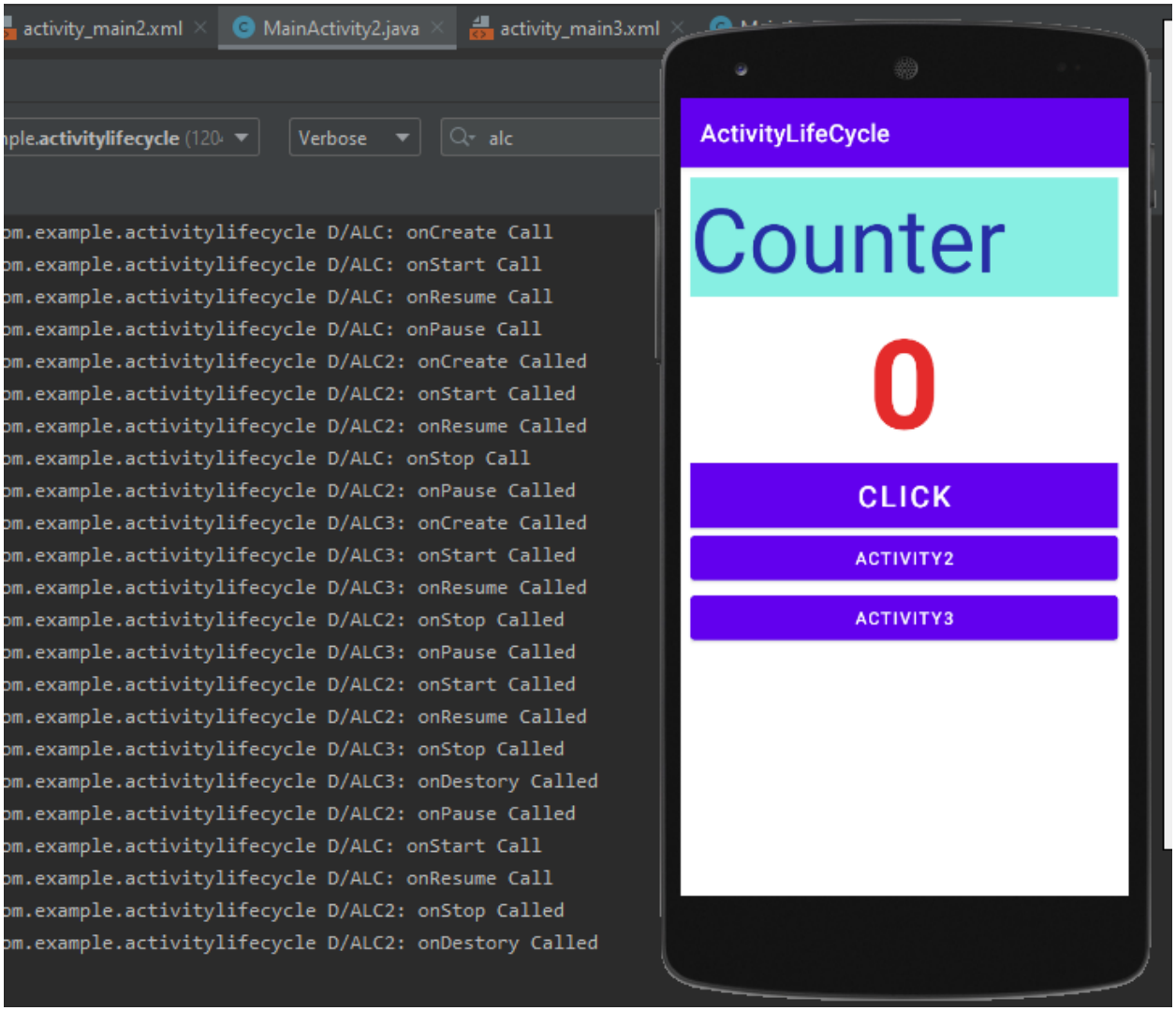
❖ **ALC of Main activity to other activity from other to another activity:**




