ACTIVITY LIFE CYCLE

```
public class SupportActivity extends Activity implements
LifecycleOwner, Component {
```



```
public class FragmentActivity extends SupportActivity
implements ViewModelStoreOwner,
OnRequestPermissionsResultCallback,
RequestPermissionsRequestCodeValidator {
```



```
public class AppCompatActivity extends FragmentActivity implements
AppCompatCallback, SupportParentable, DelegateProvider {
```



```
public class MainActivity extends AppCompatActivity {
```

➢ **MainActivity is the sub class of Activity Class**

AppcompactActivity class:

- If material design implementation is needed
- Target android devices below API 21

FragmentActivity class:

- If nested fragments needed
- If design features don't require material design for lower API levels

Activity class:

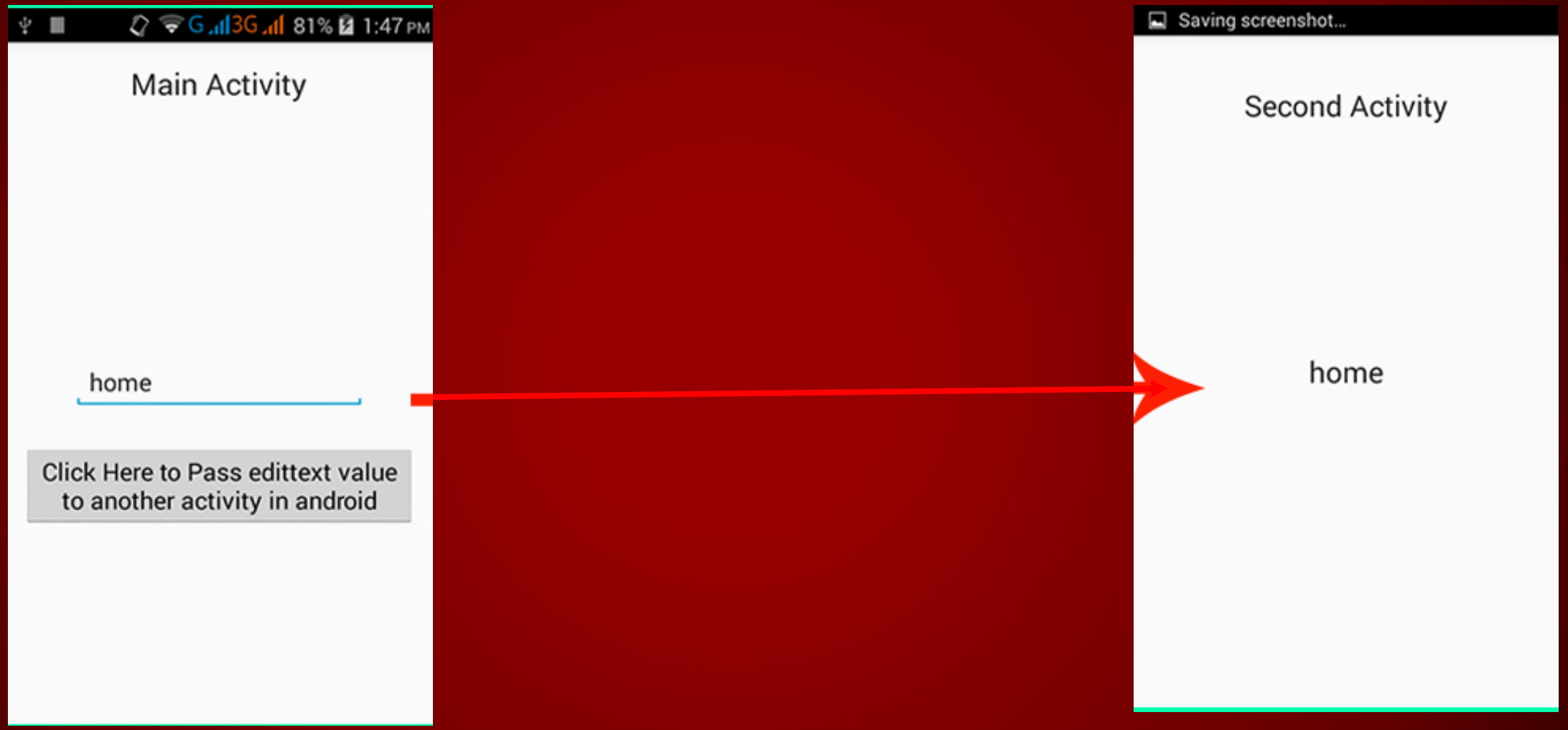
- None of the above needed

