

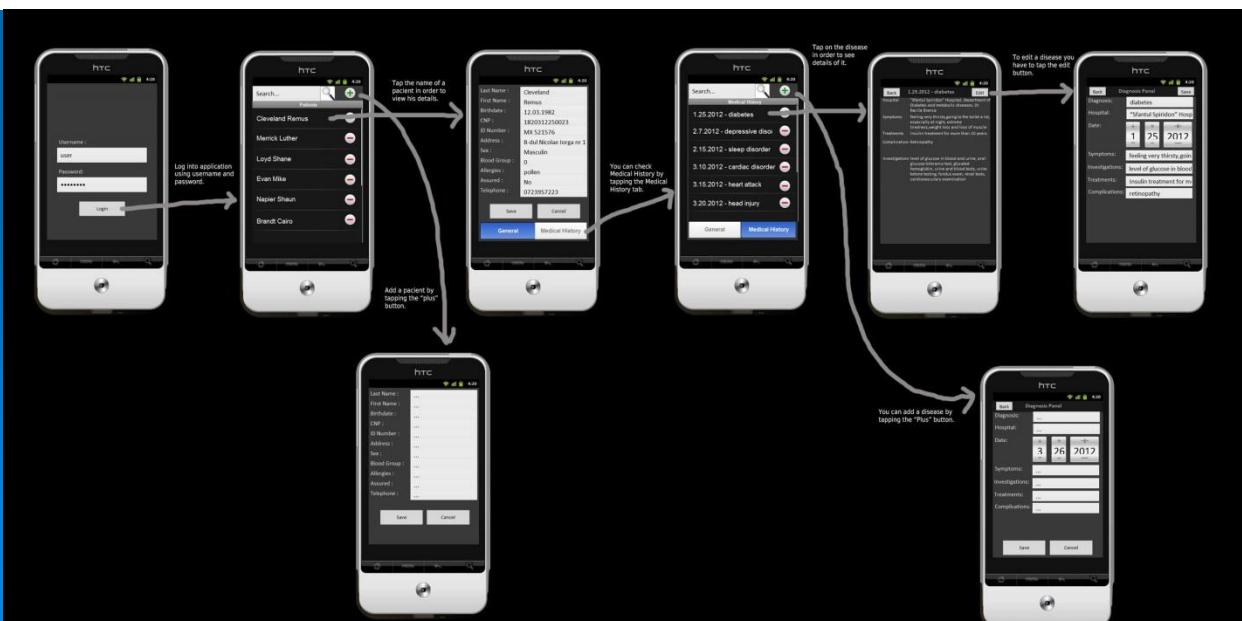
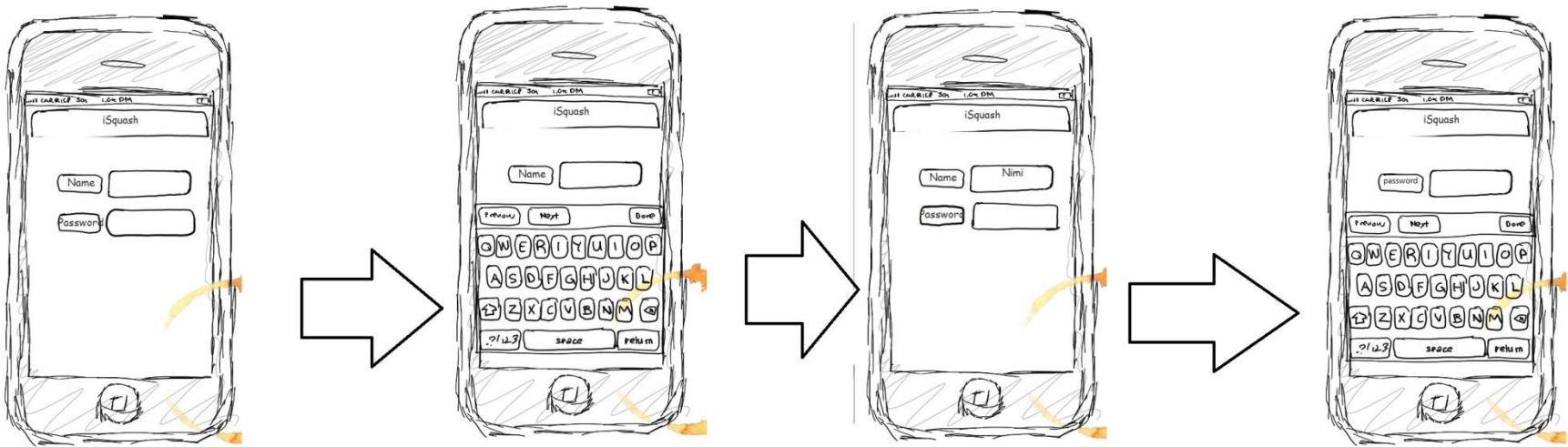
COMP08076

