# 3. Mobile Application Models

## Objectives

- Explain the concept of an Activity
- Explain the Intent and Intent Filter
- Explain the lifecycle of an Activity
- Manage the Activity lifecycle

## Activity

- An Activity equals to an UI. An app can contain multiple activities
- One of the activity is the "main" activity that launching an app
- Activity can be started by another activity within the same app or other apps using the Intent

https://developer.android.com/guide/components/activities/intro-activities

## Activity

- An activity must be declared in the Manifest File in order for it to be accessible to the system

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      <activity android:name=".SecondExampleActivity" />

      ...
  </application ... >
  ...
</manifest >
```

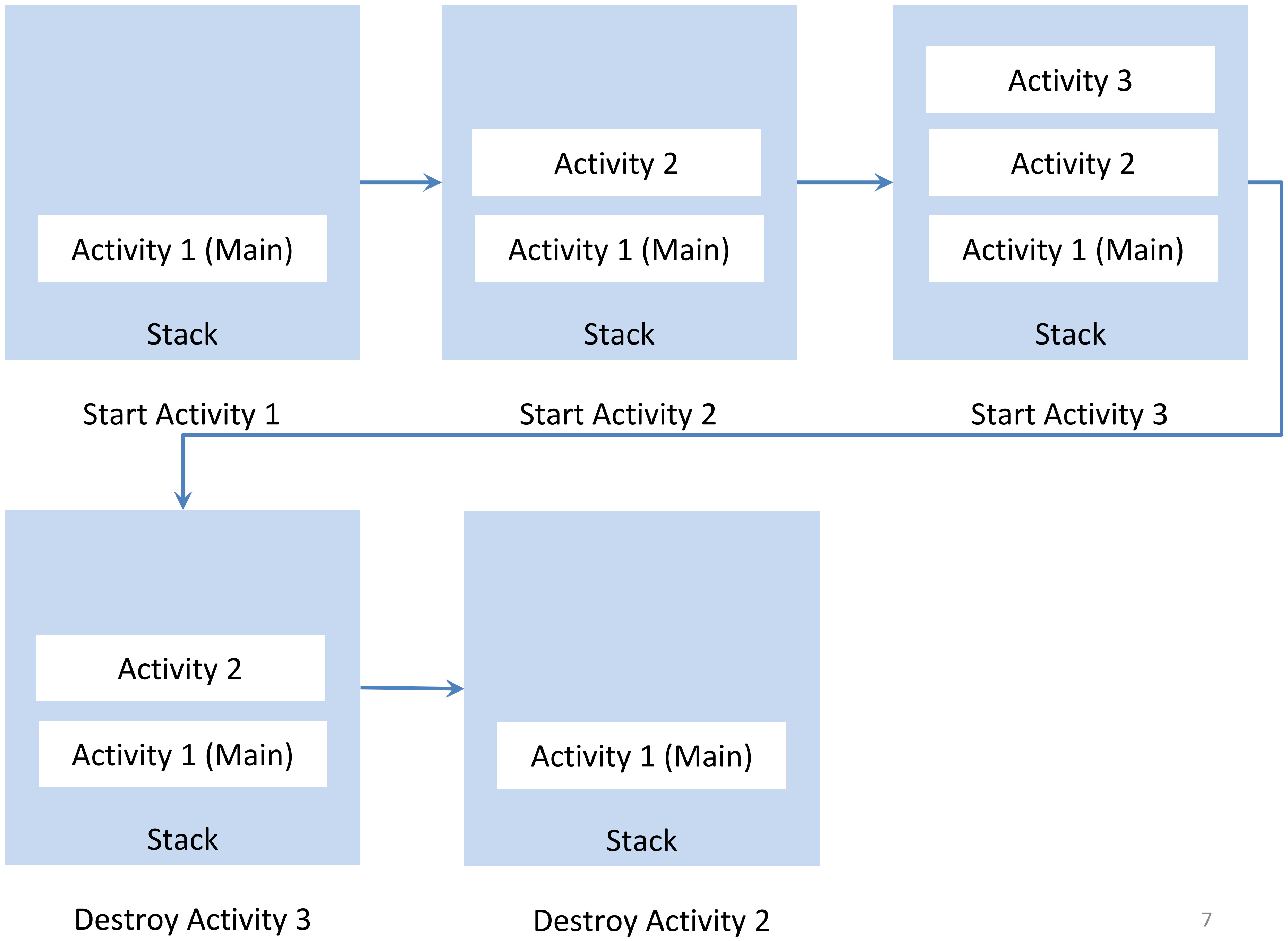
## Activity

- To shut down an Activity:
  - finish() method: shutdown current activity
  - finishActivity() method: shutdown a separate activity that was previously started

# Activity

- Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a <u>stack</u> (last in, first out)

- User press the *Back* button, the current activity is <u>popped</u> from the stack and the previous activity resumes


Activity 1 (Main)
Stack
Start Activity 1
Activity 2
Activity 1 (Main)
Stack
Start Activity 2
Activity 3
Activity 2
Activity 1 (Main)
Stack
Start Activity 3
Activity 2
Activity 1 (Main)
Stack
Destroy Activity 3
Activity 1 (Main)
Stack
Destroy Activity 2

# Question?

1. In the context of Android, what is the main purpose of an Activity?
