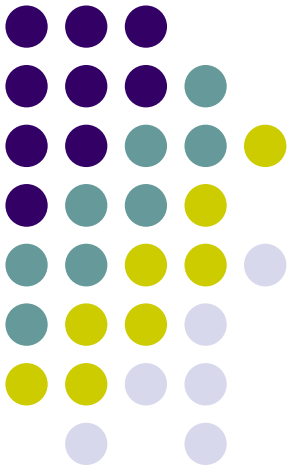
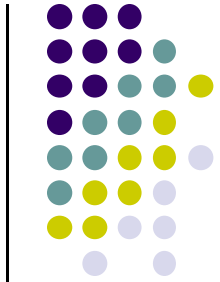
CS 403X Mobile and Ubiquitous Computing

Lecture 6: Android Activity Lifecycle

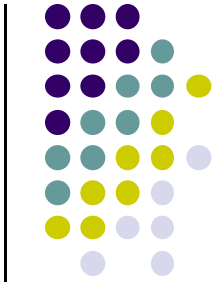
Emmanuel Agu





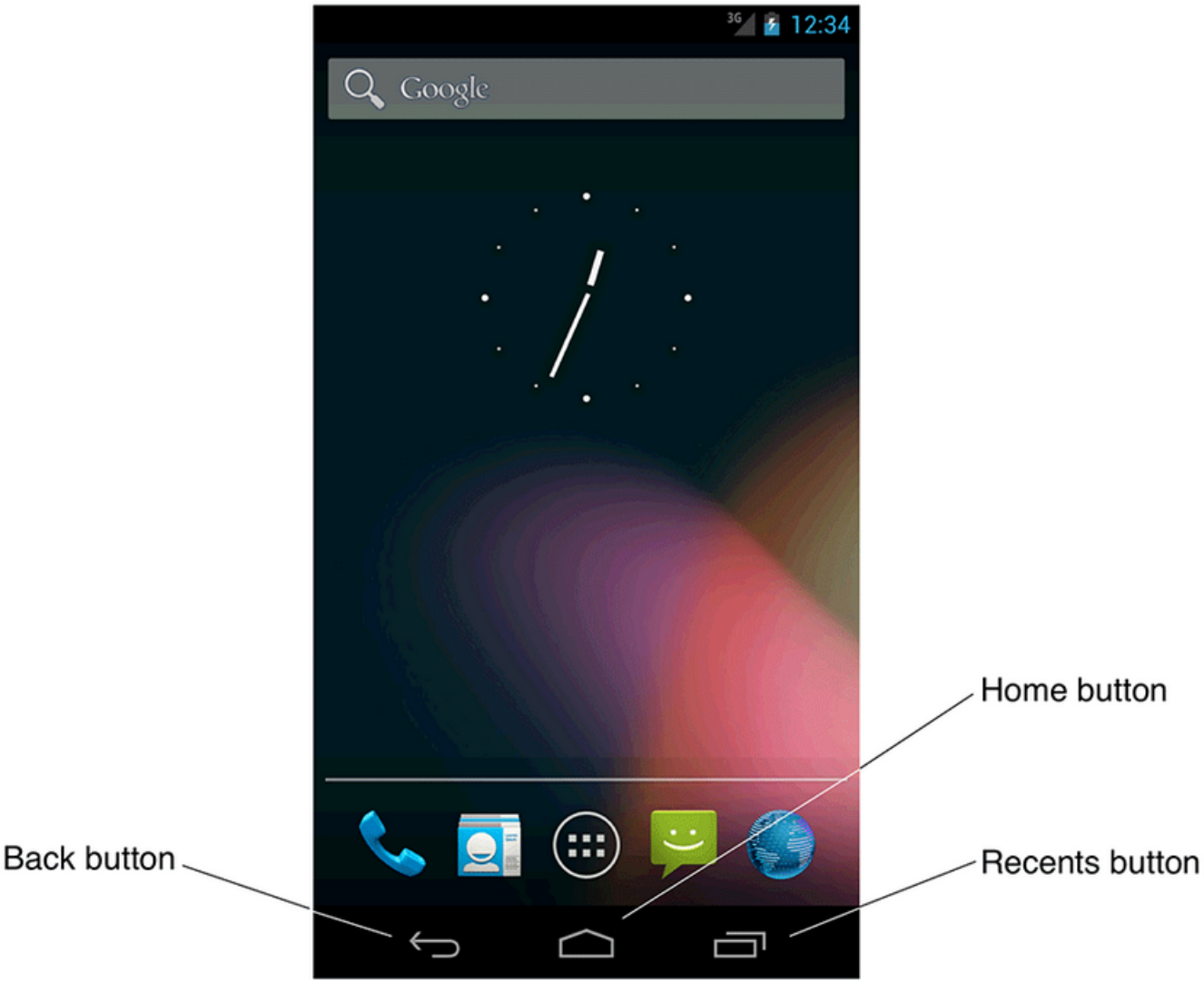
Android's Process Model

Android's Process Model

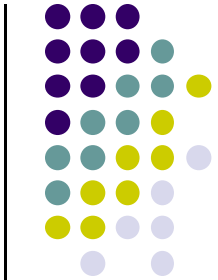


- When user launches an app, Android forks a copy of a process called **zygote** that receives
  - Copy of **Virtual Machine (Dalvik)**
  - Copy of **Android framework classes (e.g. Activity, Button)**
  - Copy of **user's app classes** loaded from their APK file
  - Any objects created by app or framework classes

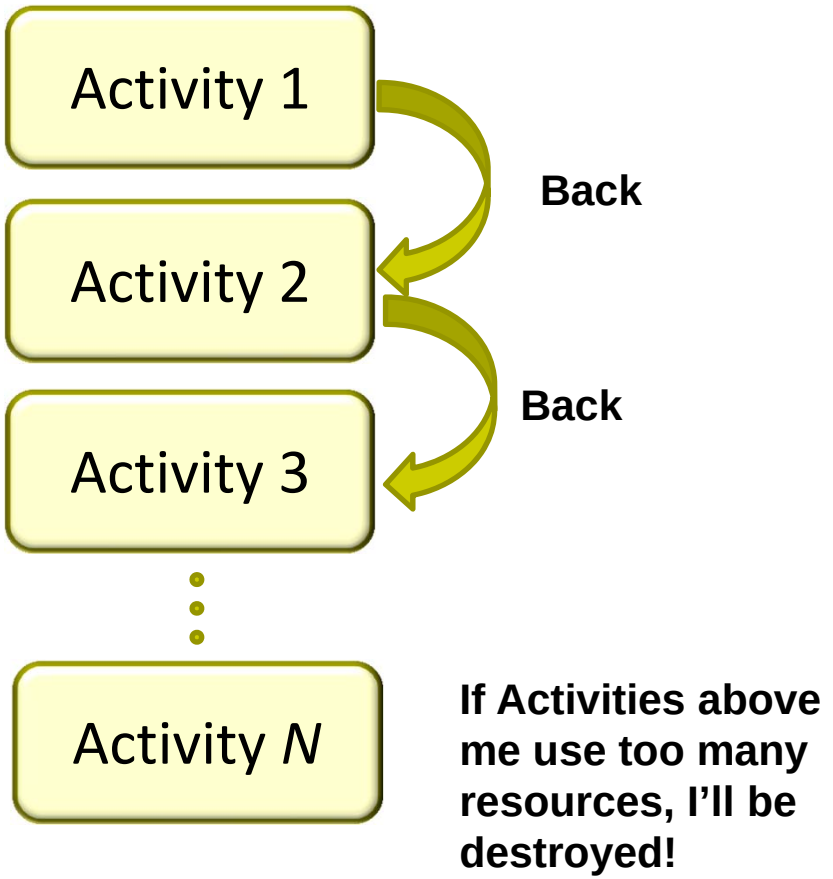
Recall: Home, Back and Recents Button

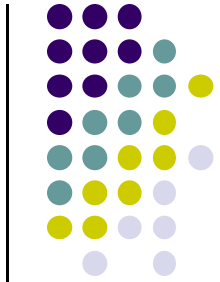


Android Activity Stack (Back vs Home Button)



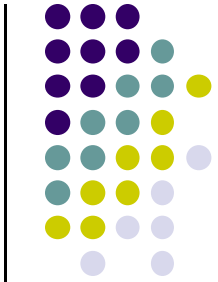
- Android maintains activity stack
- While an app is running,
  - Pressing **Back** button **destroys the app's activity** and returns app to whatever user was doing previously (e.g. HOME screen)
  - If **Home** button is pressed, activity is kept around for some time, **NOT destroyed** immediately





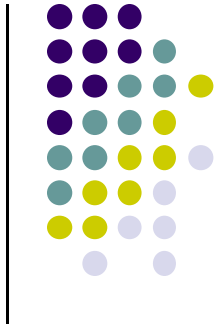
Android Activity LifeCycle

Starting Activities



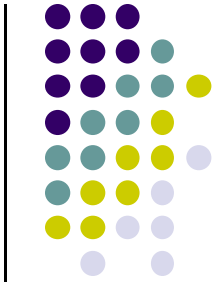
- Android applications don't start with a call to main(String[])
- Instead callbacks invoked corresponding to app state.
- Examples:
  - When activity is created, its **onCreate( )** method invoked
  - When activity is paused, its **onPause( )** method invoked
- callback methods also invoked to destroy Activity /app

Activity Callbacks



- onCreate() ← Already saw this (initially called)
- onStart()
- onResume()
- onPause()
- onStop()
- onRestart()
- onDestroy()

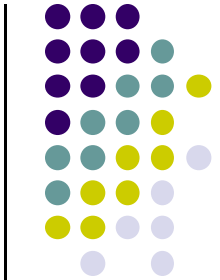
Understanding Android Lifecycle



- Many **disruptive** things could happen while app is running
  - Incoming call or text message, user switches to another app, etc
- Well designed app should NOT:
  - Crash if interrupted, or user switches to other app
  - Consume valuable system resources when inactive
  - Lose the user's state/progress (e.g state of chess game app) if they leave your app and return to it at a later time.
  - Crash or lose the user's progress when the screen rotates between landscape and portrait orientation.
    - E.g. Youtube video should continue at correct point after rotation
- To handle these situations, appropriate callback methods must be invoked appropriately

http://developer.android.com/training/basics/activity-lifecycle/starting.html

OnCreate( )