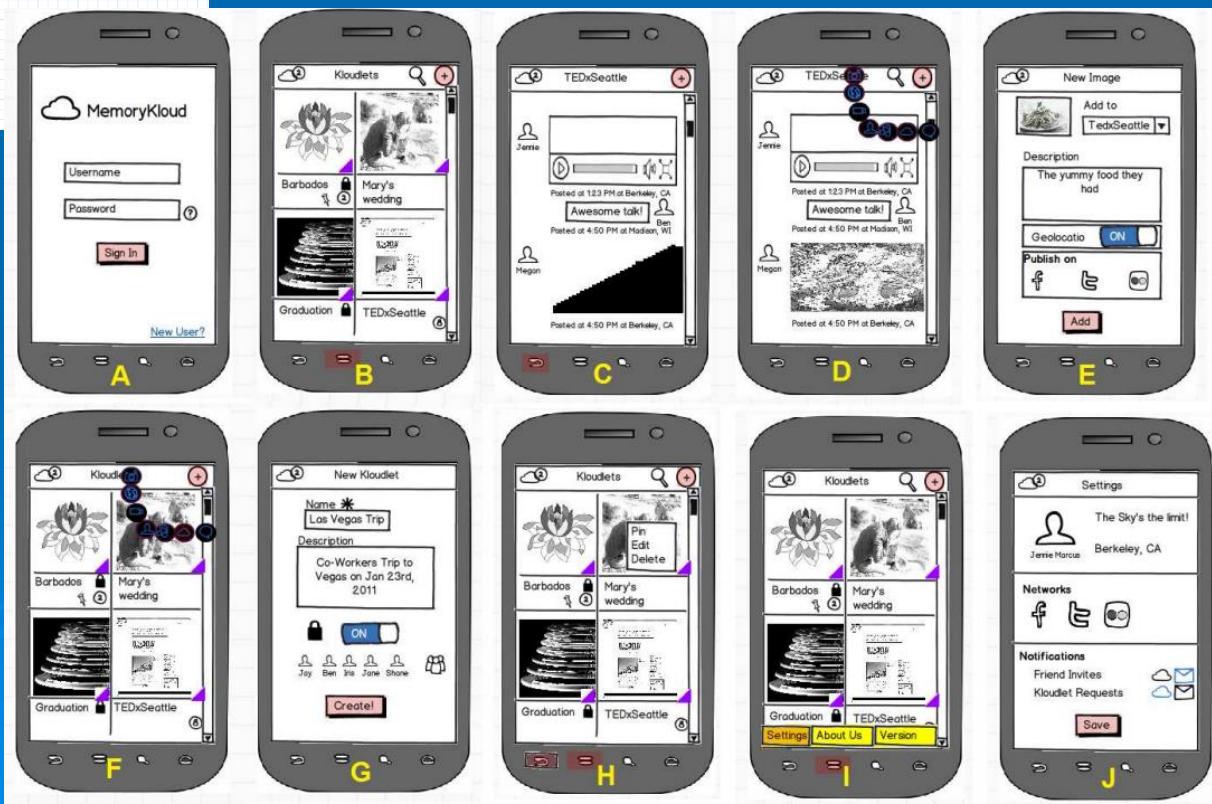
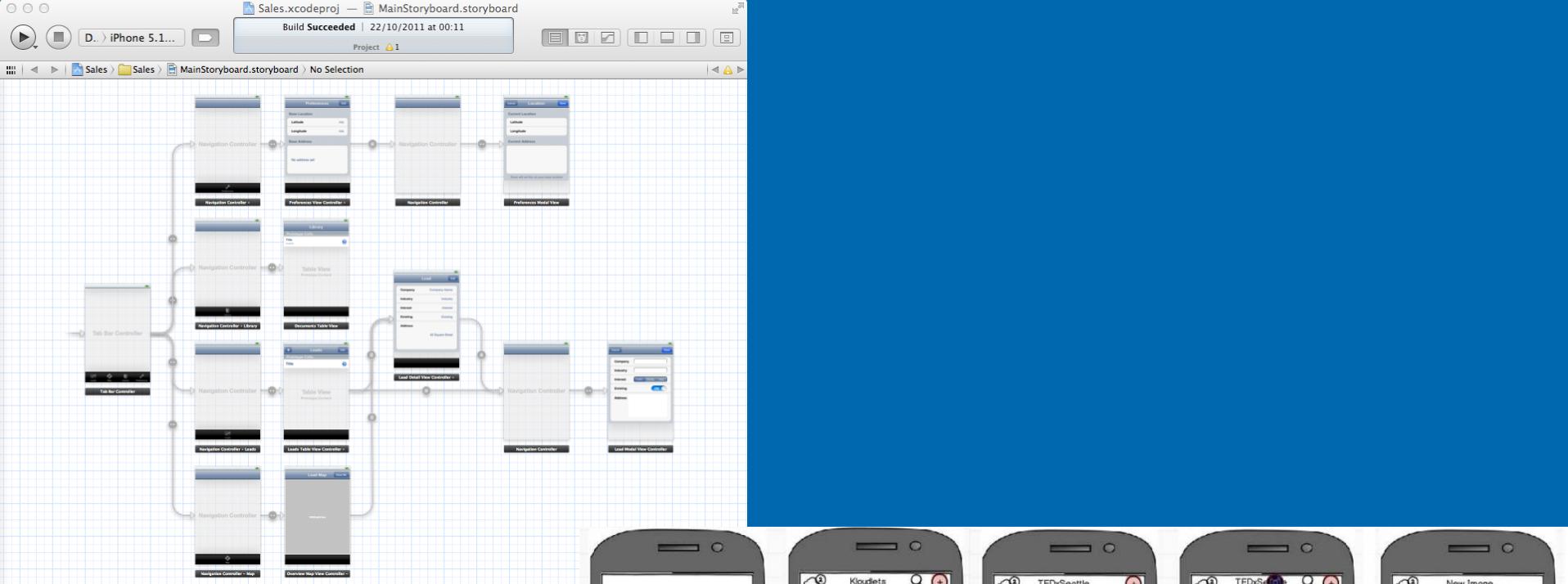
Programming Native App Interaction

