JAVA

Android

Overview

- a complete platform for mobile devices
  - based on Linux
- originally developed by Android, Inc. company
- 2005 – bought by Google
- 2007 – Open Handset Alliance
  - Google, HW and SW developing companies,...

- http://developer.android.com/
  - documentation
  - tutorials
  - tools
    - SDK – core tools
    - Android Studio – IDE, based on IntelliJ IDEA
  - ...

Java vs. Android

- ...is it Java or not...?
  yes and no
  - depends on “point of view”

- programs (primarily) developed in Java
- then it is compiled to byte-code (.class)
- the byte-code is copiled to Dalvik byte-code (.dex)
  - different one than Java byte-code
- this byte-code is executed by
  - Dalvik Virtual Machine <= Android 4.4
    - different one than the Java Virtual Machine
  - ART Virtual Machine   >= Android 5
    - different one than the Java Virtual Machine

Java vs. Android

- spring 2016 – change in Android N
  - Jack and Jill tool chain
  - direct compilation from Java to DEX

DEPRECATED
(spring 2017)

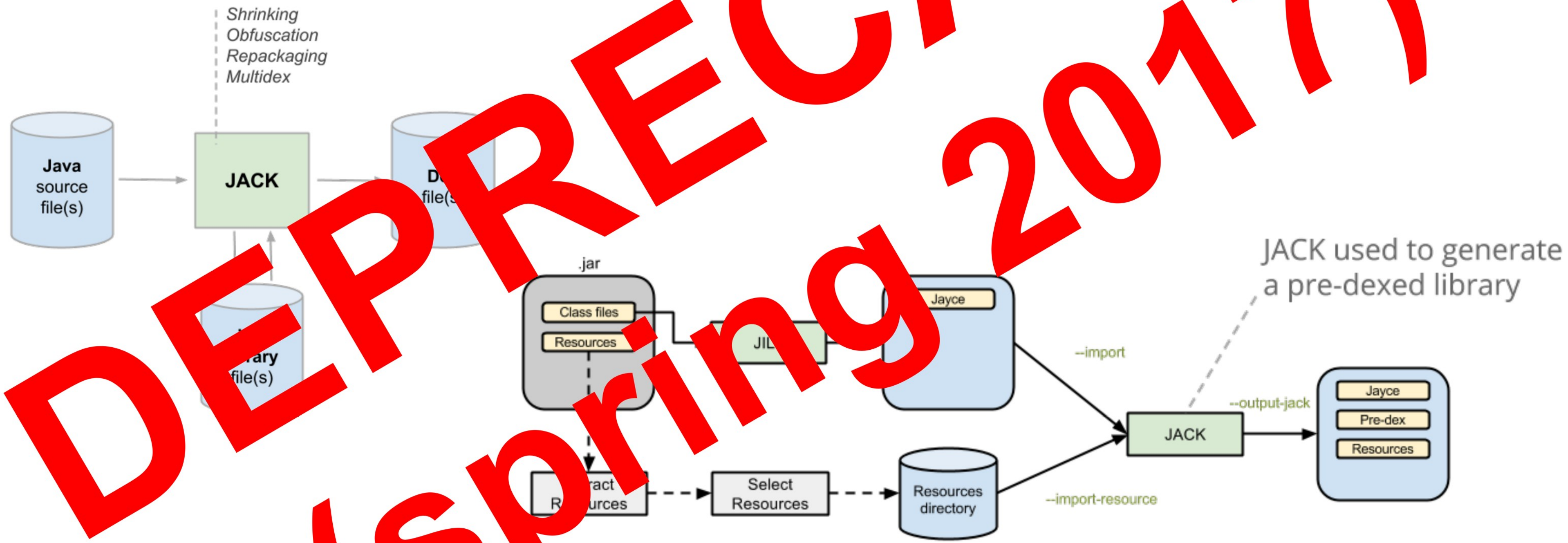
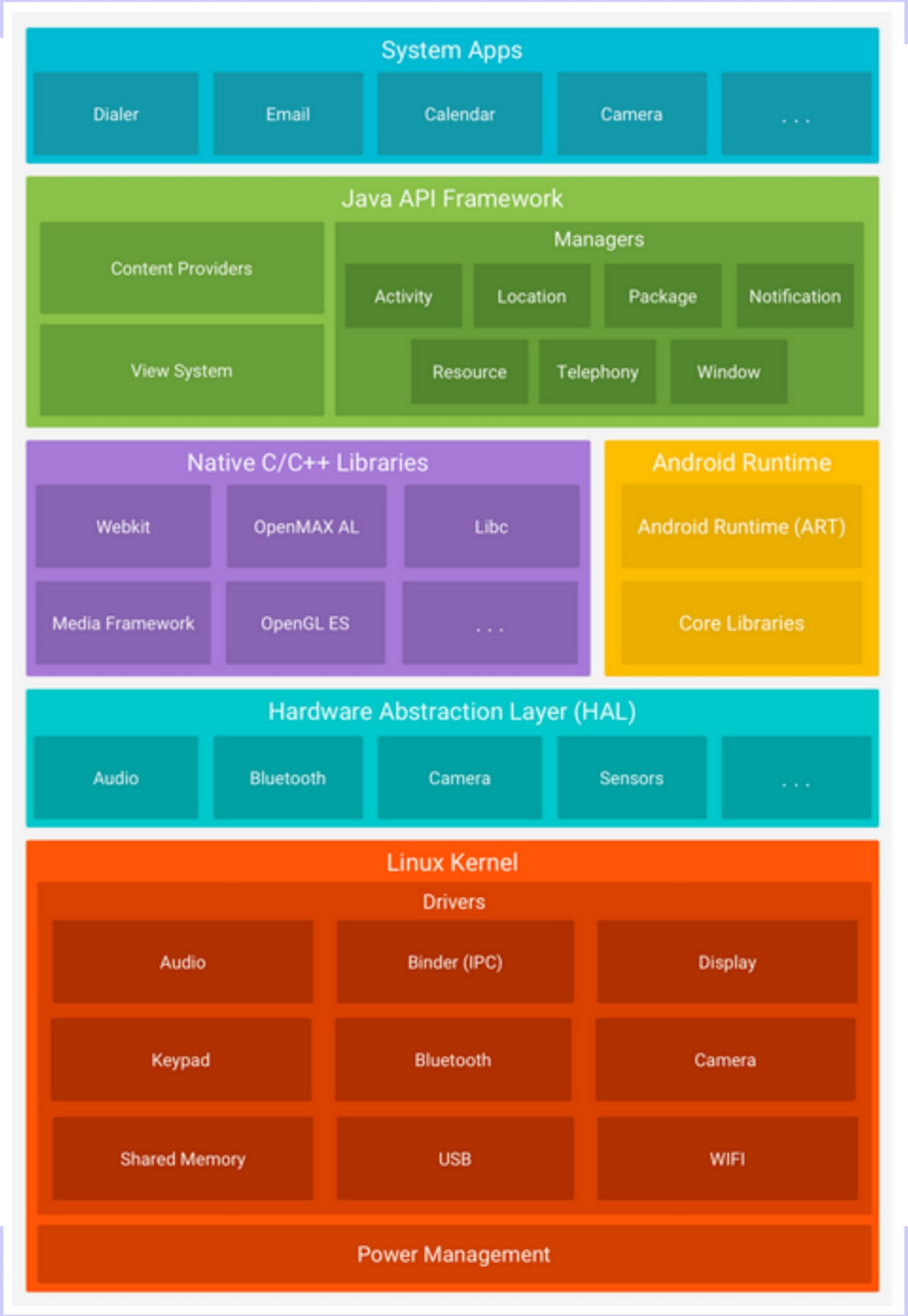


