SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

COURSE NAME :19CS603- MOBILE APPLICATION DEVELOPMENT

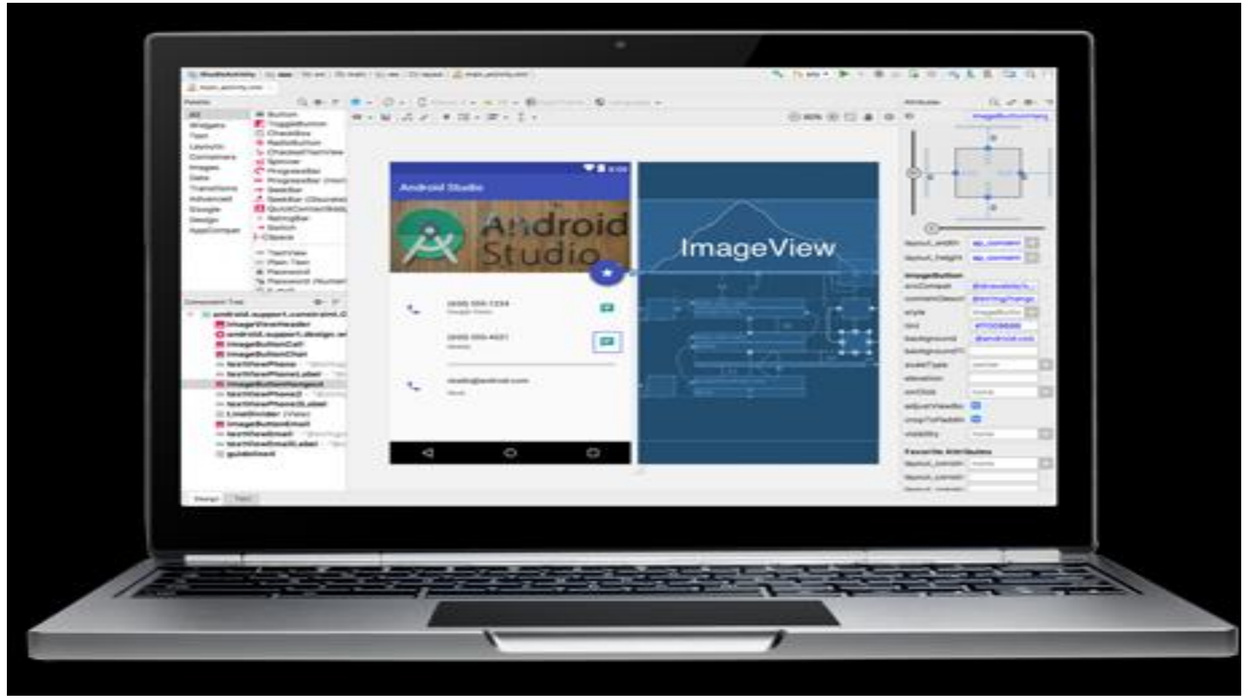
III YEAR /VI SEMESTER

Unit 4- Introduction to I-Android

