CSc 317
Activity LifeCycle, ListView, Adapters

Benjamin Dicken

Announcements

- Quiz 2 grades
- PA 2

A New App

- Create a new Application called ICALifecycle
- Will use for this class and the next

The Manifest File

- From the docs:
  - “*The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.*”

The Manifest File

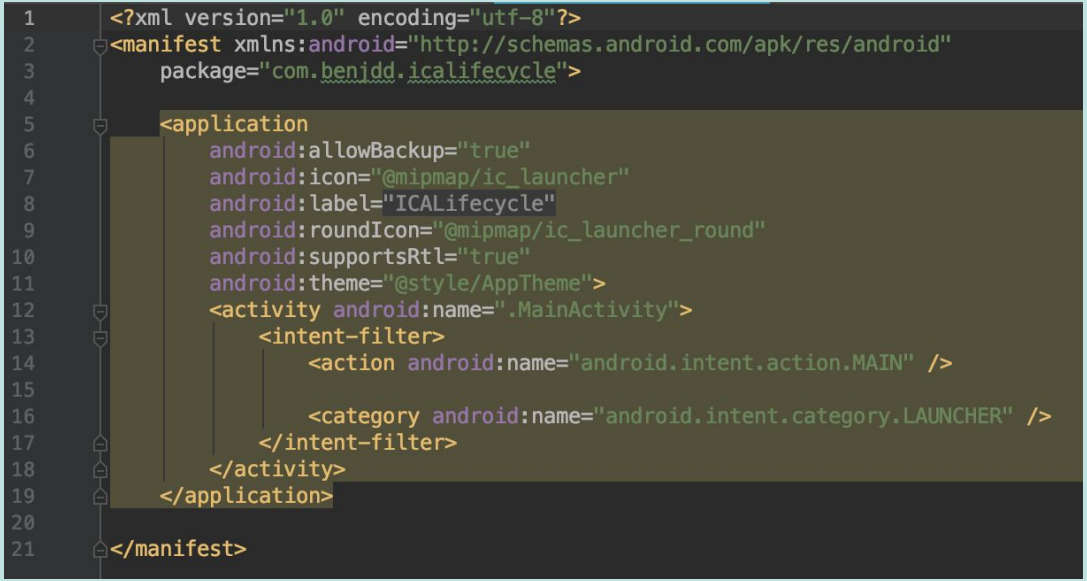
- **Use it for**
  - Setting the package name
  - The various components of the application (such as activities) and how they relate to one-another
  - Application permissions
  - Hardware and software requirements
- Created by default if using Android Studio

Specifying Components

- **From the docs**
  - For each app component that you create in your app, you must declare a corresponding XML element in the manifest file:
    - **<activity>** for each subclass of **Activity**.
    - **<service>** for each subclass of **Service**.
    - **<receiver>** for each subclass of **BroadcastReceiver**.
    - **<provider>** for each subclass of **ContentProvider**.

Look at the manifest file for ICALifecycle

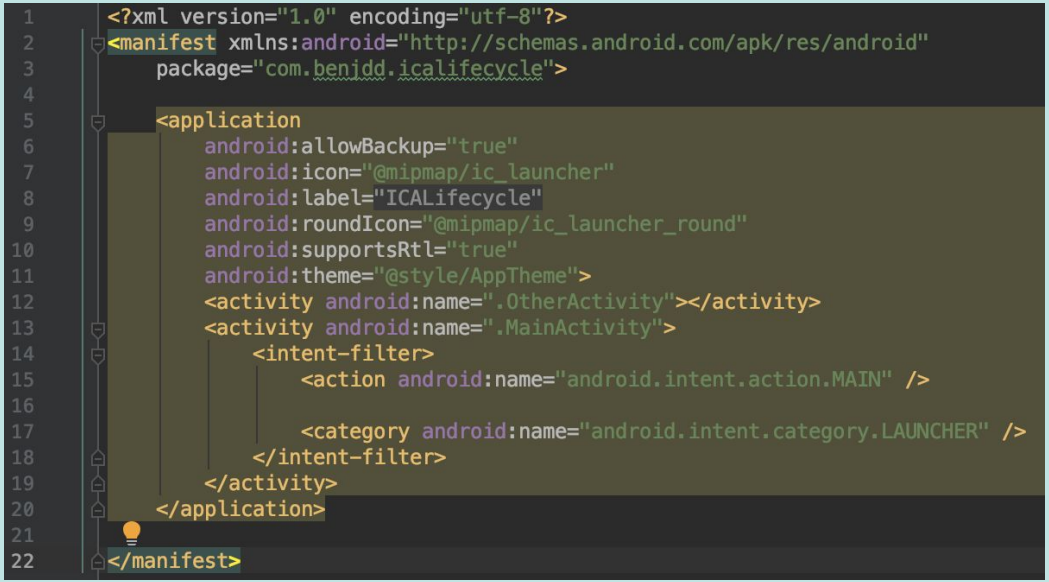
- Notice . . .
- The one **<activity>** tag
- The intent-filter
- The **theme**
- The **icon**



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.benidd.icalifecycle">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ICALifecycle"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Look at the manifest file for ICALifecycle

