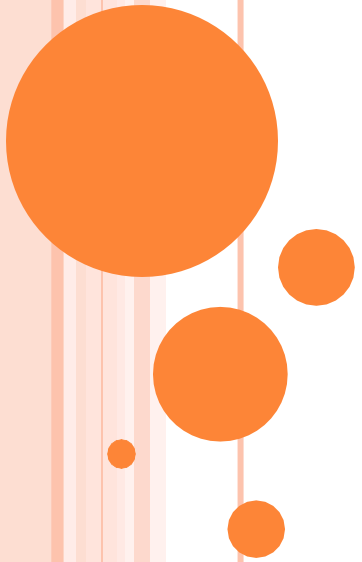
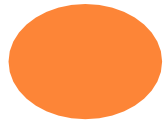
ACTIVITIES AND INTENTS



Activity

- Contains the user interface of a page in an application
- Applications have one or more activities
- Main purpose of an activity is to interact with the user through UI
- **Activity's life Cycle -** From the moment the activity starts processing the UI to the moment its footprint is cleared from the memory, it goes through a number of stages
- **Understand Activity life cycle -** to ensure app works correctly



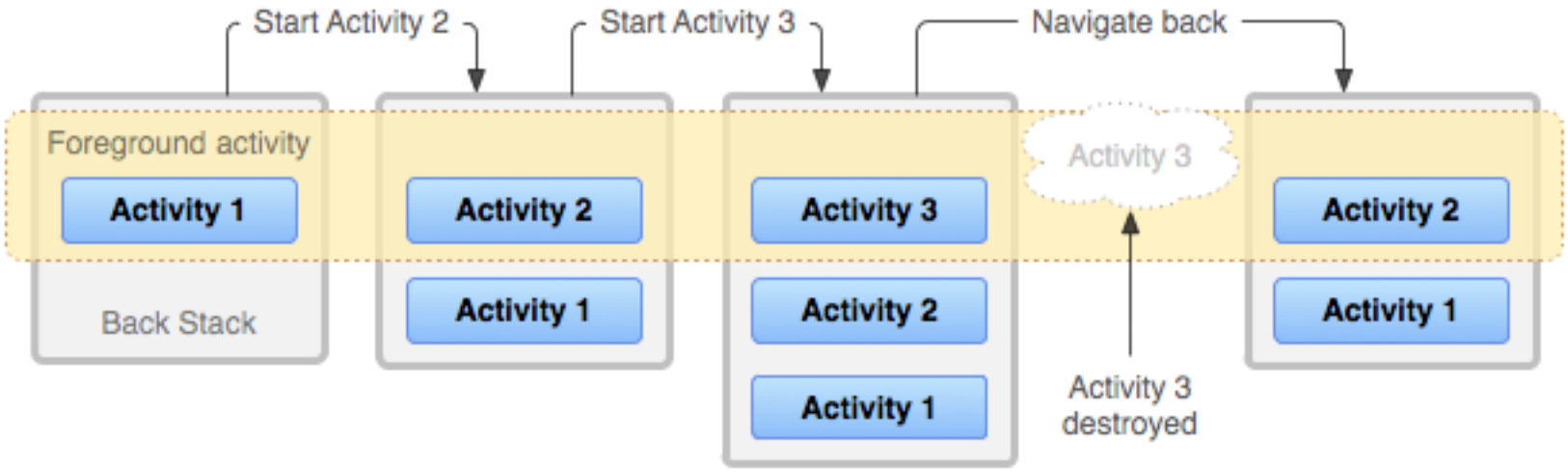
Create Activity

- To create an activity, create a java class that extends the Activity base class
  - public class Home extends Activity {}
- Activity class loads its UI components using the XML file defined in *res/layout* folder

  setContentView(R.layout.home_layout);
- Every Activity in the application must be declared in your *AndroidManifest.xml* file

Navigation Between Activities

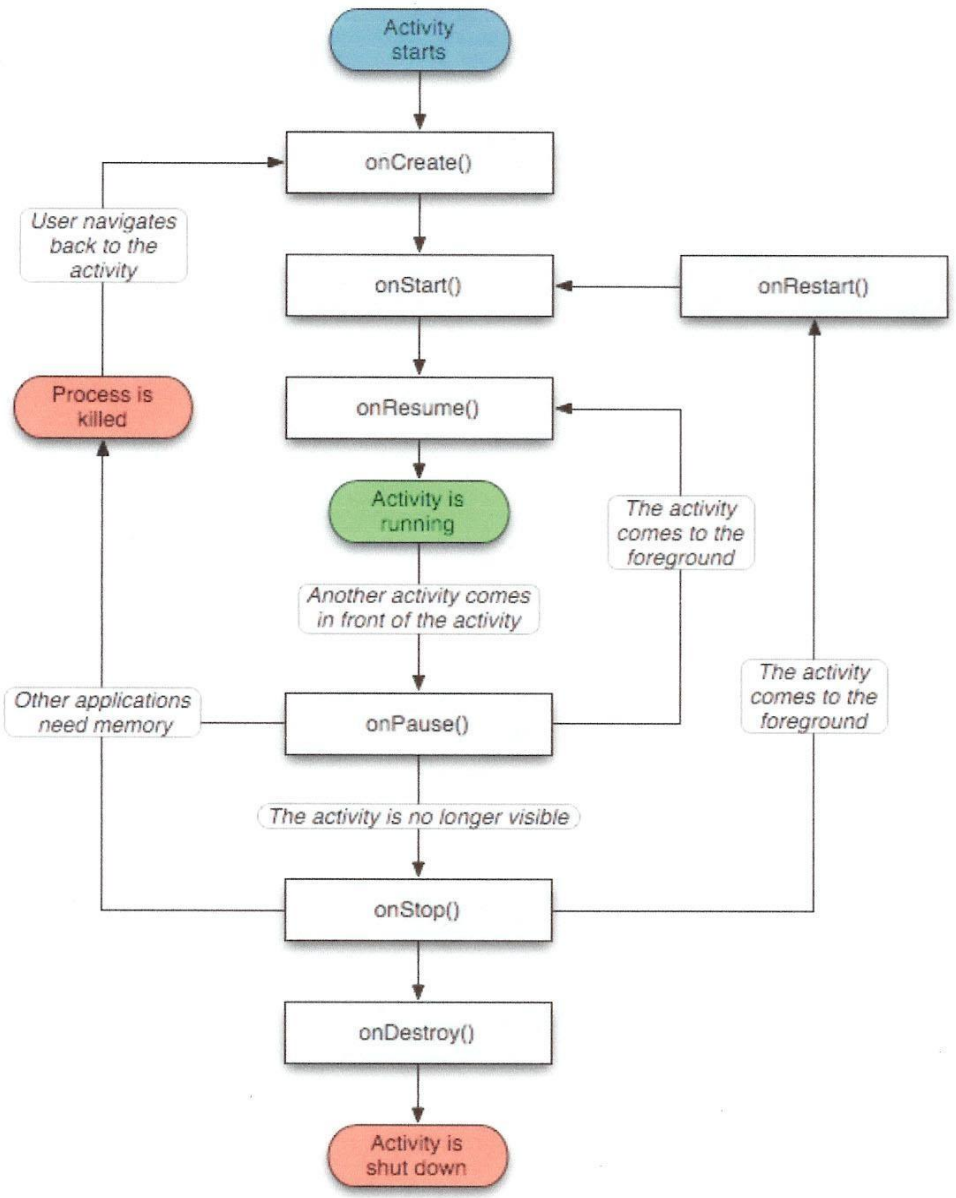
A representation of how each new activity in a task adds an item to the back stack. When the user presses the Back button, the current activity is destroyed and the previous activity resumes.