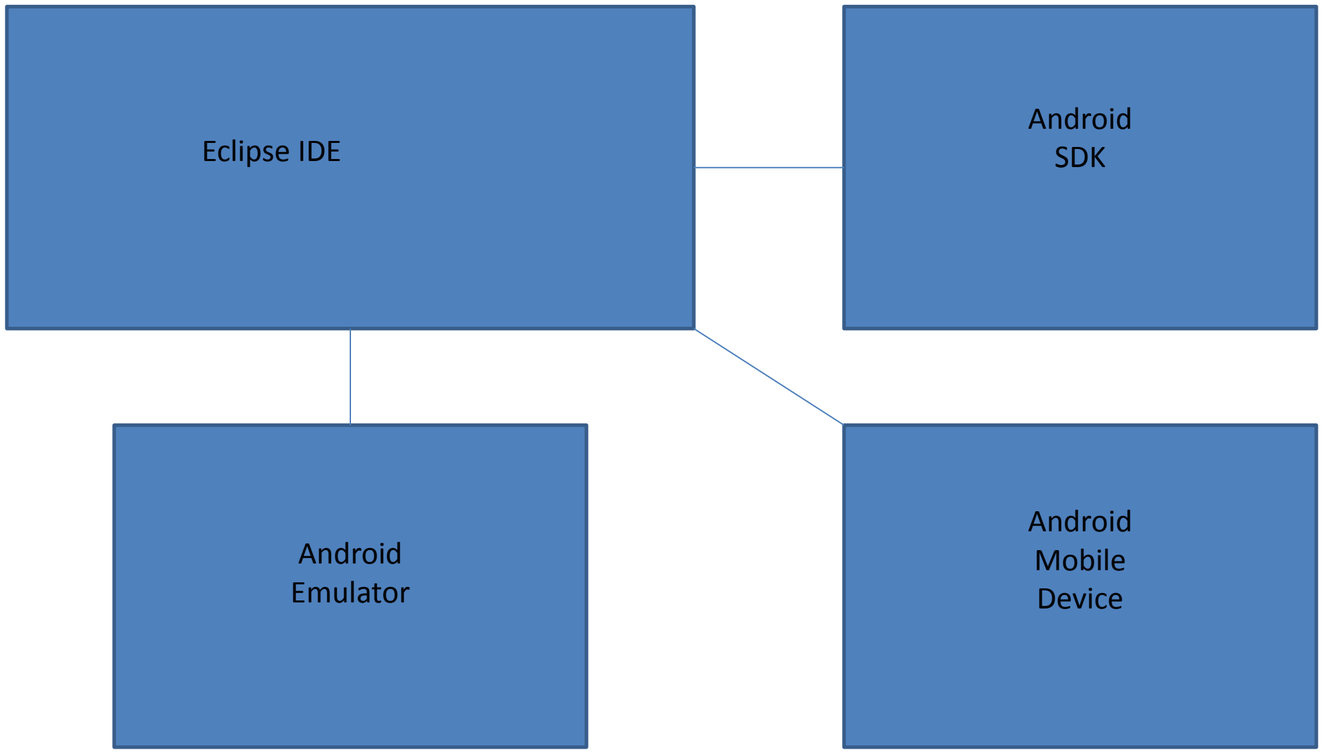
Topic : Android Architecture

Android Studio

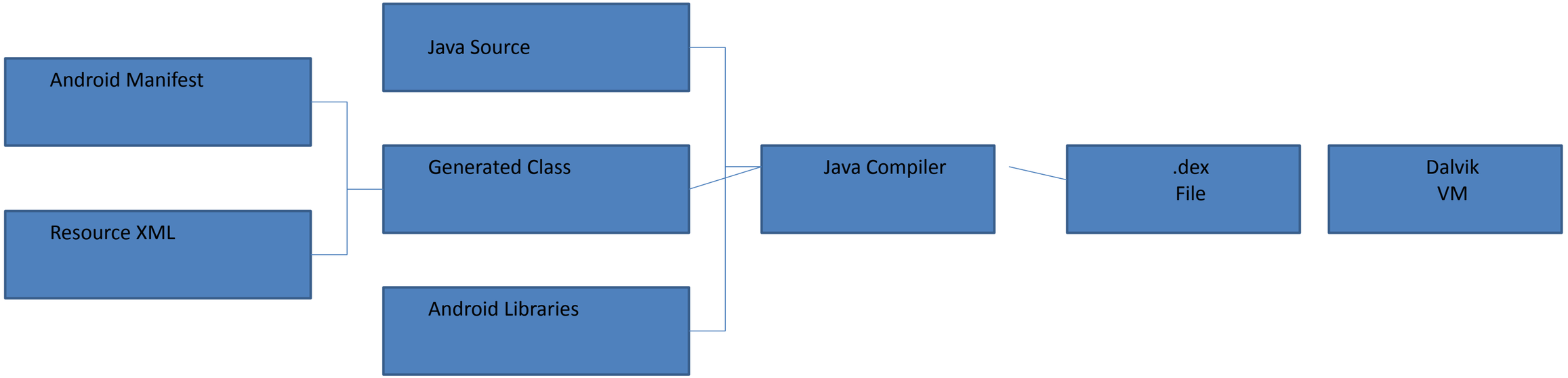
https://developer.android.com/studio/index.html



