WIA2007
MOBILE APPLICATION DEVELOPMENT
INTRODUCTION TO ANDROID:
THE MOBILE OS, COMPONENTS AND FLOW

Originally Created by S. Ong
Modified by Chiew, TK

Semester 1, 2023/2024

simying.ong@um.edu.my

REMINDER

Attendance matters!

MOBILE OS
SOME REVISIONS ON PREVIOUS TOPIC

- **Software runs on top of mobile device** to provide functionalities to the mobile users.
- Types of Mobile OS?

Android (Google)

Bada (Samsung)

BlackBerry OS (Research in Motion)

iOS (Apple)

HarmonyOS (Huawei)

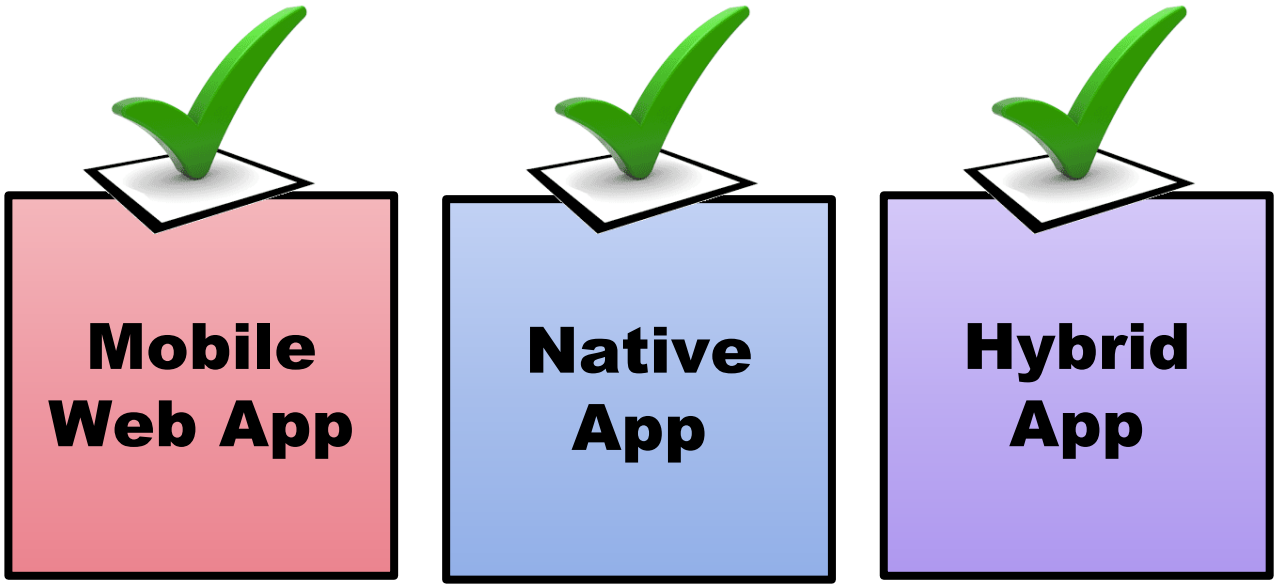
Windows OS (Windows Mobile)

Meego OS (Nokia + Intel)

And more… What else?

TYPES OF MOBILE APP DEVELOPMENT

- **3** types of Mobile App Development:



ANDROID APP FUNDAMENTAL

ANDROID APP
INTRODUCTION

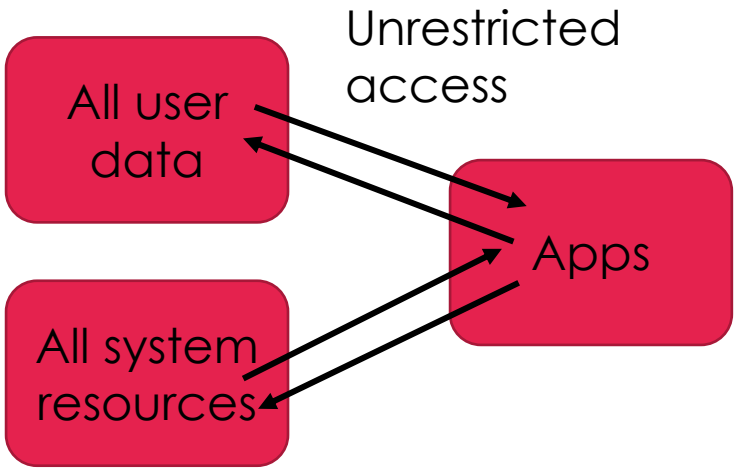
- Can be written using Java, Kotlin or C++.
- **Android SDK compiles all your mobile app codes into single Android Packages, .apk**, for download and installation purposes.
- Google Play started requiring new apps to be published with the Android App Bundle (AAB) from August 2021, replacing the Android Application Package (APK) as the standard publishing format.

ANDROID APP
INTRODUCTION

- Each installed android app is contained inside a security ***sandbox***, which provide following functionalities:
  - Android OS is in **multi-user environment** – each app is a single user.
  - System **assigns app with unique user ID** for control access purposes.
  - **Each app runs in its own Linux process**.
  - **Each process runs independently inside its own VM**.