Source: StackOverfalw

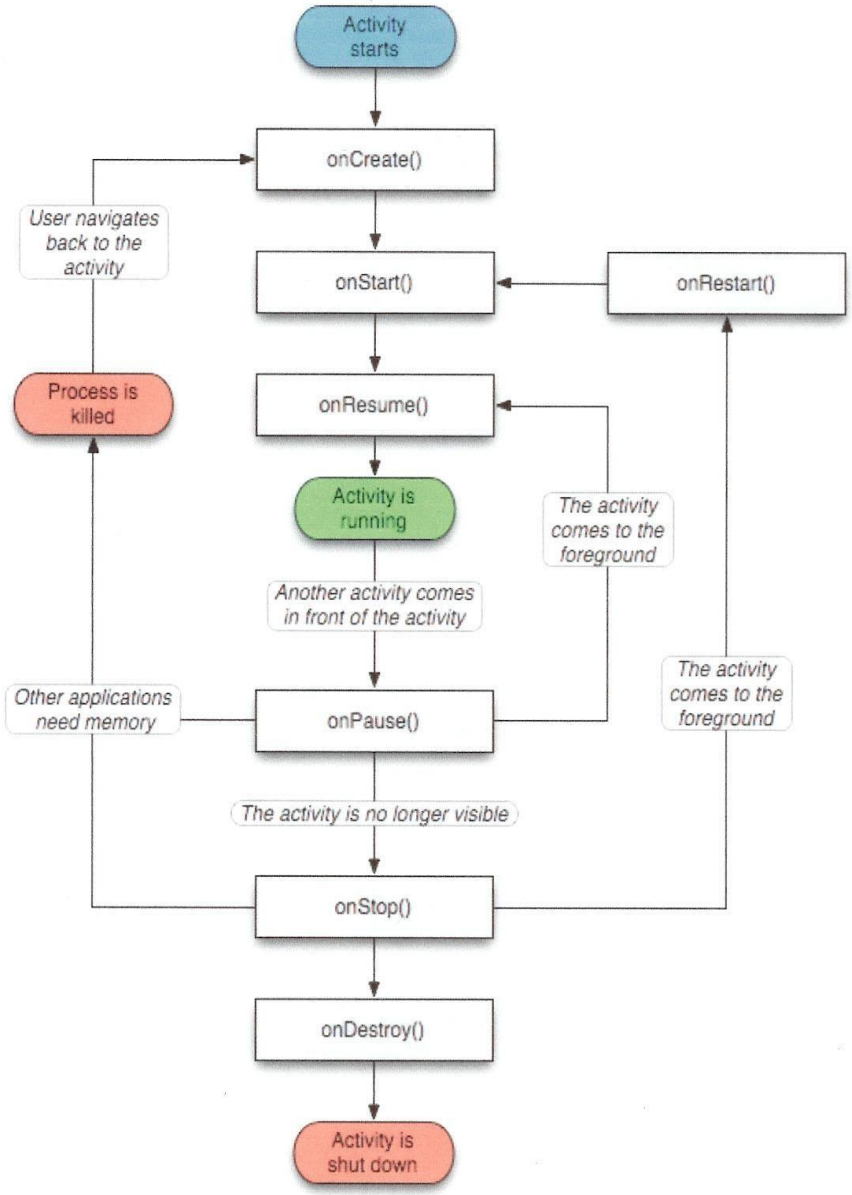
Activity Life Cycle

- Activity base class define a series of events/stages that govern the life cycle of an activity
  - A triggered event invokes a method/function