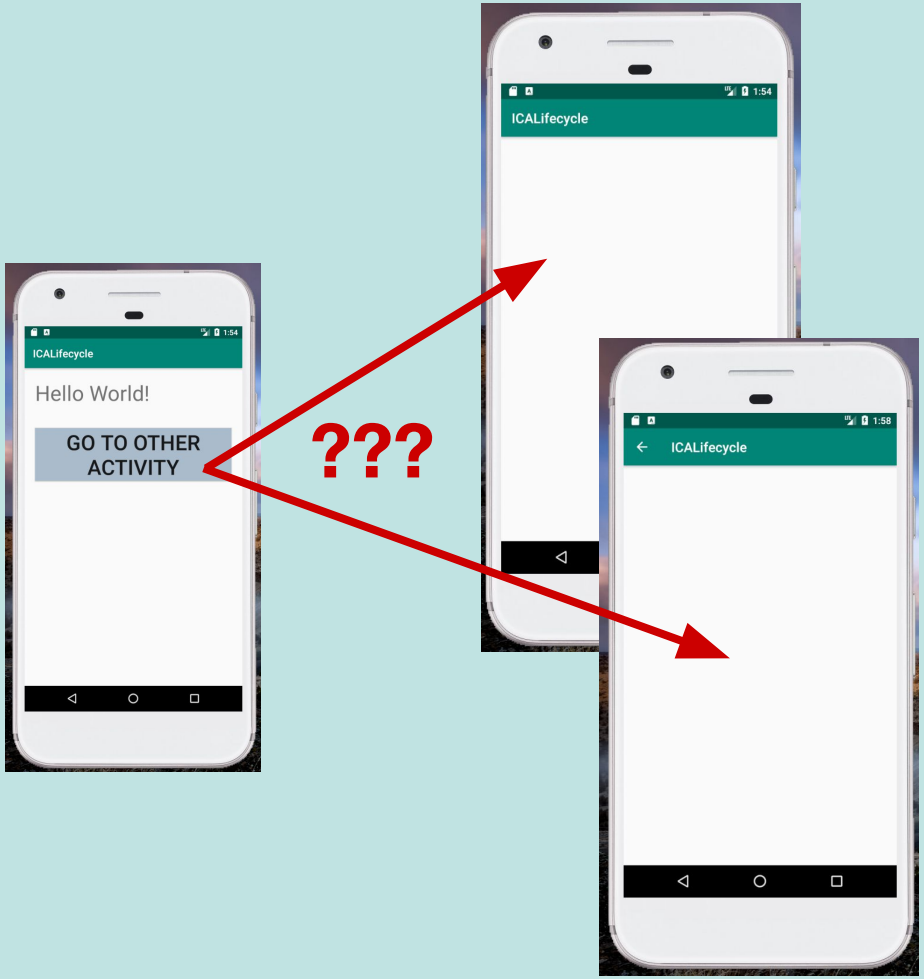
- Create a new activity called **OtherActivity**
- Then, close and re-open the manifest file.



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.benjdd.icalifecycle">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ICALifecycle"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".OtherActivity"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Add a button

- Add a button to take you to the other activity
- What does it look like in practice?
- How to get back to **MainActivity**?



