

G54MDP

Mobile Device Programming

Lecture 4 – Android Application
Components - Activities

iPhone 6S battery life 'varies by up to two hours depending on chip supplier'

TSMC and Samsung make chips for iPhone 6S and 6S Plus, but power consumption varies markedly, according to user reports



📷 A man takes pictures as Apple iPhone 6s and 6s Plus go on sale at an Apple Store in Beijing, China. Photograph: Damir Sagolj/REUTERS

Two seemingly identical, brand new, iPhones 6S may have a battery life that differs by up to two hours depending on which of two companies manufactured their processor, according to reports from owners.

Android Apps

- Applications are written in Java
 - But run on Google's own VM — Dalvik
 - Uses its own bytecode (DEX) format
- Code compiled using standard Java tools then convert to DEX format
 - Multiple class files in a single .dex file
- Code, data and resource files packed into a .apk file
 - Classes
 - Configuration
 - Resources

Android Programming Model

- Traditional OS applications
 - A single entry point
 - Main
 - OS loads the program into a process and executes it
- Java applications
 - A Java VM is instantiated
 - Loads all classes used by the application
 - Executes main

Android Programming Model

- Component based model
 - Multiple application entry points
 - The point through which the system can “enter” the application
 - Not all are entry points for the user
 - Each exists as a logical independent unique entity

Android Components

- Activities
 - UI components
- Services
 - Mechanism for doing something long-running in the background
- Broadcast Receivers
 - Respond to broadcast messages from the OS / other apps
- Content Providers
 - Make data available to / make use of data from other apps
 - No access to the file system
 - SD Card

The Manifest

- Android needs to know about the contents of each application
 - What components does it contain?
 - How are these components started?
 - What does it do?
- .apk contains a manifest file (AndroidManifest.xml) that defines them
- XML syntax

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.helloworld.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