- Module coordinator: John Nixon
 - john.nixon@uws.ac.uk, room E259, x3617
- Lecture 3
 - Wireframes and design
 - Some java
 - The Activity – the building block of Android apps
 - Activity life-cycle
 - Views

Wireframe

- A **wireframe** is a visual representation of the skeletal structure of a website or mobile application
 - Information design (information architecture)
 - Navigation design
 - Interface design
- <http://webdesign.tutsplus.com/tutorials/workflow-tutorials/a-beginners-guide-to-wireframing/>
- Numerous software packages





Balsamiq

Variables

- Are for holding data
- Declaring a variable
 - boolean flag; or boolean flag = false;
 - true or false
 - double myHeight = 4.67;
 - float myHeight = 4.67f;
 - int numberOfPages = 2;
 - String hotelName = "Imperial";
- Scope of a variable
 - Class variables, local variables

If statement

- if (condition is true) {
 - do something;}
- if (condition is true) {
 - do something;} else {
 - do something different}

Things to test in an if statement

- Is a boolean true or false
 - `if(isMoving){...}`
- Comparison of values
 - `if (x > y) {...}`
- Component properties
 - `if (myCheckbox.isChecked()) {...}`
 - `if (myRadioButton.isChecked()) {...}`
 - Use android reference to find out more about components
 - E.g. <http://developer.android.com/reference/android/widget/CheckBox.html>

```
> public class MainActivity extends Activity {  
  
>     @Override  
>     protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // other stuff  
>    }  
  
>    // other stuff  
> }
```

What is an Activity?

➤ A class

➤ Activity

- A application component that has a **screen** with which users can interact
 - To make a phone call, take a photo etc.
- Each screen of an application is a different activity (i.e. an application may have several activities)
- Activity has a life cycle
- Sub class the activity class to make one

➤ Discussion on activities

- <http://developer.android.com/guide/components/activities.html>

Objects and Classes

- A class is a template for a type of object
 - All objects are defined by a class
 - All objects of the same class have the same attributes
 - All objects of the same class have a defined set of methods
- Inheritance
- An existing class can be used as the starting point for a new one
 - The new class **extends** the existing one
 - Methods in the new class **override** existing methods

```
> public class MainActivity extends Activity {  
  
>     @Override  
>     protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // other stuff  
>    }  
  
>    // other stuff  
> }
```

- Create a subclass of activity
 - public class MainActivity extends Activity {
- Implement the onCreate method – the **system** calls this when creating your activity
 - protected void onCreate(Bundle savedInstanceState) {
 - *protected* – accessible only from within class and any subclass – inherited by subclass
 - *savedInstanceState* – stores information about activity to recreate it if required
- @Override – a compiler directive

Activity Class

http://developer.android.com/reference/android/app/Activity.html

Activity | Android Developers

Genymotion – Fast And Easy A...

