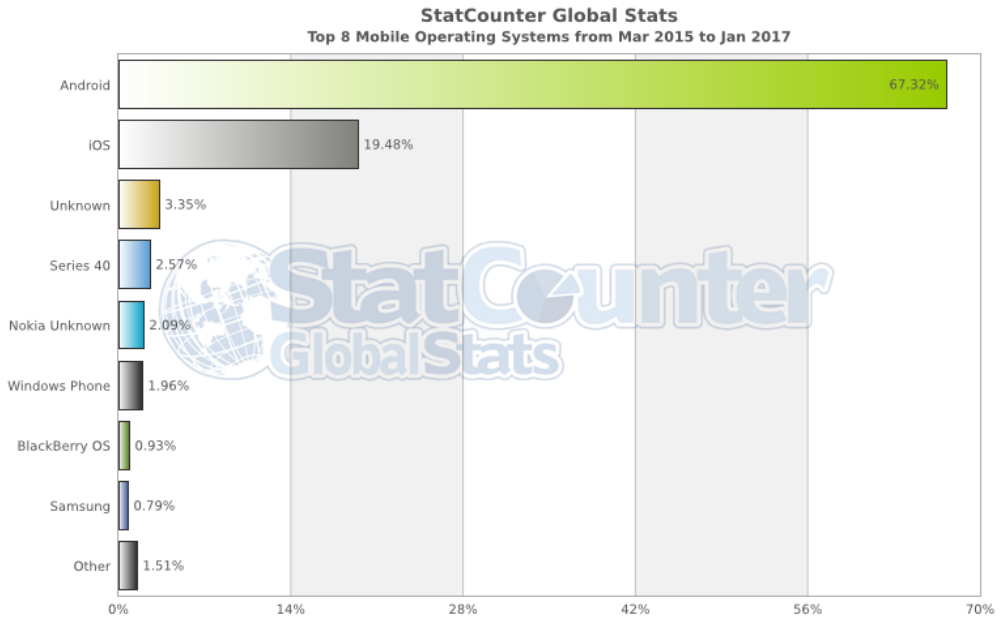
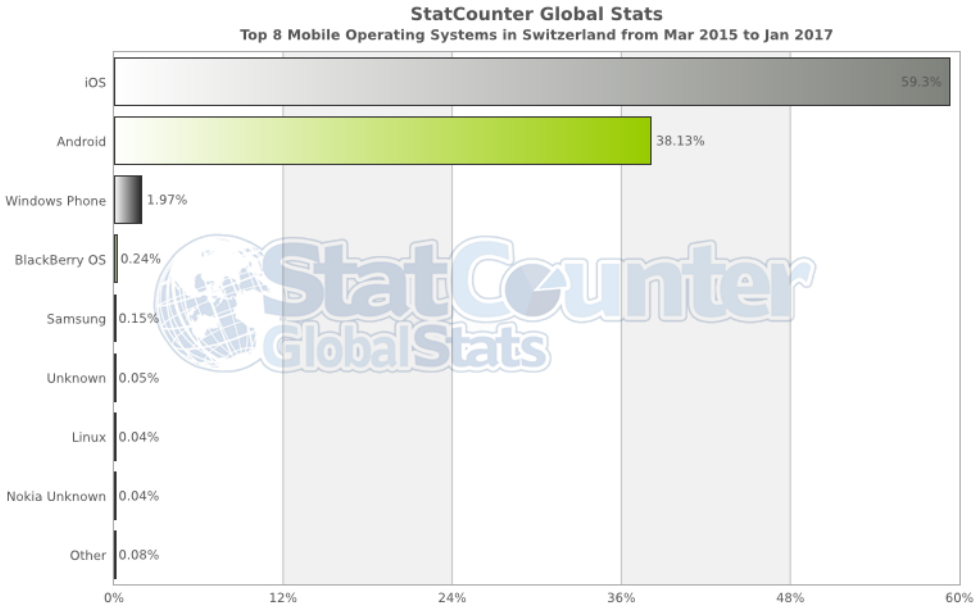
What will we look into today ?

- Basic Information about Android
- Android fundamentals
- Android components
  - Activities

Why Android?





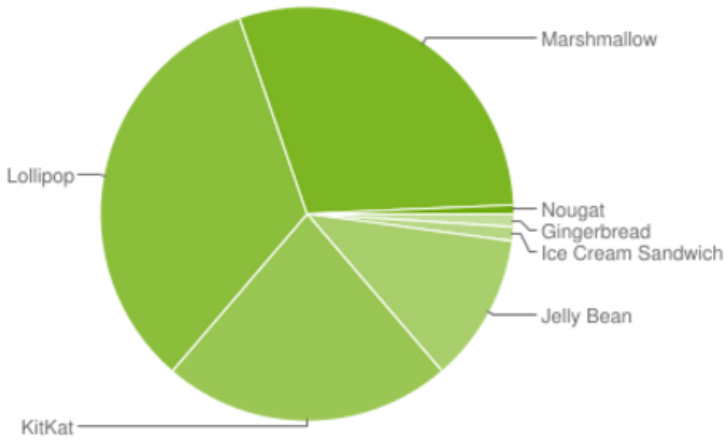
http://gs.statcounter.com/#mobile_os-ww-monthly-201503-201701-bar http://gs.statcounter.com/#mobile_os-CH-monthly-201503-201701-bar

