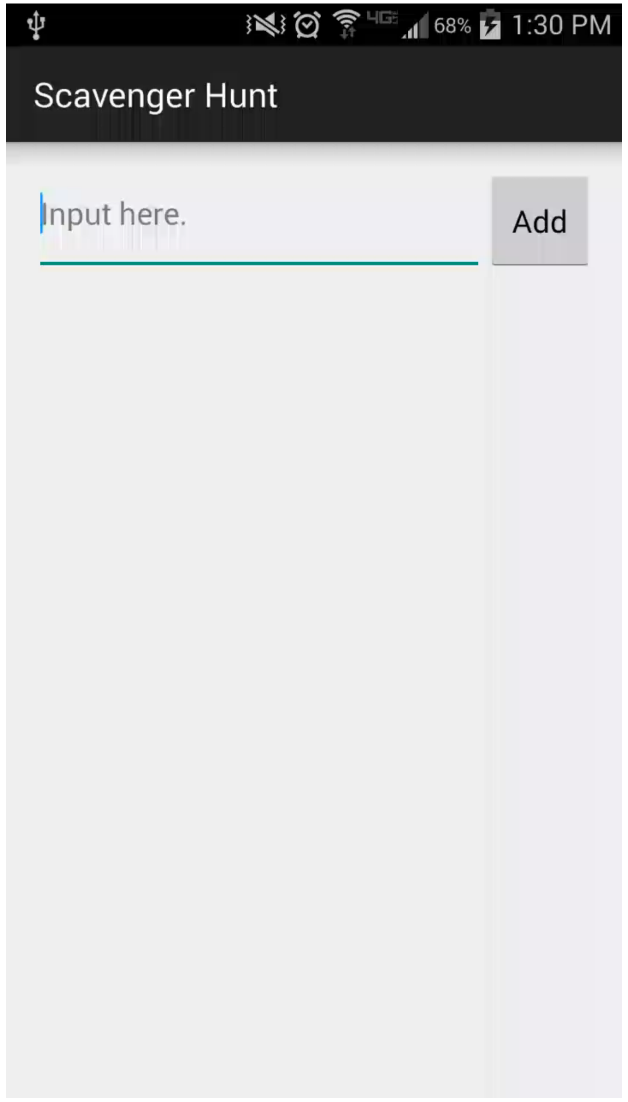
Intro to Android

Max Smith

Download Android Studio NOW

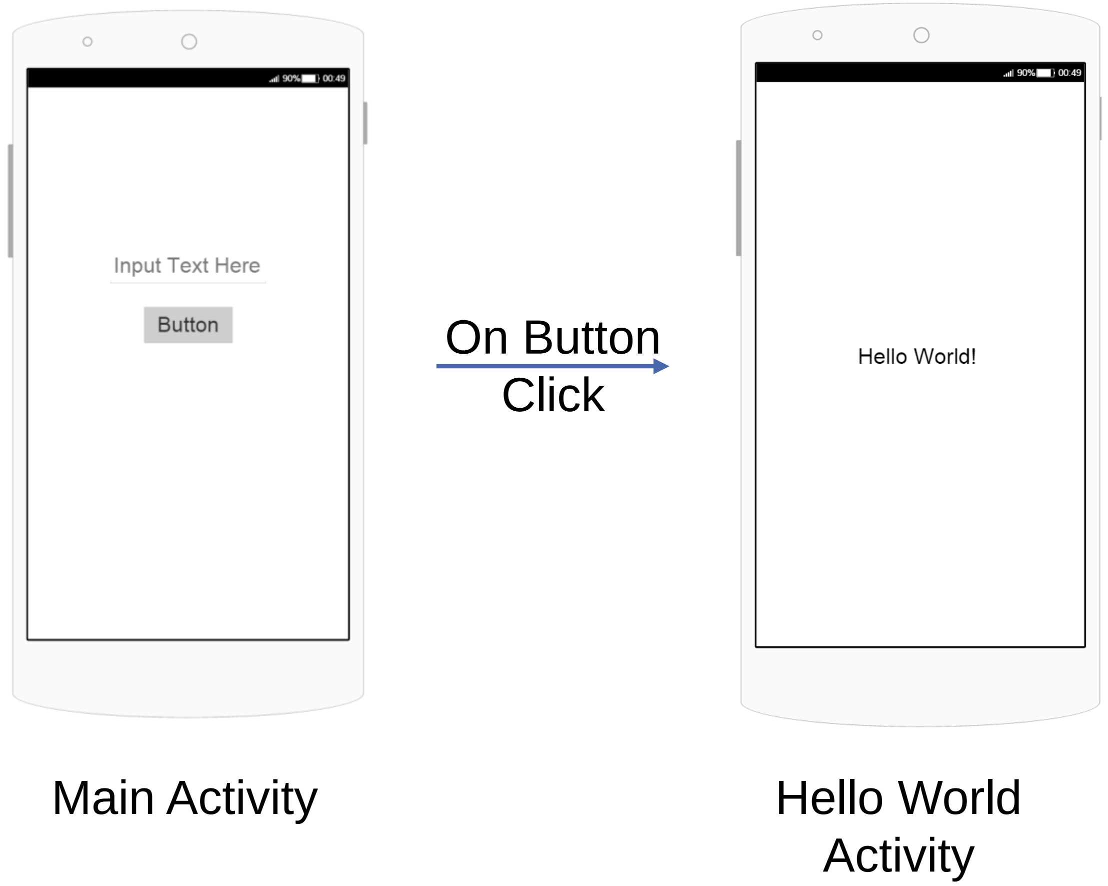

68% 1:30 PM
Scavenger Hunt
Input here.
Add

Overview

1. Fundamental Android Concepts
2. Scavenger Hunt
    a) Tools we will need
    b) Building the App
3. Concluding Remarks

Activity

- A single screen
- Several may be open at once
  - Not all are necessary visible
  - Analogous to opening new windows
- Follows a particular life-cycle



Activity

- Represented by a class

```
public class ExampleActivity extends Activity {









}
```

Activity

- Represented by a class, which has special functions

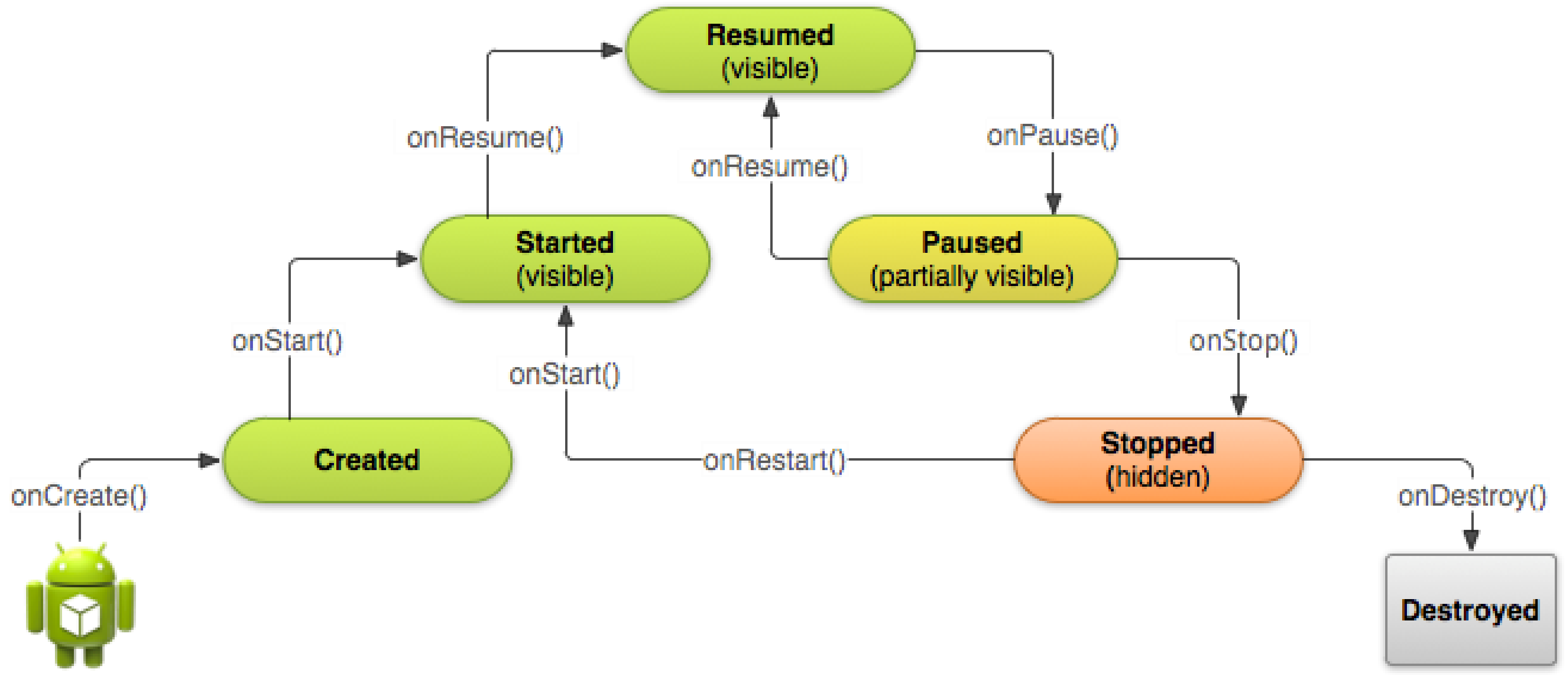
```
public class ExampleActivity extends Activity {

    @Override public void onCreate(Bundle savedInstanceState);







}
```