Developers Design Develop Distribute

Developer Console

Training API Guides Reference Tools Google Services Samples

Android APIs API level: 15

public class

Summary: Constants | Inherited Constants | Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All] **Added in API level 1**

Activity

extends ContextThemeWrapper

implements LayoutInflator.Factory2, Window.Callback, KeyEvent.Callback, View.OnCreateContextMenuListener, ComponentCallbacks2

java.lang.Object
└ android.content.Context
 └ android.content.ContextWrapper
 └ android.view.ContextThemeWrapper
 └ android.app.Activity

▶ Known Direct Subclasses
[AccountAuthenticatorActivity](#), [ActivityGroup](#), [AliasActivity](#), [ExpandableListActivity](#), [FragmentActivity](#), [ListActivity](#), [NativeActivity](#)

▶ Known Indirect Subclasses
[ActionBarActivity](#), [AppCompatActivity](#), [LauncherActivity](#), [PreferenceActivity](#), [TabActivity](#)

Class Overview

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(View)`. While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with `windowIsFloating` set) or embedded inside of another activity (using `ActivityGroup`). There are two methods almost all subclasses of Activity will implement:

- `onCreate(Bundle)` is where you initialize your activity. Most importantly, here you will usually call `setContentView(int)` with a layout resource defining your UI, and using `findViewById(int)` to retrieve the widgets in that UI that you need to interact with programmatically.
- `onPause()` is where you deal with the user leaving your activity. Most importantly, any changes made by the user should at this point be committed (usually to the `ContentProvider` holding the data).

To be of use with `Context.startActivity()`, all activity classes must have a corresponding `<activity>` declaration in their package's `AndroidManifest.xml`.

Topics covered here:

1. [Fragments](#)
2. [Activity Lifecycle](#)
3. [Configuration Changes](#)

Use Tree Navigation

➤ <http://developer.android.com/reference/android/app/Activity.html>

API level: 15

public class

Summary | Constants | Inherited Constants

Activity

extends [ContextThemeWrapper](#)implements [LayoutInflater.Factory2](#) [Window.Callback](#) [KeyEvent.Callback](#) [View.OnCreateContextMenuListener](#) [ComponentCallbacks2](#)[java.lang.Object](#)↳ [android.content.Context](#)↳ [Android.content.ContextWrapper](#)↳ [Android.view.ContextThemeWrapper](#)↳ [Android.app.Activity](#)

▶ Known Direct Subclasses

[AccountAuthenticatorActivity](#), [ActivityGroup](#), [AliasActivity](#), [ExpandableListActivity](#), [FragmentActivity](#), [ListActivity](#), [NativeActivity](#)

▶ Known Indirect Subclasses

[ActionBarActivity](#), [AppCompatActivity](#), [LauncherActivity](#), [PreferenceActivity](#), [TabActivity](#)

Class Overview

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the `Activity` class typically overrides the `onCreate(Bundle)` method to set up the UI. While activities are often presented to the user as full-screen windows, they can also be used in a `Fragment` (in which case they are called `FragmentActivity`) or embedded inside of another activity (using `ActivityGroup`). There are two methods almost all subclasses of `Activity` implement:

- `onCreate(Bundle)` is where you initialize your activity. Most importantly, here you will usually call `setContentView(int)` to retrieve the widgets in that UI that you need to interact with programmatically.