Android History

- 2003 Founding of „Android“ by Andy Rubin
- 2005 Google pruchase «Android» for 50 Mio. $
- 2007 Google establishes the «Open Handset Alliance»
- 2008 First Android device(HTC Dream)
- 2010 Google releases the«Nexus One» there first own device
- 2010 More android device than IOS devices are sold
- 2014 Android Wear, TV and car
- 2014 1 Mia. monthly active Android user

Android today

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.1%
4.1.x	Jelly Bean	16	4.0%
4.2.x		17	5.9%
4.3		18	1.7%
4.4	KitKat	19	22.6%
5.0	Lollipop	21	10.1%
5.1		22	23.3%
6.0	Marshmallow	23	29.6%
7.0	Nougat	24	0.5%
7.1		25	0.2%



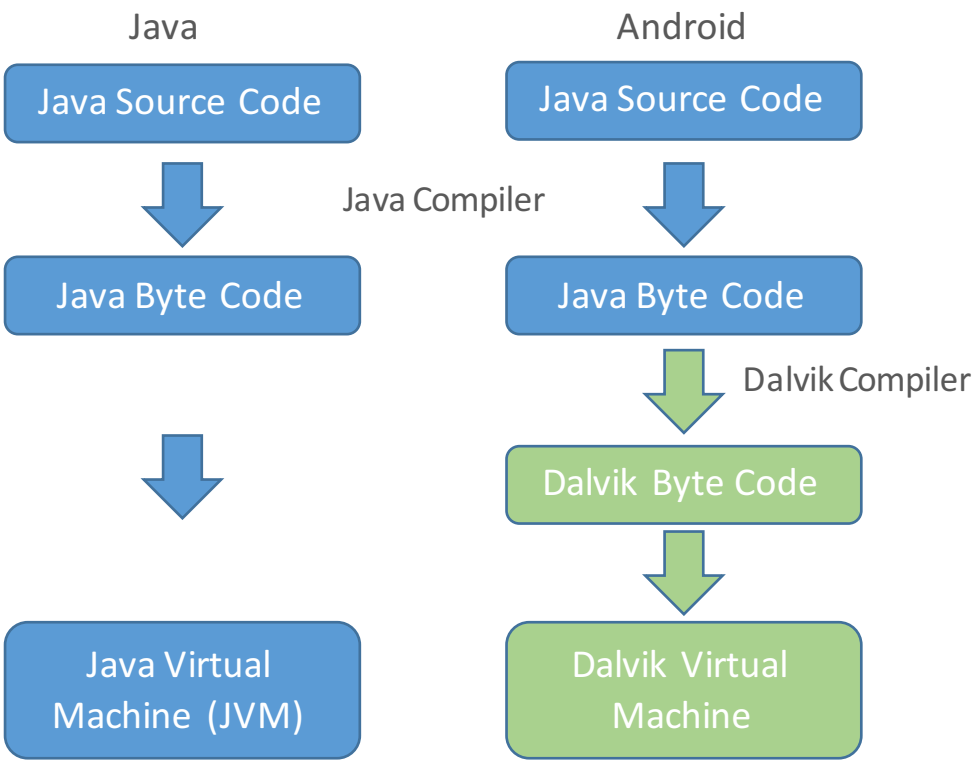
Data collected during a 7-day period ending on January 9, 2017.
Any versions with less than 0.1% distribution are not shown.

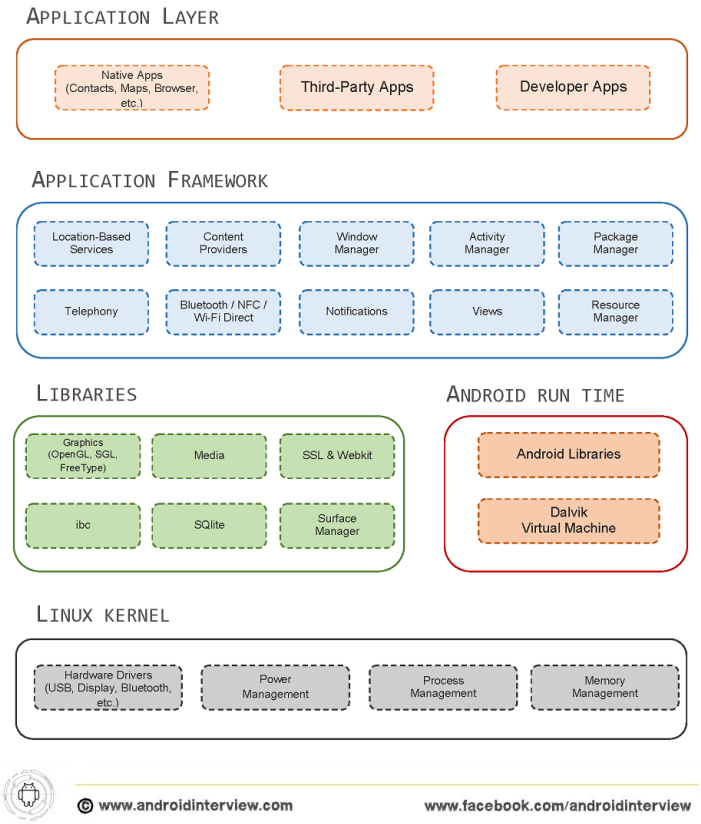
https://developer.android.com/about/dashboards/index.html