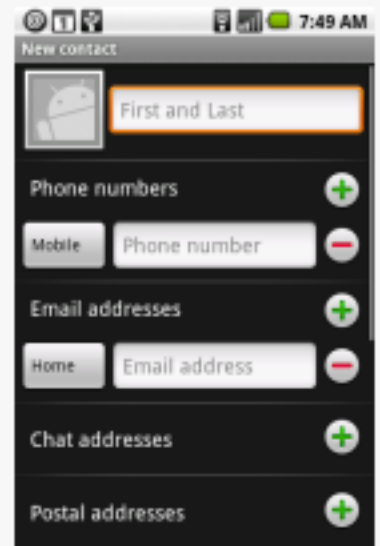
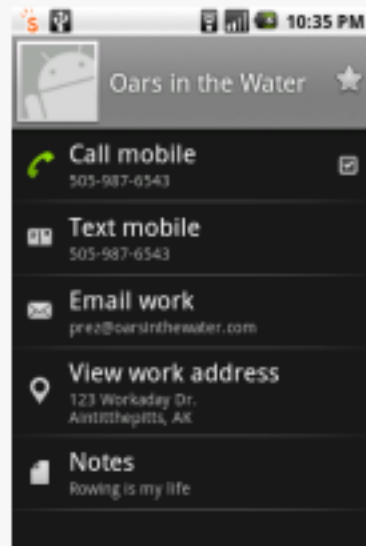
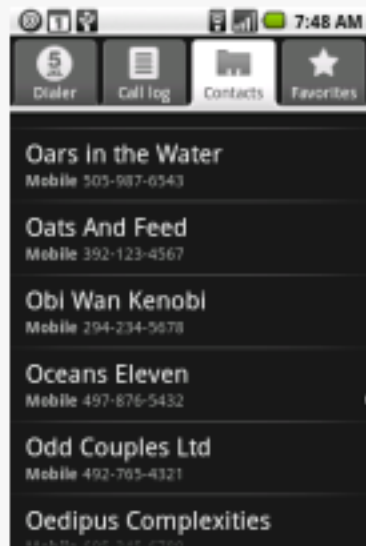
</manifest>
```

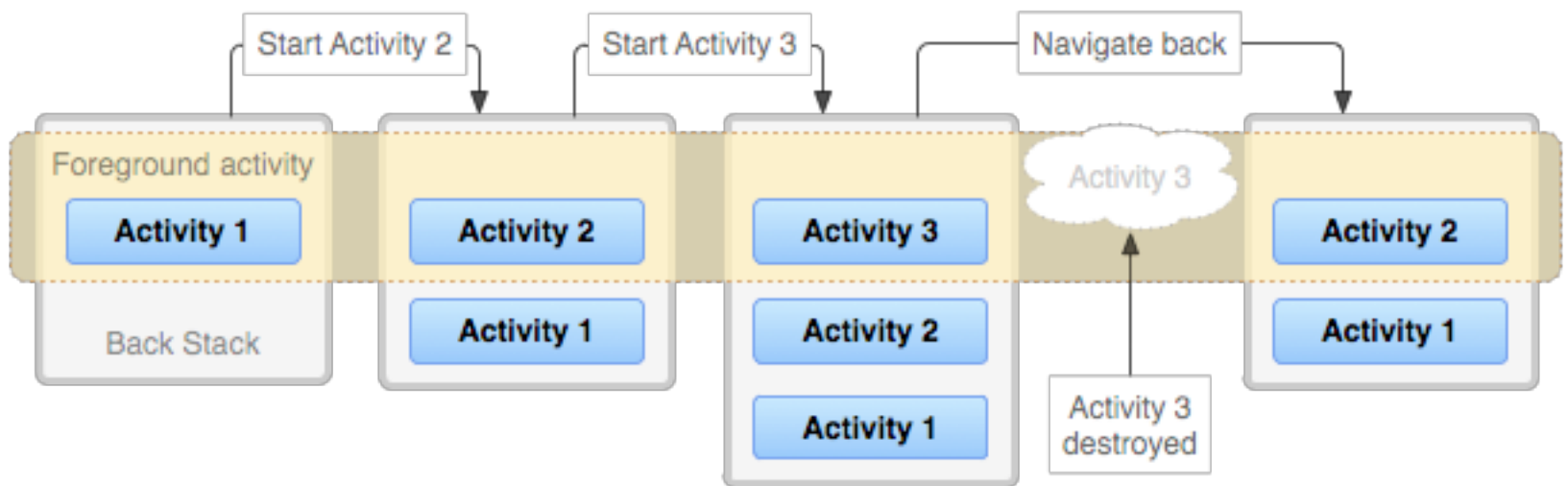

Activities

- Sub-classes of **android.app.Activity**
- Presents a visual UI
- Each Activity has its own “window”
 - Only one “window” on screen at once
- UI layout – a “View”
 - Specified in a separate XML file
 - Constructed programmatically
 - Call setContentView() to display it
- Apps usually have several Activities
 - **Context**
 - An abstract class representing the current application environment