ListActivity class, extends Activity

The screenshot shows a web browser displaying the Android developer documentation for the `ListActivity` class. The URL in the address bar is `http://developer.android.com/reference/android/app/ListActivity.html`. The page title is "ListActivity | Android Dev...". The left sidebar shows a navigation tree under "Develop > Reference" with categories like "Android APIs API level: 15", "android", "Interfaces", and "ActionBar.OnMenuItemVisibilityListener". The main content area has a large title "ListActivity" and a subtitle "extends Activity". It lists the class hierarchy: `java.lang.Object`, `↳ android.content.Context`, `↳ android.content.ContextWrapper`, `↳ android.view.ContextThemeWrapper`, `↳ android.app.Activity`, and `↳ android.app.ListActivity`. Below this, it shows "Known Direct Subclasses" with `LauncherActivity` and `PreferenceActivity`. A "Class Overview" section describes it as an activity for displaying lists. A "Screen Layout" section discusses its use of a `ListView`. At the bottom, there's a "Use Tree Navigation" button.

Android APIs API level: 15

Develop > Reference > **ListActivity**

ListActivity

extends [Activity](#)

`java.lang.Object`
 ↳ `android.content.Context`
 ↳ `android.content.ContextWrapper`
 ↳ `android.view.ContextThemeWrapper`
 ↳ `android.app.Activity`
 ↳ `android.app.ListActivity`

▶ Known Direct Subclasses
[LauncherActivity](#), [PreferenceActivity](#)

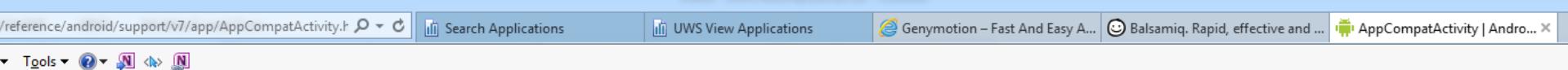
Class Overview

An activity that displays a list of items by binding to a data source such as an array or Cursor, and exposes event handlers when the user selects an item.

ListActivity hosts a `ListView` object that can be bound to different data sources, typically either an array or a Cursor holding query results. Binding, screen layout, and row layout are discussed in the following sections.

Screen Layout

AppCompatActivity



API level: 15

AppCompatActivity

extends [FragmentActivity](#)

implements [AppCompatCallback](#) [TaskStackBuilder.SupportParentable](#) [ActionBarDrawerToggle.DelegateProvider](#)

java.lang.Object
└ android.content.Context
 └ android.content.ContextWrapper
 └ android.view.ContextThemeWrapper
 └ android.app.Activity
 └ android.support.v4.app.FragmentActivity
 └ android.support.v7.app.AppCompatActivity

► Known Direct Subclasses

[ActionBarActivity](#)

Class Overview

Base class for activities that use the [support library](#) action bar features.

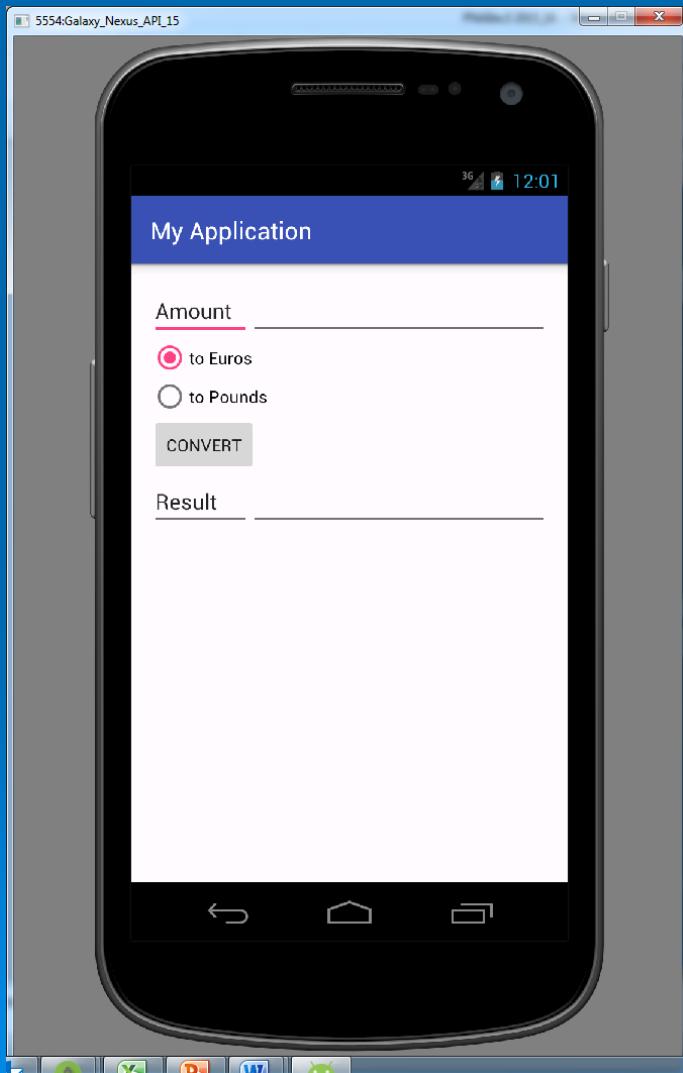
You can add an [ActionBar](#) to your activity when running on API level 7 or higher by extending this class for your activity and setting the activity theme to [Theme.AppCompat](#) or a similar