Android Developer Tools

- **Android Studio (AS)**
- SDK Manager in AS
- Android Virtual Device Manager (AVD)
- Android Emulator
- Android Device Monitor

Android System I





Android System II

- **Android < 5.0 (Dalvik Virtual Machine)**
  - Virtual Machine executing byte-code
  - Just-in-time compiler since Android 2.2
  - Optimized for Low memory requirements
  - Compile during every launch of the application
- **Android ≥ 5.0 (Android Runtime - ART)**
  - Ahead-Of-Time (AOT) – Compilation during installation
  - Backward compatible (same byte-code as Dalvik)

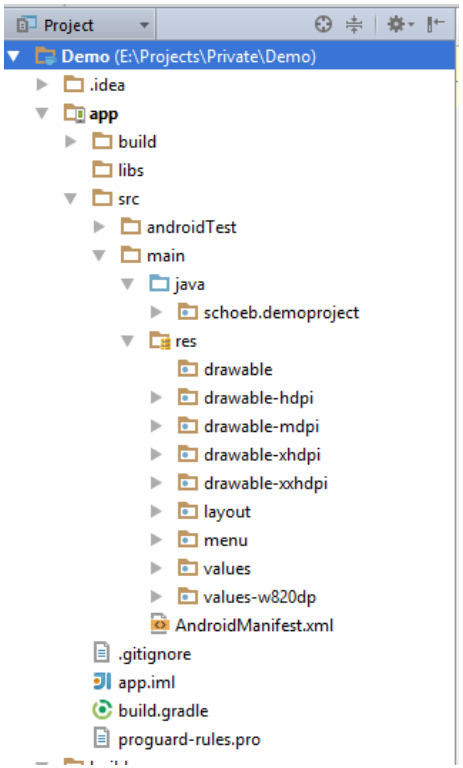
App fundamentals – general I

- Written in Java (C/C++, Kotlin)
- Packed in Android Package file «.apk» (It's just a zip)
- Installed on the device the app runs in it's own sandbox
  - Android is a multi-user-linux in which each app is a different user
  - Files are only accesable by this user
  - Each app runs in it's own process (normally…)
  - Each process has it's own VM
- Permissions give access to specific resources (e.g. SMS, Contact, Locations,…)

App fundamentals – general II

- A app consists of loosley coupled components
  - Activites, Services, Content Provider and Broadcast Receiver
- Components are hold together by «intents»
  - Some kind of messages to trigger another component
- Meta information about the app is stored in the «Manifest – File»
  - Permissions, available components, Broadcast Receivers, Services
- Application resources
  - Layouts, images, raw data , localizations, sounds,….

App fundamentals – Project structure



Folder	Description
java	Java source code
build	Generated files -> Never touch it !
libs	Include libraries (in addition to gradle dependencies)
res	Resources (layouts, colors, strings,….)

App fundamentals – Manifest

- Every app must have an «AndroidManifest.xml»
- Information about the app
  - Package name
  - Description of Components
  - Permissions (Careful: permission handling changed in Android 6.0)
  - API requirements (Will be overridden by the gradle build file)

App fundamentals – Manifest example

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="ch.schoeb.services“
          android:versionCode="1“
          android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8“ android:targetSdkVersion="17" />

    <application android:icon="@drawable/ic_launcher“
                          android:label="@string/app_name"
                 android:theme="@style/AppTheme" >
        <activity android:name="ch.schoeb.services.MainActivity"
                  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>
```

App fundamentals – Resources

- All data which is not code
- Stored in the «res»-folder
- Xml files
  - Layouts
  - Strings
  - Values (colors, dimensions,…)
  - Images (also vector graphics)
- Access these resources by the @-annotations

Declaration in Strings.xml:

```
<string name="app_name">Services</string>
```

Usage in any other XML-File:

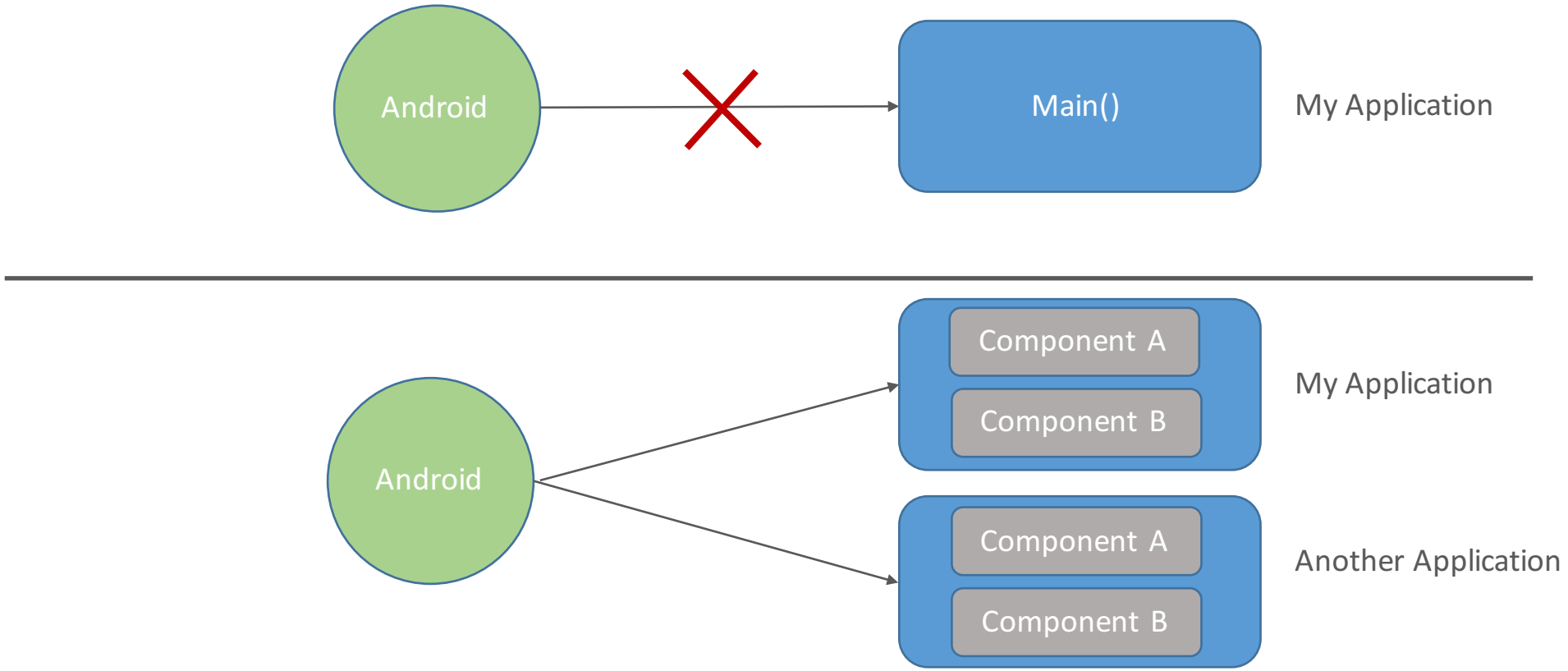
```
android:label="@string/app_name"
```

App fundamentals – R - Class

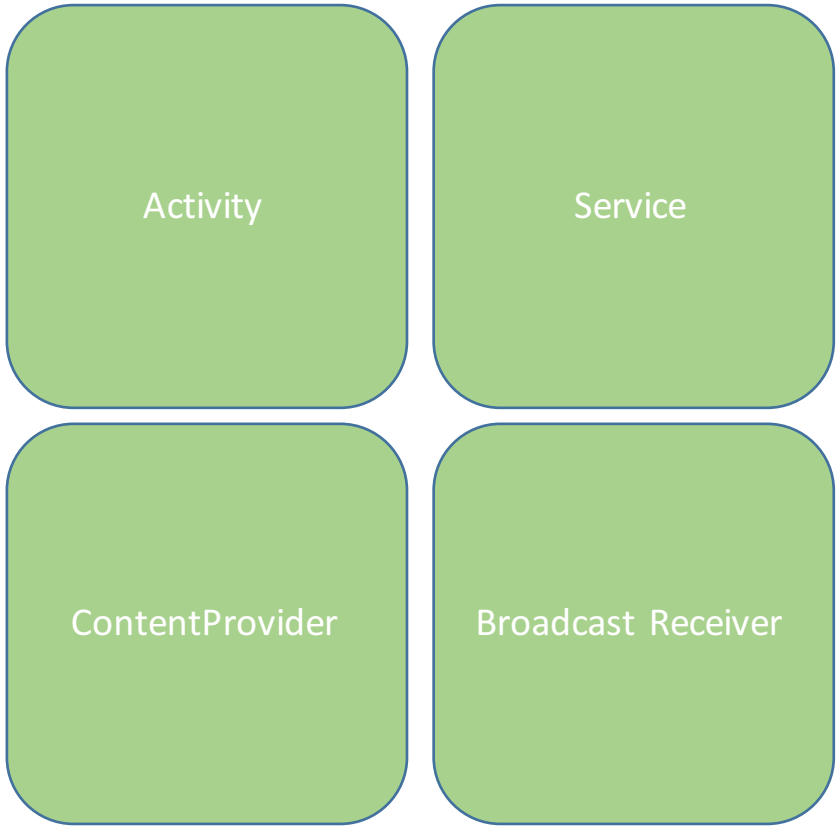
- Located in your «build/…» folder
- Automatically generated in each build
- Containts all resource Ids from your res-folder as public constants
- Subclasses for all Resources Types:
  - R.drawable, R.layouts, R.id, R.string,…
- Used to access resources in code