2. Why it is important to declare an Activity in the manifest file?
3. What is the data structure used by Android to manage components of an app?
4. What will happen to an Activity if a user navigates away from it?

# Intent

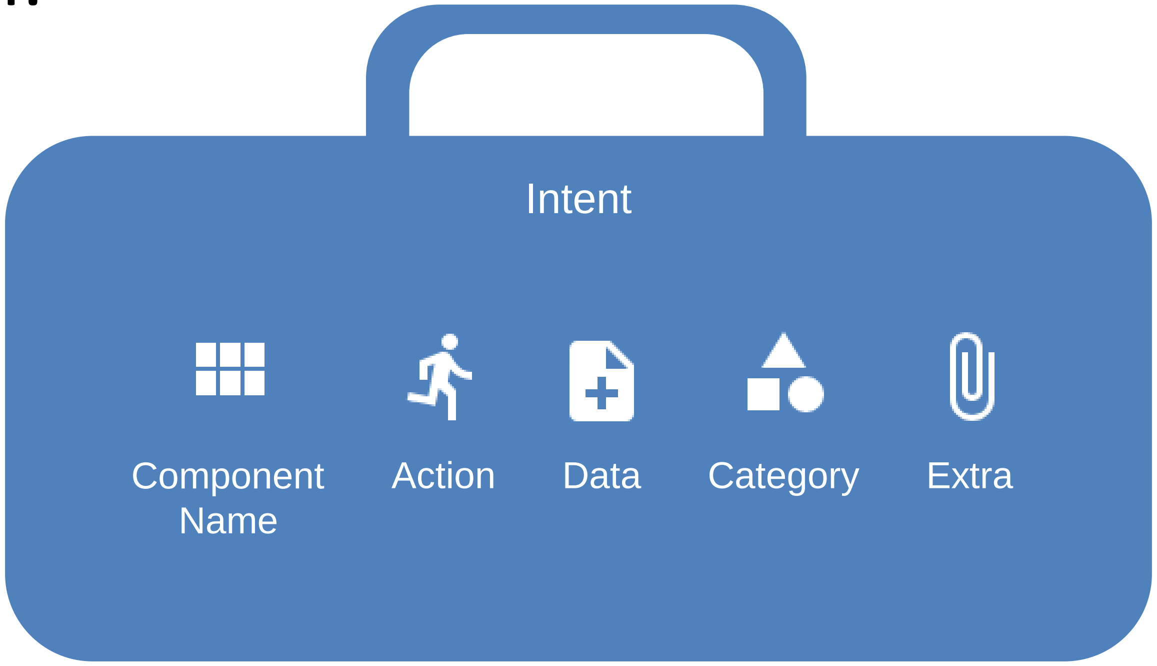
- Intent is used to start an Activity
- Two types of Intent:

| Type | Description |
|---|---|
| Explicit | To start a component in your own app. You need to supply a target app's package name or a fully-qualified component class name |
| Implicit | Declare a general action to perform, which allows a component from another app to handle it |

https://developer.android.com/training/basics/intents

# Intent

- An Intent carries the following information that the Android System uses to perform the action:



# Intent - Component

Component Name

- The exact name of the component to start

- A way for Explicit Intent to launch another Activity within an app

- When starting a Service, always specify the component name for better security

# Intent - Action

Action

- A string that specifies the generic action to perform

- It determines how the rest of the Intent is structured – data and extras

- Common actions are : `ACTION_VIEW` and `ACTION_SEND`

https://developer.android.com/reference/android/content/Intent.html

# Intent - Data

Data

- The Uniform Resource Identifier (URI) that references the data to be acted on and/or the Multipurpose Internet Mail Extension (MIME) type of that data

| Action | Data | Type |
|---|---|---|
| ACTION_VIEW | content://contacts/people/1 | |
| ACTION_VIEW | tel:031234567 | |
| ACTION_VIEW | geo:47.6,-122.3 | |
