ANDROID APP COMPONENTS

INTENTS AND INTENT FILTERS

- AN INTENT IS A MESSAGING OBJECT YOU CAN USE TO REQUEST AN ACTION FROM ANOTHER APP COMPONENT.
- ALTHOUGH INTENTS FACILITATE COMMUNICATION BETWEEN COMPONENTS IN SEVERAL WAYS, THERE ARE THREE FUNDAMENTAL USE CASES:
  - **STARTING AN ACTIVITY**
    - AN ACTIVITY REPRESENTS A SINGLE SCREEN IN AN APP.
    - YOU CAN START A NEW INSTANCE OF AN ACTIVITY BY PASSING AN INTENT TO STARTACTIVITY(). THE INTENT DESCRIBES THE ACTIVITY TO START AND CARRIES ANY NECESSARY DATA.
  - **STARTING A SERVICE**
    - FOR VERSIONS EARLIER THAN ANDROID 5.0 (API LEVEL 21), YOU CAN START A SERVICE BY USING METHODS OF THE SERVICE CLASS.
    - YOU CAN START A SERVICE TO PERFORM A ONE-TIME OPERATION (SUCH AS DOWNLOADING A FILE) BY PASSING AN INTENT TO STARTSERVICE().

CONT …

- **DELIVERING A BROADCAST**
  - A BROADCAST IS A MESSAGE THAT ANY APP CAN RECEIVE.
  - THE SYSTEM DELIVERS VARIOUS BROADCASTS FOR SYSTEM EVENTS, SUCH AS WHEN THE SYSTEM BOOTS UP OR THE DEVICE STARTS CHARGING.
  - YOU CAN DELIVER A BROADCAST TO OTHER APPS BY PASSING AN INTENT TO SENDBROADCAST() OR SENDORDEREDBROADCAST()

INTENT TYPES

- THERE ARE TWO TYPES OF INTENTS:
  - **EXPLICIT INTENTS:**
    - SPECIFY THE COMPONENT TO START BY NAME (THE FULLY-QUALIFIED CLASS NAME).
    - YOU'LL TYPICALLY USE AN EXPLICIT INTENT TO START A COMPONENT IN YOUR OWN APP, BECAUSE YOU KNOW THE CLASS NAME OF THE ACTIVITY OR SERVICE YOU WANT TO START.
  - **IMPLICIT INTENTS**
    - DO NOT NAME A SPECIFIC COMPONENT, BUT INSTEAD DECLARE A GENERAL ACTION TO PERFORM, WHICH ALLOWS A COMPONENT FROM ANOTHER APP TO HANDLE IT.
    - FOR EXAMPLE, IF YOU WANT TO SHOW THE USER A LOCATION ON A MAP, YOU CAN USE AN IMPLICIT INTENT TO REQUEST THAT ANOTHER CAPABLE APP SHOW A SPECIFIED LOCATION ON A MAP.

BUILDING AN INTENT

- AN INTENT OBJECT CARRIES INFORMATION THAT THE ANDROID SYSTEM USES TO DETERMINE WHICH COMPONENT TO START (SUCH AS THE EXACT COMPONENT NAME OR COMPONENT CATEGORY THAT SHOULD RECEIVE THE INTENT), PLUS INFORMATION THAT THE RECIPIENT COMPONENT USES IN ORDER TO PROPERLY PERFORM THE ACTION (SUCH AS THE ACTION TO TAKE AND THE DATA TO ACT UPON).
- THE PRIMARY INFORMATION CONTAINED IN AN INTENT ARE:
  - **COMPONENT NAME:** THE NAME OF THE COMPONENT TO START.
  - **ACTION:** A STRING THAT SPECIFIES THE GENERIC ACTION TO PERFORM (SUCH AS *VIEW* OR *PICK*).
    - THE ACTION LARGELY DETERMINES HOW THE REST OF THE INTENT IS STRUCTURED—PARTICULARLY THE INFORMATION THAT IS CONTAINED IN THE DATA AND EXTRAS.
    - YOU CAN SPECIFY YOUR OWN ACTIONS FOR USE BY INTENTS WITHIN YOUR APP (OR FOR USE BY OTHER APPS TO INVOKE COMPONENTS IN YOUR APP), BUT YOU USUALLY SPECIFY ACTION CONSTANTS DEFINED BY THE INTENT CLASS OR OTHER FRAMEWORK CLASSES.
  - **DATA:** THE URI (A URI OBJECT) THAT REFERENCES THE DATA TO BE ACTED ON AND/OR THE MIME TYPE OF THAT DATA. THE TYPE OF DATA SUPPLIED IS GENERALLY DICTATED BY THE INTENT'S ACTION.