image source: https://source.android.com/source/jack.html

Java vs. Android

- used from Java
  - language
    - with the same syntax and semantics
  - part of API of std library

Platform structure



source: https://developer.android.com/guide/platform

Note: native applications

- programs can be written also in C/C++
  - it is not a primary way
  - it is necessary to download a separated NDK
    - SDK support only programs in “Java”
  - support of ARM, MIPS and x86 processors

Kotlin & Android

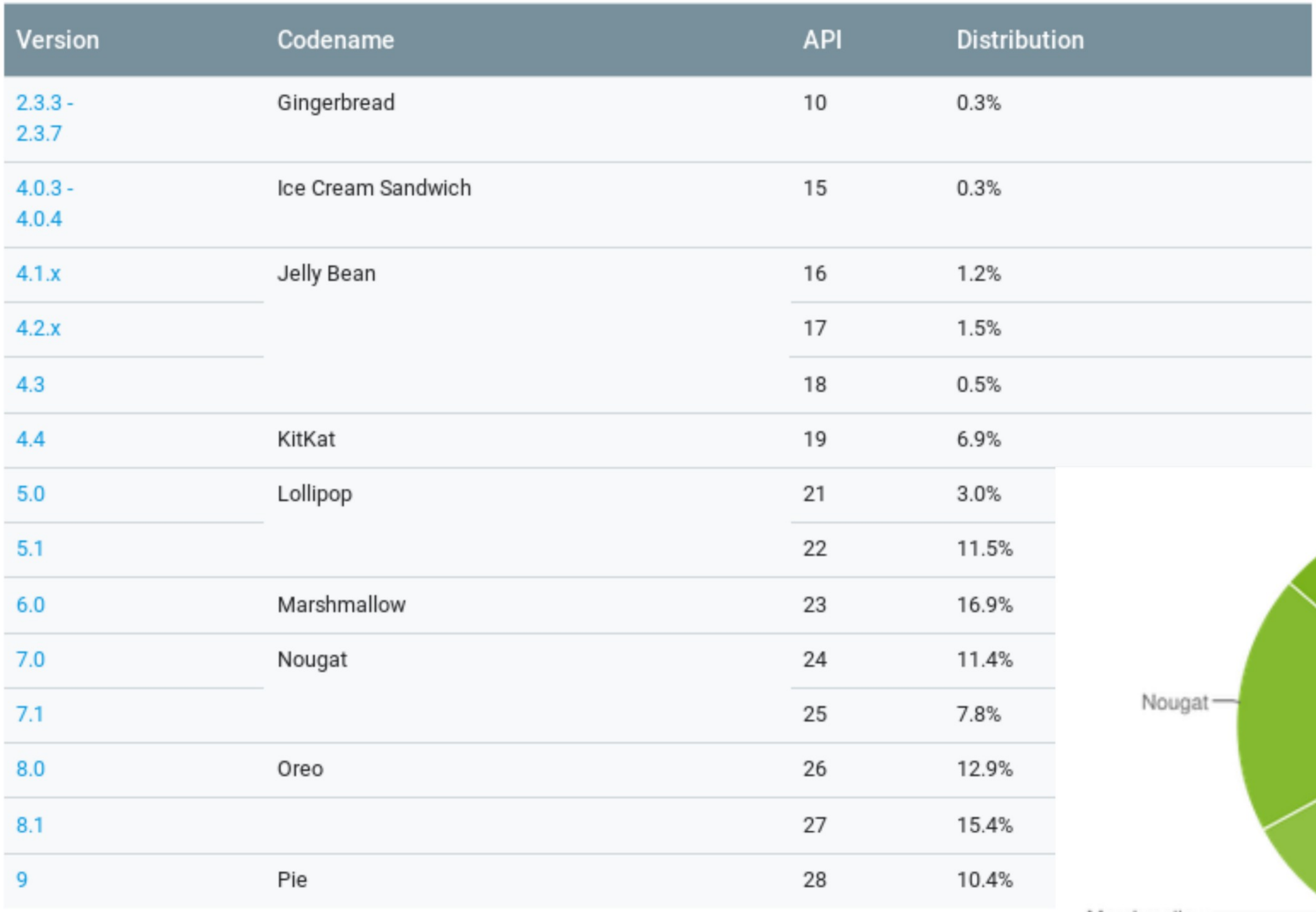
- Kotlin
  - statically typed programming language that runs on the Java virtual machine
  - developed by JetBrains
- 2nd official language for Android development
  - since May 2017