activity_main2.xml
MainActivity2.java
activity_main3.xml
MainActivity3.java
"onResume Called");
"onDestory Called");
.example.activitylifecycle (120
Verbose
alc
45/com.example.activitylifecycle D/ALC: onCreate Call
45/com.example.activitylifecycle D/ALC: onStart Call
45/com.example.activitylifecycle D/ALC: onResume Call
45/com.example.activitylifecycle D/ALC: onPause Call
45/com.example.activitylifecycle D/ALC2: onCreate Called
45/com.example.activitylifecycle D/ALC2: onStart Called
45/com.example.activitylifecycle D/ALC2: onResume Called
45/com.example.activitylifecycle D/ALC: onStop Call
ActivityLifeCycle
Activity-2
MOVE ACTIVITY 3

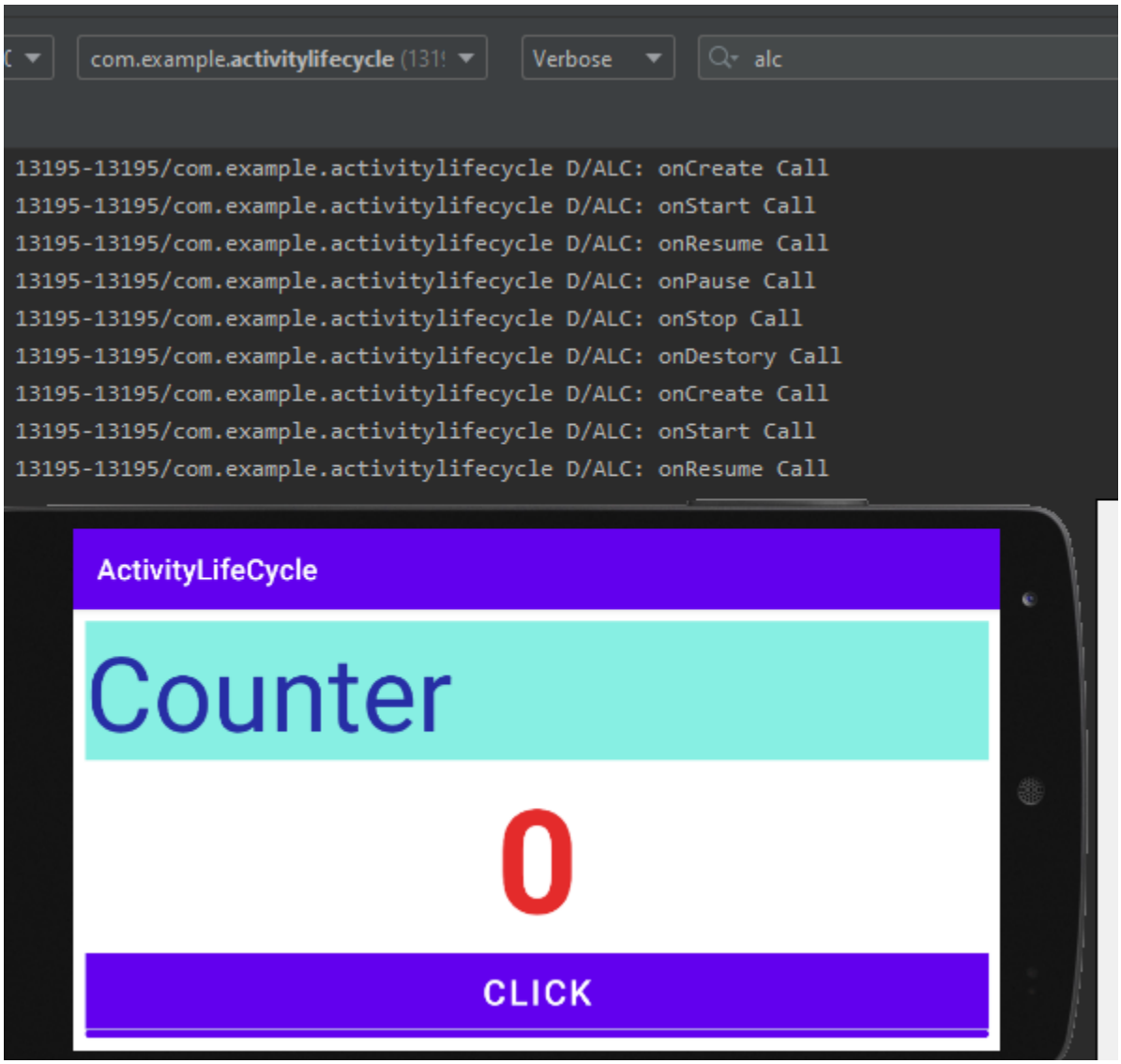

activity_main2.xml
MainActivity2.java
activity_main3.xml
estory Called");
avedInstanceState) {
State);
e.activitylifecycle (120
Verbose
alc
.example.activitylifecycle D/ALC: onCreate Call
.example.activitylifecycle D/ALC: onStart Call
.example.activitylifecycle D/ALC: onResume Call
.example.activitylifecycle D/ALC: onPause Call
.example.activitylifecycle D/ALC2: onCreate Called
.example.activitylifecycle D/ALC2: onStart Called
.example.activitylifecycle D/ALC2: onResume Called
.example.activitylifecycle D/ALC: onStop Call
.example.activitylifecycle D/ALC2: onPause Called
.example.activitylifecycle D/ALC3: onCreate Called
.example.activitylifecycle D/ALC3: onStart Called
.example.activitylifecycle D/ALC3: onResume Called
.example.activitylifecycle D/ALC2: onStop Called
ActivityLifeCycle
Activity-3


activity_main2.xml
MainActivity2.java
activity_main3.xml
onDestory Called");
mple.activitylifecycle (120
Verbose
alc
com.example.activitylifecycle D/ALC: onCreate Call
com.example.activitylifecycle D/ALC: onStart Call
com.example.activitylifecycle D/ALC: onResume Call
com.example.activitylifecycle D/ALC: onPause Call
com.example.activitylifecycle D/ALC2: onCreate Called
com.example.activitylifecycle D/ALC2: onStart Called
com.example.activitylifecycle D/ALC2: onResume Called