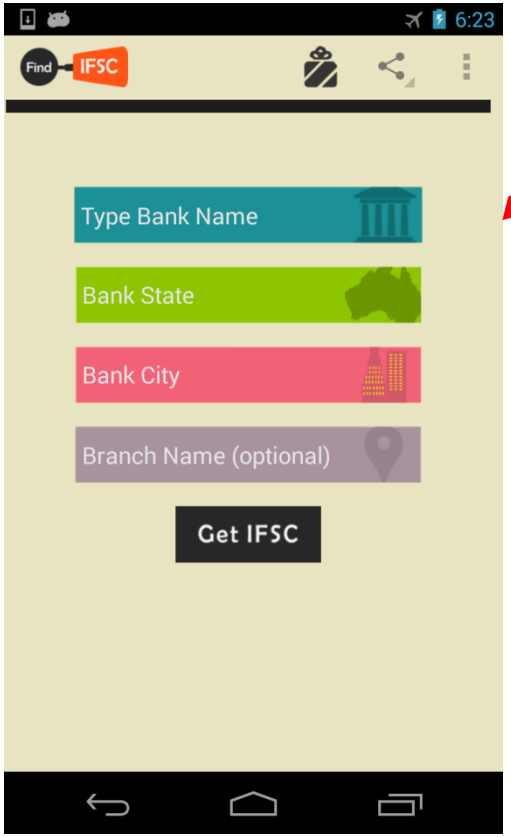
- Initializes activity once created
- Operations typically performed in onCreate() method:
  - Inflate widgets and put them on screen
    - (e.g. using layout files with setContentView( ))
  - Getting references to inflated widgets (using findViewbyId( ))
  - Setting widget listeners to handle user interaction
- Example

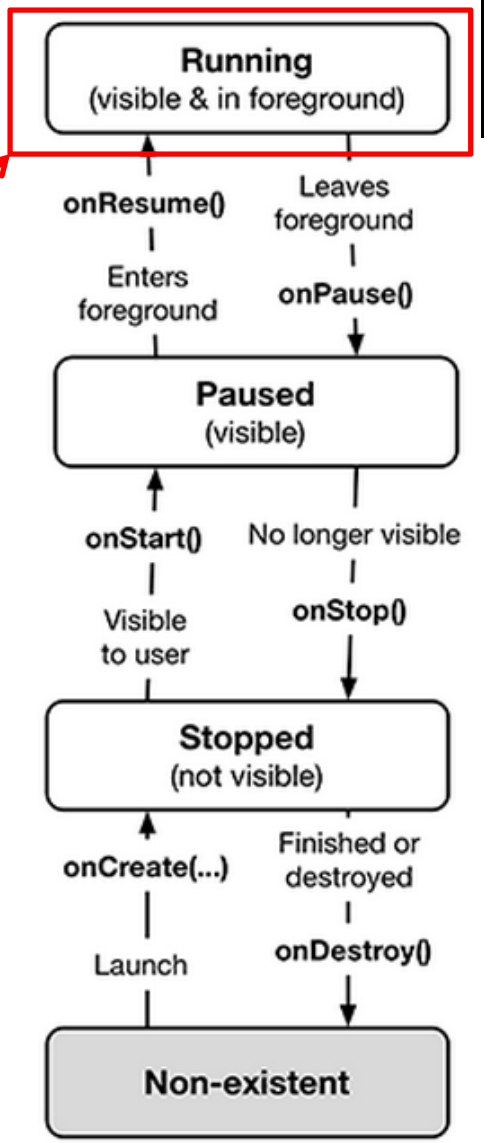
```
public class QuizActivity extends Activity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);

        mTrueButton = (Button)findViewById(R.id.true_button);
        mFalseButton = (Button)findViewById(R.id.false_button);
    }
```

- **Note:** Android OS calls apps' onCreate( ) method, NOT the app

Activity State Diagram: Running App

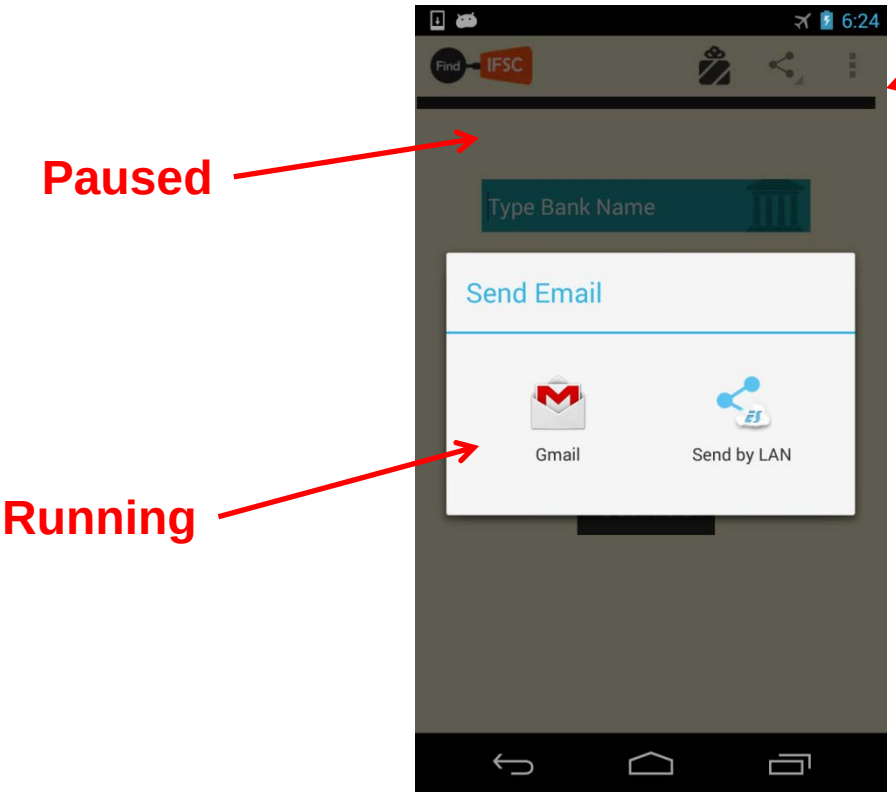
- A running app is one that user is currently using or interacting with
  - App is visible and in foreground

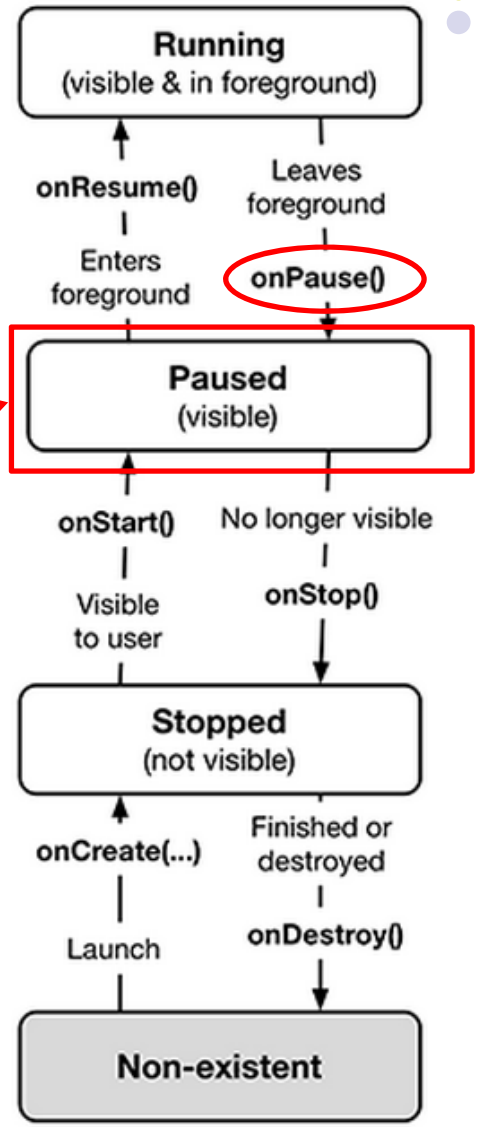




Activity State Diagram: Paused App

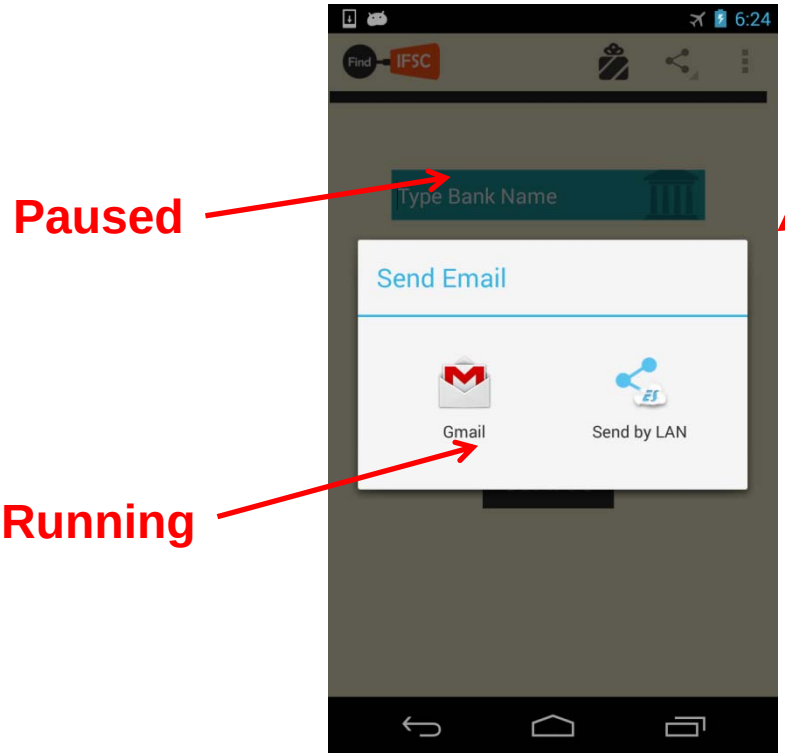
- An app is **paused** if it is **visible but no longer in foreground**
- E.g. blocked by a pop-up dialog box
- App's **onPause( )** method is called to transition from running to paused state

