Android Application Design

Chapter 2

Prepared By : Dharmendra Ambani (Harivandana College – Rajkot)

2.1) Anatomy(Structure) of an Android applications

- Src : Java source code files will be available here.

- Gen : The gen directory in an Android project contains auto generated files. you can see R.java inside this folder which is a generated class which contains references to certain resources of the project.R.java is automatically created by the Eclipse IDE.

2.1) Anatomy(Structure) of an Android applications

- Res : Android supports resources like images and certain XML configuration files , these can be keeping separate from the source code.

- /res/values : Used to define strings, colors, dimensions, styles, static arrays etc.
- Ex. String are defined in the res/values/string.xml file.

2.1) Anatomy(Structure) of an Android applications

- /res/layout : This folder contains the layouts to be used in the application.
- /res/menu : This folder contains menu resources to be used in the application.

2.1) Anatomy(Structure) of an Android applications

- /res/drawable : Drawable folders are resource directories in the application that provides differrent bitmap drawables for medium, hign and extra high density screens.

- /res/drawable-idpi : bitmap for lower density
- /res/drawable-mdpi : bitmap for medium density
- /res/drawable-hdpi : bitmap for high density
- /res/drawable-xhdpi : bitmap for extra high density
- /res/drawable-xxhdpi : bitmap for X extra high density

2.1) Anatomy(Structure) of an Android applications

- Libs : External library files will be places in this folder.
- Assets : This folder contains raw hierarchy of files and directories, with no other capabilities.
- Bin : Bin folder is the area used by the compiler to prepare the files to be finally packaged to the application’s APK file.

2.1) Anatomy(Structure) of an Android applications

- AndroidManifest.xml : All the android applications will have an AndroidManifest.xml file in the root directory. In short it is a configuration file.

- Ic_launcher-web.png : This is an icon to be used in Google play.
- 512 x 512 pixels

2.1) Anatomy(Structure) of an Android applications

- Proguard-project.txt : Everything in the proguard-project.txt file will be used just run ProGuard tool with standard settings.

- Project.properties : Is the main project's properties files containing information such as the build platform target and dependancies.

2.2) Android Terminologies

- .apk file : Android application package file.
- .dex file : Compiled Android application code file. .dex files can be created by automatically translating comipiled applications written in the java programming language.
- Action : An action is a string value assigned to Intent
- Ex. Android.intnet.action.VIEW

2.2) Android Terminologies

- Activity : A single screen in an application, with supporting java code, derieved from the Activity class.
- Adb : Android Debug Bridge, a command line debugging application includes with the SDK.
- Application : From a component viewpoint, an Android application consists of one or more activities, listeners, and intent receivers.

2.2) Android Terminologies

- Canvas : Canvas is the simplest, easiest way to draw 2D objects on the screen.
- Content Provider : A content provider is built on the ContentProvider class, which handles content query strings of a specific format to return data in a specific format.
- Dalvik: The Android Platform’s virtual machine. The Dalvik VM is an interpreter only VM that executes files in the Dalvik Executable (.dex) format, a format that is optimized for efficient storage and memory mappable execution.

2.2) Android Terminologies

- DDMS : Dalvik Debug Monitor Service, a GUI debugging application included with the SDK.
- Dialog : A dialog can have button controls only and is intended to perform a simple action.
- Drawable : A drawable is typically loaded into another UI element , Ex.

2.2) Android Terminologies

- Intent : It includes several criteria fields that you can supply, activity receives the intent and what the receiver does when handling the intent.

- Intent Filter : Through an intent filter, an application can express interest in specific data types, Intent actions etc.

2.2) Android Terminologies

- Broadcast Receiver : An application class that listens for intents that are broadcast, rather that being sent to a single target application/activity.

- Layout Resources : An XML file that describes that layout of an Activity scree.

- Manifest File : An XML file that each application must define, to describe the app's package name, version, component etc.

2.2) Android Terminologies

- Resources : Non programatic application components that are external to the compiled application code, but which can be loaded from application code.

- Service : An object class service that runs in the background to perform action.

2.2) Android Terminologies

- Surface : An object of type surface representing a block of memory that get composited to the screen
- Surface View : A view object that wraps a surface for drawing and exposes methods to specify its size and format dynamically.
- Theme : A set of Properties (Text size, color etc) bundled together to define various default display settings.

2.2) Android Terminologies

- View : An object that draws to a rectangular area on the screen and handles click and events.

- Viewgroup : A container object that groups is responsible for deciding where child views and representing.

- Widget : One of a set of fully implemented view subclasses that render from elements and other UI component.

2.3) Application Context, Activities, Services, Intents

- Context : is probably the most used element in android applications.
- This is a class whose implementation is provided by the **Android** system. It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc

- Activities
- A single screen in an application, with supporting java code, derieved from activity class.

Activities

- Priority
- 1)Foreground
- 2)visible
- 3)service
- 4)background
- 5)Empty

Activities

- Activity Life cycle
- An activity can be in different state which are describe by follow.

- Activity State
- 1)Running
- 2)Paused
- 3)Stopped
- 4)Killed

Activities

- Activity Life cycle methods
- 1)onCreate()
- 2)onResume()
- 3)onPause()
- 4)onStop()

Activities

- 1)onCreate()
- Called when the activity is created , used to initialize activity.
- For Ex. Create the user interface.