Problem – “fragmentation”

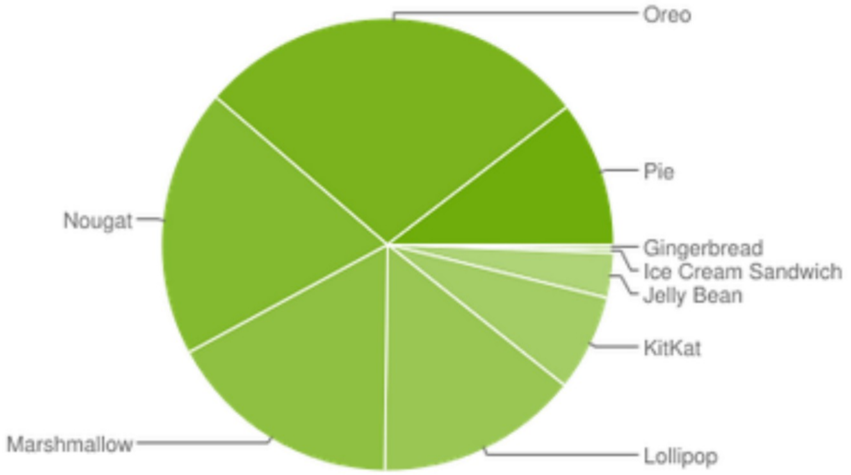
- both software and hardware

- software
  - many still used versions of the system
    - new API
    - deprecated API
    - different recommendation how to develop applications
- hardware
  - hundreds of different devices with Android with different features
    - display size, display density, (non)availability of sensors, (non)availability of HW buttons,...