Activities

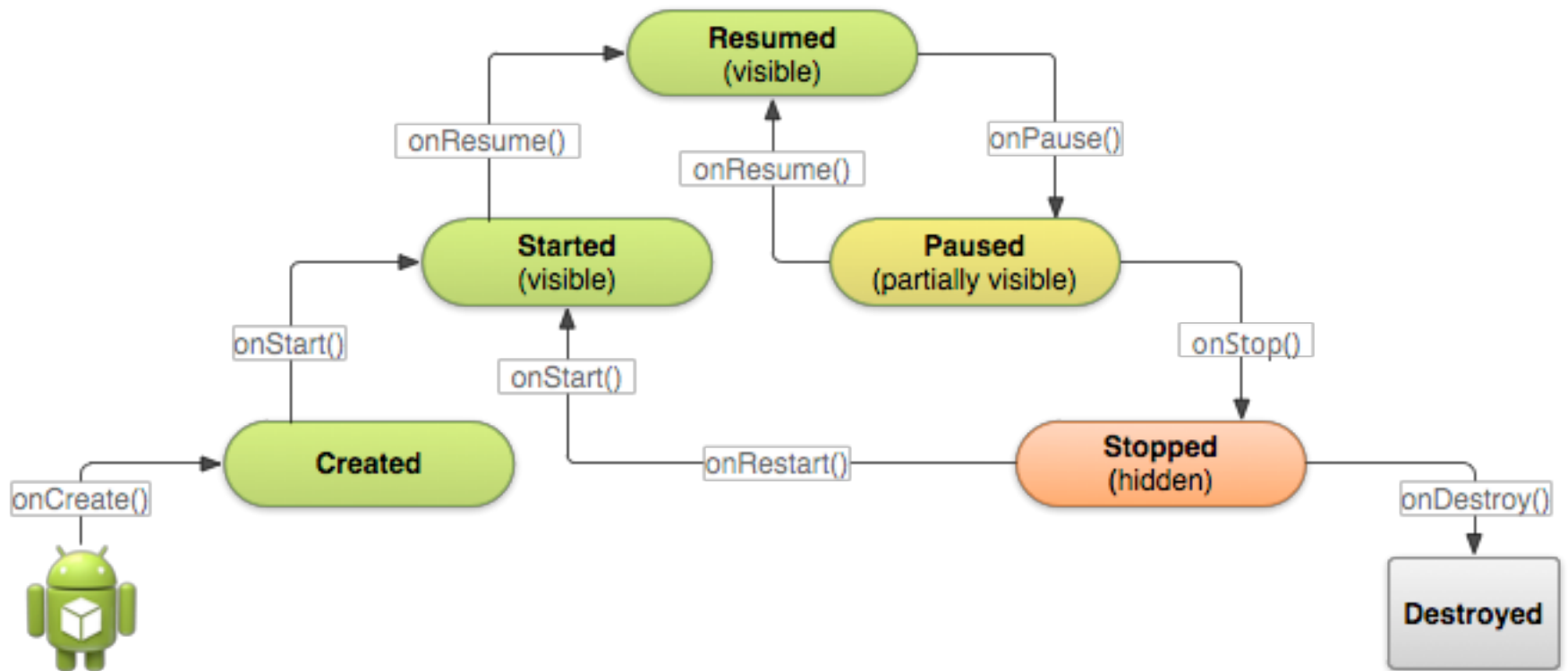
- Activities can start other activities
- Forms a stack of Activities — current activity is on the top
- An activity should be an **atomic** part of a particular task
- Multiple activities form a Task
 - Like...
 - A Task may span multiple applications
- Activities in a task move as a unit from foreground to background and vice versa





Activity Lifecycle

- Essentially three states
 - Active
 - in the foreground
 - Paused
 - still visible, but not top
 - Stopped
 - obscured by another activity