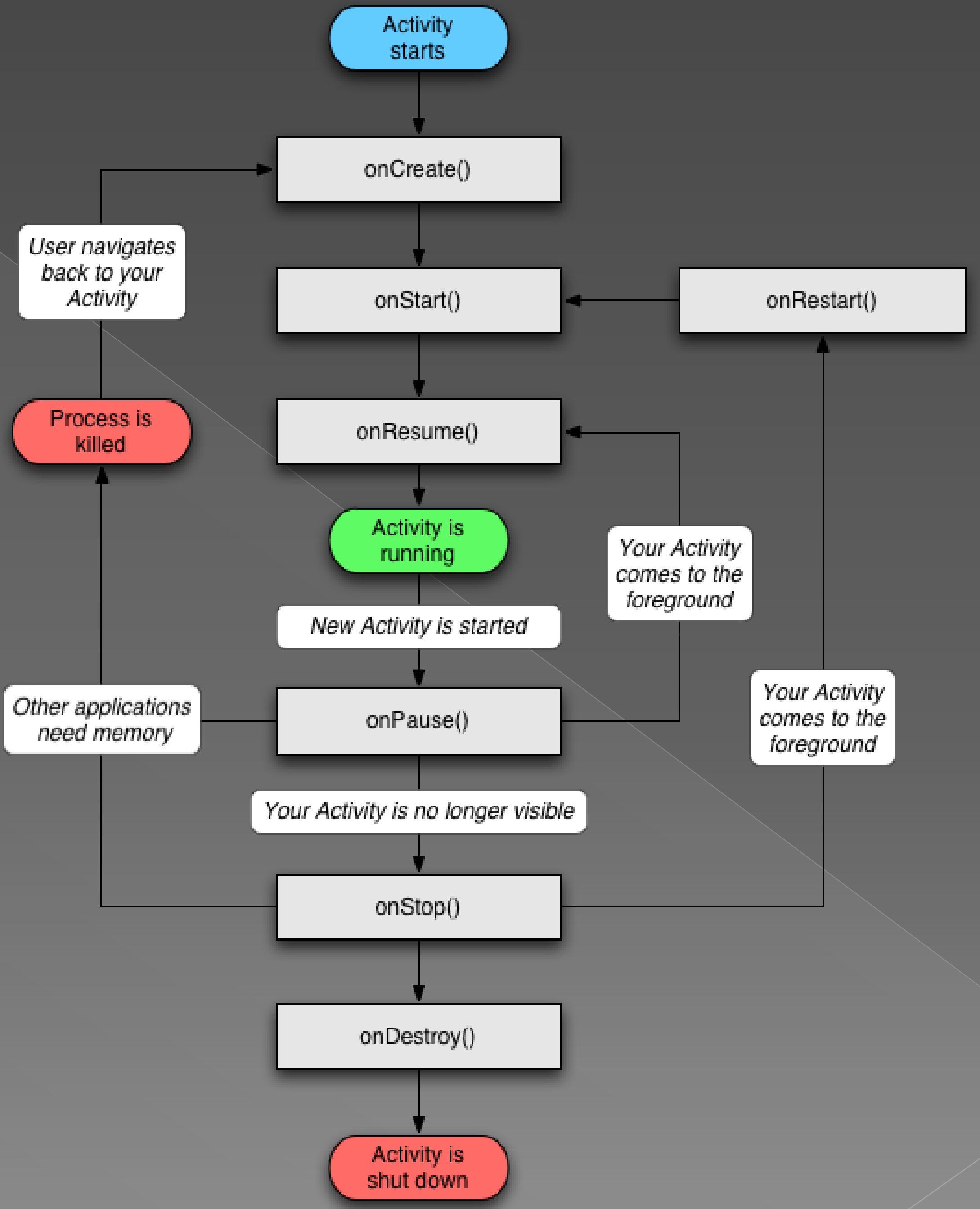
- 2)onResume()
- Called if the activity gets visible again and the user starts interacting with activity again.
- Used to initialize fields , register listeners, bind to service etc.

Activities

- 3) onPause()
- Called ones another activity gets into the foreground.
- Always called before the activity is not visible any more.
- For Ex. You unregister listeners, intent receiver, unbind from service or remove system service listeners.

Activities

- 4) onStop()
- Called ones activity is no longer visible.
- Time or cpu intensive shut down operations.
- For Ex. writing information to database shut be down in the on stop methods.



Activity starts
onCreate()
User navigates back to your Activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
Your Activity comes to the foreground
New Activity is started
Other applications need memory
onPause()
Your Activity comes to the foreground
Your Activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Services

- A service is a component that runs in the background to perform long running operations without needing to interact with the user.
- For Ex. Play music in the background.

- Service State
- 1) Started
- 2) Bound

Services

- 1) Started
- A service is started when an application component, such as an activity, starts it by calling startService().

- Once started, a service can run in the background indefinite, even if the component that stated it is destroyed.

Services

- 2) Bound
- A service is bound when an application component binds to it by calling bindService().

- A Bound service offers a client server interface that allows components to interact with the service, send request, get results and even do so across processes with interprocess communication.

Services

- Callback Method
- 1)onStartCommand()
- The system calls this method when another component, such as an activity, requests that the service be started by calling startService().

- 2)OnBind()
- The system calls this method when another components wants to bind with the service by calling bindService().

Services

- 3) onUnbind()
- The system calls this method when all clients have disconnected from a particular interface published by the service.

- 4) onRebind()
- The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its onUnbind(Intent).

Services

- 5) onCreate()
- The system calls this method when the sevice is first created using onStartCommand() of onBind(). This call is required to perform one time setup.

- 6) onDestroy()
- The system calls this method when the service is no longer used and is being destroyed.

Intent

- Types of Intent
- 1) Explicit Intent
- Specify the component to start by name(class name).
- For Example, start a new activity is response to a user action or start a service to download a file in the background.

Intent

- 2) Implicit Intents
- Do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.
- For Example, If you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a app.

1.4) Receiving and broadcasting Intents

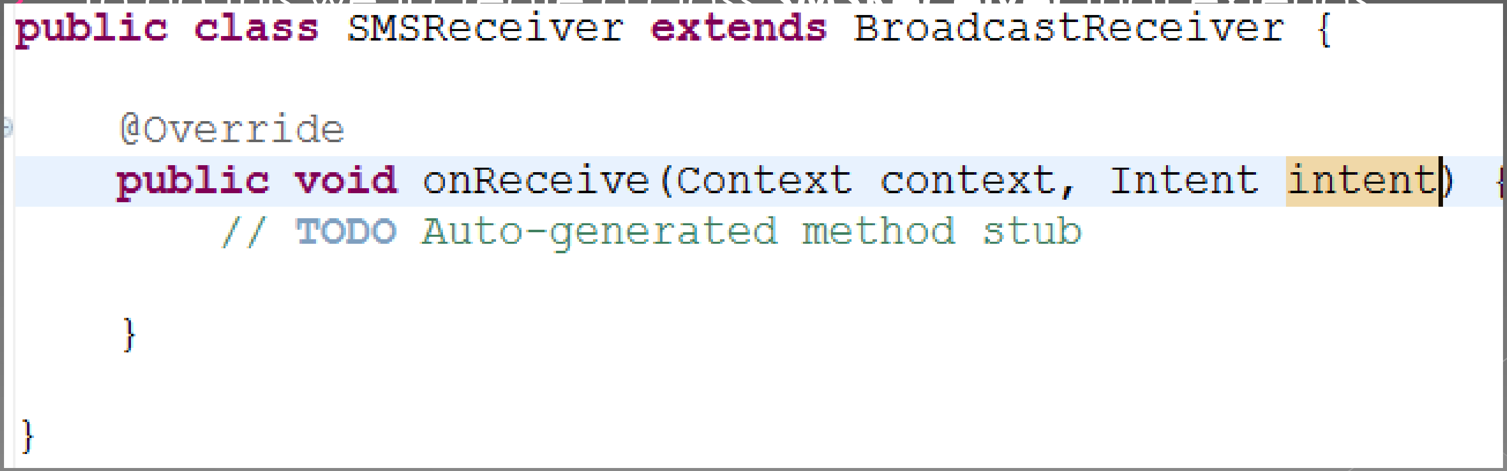
- A broadcast receiver listens for relevant broadcast messages to trigger an event.
- Example :
  - A camera button was pressed.
  - The Battery is low.
  - A new application was installed.
  - SMS is received.
  - Screen has turned off.