Activity

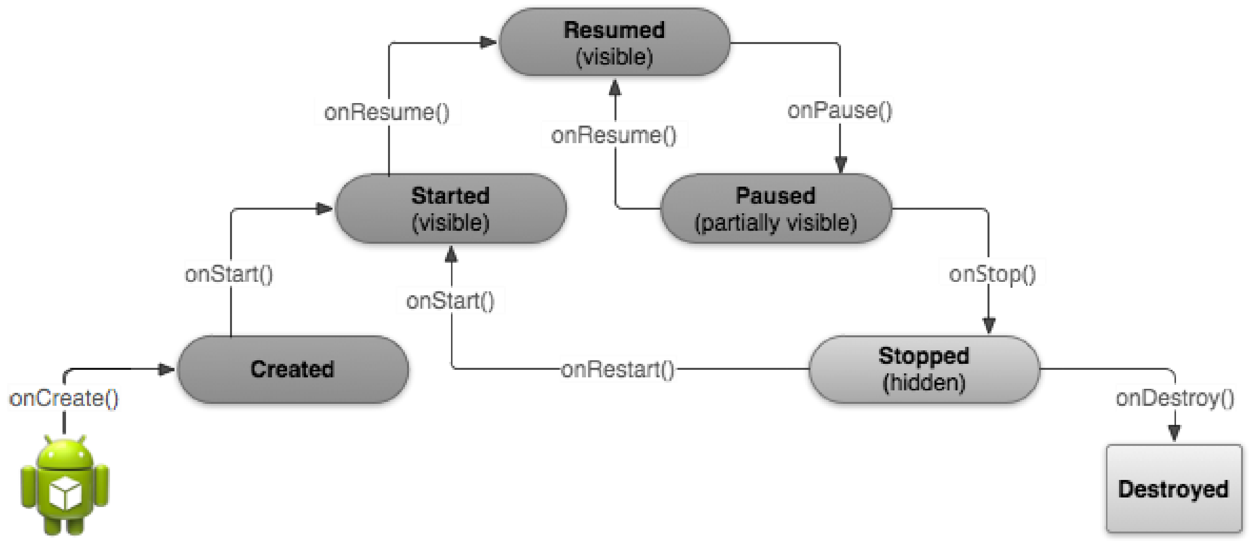
- Represented by a class, which has LOTS of special functions

```
public class ExampleActivity extends Activity {

    @Override public void onCreate(Bundle savedInstanceState);

    @Override protected void onStart();

    @Override protected void onResume();

    @Override protected void onPause();

    @Override protected void onStop();

    @Override protected void onDestroy();
}
```

