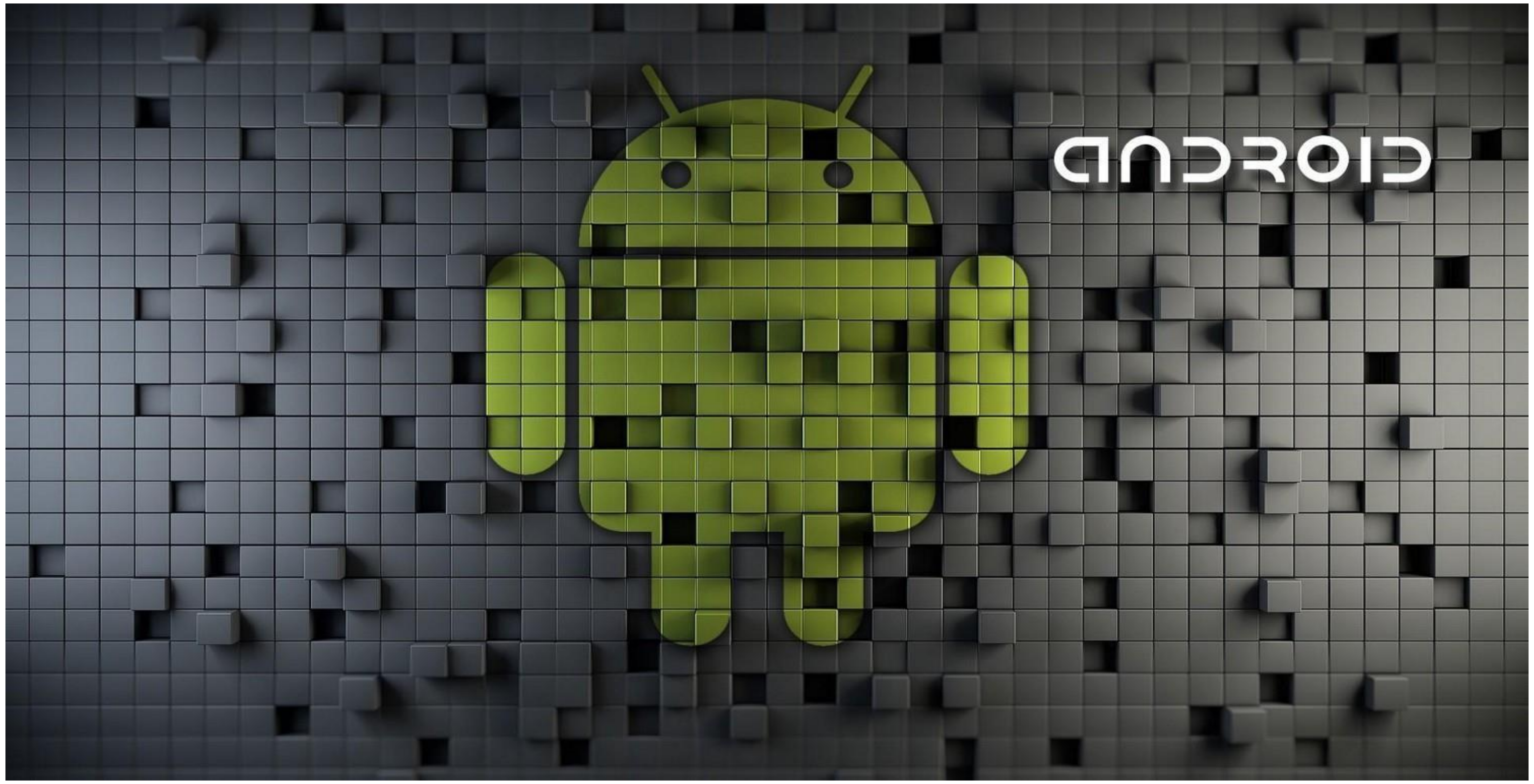
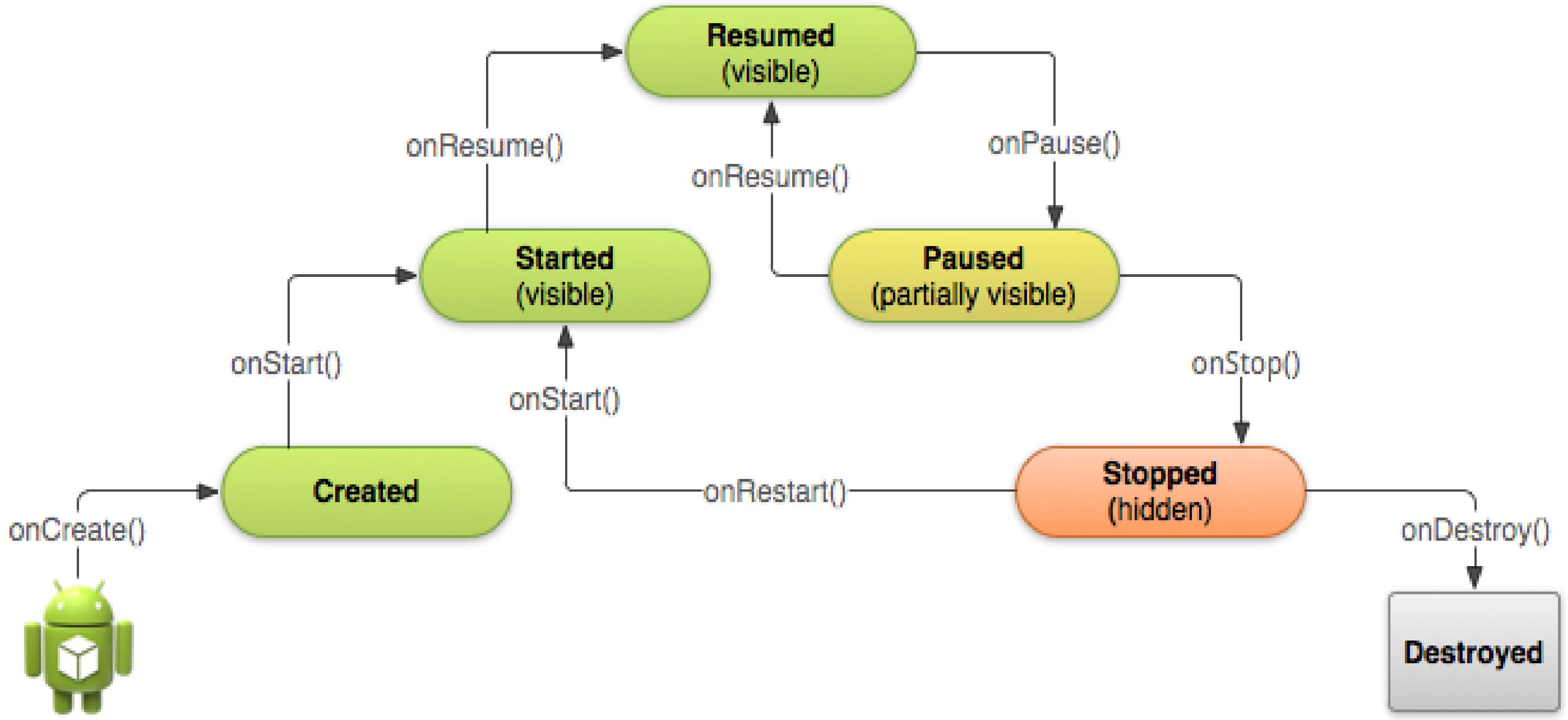