Different versions of Android

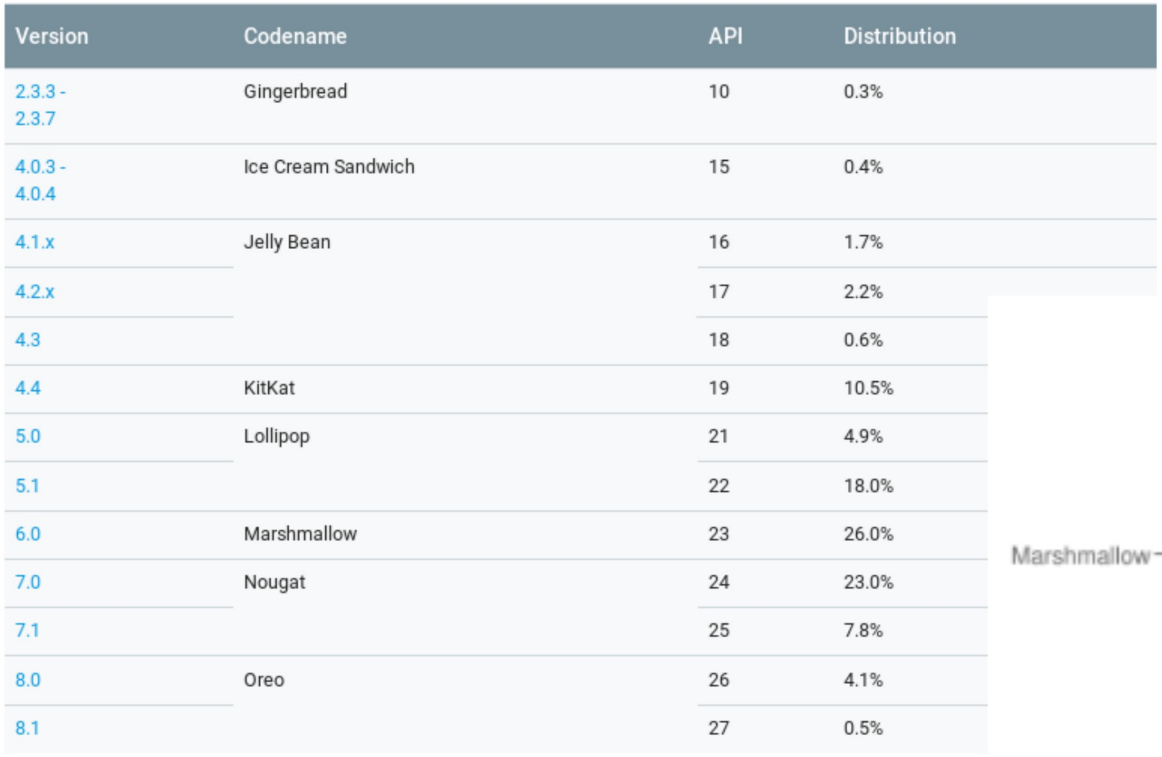


Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

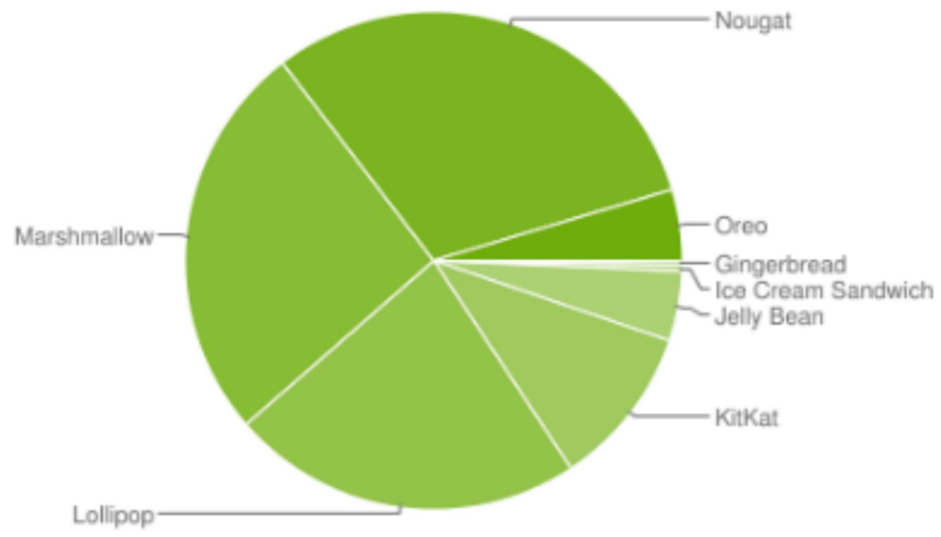


data for 7. 5. 2019
source: http://developer.android.com/about/dashboards/index.html

Different versions of Android (-1y)

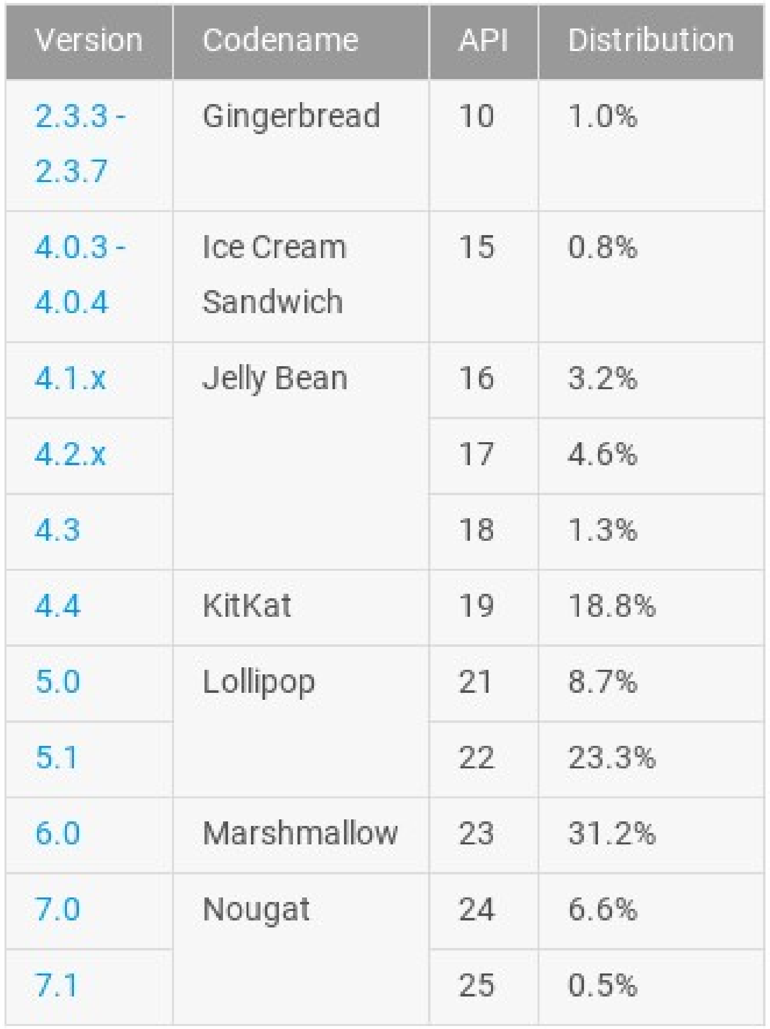


Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.5%
5.0	Lollipop	21	4.9%
5.1		22	18.0%
6.0	Marshmallow	23	26.0%
7.0	Nougat	24	23.0%
7.1		25	7.8%
8.0	Oreo	26	4.1%
8.1		27	0.5%

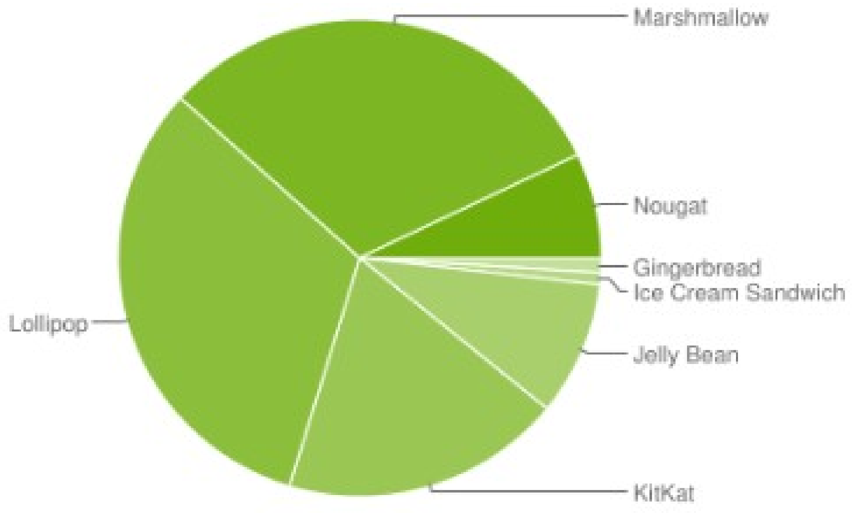


data for 16. 4. 2018
source: http://developer.android.com/about/dashboards/index.html

Different versions of Android (-2y)



Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.2%
4.2.x		17	4.6%
4.3		18	1.3%
4.4	KitKat	19	18.8%
5.0	Lollipop	21	8.7%
5.1		22	23.3%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	6.6%
7.1		25	0.5%



data for 2. 5. 2017
source: http://developer.android.com/about/dashboards/index.html