| ACTION_VIEW | http:www.example.com | |
| ACTION_SEND | This is my text to send | text/plain |
| ACTION_SEND | (URI to image) | image/jpeg |

# Intent - Category

Category

- This is optional
- Provides additional information about the kind of component that should handle the intent
- Common categories : CATEGORY_BROWSABLE and CATEGORY_LAUNCHER

## Intent - Extra

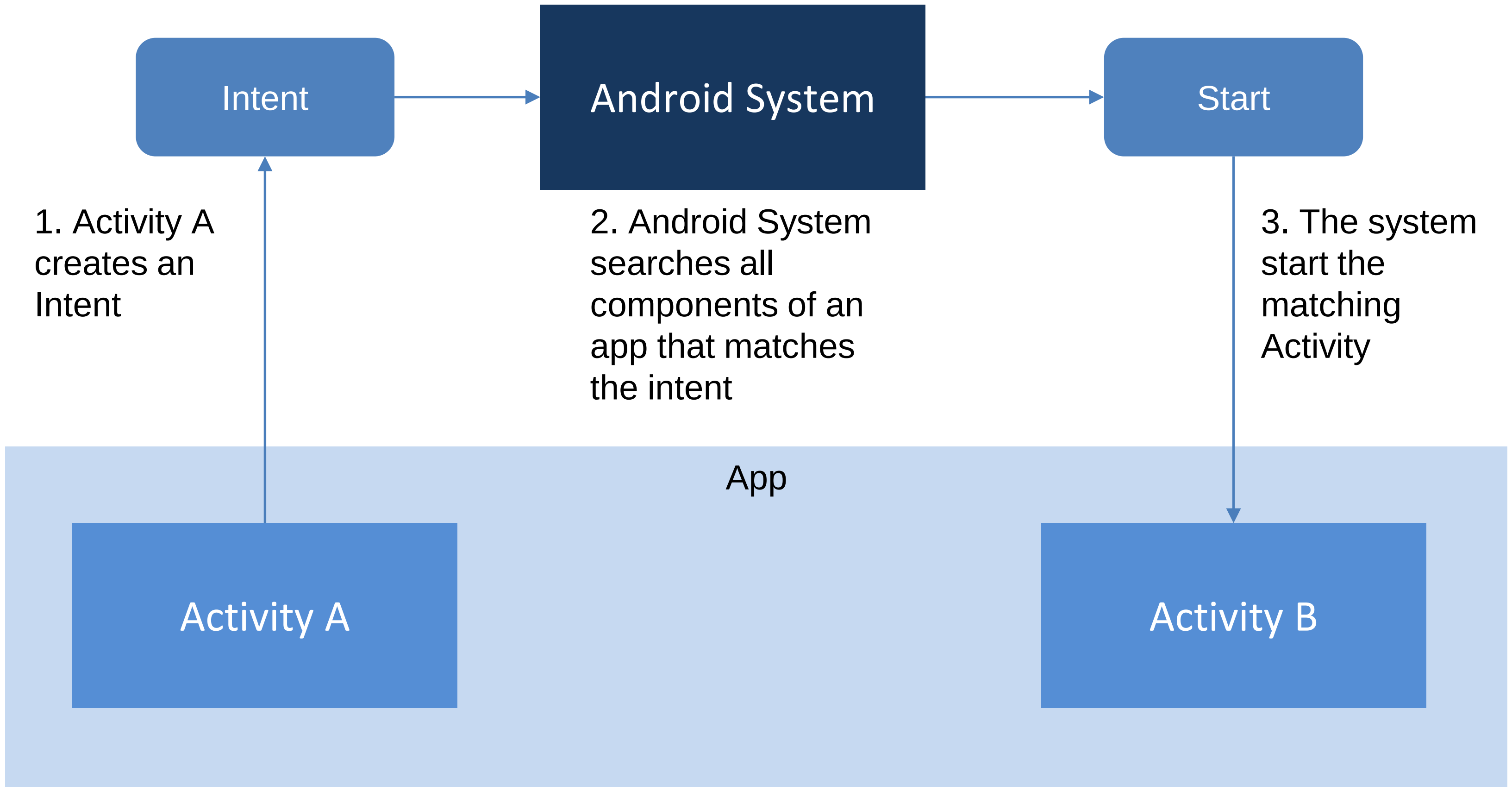
Extra

- Key-value pairs that carry additional information required to accomplish the requested action

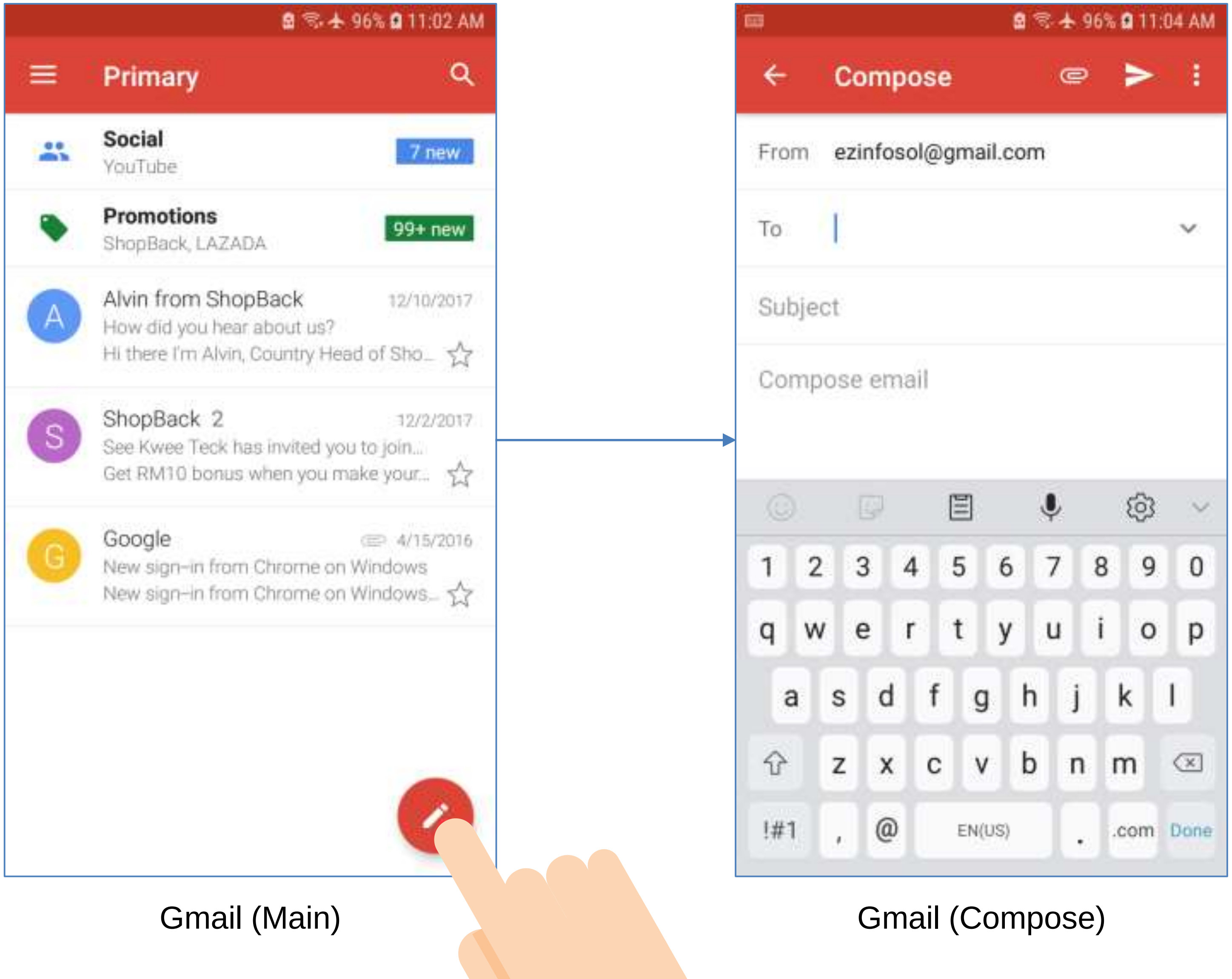
- Use the `putExtra(Key, Value)` method to insert data to an Intent

- Use the `getExtras()` to retrieve data from an Intent

# Intent - Explicit


Intent
Android System
Start
1. Activity A creates an Intent
2. Android System searches all components of an app that matches the intent
3. The system start the matching Activity
App
Activity A
Activity B

# Intent - Explicit


96% 11:02 AM
Primary
Social
YouTube
7 new
Promotions
ShopBack, LAZADA
99+ new
Alvin from ShopBack
12/10/2017
How did you hear about us?
Hi there I'm Alvin, Country Head of Sho...
ShopBack 2
12/2/2017
See Kwee Teck has invited you to join...
Get RM10 bonus when you make your...
Google
4/15/2016
New sign-in from Chrome on Windows
New sign-in from Chrome on Windows...
96% 11:04 AM
Compose
From ezinfosol@gmail.com
To
Subject
Compose email
!#1
EN(US)
.com
Done

Gmail (Main)

Gmail (Compose)

# Intent - Explicit

- You can start another activity by calling startActivity(), passing it an Intent that describes the activity you want to start.

Codes in the Gmail main Activity:

Kotlin