```
public final class R {
    public static final class string {
        public static final int action_settings=0x7f050001;
        public static final int app_name=0x7f050000;
}
```

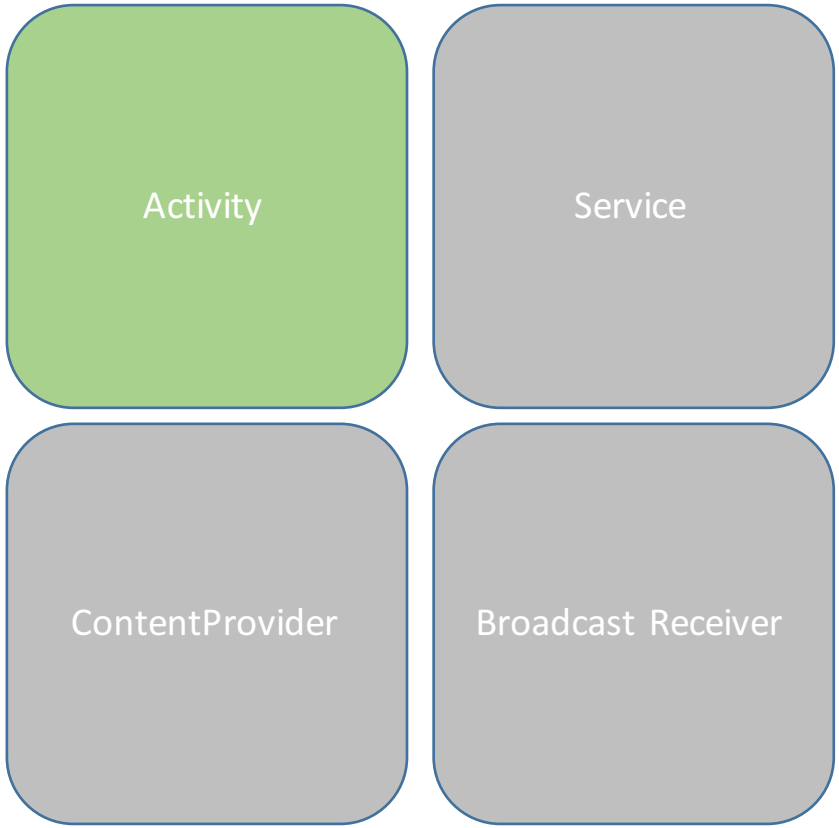
App fundamentals – Component driven



App fundamentals – Components

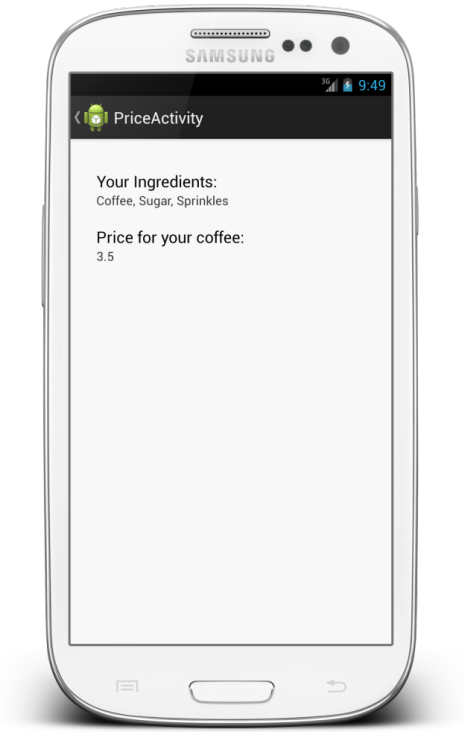


Components - Activity

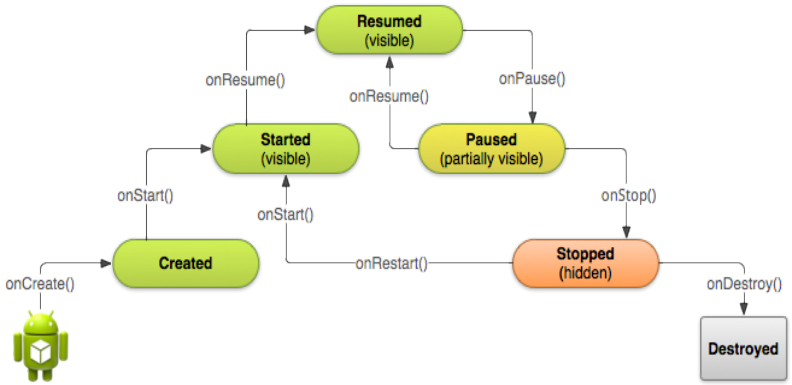


Activity - Overview

- Represents a single screen a user can interact with
- The UI is either defined in xml or directly in the code
- Normally a application consists of multiple activities
- One activity can consists of multiple fragments
- Beware of the **Activity lifecycle**
- Your classes must extend **Activity** or even better **AppCombatActivity**



Activity - Lifecycle

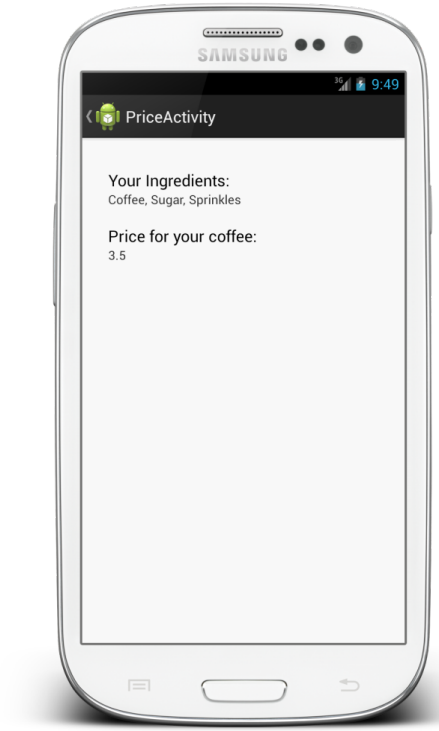


- **onCreate():** Called when the activity is first created
- **onStart():** Called when the activity becomes visible to the user
- **onResume():** Called when the activity starts interacting with the user
- **onPause():** Called when the current activity is being paused and the previos activity is being resumed
- **onStop():** Called when the activity is no longer visible to the user
- **onDestroy():** Called before the activity is destroyed by the system
- **onRestart():** Called when the activity has been stopped and is restarting again

Activity – Connect to the layout

PriceActivity.java
- Extend Activity class (better AppCompatActivity)
- Use lifecycle methods to connect to xml
- Use findViewById() to access view

activity_price.xml
- Declarative xml to define UI
- Define ID's for every view



Activity - Layout

activity_price.xml