❖ All the features above are essential for professional applications

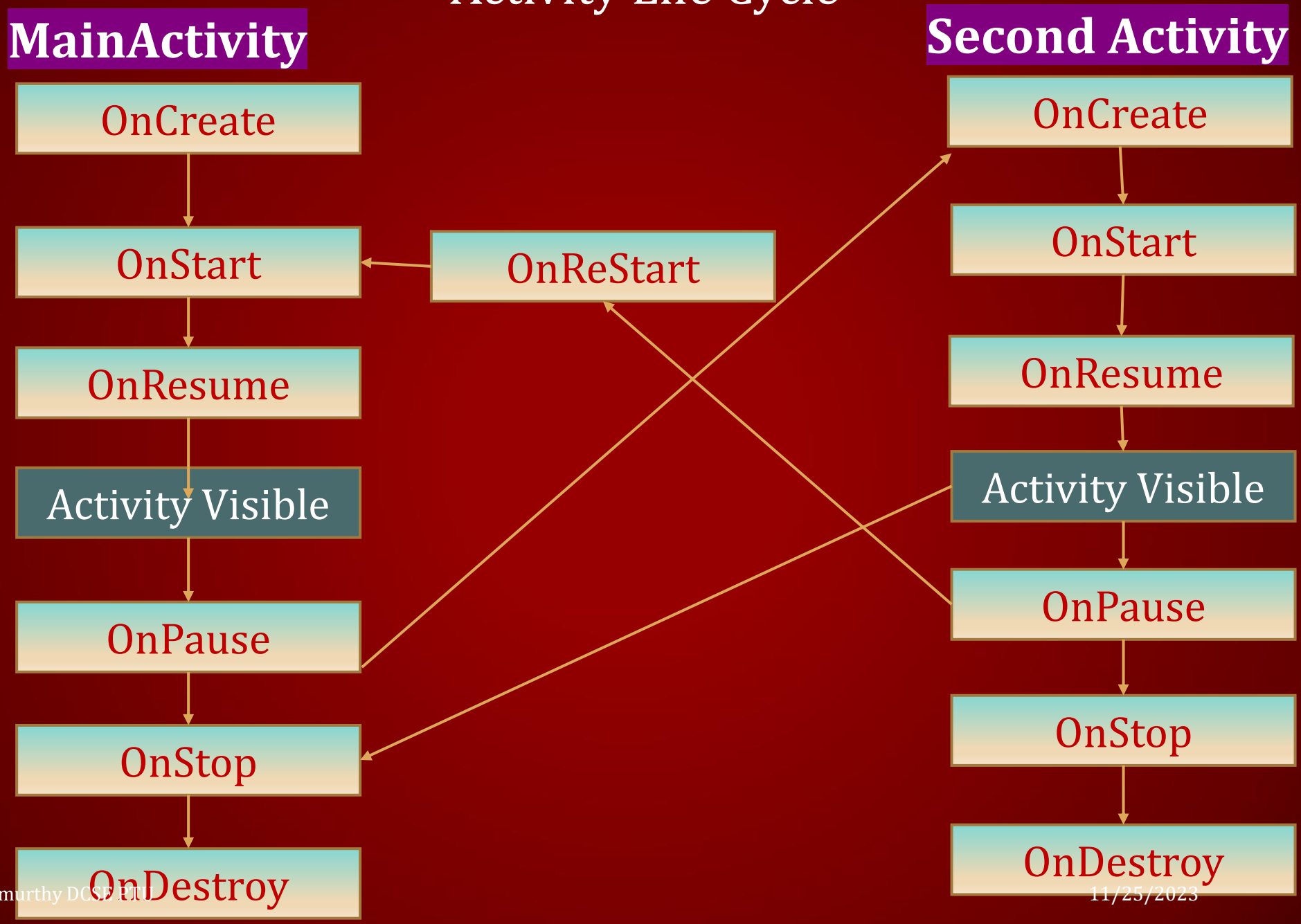
Activity Life Cycle Methods

- OnCreate
- onStart
- onResume
- onRestart
- onPause
- onStop
- onDestroy

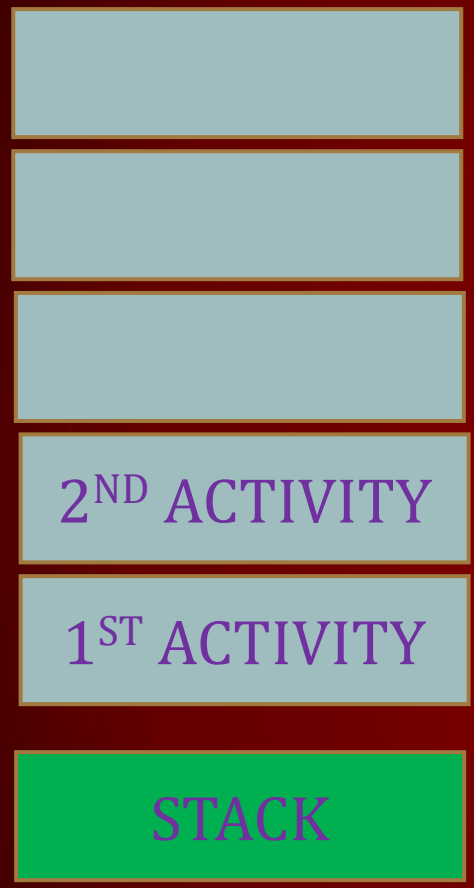
Activity Life Cycle

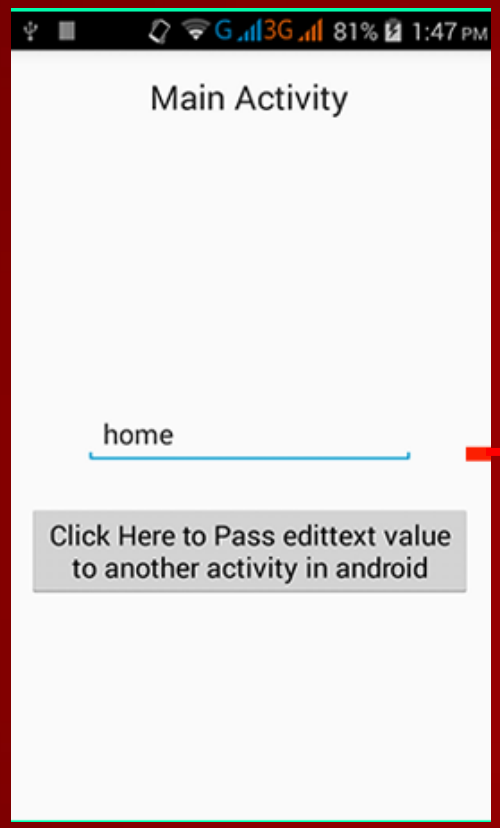


Activity Life Cycle



STACK AND ACTIVITY





Activity Life Cycle

onCreate()	Whenever an Activity starts running, the first method to get executed is onCreate(). This method is executed only once during the lifetime. If we have any instance variables in the Activity, the initialization of those variables can be done in this method. After onCreate() method, the onStart() method is executed.
onStart()	During the execution of onStart() method, the Activity is not yet rendered on screen but is about to become visible to the user. In this method, we can perform any operation related to UI components.