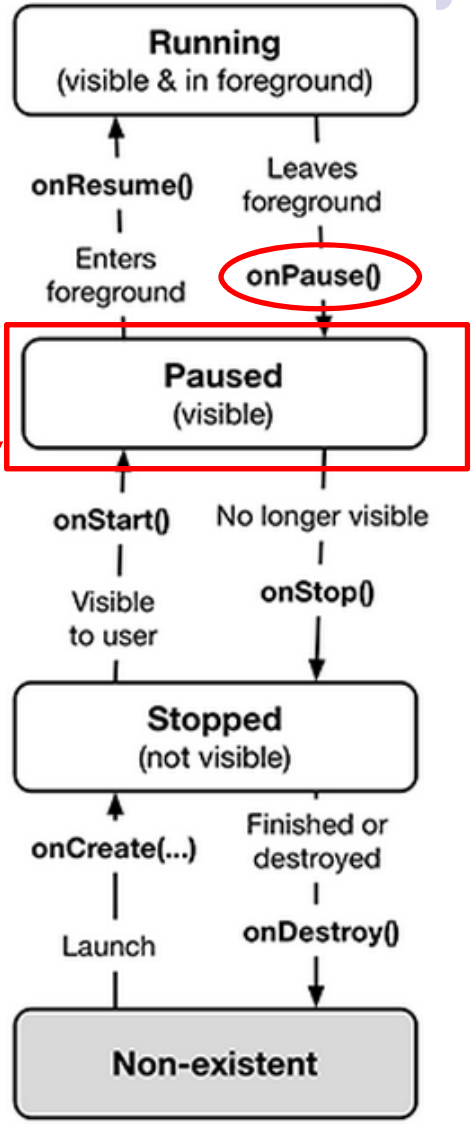




Activity State Diagram: onPause( ) Method

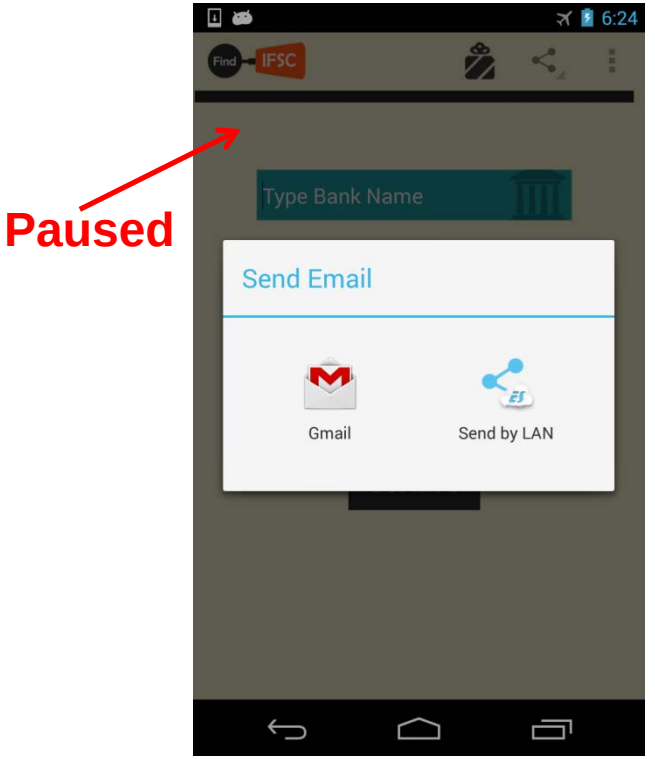
- Typical actions taken in onPause( ) method
  - Stop animations and CPU intensive tasks
  - Stop listening for GPS, broadcast information
  - Release handles to sensors (e.g GPS, camera)
  - Stop audio and video if appropriate

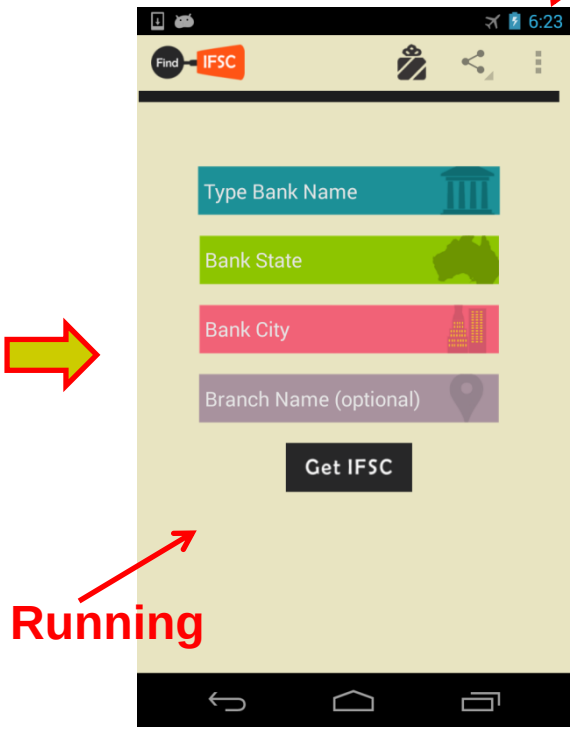


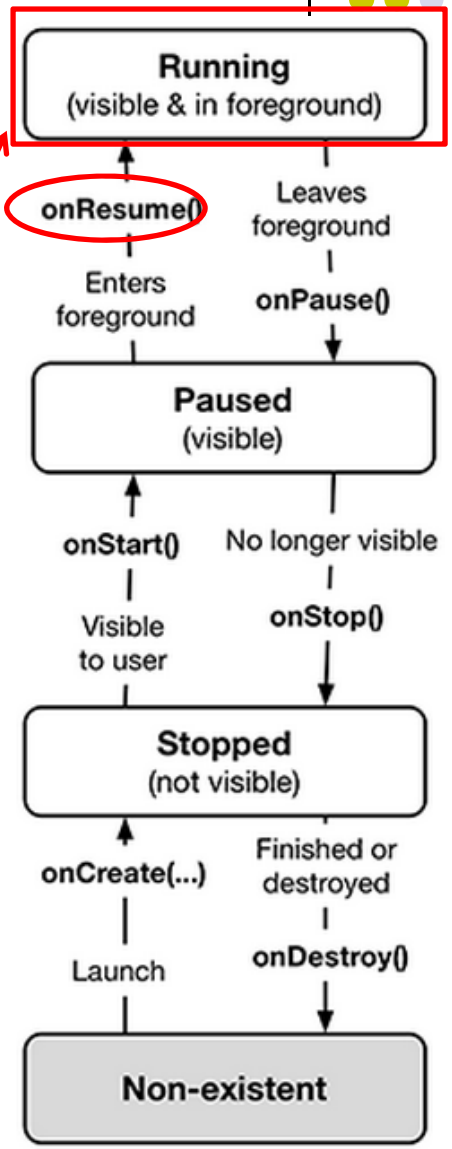


Activity State Diagram: Resuming Paused App

- A **paused** app resumes **running** if it becomes fully visible and in foreground
  - E.g. pop-up dialog box blocking it goes away
- App's **onResume( )** method is called to transition from **paused** to **running** state