ACTIVITY LIFE CYCLE

ANDROID ACTIVITY (STATES)

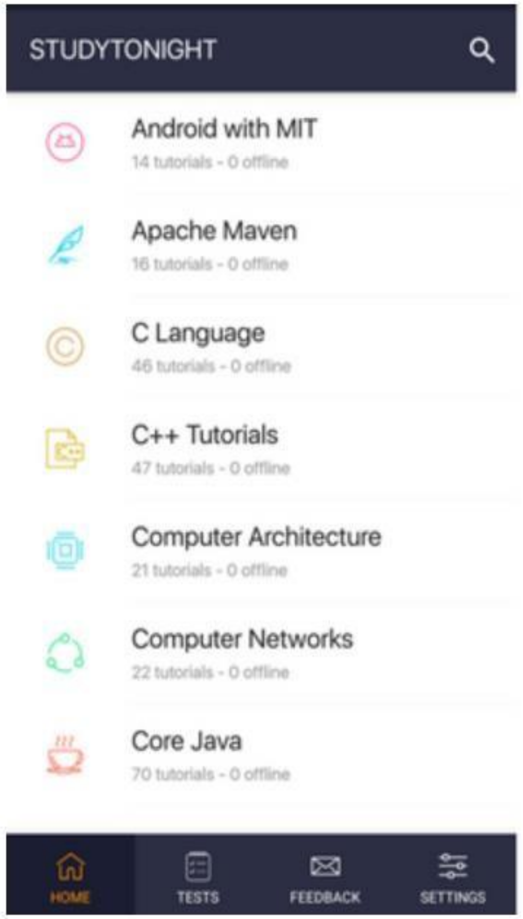
- An activity can have five states, which are :
  1. Started / Running (Active)
  2. Paused
  3. Resumed
  4. Stopped
  5. Destroyed



1. Running (Active) State

- When an Activity is in active state, it means it is active and running.
- It is visible to the user and the user is able to interact with it.
- Android Runtime treats the Activity in this state with the highest priority and never tries to kill it.

1. Running (Active) State ...


STUDYTONIGHT

Android with MIT
14 tutorials - 0 offline

Apache Maven
16 tutorials - 0 offline

C Language
46 tutorials - 0 offline

C++ Tutorials
47 tutorials - 0 offline

Computer Architecture
21 tutorials - 0 offline

Computer Networks
22 tutorials - 0 offline

Core Java
70 tutorials - 0 offline

Activity State:
Created, Started
or Resumed

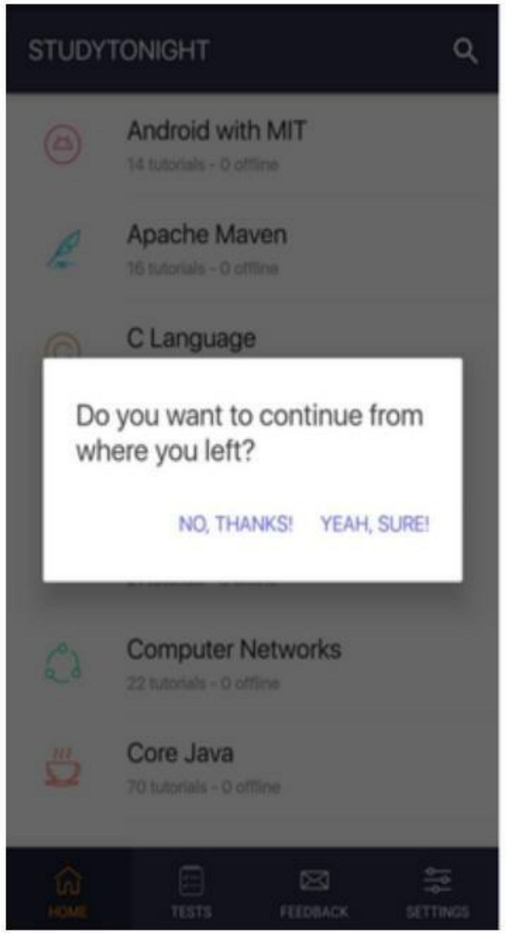
Process state is Foreground.

And the likelihood of killing the app is Least.

2. Paused State

- An activity being in this state means that the user can still see the Activity in the background such as behind a transparent window or a dialog box i.e it is partially visible.
- The user cannot interact with the Activity until he/she is done with the current view.
- Android Runtime usually does not kill an Activity in this state but may do so in an extreme case of resource crunch.