1.4) Receiving and broadcasting Intents

- A broadcast receiver is a class which extends "BroadcastReceiver" and which is registered as a receiver in an Android Application via the AndroidManifest.xml (or via code).
- This class will be able to receive intents via the sendBroadcast() method.
- "BroadCastReceiver" defines the method "onReceive()".

1.4) Receiving and broadcasting Intents

1. A broadcast receiver is implemented as a subclass of **BroadcastReceiver** and each
   - broadcast is delivered as an **Intent** object. In this case the intent is detected by
   - android.provider.Telephony.SMS_RECEIVED
   - To do this we'll create a class **SMSReceiver** that extends



```
public class SMSReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO Auto-generated method stub

    }

}
```

1.5) Android Manifest File and Its Common settings

- It is a specially formatted XML File that much accompany each android application.
- This file contains important information about the application's identity.
- In this file we can define the
  - Applications' name
  - Version information
  - Permissions and
  - Other application configurations

1.5) Android Manifest File and Its Common settings

- Information in this file is used by the android system to
  - Install and Upgrade the application package.
  - Display the application details such as the application name, description, and icon to users.
  - Specify application system requirements including which Android SDKs are supported,
  - What hardware configuration are required.
  - Which platform features the application relies upon(Eg. Uses multi touch capabilities).
  - Launch application activities.
  - Manage application permissions.

1.5) Android Manifest File and Its Common settings

- Eclipse Manifest file resource editor organizes the manifest information into categories :
  - The Manifest tab
  - The Application tab
  - The Permissions tab
  - The Instrumentation tab
  - The AndroidManifest.xml tab

1.5) Android Manifest File and Its Common settings

Configuring Package-Wide Settings using the Manifest Tab

- Its Include the package name.
- Version Information.
- Supported Android SDK.
- Also set any hardware or
  or feature requirement.

1.5) Android Manifest File and Its Common settings

Managing Application and Activity Settings using the Application Tab

- It's contains Application label and icon.
- Application components such as activities, intent filters and other application components.
- Configuration for service, intent filter, and content providers.

1.5) Android Manifest File and Its Common settings

Enforcing Application Permissions using the Permissions Tab

- A Permission is a restriction limiting access to a part of the code or to data on the device.
- The limitation is imposed to protect critical data and code that could be misused to damage the user experience.
- Each permission is identified by a unique label.
- Eg. :
  - android.permission.CALL_EMERGENCY_NUMBERS
  - android.permission.SET_WALLPAPER
  - android.permission.DEVICE_POWER

1.5) Android Manifest File and Its Common settings

Managing Test Instrumentation using the Instrumentation Tab

- It allows the developer to declare any instrumentation classes for monitoring the application.
- Instrumentation classes that provide profiling and other information as the application is running.
- This declarations are present in manifest only while the application is being developed and tested, they are removed before the application is published.

1.5) Android Manifest File and Its Common settings

Editing the Manifest file manually

- Android Manifest file is a specially formatted XML file.
- We can edit manually by clicking on the AndroidManifest.xml tab.
- It's include a single <manifest> tag with a single <application> tag.
- Following is a sample AndroidManifest.xml file.

1.5) Android Manifest File and Its Common settings

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.android.multimedia"
        android:versionCode="1"
        android:versionName="1.0">
        <application android:icon="@drawable/icon"
                android:label="@string/app_name"
                android:debuggable="true">
                <activity android:name=".MultimediaMenuActivity"
                        android:label="@string/app_name">
                        <intent-filter>
                                <action
                                android:name="android.intent.action.MAIN" />
                                <category
                                android:name="android.intent.category.LAUNCHER"
    />
                        </intent-filter>
                </activity>
```

1.5) Android Manifest File and Its Common settings

AndroidManifest.xml