Update the manifest file

```
<activity
    android:name=".OtherActivity"
    android:parentActivityName=".MainActivity">
</activity>
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

Launcher Icon

- The default launcher icon probably looks something like this:

- Let’s change it

Change the icon

- Find a png icon to use on the web
- Drag to mipmap directory, set to proper pixel amount type
- Change Manifest
- Run app, see if new icon is used

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="ICALifecycle"
    android:roundIcon="@mipmap/ica_lifecycle"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

Launcher Icon

- If you want to have your application icon to be correct for many devices, create a version for each pixel density type
- For now, don’t worry about handling them all

Activities

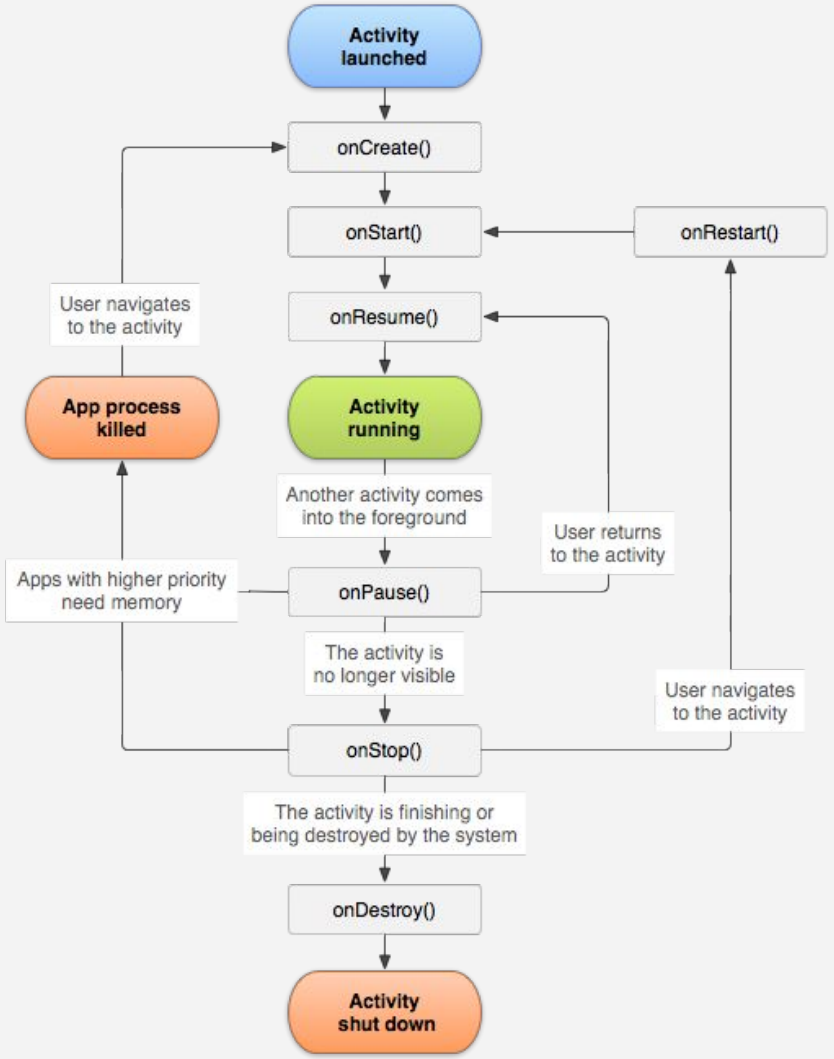
- An application can be composed of a number of **Activities**
- Activities can be initiated via an **Intent**