2. Paused State …


STUDYTONIGHT

Android with MIT
14 tutorials - 0 offline

Apache Maven
16 tutorials - 0 offline

C Language

Do you want to continue from where you left?

NO, THANKS! YEAH, SURE!

Computer Networks
22 tutorials - 0 offline

Core Java
70 tutorials - 0 offline

Activity State:
Paused

Process state is Background(lost focus).

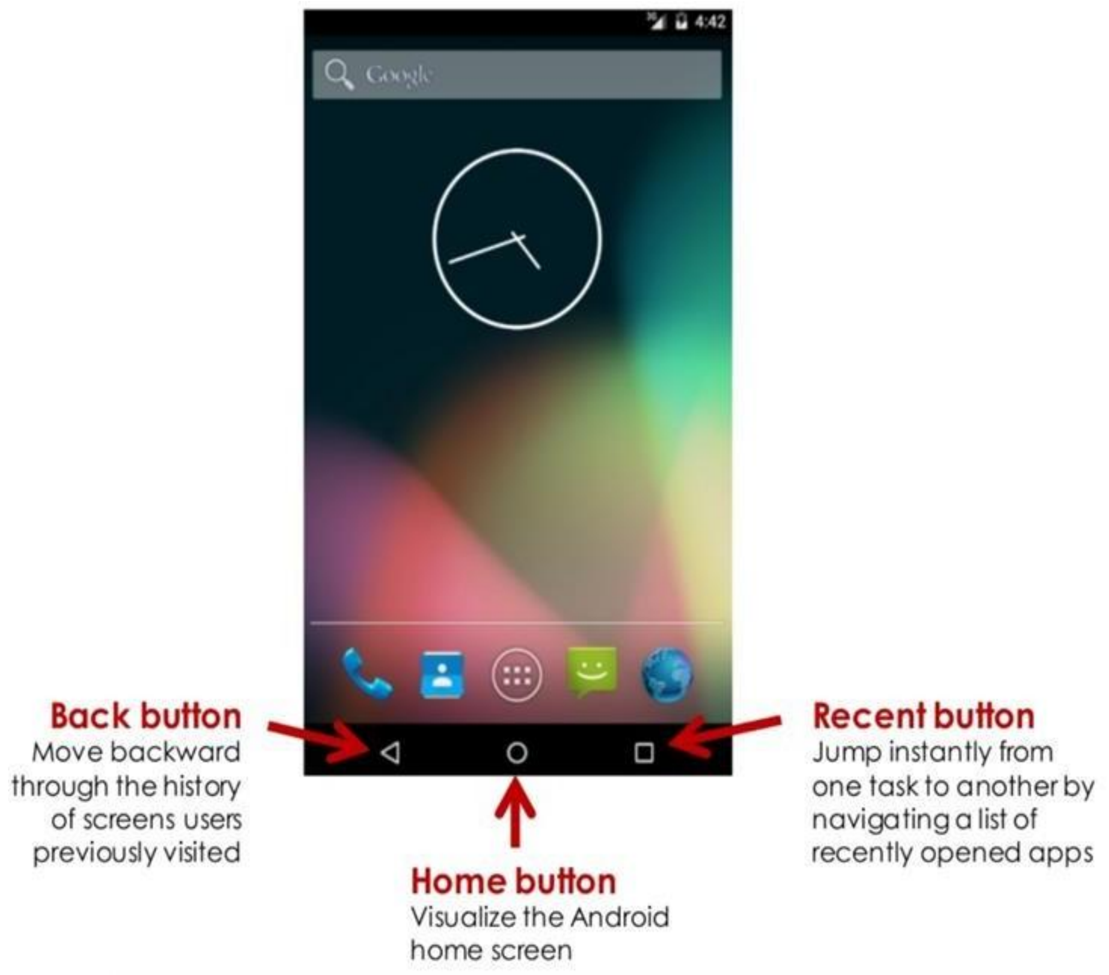
And the likelihood of killing the app is More.

3. Resumed State …

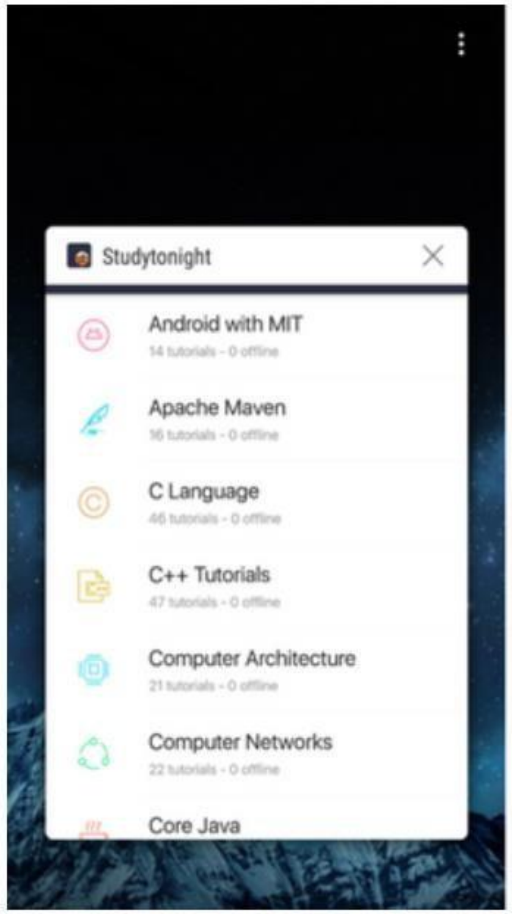
- It is when an activity goes from the paused state to the foreground that is an active state.

4. Stopped State

- When a new Activity is started on top of the current one or when a user hits the Home key, the activity is brought to Stopped state.
- The activity in this state is invisible, but it is not destroyed.
- Android Runtime may kill such an Activity in case of resource crunch.


