

EMBEDDED SYSTEMS PROGRAMMING 2014-15

Application Basics

APPLICATIONS

- Application components (e.g., UI elements) are **objects** instantiated from the platform's frameworks
- Applications are **event driven**
(\Rightarrow there are callbacks, delegates, notifications...)
- Many use **design patterns**
(e.g., delegation, model-view-controller, ...)

APPLICATIONS AND IDES

Integrated Development Environments (IDEs) simplify application development by

- automatically setting up a default application environment,
- automatically creating skeleton code every time a new application component is added,
- allowing the programmer to design the UI in a graphical way

Nonetheless, a lot of details must be managed by hand

APPLICATION RESOURCES

- **Resources:** non-code files that are part of an application
 - Images, sounds, videos
 - The application icon
 - The application preferences
 - And more

EMBEDDED APPS (1/2)

Mobile apps behave differently from PC applications

- **Only one app is visible at a time to interact with the user. The operating system manages the switch from one application to another**
- **The visible app has no windows**
- **Apps not interacting with the user are scheduled differently**

EMBEDDED APPS (2/2)

- The operating system may decide to **terminate apps not interacting with the user** so as to free up resources
- The **lifecycle of apps is optimized for the embedded setting**
- **No page file**

ANDROID: USER VS. SYSTEM

- **System apps do not use special APIs:** they have to go through the same public APIs available to user-developed applications
- Android can be told to make **your application replace a standard application**. For instance, a user-programmed keyboard may replace the system-provided keyboard
- This open, modular design is unique to Android

ANDROID: ADDING USER APPS

- Adding applications is done through Android **APK (.apk) packages**.
An APK package is a binary archive containing compiled Dalvik code, non-compiled resources such as images, and other informations for installing the application
- Every APK should be digitally signed. However, it is not mandatory that the signature is validated by a CA

ANDROID: APP TYPES

- **Applications:** take over the whole screen.
Example: Android's standard web browser
- **Widgets:** take a small, fixed portion of the home screen. Example: Android's standard clock.
(Note: *UI widgets* are a different thing!)

APPLICATION ISOLATION

- Each application runs in its own Linux process with its own user ID.
The Linux OS ensures that applications cannot access privileged OS components or hamper one another's memory and data
- Each application runs in its own copy of the VM.
Malfunctions cannot propagate from one application to another
- Applications communicate through content providers

APP COMPONENTS

- **Activity:** a single screen with a user interface
- **Service:** performs (in the *background*) long-running operations (e.g., music playback). No user interface
- **Content provider:** encapsulates data that needs to be shared between applications. No user interface
- **Broadcast receiver:** responds to system-wide broadcast events (e.g., “battery low!”). No user interface
- Four classes are available, one for each type of component

ABOUT COMPONENTS

- An application is a mixture of components, each with a **distinct lifecycle**
- Any application can ask Android to start another application's component (e.g., the “take a shot” activity of the camera application)
- No “first” component, no application entry point (although a “main activity” exists)

INTENTS

- Activities, services and broadcast receivers are activated by intents
- Intent: asynchronous message that requests an action
- An intent may request a specific component or just a type of component: in the latter case, it is the system's task to bind the intent to an available component that performs the required action

ACTIVITIES VS. APPLICATIONS

- **Activity:** a single, focused thing that the user can do
- **Example:** an activity to edit a text-only note
- **Application:** contains activities (and other components)
- **Example:** a notepad application has an activity to edit a note and another activity to manage a list of notes