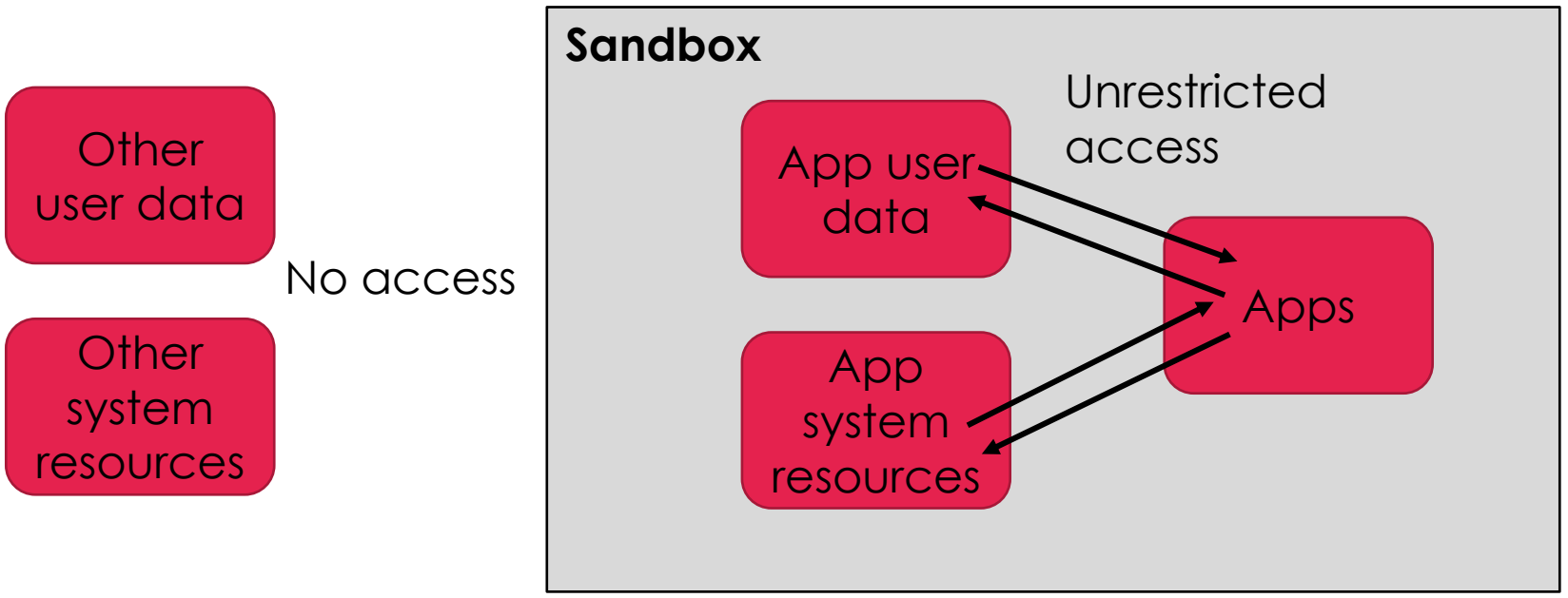
** Sandbox : security mechanism to run various running program separately*

ANDROID APP
WITHOUT SANDBOX



** Sandbox : security mechanism to run various running program separately*

ANDROID APP
WITH SANDBOX MECHANISM



** Sandbox : security mechanism to run various running program separately*

ANDROID APP COMPONENTS

- **Essential building blocks** of Android App
- **Entry point** where user / system can access the App
- **Four** different components:
  a. Activities
  b. Services
  c. Broadcast Receivers
  d. Content Providers

Some components depend on others.

- Each component serves different purposes and have different lifecycle (i.e., how it is created and killed)

ANDROID APP COMPONENTS
ACTIVITIES

- **Entry point to interact with user**
- One Activity = **Single screen with user interface**
- E.g., Email App:
  - Activity to show list of emails
  - Activity to compose email
  - Activity to read email

Works together to provide cohesive user experience, yet it's independent from each other.

ANDROID APP COMPONENTS
ACTIVITIES

- **Facilitates key interaction between system and app, e.g.**:
  - ✓**Keep track of what currently user cares about** (i.e., on screen) to ensure the process keeps running.
  - ✓Help app on **handling killed process** so that previous activity can be restored to previous state if being access again.
  - ✓**Prioritize which processes** that users might return.
  - ✓**Provide apps to implement user flows and for system to coordinate them**.