Back button
Move backward through the history of screens users previously visited
Home button
Visualize the Android home screen
Recent button
Jump instantly from one task to another by navigating a list of recently opened apps
4:42
Google

4. Stopped State…



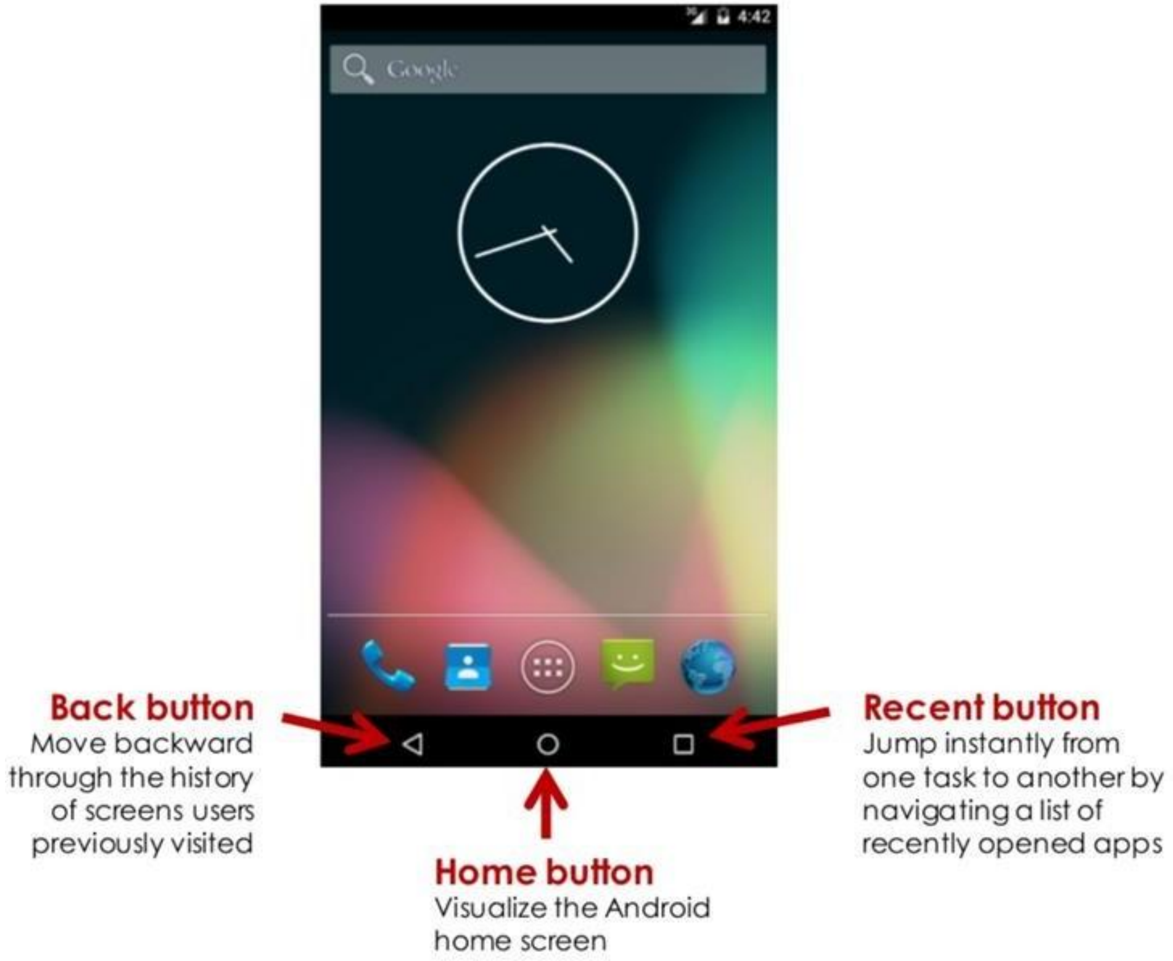
Activity State:
Stopped

Process state is Background(not visible).

And the likelihood of killing the app is Most.