Developer Guides

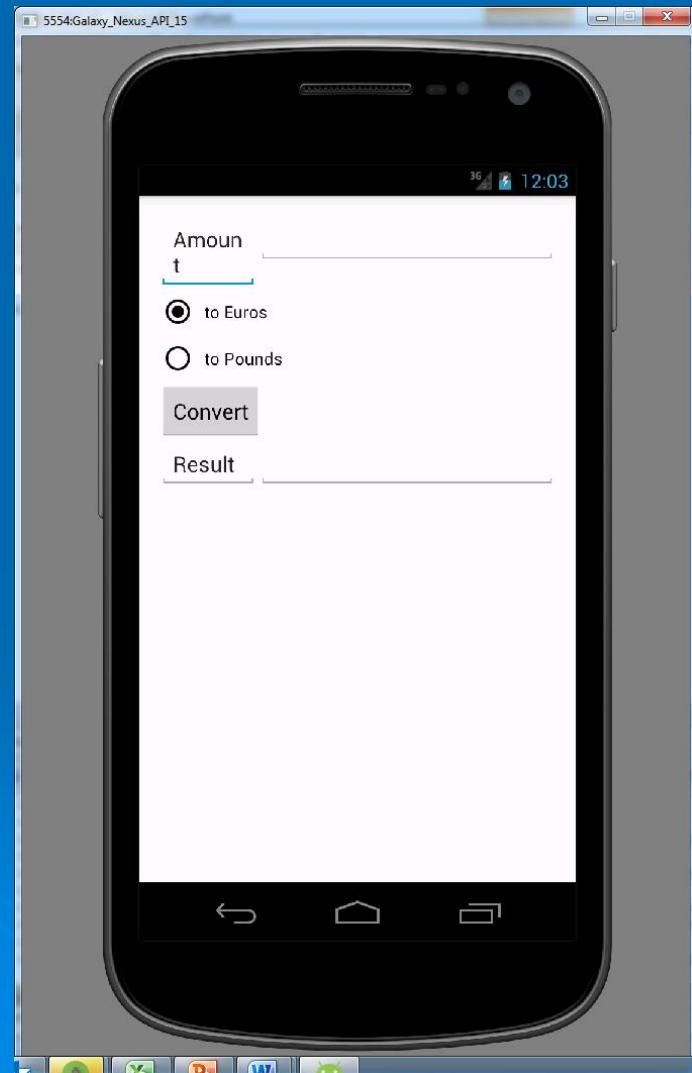
For information about how to use the action bar, including how to add action items, navigation modes and more, read the [ActionBar API guide](#).

