Activities and Intents

ACTIVITY

- A window that contains the user interface of the application
- Applications have one or more activities
- Main purpose of an activity is to interact with the user
- **Activity's life Cycle-** From the moment the activity appears on the screen to the moment it is hidden, it goes through a number of stages
- Understand Activity life cycle- to ensure app works correctly

Create Activity

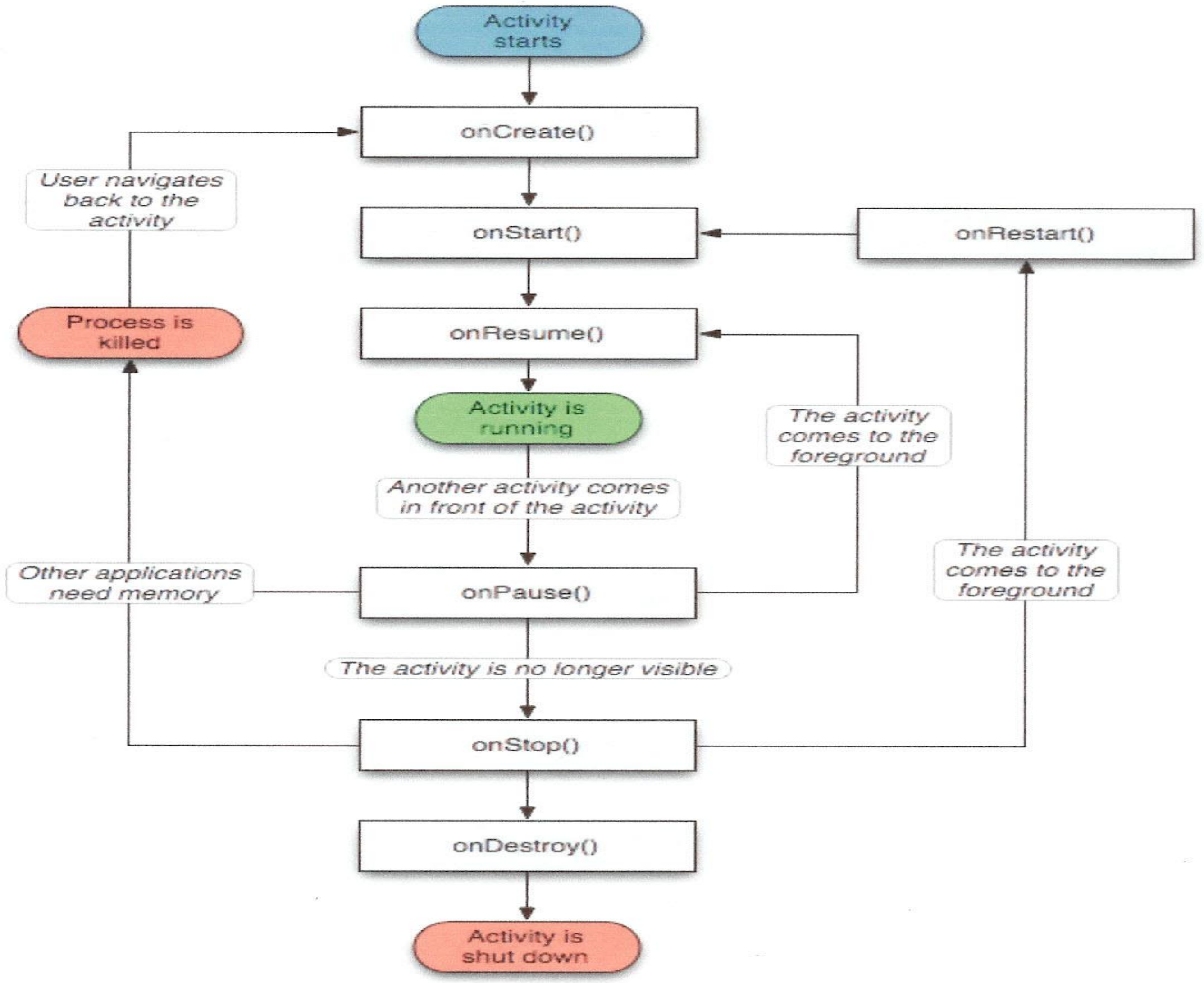
- To create an activity, create a java class that extends the Activity base class

- Activity class loads its UI component using the XML file defined in *res/layout* folder

  setContentView(R.layout.main);

- Every Activity in the application must be declared in your *AndroidManifest.xml* file

Activity Life Cycle (Cntd…)


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Activity Life Cycle

- Activity base class define a series of events that govern the life cycle of an activity

- On Create()
  - Called when the activity is first created
  - By default, the activity created contains the onCreate() event.
  - Within onCreate() event handler write the code to display the UI elements of screen.
  - Use onCreate() method to create and instantiate the objects to be used in the application

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 the onStart() and onResume() methods are called 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 services or code that needs to run when your activity is in the foreground.