Activity – Life Cycle

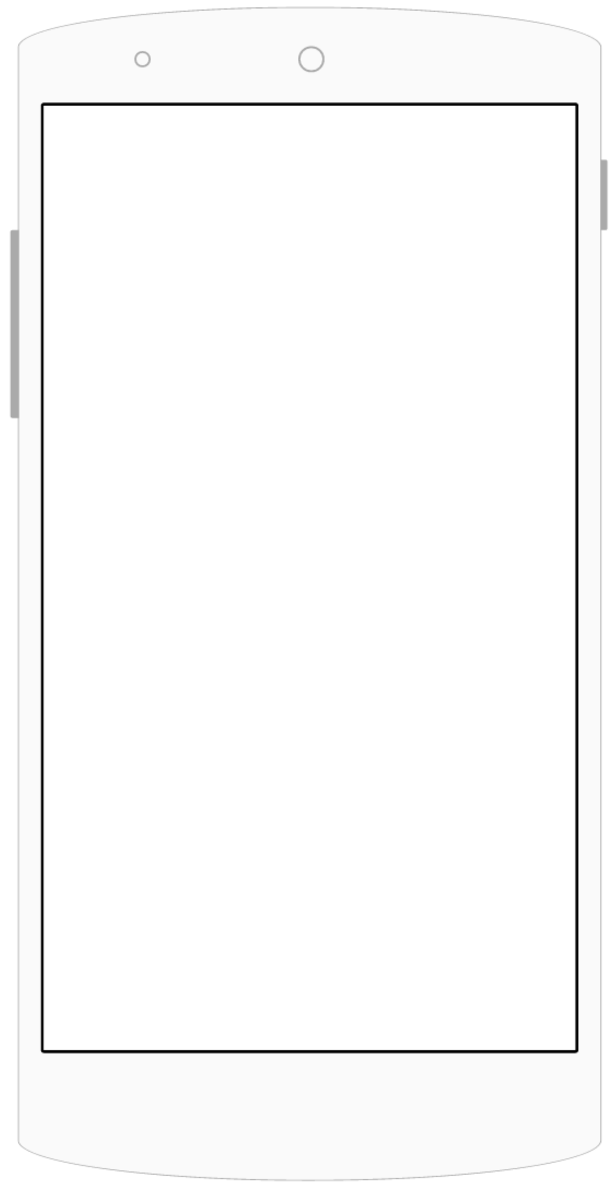


Activity – Life Cycle – Example

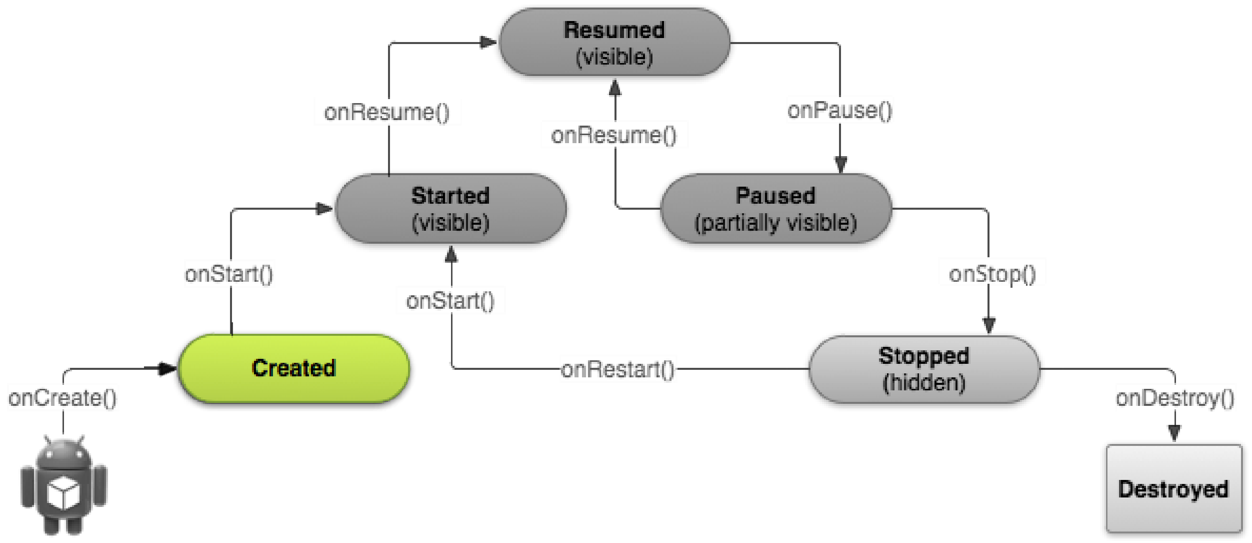


Main Activity

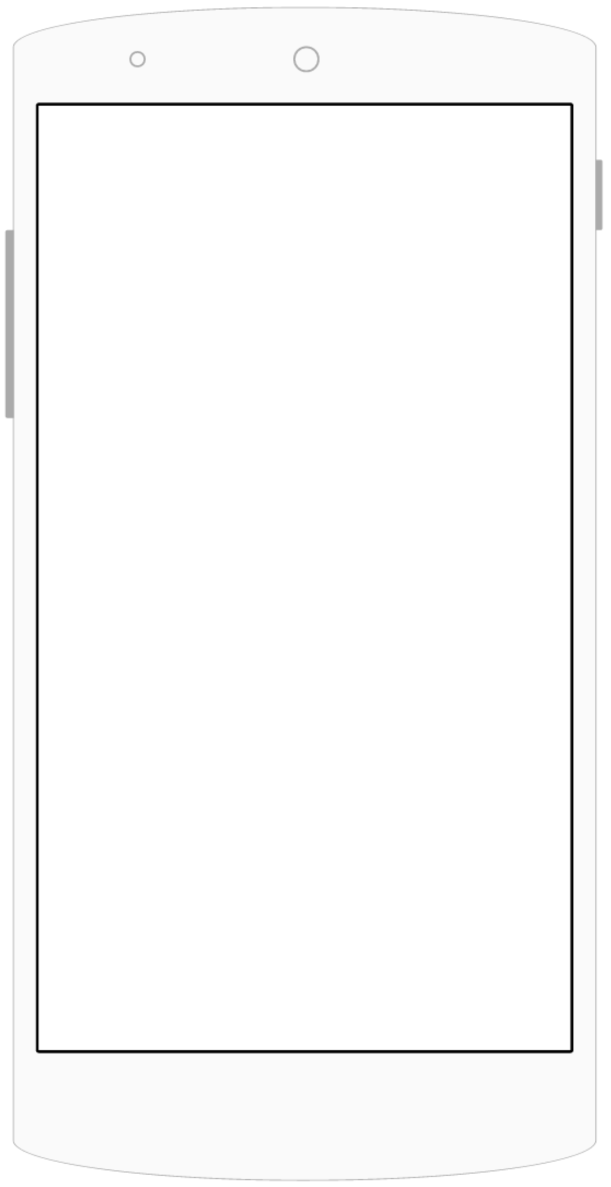
1. Select application



Activity – Life Cycle – Example

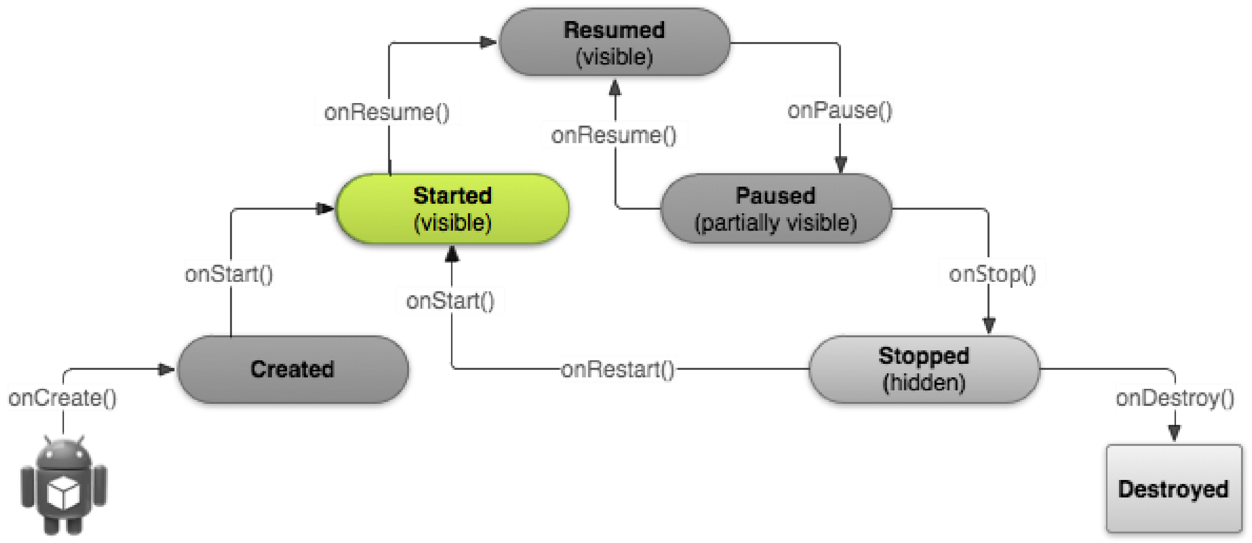


Main Activity



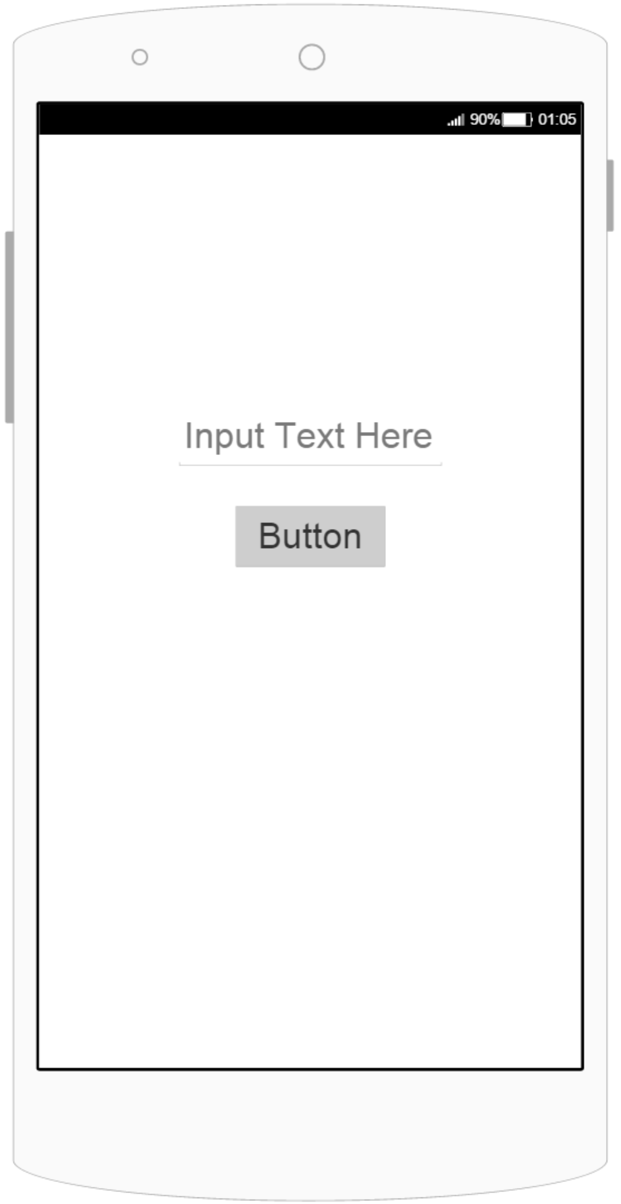
2. Activity initializes
   - UI isn't loaded
   - Slow functions here will cause the UI to not appear for a long time (BAD)

Activity – Life Cycle – Example

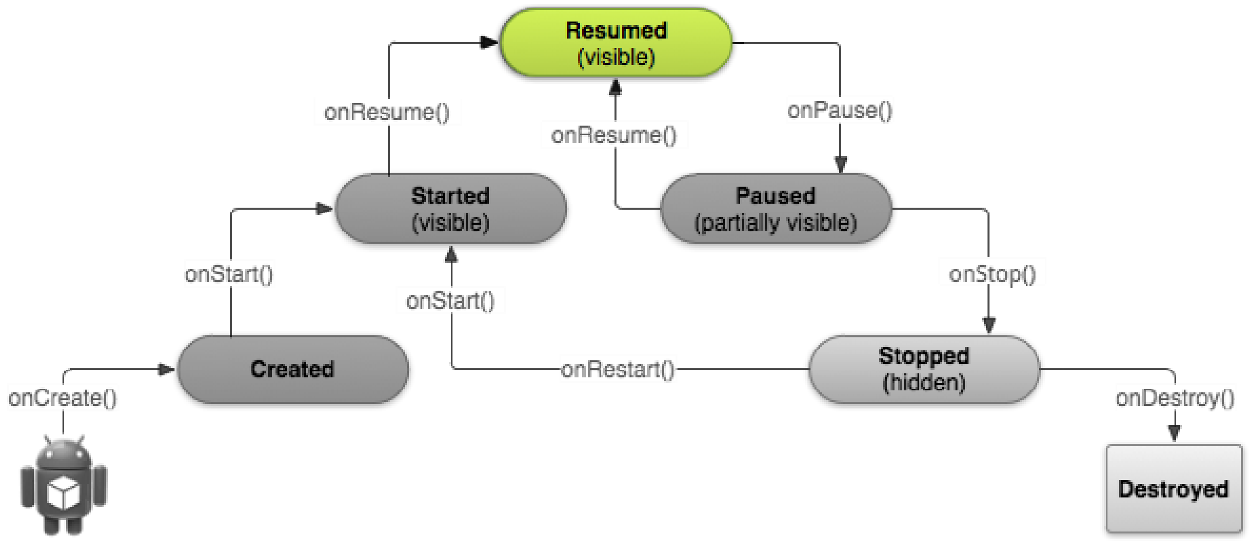


Main Activity

3. Activity Starts & UI Initializes
   - Activity is “frozen” – Needs to be resumed
   - Slow functions here will cause the UI to not appear for a long time (BAD)