```
            <activity android:name="AudioActivity"></activity>
            <activity android:name="StillImageActivity"></activity>
            <activity android:name="VideoPlayActivity"></activity>
            <activity android:name="VideoRecordActivity"></activity>
      </application>
      <uses-permission
            android:name="android.permission.RECORD_AUDIO" />
      <uses-permission
            android:name="android.permission.SET_WALLPAPER" />
      <uses-permission
            android:name="android.permission.CAMERA"></uses-permission>
      <uses-sdk
            android:minSdkVersion="3"
            android:targetSdkVersion="8">
      </uses-sdk>
      <uses-feature
            android:name="android.hardware.camera" />
</manifest>
```

1.6) Using Intent Filter, Permissions

- Android system will determine suitable application for an implicit intent and if several applications exists offer the user the choice to open one.
- Intent filters are typically defined via the "AndroidManifest.xml" file.
- Implicit intents an application component must register itself via an IntentFilter in the "AndroidManifest.xml" to this event.

1.6) Using Intent Filter, Permissions

- Define a permission using the <uses-permission> tag.
- Eg.:

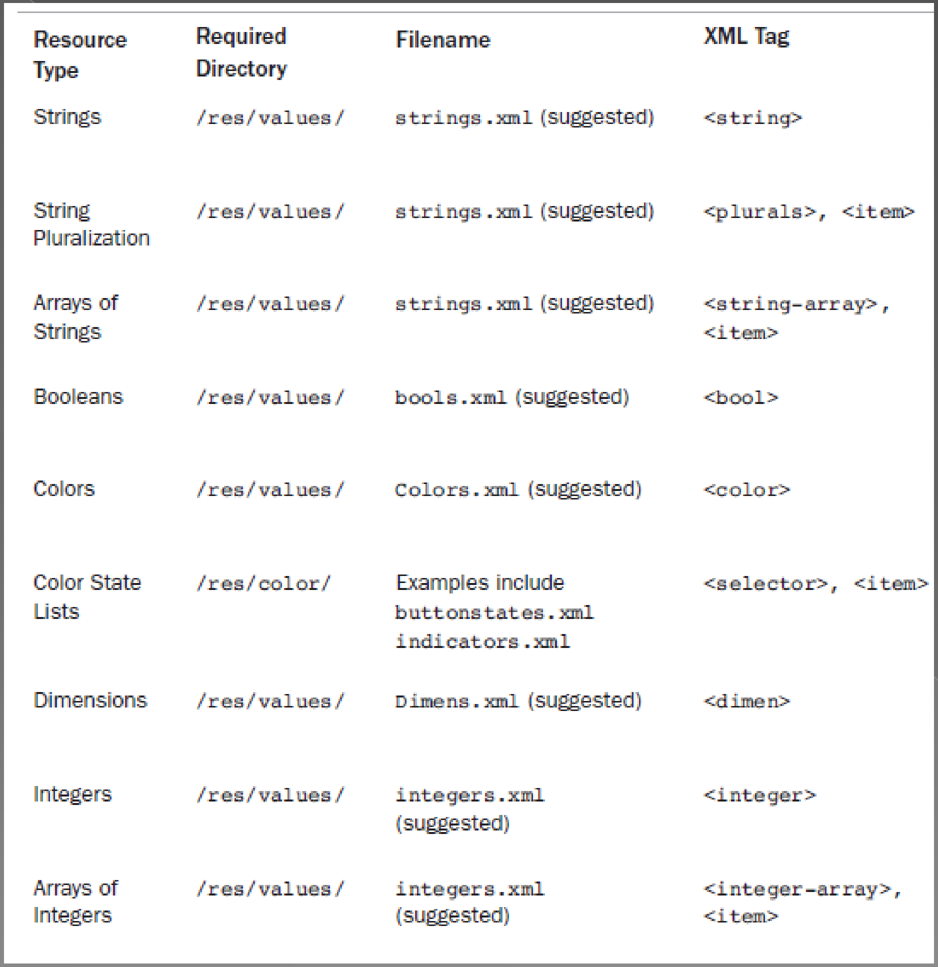
```
<uses-permission
        android:name="android.permission.came
ra"/>
```

- Permission can be enforce at several point :
  - Starting an Activity or Service
  - Accessing data provided by a content provider
  - Sending or receiving broadcasts by an Intent

1.7) Managing Application resources in a hierarchy

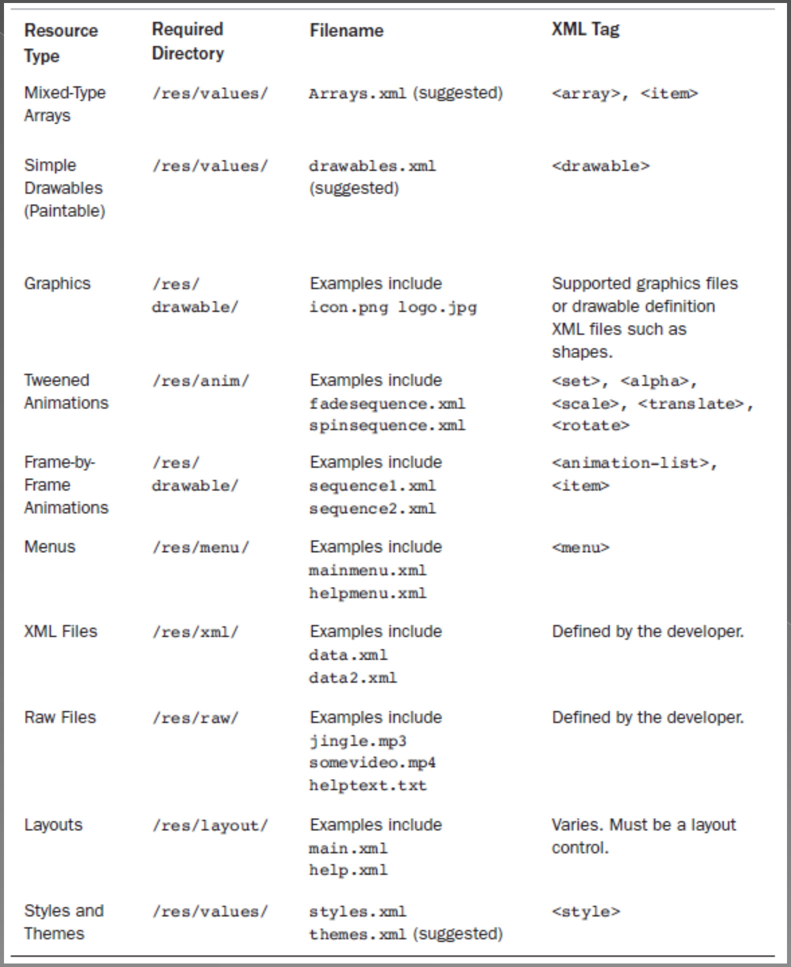
- Storing Application Resources.
- Understanding the Resource Directory Hierarchy.
  - /res/drawable/
  - /res/layout/
  - /res/values/

1.7) Managing Application resources in a hierarchy



Resource Type	Required Directory	Filename	XML Tag
Strings	`/res/values/`	`strings.xml` (suggested)	`<string>`
String Pluralization	`/res/values/`	`strings.xml` (suggested)	`<plurals>, <item>`
Arrays of Strings	`/res/values/`	`strings.xml` (suggested)	`<string-array>, <item>`
Booleans	`/res/values/`	`bools.xml` (suggested)	`<bool>`
Colors	`/res/values/`	`Colors.xml` (suggested)	`<color>`
Color State Lists	`/res/color/`	Examples include `buttonstates.xml` `indicators.xml`	`<selector>, <item>`
Dimensions	`/res/values/`	`Dimens.xml` (suggested)	`<dimen>`
Integers	`/res/values/`	`integers.xml` (suggested)	`<integer>`
Arrays of Integers	`/res/values/`	`integers.xml` (suggested)	`<integer-array>, <item>`

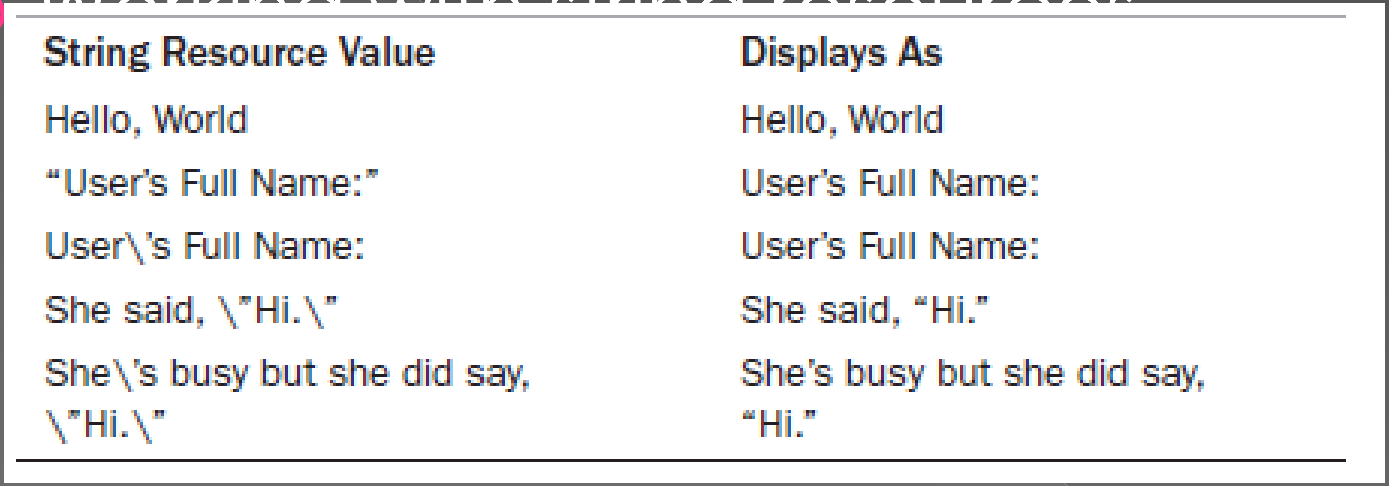
1.7) Managing Application resources in a hierarchy



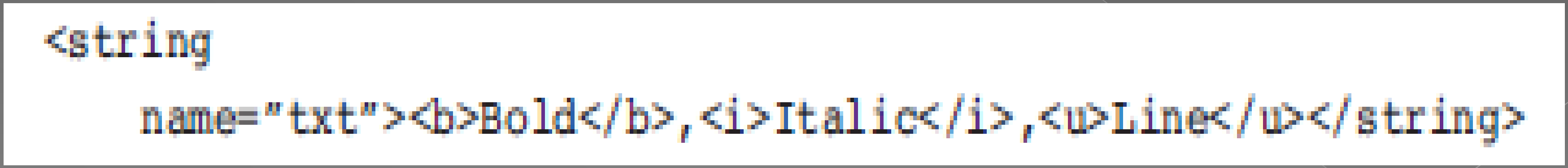
Resource Type	Required Directory	Filename	XML Tag
Mixed-Type Arrays	/res/values/	Arrays.xml (suggested)	<array>, <item>
Simple Drawables (Paintable)	/res/values/	drawables.xml (suggested)	<drawable>
Graphics	/res/drawable/	Examples include icon.png logo.jpg	Supported graphics files or drawable definition XML files such as shapes.
Tweened Animations	/res/anim/	Examples include fadesequence.xml spinsequence.xml	<set>, <alpha>, <scale>, <translate>, <rotate>
Frame-by-Frame Animations	/res/drawable/	Examples include sequence1.xml sequence2.xml	<animation-list>, <item>
Menus	/res/menu/	Examples include mainmenu.xml helpmenu.xml	<menu>
XML Files	/res/xml/	Examples include data.xml data2.xml	Defined by the developer.
Raw Files	/res/raw/	Examples include jingle.mp3 somevideo.mp4 helptext.txt	Defined by the developer.
Layouts	/res/layout/	Examples include main.xml help.xml	Varies. Must be a layout control.
Styles and Themes	/res/values/	styles.xml themes.xml (suggested)	<style>

1.8) Working with different types of resources

- Working with string resources



String Resource Value	Displays As
Hello, World	Hello, World
"User's Full Name:"	User's Full Name:
User\'s Full Name:	User's Full Name:
She said, \"Hi.\"	She said, "Hi."
She\'s busy but she did say, \"Hi.\"	She's busy but she did say, "Hi."