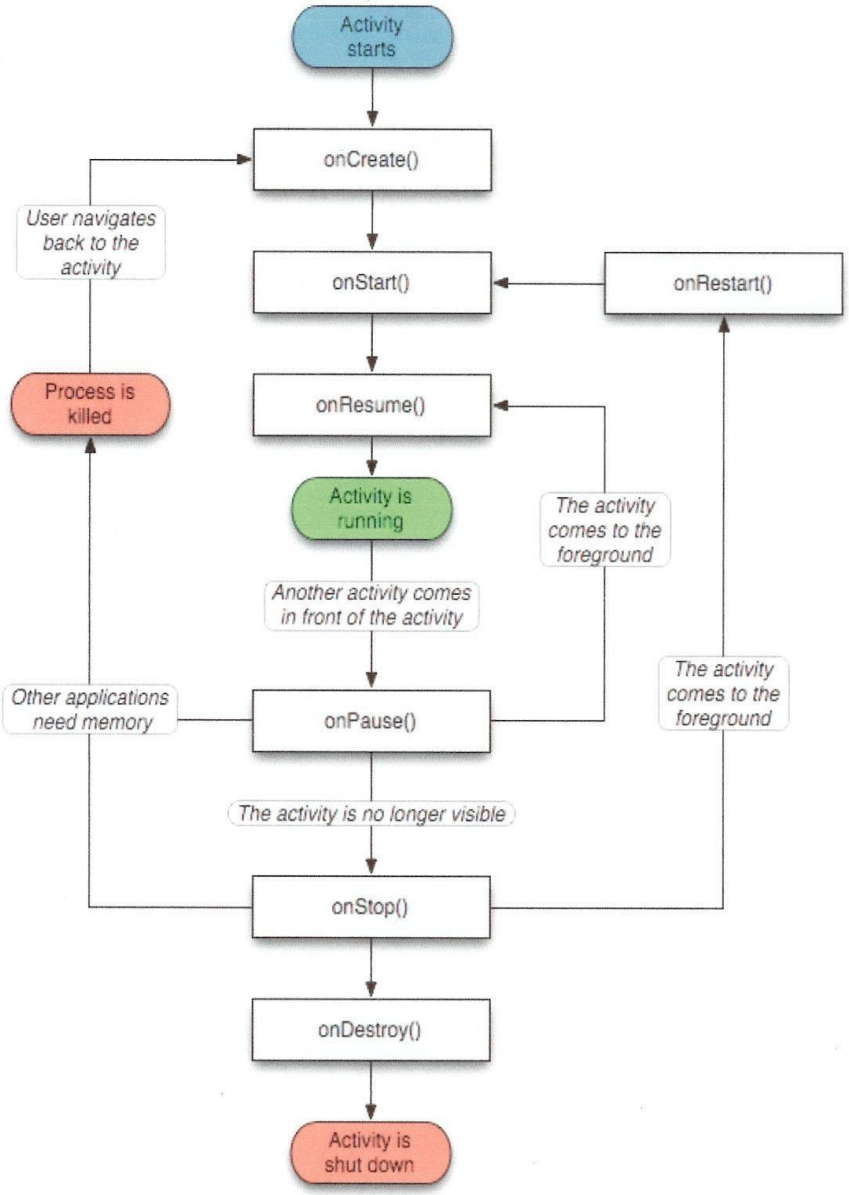
Activity Life Cycle

- **onCreate()**
  - Called when the activity is first created
  - By default, a created activity always contains the onCreate() event
  - Within onCreate() event handler, write the code to display the UI elements of screen.
    - setContentView(R.layout.main);
  - Use onCreate() method to create and instantiate the objects to be used in the application
    - EventCalendar ec = new EventCalendar();
- Splash/Loading Screen ??