CONT …

- **CATEGORY:** A STRING CONTAINING ADDITIONAL INFORMATION ABOUT THE KIND OF COMPONENT THAT SHOULD HANDLE THE INTENT. ANY NUMBER OF CATEGORY DESCRIPTIONS CAN BE PLACED IN AN INTENT, BUT MOST INTENTS DO NOT REQUIRE A CATEGORY
- **EXTRAS:** KEY-VALUE PAIRS THAT CARRY ADDITIONAL INFORMATION REQUIRED TO ACCOMPLISH THE REQUESTED ACTION. JUST AS SOME ACTIONS USE PARTICULAR KINDS OF DATA URIS, SOME ACTIONS ALSO USE PARTICULAR EXTRAS. YOU CAN ADD EXTRA DATA WITH VARIOUS PUTEXTRA() METHODS, EACH ACCEPTING TWO PARAMETERS: THE KEY NAME AND THE VALUE.
- **FLAGS**: FLAGS ARE DEFINED IN THE INTENT CLASS THAT FUNCTION AS METADATA FOR THE INTENT. THE FLAGS MAY INSTRUCT THE ANDROID SYSTEM HOW TO LAUNCH AN ACTIVITY (FOR EXAMPLE, WHICH TASK THE ACTIVITY SHOULD BELONG TO) AND HOW TO TREAT IT AFTER IT'S LAUNCHED (FOR EXAMPLE, WHETHER IT BELONGS IN THE LIST OF RECENT ACTIVITIES).

ACTIVITIES

- ACTIVITIES ARE ONE OF THE FUNDAMENTAL BUILDING BLOCKS OF APPS ON THE ANDROID PLATFORM.
- THEY SERVE AS THE ENTRY POINT FOR A USER'S INTERACTION WITH AN APP, AND ARE ALSO CENTRAL TO HOW A USER NAVIGATES WITHIN AN APP (AS WITH THE BACK BUTTON) OR BETWEEN APPS (AS WITH THE RECENTS BUTTON).
- SKILLFULLY MANAGING ACTIVITIES ALLOWS YOU TO ENSURE THAT, FOR EXAMPLE:
  - ORIENTATION CHANGES TAKE PLACE SMOOTHLY WITHOUT DISRUPTING THE USER EXPERIENCE.
  - USER DATA IS NOT LOST DURING ACTIVITY TRANSITIONS.
  - THE SYSTEM KILLS PROCESSES WHEN IT'S APPROPRIATE TO DO SO.
- UNLIKE PROGRAMMING PARADIGMS IN WHICH APPS ARE LAUNCHED WITH A MAIN() METHOD, THE ANDROID SYSTEM INITIATES CODE IN AN ACTIVITY INSTANCE BY INVOKING SPECIFIC CALLBACK METHODS THAT CORRESPOND TO SPECIFIC STAGES OF ITS LIFECYCLE.

CONT …

- GENERALLY, ONE ACTIVITY IMPLEMENTS ONE SCREEN IN AN APP.
- MOST APPS CONTAIN MULTIPLE SCREENS, WHICH MEANS THEY COMPRISE MULTIPLE ACTIVITIES. TYPICALLY, ONE ACTIVITY IN AN APP IS SPECIFIED AS THE MAIN ACTIVITY, WHICH IS THE FIRST SCREEN TO APPEAR WHEN THE USER LAUNCHES THE APP.
- EACH ACTIVITY CAN THEN START ANOTHER ACTIVITY IN ORDER TO PERFORM DIFFERENT ACTIONS.
- ALTHOUGH ACTIVITIES WORK TOGETHER TO FORM A COHESIVE USER EXPERIENCE IN AN APP, EACH ACTIVITY IS ONLY LOOSELY BOUND TO THE OTHER ACTIVITIES; THERE ARE USUALLY MINIMAL DEPENDENCIES AMONG THE ACTIVITIES IN AN APP. IN FACT, ACTIVITIES OFTEN START UP ACTIVITIES BELONGING TO OTHER APPS.
- TO USE ACTIVITIES IN YOUR APP, YOU MUST REGISTER INFORMATION ABOUT THEM IN THE APP’S MANIFEST, AND YOU MUST MANAGE ACTIVITY LIFECYCLES APPROPRIATELY.