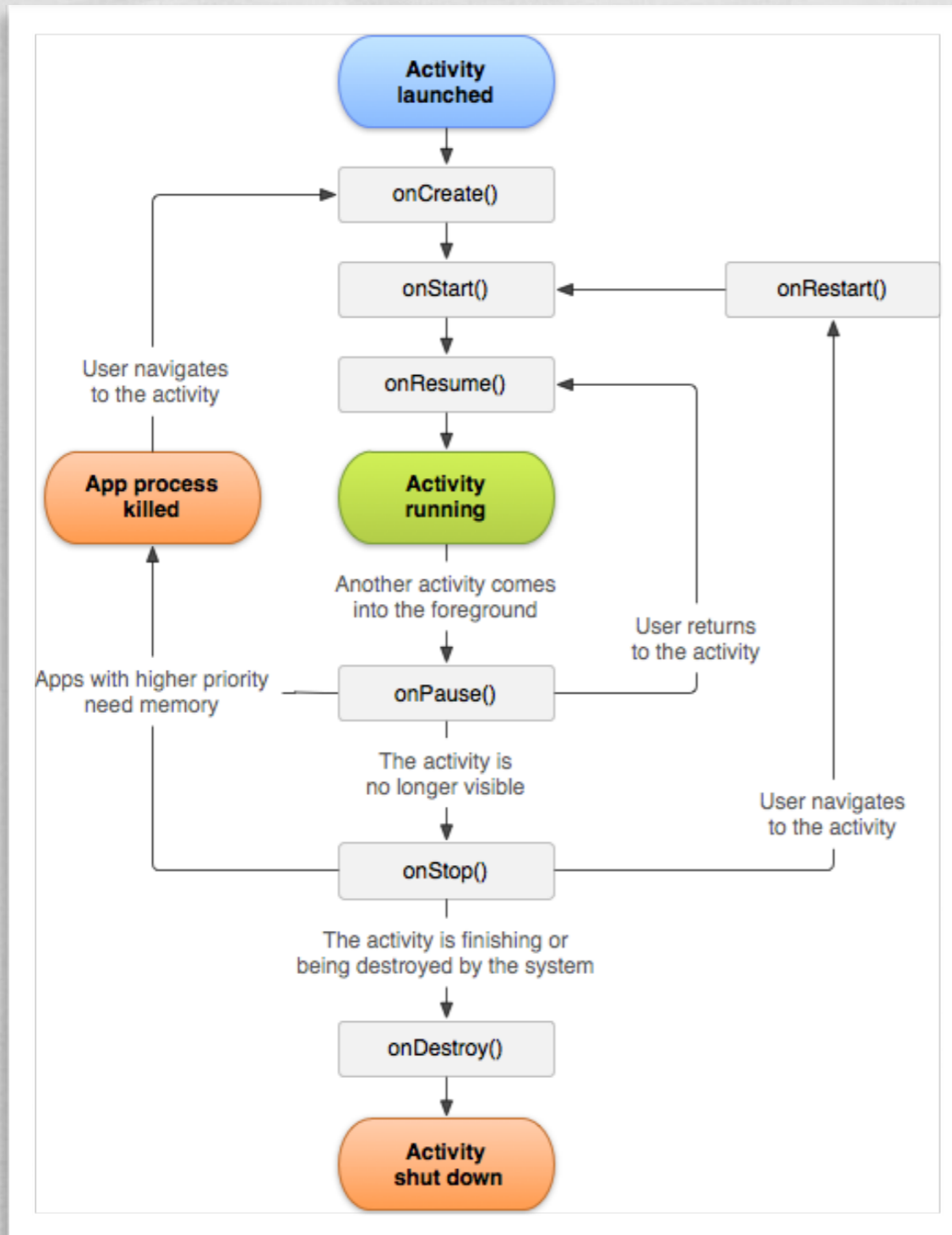
ACTIVITY LIFECYCLE (1/2)

An activity passes through the following states.

- **Active** (aka running): visible, receiving user input
- **Paused**: partially visible, not receiving user input (e.g., an alert dialog is above it)
- **Stopped**: not visible
- **Destroyed**: removed from memory by Android

Callback methods allow an activity to perform actions on state transitions

ACTIVITY LIFECYCLE (2/2)



State transitions are a consequence of **user or system** decisions.

An activity cannot avoid state transitions: it can only manage them by undertaking appropriate actions

CALLBACK METHODS

- **onCreate ()**
Called when the activity is first created
- **onStart ()**
Called when the activity is becoming visible to the user
- **onResume ()**
Called when the activity will start interacting with the user
- **onPause ()**
Called when the system is about to start resuming a previous activity
- **onStop ()**
Called when the activity is no longer visible to the user
- **onRestart ()**
Called after the activity has been stopped, prior to it being started again
- **onDestroy ()**
Last method to be called before the activity is destroyed

APPLICATION TERMINATION

- Any activity that is not running may be terminated by Android at any time to free up resources
- The `onStop()` and `onDestroy()` methods are not guaranteed to be called: only `onPause()` is