```
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"  >

    <Button
         android:id="@+id/my_button"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="MyButton"  />

</RelativeLayout>
```

Activity – Connect to layout

Set layout for activity:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_price);
}
```

Access views in activity:

```
TextView textView= (TextView) findViewById(R.id.demoTextView);
```

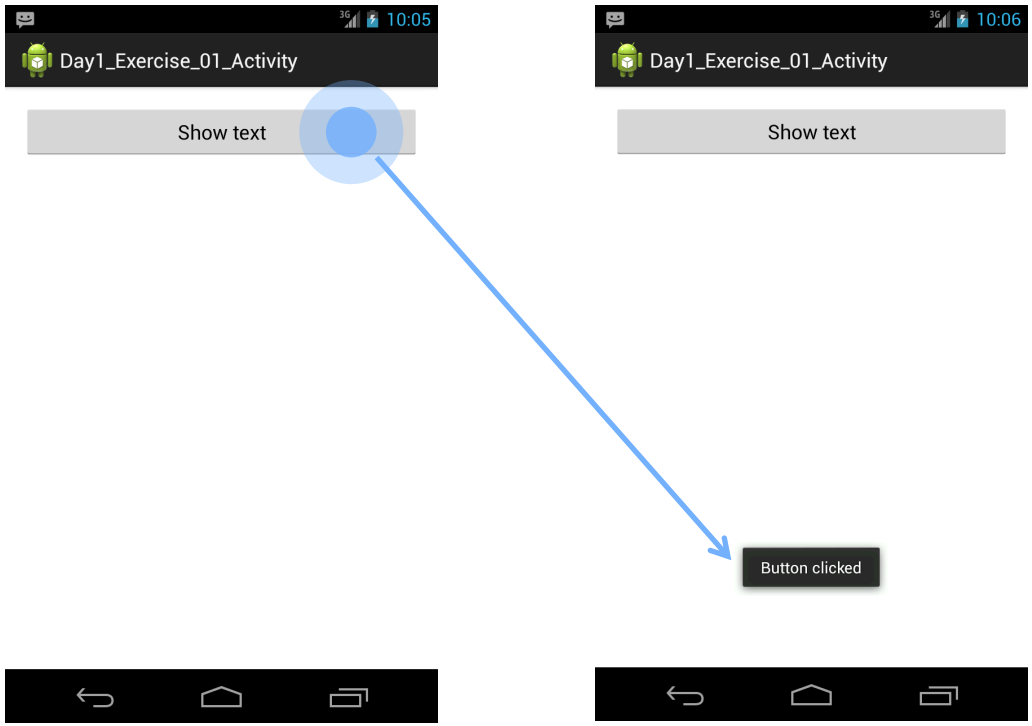
There are easier ways to bind the views in the activities, which are not covered in this class:

- Android Databinding (bind views bi-directional to a Java model)
- Butterknive (external library)

Activity – Special activities

- ListActivity
- FragmentActivity
- PreferenceActivity
- TabActivity
- AppCombatActivity replaced the *ActionBarActivity*
- *... Many more*