onResume()	When the Activity finally gets rendered on the screen, onResume() method is invoked. At this point, the Activity is in the active state and is interacting with the user.
onPause()	If the activity loses its focus and is only partially visible to the user, it enters the paused state. During this transition, the onPause() method is invoked. In the onPause() method, we may commit database transactions or perform light-weight processing before the Activity goes to the background.

Activity Life Cycle

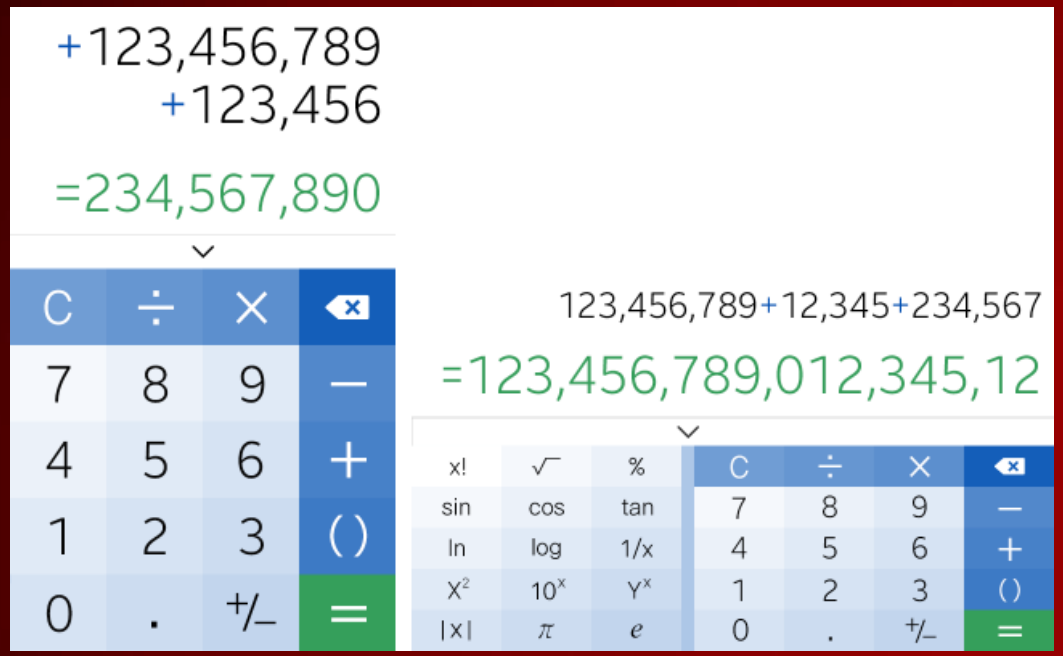
onStop()	From the active state, if we hit the Home key, the Activity goes to the background and the Home Screen of the device is made visible. During this event, the Activity enters the stopped state. Both onPause() and onStop() methods are executed.
onDestroy()	When an activity is destroyed by a user or Android system, onDestroy() function is called.