ANDROID APP COMPONENTS
ACTIVITIES

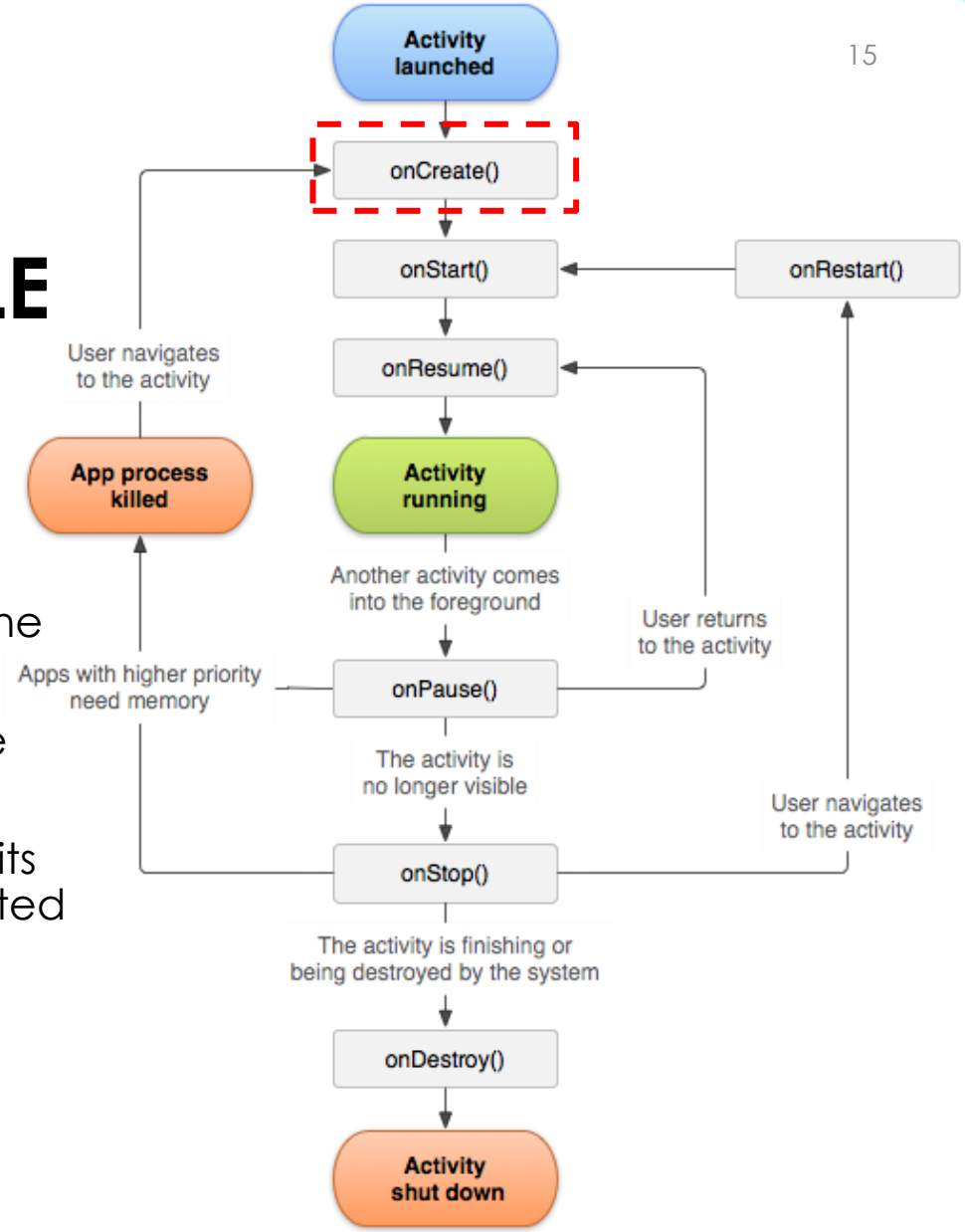
- Activity class provides a number of callbacks that allow the activity to know that **a state has changed**.
  - That the system is creating / stopping / resuming an activity / destroying process in which activity resides.
- In callback methods, we can declare how our activity behaves when the user leaves and re-enters the activity.
  - E.g., Video streaming app (pausing + disconnect network)

ANDROID APP COMPONENTS
ACTIVITIES

- Doing the right work at the right time and handling transitions properly make your app more robust and perform better:
  - Avoid crashing if user switch off the screen / switch to other apps / change orientation.
  - Save valuable system resources when it's inactive
  - Save states / users' progress when they leave app and come back later
  - And others.

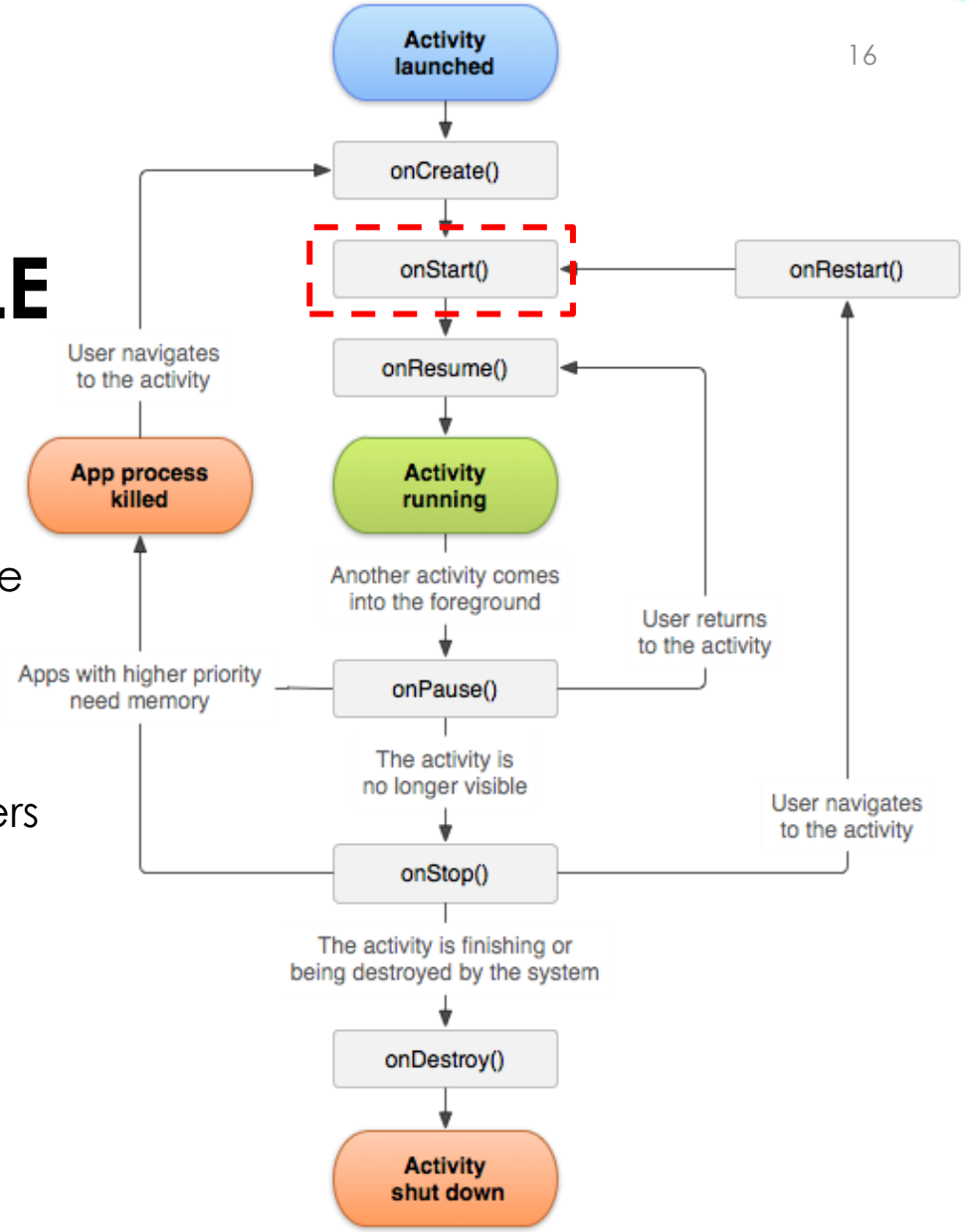
ACTIVITY LIFECYCLE
ONCREATE()