5. Destroy State

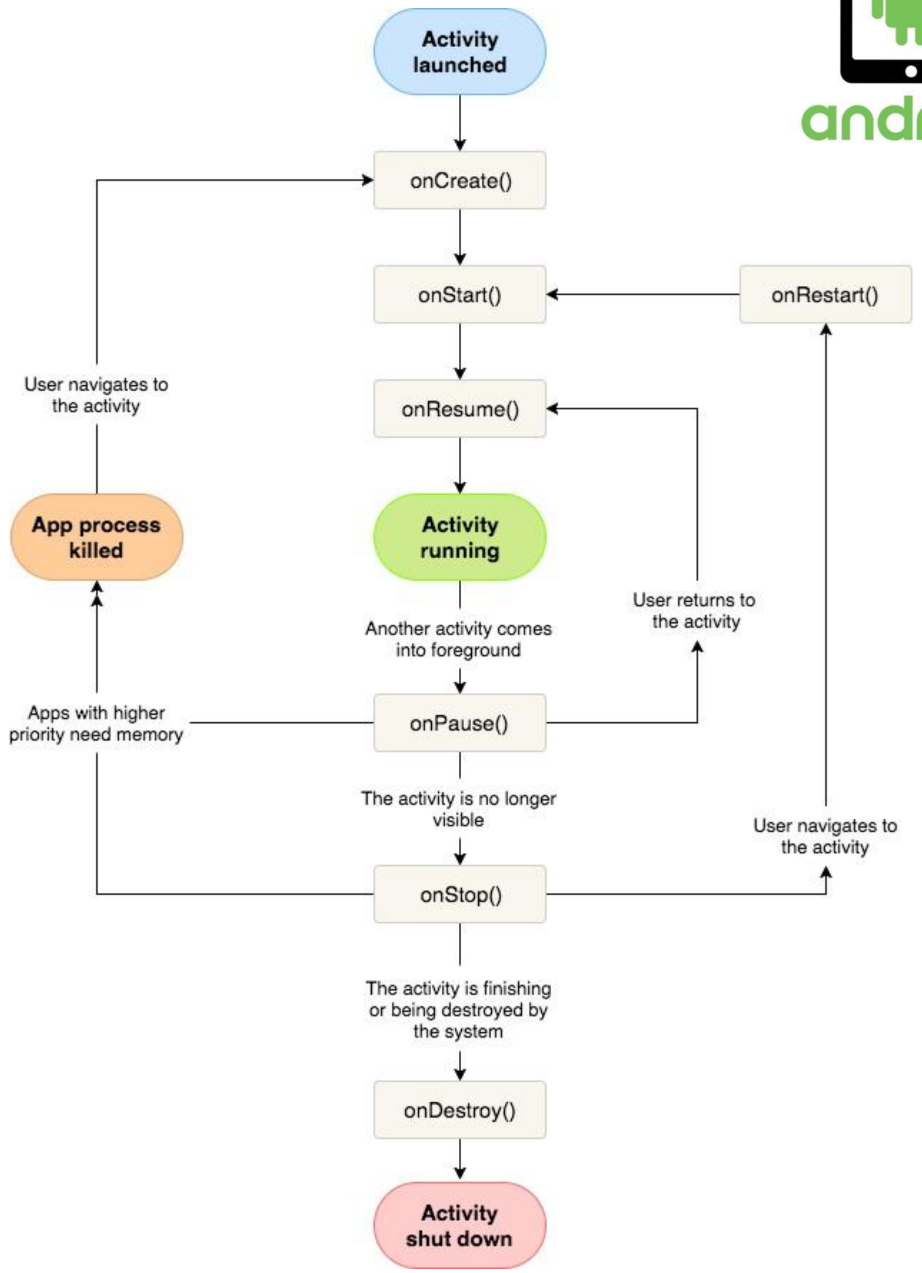
- When a user hits a Back key or Android Runtime decides to reclaim the memory allocated to an Activity i.e in the paused or stopped state, It goes into the Destroyed state.
- The Activity is out of the memory and it is invisible to the user.


Google
4:42
Back button
Move backward through the history of screens users previously visited
Recent button
Jump instantly from one task to another by navigating a list of recently opened apps
Home button
Visualize the Android home screen

Android Activity Methods

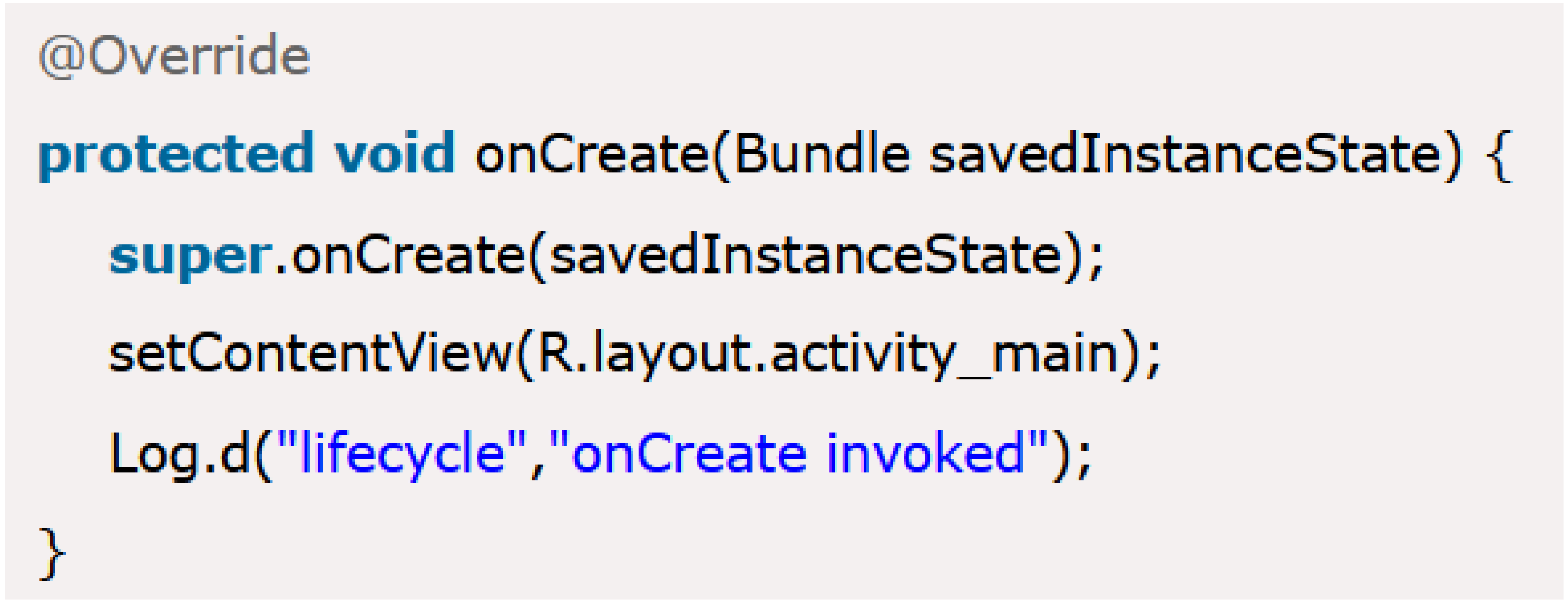
- Android activities go through **five states** during their entire **lifecycle**. These activities have callback methods() to describe each activity in each of the four stages. These methods need to be overridden by the implementing subclass. In Android, we have the following **7 callback methods** that activity uses to go through the four states:

1. **onCreate()**
2. **onStart()**
3. **onResume()**
4. **onPause()**
5. **onStop()**
6. **onRestart()**
7. **onDestroy()**


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
User returns to the activity
Another activity comes into foreground
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

onCreate()

- The Android oncreate() method is called at the very start when an activity is created. An activity is created as soon as an application is opened. This method is used in order to create an Activity.