- An activity from one application can even initiate an **Activity** from another application, via an **Intent**
    - For instance, one app that initiates the camera activity, or opens up an email compose window
- How to determine the various stages of an activity

Activity Lifecycle

- From the docs:
  - As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle.
  - The Activity class provides a number of callbacks that allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.

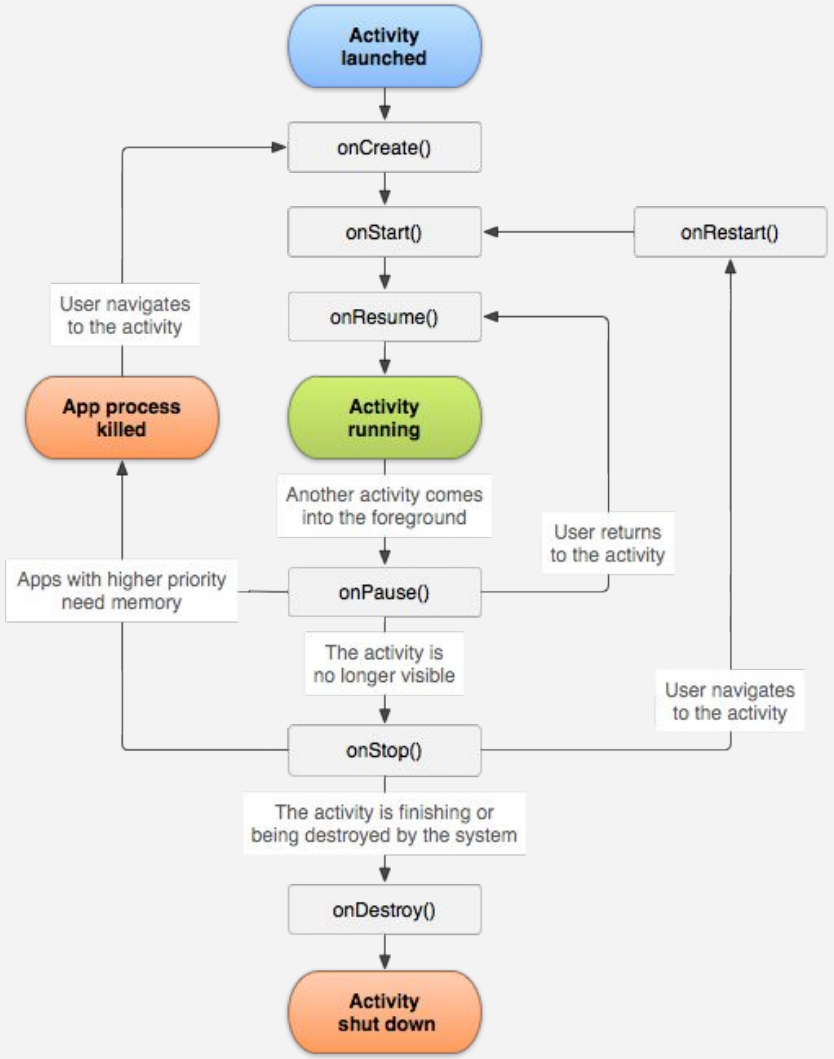
Activity Lifecycle

- There are a number of callback functions that can be implemented to have action(s) happen at various times in the life-cycle
- When you create a new empty activity, **onCreate** is created by default



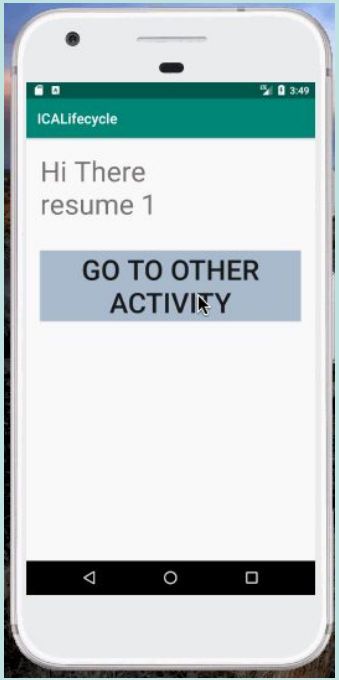
Activity Lifecycle

- *Let's do a few experiments to see how it works*



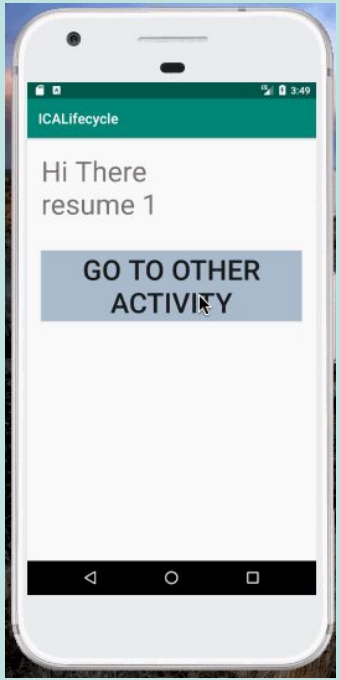
Add **onResume** to MainActivity

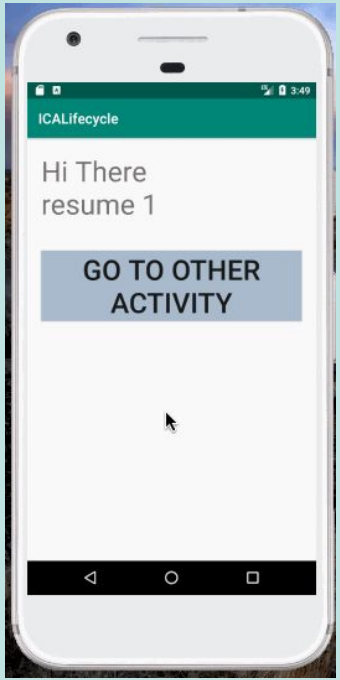
- Add a String member class variable to keep track of the message to be displayed