- Fires when system **first creates** the activity
- Perform **basic startup logic** that **should happened only once** for the entire life of the activity
- E.g., bind data to a list, instantiate some class-scope variables
- After onCreate() method finishes its execution, the activity enters Started state, system calls
  - OnStart()
  - OnResume() in quick succession



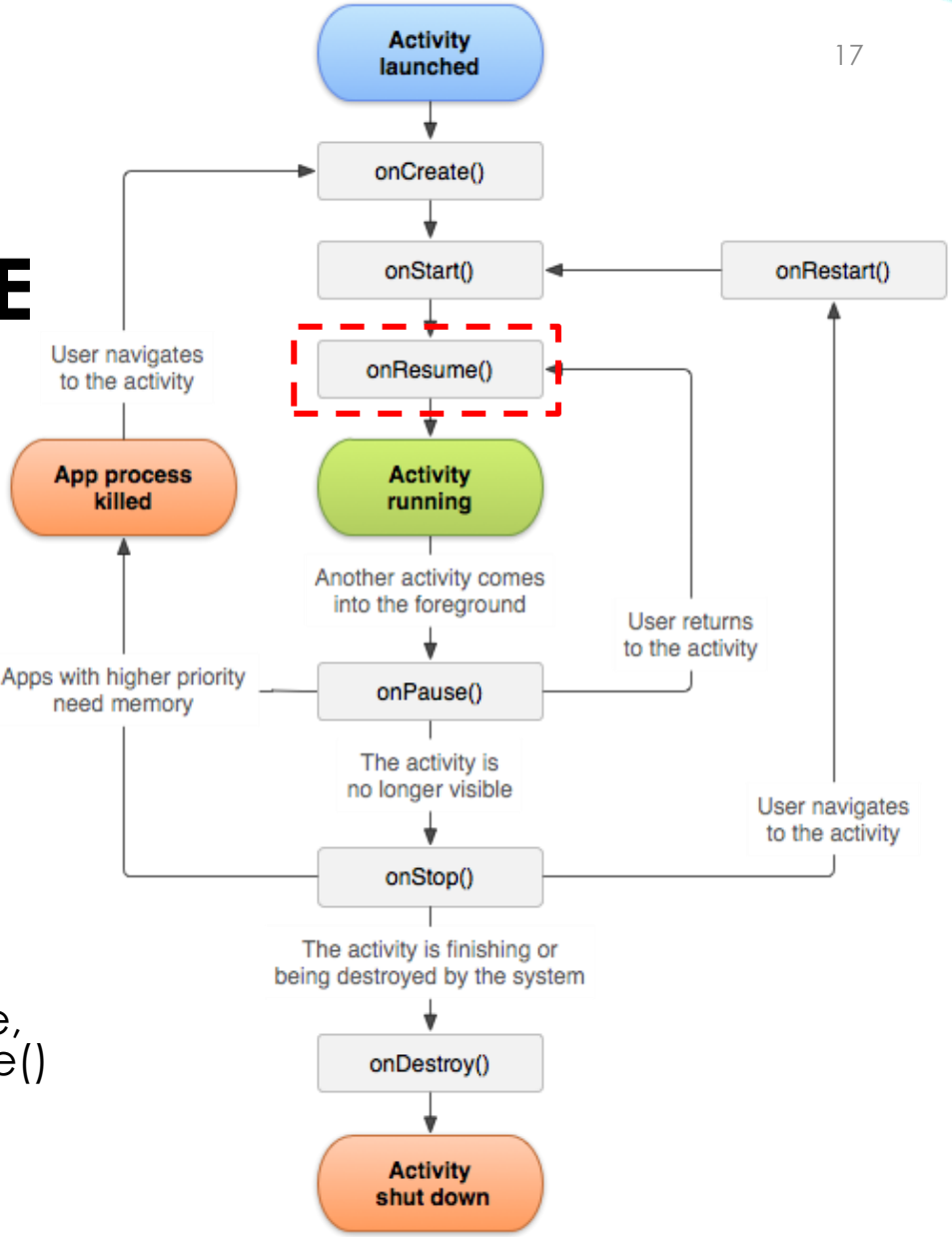
ACTIVITY LIFECYCLE
ONSTART()

- **Makes the activity visible to user**
- App **prepares for the activity to enter the foreground** and become interactive
- E.g., initialize code that maintains the UI
- Once the callback finishes, it enters Resumed state
  - System invokes onResume()