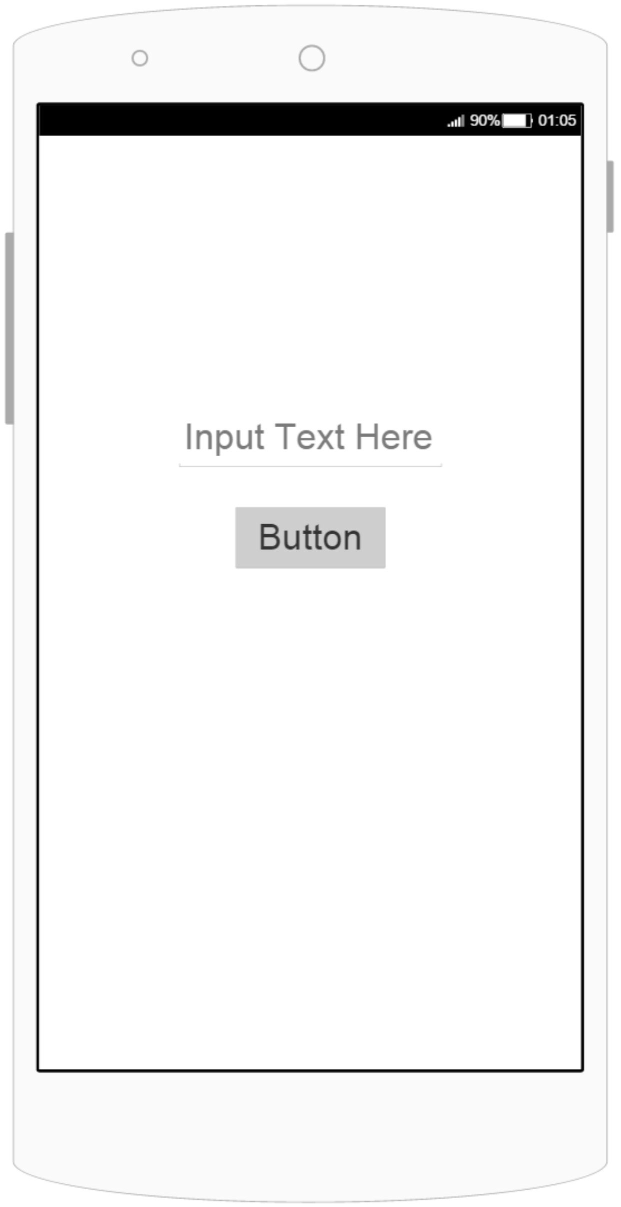
Activity – Life Cycle – Example



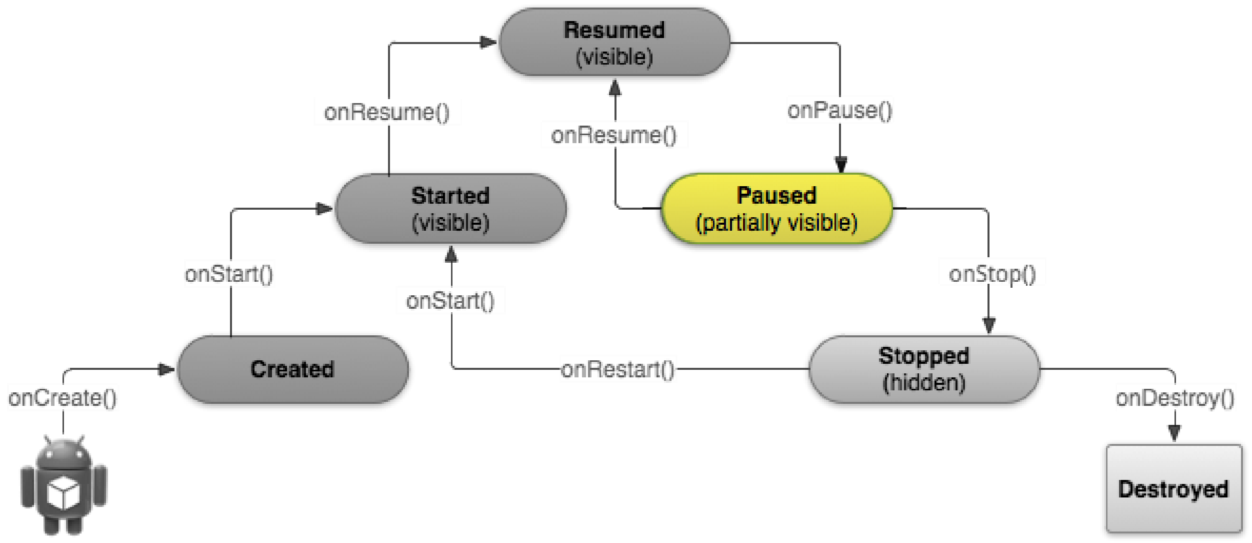
Main Activity

4. Ready to go

What happens when we click the button?