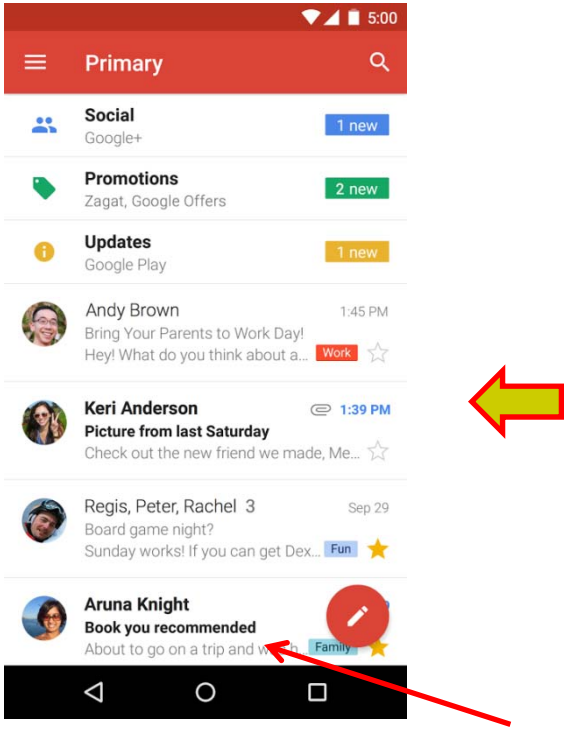




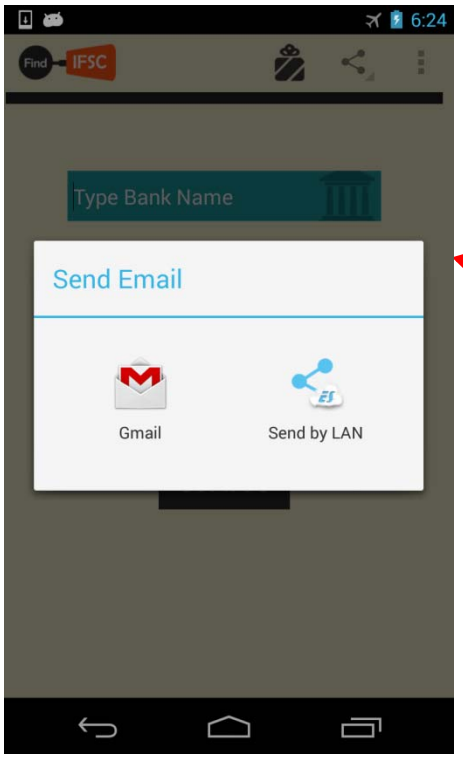


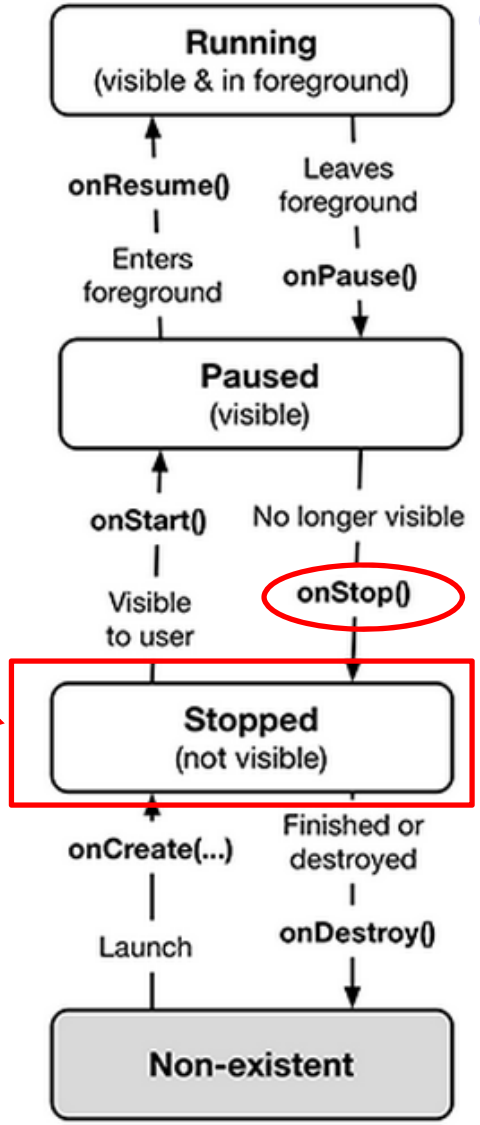
Activity State Diagram: Stopped App

- An app is **stopped** if it **no longer visible and no longer in foreground**
- E.g. user starts using another app
- App's **onStop( )** method is called to transition from paused to stopped state



Running



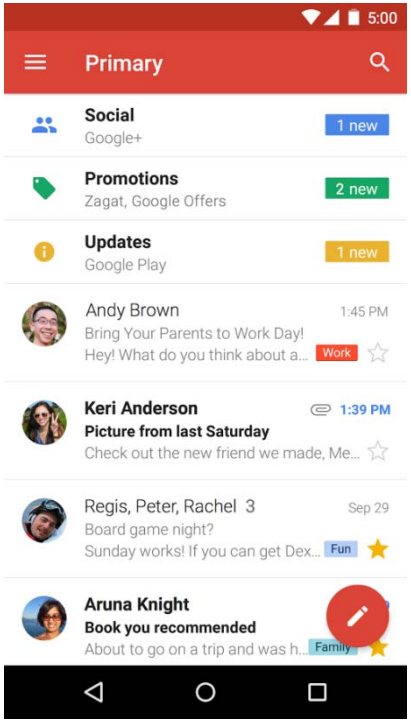


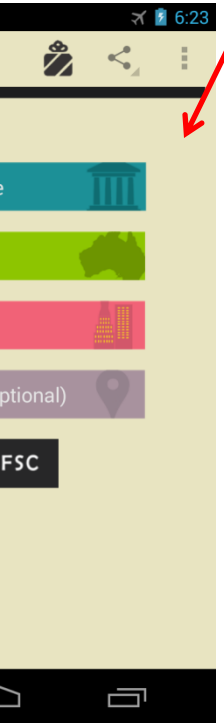
onStop() Method

- An activity is stopped when:
  - User receives phone call
  - User starts a new application
  - Activity 1 launches new Activity 2
- Activity instance and variables of stopped app are retained but no code is being executed by the activity
- If activity is stopped, in onStop( ) method, well behaved apps should
  - save progress to enable seamless restart later
  - Release all resources, save info (persistence)

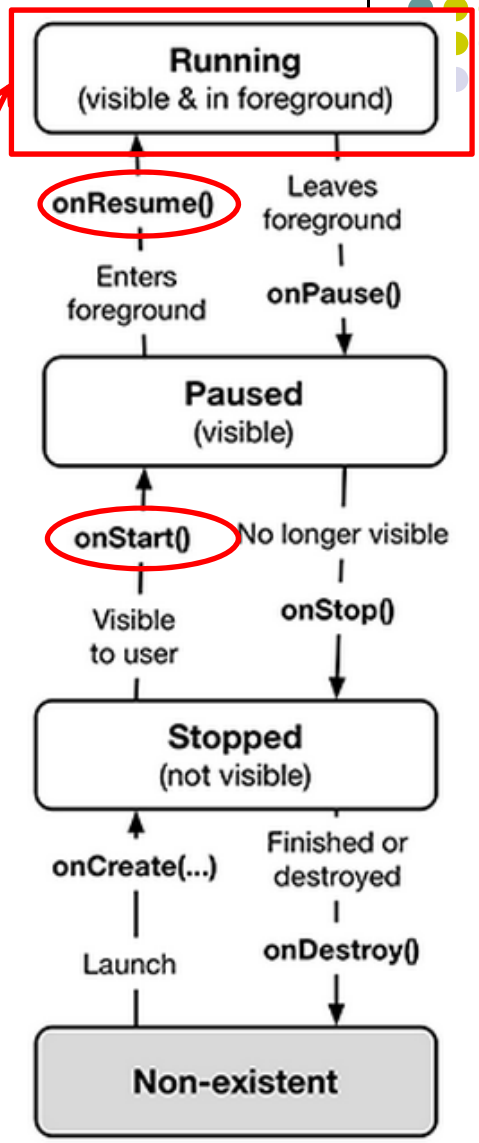
Activity State Diagram: Stopped App

- A **stopped** app can go back into **running** state if becomes visible and in foreground
- App's **onStart( )** and **onResume( )** methods called to transition from **stopped** to **running** state



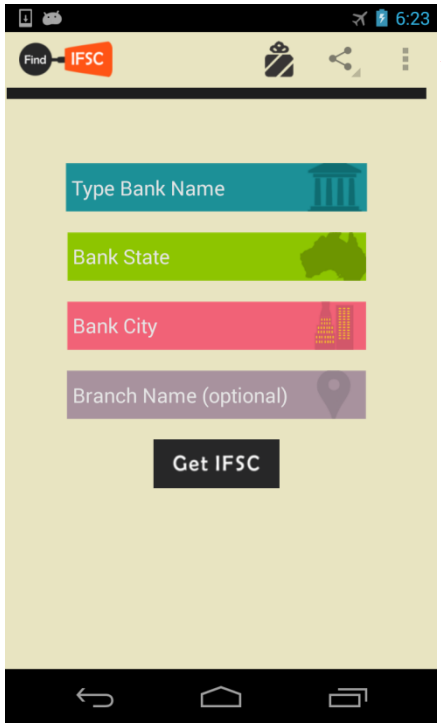


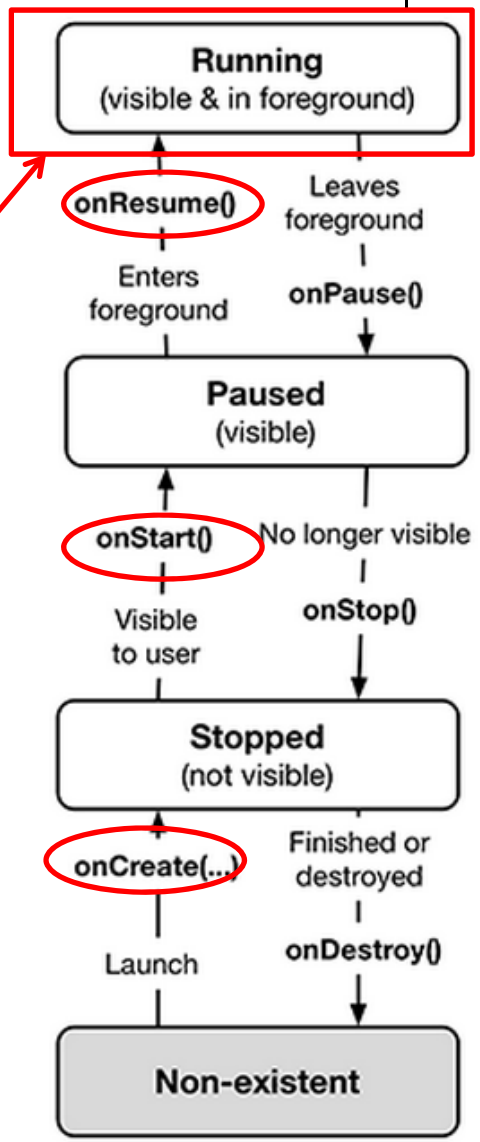
Running

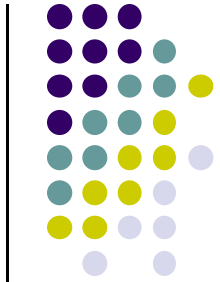


Activity State Diagram: Starting New App

- To start new app, app is launched
- App's **onCreate( )**, **onStart( )** and **onResume( )** methods are called
- Afterwards new app is **running**

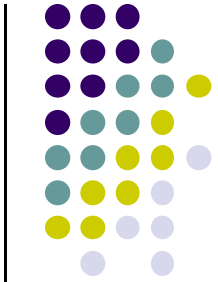






Logging Errors in Android

Logging Errors in Android



- Android can log and display various levels of errors
- Error logging is in **Log** class of **android.util** package

```
import  android.util.Log;
```