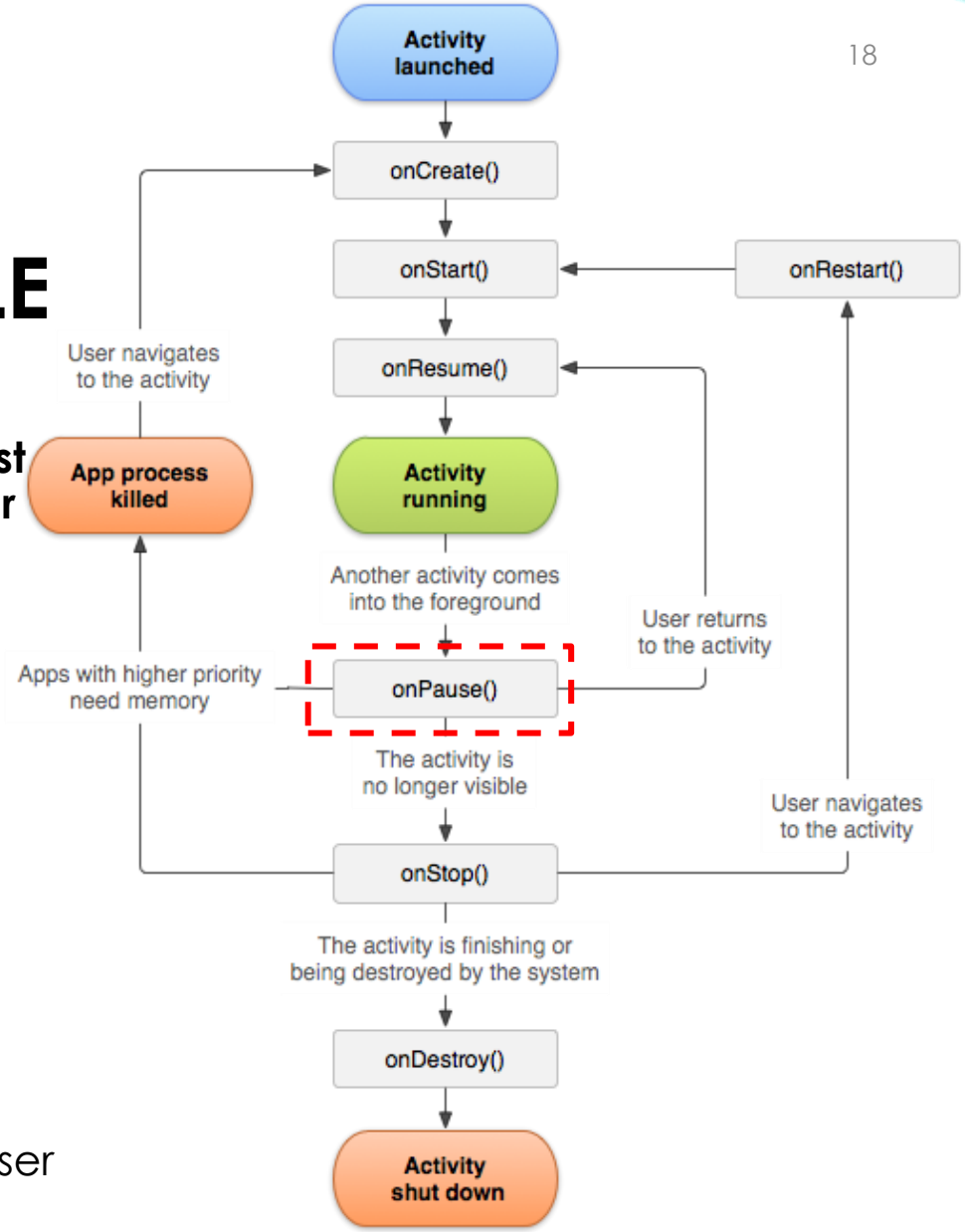
ACTIVITY LIFECYCLE
ONRESUME()

- State where the **app <u>interacts</u> with the user**
- Stays in this state until something happens to take focus away from the app
- E.g., receiving phone call, user navigates away to another app, device's screen turns off.
- When interruptive event occurs, activity enters Paused state.
  - System invokes onPause()
- If activity returns from Paused state, system once again calls onResume()
  - Should implement onResume() to initialize components that release during onPause()



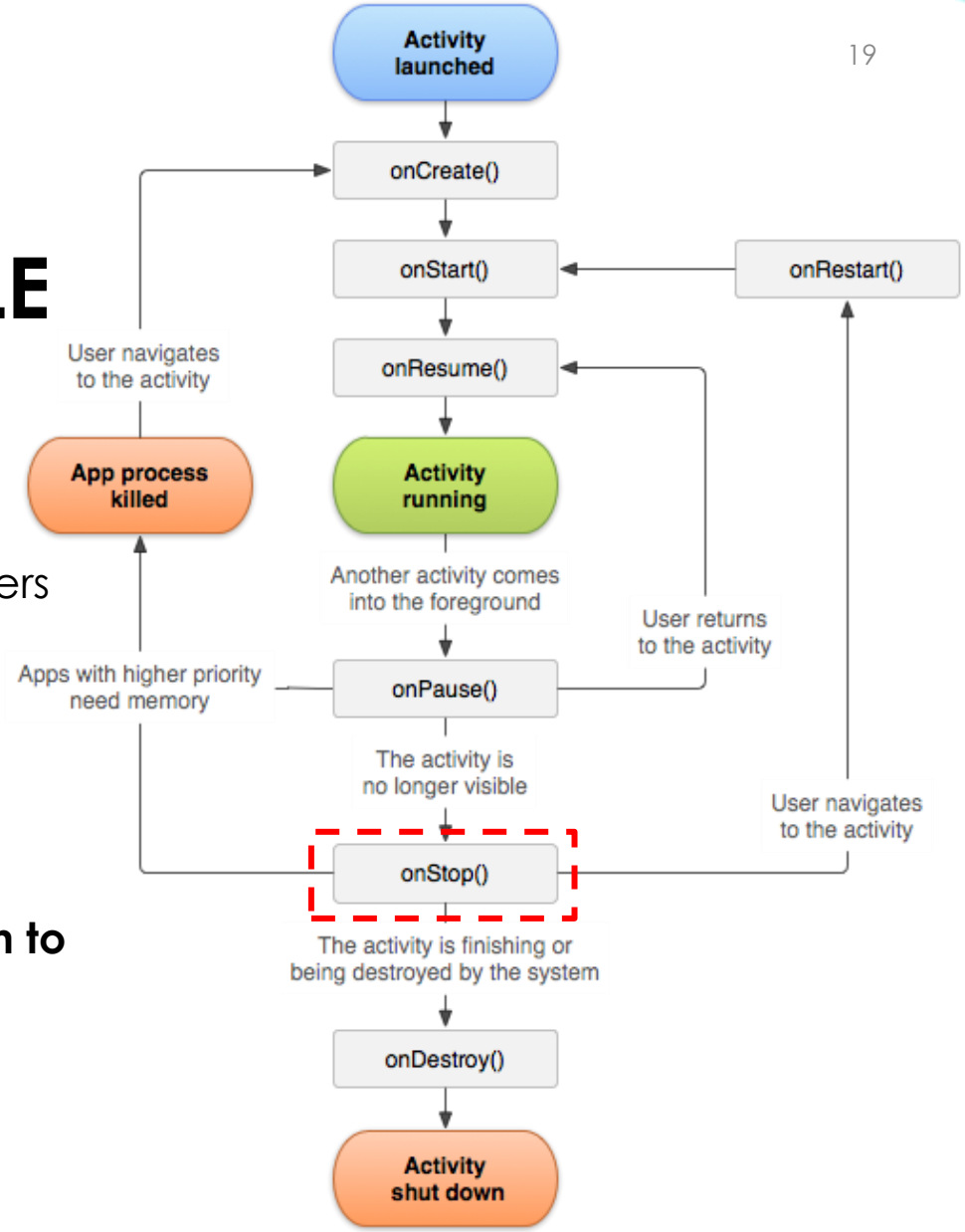
ACTIVITY LIFECYCLE
ONPAUSE()