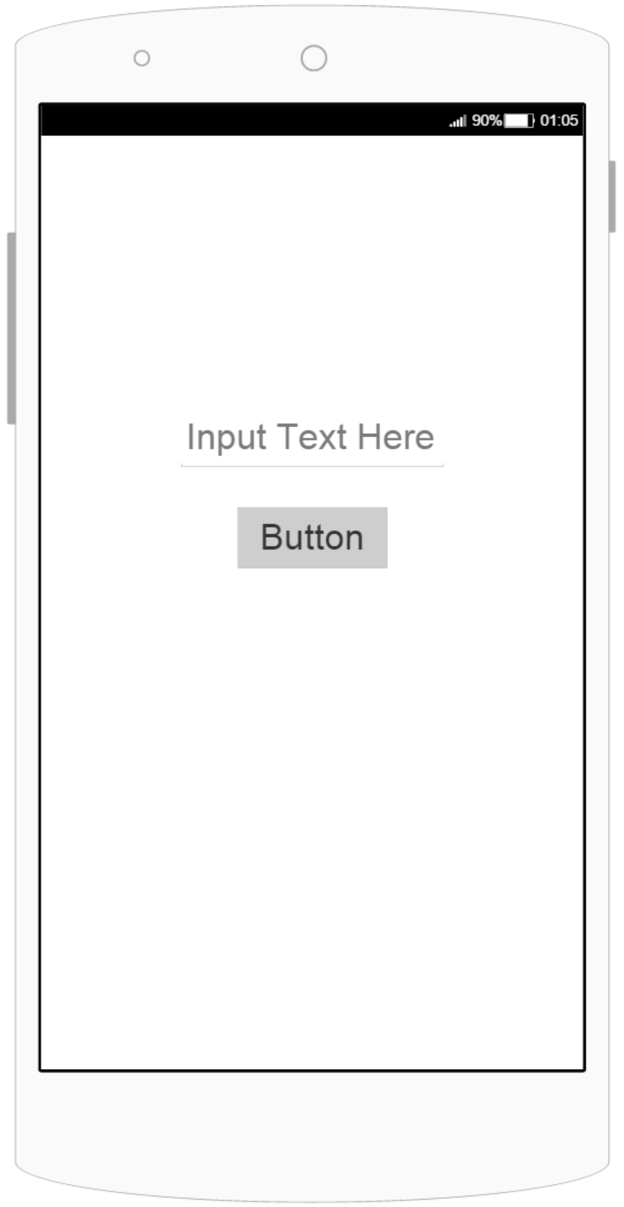


Activity – Life Cycle – Example

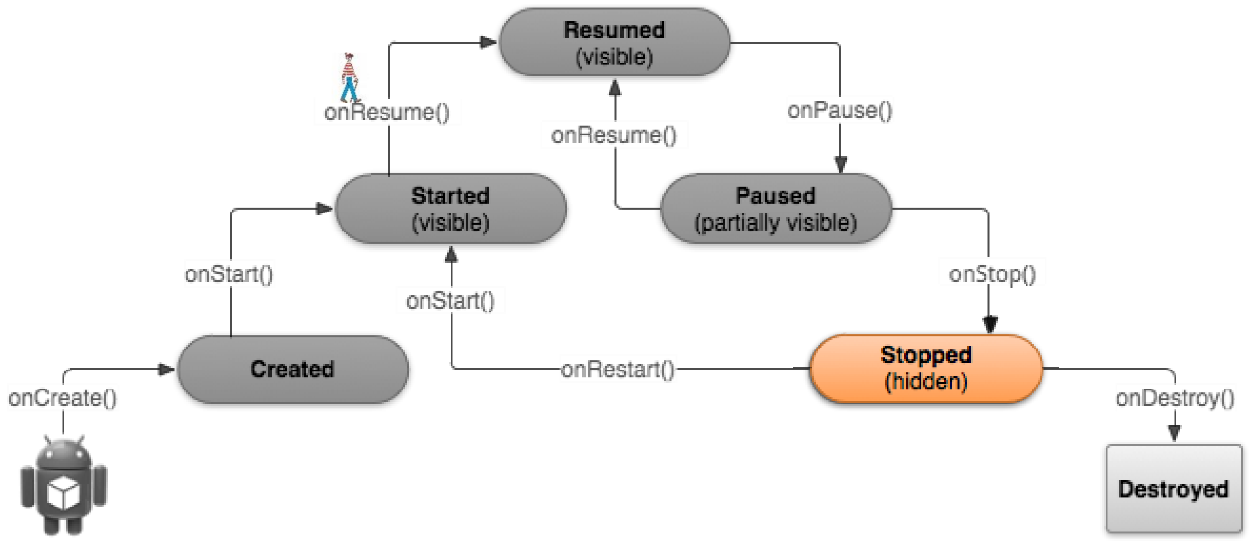


Main Activity

5. Partially visible, (usually) preparing to stop
   - Release memory
   - Save changes

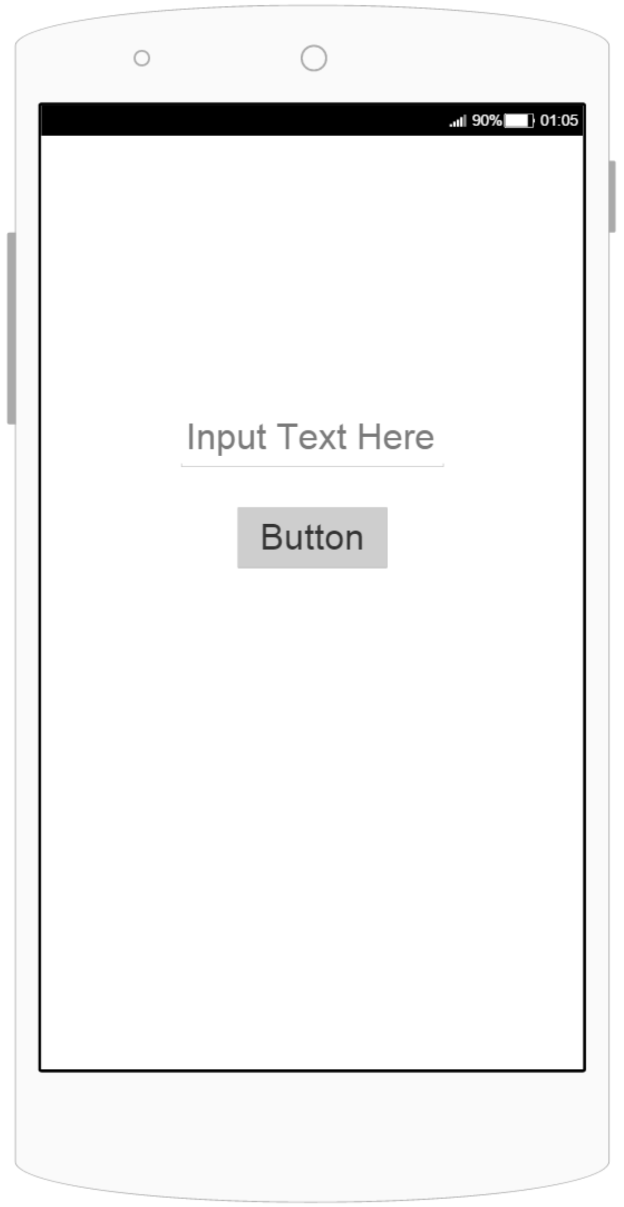


Activity – Life Cycle – Example

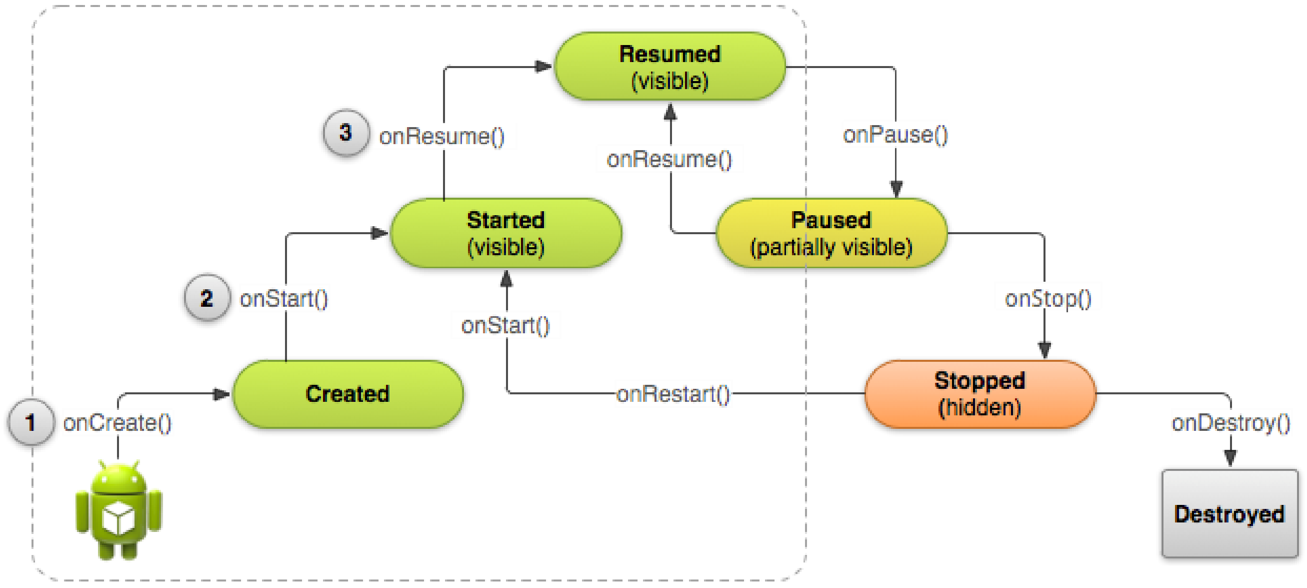


Main Activity

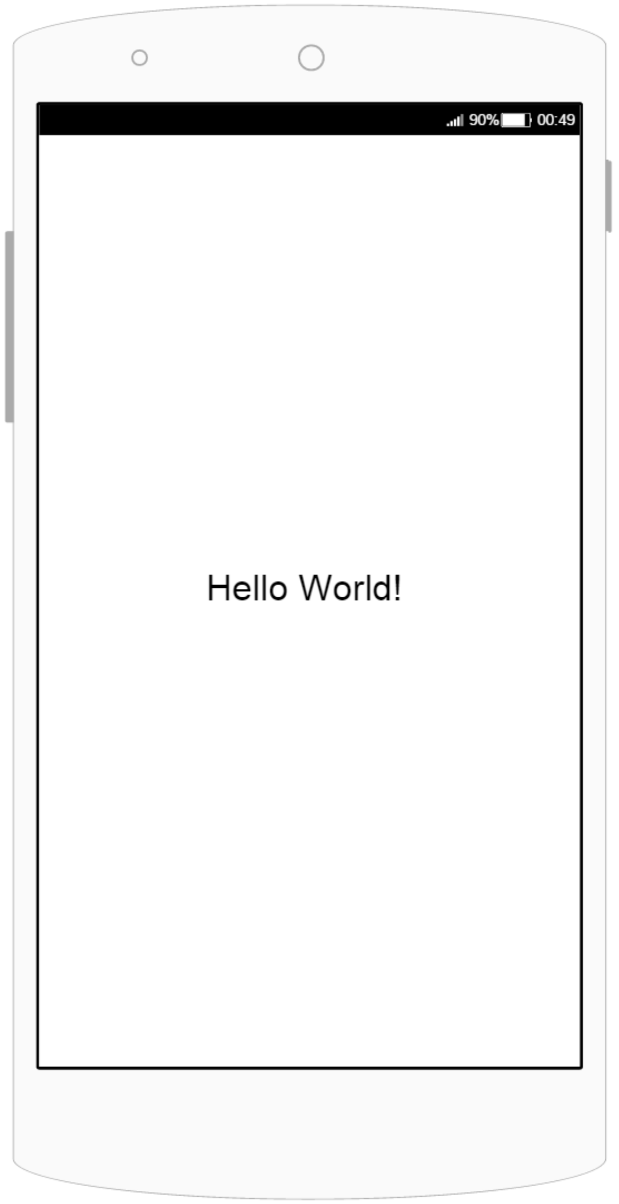
6. Stopped, will be hidden by new activity



Activity – Life Cycle – Example



Hello World Activity

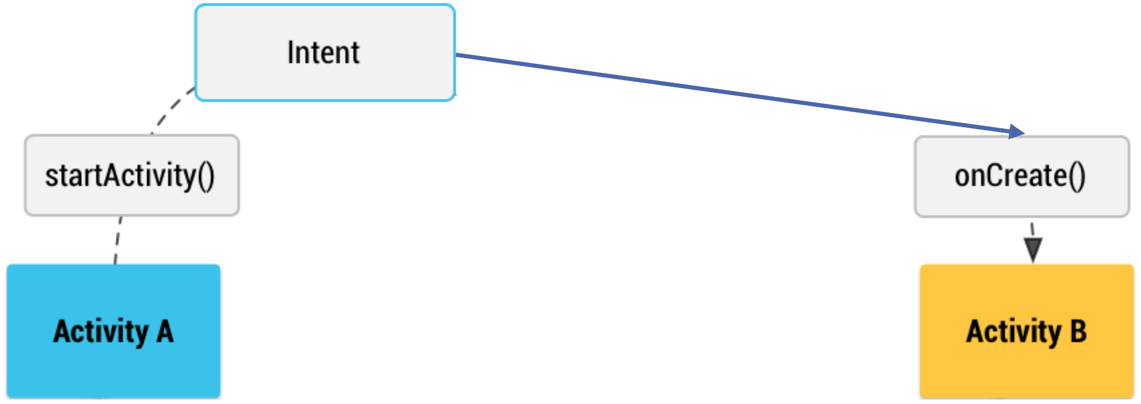


7. New activity starts
   - Create, Start, Resume called in quick succession
   - Activity is ready

Intent

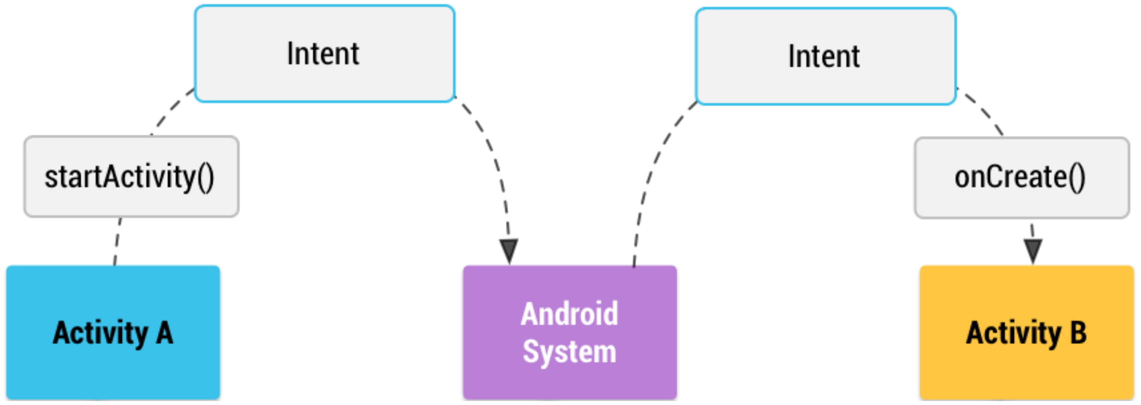
- **Explicit**: specify by name the activity you want to start