- Turn on logging of different message types by calling appropriate method

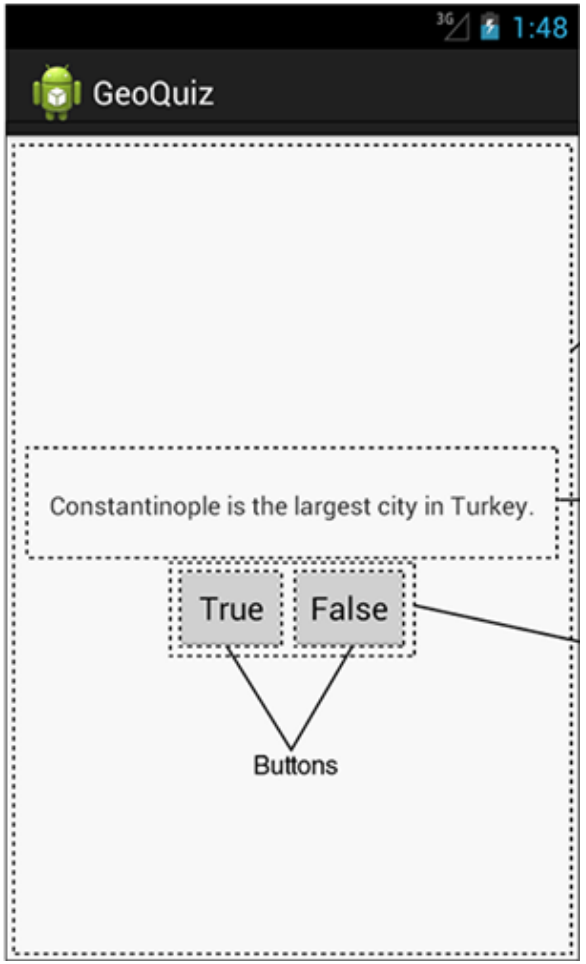
Method	Purpose
`Log.e()`	Log errors
`Log.w()`	Log warnings
`Log.i()`	Log informational messages
`Log.d()`	Log debug messages
`Log.v()`	Log verbose messages

Ref: Introduction to Android Programming, Annuzzi, Darcey & Conder

QuizActivity.java

- A good way to understand Android lifecycle methods is to print debug messages when they are called
- E.g. print debug message from onCreate method below

```
package com.bignerdranch.android.geoquiz;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

public class QuizActivity extends Activity {

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



QuizActivity.java

- Debug (d) messages have the form

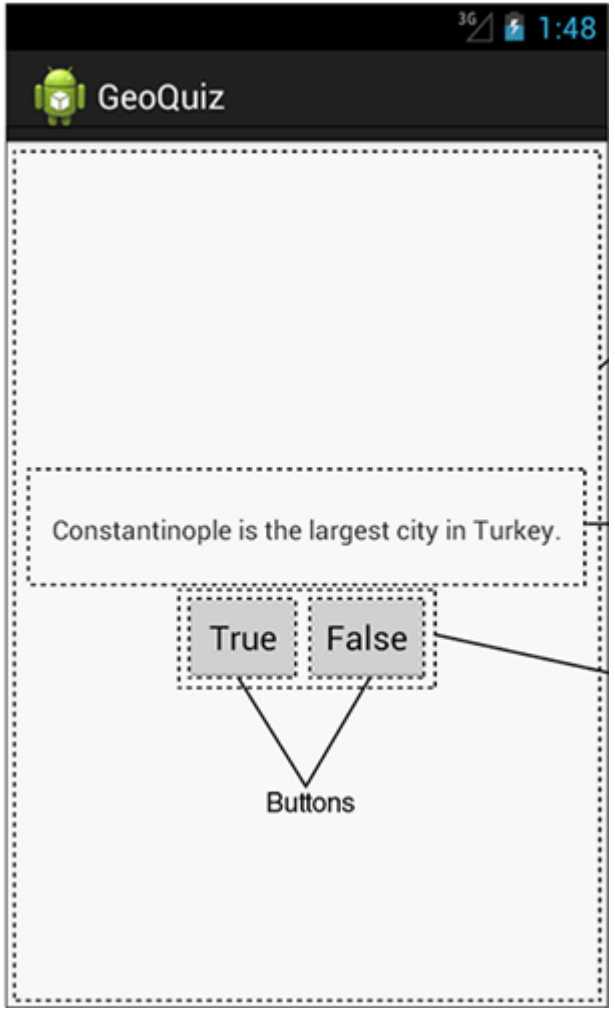
```
public static int d(String tag, String msg)
```

- **TAG** indicates source of message
- Declare string for **TAG**

```
public class QuizActivity extends Activity {

    private static final String TAG = "QuizActivity";

    ...

}
```

- Can then print a message in **onCreate( )**

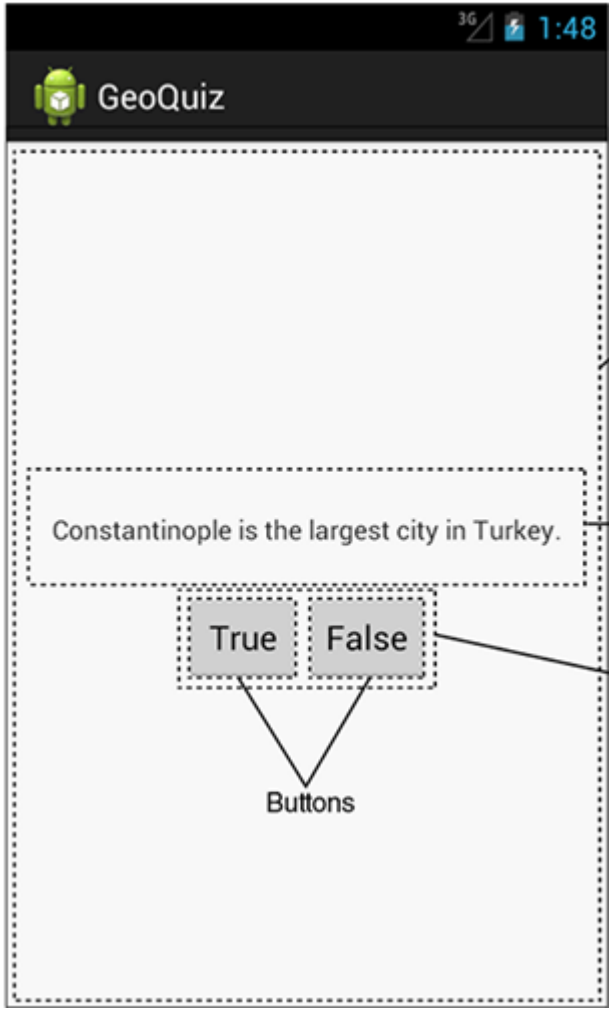
```
Log.d(TAG, "onCreate(Bundle) called");
```



QuizActivity.java

- Putting it all together

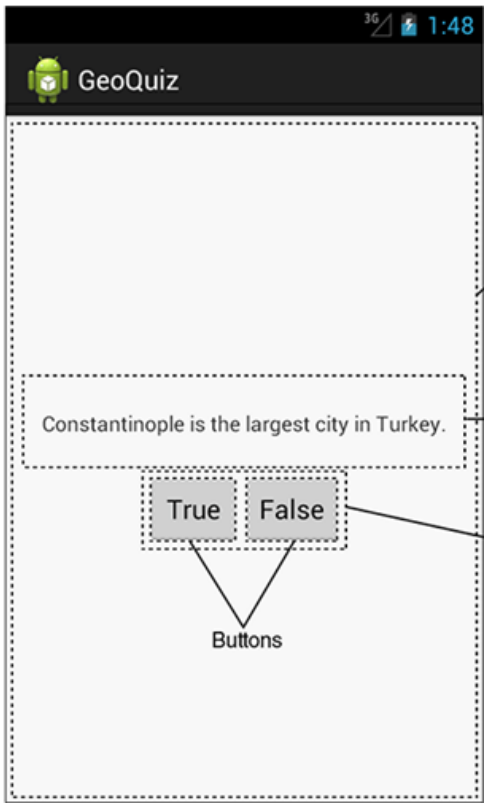
```
public class QuizActivity extends Activity {

    ...

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, "onCreate(Bundle) called");
        setContentView(R.layout.activity_quiz);

        ...
```



QuizActivity.java

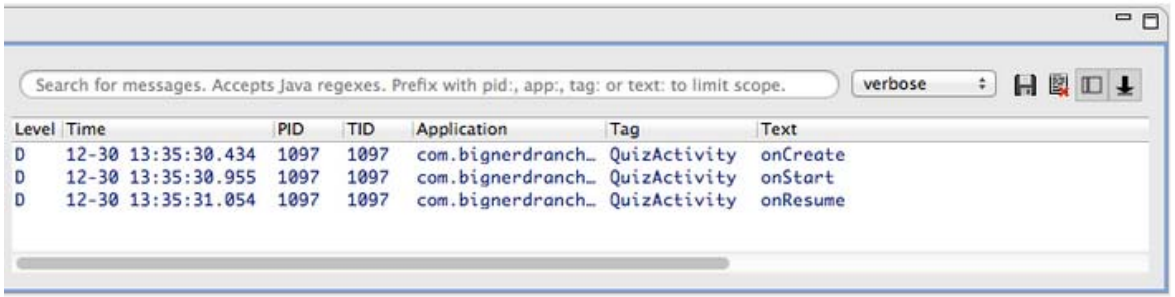
- Can overide more lifecycle methods
- Print debug messages from each method
- Superclass calls called in each method
- @Override asks compiler to ensure method exists in super class

```
} // End of onCreate(Bundle)

@Override
public void onStart() {
    super.onStart();
    Log.d(TAG, "onStart() called");
}

@Override
public void onPause() {
    super.onPause();
    Log.d(TAG, "onPause() called");
}

@Override
public void onResume() {
    super.onResume();
    Log.d(TAG, "onResume() called");
}

@Override
public void onStop() {
    super.onStop();
    Log.d(TAG, "onStop() called");
}

@Override
public void onDestroy() {
    super.onDestroy();
    Log.d(TAG, "onDestroy() called");
}