**Morale: save your state
every time your activity is paused**

ORIENTATION CHANGE

Any change between portrait and landscape mode will cause an activity to be

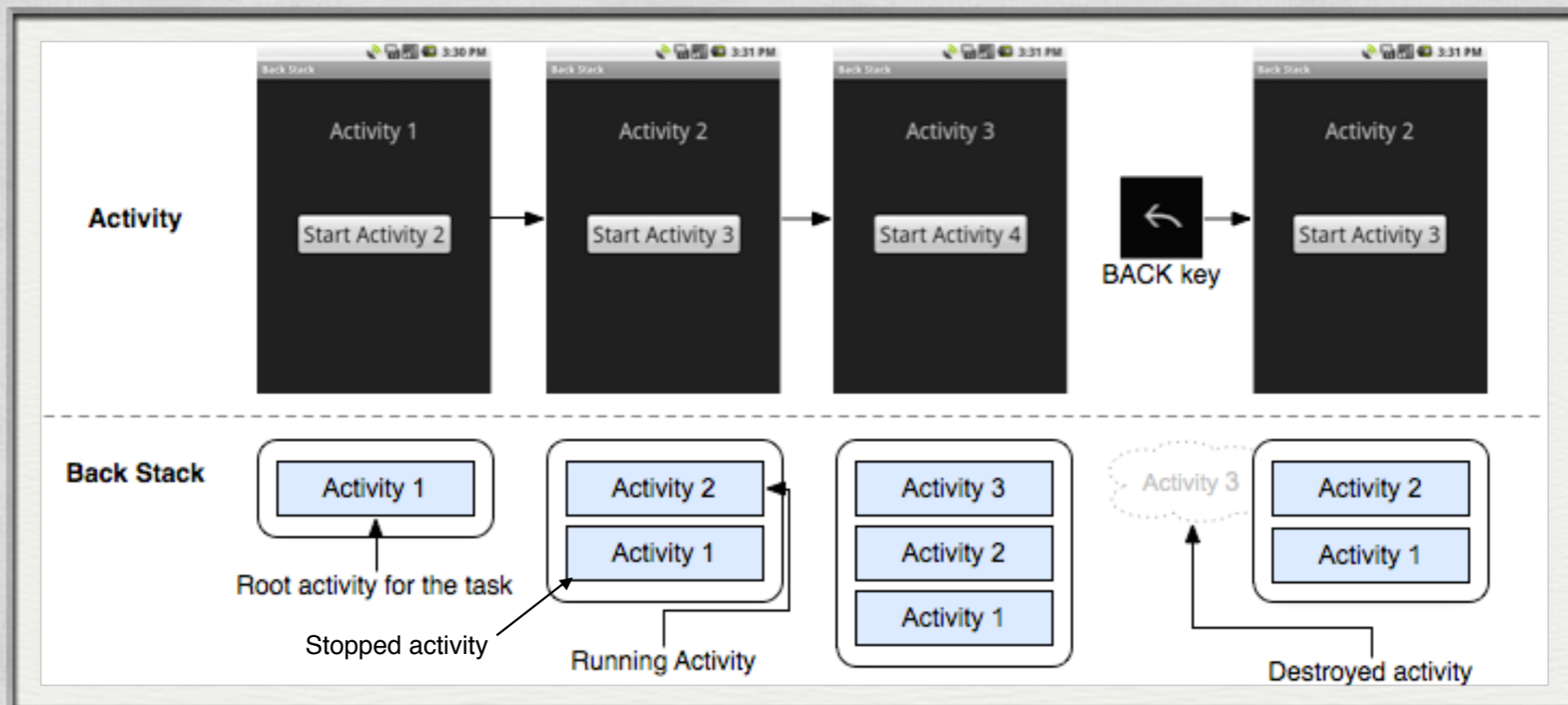
- **paused, then**
- **stopped, then**
- **destroyed,**

then a new instance of the activity will be created

**Once again: save your state
every time your activity is paused**

BACK STACK

- Android's Activity Manager arranges activities in a stack (the back stack) according to the order in which they were opened



TASKS

- **Task:** set of related activities
(i.e., the first activity in the task called all the others)
- A task is moved to the background as a whole -- and all its activities stopped -- if the user starts a new, unrelated task (e.g., by tapping the “Home” button and launching a new application)
- Different instances of the same activity can be present in different tasks

THE MANIFEST FILE

The manifest (`AndroidManifest.xml`) of an application

- declares the components of the app
- declares hardware and software features (e.g., API level, additional API libraries) required by the app
- lists any user permissions required by the app

THE FIRST ANDROID APP

- The application shows a button and a string of text
- When the button is pressed, the text changes



APPLICATION CLASSES

- **View**: base class for user interface components (both widgets and layouts)
- **ViewGroup**: base class for layouts and views containers. Extends `View`
- **Activity**: base class for activities
- **LinearLayout**: arranges its children in a single row (default) or column. Extends `ViewGroup`
- **Button, TextView**: UI widgets. Extend `View`