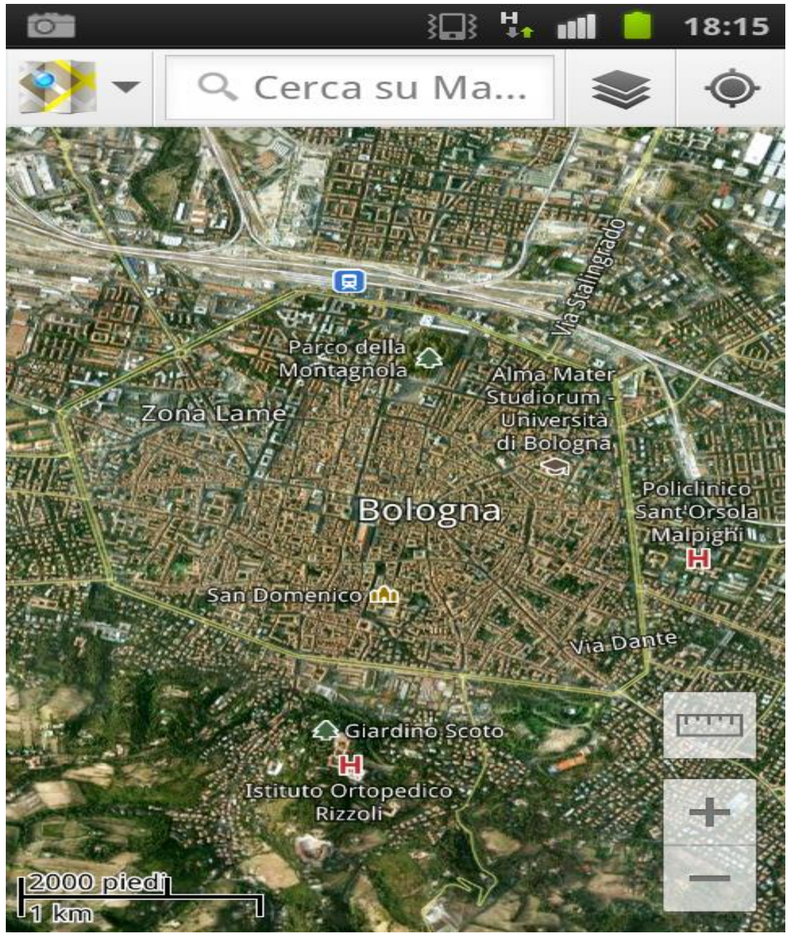
Android Application Development



Android development



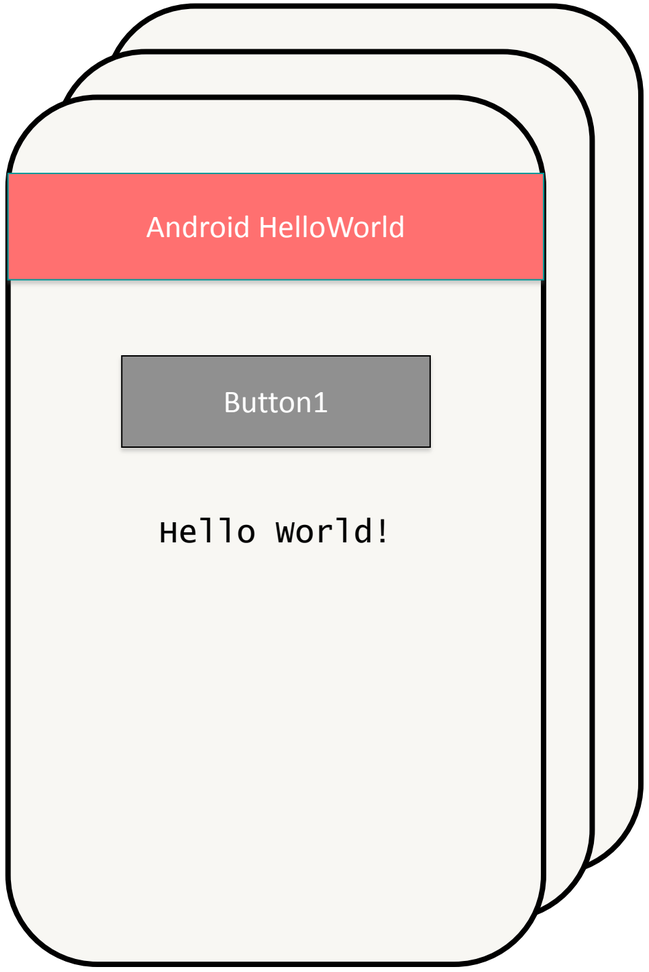
Android Applications Design



APPLICATION COMPONENTS

- **Activities**
- **Intents**
- **Services**
- **Content Providers**
- **Broadcast Receivers**

Android Components: Activities



- An Activity corresponds to a single screen of the Application.
- An Application can be composed of *multiples screens* (Activities).
- The Home Activity is shown when the user launches an application.
- Different activities can exhange information one with each other.

Android Components: Activities

- **Each activity is composed by a list of *graphics components*.**
- **Some of these components (also called Views) can interact with the user by handling events (e.g. Buttons).**
- **<u>Two ways</u> to build the graphic interface:**

PROGRAMMATIC APPROACH

MainActivity.java

Example:

```
Button button=new Button (this);
TextView text= new TextView();
text.setText("Hello world");
```

Android Components: Activities

- Each activity is composed by a list of *graphics components*.
- Some of these components (also called Views) can interact with the user by handling events (e.g. Buttons).
- Two ways to build the graphic interface:

DECLARATIVE APPROACH

activity_main.xml

Example:

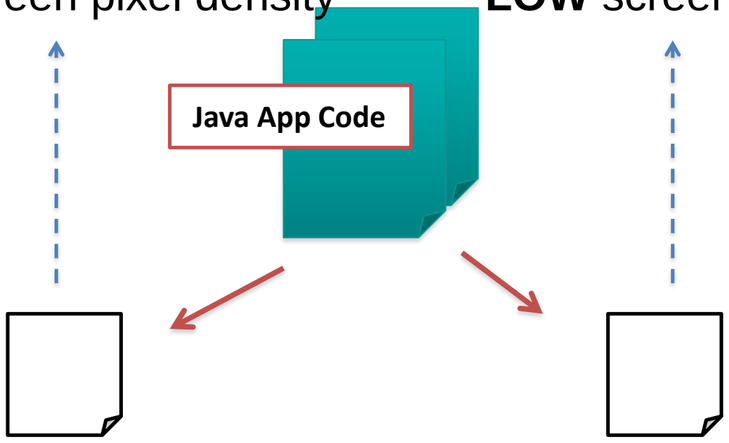
```
< TextView android.text=@string/hello”
android:textcolor=@color/blue android:layout_width=“fill_parent”
android:layout_height=“wrap_content” />
< Button android.id=“@+id/Button01”
android:textcolor=“@color/blue” android:layout_width=“fill_parent”
android:layout_height=“wrap_content” />
```

Android Components: Activities

EXAMPLE

Device 1
HIGH screen pixel density

Device 2
LOW screen pixel density



XML Layout File
Device 1

XML Layout File
Device 2

-Build the application layout through XML files (like HTML)

-Define two different XML layouts for two different devices

-At runtime, Android detects the current device configuration and loads the appropriate resources for the application

-No need to recompile!

-Just add a new XML file if you need to support a new device

Android Components: Activities

➢ ***Android applications typically use both the approaches*!**

DECLARATIVE APPROACH

⇩

XML Code

⇨

Define the Application **layouts** and **resources** used by the Application (e.g. labels).

PROGRAMMATIC APPROACH

⇩

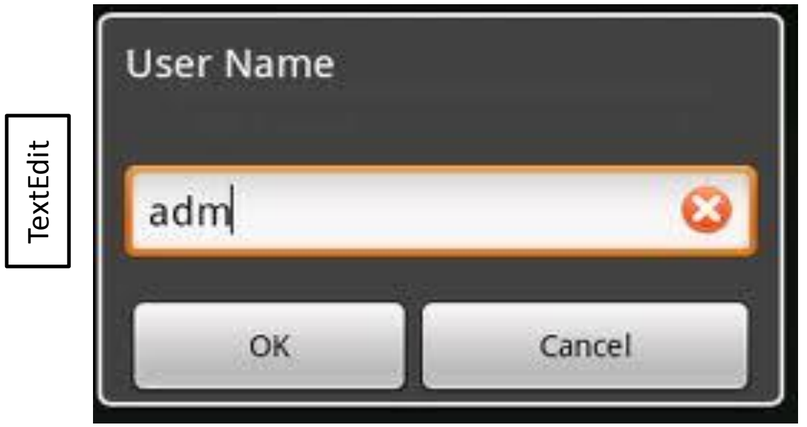
Java Code

⇨

Manages the **events**, and handles the **interaction** with the user.

Android Components: Activities

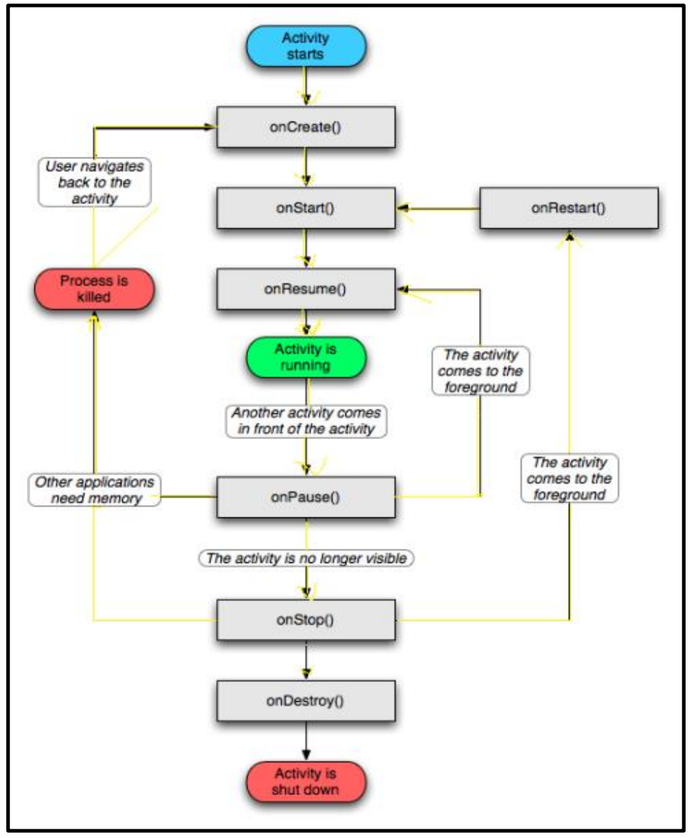
- Views can generate events (caused by human interactions) that must be managed by the Android-developer.