}
```

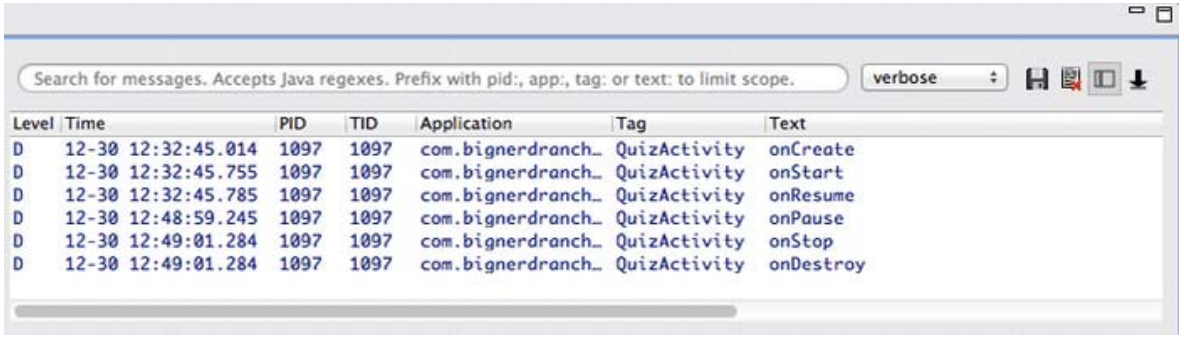


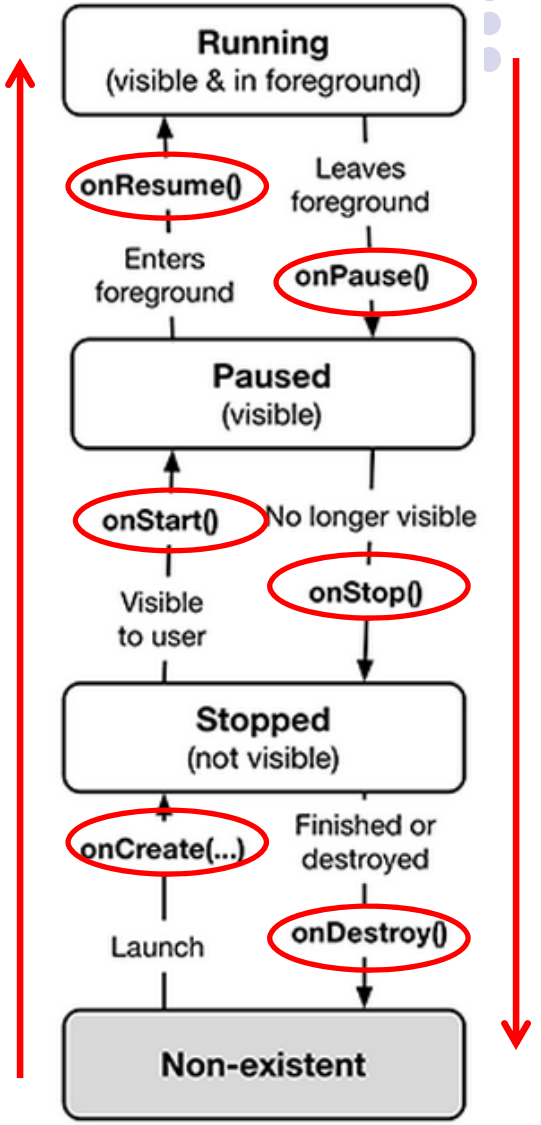
QuizActivity.java Debug Messages

- Launching GeoQuiz app **creates, starts and resumes** an activity



- Pressing **Back** button destroys the activity (calls onPause, onStop and onDestroy)



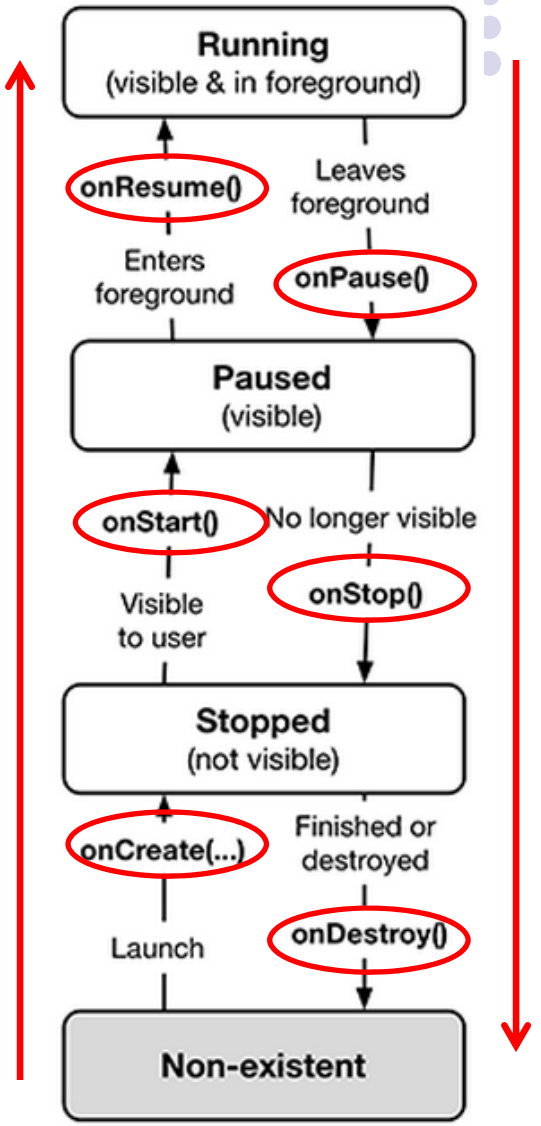


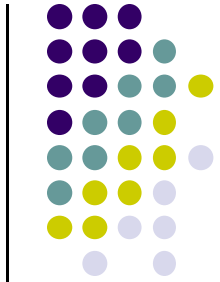
QuizActivity.java Debug Messages

- Pressing **Home** button stops the activity

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

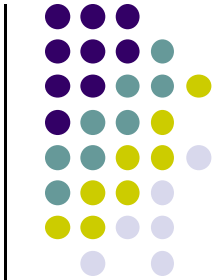
Level	Time	PID	TID	Application	Tag	Text
D	12-30 12:49:01.284	1097	1097	com.bignerdranch…	QuizActivity	onStop
D	12-30 12:49:01.284	1097	1097	com.bignerdranch…	QuizActivity	onDestroy
D	12-30 12:50:01.087	1097	1097	com.bignerdranch…	QuizActivity	onCreate
D	12-30 12:50:01.715	1097	1097	com.bignerdranch…	QuizActivity	onStart
D	12-30 12:50:01.715	1097	1097	com.bignerdranch…	QuizActivity	onResume
D	12-30 12:50:47.075	1097	1097	com.bignerdranch…	QuizActivity	onPause
D	12-30 12:50:49.945	1097	1097	com.bignerdranch…	QuizActivity	onStop



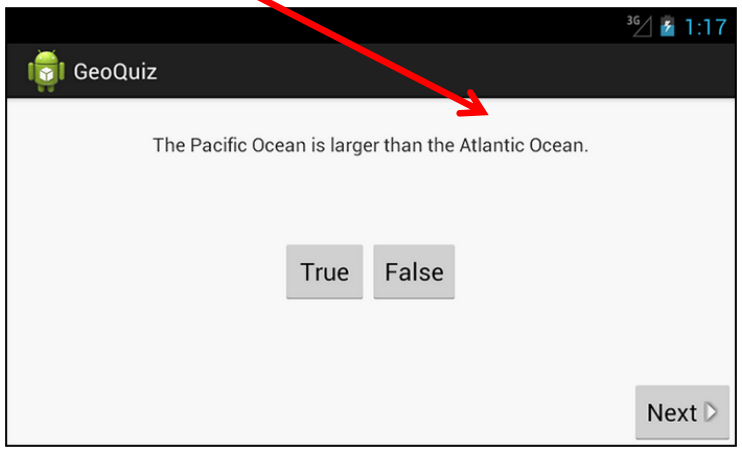


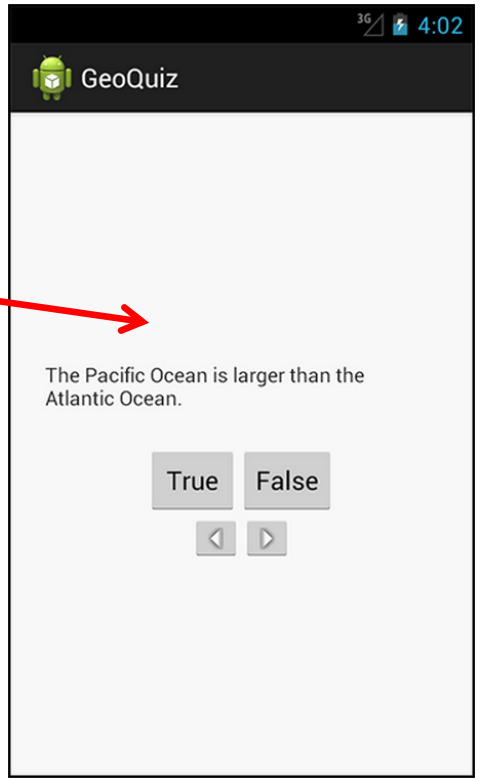
Rotating Device

Rotating Device: Using Different Layouts

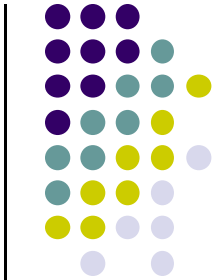


- Rotating device (e.g. portrait to landscape) kills current activity and creates new activity in landscape mode
- Rotation changes **device configuration**
- **Device configuration**: screen orientation/density/size, keyboard type, dock mode, language, etc.
- Apps can specify different resources to use for different device configurations
- E.g. use different app layouts for portrait vs landscape screen orientation

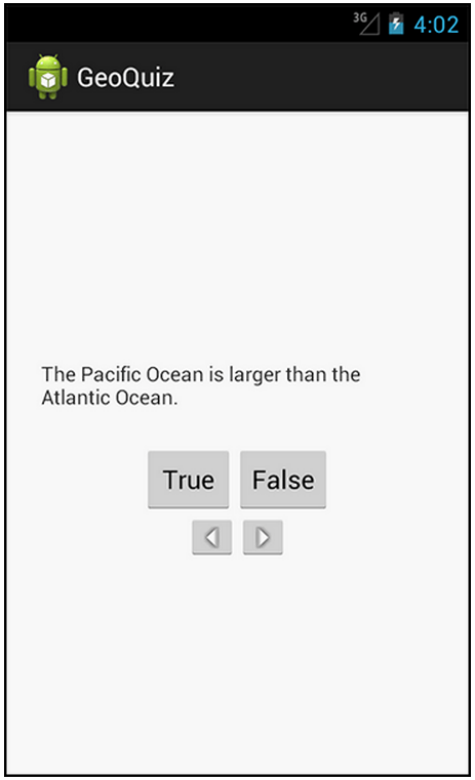


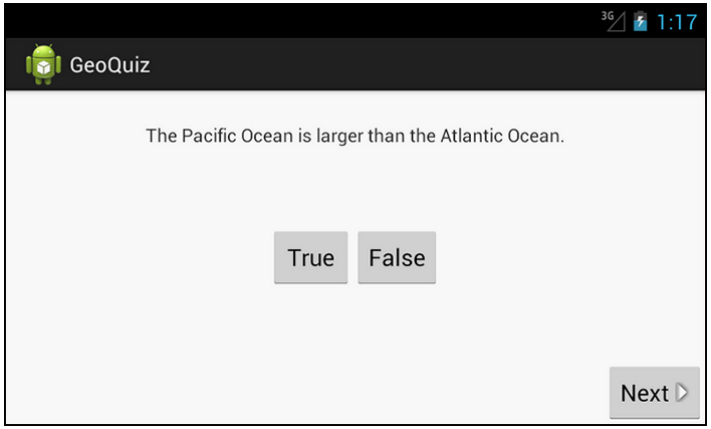


Rotating Device: Using Different Layouts



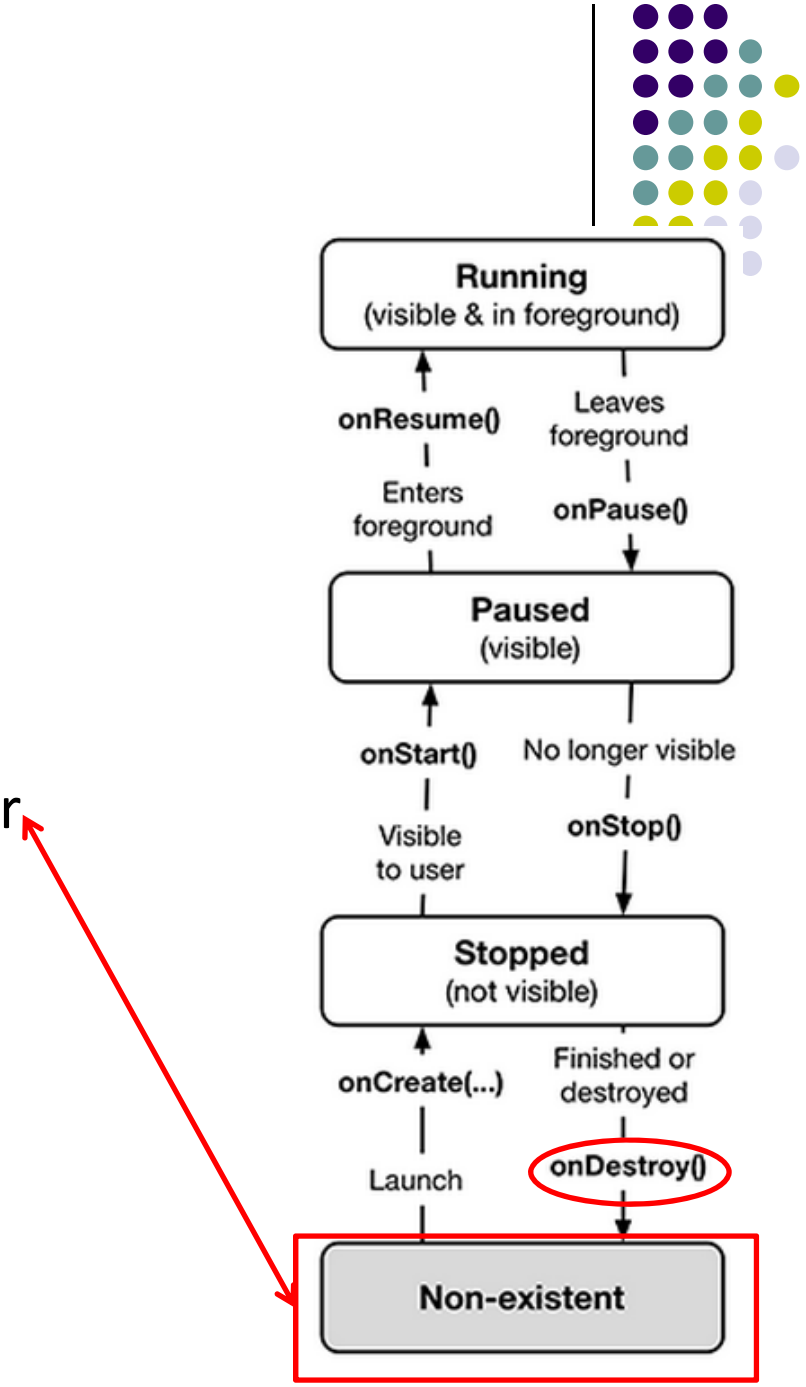
- When device in landscape, uses layout (XML) file in **res/layout-land/**