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d("lifecycle","onCreate invoked");
}
```

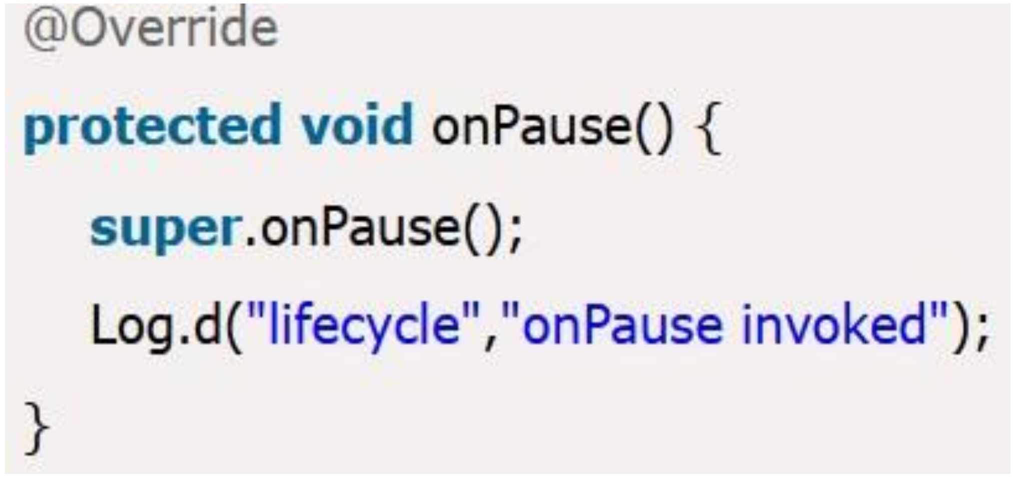
onStart()

- The Android onstart() method is invoked as soon as the activity becomes visible to the users. This method is to start an activity. The activity comes on the forescreen of the users when this method is invoked.

```
@Override
protected void onStart() {
    super.onStart();
    Log.d("lifecycle","onStart invoked");
}
```

onPause()

- The Android onPause() method is invoked when the activity doesn’t receive any user input and goes on hold. In the pause state, the activity is partially visible to the user. This is done when the user presses the back or home buttons. Once an activity is in the pause state, it can be followed by either onResume() or onStopped() callback method.



```
@Override
protected void onPause() {
    super.onPause();
    Log.d("lifecycle","onPause invoked");
}
```

onRestart()

- The Android onRestart() method is invoked when activity is about to start from the stop state. This method is to restart an activity that had been active some time back. When an activity restarts, it starts working from where it was paused.

```
@Override
protected void onRestart() {
    super.onRestart();
    Log.d("lifecycle","onRestart invoked");
}
```

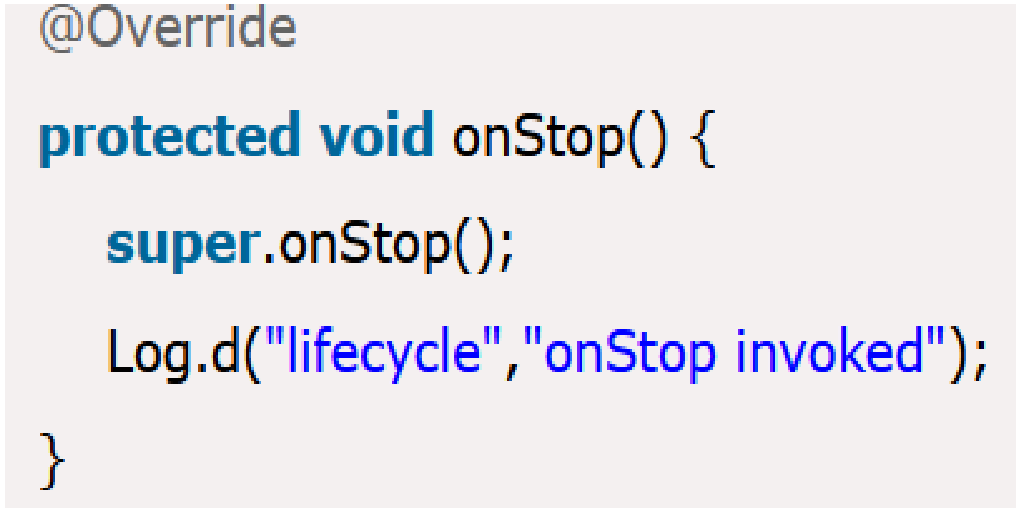
onResume()

- The Android onResume() method is invoked when the user starts interacting with the user. This callback method is followed by onPause(). Most of the functionalities of an application are implemented using onResume().

```
@Override
protected void onResume() {
    super.onResume();
    Log.d("lifecycle","onResume invoked");
}
```

onStop()

- The Android onStop() method is invoked when the activity is no longer visible to the user. The reason for this state is either activity is getting destroyed or another existing activity comes back to resume state.



```
@Override
protected void onStop() {
    super.onStop();
    Log.d("lifecycle","onStop invoked");
}
```

onDestroy()

- The Android onDestroy() is the method that is called when an activity finishes and the user stops using it. It is the final callback method received by activity, as after this it is destroyed.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d("lifecycle","onDestroy invoked");
}
```

Write Log Messages

- The Log class allows you to create log messages that appear in **logcat**. Generally, you should use the following log methods, listed in order from the highest to lowest priority (or, least to most verbose):

  - `Log.e(String, String)` (error)
  - `Log.w(String, String)` (warning)
  - `Log.i(String, String)` (information)
  - `Log.d(String, String)` (debug)
  - `Log.v(String, String)` (verbose)