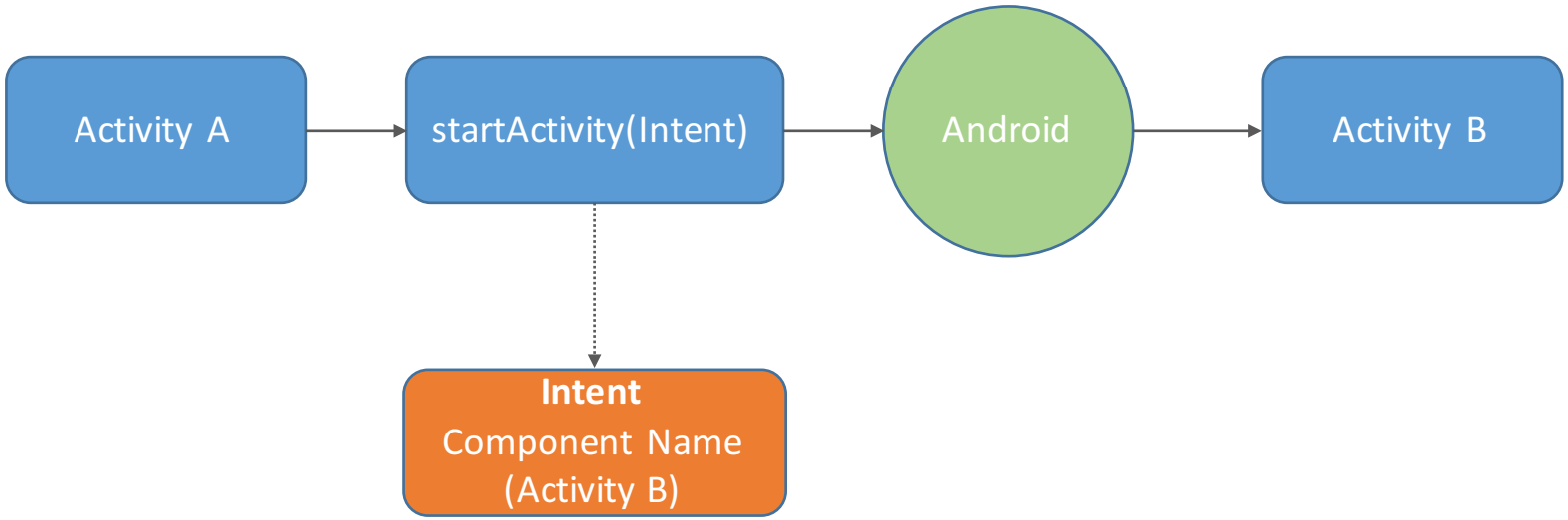
Activity – Exercise01_Activity



Intents - Overview

- An intent is a message to start another component (Not only activities)
- There are two types of Intents
  - **Explicit** (Specify the component to start by name (the fully qualified class name), typically used to start a component in your own app
  - **Implicit** (Do not name a specific component name, instead declare a general action to be performed, which allows a component from another app to handle it)
- An intent can contain data
  - Component names, actions, categories, extras, flags
- **The target component can be in another application!**