Example

```
public void onClick(View arg0) {
    if (arg0 == Button) {
        // Manage Button events
    }
}
```

Android Components: Activities



- The Activity Manager is responsible for creating, destroying, managing activities.
- Activities can be on different states: starting, running, stopped, destroyed, paused.
- Only one activity can be on the running state at a time.
- Activities are organized on a stack, and have an event-driven life cycle (details later …)

Android Components: Activities

- Main difference between Android-programming and Java (Oracle) -programming:
  - Mobile devices have constrained resource capabilities!

- Activity lifetime depends on users' choice (i.e. change of visibility) as well as on system constraints (i.e. memory shortage).

- Developer must implement lifecycle methods to account for state changes of each Activity …

Android Components: Activities

```
public class MyApp extends Activity {

    public void onCreate() { ... }
    public void onPause()  { ... }
    public void onStop()   { ... }
    public void onDestroy(){ ... }
    ….
}
```

Called when the Activity is **created** the first time.

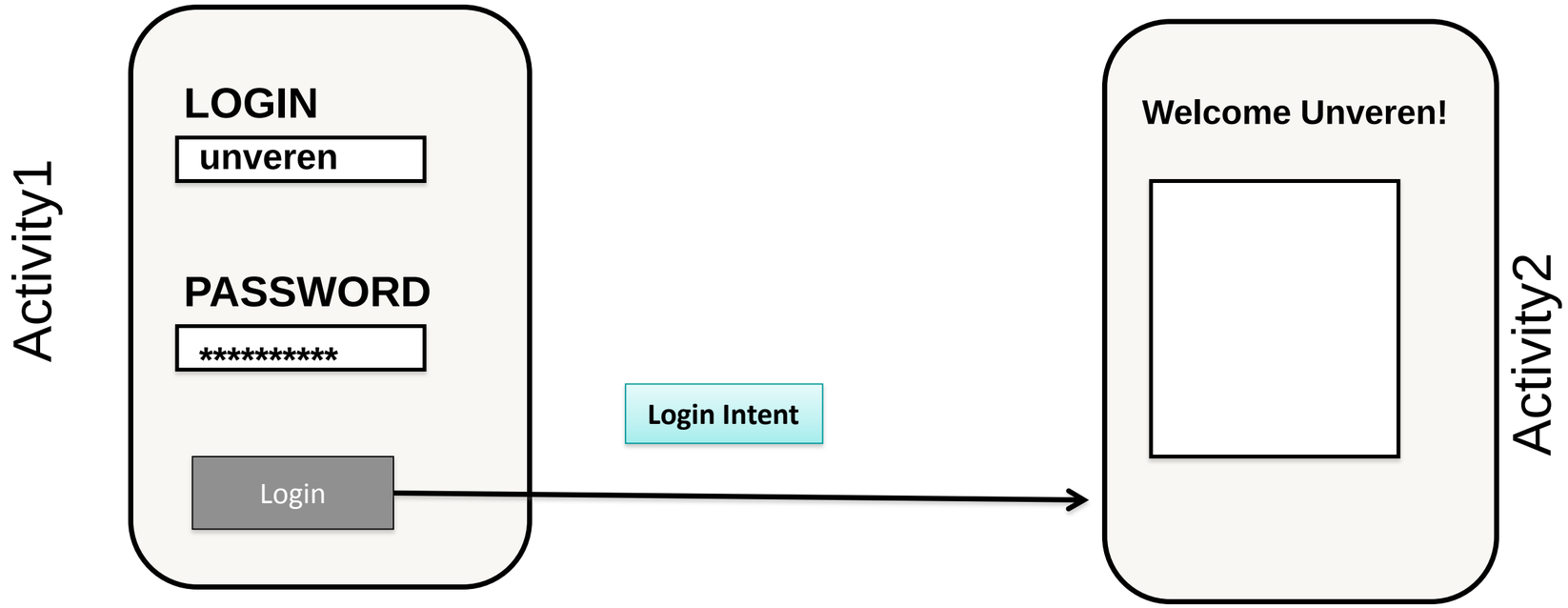
Called when the Activity is **partially visible**.

Called when the Activity is **no longer visible**.

Called when the Activity is **dismissed**.