```kotlin
intent = Intent(this, ComposeActivity::class.java)
startActivity(intent)
```

Java

```java
Intent intent = new Intent(this, ComposeActivity.class);
startActivity(intent);
```

## Intent - Explicit

- An Intent carries data from one Activity to another using the Extra

Kotlin

```
const val EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE“
const val EXTRA_VALUE = "com.example.myfirstapp.VALUE"
...
val intent = Intent(this, DisplayMessageActivity::class.java).apply {
                putExtra(EXTRA_MESSAGE, message)
            }
startActivity(intent)
```

---

Java

```
public static final String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
...
Intent intent = new Intent(this, DisplayMessageActivity.class);

EditText editText = (EditText) findViewById(R.id.editText);
String message = editText.getText().toString();

intent.putExtra(EXTRA_MESSAGE, message);
startActivity(intent);
```

## Intent - Explicit

- To receive data from an Activity, use the Intent's get methods

Kotlin

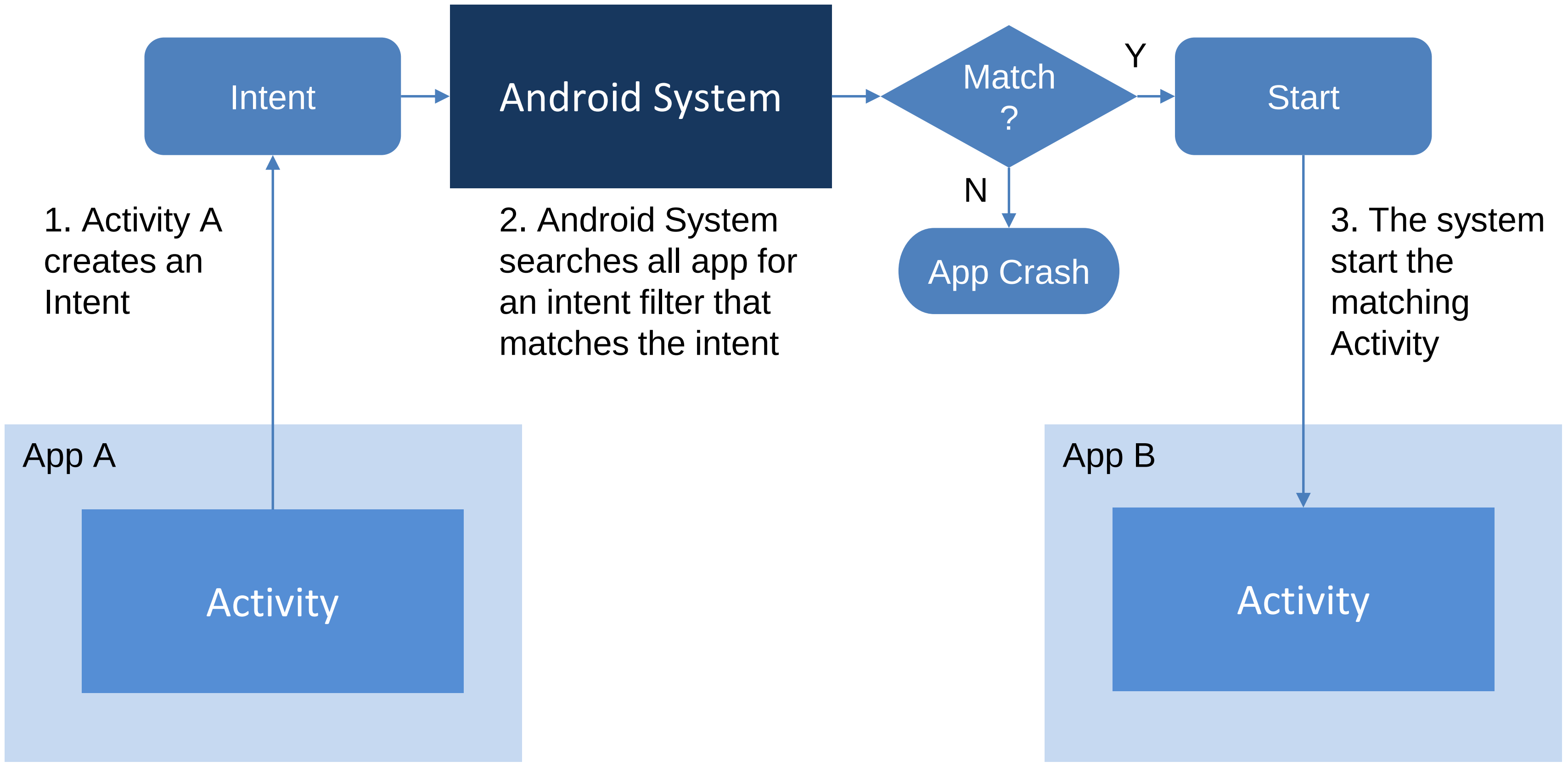
```kotlin
// Get the Intent that started this activity and extract the string
val message = intent.getStringExtra(EXTRA_MESSAGE)
```

Java

```java
// Get the Intent that started this activity and extract the string
Intent intent = getIntent();

String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

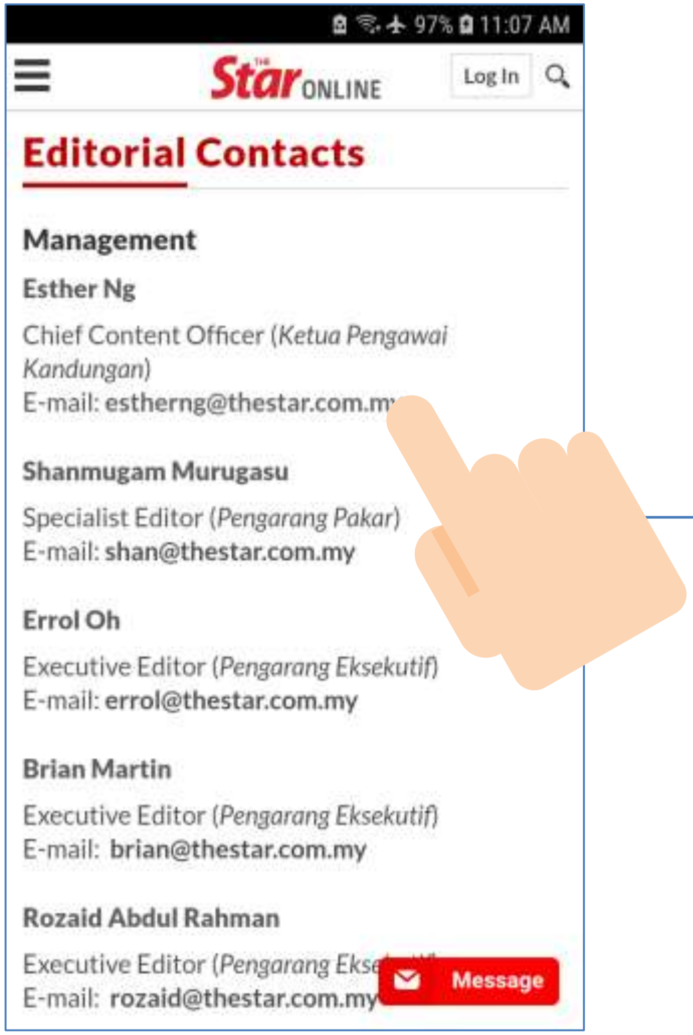
https://developer.android.com/reference/android/content/Intent#public-methods