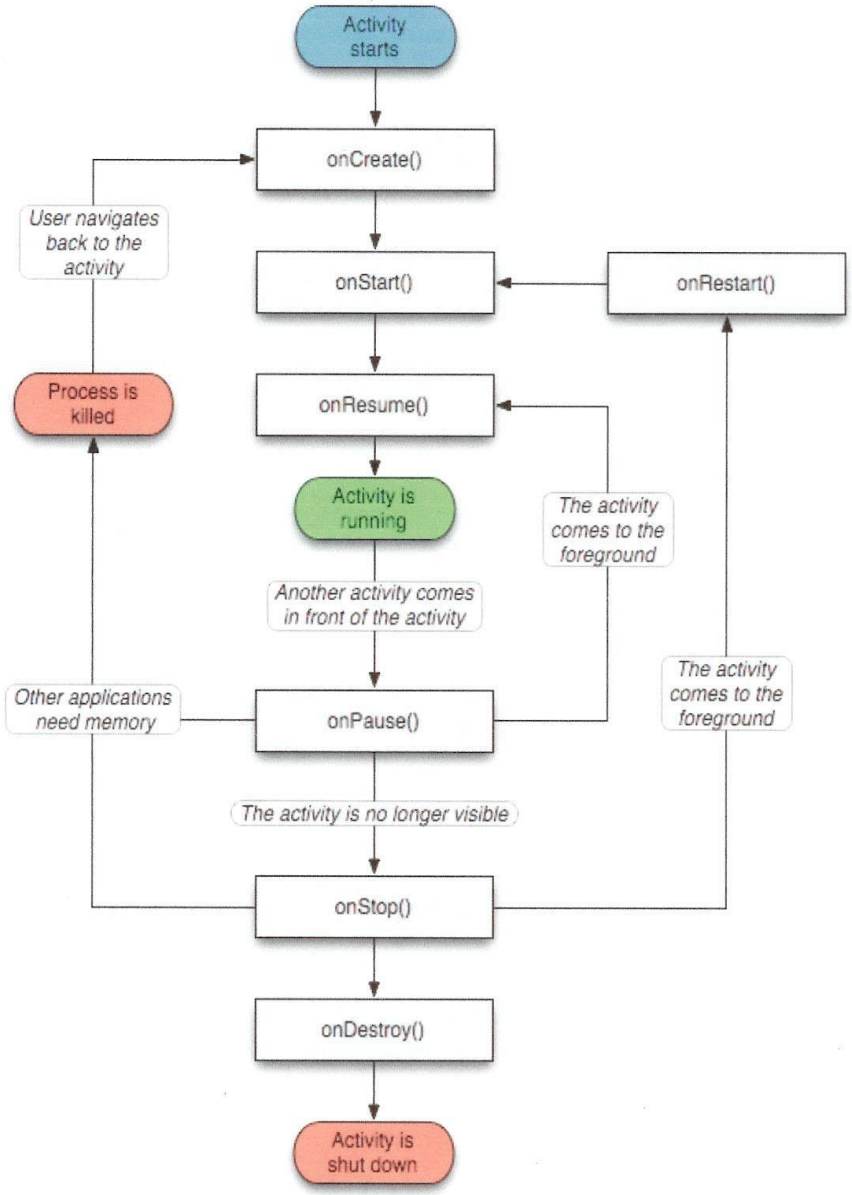
Activity Life Cycle (Cntd…

- **OnStart()**
  - Called when the activity becomes visible to the user
  - To initiate the "visible" lifespan of the application (any time between **onStart** and **onStop)**
  - Either be **onResume'd** or **onStop'ped** from this state.
  - When an activity is started, both the onStart() and onResume() methods are called one after another whether the activity is restored from the background or newly created.
  - An event for **onRestart**, which is called before **onStart** if the application is transitioning from **onStop** to **onStart** instead of being started from scratch.

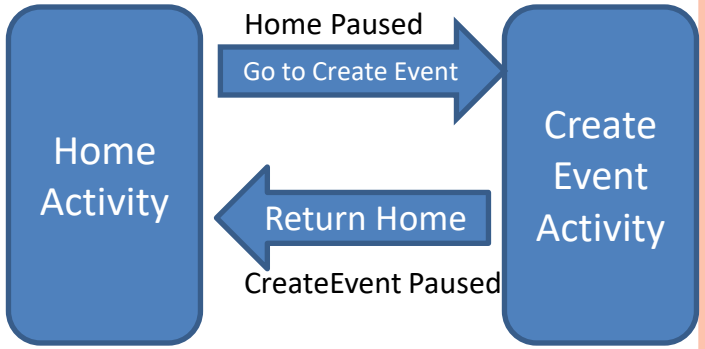


Activity Life Cycle (Cntd…

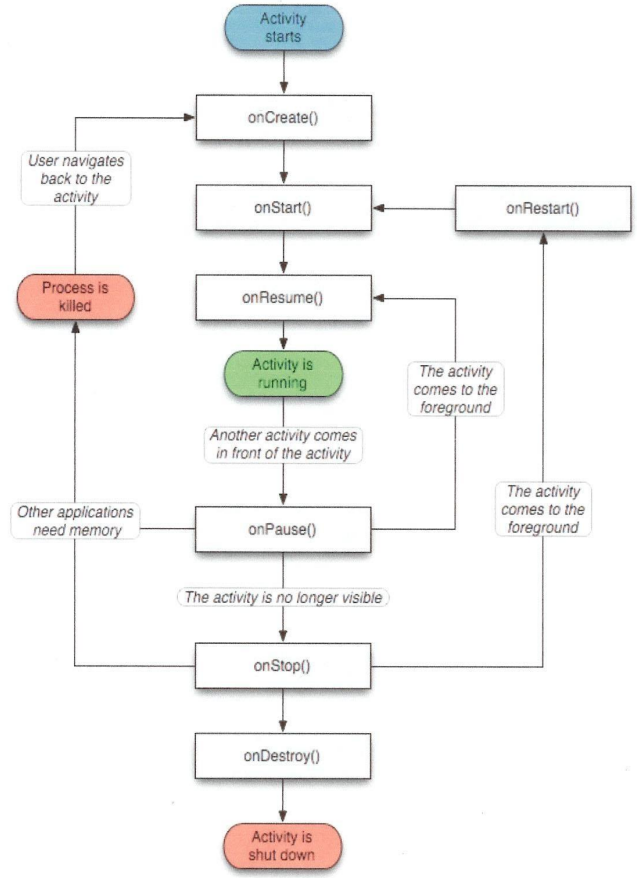
- **onResume()**
  - Called when the activity starts interacting with the user
  - Use the onResume( ) method to start
    - any background services
      - i.e. Broadcast Receiver, etc.
    - or any code that needs to run when your activity is in the foreground
      - Timer – StopWatch, CountDown, etc.



Activity Life Cycle (cntd…)