- Also add a string counter
- Add an **onResume** function that adds to the message string and update the view to show the string



Add **onResume** to MainActivity

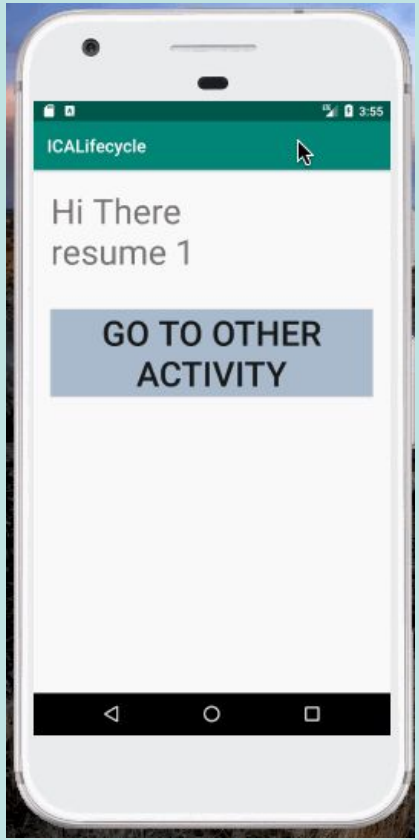
- What is going on here?
- Why the difference?





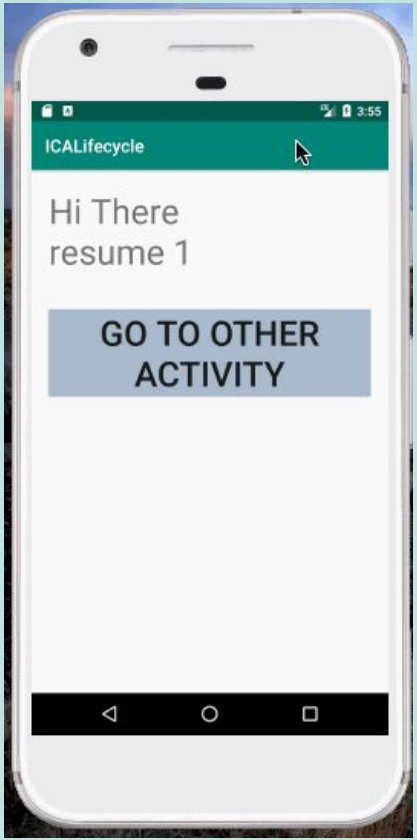
Add **onResume** to MainActivity

- How can it be changed to that both back buttons preserve the resume string ?



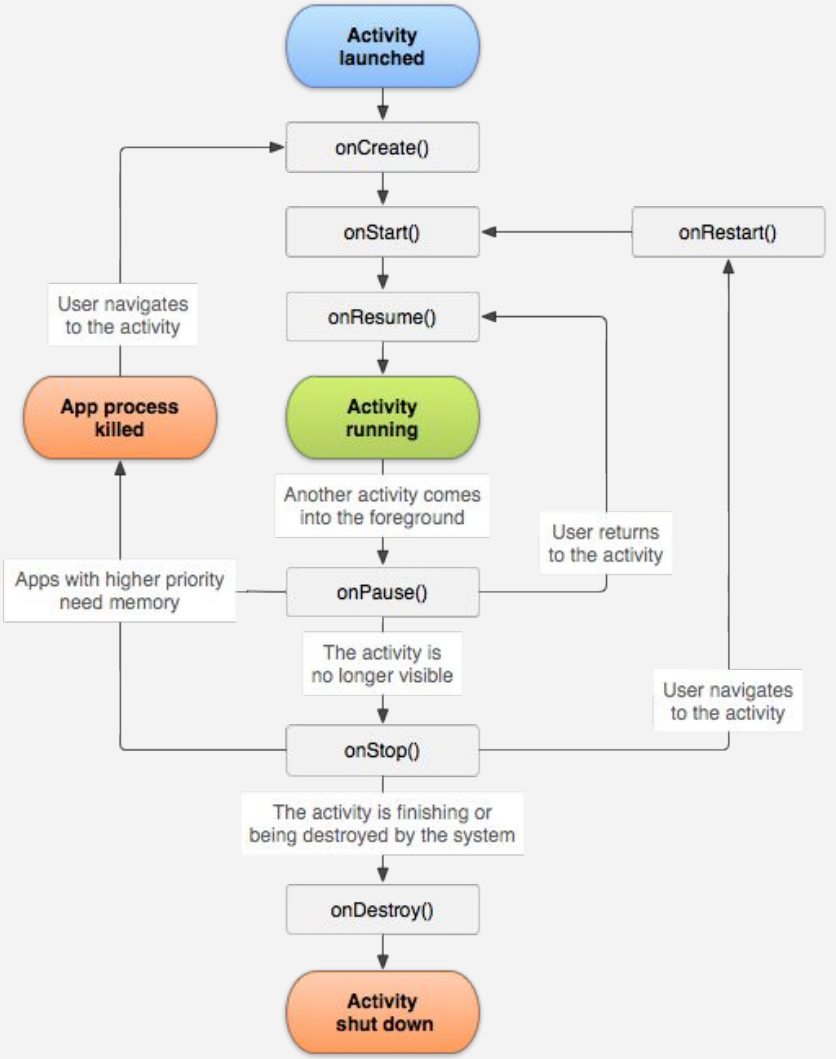
Add **onResume** to MainActivity

- How can it be changed to that both back buttons preserve the resume string ?
- Change the variables to be static!



Activity Lifecycle

- What is the difference between hitting the **system back button** and the **in-app back button**?
- What functions are called, and which are not?
- Let's experiment

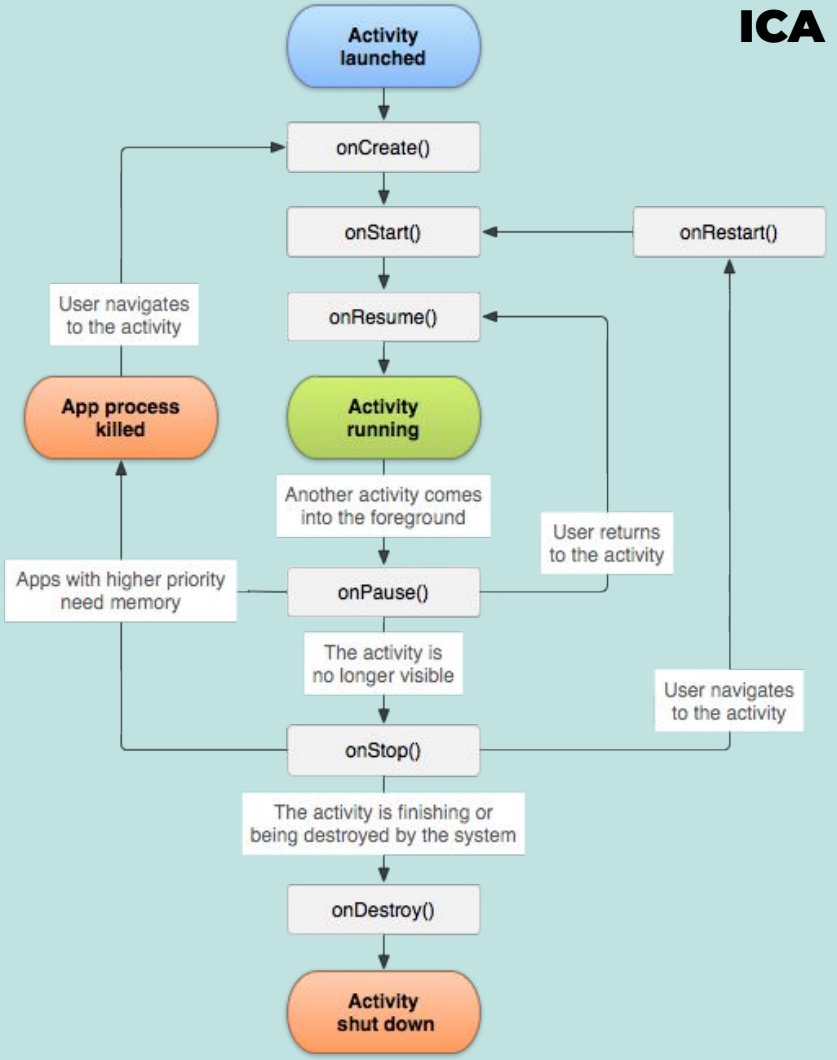


Add Event functions

- Add a function like so, but for each callback event function

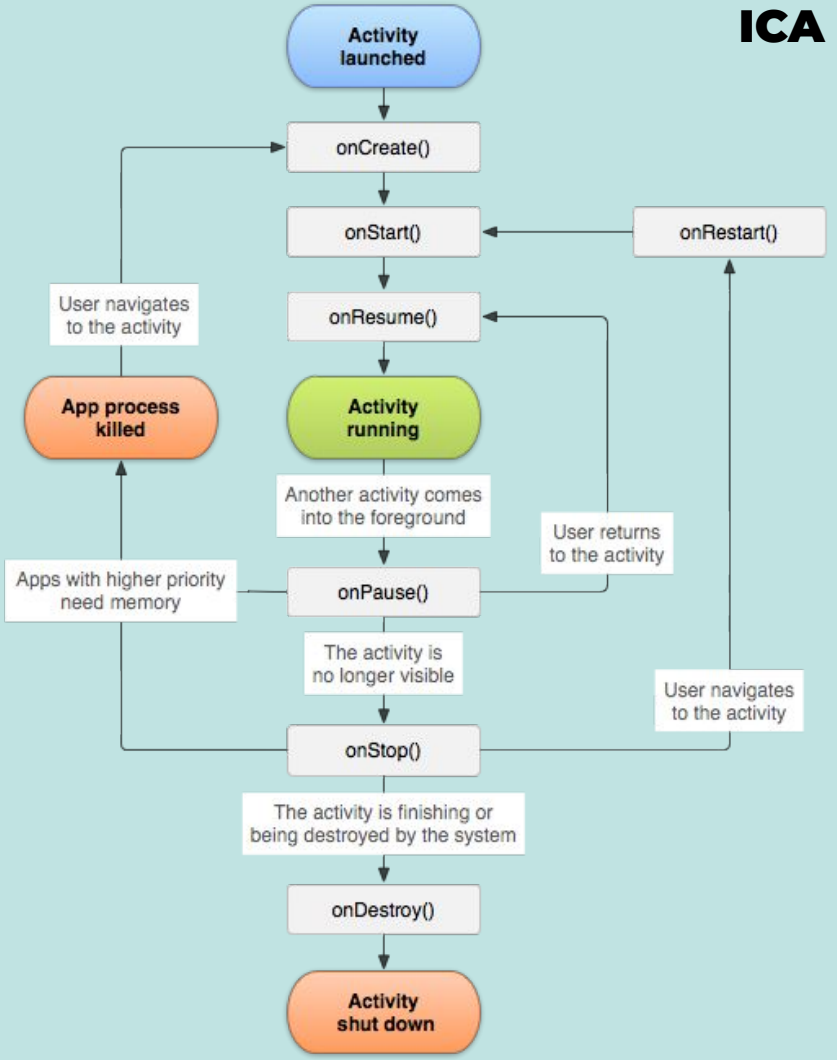