- System calls this method as the **first indication that user is <u>leaving</u> your activity.**
- Activity **no longer in foreground**
- Use onPause() to pause or adjust operations that should not be continued.
- onPause() execution is **very brief** and does not necessarily afford enough time to perform save operation – may not complete before method finishes.
- Remain until activity resume or become completely invisible to user
  - System calls onStop()



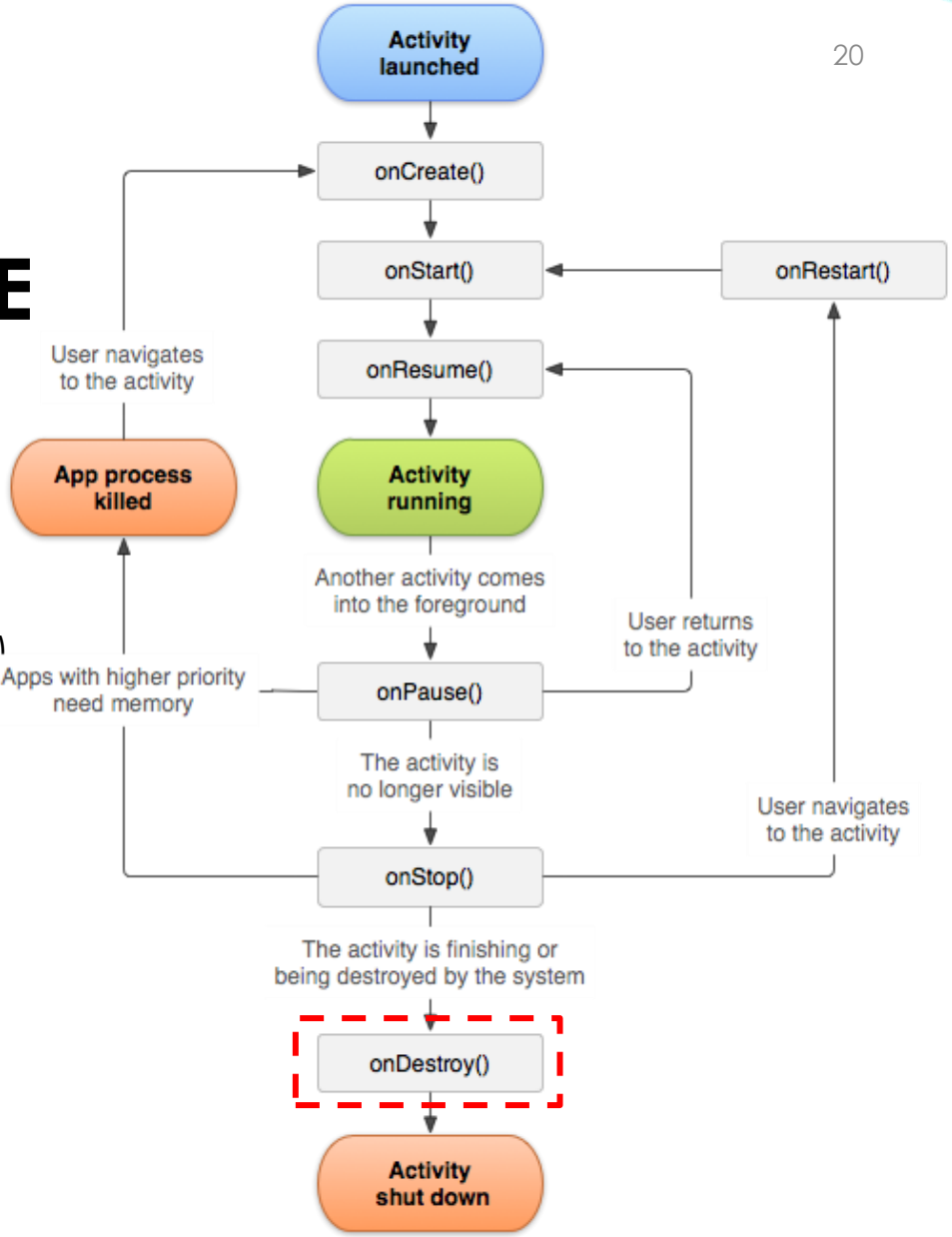
ACTIVITY LIFECYCLE
ONSTOP()