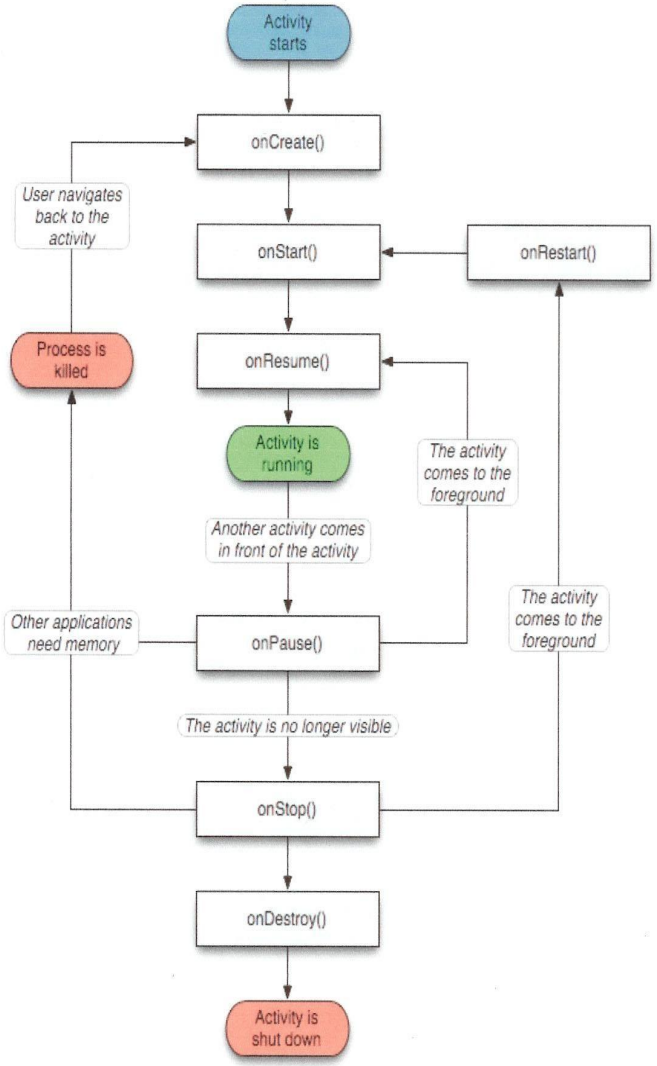
- **onPause()**
  - Called when the activity is being paused or the previous activity is being resumed.
  - Called in two scenarios-
    - when activity sent to back ground
      - A new page appears
    - when the activity is killed because of pressing the back button
  - Use the onPause() method
    - To stop any services/threads
    - Or to free some memory that is not required when your activity is not in the foreground
  - Either be **onResume'd** or **onStop'ped** from this state
    - **onResume** - the activity comes to foreground
    - **onStop-** the activity is no loner visible



Activity Life Cycle (cntd…)

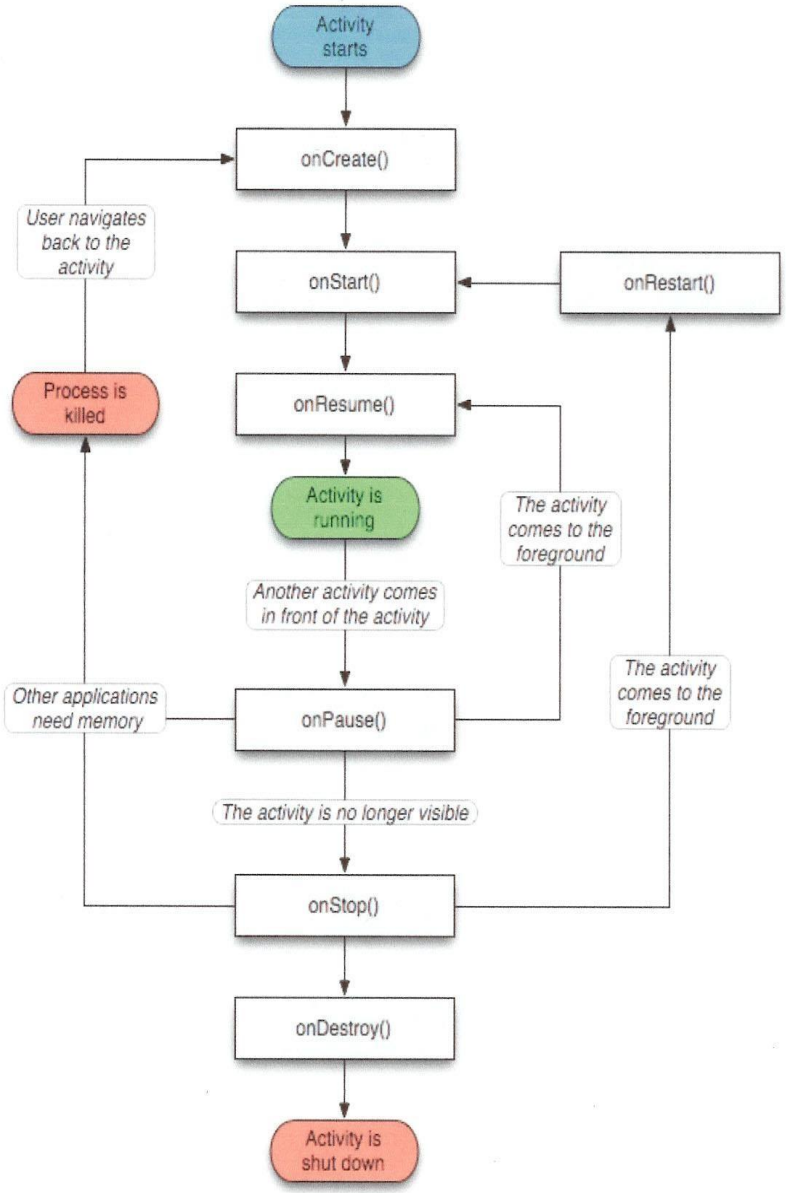
- **onStop()**
  - Called when the activity is no longer visible to the user
  - End of the current visible lifespan of the app
  - Either be **onRestart'ed** or **onDestroy'ed** from this state
    - **onRestart** - the activity to become visible again
    - **onDestroy** – for shutting down the activity.

- **onRestart()**
  - Called when the activity has been stopped and is restarting again
  - What should we do in onRestart()?
  - If onCreate() is executed, will onRestart() be executed? Why?



Activity Life Cycle (Cntd…)

- **onDestroy()**
  - Called before the activity is destroyed by the system
    - either manually or by the system to conserve memory
  - Use the onDestroy method to free up resources before the activity is destroyed.
    - Free the data in EventCalendar object
    - Store the data persistently (in file/database) if required
  - Called when the Java class is about to be destroyed.
  - Once this function is called, there is only one option for transition (other than being killed): **onCreate**.



Intents

- Applications can have one or more activities,
  - so need to navigate from one to another.

- In Android - navigation between activities is happened through Intent

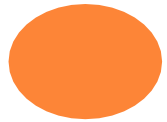
- Intents- “glue" that enables different activities from different applications to work together, ensuring that tasks can be performed **as though they all belong to one single application**.