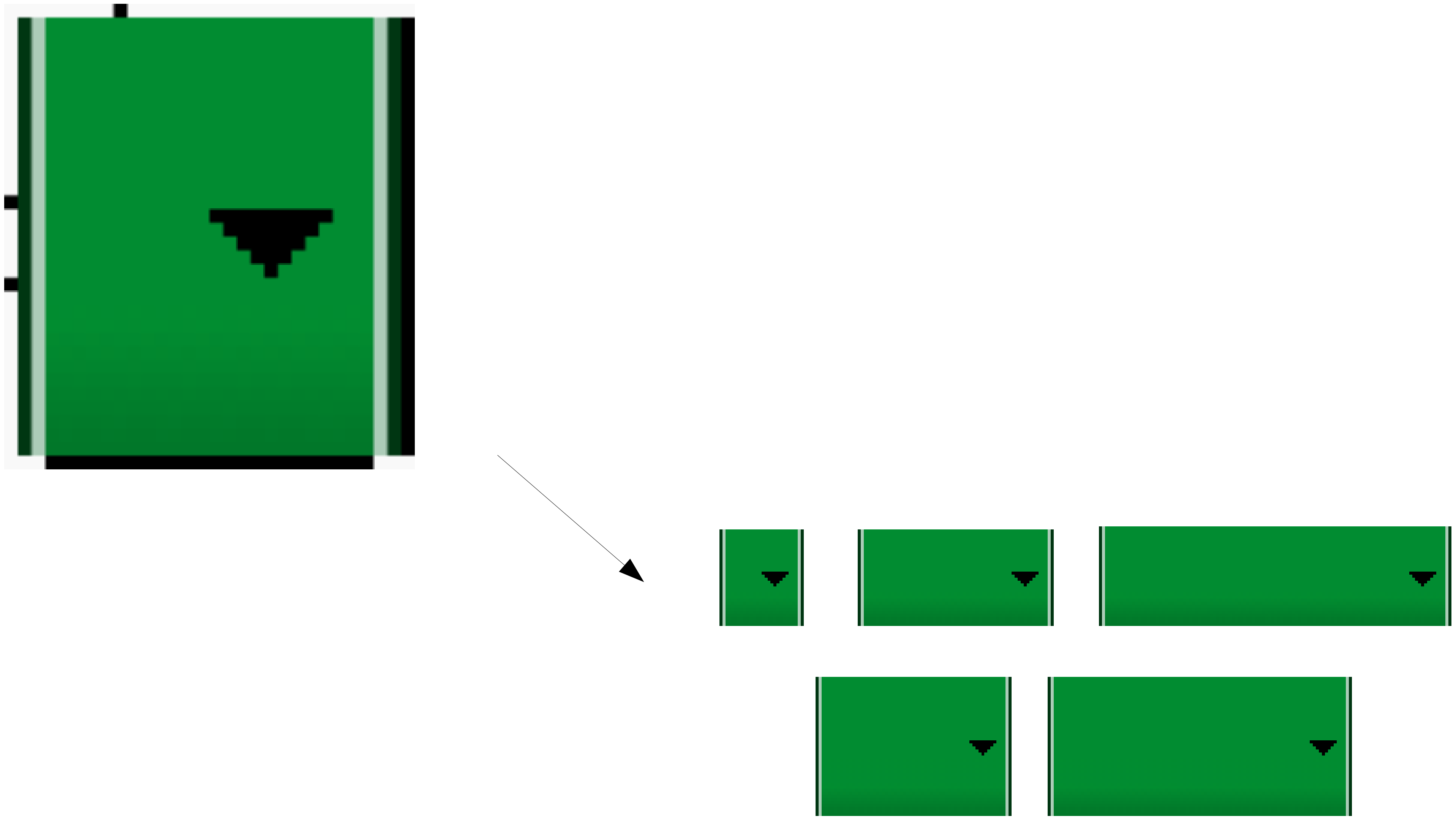
Different versions of Android

- minimum SDK version
  - application property (defined in the manifest)
  - the minimum API Level required for the application to run
    - cannot be installed on lower level devices
  - should be always specified
    - default value = 1
- target SDK version
  - the API Level that the application targets
  - the system should not enable any compatibility behaviors to maintain the app's forward-compatibility
  - default value = minSdkVersion
- maximum SDK version
  - should be used
    - new Android versions should be always backward-compatible

Different display size/density

- density-independent pixel
  - dp
  - 1dp = 160px/dpi
- images in multiple variants
  - according to size/density
    - will be discussed later
- 9-patch PNG
  - “stretchable” images
  - .9.png extension
  - a PNG image in which the borders have special meaning
    - left and top – where the image can be stretched
    - right and bottom – content border (e.g. button content)
  - creations – draw9patch tool in SDK

9-patch PNG



images source: http://developer.android.com/training/multiscreen/screensizes.html

Security

- applications run in “sandbox”
- by default, application are allowed to “almost” nothing
- permissions
  - specified in the manifest
  - during application installation, the system shows to a user all required permissions
    - the user has to confirm installation
  - permission examples
    - location (GPS)
    - bluetooth
    - phone function
    - SMS/MMS
    - net access
    - ...

Application structure

- Activities
  - UI components
  - application's entry points
- Views
  - UI elements
- Intents
  - asynchronous messages
- Services
  - long-running services in the background without UI
- Content providers
  - data providers for other applications
- Broadcast Intent Receivers
  - broadcast listeners (e.g. low battery level notifications)
- (HomeScreen) Widgets
  - interactive components on “desktop”

Project creation

- in IDE
  - New project...
- formerly also from the command-line
  - `android` tool
  - deprecated

Project creation

- project "parameters"
  - Application Name
    - human readable name
  - Package Name
    - "root" package serving as the application identifier
    - naming convention should be held
  - Target (min SDK version)
    - it is not directly the API level
    - command **`android list`**
      - a list of all supported targets

Project structure

- **`AndroidManifest.xml`**
- **`res/`**
- **`src/`**

Project structure

- AndroidManifest.xml
  - application description
    - components
    - requiremenets
    - ...