JAVA

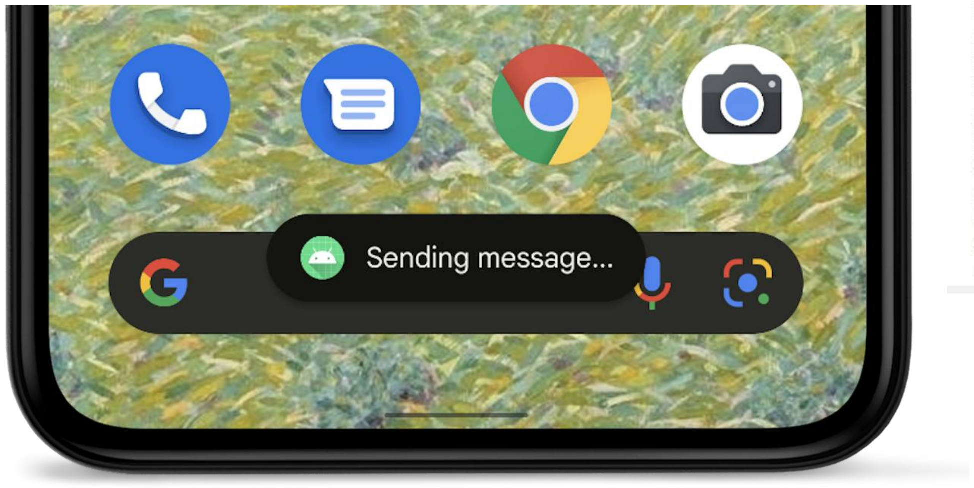
```
Log.d(tag, message);
```

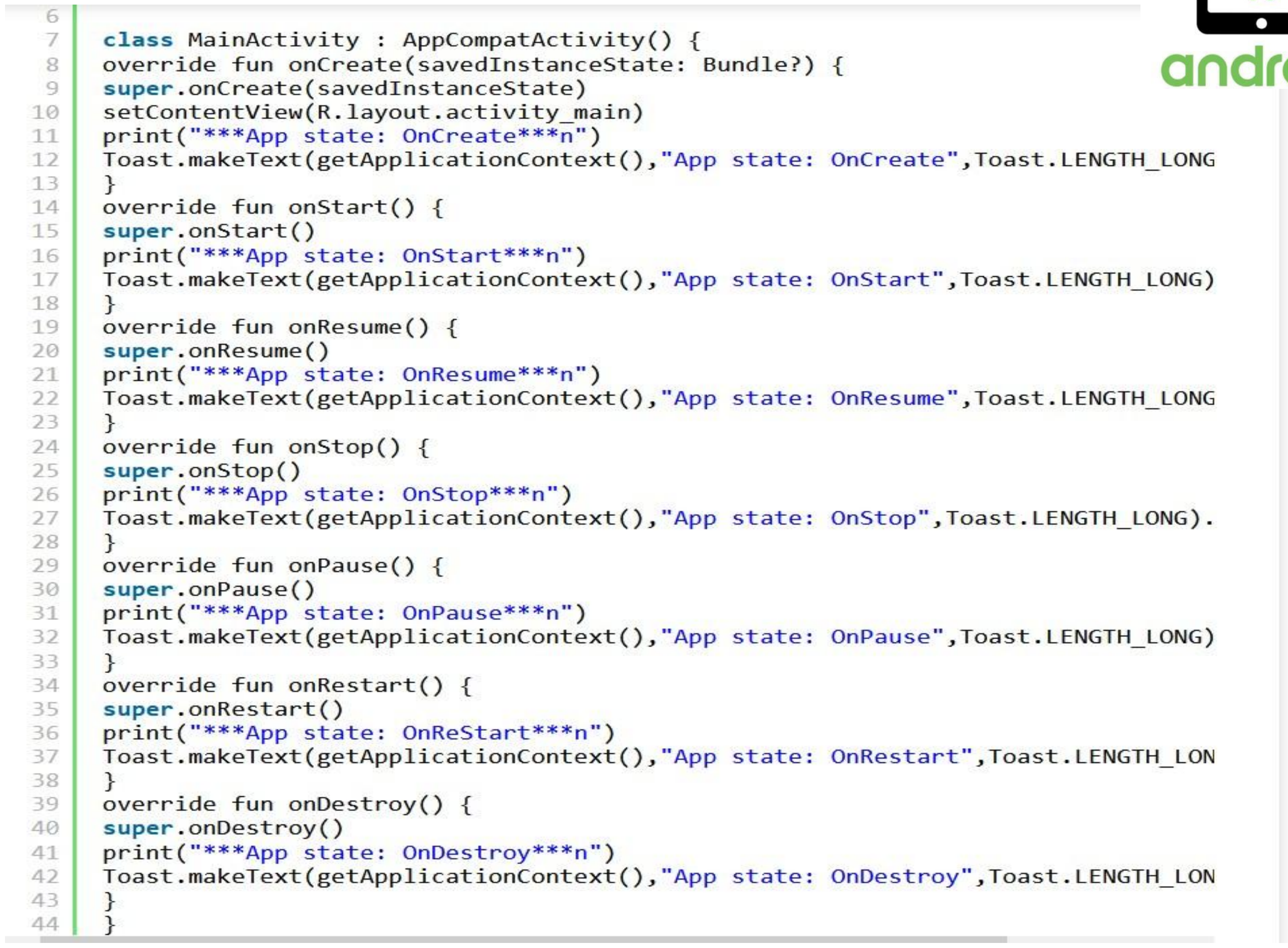
The priority is one of the following values:

- **V:** Verbose (lowest priority)
- **D:** Debug
- **I:** Info
- **W:** Warning
- **E:** Error
- **A:** Assert

TOASTS

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.