- Copy XML layout file (activity_quiz.xml) from **res/layout** to **res/layout-land/** and tailor it
- When configuration changes, current activity destroyed, **onCreate (setContentView (R.layout.activity_quiz)** called again

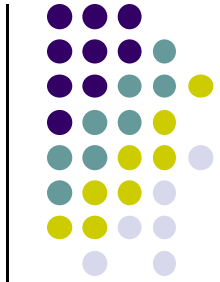




Dead or Destroyed Activity

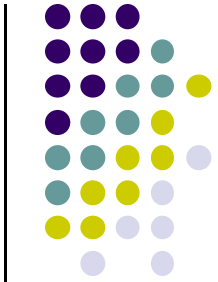
- Dead, activity terminated (or never started)
- onDestroy( ) called to destroy a stopped app





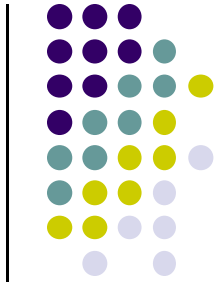
Saving State Data

Activity Destruction



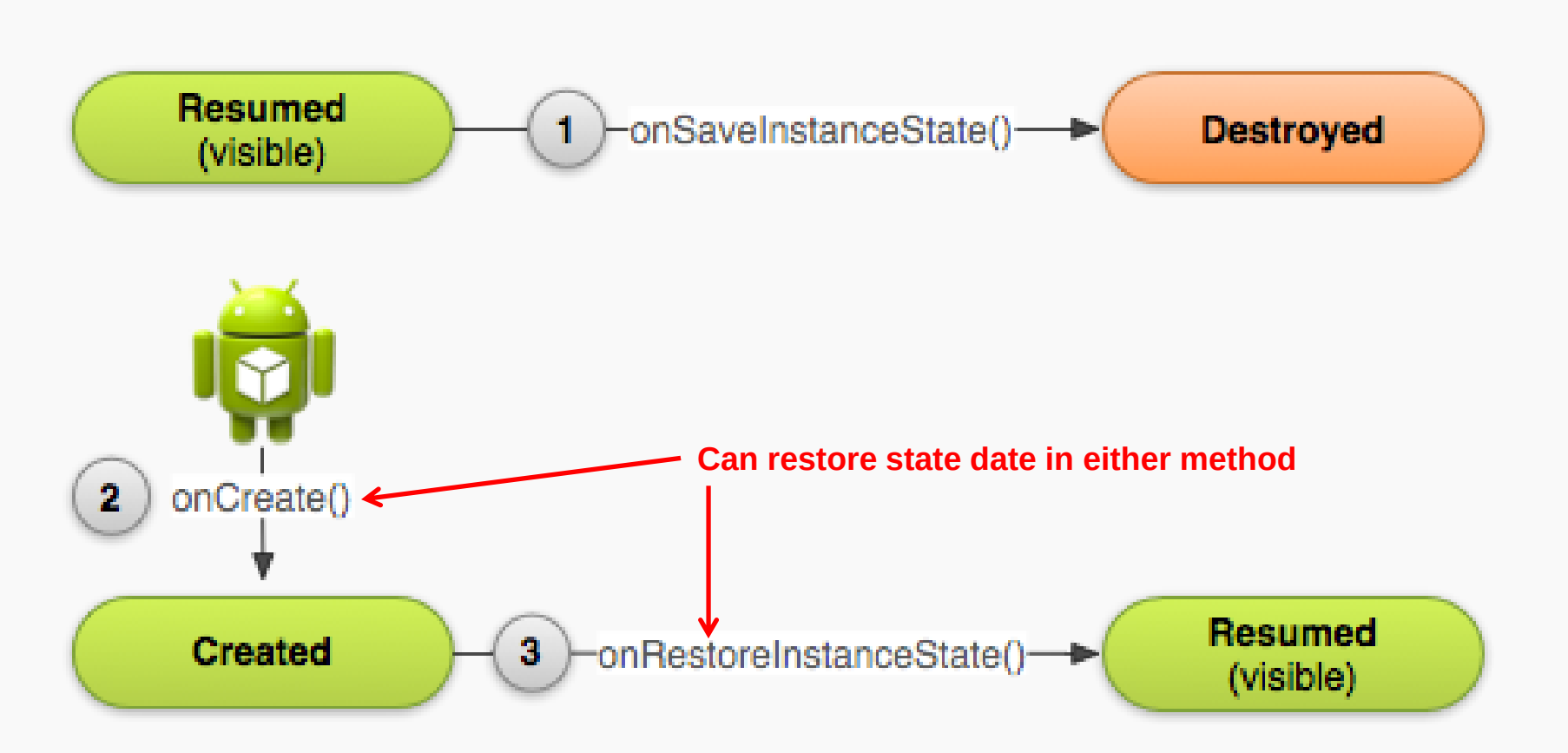
- App may be destroyed
  - On its own by calling finish
  - If user presses **back button**
- Before Activity destroyed, system calls onSaveInstanceState (Bundle outState) method
- Saves state required to recreate Activity later

onSaveInstanceState
onRestoreInstanceState()



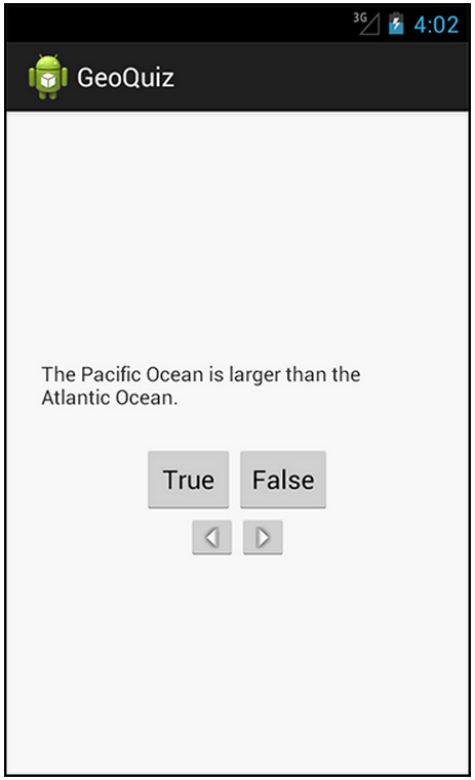
- Systems write info about views to Bundle
- other (app-specific) information must be saved by programmer
  - E.g. board state in a board game such as mastermind
- When Activity recreated Bundle sent to onCreate and onRestoreInstanceState()
- use either method to restore state data / instance variables

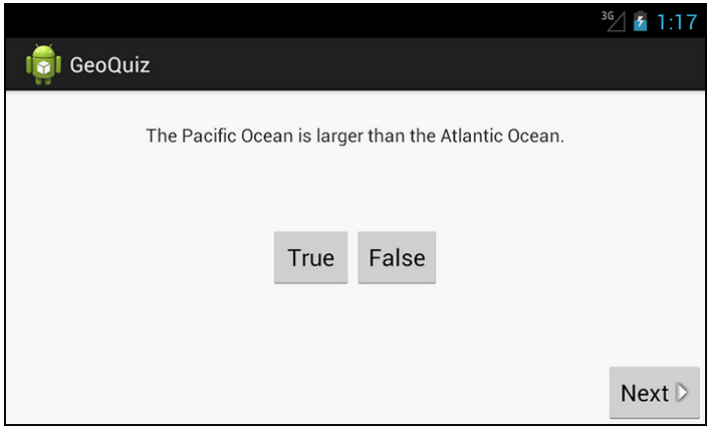
Saving State on Activity Destruction



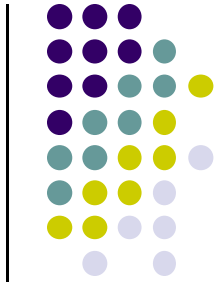
Saving Data Across Device Rotation

- Since rotation causes activity to be destroyed and new one created, values of variables lost or reset
- To stop lost or reset values, save them using **onSaveInstanceState** before activity is destroyed
- System calls **onSaveInstanceState** before **onPause( )**, **onStop( )** and **onDestroy( )**





Saving Data Across Device Rotation

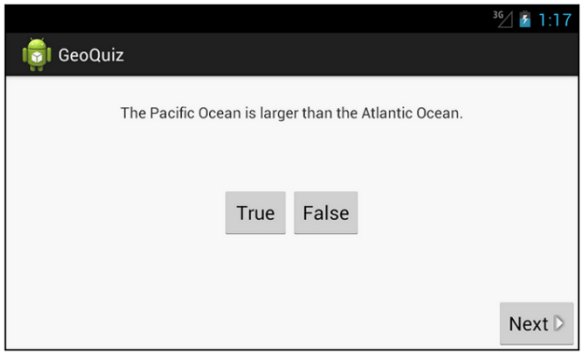


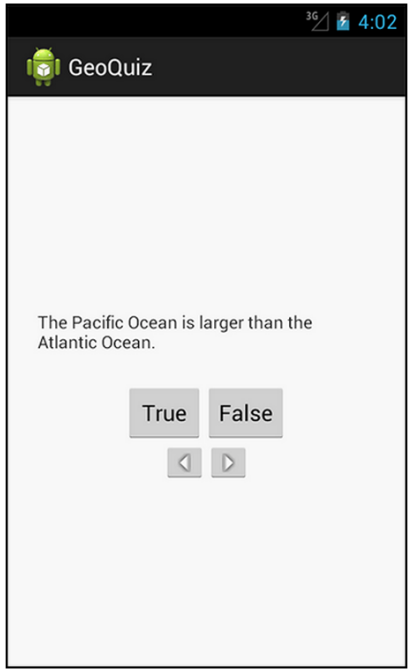
- For example, if we want to save the value of a variable **mCurrentIndex** during rotation
- First, create a constant as a key for storing data in the bundle