```
protected void onStart() {
    super.onStart();
    System.out.println("onStart");
}
```

- Run the app, try both back buttons, and look at the output



Add Event functions

- How to fix so that **onDestroy** is not called?
- See here
- Add `android:launchMode="singleTop"` To the activity tag in the manifest file

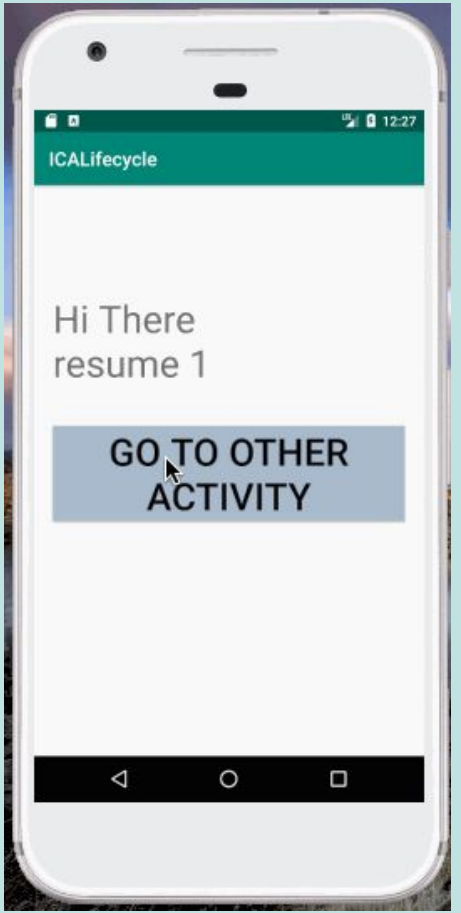


Add a timer

Update your application to have the shown behavior. You should use:

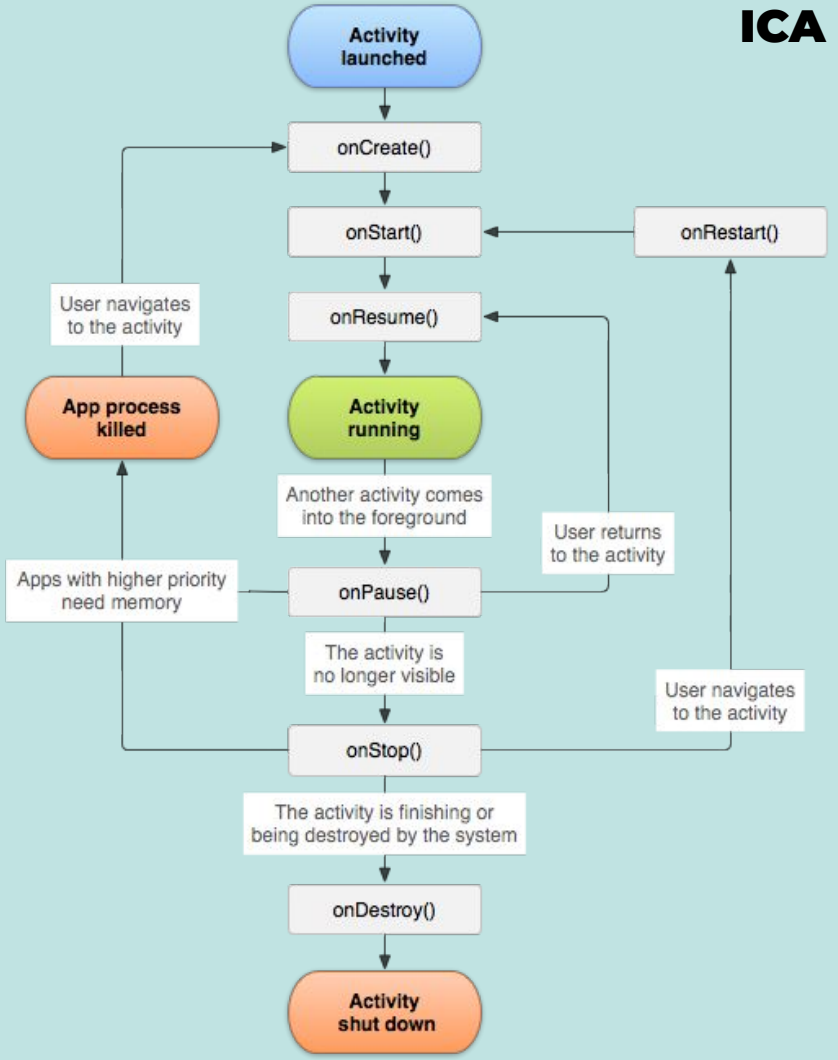
The **`Date`** class from **`java.util`** or **`System.currentTimeMillis`**

The **`onPuase`** and the **`onResume`** methods



Discuss

- Discuss in your groups and come up with 3 **practical** uses for the various lifecycle functions, such as **onPause**, **onDestroy**, and **onResume**

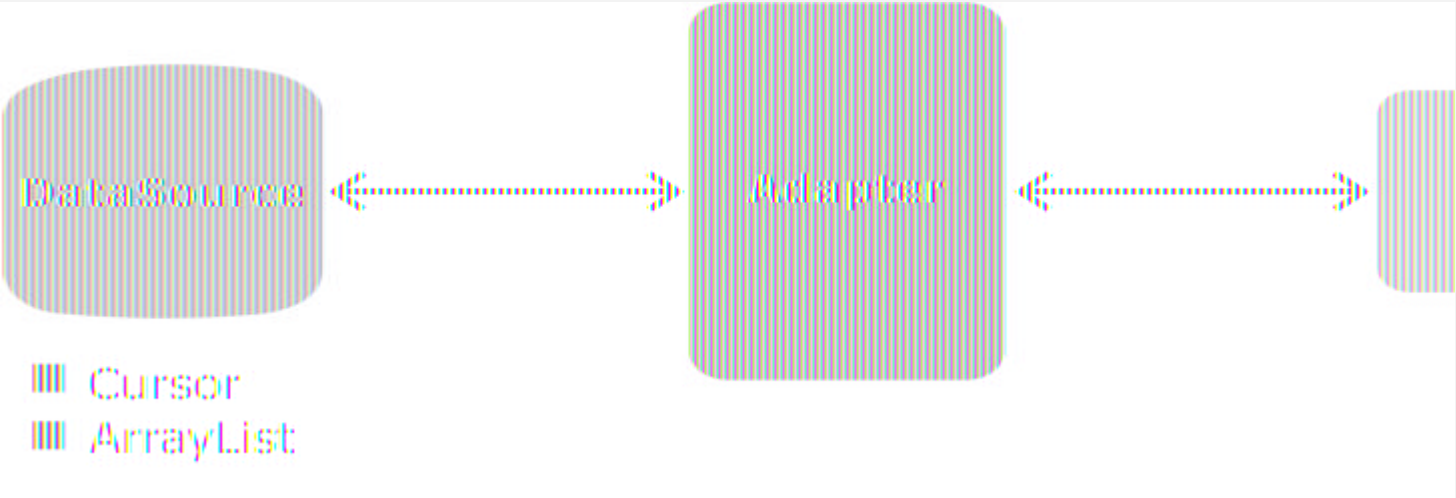


ListView

- A **ListView** is a type of view, designed to display lists!
  - Does not know details of what it is displaying
  - Generates rows on-demand