HELLOWITHBUTTON.JAVA (1/2)

```
package it.unipd.dei.espl112.hellowithbutton;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.LinearLayout;

public class HelloWithButton extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // Create the TextView
        final TextView tv = new TextView(this);
        tv.setText("Press the button, please");

        // Create the Button
        Button bu = new Button(this);
        bu.setText("Press me");

        ...
    }
}
```

HELLOWITHBUTTON.JAVA (2/2)

...

```
// Set the action to be performed when the button is pressed
bu.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Perform action on click
        tv.setText("Good job!");
    }
});
```

```
// Create the layout
LinearLayout mylayout = new LinearLayout(this);
```

```
// Add the UI elements to the layout
mylayout.addView(bu);
mylayout.addView(tv);
```

```
// Display the layout
setContentView(mylayout);
```

```
}
```

```
}
```

ANDROIDMANIFEST.XML

- Automatically generated from properties that the programmer specifies via Eclipse

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="it.unipd.dei.esp1112.hellowithbutton"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloWithButton"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

STRINGS.XML

- Automatically generated by Eclipse when the app project is created

```
<resources>

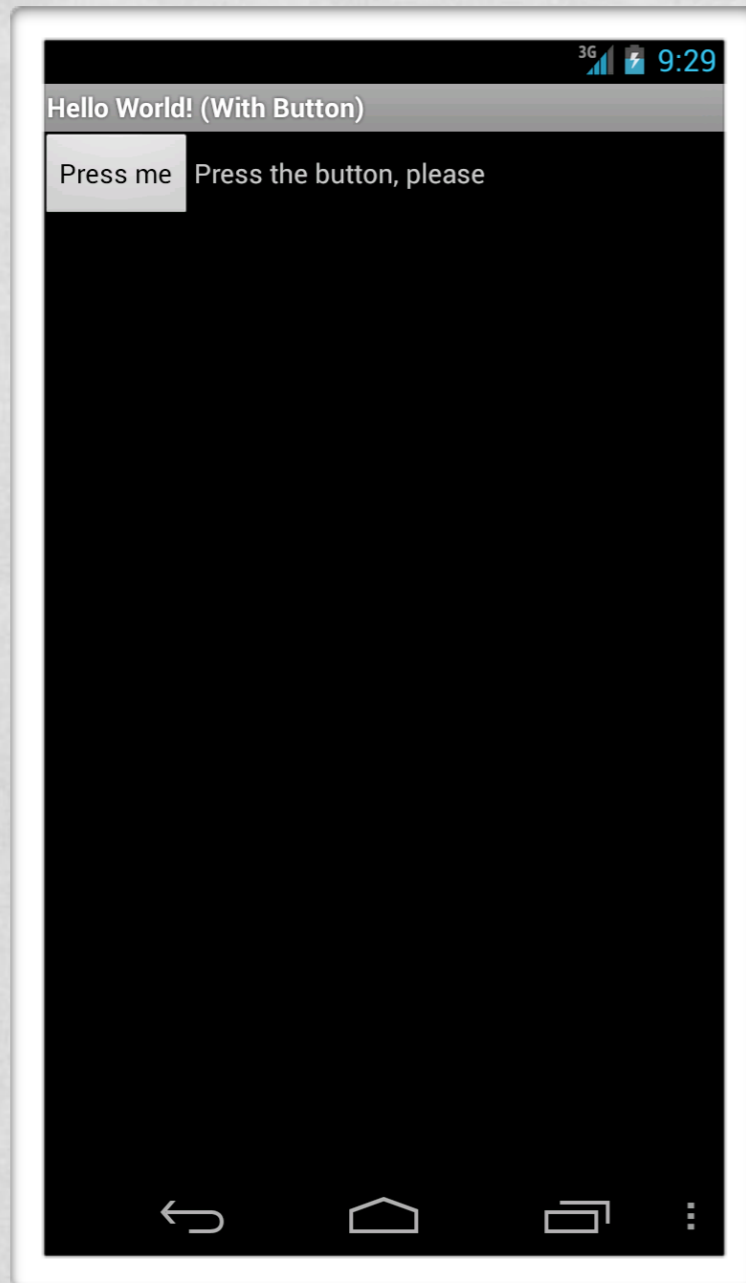
    <string name="app_name">HelloWithButton (Froyo)</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_hello_with_button">Hello World! (With Button)</string>

</resources>
```