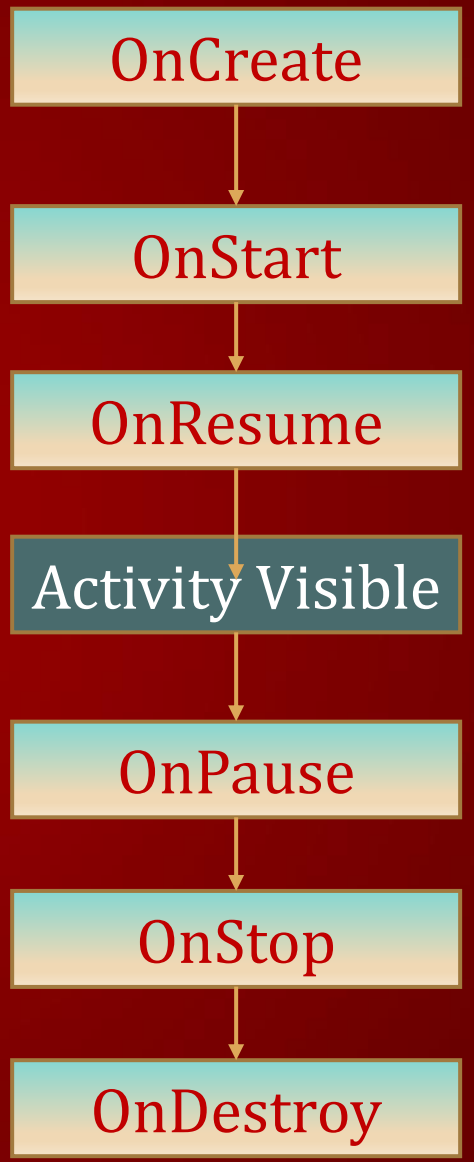
Screen Orientation and Activity Life cycle

Screen Orientation and Activity Life cycle

- On rotation of Screen
  - Portrait to landscape or vice versa
    - Activity is destroyed
    - Activity is recreated fresh in requested orientation