```
<string
    name="txt"><b>Bold</b>,<i>Italic</i>,<u>Line</u></string>
```

1.8) Working with different types of resources

- Using string resources as Format String.
- Using string resources programmatically.

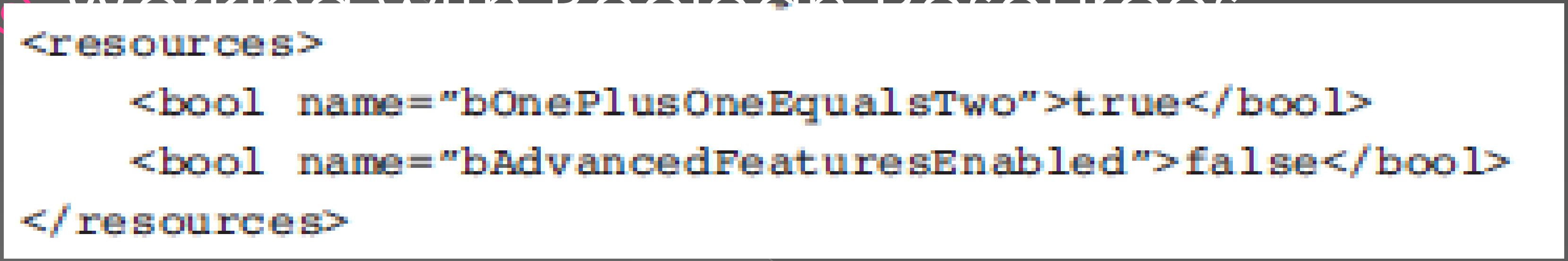
```
String myStrHello =
    getResources().getString(R.string.hello);
```

-

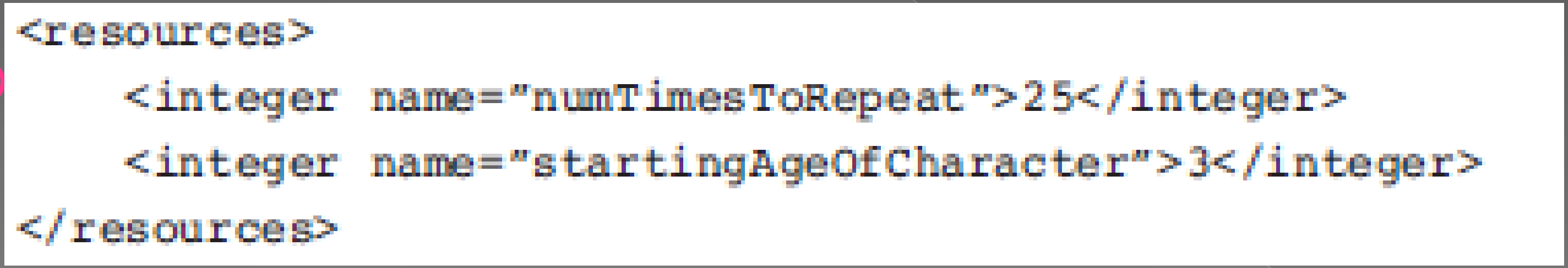
```
<string-array name="soups">
    <item>Vegetable minestrone</item>
    <item>New England clam chowder</item>
    <item>Organic chicken noodle</item>
</string-array>
```

1.8) Working with different types of resources

- Working with Boolean Resources



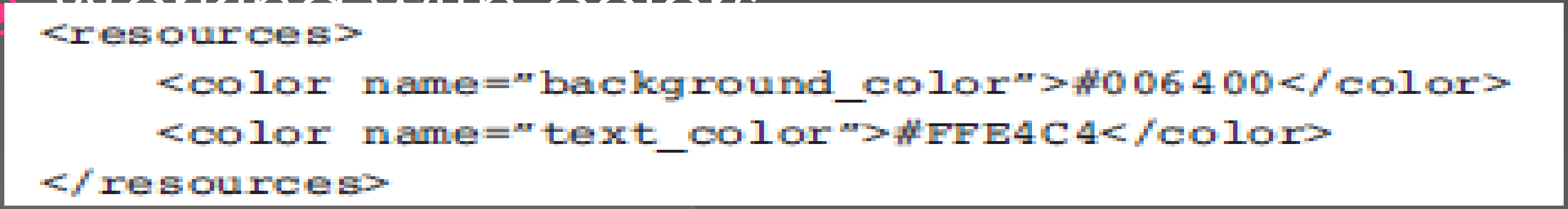
```
<resources>
    <bool name="bOnePlusOneEqualsTwo">true</bool>
    <bool name="bAdvancedFeaturesEnabled">false</bool>
</resources>
```

- 

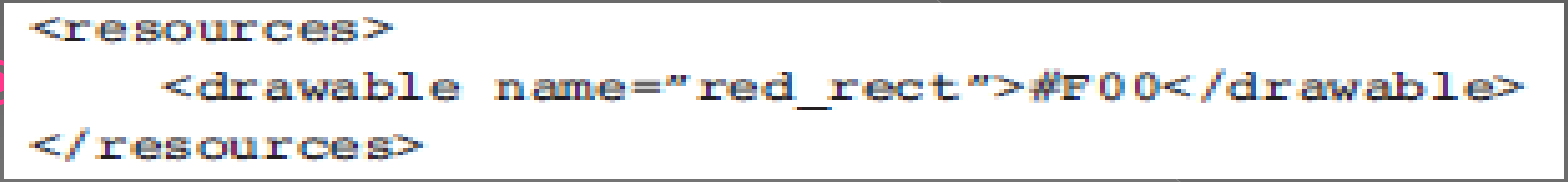
```
<resources>
    <integer name="numTimesToRepeat">25</integer>
    <integer name="startingAgeOfCharacter">3</integer>
</resources>
```

1.8) Working with different types of resources

- Working with colors

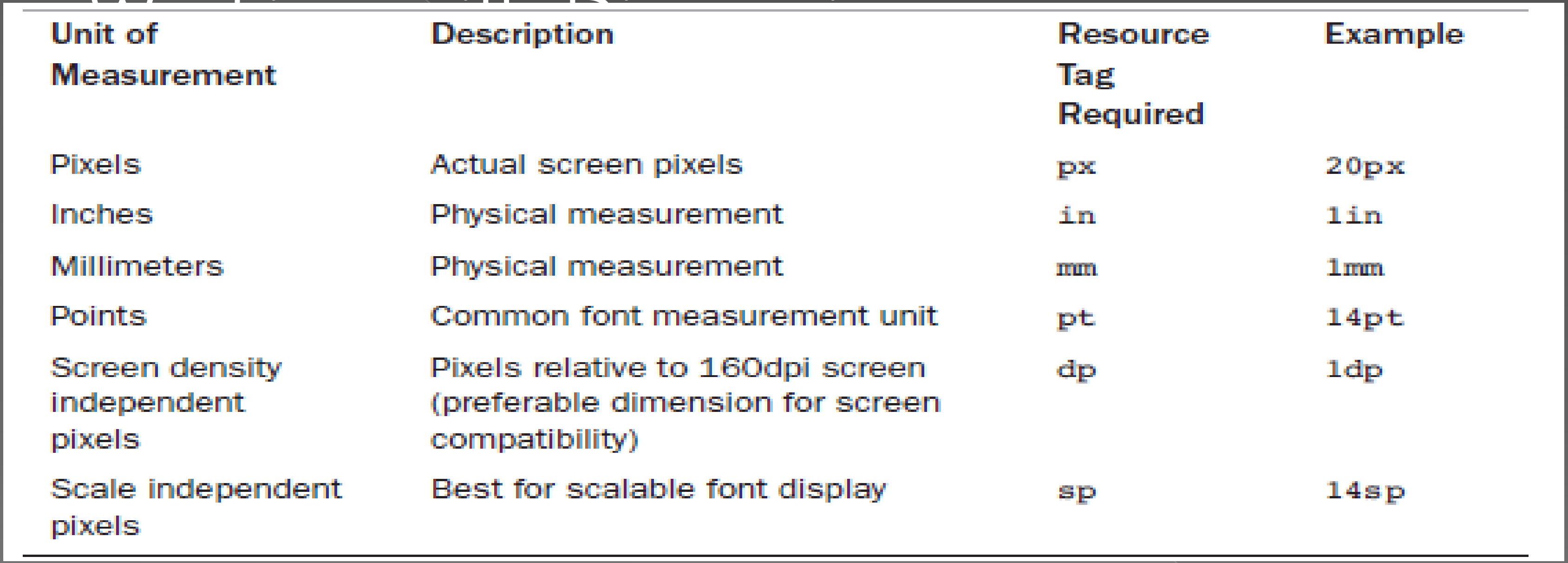