- **No longer visible to user**
  - Entered Stopped state
- E.g., newly launched activity covers entire screen / activity finishes, system calls onStop()
- **Release resources**
- **Perform CPU-intensive shutdown operation**
- Can also used to **save information to database**
- Activity comes back interact with user – calls onRestart(), OR completely go away – calls onDestroy()



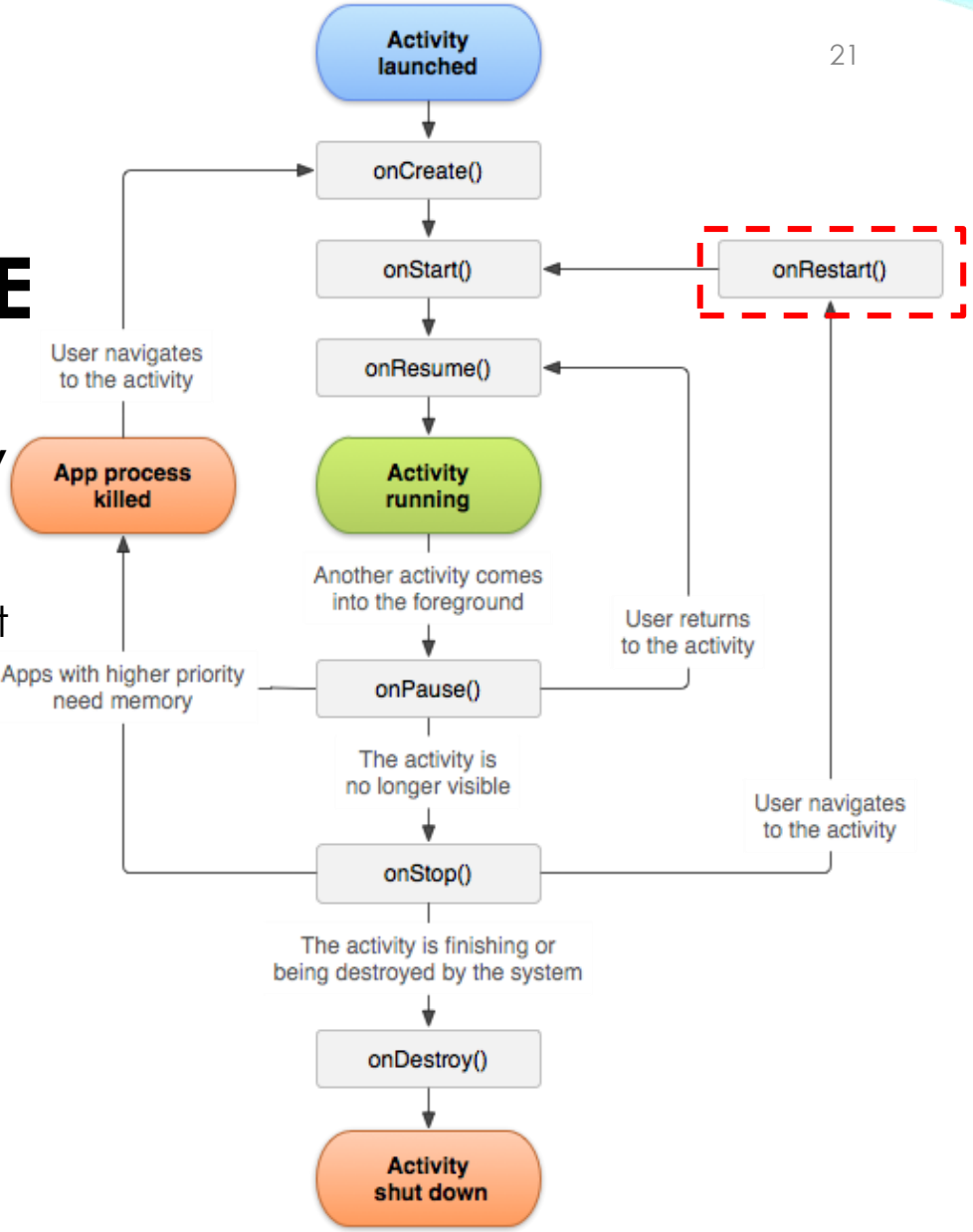
ACTIVITY LIFECYCLE
ONDESTROY()

- **Called before activity is destroyed**
  - Activity finishing
  - System temporarily destroying the activity due to configuration changes (e.g., multi-window view)
- **Clean up procedures** before it is destroyed



ACTIVITY LIFECYCLE
ONRESTART()

- Invoked when an **activity currently reside Stopped state is about the restart.**
- **Restore the state of activity** when it was stopped.
- Usually **followed by onStart()**



ANDROID APP COMPONENTS
ACTIVITIES

Quick Question 1:

- Which of the following is **FALSE** about *android app in sandbox environment*:
  A. Each android app is a user with unique identifier.
  B. Each android app runs within its own virtual machine.
  C. Each android app can interrupt other user's process when error occurs.
  D. Each android app can access to other user's data with special permission.

Answer your question in the designated conversation in Microsoft Team Post tab under General Channel.

ANDROID APP COMPONENTS
ACTIVITIES

- Thinking Question – Do we need to implement **ALL** lifecycle methods?

ANDROID APP COMPONENTS
SERVICES

- **General purpose entry point**
- Keep the app **running in background**
  - Perform long-running operation
  - Perform work for remote process
- **No user interface**
- E.g., while user is interacting with other activity…
  - Play music – example of user-aware services
  - Fetch data over network -- example of less user-aware services (can be killed if too many RAM is used)

ANDROID APP COMPONENTS
SERVICES

- Two distinct types of service:
  - **User is directly aware of**: app tells the system by saying it wants to be in foreground with a notification to tell user about it – system knows that it should try very hard to keep it running
  - **User is NOT directly aware of:** regular background service that system has more freedom in managing it (e.g., killing it, restarting it)
- **Bound service**: service A is needed to run service B
- Useful building block for all high-level system concepts, e.g., Live Wallpaper, accessibility services, notification listeners, etc.