Consequences of Screen Orientation



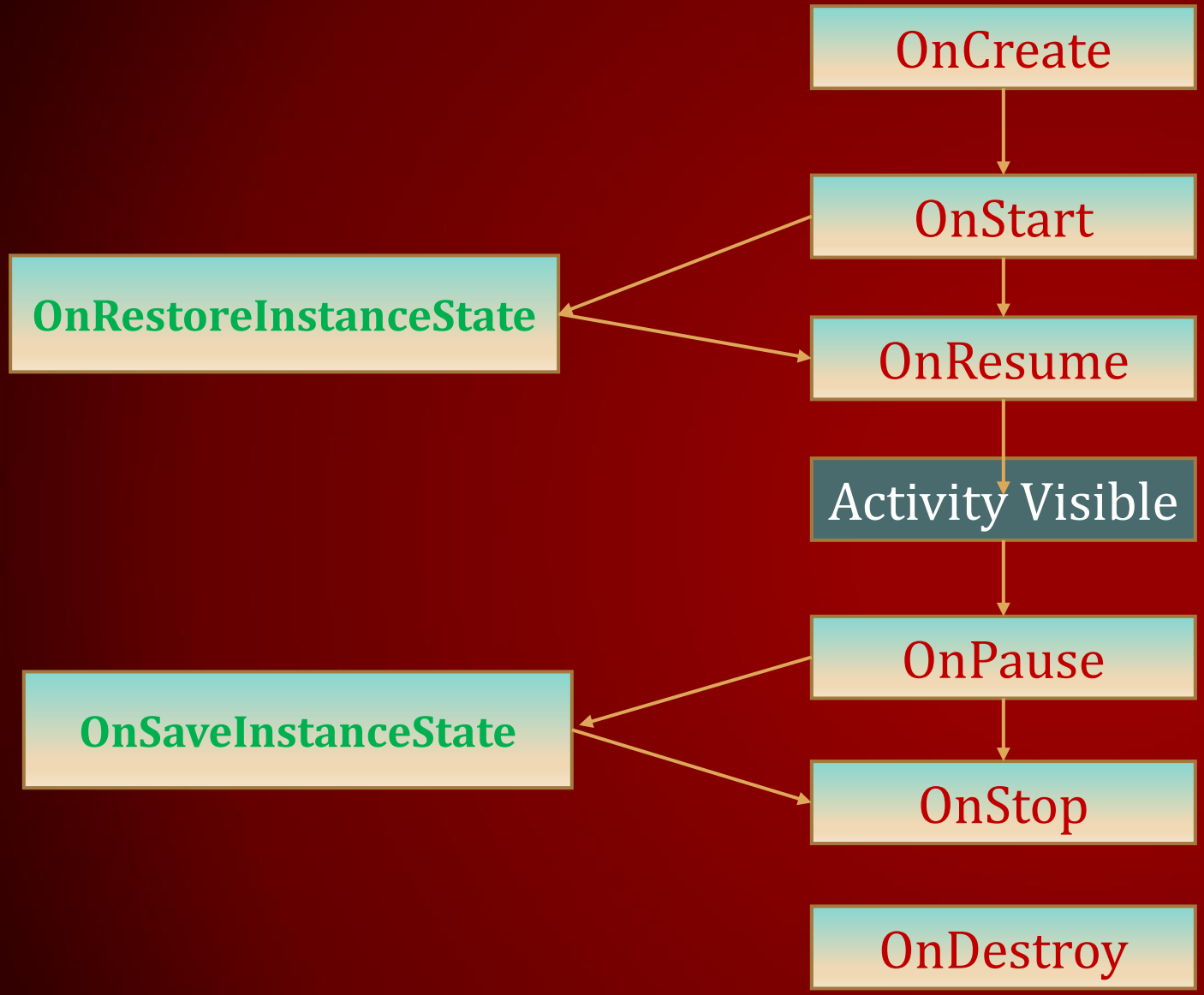


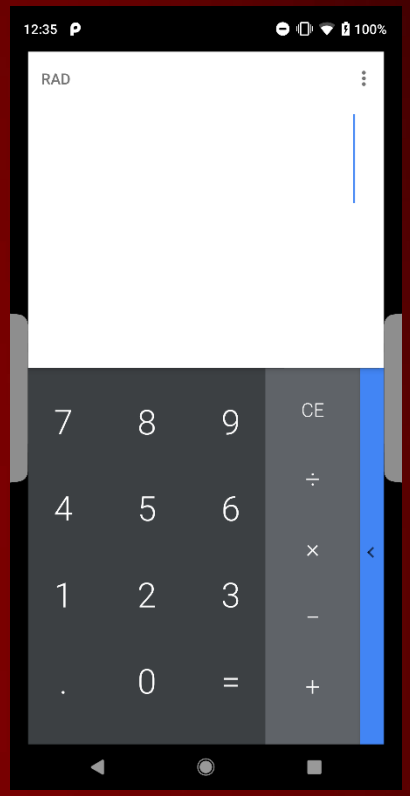
- Preserve the state of previous screen orientation
- User should not know that the activity is destroyed

Solutions to Handle Screen Orientation

1. Restoring the Activity State
   - Activity is destroyed and recreated
   - Use onRestoreInstanceState and onSaveInstanceState
2. Handling configuration change yourself
   - Activity is not destroyed
   - Use onConfigurationChanged

Method 1- Solution to Handle Screen Orientation





Method 2- Solution to Handle Screen Orientation

- Handling Configuration Change

- Declare android:configchanged attribute in Manifest file under <activity>
- Override onConfigurationChanged() in Activity class.

Summary- Solutions to Handle Screen Orientation

- Method1
  - Is generally preferred and recommended
- Against GuideLines
  - Not Recommended
  - Should be used only when restoring activity back is expensive and slow