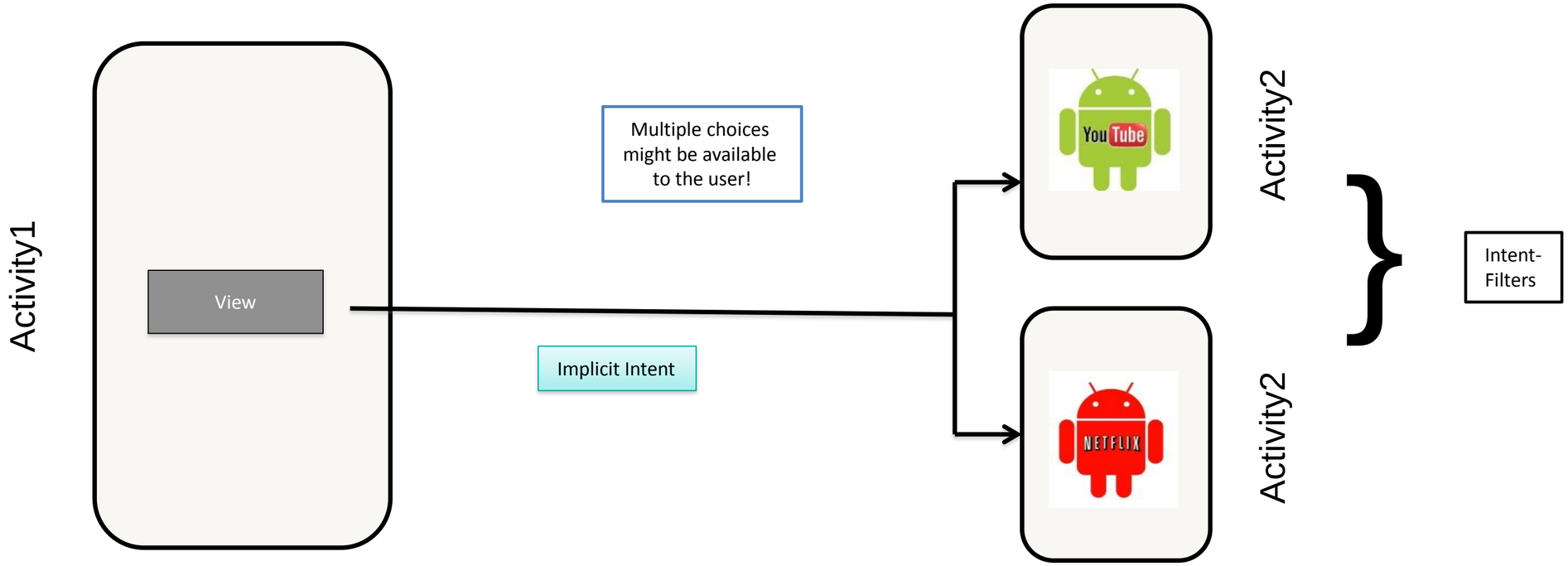
Android Components: Intents

- Intents: asynchronous messages to activate core Android components (e.g. Activities).
- Explicit Intent → The component *(e.g. Activity1)* specifies the destination of the intent *(e.g. Activity 2)*.