- ***Intents*** are used to share content and trigger actions within and among applications

- Moreover, an Intent carries information about any android component to/from Android Runtime system

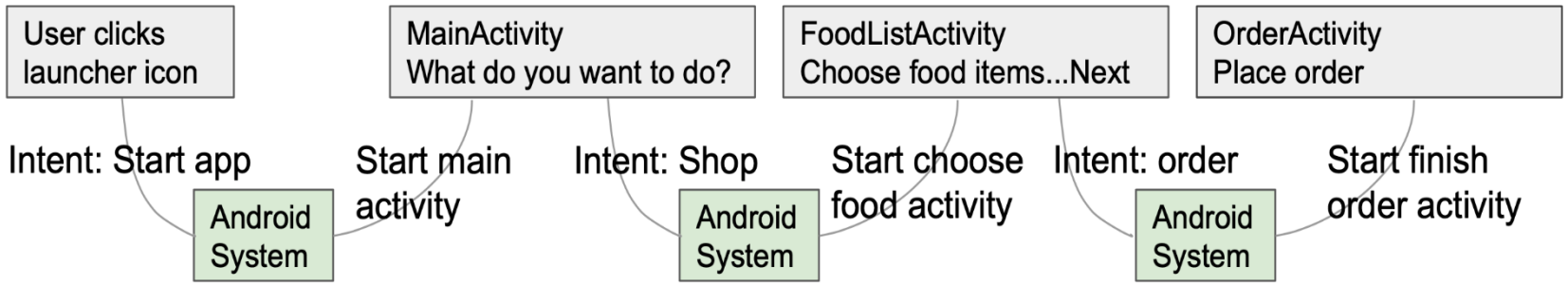
AN INTENT TO ANDROID RUNTIME

- An Intent object carries information that the Android system uses to determine two things –
  - which component to start (component name or component category that should receive the intent)
    - Notification, Alarm, Activity, Service, etc.
  - information that recipient component uses in order to perform the action (such as the action to take and the data to act upon).
    - Notification message/data, Alarm audio, any data passing to another activity



How Intents Work

- All activities are managed by the Android runtime
- Started by an "intent", a message to the Android runtime to run an activity



Building an Intent

- The primary information contained in an Intent are the following: (passed as an argument)

  - Component name – to be opened

    ```
    Intent i = new Intent(this, SecondActivity.class);
    ```

  - Action – to be taken
    - Data – information about an action
    - Category – component category

    ```
    Intent i = new Intent(action, uri);
    ```

  - Extras – information to be sent

INTENT TYPES

- Component name makes an intent **explicit**, meaning that the intent should be delivered only to the app component defined by the component name
  - Open the Settings page page

```
Intent i = new Intent(this, SettingsActivity.class);
startActivity(i);
```

- Without a component name, the intent is **implicit** and the system decides which component should receive the intent based on the other intent information (such as the action, data).
  - Selecting a photo
    - Can be given option of Camera and one or multiple Photo Gallery
    - Clicking Share opens a chooser with a list of apps

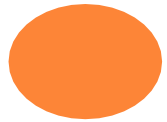
```
Intent i = new Intent(action, uri);
```

COMPONENT NAME: How to declare or use

- To start a specific component in app, specify the component name.
- ComponentName object is specified using a fully qualified class name of the target component, including the package name of the app.
  - edu.ewubd.EventInformationActivity.class

```
Intent i = new Intent(this, edu.ewubd.EventInformationActivity.class);
startActivity(i);
```

- Set the component name with
  - setComponent():
    ```
    i.setComponent(new ComponentName("edu.ewubd","edu.ewubd.EventInformationActivity"));
    ```
  - setClass():
    ```
    i.setClass(this, ewu.ewubd.EventInformationActivity.class);
    ```
  - setClassName()
    ```
    i.setClassName(this, "edu.ewubd.EventInformationActivity");
    i.setClassName("edu.ewubd", "edu.ewubd.EventInformationActivity");
    ```
  - with the Intent constructor
    - new Intent(this, edu.ewubd.EventInformationActivity.class);



ACTION

- A string that specifies the generic action to perform (such as *view* or *pick*).
- Action determines how the rest of the intent is structured—ie; what is contained in the data and extras.
- Common actions for starting an activity:
  - ACTION_VIEW some information that an activity can show to the user, such as a photo to view in a gallery app, or an address to view in a map app, or a website link to view in a browser app.

Show a web page

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

  - ACTIOND_DIAL user can call through another app, such as a calling app – the default caller or a VOIP app or whatsapp or imo.

Dial a phone number

```
Uri uri = Uri.parse("tel:8005551234");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
startActivity(it);
```

  - ACTIOND_SEND ??

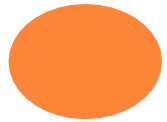
ACTION: How to declare or use

- Specify the action for an intent with
  - setAction()
  - with an Intent constructor.

- Can specify your own actions for use by intents within your app.

- To define your own actions, include application package name as a prefix
  - Why should we include application package name to define action in own app?

```
static final String ACTION_TIMETRAVEL="com.example.action.TIMETRAVEL";
```



DATA and DATA TYPE

- Action describes what is to be performed such as editing an item, viewing the content of the item and so on.

- The type of data supplied is generally dictated by the intent's action
  - URI of an image cannot be set for the action ACTION_DIAL
  - If the action is ACTION_EDIT, the data should contain the URI of the document to edit.

- Data references the information to be acted on
  - and/or the MIME type of that data
- Data is specified as Uri object.

Show a web page

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

MIME: How to declare or use

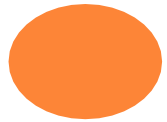
- MIME is something like an URL on the Internet.
- MIME types like
  - **text/html** for web pages
  - **image/jpeg** for .jpg images
- To set only the data URI, call setData().

```
intent.setData(
    Uri.parse("http://www.google.com"));
```

- To set only the MIME type, call setType().