```
<resources>
    <color name="background_color">#006400</color>
    <color name="text_color">#FFE4C4</color>
</resources>
```

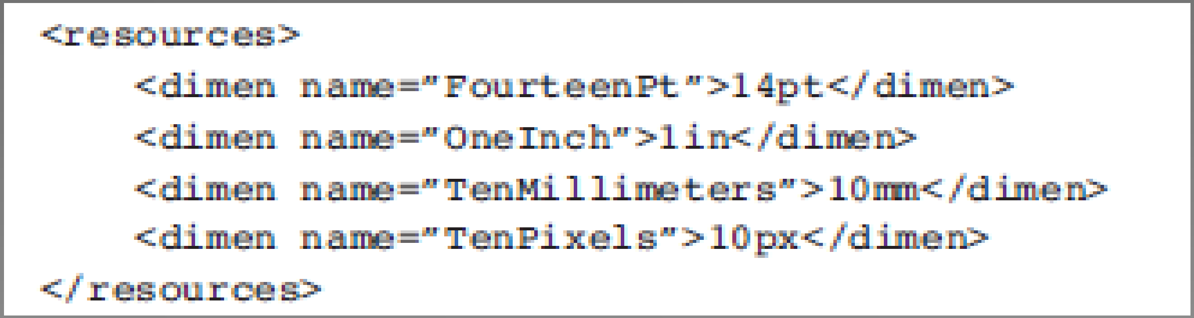