- If paused or stopped, the system can drop the Activity from memory
 - Stopped activities are suspended in memory
 - Consume no processing resources
 - Inactive activities are destroyed if memory is required
 - Oldest first



Android UI

- An Activity has a window associated with it
- Usually full screen
 - Can hover over another activity
 - Can be transparent
- Within the window there is a hierarchy of View objects
- Set with setContentView()
- Usually specified via an XML resource
 - /res/layout/mylayout.xml

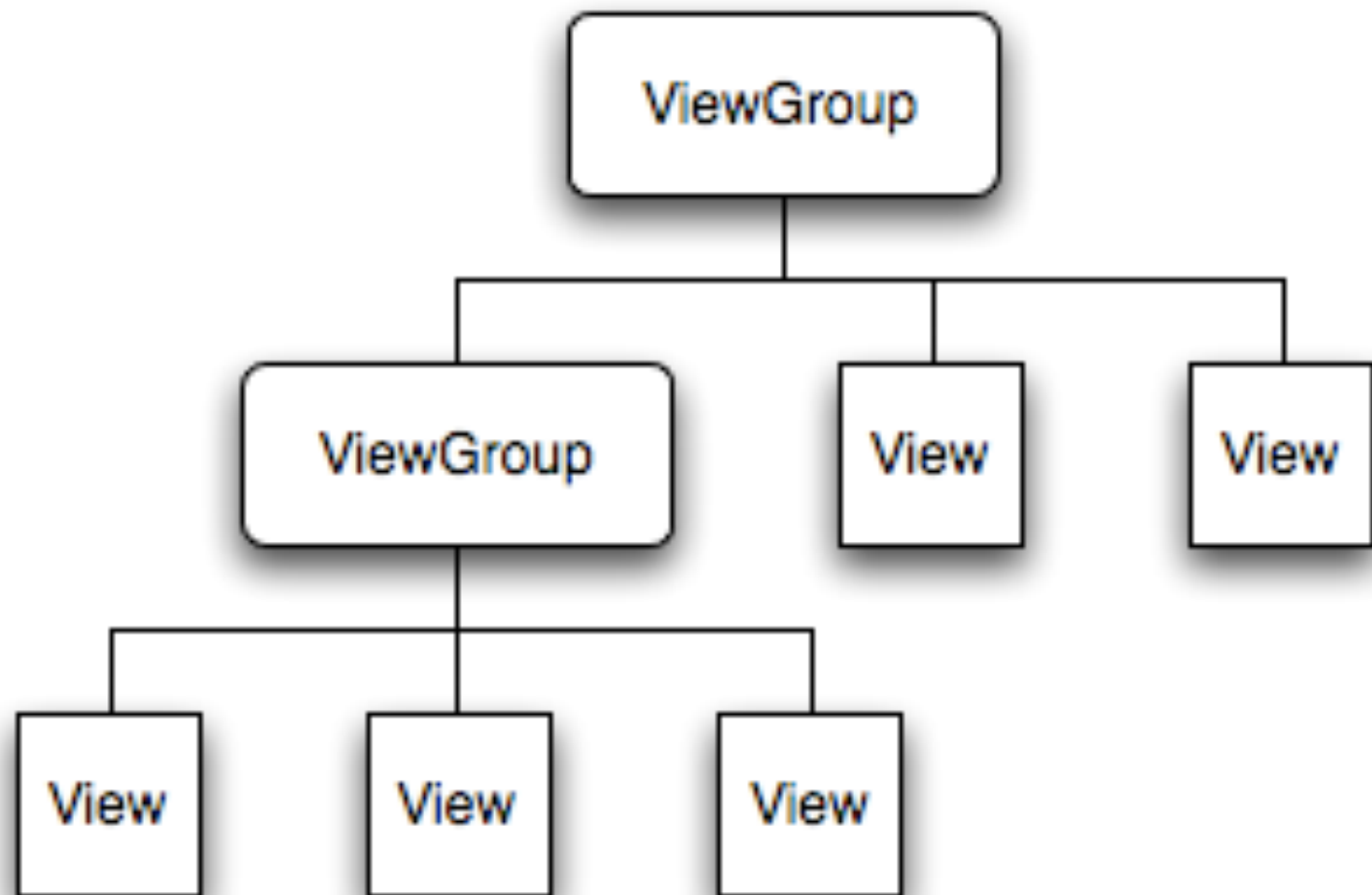
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
```