- To set both the URI and MIME type, **do not** call setData() and setType()
  - because they each nullify the value of the other.

- Use setDataAndType() to set both URI and MIME type.



EXTRAS: carries primitive data to next component

- Key-value pairs that carry additional information required to accomplish the requested action.
- Add extra data with various putExtra() methods, each accepting two parameters: the key name and the value.
  - putExtra(String name, int value)
    ⇒ intent.putExtra("level", 406);
  - putExtra(String name, String[] value)
    ⇒ String[] foodList = {"Rice", "Beans", "Fruit"};
    intent.putExtra("food", foodList);

- Also create a Bundle object with all the extra data, then insert the Bundle in the Intent with putExtras().

```
Bundle b = new Bundle();
b.putString("key1", "value1");
intent.putExtras(b);
```

putExtras(bundle);
⇒ if lots of data, first create a bundle and pass the bundle.

INTENT TYPES

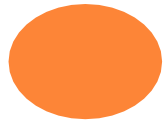
- There are two basic kinds of intents in Android:
  - ***Explicit intents***
  - ***Implicit intents***

- Explicit intents are used for communication between components of a single application.
  - Can we use Explicit intents for different applications? Why? Explain with a real life example.

- Implicit intents enable interoperability between different applications.
  - Can we use Implicit intents for a single application? Why? How? Explain with a real life example.

Explicit Intents: When/where to use

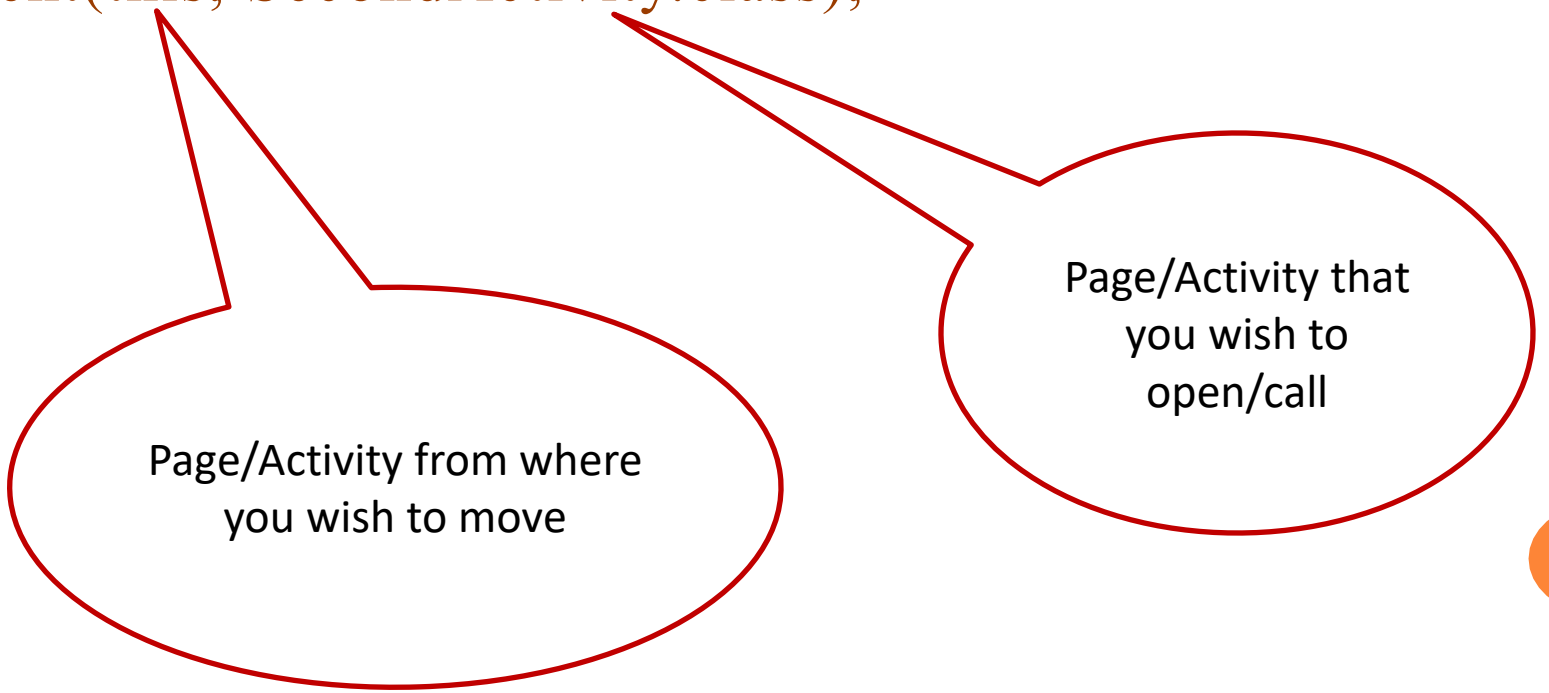
- Explicit intents –used to launch a specific app component, such as a particular activity or service in your app.
- Explicit intents require that specific named class to implement the desired action.
- Class structure of an application is usually not known outside the application, so explicit intents are used for actions that occur **within a single application**



EXPLICIT INTENT: How to use or declare

- To create an explicit intent, define the component name for the Intent object—all other intent properties are optional.

```
Intent i = new Intent(this, SecondActivity.class);
startActivity(i);
```



Explicit Intents: Passing/Receiving information

- Explicit Intents to launch a new *Activity* (associated with the class *Webscreen*)

```
public void onClick(View v) {
        Intent j = new Intent(this, Webscreen.class);
        j.putExtra("Web_URL", "http://eagle.phys.utk.edu/guidry/recipes/mojito.html");
        startActivity(j);
}
```

```
Bundle b = new Bundle();
b.putString("key1", "value1");
b.putString("key2", "value2");
intent.putExtras(b);
```

- Data passed to the new *Activity* using the *putExtra()* method

- Receiving information from the previous page/activity

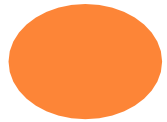
```
public void onCreate(….) {
        Intent i = this.getIntent();
        String urlString = i.getStringExtra("Web_URL");
}
```

Implicit Intents

- **Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another application to handle it.