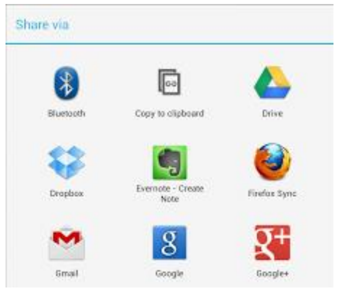
```
Intent myIntent = new Intent(this, ActivityB.class);
startActivity(myIntent);
```



Intent

- **Implicit**: specify a *general* action for the activity to perform
  - You can define an activity to have general actions in the manifest
  - Selection is handled by Android System





Intent

- **Implicit**: specify a *general* action for the activity to perform
  - You can define an activity to have general actions in the manifest
  - Selection is handled by Android System

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://shoulditake381.com/"));
startActivity(i);
```

Resources

- All external resources and constants should be stored in Resources
  - Images, Videos, Strings, Dimensions, Layouts, etc.
  - Similar to statics
- Each resource type has it’s own file
- Defined in XML files

```
Project/
    src/
        MainActivity.java
    res/
        drawable/
            banana.png
        layout/
            main.xml
            row.xml
        values/
            strings.xml
            colors.xml
            arrays.xml
```

Resources

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
        <string name=“greeting">Hello World!</string>
</resources>
```

Resources

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
        <color name=“maize">#ffcb05</color>
        <color name=“blue">#00274c</color>
</resources>
```

Resources

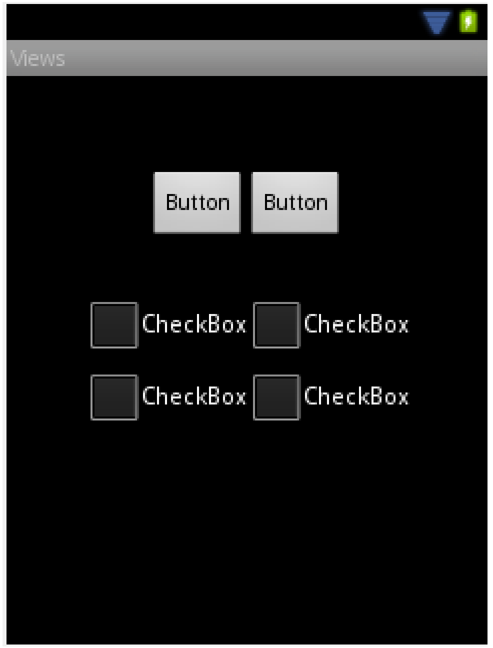
- How do I access my resources?
  - Using a unique identifier that android creates: **R.<subclass>.<name>**

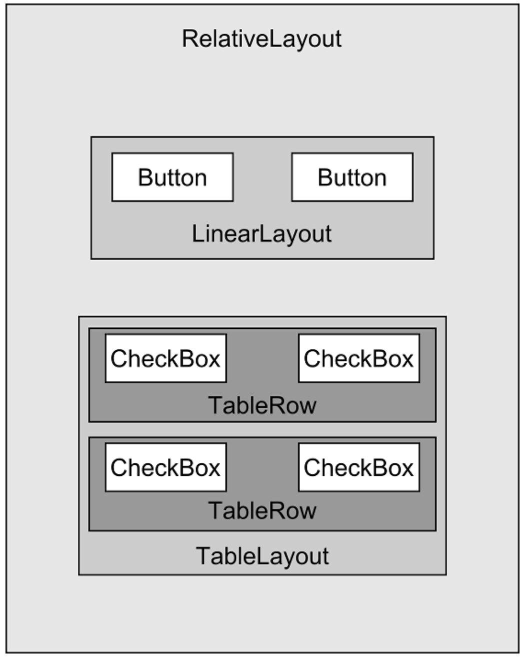
- All resources are packed into a resource class called **R**
- Subclass corresponds with the type of resource: string, drawable, id, etc.
- Name is defined in the XML files

Android Studio

Views & ViewGroups

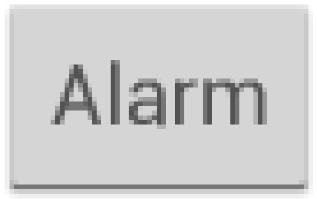
- **View**: one particular UI element
  - Eg. Button, text view, edit text
- **ViewGroups**: groups of UI elements
  - Eg. Linear layout, Table layout, relative layout





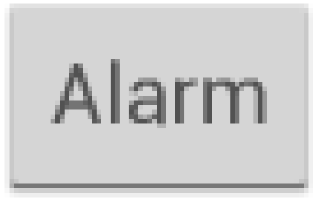
Views – XML

```
<Button android:id="@+id/myID"/>
```



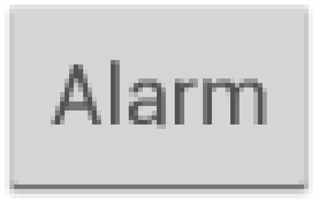
Views – XML

```xml
<Button android:id="@+id/myID"
         android:text="Alarm"
         android:layout_width="@dimen/button_width"
                                                  />
```



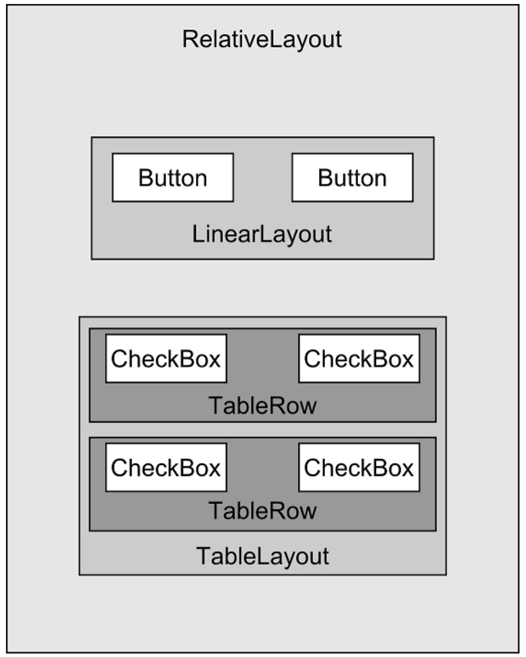
Views – XML

```
<Button android:id="@+id/myID"
         android:text="Alarm"
         android:layout_width="@dimen/button_width"
         android:layout_height="100dp" />
```



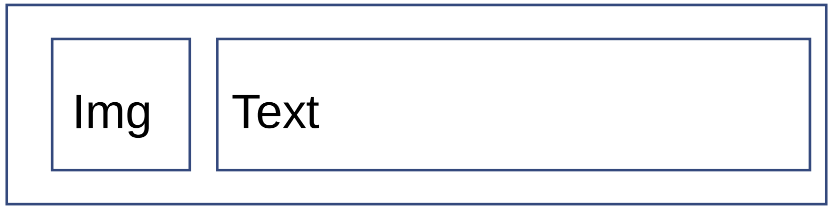
ViewGroup – XML

```
<LinearLayout
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:orientation="horizontal">

     <!-- Put Views Here -->


</LinearLayout>
```



ViewGroup – Exercise

For our scavenger hunt, we need the following layout for our hunt items:



It uses 3 UI objects:

LinearLayout
ImageView
TextView

Assuming the ImageView is 50dpx50dp

```
<Button android:id="@+id/myID"
        android:text="Alarm"
        android:layout_width="@dimen/button_width"
        android:layout_height="100dp" />
```

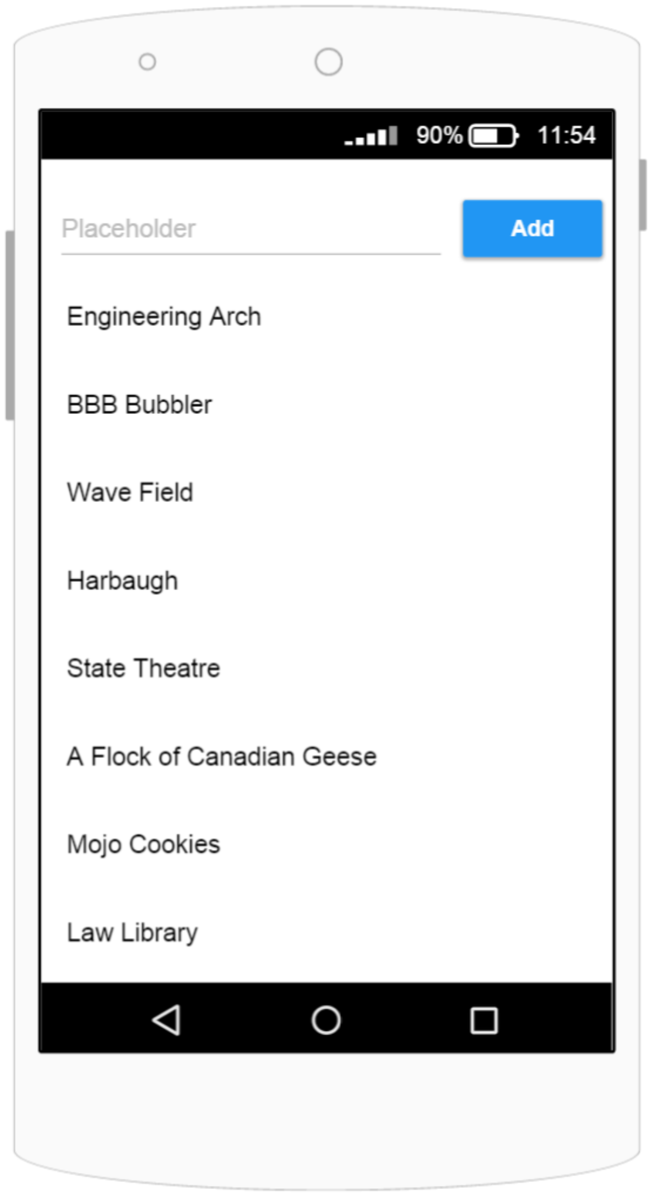
```
<LinearLayout
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:orientation="horizontal">