```
<resources>
    <drawable name="red_rect">#F00</drawable>
</resources>
```

1.8) Working with different types of resources

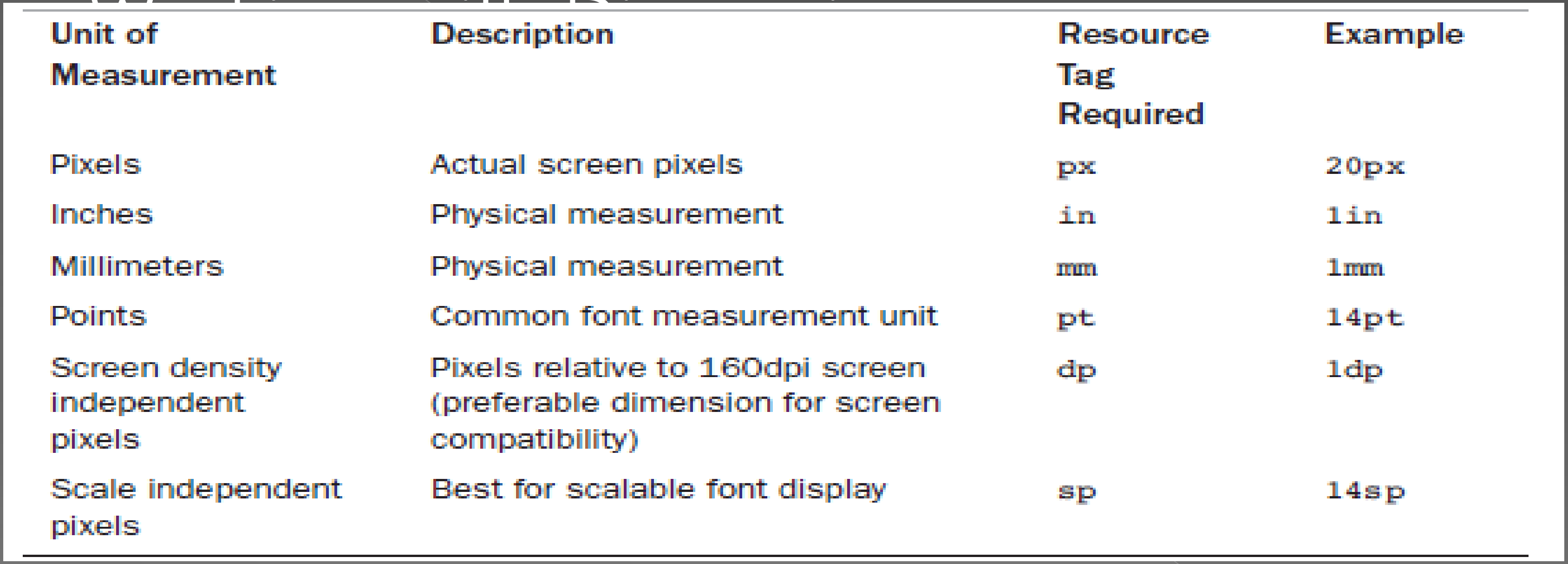


Unit of Measurement	Description	Resource Tag Required	Example
Pixels	Actual screen pixels	px	20px
Inches	Physical measurement	in	1in
Millimeters	Physical measurement	mm	1mm
Points	Common font measurement unit	pt	14pt
Screen density independent pixels	Pixels relative to 160dpi screen (preferable dimension for screen compatibility)	dp	1dp
Scale independent pixels	Best for scalable font display	sp	14sp

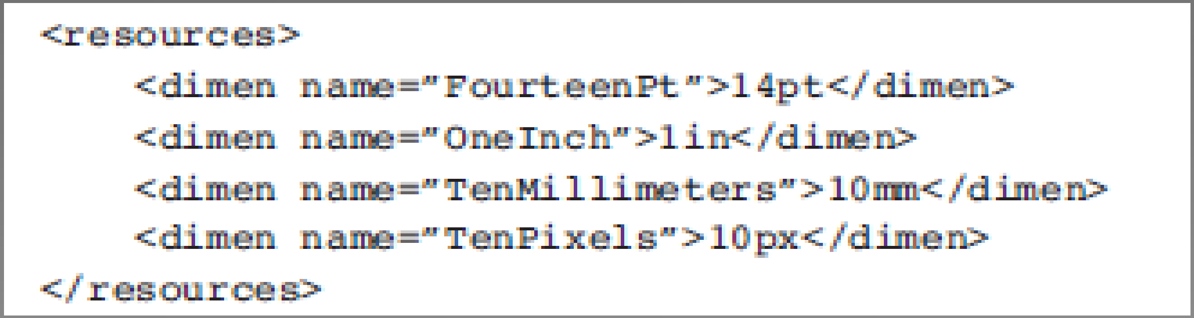