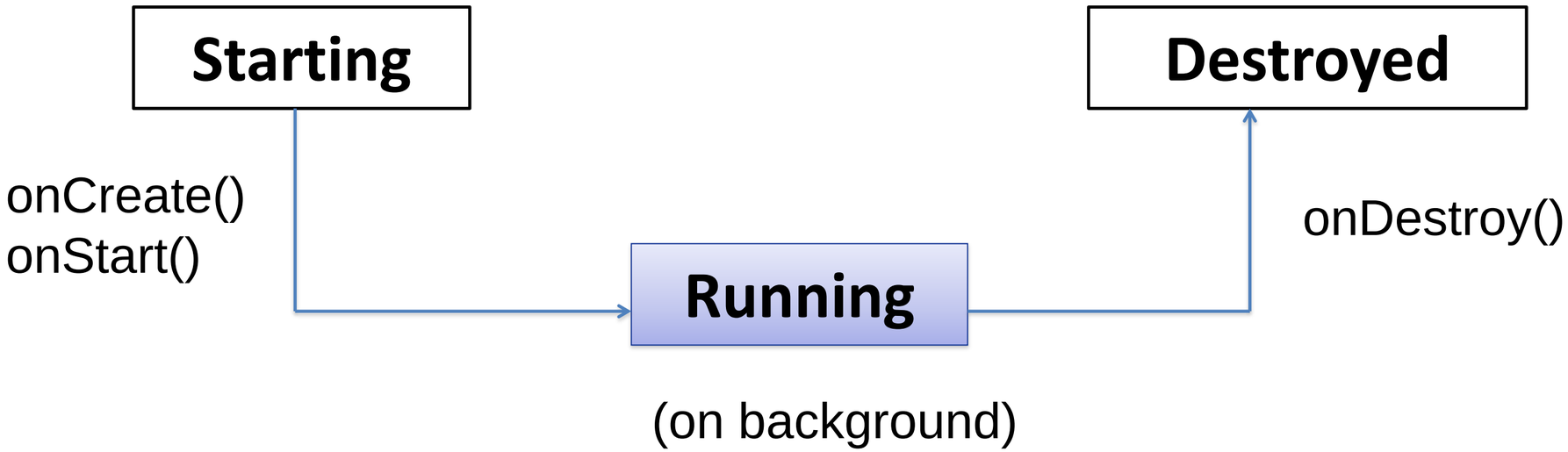
Android Components: Intents

- Intents: asynchronous messages to activate core Android components (e.g. Activities).
- Implicit Intent → The component *(e.g. Activity1)* specifies the type of the intent *(e.g. "View a video")*.



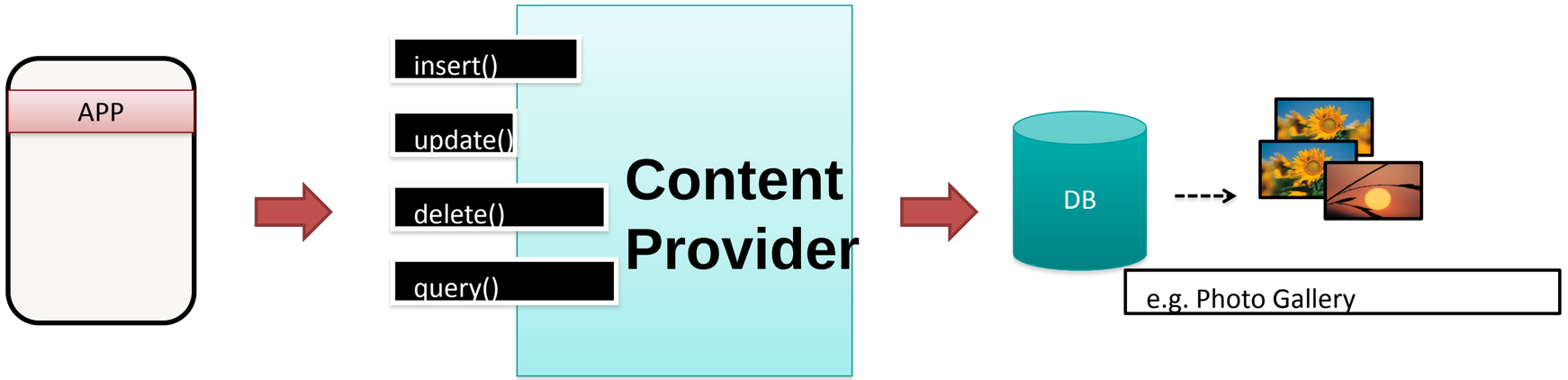
Android Components: Services

- Services: like Activities, but run in background and do not provide an user interface.
- Used for non-interactive tasks (e.g. networking).
- Service life-time composed of 3 states:

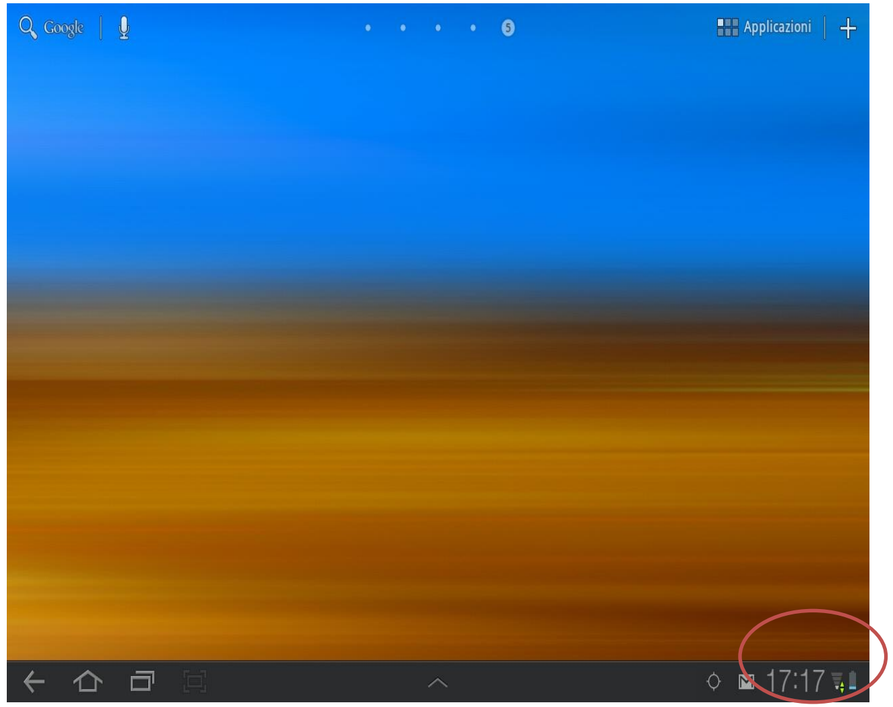


Android Components: Content Providers

- Each Android application has its own private set of data (managed through *files* or through *SQLite* database).
- Content Providers: Standard interface to *access and share data among different applications*.