CONFIGURING THE MANIFEST (ACTIVITY)

- FOR YOUR APP TO BE ABLE TO USE ACTIVITIES, YOU MUST DECLARE THE ACTIVITIES, AND CERTAIN OF THEIR ATTRIBUTES, IN THE MANIFEST
- DECLARE ACTIVITIES
  - TO DECLARE YOUR ACTIVITY, OPEN YOUR MANIFEST FILE AND ADD AN <ACTIVITY> ELEMENT AS A CHILD OF THE <APPLICATION> ELEMENT.

```
<MANIFEST ... >
 <APPLICATION ... >
   <ACTIVITY ANDROID:NAME=".EXAMPLEACTIVITY" />
   ...
 </APPLICATION ... >
 ...
</MANIFEST >
```

  - THE ONLY REQUIRED ATTRIBUTE FOR THIS ELEMENT IS ANDROID:NAME, WHICH SPECIFIES THE CLASS NAME OF THE ACTIVITY.
  - YOU CAN ALSO ADD ATTRIBUTES THAT DEFINE ACTIVITY CHARACTERISTICS SUCH AS LABEL, ICON, OR UI THEME.

CONT …

- DECLARE INTENT FILTERS
  - INTENT FILTERS ARE A VERY POWERFUL FEATURE OF THE ANDROID PLATFORM.
  - THEY PROVIDE THE ABILITY TO LAUNCH AN ACTIVITY BASED NOT ONLY ON AN *EXPLICIT* REQUEST, BUT ALSO AN *IMPLICIT* ONE.
  - FOR EXAMPLE, AN EXPLICIT REQUEST MIGHT TELL THE SYSTEM TO “START THE SEND EMAIL ACTIVITY IN THE GMAIL APP".
  - BY CONTRAST, AN IMPLICIT REQUEST TELLS THE SYSTEM TO “START A SEND EMAIL SCREEN IN ANY ACTIVITY THAT CAN DO THE JOB."
  - WHEN THE SYSTEM UI ASKS A USER WHICH APP TO USE IN PERFORMING A TASK, THAT’S AN INTENT FILTER AT WORK.

CONT …

- YOU CAN TAKE ADVANTAGE OF THIS FEATURE BY DECLARING AN <INTENT-FILTER> ATTRIBUTE IN THE <ACTIVITY> ELEMENT.
- THE DEFINITION OF THIS ELEMENT INCLUDES AN<ACTION> ELEMENT AND, OPTIONALLY, A <CATEGORY> ELEMENT AND/OR A <DATA> ELEMENT.
- THESE ELEMENTS COMBINE TO SPECIFY THE TYPE OF INTENT TO WHICH YOUR ACTIVITY CAN RESPOND.
- FOR EXAMPLE, THE FOLLOWING CODE SNIPPET SHOWS HOW TO CONFIGURE AN ACTIVITY THAT SENDS TEXT DATA, AND RECEIVES REQUESTS FROM OTHER ACTIVITIES TO DO SO:

```
<ACTIVITY ANDROID:NAME=".EXAMPLEACTIVITY" ANDROID:ICON="@DRAWABLE/APP_ICON">
  <INTENT-FILTER>
    <ACTION ANDROID:NAME="ANDROID.INTENT.ACTION.SEND" />
    <CATEGORY ANDROID:NAME="ANDROID.INTENT.CATEGORY.DEFAULT" />
    <DATA ANDROID:MIMETYPE="TEXT/PLAIN" />
  </INTENT-FILTER>
<ACTIVITY>
```

CONT …

- DECLARE PERMISSIONS
  - YOU CAN USE THE MANIFEST'S <ACTIVITY> TAG TO CONTROL WHICH APPS CAN START A PARTICULAR ACTIVITY.
  - A PARENT ACTIVITY CANNOT LAUNCH A CHILD ACTIVITY UNLESS BOTH ACTIVITIES HAVE THE THE SAME PERMISSIONS IN THEIR MANIFEST.
  - IF YOU DECLARE A <USES-PERMISSION> ELEMENT FOR A PARTICULAR ACTIVITY, THE CALLING ACTIVITY MUST HAVE A MATCHING <USES-PERMISSION> ELEMENT.

```
<ACTIVITY ANDROID:NAME="...."
  ANDROID:PERMISSION=”COM.GOOGLE.SOCIALAPP.PERMISSION.SHARE_POST”
/>
```