     <!-- Put Views Here -->


</LinearLayout>
```

Row Item

Scavenger Hunt - Prototyping

- Based on functionality, pick out views that best match your needs
- Mock-up required activities
- Sketch out flow-chart of activities

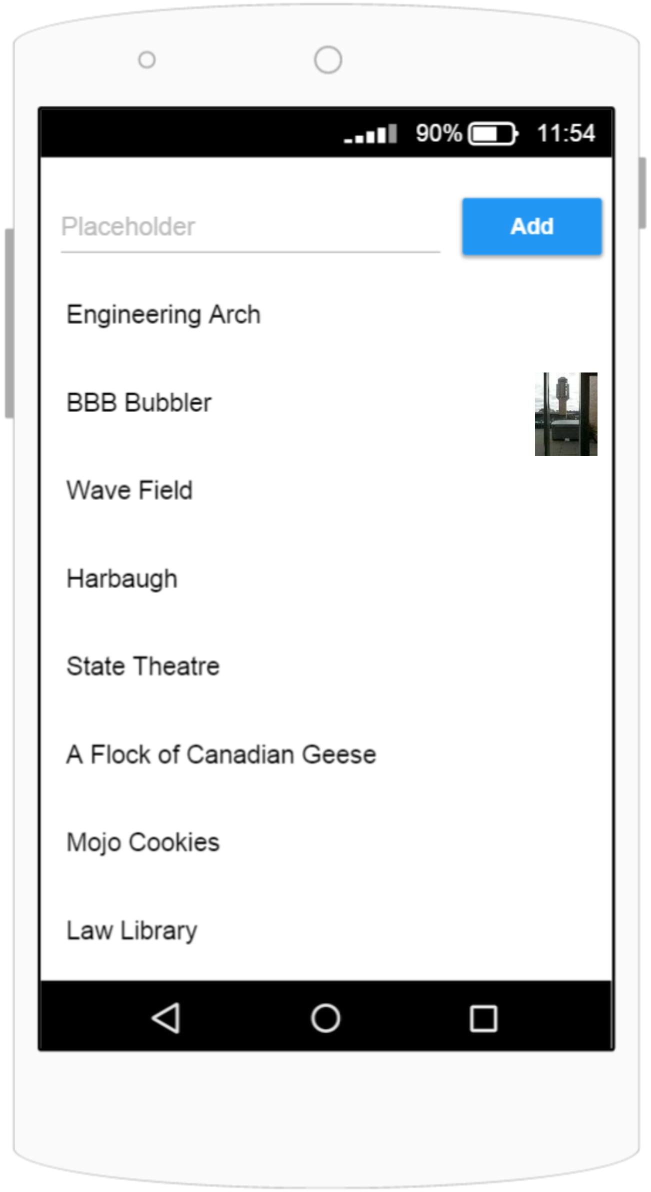
- Online prototyping tool: **proto.io**



Engineering Arch
BBB Bubbler
Wave Field
Harbaugh
State Theatre
A Flock of Canadian Geese
Mojo Cookies
Law Library

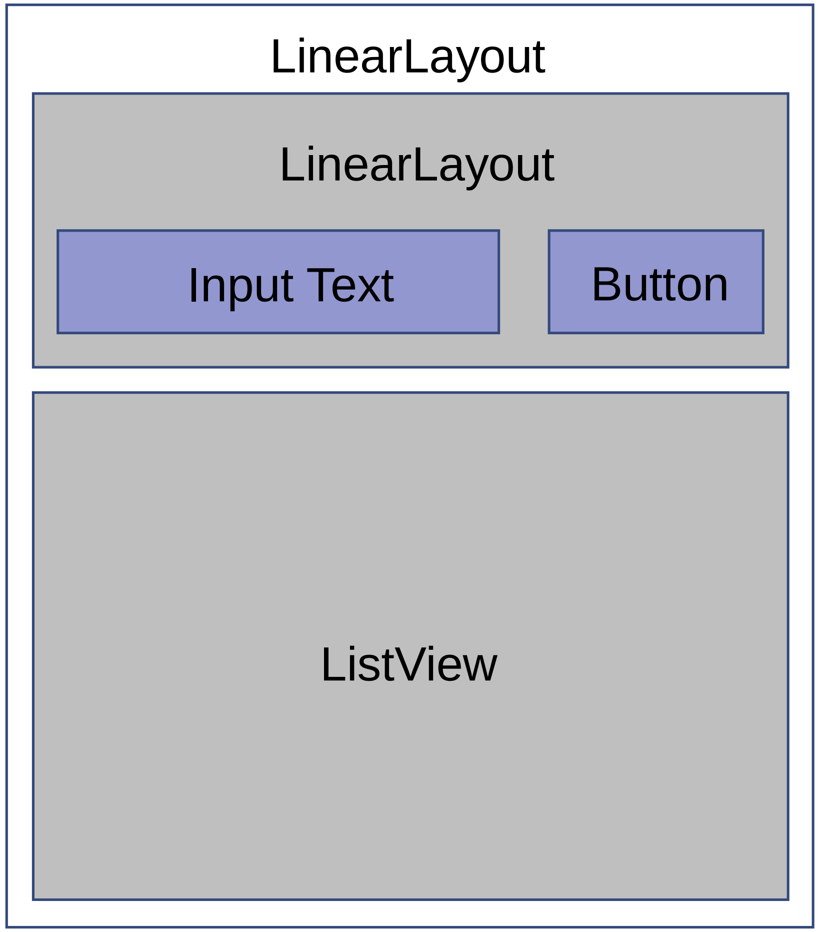
Scavenger Hunt - Layout

- Text box: input item name
- Button: add item to list
- List: all scavenger hunt items



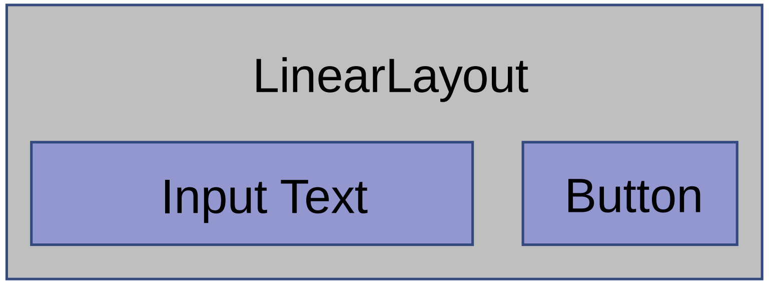
Engineering Arch
BBB Bubbler
Wave Field
Harbaugh
State Theatre
A Flock of Canadian Geese
Mojo Cookies
Law Library

Building the UI



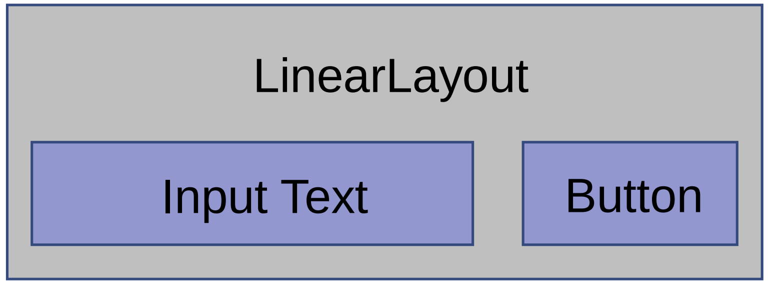
Building the UI

- How do we get the text field and button to share the same line?



Building the UI

- How do we get the text field and button to share the same line?

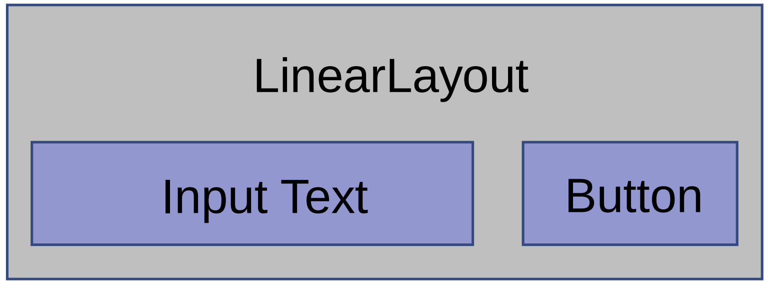


- Assign a **layout_weight**
  - Allows you to specify the ratio of the layout the view should take up