Summary

AppCompatActivity

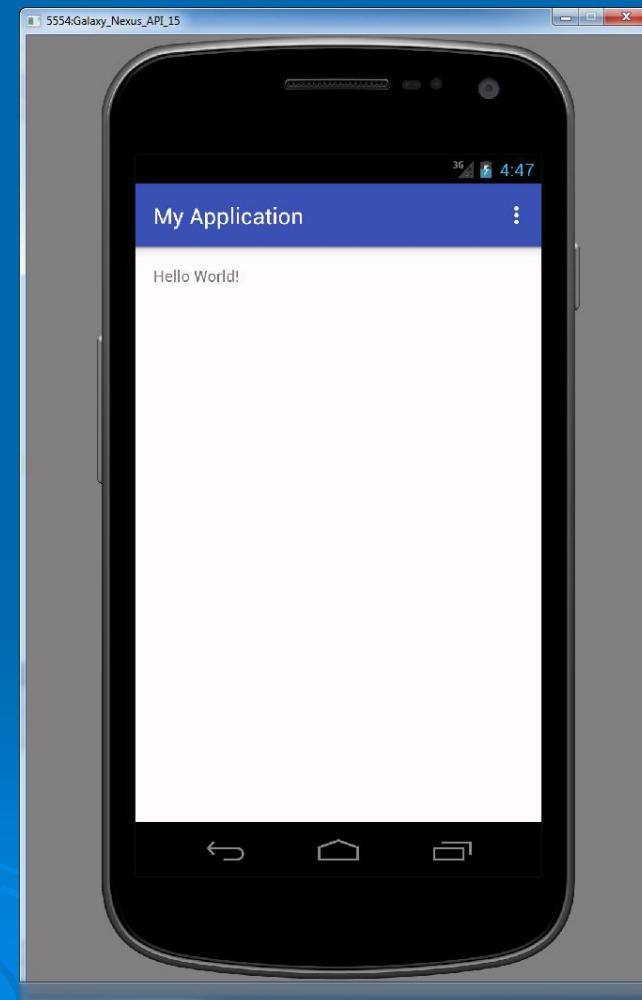
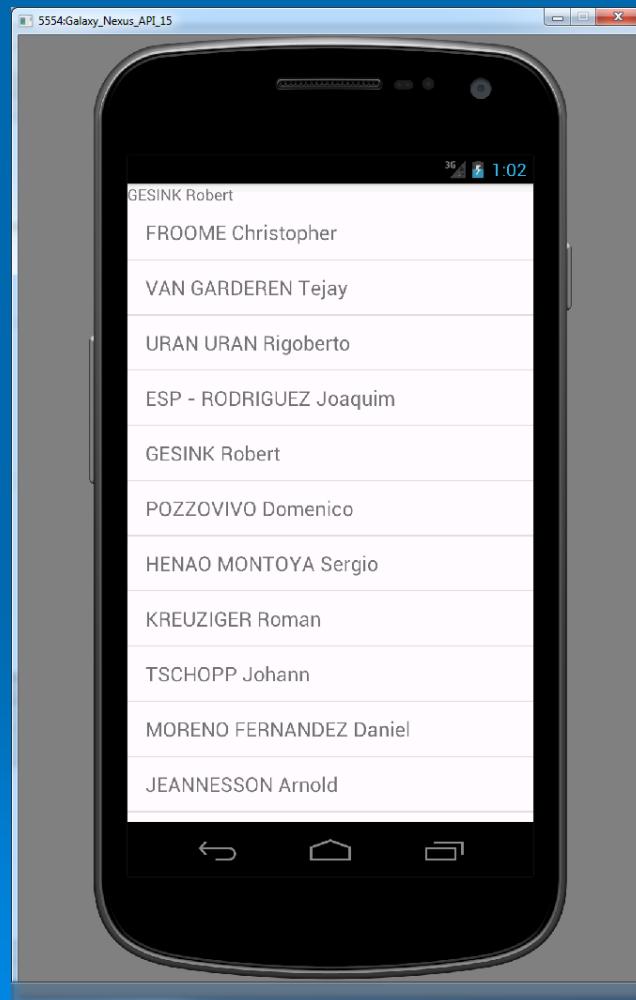