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <uses-sdk android:minSdkVersion="8"
                              android:targetSdkVersion="17" />
  <application android:icon="@drawable/app_icon.png" ... >
    <activity android:name="com.example.project.ExampleActivity"
       android:label="@string/example_label" ... >
    </activity>
    ...
  </application>
</manifest>
```

Project structure

- `res/` – resources
  - subdirectories in the directory `res`
    - drawable
      - images
      - ...
    - values
      - strings
      - …
    - layouts
      - screens
  - the R class
    - generated class
    - contains resource identifiers
      - as static fields
      - these are used in code

Project structure

- resources can have variants
  - specified by extension
  - drawable-hdpi, drawable-ldpi, drawable-mdpi
    - images for high, low and middle density of a display
  - other extension
    - land, port – display orientation
    - cs, en, fr, … – device language
    - small, normal, large – display size
    - ...
  - extensions can be combined
  - př:
    - res/values-de/
    - res/values-cs/
    - res/drawable-cs/
    - res/drawable-en-rUK/

Launching application

- in an emulator
  - IDE – Menu Tools-> AVD manager
- in a real device
  - attached via USB

- compilation
  - **`gradlew assembleDebug`**
- installation (to emulator/device)
  - **`adb install app/build/outputs/MyFirstApp-debug.apk`**

Activity

- extends **`android.app.Activity`**
- a window of the application
  - can serve as an entry point of the application
    - launcher
- its appearance typically described in an xml file
  - in res/layout

Hello World (1)

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       TextView tv = new TextView(this);
       tv.setText("Hello, Android");
       setContentView(tv);
   }
}
```

Hello World (2)

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
   }
}
```

Hello World (2)

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
xmlns:android="http://schemas.android.com/apk/res/and
roid"
  android:id="@+id/textview"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:text="@string/hello"/>
```