- For example - To show user a location on a map, use an implicit intent to request another capable application to show a specified location on a map.

- When implicit intent called , the Android system finds the appropriate component to start by comparing the contents of the *intent* to the *intent filters* declared in the manifest file of other apps on the device.



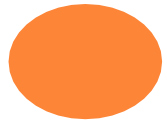
Example - implicit intent

Example 1:

```
String url = "http://www.vogella.com";
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(url));
startActivity(i);
```

Example 2:

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



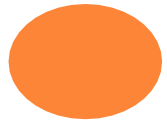
EXAMPLE IMPLICIT INTENT

Example 3:

```
Intent i = new Intent();
i.setAction(Intent.ACTION_SEND);
i.putExtra(Intent.EXTRA_TEXT, textMessage);
i.setType(HTTP.PLAIN_TEXT_TYPE);
```

Example 4:

```
Intent i= new Intent (Intent.ACTION_DIAL, Uri.parse(—tel+65789999‖))
```



IntentFilter

- Intent Filter defines –
  - How your activity can be invoked by another activity.

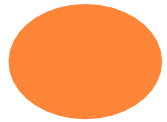
- An intent filter is an expression in an app's manifest file that specifies the type of intents that the component would like to receive.

- From other activities to invoke your activity,
  - specify the action and category within the **<intent-filter>** element in the Manifest.xml file

Implicit Intents: Component Selection

- If the intent matches an intent filter, the system starts that component and delivers it the Intent object.

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



How to make an activity accessible from other apps

- By declaring an intent filter for an activity, it is possible for other apps to start your activity with a certain kind of intent.

- If intent filters are not declared for an activity, then it can be started only with an explicit intent.

```
<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>
```

How does IntentFilter work?

- Each intent filter specifies the type of intents it accepts based on the intent's action, data, type, and category.
- The system will deliver an implicit intent to your app component only if the intent can pass through one of your intent filters.
- An explicit intent is always delivered to its target, regardless of any intent filters the component declares
  - However, it can be restricted by setting “export” to “false”

Elements of IntentFilter

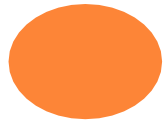
- Each intent filter is defined by an <intent-filter> element in the app's manifest file.

- Inside the <intent-filter>, specify the type of intents to accept using one or more of these three elements:

- <action>Declares the intent action accepted, in the name attribute. Value: literal string value of an action.

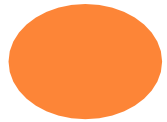
- <category>Declares the intent category accepted, in the name attribute. Value: literal string value of an action.

- <data>Declares the type of data accepted, using one or more attributes that specify various aspects of the data URI (scheme, host, port, path, etc.) and MIME type.



INTENT FILTER for Components

- An application component should declare separate filters for each unique job it can do.
- For example, one activity in an image gallery app may have two filters: one filter to view an image, and another filter to edit an image.
- When the activity starts, it inspects the Intent and decides how to behave based on the information in the Intent (such as to show the editor controls or not).



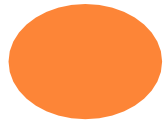
IntentFilter

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.lightcone.sharingintents"
   android:versionCode="1" android:versionName="1.0"
    >
   <uses-sdk android:minSdkVersion="7" />

   <application android:icon="@drawable/ic_launcher" android:label="@string/app_name" >
      <activity android:name=".SharingIntents" android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity android:name=".MyLittleBrowser" android:label="@string/little_browser_name">
         <intent-filter>
           <action android:name="android.intent.action.VIEW" />
           <category android:name="android.intent.category.DEFAULT" />
           <data android:scheme="http"/>
         </intent-filter>
      </activity>
   </application>

</manifest>
```

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.

- Common categories:
  - CATEGORY_BROWSABLE - target activity allows itself to be started by a web browser to display data referenced by a link—such as an image or an e-mail.
  - CATEGORY_LAUNCHER - activity is the initial activity of a task and is listed in the system's application launcher.
- You can specify a category with addCategory().



CATEGORY for IMPLICIT INTENT

- In order to receive implicit intents, **include** the CATEGORY_DEFAULT category in the intent filter.
  - If you do not declare this category in your intent filter, no implicit intents will resolve to your activity.

- The methods startActivity() and startActivityForResult() treat all intents as if they declared the CATEGORY_DEFAULT category.

Example for startActivityForesult(...)

// @First Activity

```
1.      button1.setOnClickListener(new OnClickListener() {
2.         @Override
3.         public void onClick(View arg0) {
4.            Intent intent=new Intent(MainActivity.this, SecondActivity.class);
5.            startActivityForResult(intent, 2);// Activity is started with requestCode 2
6.         }
7.      });


8.     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
9.            super.onActivityResult(requestCode, resultCode, data);
10.           // check if the request code is same as what is passed  here it is 2
11.           if(requestCode==2) {
12.                  String message=data.getStringExtra("MESSAGE");
13.                   textView1.setText(message);
14.           }
15.    }
```

// @Second Activity

```
1.         button1.setOnClickListener(new OnClickListener() {
2.            @Override
3.            public void onClick(View arg0) {
4.               String message=editText1.getText().toString();
5.               Intent intent=new Intent();
6.               intent.putExtra("MESSAGE",message);
7.               setResult(2,intent);
8.               finish();//finishing activity
9.            }
10.         });
```

Thank You