- https://developer.android.com/reference/android/widget/ListView

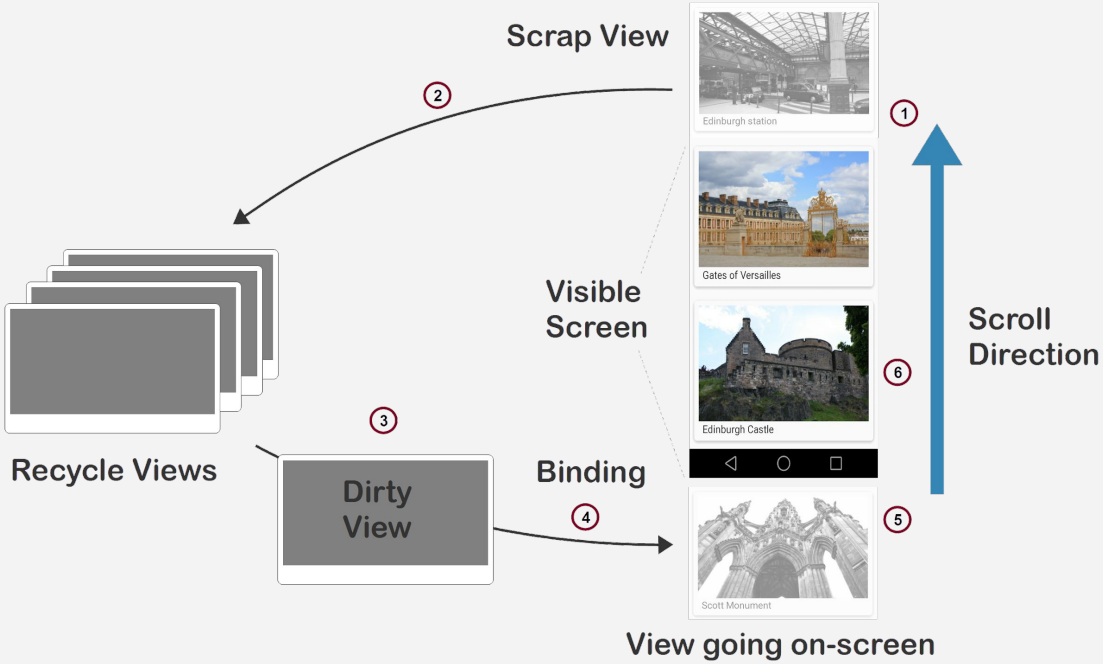
Data, Adapters, and ListViews

https://guides.codepath.com/android/Using-an-ArrayAdapter-with-ListView



Data, Adapters, and ListViews

https://guides.codepath.com/android/Using-an-ArrayAdapter-with-ListView



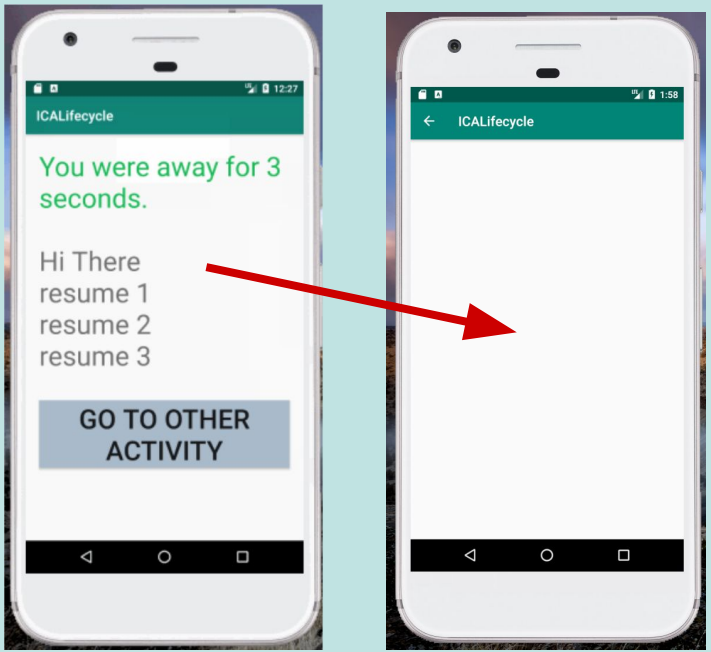
ArrayAdapter

- Adapter that maps simple data in an array to being displayed via a view

- What if there are multiple elements to be displayed per list item?

Open up the app

- Open up the ICALifecycle application
- Open up the xml file for the **OtherView**



Add a LinearLayout

- Ensure that you are using a LinearLayout
- Add a Listview, as shown
- Run it - What does it look like?

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

<LinearLayout
  ...
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context=".OtherActivity">

  <ListView
     android:id="@+id/words_view"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
  />

</LinearLayout>
```