MANAGING THE ACTIVITY LIFECYCLE

- OVER THE COURSE OF ITS LIFETIME, AN ACTIVITY GOES THROUGH A NUMBER OF STATES. YOU USE A SERIES OF CALLBACKS TO HANDLE TRANSITIONS BETWEEN STATES.
  - ONCREATE()
    - YOU **MUST** IMPLEMENT THIS CALLBACK, WHICH FIRES WHEN THE SYSTEM CREATES YOUR ACTIVITY.
    - YOUR IMPLEMENTATION SHOULD INITIALIZE THE ESSENTIAL COMPONENTS OF YOUR ACTIVITY: FOR EXAMPLE, YOUR APP SHOULD CREATE VIEWS AND BIND DATA TO LISTS HERE.
    - MOST IMPORTANTLY, THIS IS WHERE YOU MUST CALL **SETCONTENTVIEW()** TO DEFINE THE LAYOUT FOR THE ACTIVITY'S USER INTERFACE.
    - WHEN ONCREATE() FINISHES, THE NEXT CALLBACK IS ALWAYS ONSTART()
  - ONSTART()
    - AS ONCREATE() EXITS, THE ACTIVITY ENTERS THE STARTED STATE, AND THE ACTIVITY BECOMES VISIBLE TO THE USER. THIS CALLBACK CONTAINS WHAT AMOUNTS TO THE ACTIVITY’S FINAL PREPARATIONS FOR COMING TO THE FOREGROUND AND BECOMING INTERACTIV

CONT …

- ONRESUME()
  - THE SYSTEM INVOKES THIS CALLBACK JUST BEFORE THE ACTIVITY STARTS INTERACTING WITH THE USER.
  - AT THIS POINT, THE ACTIVITY IS AT THE TOP OF THE ACTIVITY STACK, AND CAPTURES ALL USER INPUT.
  - MOST OF AN APP'S CORE FUNCTIONALITY IS IMPLEMENTED IN THE ONRESUME() METHOD.
  - THE ONPAUSE() CALLBACK ALWAYS FOLLOWS ONRESUME()
- ONPAUSE()
  - THE SYSTEM CALLS ONPAUSE() WHEN THE ACTIVITY LOSES FOCUS AND ENTERS A PAUSED STATE.
  - THIS STATE OCCURS WHEN, FOR EXAMPLE, THE USER TAPS THE BACK OR OVERLAY BUTTON.
  - WHEN THE SYSTEM CALLS ONPAUSE() FOR YOUR ACTIVITY, IT TECHNICALLY MEANS YOUR ACTIVITY IS STILL PARTIALLY VISIBLE, BUT MOST OFTEN IS AN INDICATION THAT THE USER IS LEAVING THE ACTIVITY, AND THE ACTIVITY WILL SOON ENTER THE STOPPED OR RESUMED STATE.