Activity



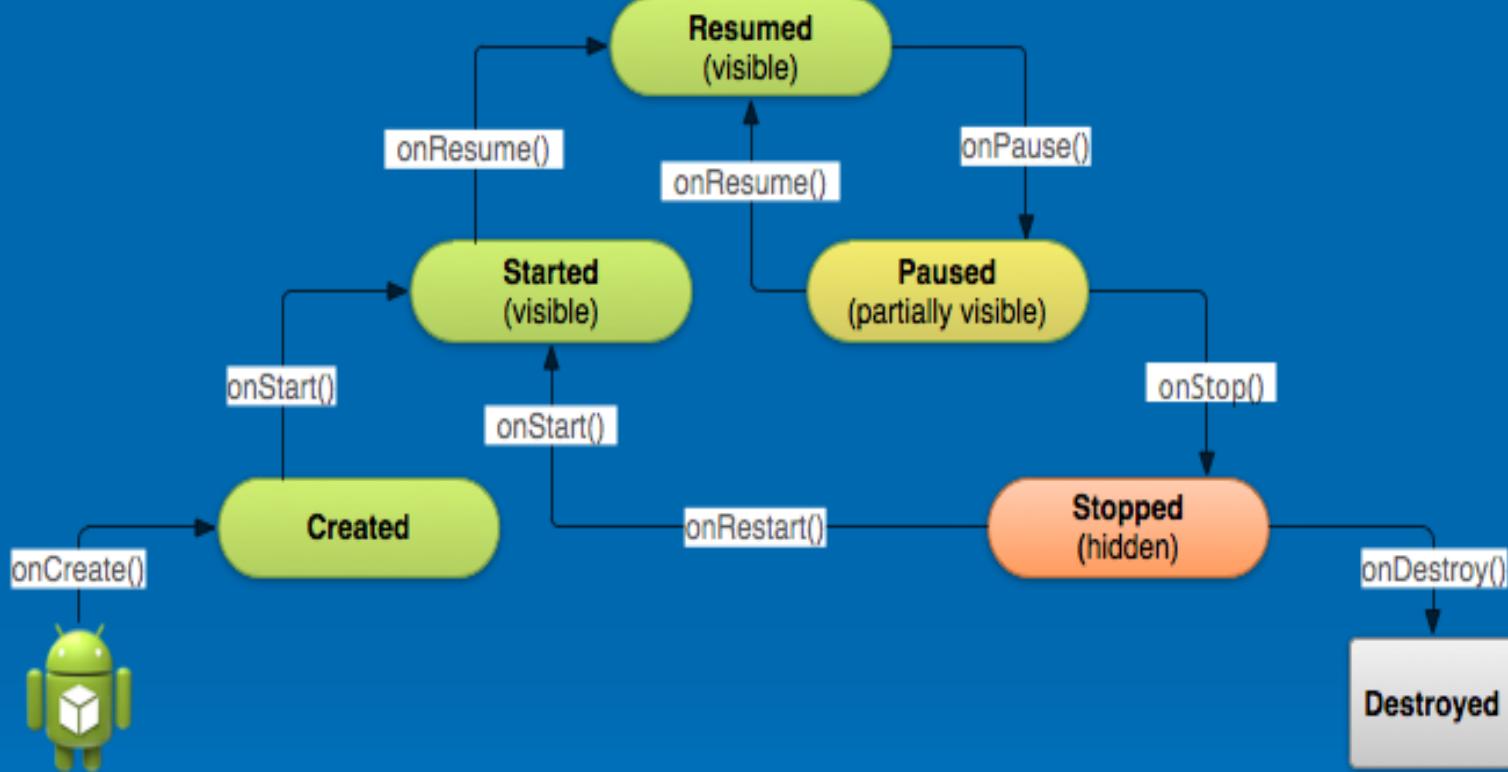
List Activity

AppCompatActivity with menu



Activity Life Cycle Events

- **Resumed (Active):** The activity was started by the user, is running, and is in the foreground.
- **Paused:** The activity was started by the user, is running, and is visible, but a notification or something is overlaying part of the screen.
 - During this time, the user can see your activity but may not be able to interact with it (e.g. due to incoming phone call, battery low alert).
- **Stopped:** The activity was started by the user, is running, but is hidden by other activities that have been launched or switched to.
- **Dead:** Either the activity was never started (e.g., just after a phone reset) or the activity was terminated, perhaps due to lack of available memory.



- <http://developer.android.com/training/basics/activity-lifecycle/startin...>

onCreate() and onDestroy()

- We have been implementing onCreate() in all of our Activity subclasses in all the examples. This method will be called in three situations:
 - When the activity is first started (e.g., since a system restart)
 - If the activity had been running, then sometime later was killed off, onCreate() will be invoked with the Bundle from onSaveInstanceState() as a parameter
 - If the activity had been running and you have set up your activity to have different resources based on different device states
- onDestroy() may be called when the activity is
 - shutting down, either because the activity called finish()
 - or because Android needs RAM and is closing the activity prematurely.
 - onDestroy() may not be called if the need for RAM is urgent (e.g., an incoming phone)

View Class

- This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for *widgets*, which are used to create interactive UI components (buttons, text fields, etc.).
- <http://developer.android.com/reference/android/view/View.html>
- Extends object
- Has subclasses
 - ImageView
 - SurfaceView
 - TextView
 - Etc.
- ImageView has subclass ImageButton
- TextView has subclasses Button, EditText etc
- Button has subclasses CheckBox, RadioButton etc

Numerical Input from an EditText

- In the onClick() method
- EditText weightText = (EditText)findViewById(R.id.inWeight);
- Text property
- weightText.getText()
- Convert text to String to floating point number
- float myWeight = Float.parseFloat(weightText.getText().toString());

Output in EditText

- bmiResult a floating point number
- EditText bmiResult = (EditText)findViewById(R.id.bmiResult);
- Take value of float, convert to string and put in text property of EditText
-
- bmiResult.setText(String.valueOf(bmi));