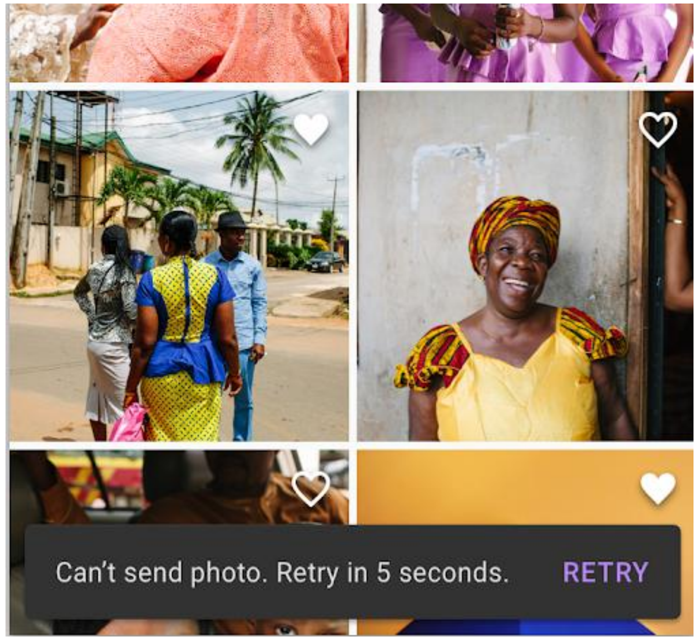
```
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
print("***App state: OnCreate***n")
Toast.makeText(getApplicationContext(),"App state: OnCreate",Toast.LENGTH_LONG
}
override fun onStart() {
super.onStart()
print("***App state: OnStart***n")
Toast.makeText(getApplicationContext(),"App state: OnStart",Toast.LENGTH_LONG)
}
override fun onResume() {
super.onResume()
print("***App state: OnResume***n")
Toast.makeText(getApplicationContext(),"App state: OnResume",Toast.LENGTH_LONG
}
override fun onStop() {
super.onStop()
print("***App state: OnStop***n")
Toast.makeText(getApplicationContext(),"App state: OnStop",Toast.LENGTH_LONG).
}
override fun onPause() {
super.onPause()
print("***App state: OnPause***n")
Toast.makeText(getApplicationContext(),"App state: OnPause",Toast.LENGTH_LONG)
}
override fun onRestart() {
super.onRestart()
print("***App state: OnReStart***n")
Toast.makeText(getApplicationContext(),"App state: OnRestart",Toast.LENGTH_LON
}
override fun onDestroy() {
super.onDestroy()
print("***App state: OnDestroy***n")
Toast.makeText(getApplicationContext(),"App state: OnDestroy",Toast.LENGTH_LON
}
}
```

[Toasts overview | Android Developers]

Alternatives to using Toasts

1. *SNACKBAR*

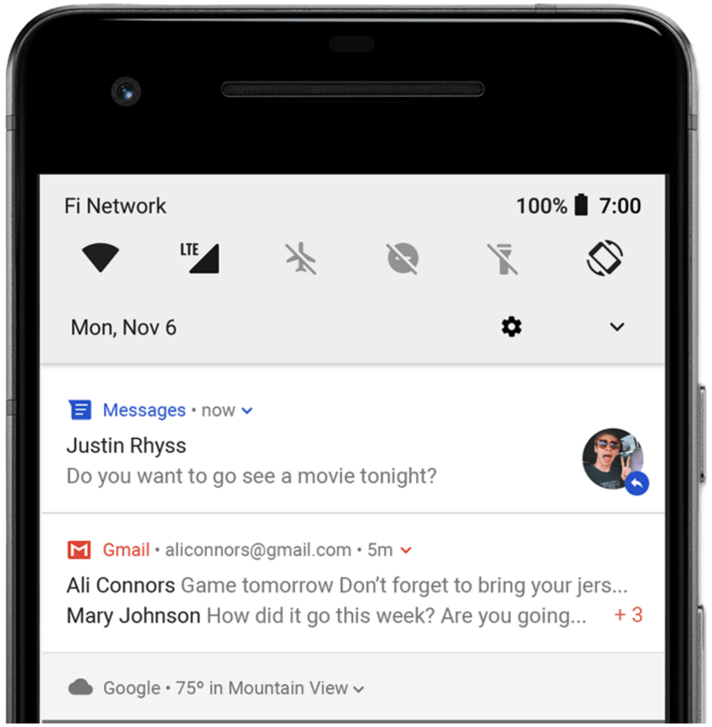
Snackbars provide brief messages about app processes at the bottom of the screen.



Can't send photo. Retry in 5 seconds. RETRY

2. *NOTIFICATIONS*

A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open your app or take an action directly from the notification.



Fi Network 100% 7:00

Mon, Nov 6

Messages • now
Justin Rhyss
Do you want to go see a movie tonight?

Gmail • aliconnors@gmail.com • 5m
Ali Connors Game tomorrow Don't forget to bring your jers...
Mary Johnson How did it go this week? Are you going... + 3

Google • 75° in Mountain View

Real Time Situation

- When you open the app it will go through below states:

  onCreate() –> onStart() –> onResume()

- When you press the back button and exit the app

  onPaused() — > onStop() –> onDestory()

- When you press the home button

  onPaused() –> onStop()

- After pressing the home button, again when you open the app from a recent task list

  onRestart() –> onStart() –> onResume()

Real Time Situation …

- After dismissing the dialog or back button from the dialog

    onResume()
- If a phone is ringing and user is using the app

  onPause() –> onResume()
- After the call ends

  onResume()
- When your phone screen is off

    onPaused() –> onStop()
- When your phone screen is turned back on

    onRestart() –> onStart() –> onResume()

Screen Orientation (Horizontal / Vertical)

- Some device configurations change while the application is running, such as when the device changes its orientation. Such a change restarts the activity by calling all its methods again. So when the device orientation changes, first the Activity will disappear for a millisecond when the onPause(), OnStop, and onDestroy() methods are called.
- After a few milliseconds, the activity will be restarted when the onCreate(), onStart() and onResume() methods are called.

- https://medium.com/hootsuite-engineering/handling-orientation-changes-on-android-41a6b62cb43f

References

- https://developer.android.com/guide/topics/manifest/manifest-intro
- https://www.studytonight.com/android/activity-in-android
- https://data-flair.training/blogs/android-activity/

Toasts overview | Android Developers