```
android:layout_weight="2"
```

Building the UI

- How do we get the text field and button to share the same line?



- Assign a **layout_weight**
  - Allows you to specify the ratio of the layout the view should take up

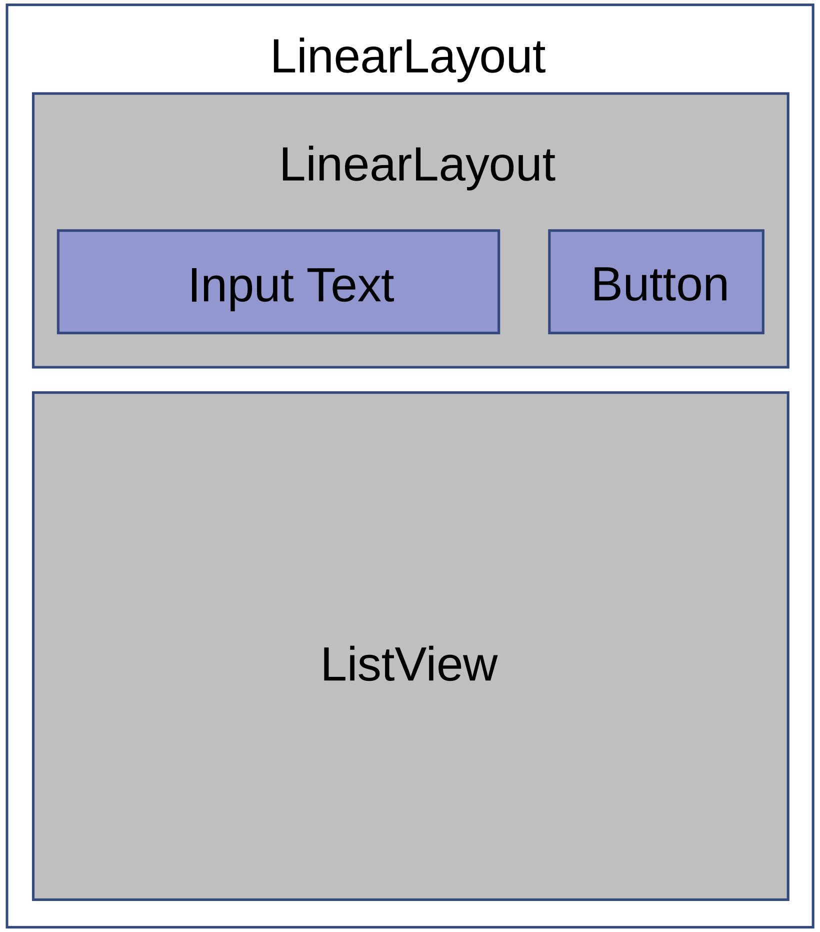
```
android:layout_weight="2"
```

  - Eg. If inputText's weight is 4, and button's weight is 2:
    - InputText will take up 4/(4+2) = 4/6 of the layout
    - Button will take up 2/(4+2) = 2/6 of the layout

Building the UI – Exercise

```
<Button android:id="@+id/myID"
        android:text="Alarm"
        android:layout_width="@dimen/button_width"
        android:layout_height="100dp "

        android:layout_weight="2" />
```

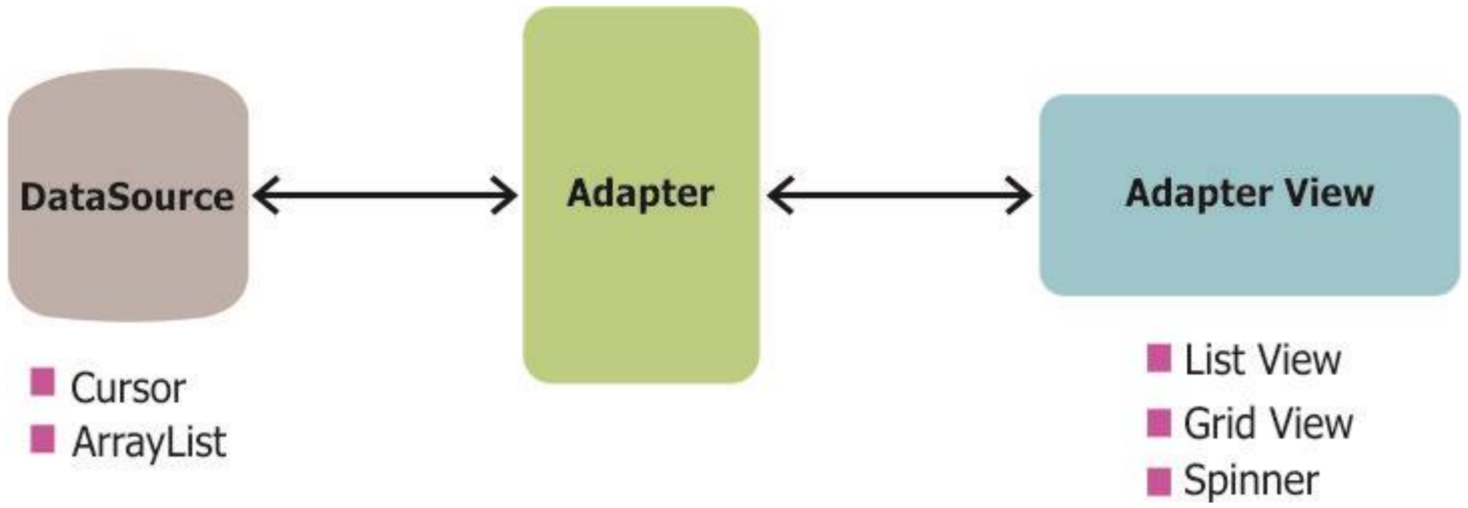
```
<LinearLayout
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:orientation="horizontal">

     <!-- Put Views Here -->


</LinearLayout>
```



Main Activity UI

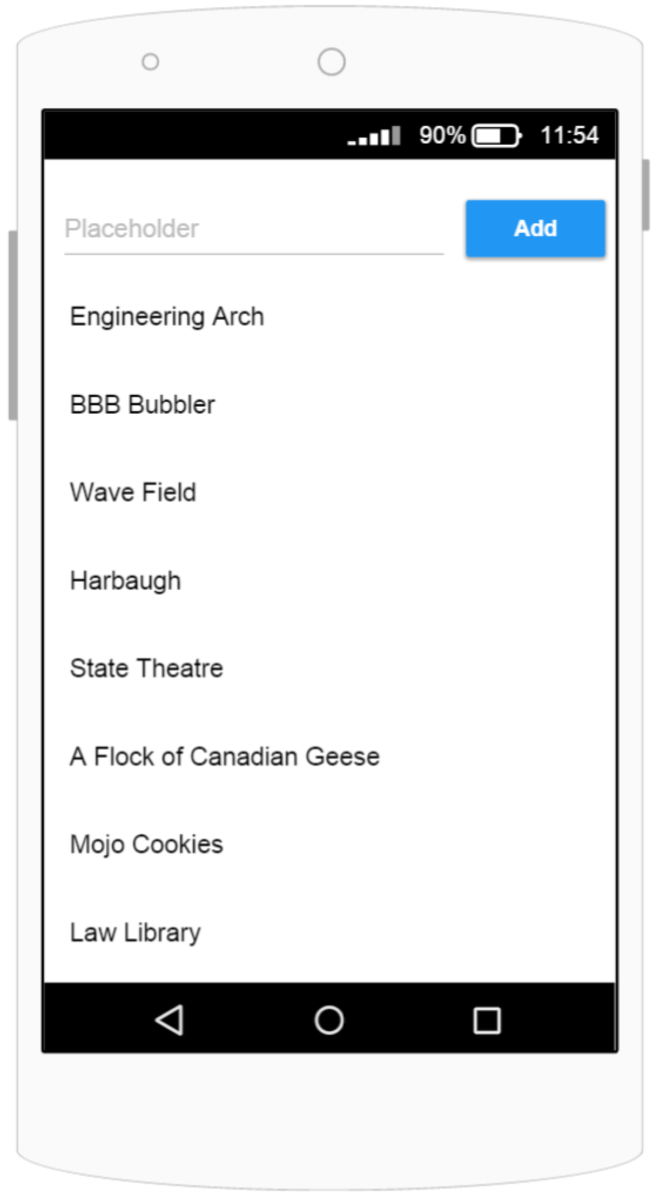
Building a ListView

- A ListView “adapts” a list object into the UI



Building the ListView

- Just the text:
  - List[0] = Engineering Arch
  - List[1] = BBB Bubbler
  - ….



Engineering Arch

BBB Bubbler

Wave Field

Harbaugh

State Theatre

A Flock of Canadian Geese

Mojo Cookies

Law Library

Coding the ListView

```java
// Link UI ListView to variable
ListView list = (ListView) findViewById(R.id.listView);

// Container to hold list objects
ArrayList<String> items = new ArrayList<String>();

// Create an adapter for “items” that map to the UI “row” for this activity
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, R.layout.row, items);

// Attach adapter to list
list.setAdapter(adapter);
```

List View & Camera

goo.gl/i3XXHQ

Final Remarks

- Some topics were briefly covered, due to time constraints
- Please take a quick survey:

http://goo.gl/forms/vYYRkIwMua

- Tell me what you hated, liked, didn’t care about, ate for dinner, etc.
- If you would like more of these, that go into more detail, do it in the survey!
- Any and all feedback will be used to better this talk, and possible future talks

- Did you find Waldo?