# Intent - Implicit



Intent
Android System
Match ?
Y
N
Start
App Crash
1. Activity A creates an Intent
2. Android System searches all app for an intent filter that matches the intent
3. The system start the matching Activity
App A
Activity
App B
Activity

# Intent - Implicit



**Editorial Contacts**

**Management**

**Esther Ng**
Chief Content Officer (*Ketua Pengawai Kandungan*)
E-mail: estherng@thestar.com.my

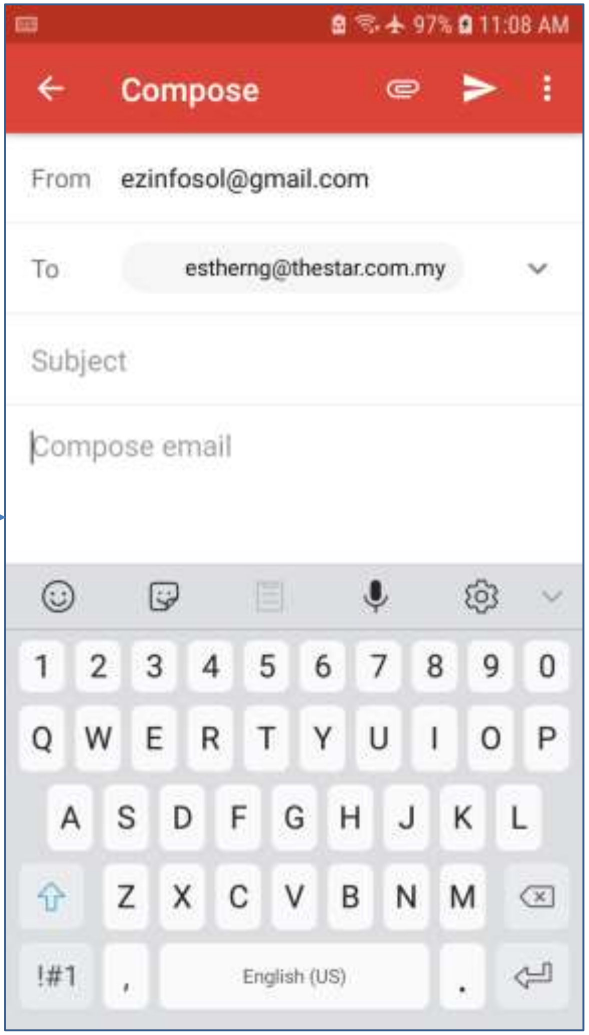
**Shanmugam Murugasu**
Specialist Editor (*Pengarang Pakar*)
E-mail: shan@thestar.com.my

**Errol Oh**
Executive Editor (*Pengarang Eksekutif*)
E-mail: errol@thestar.com.my

**Brian Martin**
Executive Editor (*Pengarang Eksekutif*)
E-mail: brian@thestar.com.my

**Rozaid Abdul Rahman**
Executive Editor (*Pengarang Eks*
E-mail: rozaid@thestar.com.my

Browser (Contact)



From ezinfosol@gmail.com
To estherng@thestar.com.my
Subject
Compose email

Gmail (Compose)

# Intent - Implicit

Kotlin

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);
// The intent does not have a URI, so declare the "text/plain" MIME type
emailIntent.setType(HTTP.PLAIN_TEXT_TYPE);

// recipients
emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[] {"jon@example.com"});

emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Email subject");
emailIntent.putExtra(Intent.EXTRA_TEXT, "Email message text");
emailIntent.putExtra(Intent.EXTRA_STREAM,
                          Uri.parse("content://path/to/email/attachment"));
```

Java

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);
// The intent does not have a URI, so declare the "text/plain" MIME type
emailIntent.setType(HTTP.PLAIN_TEXT_TYPE);

// Recipients
emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[] {"jon@example.com"});

emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Email subject");
emailIntent.putExtra(Intent.EXTRA_TEXT, "Email message text");
emailIntent.putExtra(Intent.EXTRA_STREAM,
                          Uri.parse("content://path/to/email/attachment"));
```

# Intent - Implicit

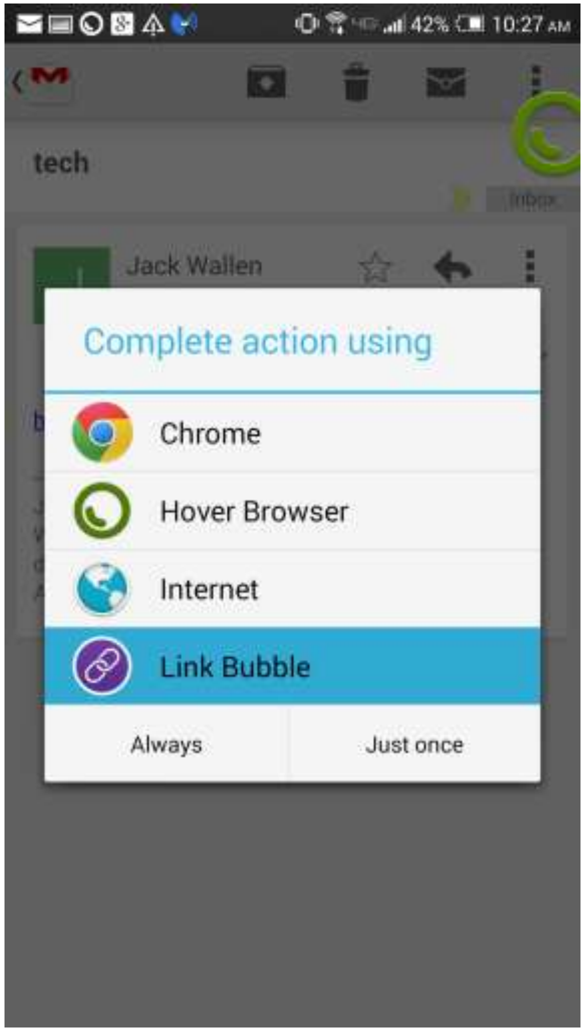
Examples of implicit intent for call and map.