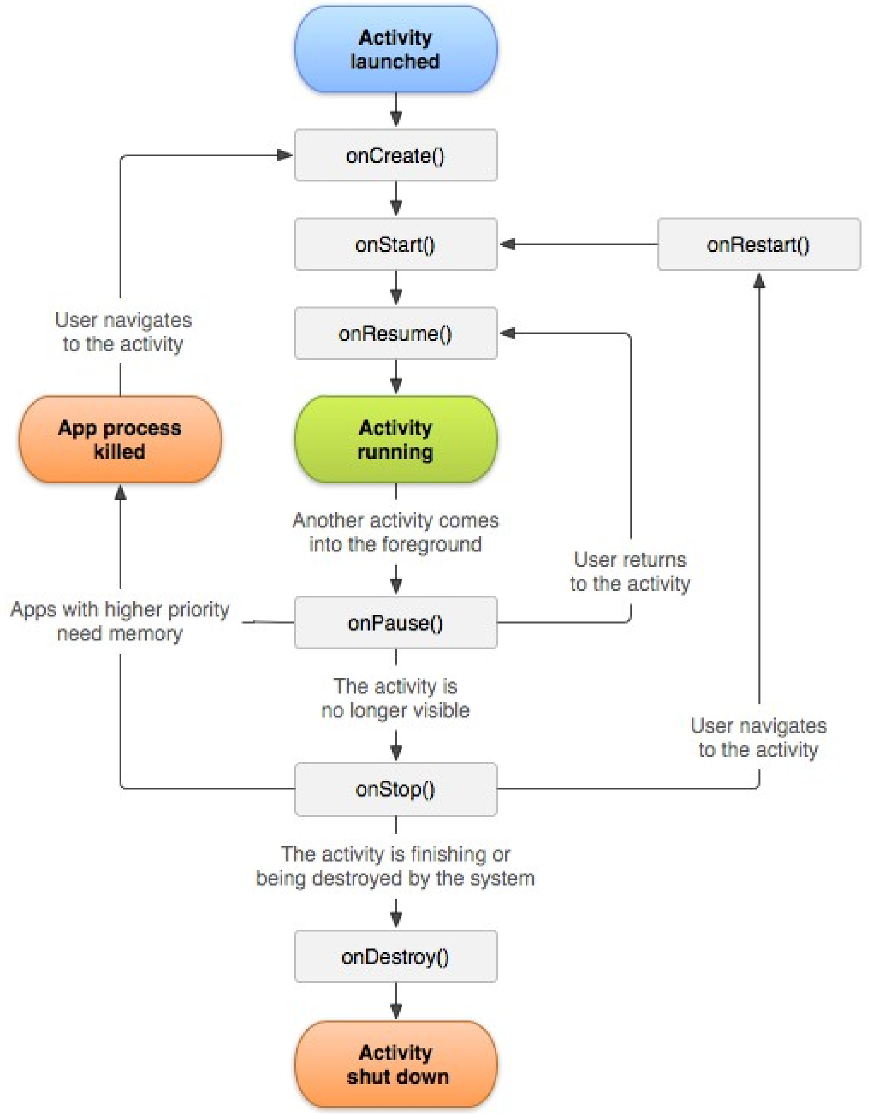
jednoznačné ID

reference

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, Android!
                  I am a string resource!</string>
    <string name="app_name">Hello, Android</string>
</resources>
```

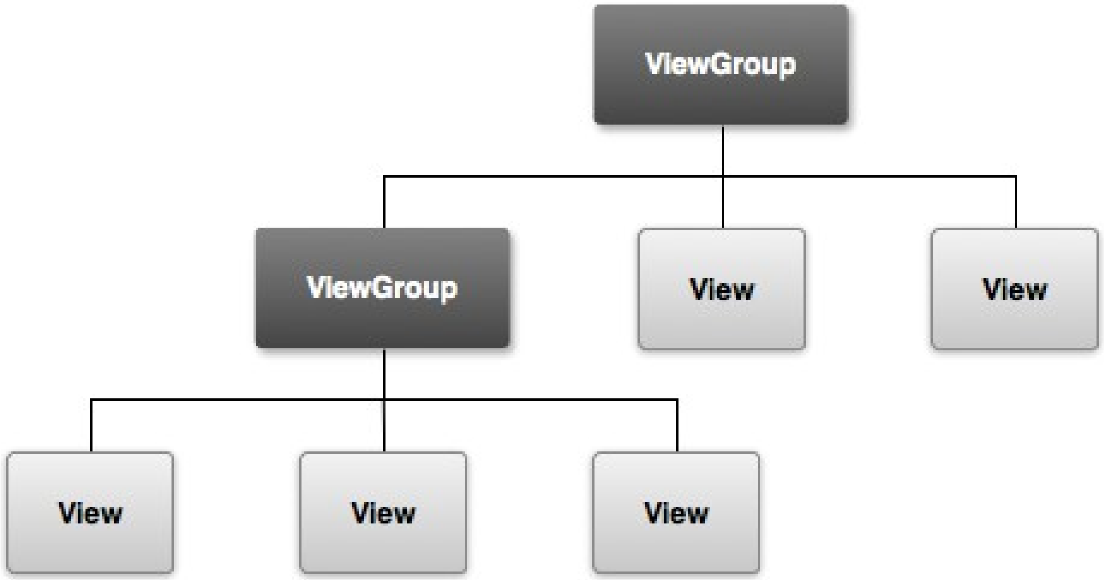
Activity lifecycle


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

source: https://developer.android.com/guide/components/activities/activity-lifecycle.html

UI

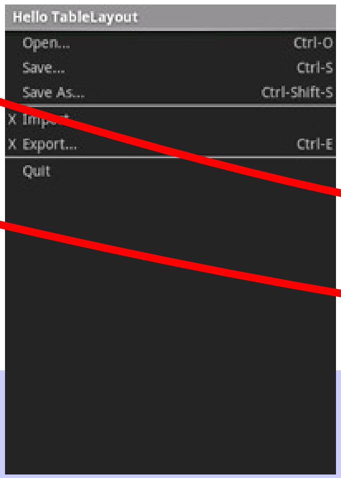
- similarly to Swing
- a hierarchy of objects
  - children of **`View`** and **`ViewGroup`**

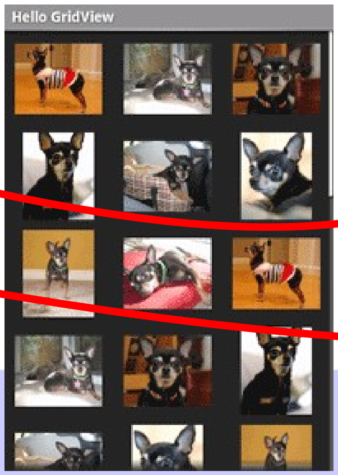


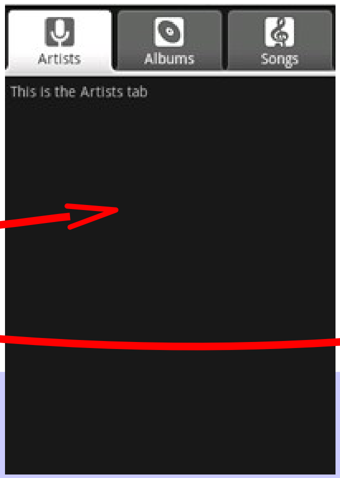
source: https://developer.android.com/guide/topics/ui/declaring-layout

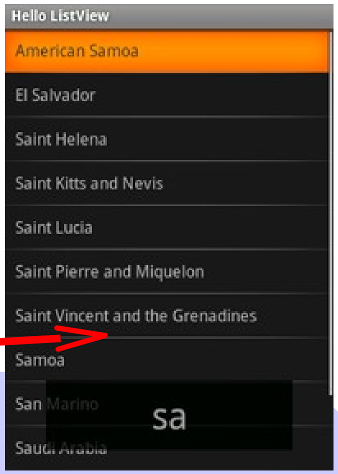
ViewGroup ~ Layout

- children of ViewGroup
- LinearLayout
  - places elements „in a row“
    - android:orientation="vertical"
    - android:orientation="horizontal"
- RelativeLayout
  - element placement relative to other elements
  - an example on the next slide
- TableLayout
- GridLayout
- TabLayout
- ListView



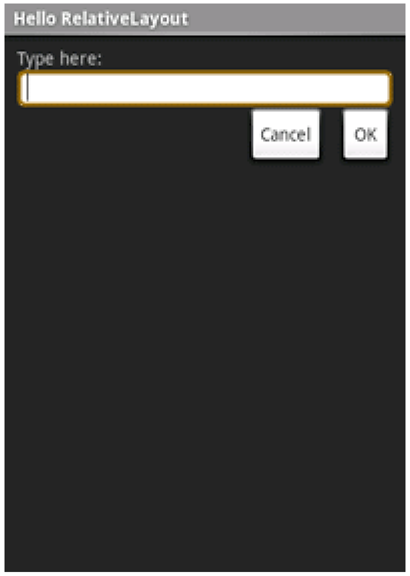






RelativeLayout example

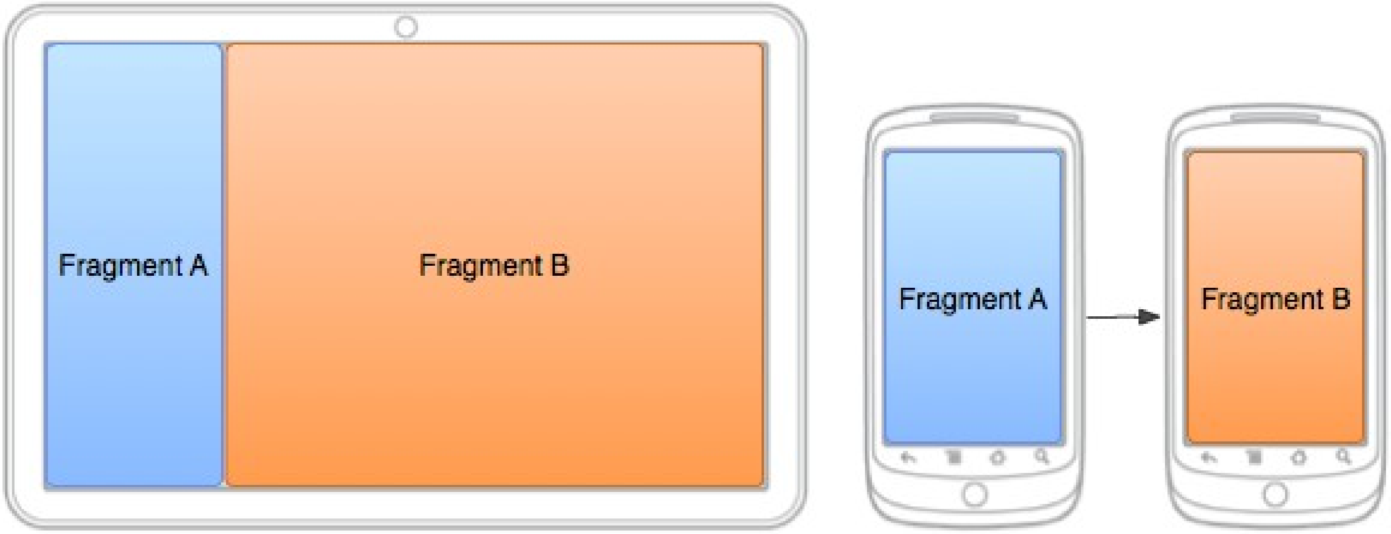
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label"/>
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancel" />
</RelativeLayout>
```