Create a new View for Row

- Right click on the **layouts** directory, then choose **New** -> **XML** -> **XML Layout File**
- Name it **shopping_list_row**
- Use a linearLayout with a textview

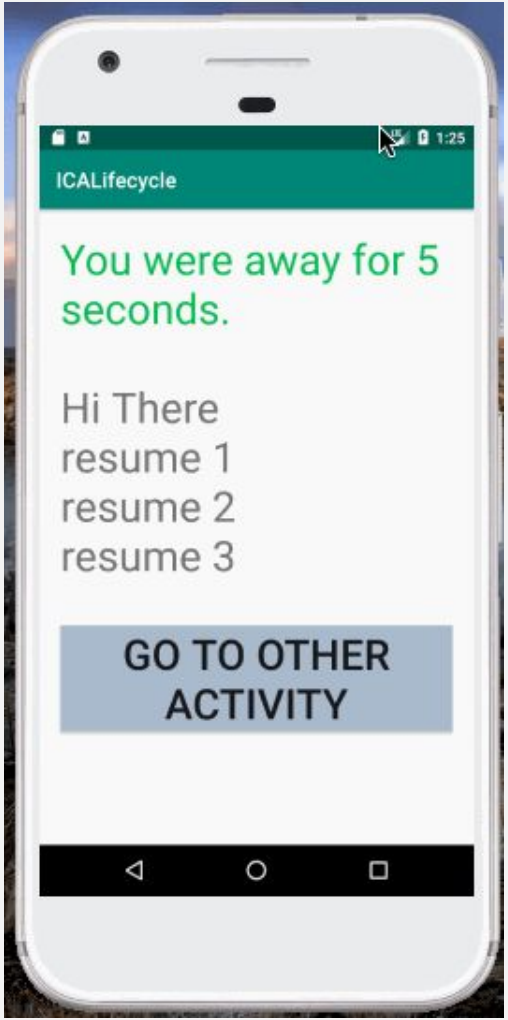
```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    . . .
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/shopping_list_row_item"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        . . .
        android:textSize="55dp" />

</LinearLayout>
```

Editing the Code

- Go to **OtherView.java**
- Add a ListView member variable
- Create an array with 10+ strings (shopping list)

Create Adapter

```
// Get a reference to the ListView
shoppingListView =
    (ListView)findViewById(R.id.shopping_list_view);

// Create a new Array Adapter
// Specify which layout and view to use for a row
// and the data (array) to use
ArrayAdapter<String> arrayAdapter = new
      ArrayAdapter<String>(this, R.layout.shopping_list_row,
                           R.id.shopping_list_row, shoppingList);

// Link the ListView and the Adapter
shoppingListView.setAdapter(arrayAdapter);
```


1:25
ICALifecycle
You were away for 5 seconds.
Hi There
resume 1
resume 2
resume 3
GO TO OTHER ACTIVITY