Kotlin

```kotlin
val callIntent: Intent = Uri.parse("tel:5551234").let { number ->
    Intent(Intent.ACTION_DIAL, number)
}

// Map point based on address
val mapIntent: Intent = Uri.parse(
        "geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California"
).let { location ->
    // Or map point based on latitude/longitude
    // Uri location = Uri.parse("geo:37.422219,-122.08364?z=14"); // z param is
zoom level
    Intent(Intent.ACTION_VIEW, location)
}
```

# Intent - Implicit

- If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use



# Intent - Implicit

- It's possible that a user won't have *any* apps that handle an implicit intent

Kotlin

```kotlin
// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(packageManager) != null) {
    startActivity(sendIntent)
} else
    // Inform user intent is unresolved
}
```

Java

```java
// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
} else
    // Inform user intent is unresolved
}
```

## Question?

1. What is the main difference between explicit and implicit intents?
2. Among explicit and implicit intents, which one is suitable for each of the following tasks:
    a. To share a web site URL
    b. To listen to an audio file downloaded from the Internet
    c. To change the system sound profile to silent mode

# Intent Filter

- To advertise which implicit intents your app can receive using the `<intent-filter>` element

E.g. An activity declaration with an intent filter to receive an ACTION_SEND intent when the data type is text

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

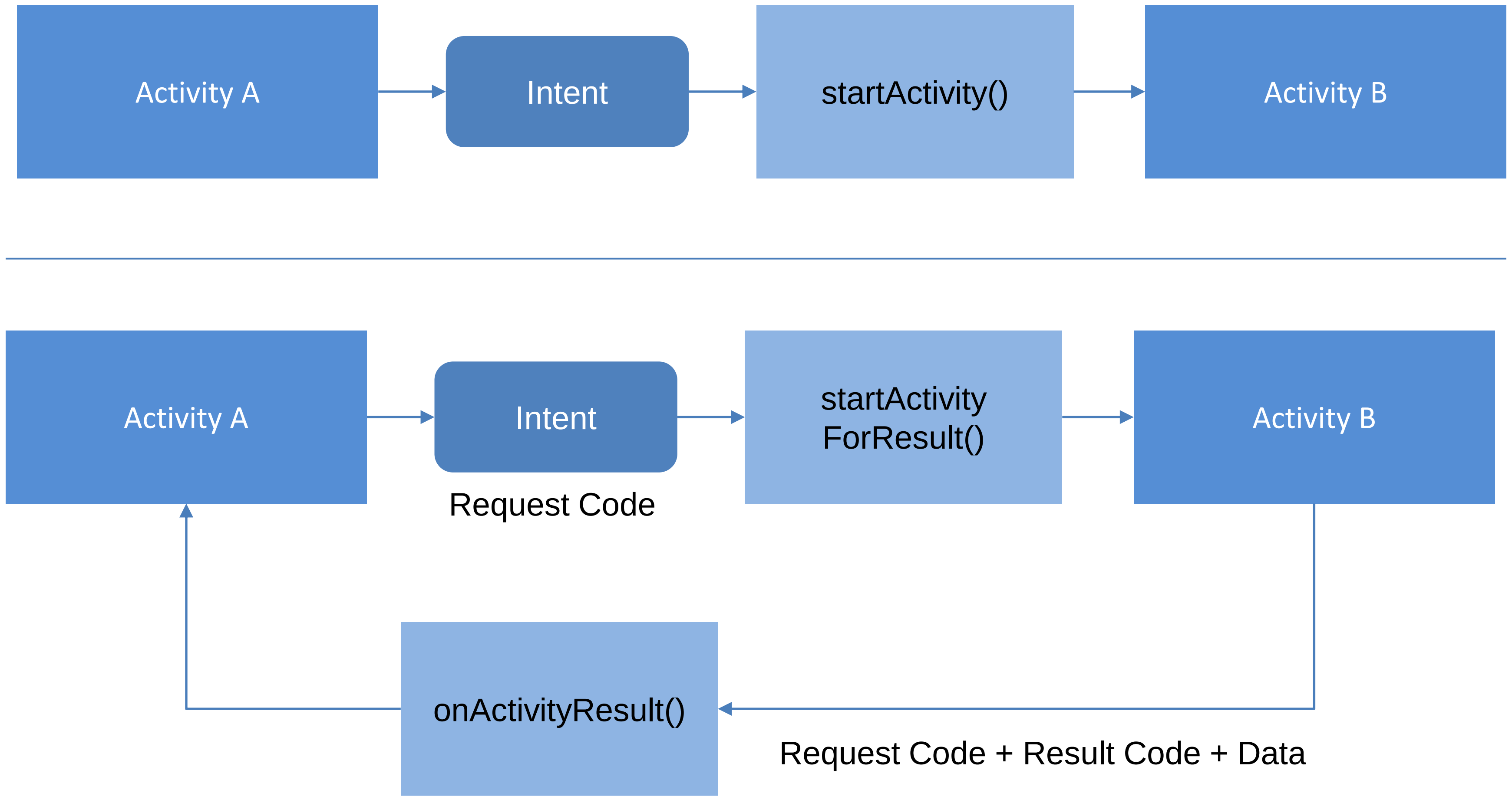
https://developer.android.com/guide/components/intents-filters.html

# Getting a result from an Activity

- You can start an Activity and receive a result back.