Android Components: Broadcast Receivers



- *Publish/Subscribe* paradigm
- Broadcast Receivers: An application can be signaled of external events.
- Notification types: Call incoming, SMS delivery, Wifi network detected, etc

Android Components: Broadcast Receivers

BROADCAST RECEIVER example

```
class WifiReceiver extends BroadcastReceiver {
        public void onReceive(Context c, Intent intent) {
            String s = new StringBuilder();
            wifiList = mainWifi.getScanResults();
            for(int i = 0; i < wifiList.size(); i++){
                s.append(new Integer(i+1).toString() + ".");
                s.append((wifiList.get(i)).toString());
                s.append("\\n");
            }
            mainText.setText(sb);
        }
    }
```

Android Components: Broadcast Receivers

BROADCAST RECEIVER example

```
public class WifiTester extends Activity {
        WifiManager mainWifi;
        WifiReceiver receiverWifi;
        List<ScanResult> wifiList;
         public void onCreate(Bundle savedInstanceState) {
            ...
            mainWifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
            receiverWifi = new WifiReceiver();
            registerReceiver(receiverWifi, new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
            mainWifi.startScan();
}
```

Android Components: System API

- Using the **components** described so far, Android applications can then leverage the system API …

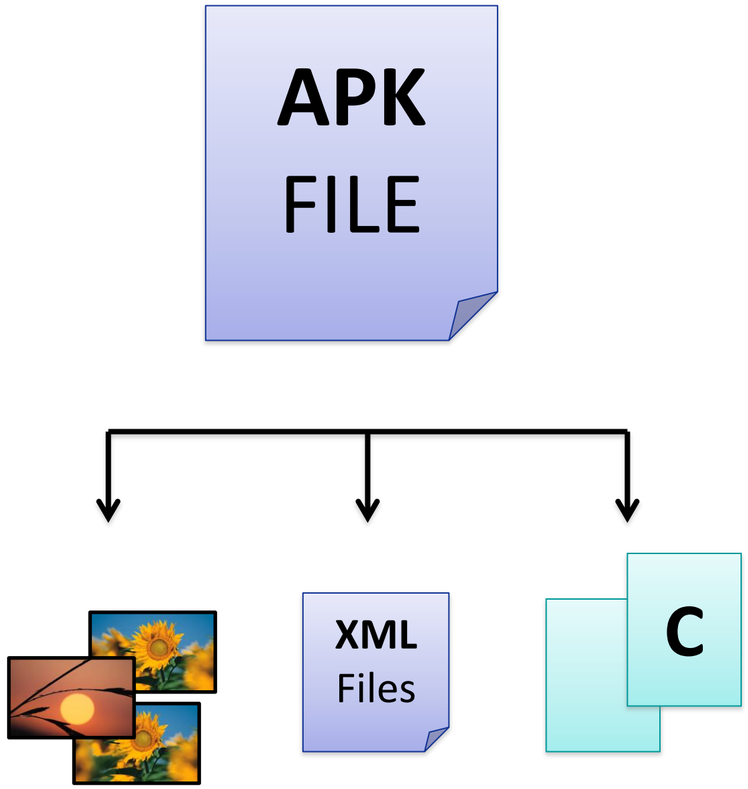
SOME EXAMPLEs …

- *Telephon Manager* data access (call, SMS, etc)
- *Sensor* management (GPS, accelerometer, etc)
- Network *connectivity* (Wifi, bluetooth, NFC, etc)
- *Web* surfing (HTTP client, WebView, etc)
- *Storage* management (files, SQLite db, etc)
- ….

Android Components: Google API

- … or easily interface with other **Google services**:

Distribution



Each Android application is contained on a single APK file.

- Java **Byte-code** (*compiled for Dalvik JVM*)
- **Resources** (e.g. images. videos, XML layout files)
- **Libraries** (optimal native C/C++ code)

Android Application Security

- **Android applications run with a distinct system identity (Linux user ID and group ID), in an isolated way.**
- **Applications must explicitly share resources and data. They do this by declaring the *permissions* they need for additional capabilities.**
  - Applications statically **declare** the permissions they require.
  - User must **give his/her consensus** during the installation.

ANDROIDMANIFEST.XML

```
<uses-permission android:name="android.permission.IACCESS_FINE_LOCATION" />

<uses-permission android:name="android.permission.INTERNET" />
```