Intents - Explicit

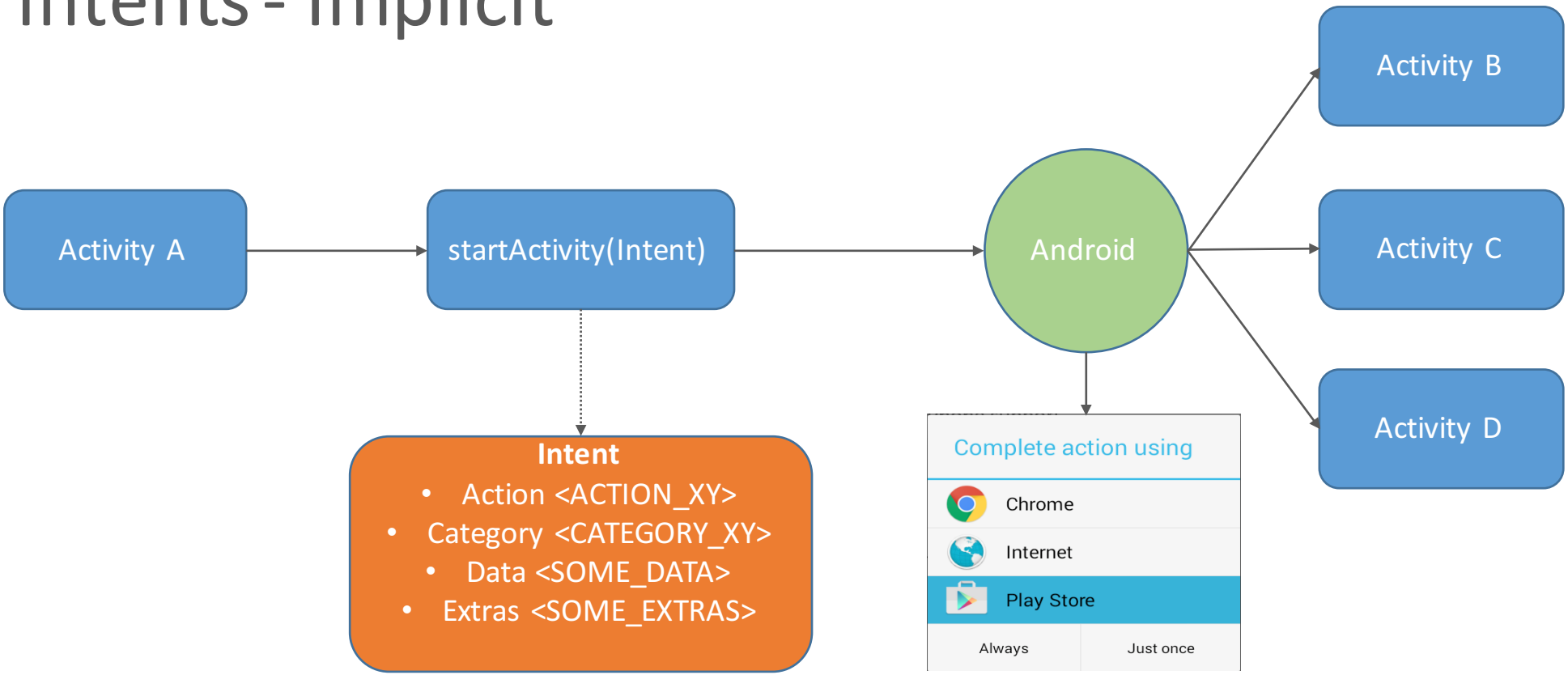


Intents – Explicit Example

```
// Intent erstellen (this = Context)
Intent intent = new Intent(this, TargetActivity.class);

// Daten in den Intent packen für die neue Activity
intent.putExtra("MyKey", "Die Daten");

// Neue Activity durch Android starten
startActivity(intent);
```

```
// In der onCreate()-Methode der TargetActivity:
// Intent in TargetActivity abfragen
Intent intent = getIntent();

// Extra Daten abfragen
String data = intent.getStringExtra("MyKey");
```

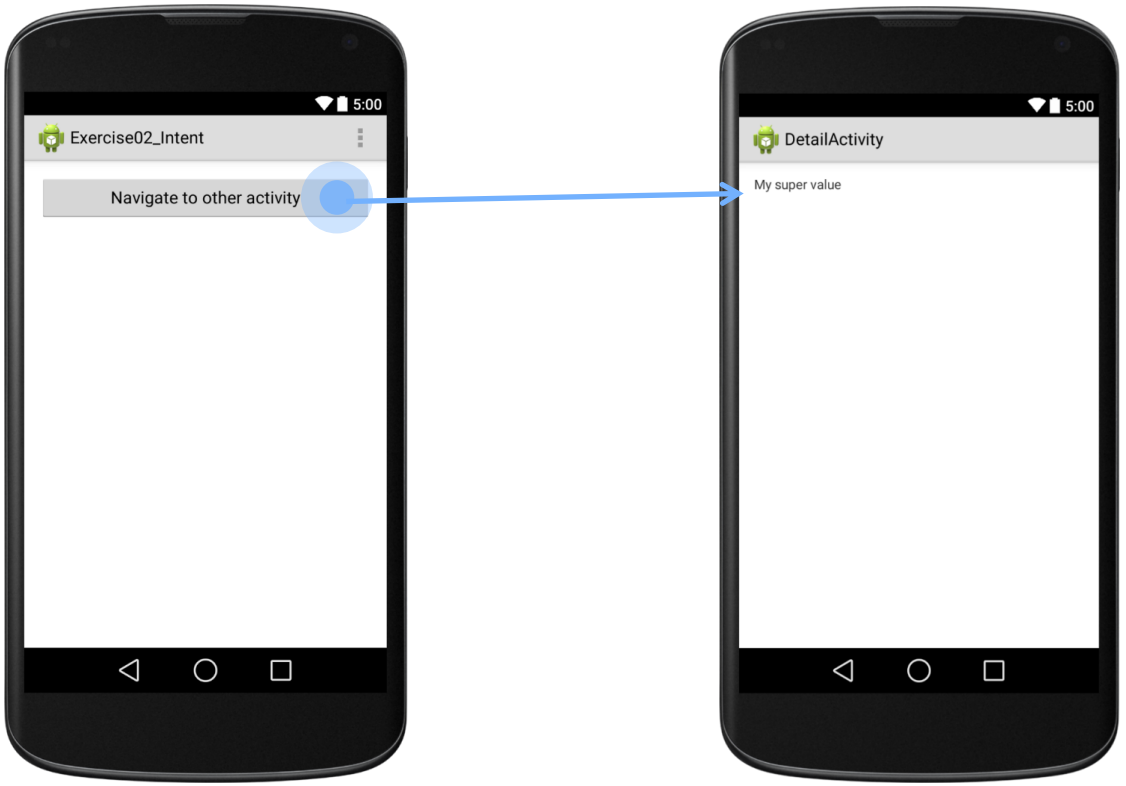
Intents - Implicit



Intents – Implicit Example

```
// Create a text message with a string
Intent intent = new Intent();
intent.setAction(Intent.ACTION_SEND);
intent.putExtea(Intent.EXTRA_TEXT,"My  awesome Text message");
intent.setType ("text/plain");
// Verify that the intent will resolve to a activity
if(intent.resolveActivity(getPackageManager())  != null){
        startActivity(intent);
}
```

Intents – Excerise02_Intent



Components - Task & Back Stack

- Task
  - A Task is a collection of activities that users interact with when performing a certain job.
- Back Stack
  - The activities are arranged in a stack («the back stack), in the order in which each activity is opened.
  - When apps are running simultaneously in a multi-window env. (Android 7.0, API Level 24) the system manages tasks separately for each window.

Components - Task & Back Stack Example