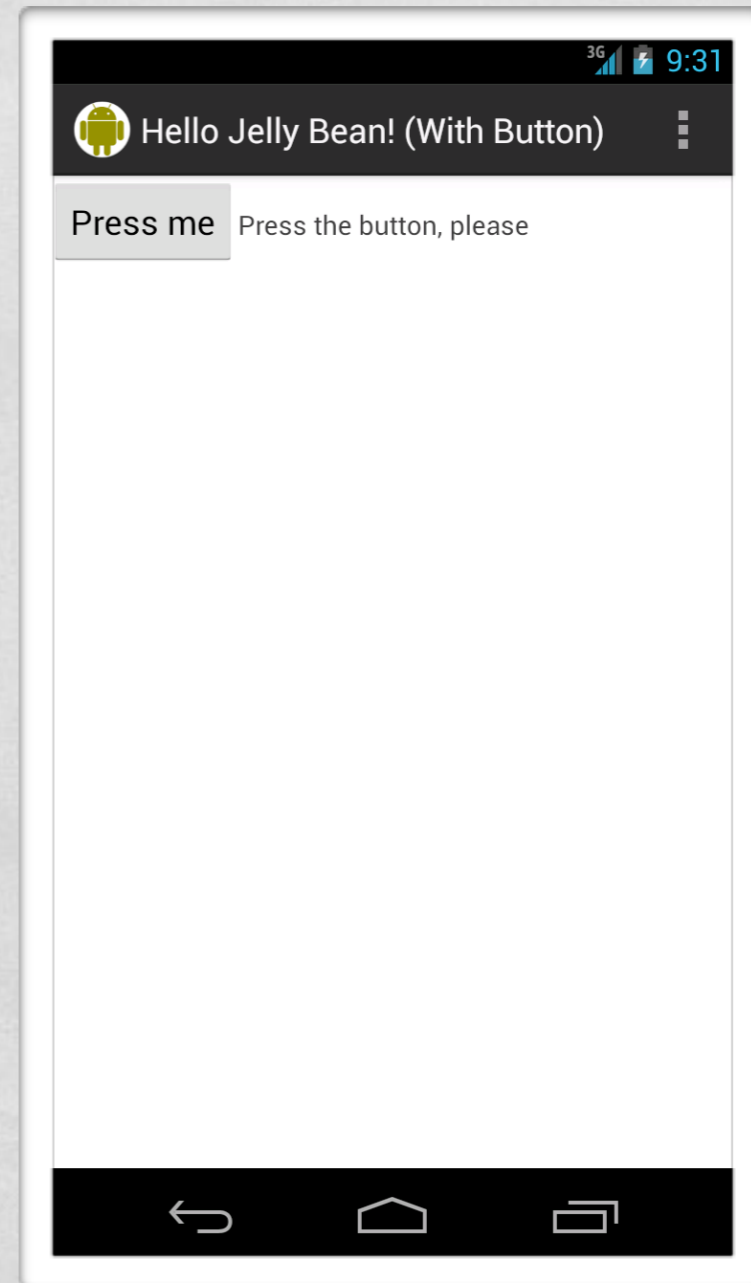
<USES-SDK>:ATTRIBUTES

- **android:minSdkVersion** (integer)
Minimum API Level required for the app to run.
If not properly set, the app will crash at runtime as soon as it accesses an unsupported API
- **android:targetSdkVersion** (integer)
API level against whom the app has been tested. The app is still able to run on lower API levels, down to `minSdkVersion`.
If not set, it is assumed equal to `minSdkVersion`.
- **android:maxSdkVersion** (integer)
Maximum API Level on which the app can run.
Declaring this attribute is not recommended: new versions of Android are designed to be backward-compatible

<USES-SDK>: EXAMPLE



targetSdkVersion=8,
Android 4.2.2 (API Level 17)



targetSdkVersion=17,
Android 4.2.2 (API Level 17)

API LEVEL

- The **API Level** is an integer that identifies the set of APIs supported by a given version of Android
- Examples:
 - Android 2.2 ↔ API Level 8
 - Android 2.3 ↔ API Level 9
 - Android 2.3.3, 2.3.4, ... ↔ API Level 10
 - Android 4.0.3, 4.0.4 ↔ API Level 15
- To sum things up, each Android release is identified by a Platform Version (e.g., 2.2), an API Level (e.g., 8) and a Version Code (e.g., “Froyo”)

LAST MODIFIED: MARCH 5, 2015

COPYRIGHT HOLDER: CARLO FANTOZZI (FANTOZZI@DEI.UNIPD.IT)
LICENSE: [CREATIVE COMMONS ATTRIBUTION SHARE-ALIKE 3.0](https://creativecommons.org/licenses/by-sa/3.0/)