ANDROID APP COMPONENTS
BROADCAST RECEIVERS

- **Enable the system to deliver the events to the app OUTSIDE of a regular user flow.**
- Allowing the app to **response to system-wide broadcast announcement**
- Another well-defined entry to the app
  - Can deliver broadcast to the app that are currently not running.
- E.g., App scheduled alarm to notify user for upcoming event…
  - No need keep on running after delivering the alarm to broadcast receivers

ANDROID APP COMPONENTS
BROADCAST RECEIVERS

- E.g., announce battery low, data has been downloaded, etc.
- Does not display full user interface, but **may create status bar notification**.
- Normally acts just as gateway to other components
- Intended to do very minimal amount of work

ANDROID APP COMPONENTS
CONTENT PROVIDERS

- **Manages a shared set of app data** that you can stored in a file system / SQLite database / on web / any persistent storage location.
- Other app can query or modify the data if content provider allows it (security feature).
- Provide **entry point into the app for publishing named data items**, identified by URI (uniform resource identifier) scheme.

ACTIVATING COMPONENT VIA *INTENT*

- In Android system design, **any app can start another app's component.**
  - E.g., taking photo using other app
- But because system runs each app in a separate process with file permission that restrict access to other apps, your app **CANNOT directly activate a component from another app**.
- You need to **use Android System**. HOW?
  - Deliver a message to the system that specifies your ***INTENT*** to start a particular component, system will then activate the component for you.

ACTIVATING COMPONENT VIA *INTENT*

- An **asynchronous message** **utilized to activate activity, service and broadcast receiver**.
- **Bind individual component to each other during runtime**.
- Act as **messenger** that request an action from other components, whether the component is belong to your app or not.
- Can choose to activate either specific component (*explicit* intent) or specific type of component (*implicit* intent)

ACTIVATING COMPONENT VIA *INTENT*

- For **activity and service**:
  - Define action to perform (e.g., view, send, etc.)
  - Specify the URI of data to act on
  - E.g., issue intent to let user pick a personal contact and return the URI of chosen contact.
- For **broadcast receiver**:
  - Define the announcement being broadcast

ACTIVATING COMPONENT VIA *CONTENT RESOLVER*

- Handle all **direct transactions with content provider**
- Provide a layer of abstraction between content provider and component requesting information (for security purposes).

METHODS TO ACTIVATE COMPONENTS

- Start an **activity**
  - Pass an Intent object to startActivity() or startActivityForResult()
- Start a **service**
  - Pass an Intent object to startService()
- Initiate **broadcast**
  - Pass an Intent object to sendBroadcast() / sendOrderedBroadcast() / sentStickyBroadcast()
- Query to **content provider**
  - Call query() on ContentResolver object.

METHODS TO ACTIVATE COMPONENTS

- Quick Question 2: Which of the following is NOT a component in Android App?
  A. Broadcast provider
  B. Activity
  C. Services
  D. Content provider

  Answer the question in Microsoft Team.

THE MANIFEST FILE

- By reading the app's manifest file (*where is it?*):
  - System has to know the **existence of components** in your app
  - Place to **declare all the components**
  - Identify any **user permissions** needed (e.g., read access to user's contact)
  - Declare **minimum API required** (i.e., 1 to 33)
  - Declare **hardware and software feature** used or needed (e.g., camera, Bluetooth, etc.)
  - Declare **API libraries** that need to be linked (other than Android framework's API) (e.g., Google Maps library)

THE MANIFEST FILE

DECLARING COMPONENTS

- Primary task of manifest file, e.g.,:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

- MUST declare all app components using following elements:
  - <activity>, <service>, <receiver>, <provider>
- **If no declaration – not visible to the system and can never run.**

THE MANIFEST FILE

DECLARING COMPONENTS

- Other elements:
  - <uses-feature> - hardware and software needed
  - <uses-sdk> - API used
  - <uses-library> - other libraries needed to be linked
  - <uses-configuration> - specific input features required
  - <uses-permission> - system permission needed to be granted
  - And others…

THE MANIFEST FILE
DECLARING COMPONENTS CAPABILITIES

- Include **<intent-filter>** element that declares the capabilities of the activity (*optional*)
  - So that it can respond to intents from other app
- E.g., if build email app with an activity to compose email , we can declare an intent-filter to respond to "send" intent (in order to send a new email):

```
<manifest ... >
  ...
  <application ... >
    <activity android:name="com.example.project.ComposeEmailActivity">
      <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <data android:type="*/*" />
        <category android:name="android.intent.category.DEFAULT" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

If another app creates an intent with the ACTION_SEND action and passes it to startActivity(), the system may start your activity so the user can draft and send an email.

THE MANIFEST FILE
DECLARING COMPONENTS CAPABILITIES

- Declare **device and software requirements**
- *Google Play* reads them and filter depending on user's devices capabilities.
- E.g., if an app requires a camera and Android 2.1 (API level 7):

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />
    ...
</manifest>
```

With such declaration, devices with NO camera or have Android version lower than 2.1 cannot install your app from Google Play.

APP RESOURCES

- Android App **needs more than code**!
  - Images, Audio, etc. that visualize the content
  - XML files to define style, animation, colours and layout
  - Alternative resources to define various languages and device sizes

APP RESOURCES

- **SDK build tools define unique integer ID for every resources in the app**
  - E.g., if your app contains logo.png (saved in res/drawable/ directory), SDK will generate a resource ID named R.drawable.logo
  - Can be used to **reference** the image and **insert** into the UI of the app
- Define **string translations** to other languages that stored in separate files, e.g., res / values-fr, for French translation.
- Or use it to **define the layout** when different orientation is in-used – the *qualifier* that define different device configurations.

ASSIGNMENT

- Anyone else still looking for group?