View Hierarchy

- Types of View subclasses
 - Those that display something
 - Those that do something (Widgets)
 - And ViewGroups which layout a collection of subviews
- Various layout types available — can specify in the XML resource



ViewGroups - Layouts

- **FrameLayout**
 - Simplest, contains a single object
- **LinearLayout**
 - Aligns all children in a single direction, based on the orientation attribute
- **TableLayout**
 - Positions children into rows and columns
- **RelativeLayout**
 - Lets the child views specify their position relative to the parent view or to each other
- **ScrollView**
 - A vertically scrolling view, like FrameLayout only contains a single element (e.g. a LinearLayout)

Views - Widgets

- A child View that the user can (optionally) interact with
 - Button (a button)
 - EditText (text entry)
 - CalendarViewer (a calendar widget)
 - ImageView (displays an image)
 - ...
- Handle appropriate UI events
 - In code, register `setOnClickListener()`
 - In XML layout, set `android:onClick` parameter
- Properties / parameters
 - Set via XML at build-time
 - Equivalent set / get methods for modifying at run-time
`android:text="@string/hello_world" />`
`.getText(...)`, `.setText(...)`

Views - Parameters

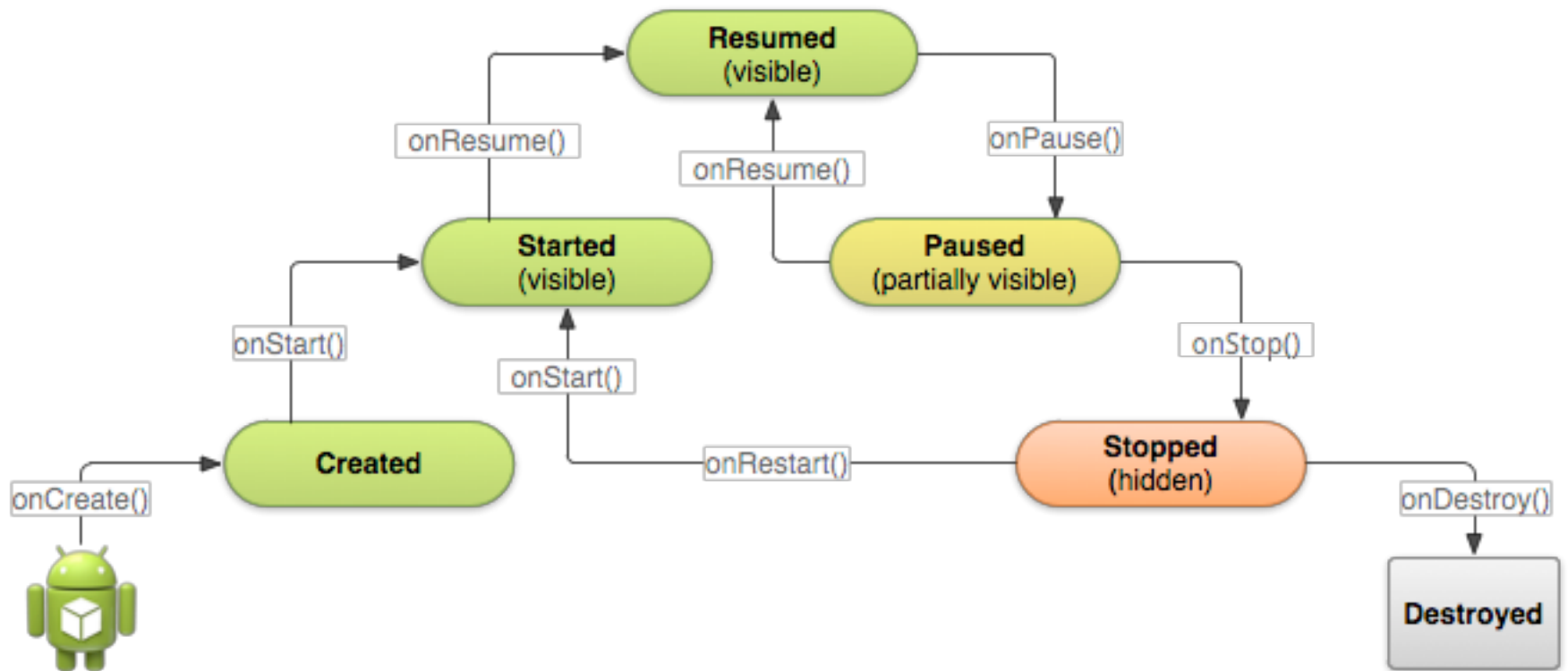
- Parameters specify the details of particular Views
 - Width, height
 - Generally **not** in terms of absolute pixels
 - Relative to parent, percentages, offsets, margins
 - `android:layout_width="match_parent"`
 - `android:layout_height="wrap_content"`
 - ID
 - Used to generate a Java member variable we can refer to programmatically
 - `android:id="@+id/my_button"`
 - Methods
 - Used to automatically bind UI events to code
 - `android:onClick="myMethod"`

Manipulating Views

- We can alter the view hierarchy programmatically as the application runs
- ViewGroup provides methods
 - To add `addView()`
 - Need to either keep a reference to it or call `setId()` on the view so we can find it later
 - Or use references generated from layout XML for existing views
 - Or to remove `removeView()` children
 - Can add new buttons, layouts as required

Let's have a look...





Saving State

- Shouldn't rely on an Activity storing UI state
 - E.g. rotating the device
 - Destroys and recreates the activity
 - If you need to keep it, save it
- Before `onStop()` is called, Android will call `onSaveInstanceState()`
 - To restore the **UI** to its previous state on restore
 - This allows you to save any **UI state** into a Bundle object
 - When the Activity is recreated, the Bundle is passed to `onCreate()` and `onRestoreInstanceState()`
 - Giving the Activity chance to restore its state
 - Save **other state** to more persistent storage
 - SQLite database / user preferences
 - More on this later

Saving UI State

- Bundle
 - A collection of key/value pairs
 - Key
 - Unique String identifier
 - Value
 - A primitive value
 - A Serializable / Parcelable object
 - Writing and reading a complex class
 - More on this later on (IPC)
 - i.e. `myBundle.putInt("myInteger", 5);`
 - ... `int i = myBundle.getInt("myInteger");`

Let's have a look...



References

- <http://developer.android.com/training/basics/activity-lifecycle/index.html>
- <http://developer.android.com/guide/topics/ui/declaring-layout.html>
- <http://developer.android.com/guide/topics/ui/controls.html>