com.example.activitylifecycle D/ALC: onStop Call
com.example.activitylifecycle D/ALC2: onPause Called
com.example.activitylifecycle D/ALC3: onCreate Called
com.example.activitylifecycle D/ALC3: onStart Called
com.example.activitylifecycle D/ALC3: onResume Called
com.example.activitylifecycle D/ALC2: onStop Called
com.example.activitylifecycle D/ALC3: onPause Called
com.example.activitylifecycle D/ALC2: onStart Called
com.example.activitylifecycle D/ALC2: onResume Called
com.example.activitylifecycle D/ALC3: onStop Called
com.example.activitylifecycle D/ALC3: onDestory Called
ActivityLifeCycle
Activity-2
MOVE ACTIVITY 3


activity_main2.xml
MainActivity2.java
activity_main3.xml
ple.activitylifecycle (120
Verbose
alc
om.example.activitylifecycle D/ALC: onCreate Call
om.example.activitylifecycle D/ALC: onStart Call
om.example.activitylifecycle D/ALC: onResume Call
om.example.activitylifecycle D/ALC: onPause Call
om.example.activitylifecycle D/ALC2: onCreate Called
om.example.activitylifecycle D/ALC2: onStart Called
om.example.activitylifecycle D/ALC2: onResume Called
om.example.activitylifecycle D/ALC: onStop Call
om.example.activitylifecycle D/ALC2: onPause Called
om.example.activitylifecycle D/ALC3: onCreate Called
om.example.activitylifecycle D/ALC3: onStart Called
om.example.activitylifecycle D/ALC3: onResume Called
om.example.activitylifecycle D/ALC2: onStop Called
om.example.activitylifecycle D/ALC3: onPause Called
om.example.activitylifecycle D/ALC2: onStart Called
om.example.activitylifecycle D/ALC2: onResume Called
om.example.activitylifecycle D/ALC3: onStop Called
om.example.activitylifecycle D/ALC3: onDestory Called
om.example.activitylifecycle D/ALC2: onPause Called
om.example.activitylifecycle D/ALC: onStart Call
om.example.activitylifecycle D/ALC: onResume Call
om.example.activitylifecycle D/ALC2: onStop Called
om.example.activitylifecycle D/ALC2: onDestory Called
ActivityLifeCycle
Counter
0
CLICK
ACTIVITY2
ACTIVITY3