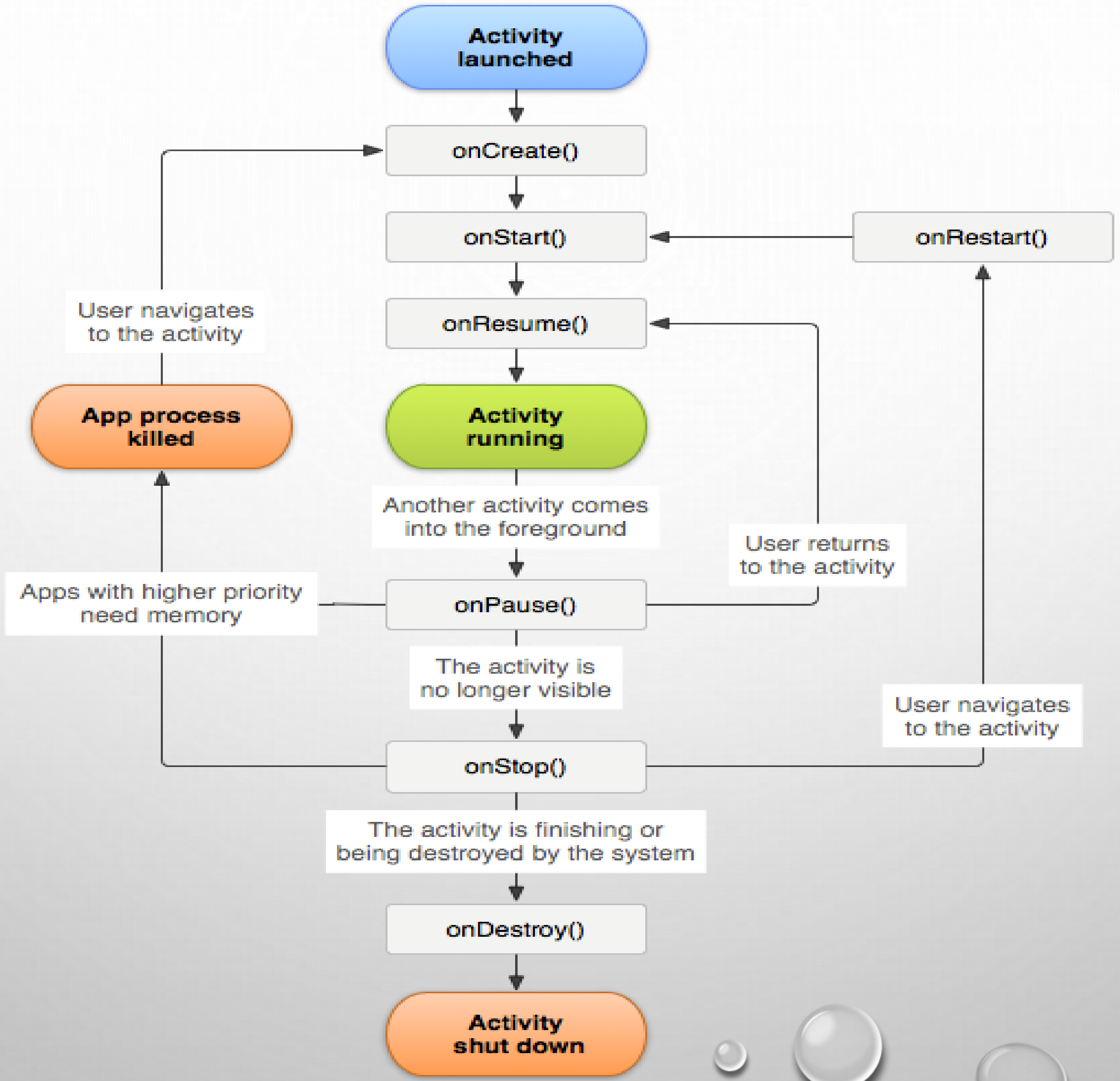
CONT …

  - YOU SHOULD NOT USE ONPAUSE() TO SAVE APPLICATION OR USER DATA, MAKE NETWORK CALLS, OR EXECUTE DATABASE TRANSACTIONS.
  - ONCE ONPAUSE() FINISHES EXECUTING, THE NEXT CALLBACK IS EITHER ONSTOP() OR ONRESUME(), DEPENDING ON WHAT HAPPENS AFTER THE ACTIVITY ENTERS THE PAUSED STATE.

- ONSTOP()
  - THE SYSTEM CALLS ONSTOP() WHEN THE ACTIVITY IS NO LONGER VISIBLE TO THE USER.
  - THIS MAY HAPPEN BECAUSE THE ACTIVITY IS BEING DESTROYED, A NEW ACTIVITY IS STARTING, OR AN EXISTING ACTIVITY IS ENTERING A RESUMED STATE AND IS COVERING THE STOPPED ACTIVITY.
  - IN ALL OF THESE CASES, THE STOPPED ACTIVITY IS NO LONGER VISIBLE AT ALL.
  - THE NEXT CALLBACK THAT THE SYSTEM CALLS IS EITHER ONRESTART(), IF THE ACTIVITY IS COMING BACK TO INTERACT WITH THE USER, OR BY ONDESTROY() IF THIS ACTIVITY IS COMPLETELY TERMINATING.

CONT …

- ONRESTART()
  - THE SYSTEM INVOKES THIS CALLBACK WHEN AN ACTIVITY IN THE STOPPED STATE IS ABOUT TO RESTART.
  - ONRESTART() RESTORES THE STATE OF THE ACTIVITY FROM THE TIME THAT IT WAS STOPPED.
  - THIS CALLBACK IS ALWAYS FOLLOWED BY ONSTART()
- ONDESTROY()
  - THE SYSTEM INVOKES THIS CALLBACK BEFORE AN ACTIVITY IS DESTROYED.
  - THIS CALLBACK IS THE FINAL ONE THAT THE ACTIVITY RECEIVES. ONDESTROY() IS USUALLY IMPLEMENTED TO ENSURE THAT ALL OF AN ACTIVITY’S RESOURCES ARE RELEASED WHEN THE ACTIVITY, OR THE PROCESS CONTAINING IT, IS DESTROYED.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down