- Two methods:
  - startActivityForResult() method: start the activity which will return a result
  - onActivityResult() callback method : to receive the result from the subsequent activity

# Getting a result from an Activity



Kotlin

```
const val PICK_CONTACT_REQUEST = 1  // The request code
...
private fun pickContact() {
    // Show user only contacts w/ phone numbers
    Intent(Intent.ACTION_PICK, Uri.parse("content://contacts"))
          .also { pickContactIntent ->pickContactIntent.type =Phone.CONTENT_TYPE

          startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST)
    }
}


...
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == Activity.RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which contact was selected.

            // Do something with the contact here (bigger example below)
        }
    }
}
```

Java

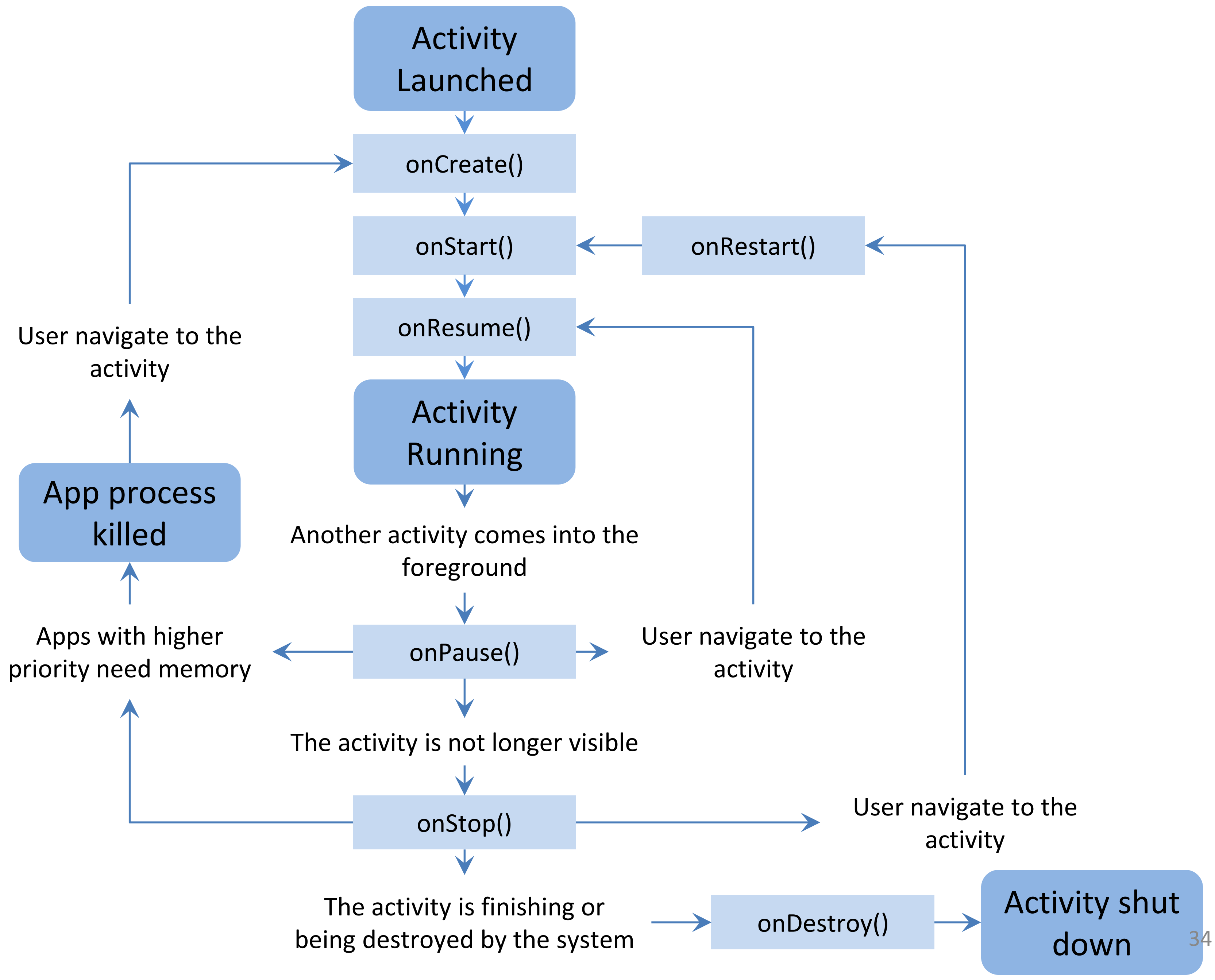
```java
static final int PICK_CONTACT_REQUEST = 1;  // The request code
...
private void pickContact() {
    Intent pickContactIntent = new Intent(Intent.ACTION_PICK,
                                          Uri.parse("content://contacts"));

    // Show user only contacts w/ phone numbers
    pickContactIntent.setType(Phone.CONTENT_TYPE);
    startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
}

...
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which contact was selected.

            // Do something with the contact here (bigger example below)
        }
    }
}
```

## Activity Lifecycle

- 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 callbacks that allow the Activity to know that a state has changed:
  1. onCreate
  2. onStart
  3. onResume
  4. onPause
  5. onStop
  6. onDestroy

## Activity Lifecycle


Activity Launched
onCreate()
onStart()
onRestart()
onResume()
User navigate to the activity
Activity Running
App process killed
Another activity comes into the foreground
Apps with higher priority need memory
onPause()
User navigate to the activity
The activity is not longer visible
onStop()
User navigate to the activity
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

# Activity Lifecycle - States

| State | Resumed ▶ | Paused ⏸ | Stopped ⏹ | Destroyed ⊗ |
|---|---|---|---|---|
| Visible | Yes | Yes (Partially) | No | No |
| User Interaction | Yes | No | No | No |
| Code Execution | Yes | No | No | No |
| Instance State | Saved | Saved | Saved | Destroyed |

# Activity – 1. onCreate

- It creates the Activity
- Performs basic application <u>startup logic</u> that should <u>happen only once</u> for the entire life of the activity
- E.g. Setup the app UI and instantiate some class-scope variables

# Activity – 2. onStart

- Makes the Activity visible to the user
- Activity enters the foreground
- Activity becomes interactive
- Initializes the code that maintains the UI. E.g. Drawing visual elements and running animations

## Activity – 3. onResumed

- Activity comes to the foreground
- Could be called several times:
  - After onStart(): first call
  - After onPause(): user return to the activity
- Initialize components that you release during onPause(), and perform any other initializations that must occur each time the activity enters the Resumed state. E.g. starting a camera preview

## Activity – 4. onPause

- User is leaving an activity. Does not always mean the activity is being destroyed

- The Activity is no longer in the foreground but still visible in multi-window mode