Reason =>Rotation-Action Problem:

- **On screen rotation onDestroy() is called so value of count doesn't get saved and after rotation activity starts again .**



Solution =>Rotation-Action Problem:

- **Save values before destroy**

Add the following method in the activity and save value of count in the bundle.So that value of count can be stored before call of onDestory() and reused later on.

```
@Override
protected void onSaveInstanceState(@NonNull Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putInt("value",count);
```

```
}
```

Put an if condition to check instance state and display value stored by bundle.

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

   if (savedInstanceState!=null)
{
    count=savedInstanceState.getInt("value");
    txtView.setText(String.valueOf(count) );
}

}
```

Now check the screen after rotation it will display the count exactly that was before rotation.

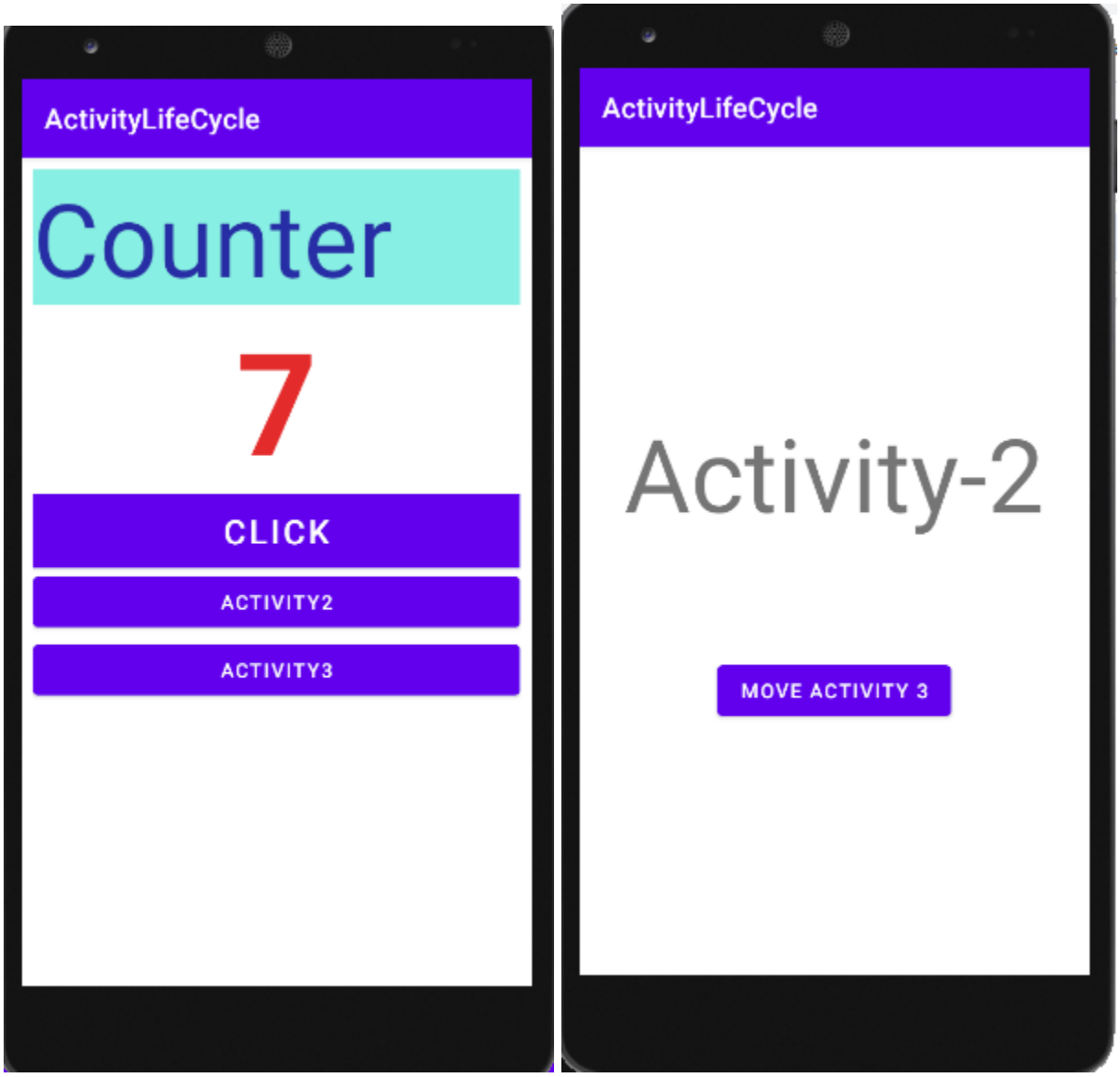
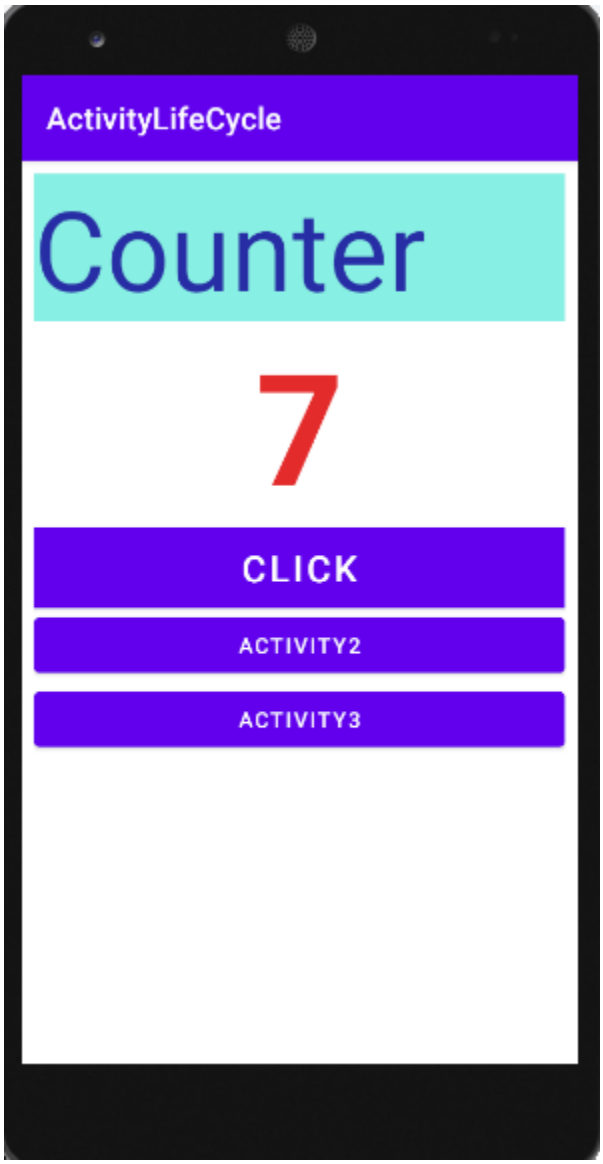
Checking Counter on Activity Navigation:

Now comment the display code in if condition.

```
if (savedInstanceState!=null)
{
    count=savedInstanceState.getInt( key: "value");
    //txtView.setText(String.valueOf(count) );
}
```

Run the emulator and see add come counts,then rotate and see behavoiur.

Now get back to original rotation format and then move to other activity.



Move back from second activity and see the behaviour. The count value will be the same as you saved.