```
<resources>
    <dimen name="FourteenPt">14pt</dimen>
    <dimen name="OneInch">1in</dimen>
    <dimen name="TenMillimeters">10mm</dimen>
    <dimen name="TenPixels">10px</dimen>
</resources>
```

1.8) Working with different types of resources



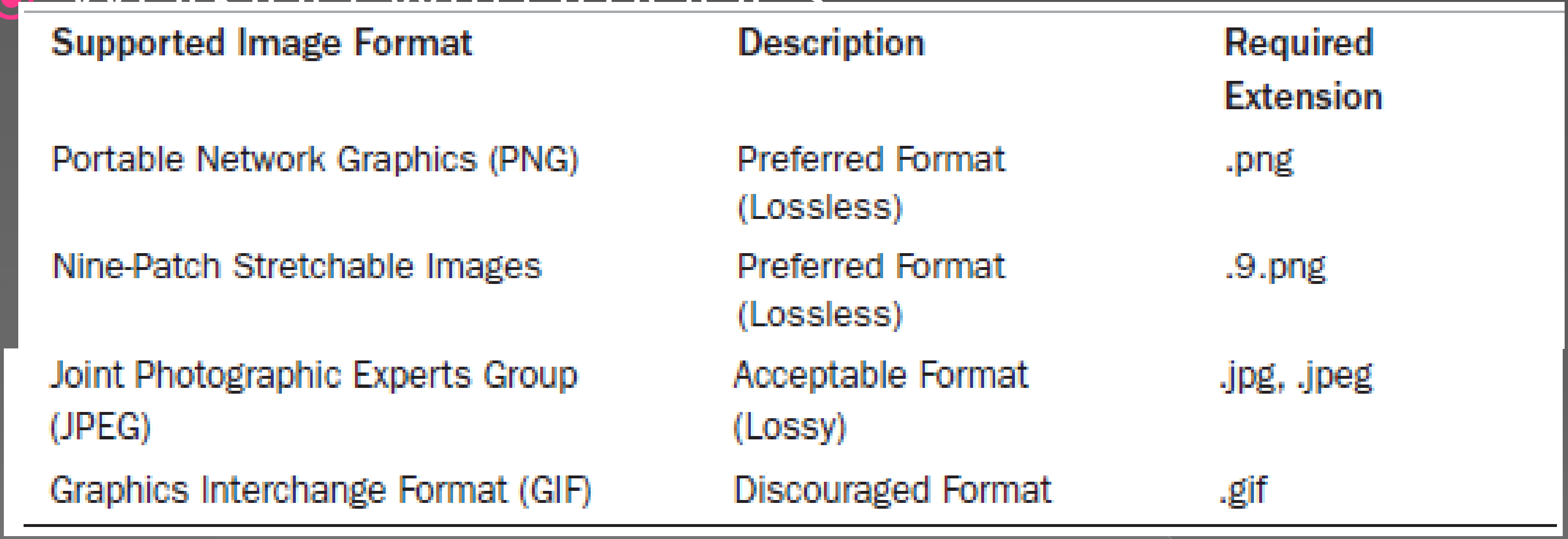
Unit of Measurement	Description	Resource Tag Required	Example
Pixels	Actual screen pixels	px	20px
Inches	Physical measurement	in	1in
Millimeters	Physical measurement	mm	1mm
Points	Common font measurement unit	pt	14pt
Screen density independent pixels	Pixels relative to 160dpi screen (preferable dimension for screen compatibility)	dp	1dp
Scale independent pixels	Best for scalable font display	sp	14sp



```
<resources>
    <dimen name="FourteenPt">14pt</dimen>
    <dimen name="OneInch">1in</dimen>
    <dimen name="TenMillimeters">10mm</dimen>
    <dimen name="TenPixels">10px</dimen>
</resources>
```

1.8) Working with different types of resources

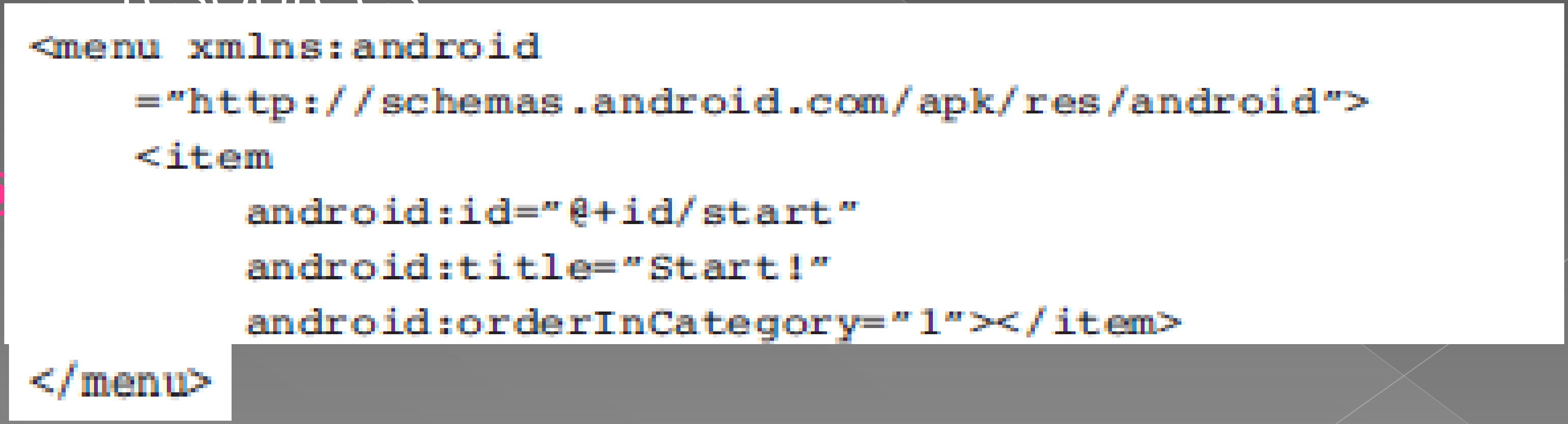
- Working with Images



Supported Image Format	Description	Required Extension
Portable Network Graphics (PNG)	Preferred Format (Lossless)	.png
Nine-Patch Stretchable Images	Preferred Format (Lossless)	.9.png
Joint Photographic Experts Group (JPEG)	Acceptable Format (Lossy)	.jpg, .jpeg
Graphics Interchange Format (GIF)	Discouraged Format	.gif

1.8) Working with different types of resources

- Working with Animation.
  - Defining and using frame by frame Animation Resources.
  - Defining and using tweened Animation resources



```
<menu xmlns:android
    ="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/start"
        android:title="Start!"
        android:orderInCategory="1"></item>
</menu>
```

1.8) Working with different types of resources

- Working with XML Files.
- Working with Raw Files.
- Working with Layouts.
- Working with Styles.
- References System Resources.