```
private static final String KEY_INDEX = "index";
```

- Then override **onSaveInstanceState** method

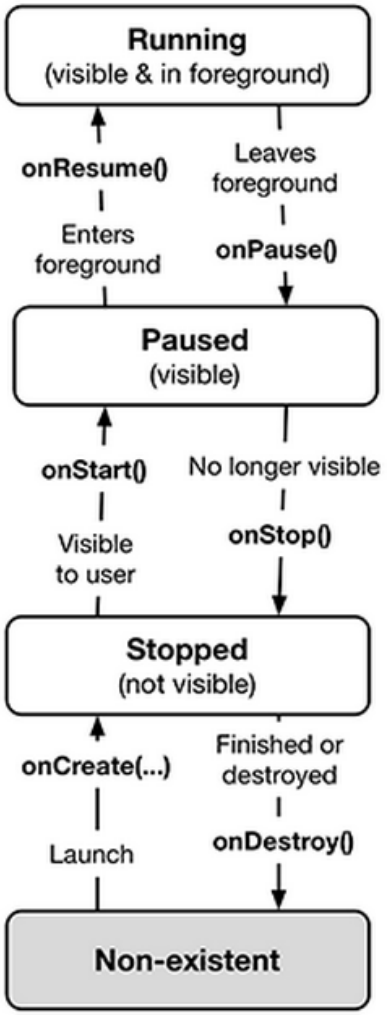
```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    Log.i(TAG, "onSaveInstanceState");
    savedInstanceState.putInt(KEY_INDEX, mCurrentIndex);
}
```

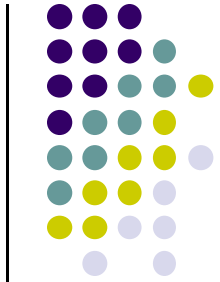




Quiz

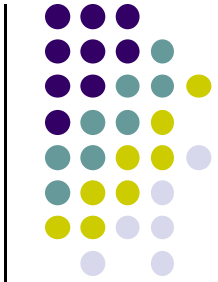
- Whenever I watch YouTube video on my phone, if I receive a phone call and video stops at 2:31, after call, when app resumes, it should restart at 2:31.
- How do you think this is implemented?
  - In which Activity life cycle method should code be put into?
  - How?



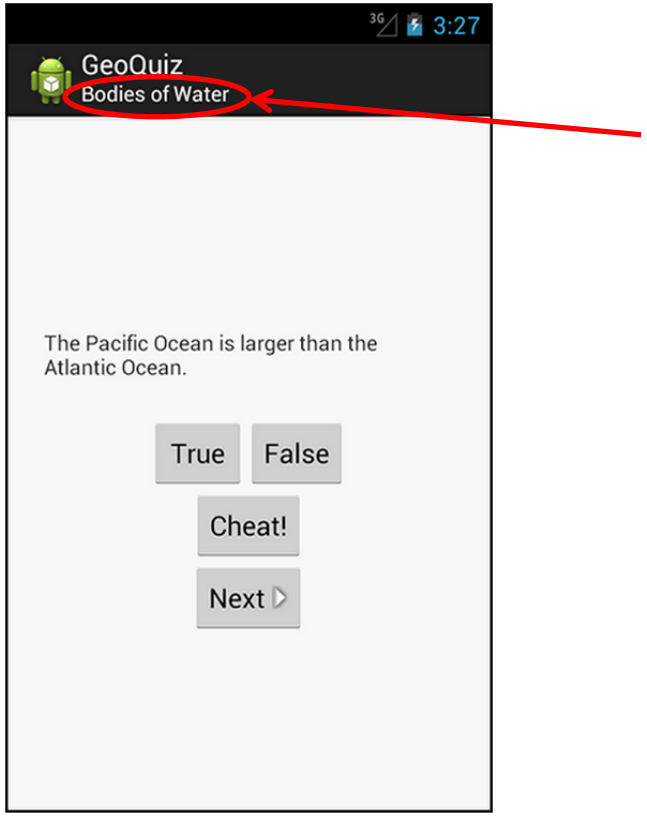


Action Bar

Action Bar (Ref: Android Nerd Ranch 1st Edition)



- Can add Action bar to the onCreate( ) method of GeoQuiz to indicate what part of the app we are in

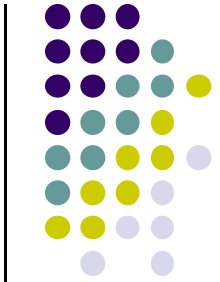


Action bar

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate() called");
    setContentView(R.layout.activity_quiz);

    ActionBar actionBar = getActionBar();
    actionBar.setSubtitle("Bodies of Water");
```

Code to add action bar

References



- Android Nerd Ranch, 1st edition
- Busy Coder's guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014