- Commit any changes that should be persisted beyond the current user session

- Execution must very brief. Stop things that consume CPU

# Activity – 5. onStop

- An activity is no longer visible to the user
- Releases almost all resources that aren't needed
- Performs relatively CPU-intensive shutdown operations
- E.g. save information to a database

## Activity – 6. onDestroy

- It is called because:
  - the `finish()` is called or the system destroying the process containing the activity to save space
  - an orientation change occurs (will be discussed later)
- Releases all resources
- The system may skip this callback

## Activity - Configuration Change

- Configuration Change:
  - E.g. screen orientation, language, and input devices
  - Current activity will be *destroyed*, going through the normal activity lifecycle process of onPause(), onStop(), and onDestroy()

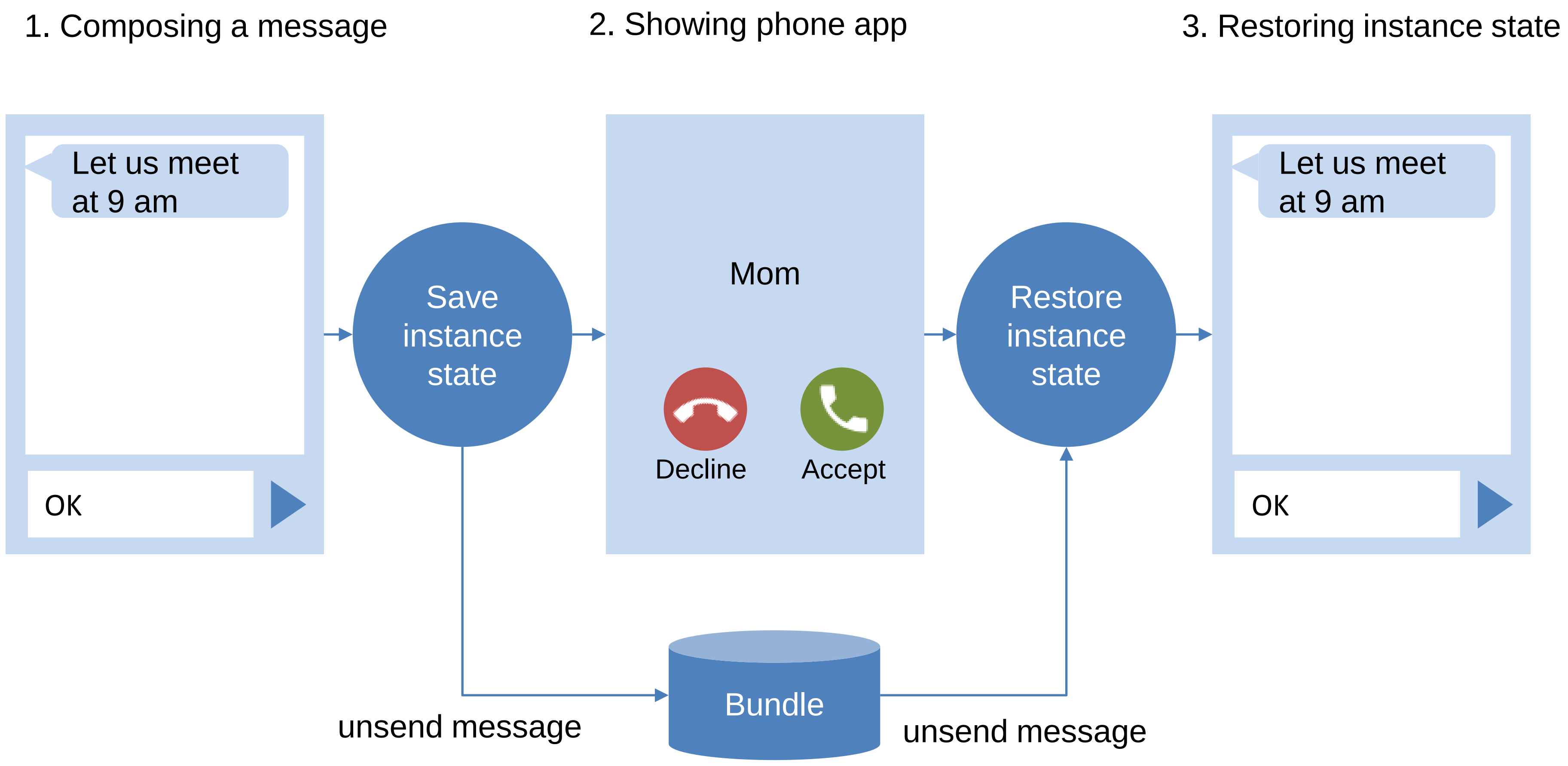
## Question?

1. Why the onCreate and onPause callbacks are important to an Activity?
2. Describe a scenario where onPause and onStop would not be invoked.
3. Identify a suitable callback to implement the following tasks:
   a. Playing background music
   b. Save game level
   c. Establish connection to a server

## Saving and restoring activity state

- Instance state = the saved data that the system uses to restore the previous state
- It is a collection of key-value pairs stored in a Bundle object
- The system uses the Bundle instance state to save information about each UI component. E.g. value entered by user in a text field

# Saving and restoring activity state



## Saving and restoring activity state

Kotlin

```
lateinit var textViewMsg: TextView
...
override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
    textViewMsg.text = savedInstanceState?.getString(MSG_VIEW_KEY)
}

override fun onSaveInstanceState(outState: Bundle?) {
    outState?.run {
                putString(MSG_VIEW_KEY, textViewMsg.text.toString())
    }
    // call superclass to save any view hierarchy
    super.onSaveInstanceState(outState)
}
```

# Saving and restoring activity state

Java

```java
TextView textViewMsg;
String msgState;
...
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    textViewMsg.setText(savedInstanceState.getString(MSG_VIEW_KEY));
}

@Override
public void onSaveInstanceState(Bundle outState) {
    outState.putString(MSG_VIEW_KEY, textViewMsg.getText());

    // call superclass to save any view hierarchy
    super.onSaveInstanceState(outState);
}
```

# Review Questions

1. Multi-window and multitasking is a common feature of modern mobile devices. As a mobile app developer, explain how to handle the transition from one window/task to another to ensure important information is not lost.

2. Identify two callbacks/methods that could be used to save data in a database when an Activity moves between states.