Activity Life Cycle (cntd…)

- onPause()
  - Called when the activity is being paused and the previous activity is being resumed.
  - Called in two scenarios-
    - when activity sent to back ground
    - when the activity is killed when the user presses back button
  - Use the onPause() method to stop any services or code that does not need to run 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 **onResume'd** or **onStop'ped** 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

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

- In Android - navigation between activities is 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.

Building an Intent

- An Intent object carries information that the Android system uses to determine
  - which component to start (component name or component category that should receive the intent)
  - information that recipient component uses in order to perform the action (such as the action to take and the data to act upon).

Building an Intent

- The primary information contained in an Intent are the following:
  - Component name
  - Action
  - Data
  - Category
  - Extras

Component name

- Component name (optional)
- Component name makes an intent **explicit**, meaning that the intent should be delivered only to the app component defined by the component name.
- 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).

Component name

- To start a specific component in app, specify the component name.

- ComponentName object- specify using a fully qualified class name of the target component, including the package name of the app.

→ com.example.ExampleActivity

- Set the component name with
  - setComponent()
  - setClass()
  - setClassName()
  - with the Intent constructor.

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.
  - ACTION_SEND user can share through another app, such as an email app or social sharing app.

ACTION

- 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
- static final String ACTION_TIMETRAVEL =

  "com.example.action.TIMETRAVEL";

Data

- Äction 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.
- For example, if the action is ACTION_EDIT, the data should contain the URI of the document to edit.
- Data is specified as Uri object.
- Uri object - references the data to be acted on and/or the MIME type of that data.

Data

- 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().
- 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.

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 message.
  - 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().

Extras

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

Extras

- The Intent class specifies many EXTRA_* constants for standardized data types.
- To declare your own extra keys - include your app's package name as a prefix.
  - static final String EXTRA_GIGAWATTS = "com.example.EXTRA_GIGAWATTS";

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.

- Implicit intents enable interoperability between different applications.

Explicit Intents

- 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 not known outside the application, so explicit intents are used for actions that occur **within a single application**.

Example explicit intent

- 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

```
public void onClick(View v) {
    switch(v.getId()){
      case R.id.button1:
        Intent j = new Intent(this, Webscreen.class);

        j.putExtra(Web_URL,
          "http://eagle.phys.utk.edu/guidry/recipes/mojito.html");

        startActivity(j);
      break;
   }
}
```

- Explicit Intents to launch a new *Activity* (associated with the class *Webscreen*)
- Data passed to the new *Activity* using the *putExtra()* method.

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

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

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

Example implicit intent

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

Intent i= new Intent
(android.content.Intent.ACTION_DIAL,
        Uri.parse("tel+65789999"))
```

Implicit Intents

- 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.
- The determination of Android by which components can handle a given request issued through an implicit intent is implemented through an *IntentFilter*.

Implicit Intents

- 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.
- By declaring an intent filter for an activity, it possible for other apps to directly 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.

Implicit Intents

- To ensure application security, always use an explicit intent when starting a Service and do not declare intent filters for services.

- Using an implicit intent to start a service is a security hazard because it cannot be certain what service will respond to the intent, and the user cannot see which service starts.

Implicit Intent

- In order to receive implicit intents, **include** the CATEGORY_DEFAULT category in the intent filter.
- Themethods startActivity() and startActivityForResult() treat all intents as if they declared the CATEGORY_DEFAULT category.
- If you do not declare this category in your intent filter, no implicit intents will resolve to your activity.

IntentFilter

- Intent Filter defines how your activity can be invoked by another activity.
- An *IntentFilter* specifies the types of intents that an activity, service, or broadcast receiver can respond to.
- IntentFilters are defined in the *AndroidManifest.xml* file.
- For other activities to invoke your activity, specify the action and category within the <intent-filter> element in the Manifest.xml file
- <intent-filter> element nested in the app component (such as an <activity> element).

IntentFilter

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

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, not the class constant.
- <category>Declares the intent category accepted, in the name attribute. Value : literal string value of an action, not the class constant.
- <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

- 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).

Intent filter -- (manifest file)

```
<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>
```

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>
```

Thank You