Fragments

- since Android 3.0
  - there is the "support library", which addes support for older versions (for the API level 4 and higher)
    - beware of the package
      `android.app.Fragment`
      `android.support.v4.app.Fragment`
- a reusable part of user interface
  ~ an "inner activity" with own layout and life-cycle
- an activity can show several fragments
- easy creation of UI for different types of display
  - phone
  - tablet

Using fragments



source: http://developer.android.com/training/basics/fragments/fragment-ui.html

Using fragmentů

- fragment

```
public class ArticleFragment extends Fragment {
  @Override
  public View onCreateView(LayoutInflater inflater,
           ViewGroup container, Bundle savedInstanceState) {
    return inflater.inflate(R.layout.article_view,
                                       container, false);
  }
}
```

- res/layout-large/news_articles.xml:

```
<LinearLayout xmlns:android="....."
  android:orientation="horizontal"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <fragment android:name="HeadlinesFragment"
            android:id="@+id/headlines_fragment"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
  <fragment android:name="ArticleFragment" ....  />
```

Using fragments

- activity

```
public class MainActivity extends FragmentActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.news_articles);
  }
}
```

- if the min API level is at least 11, the regular **`Activity`** can be used

Using fragments

- the previous example – fixed UI with two fragments suitable e.g. for a tablet
  - note the **`large`** extension of the layout

- for switching fragments (e.g. on a phone) it is necessary to manipulate fragments from code
- res/layout/news_articles.xml

```
<FrameLayout xmlns:android="..."
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

  - empty layout – content is added from code
  - without the **`large`** extension, i.e. for other display sizes

Using fragments

```
public class MainActivity extends FragmentActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.news_articles);
    if (findViewById(R.id.fragment_container) != null) {
      if (savedInstanceState != null) {
        return;
      }
      HeadlinesFragment firstFragment = new HeadlinesFragment();
      firstFragment.setArguments(getIntent().getExtras());
      getSupportFragmentManager().beginTransaction()
          .add(R.id.fragment_container, firstFragment).commit();
    }
  }
}
```

Using fragments

- replacing the shown fragment

```
ArticleFragment newFragment = new ArticleFragment();
FragmentTransaction transaction =
        getSupportFragmentManager().beginTransaction();
transaction.replace(R.id.fragment_container,
                                          newFragment);
transaction.addToBackStack(null);
transaction.commit();
```

Intents

- application components (activities, services, broadcast receivers) are activates by Intents
  - “messages”
  - Intent – a passive object
    - extends android.content.Intent
    - properties
      - component name
      - action
        - string
          - many predefined
          - own ones can be created
      - data
        - URI of data to work with
      - category
        - other information about component type, which should react to the intent
      - extras
      - flags

Intents

- explicit
  - with a name of the target component
  - typically used inside an application
- implicit
  - without a component name
  - typically communication between applications

- intent filters
  - which intents the component can serve
  - declared in the manifest

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Intents

- a permission to react to an intent can be set
  - declared in the manifest
  - confirmed during installation
- also the “system” applications react to intents
  - > own “system” applications can be developed
    - Mailer, SMS app, Homepage,...

Intents – example

```
private static final int ACTIVITY_PICK_CONTACT = 42;
private void pickContact() {
  Intent intent = new Intent(Intent.ACTION_PICK,
                        ContactsContract.Contacts.CONTENT_URI);
  startActivityForResult(intent, ACTIVITY_PICK_CONTACT);
}


@Override
protected void onActivityResult(int requestCode, int resultCode,
                                                 Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  switch (requestCode) {
    case (ACTIVITY_PICK_CONTACT) :
      if (resultCode == Activity.RESULT_OK) {
        Uri pickedContact = data.getData();
        return;
      }
    break;
  }
}
```

Task

- a stack of launched activities
  - an activity reacts to an intent = a new instance is created and put to a stack
- a user communicates with an activity on the top
- several parallel tasks can exist

- task ~ running application

Services

- background running services
- potomci od android.app.Service
  - they do not automatically start their thread!

- IntentService
  - extends Service
  - intended for services reacting to intents
  - they contain thread management
  - it is enough to override void onHandleIntent(Intent intent)

Threads

- activities of an application are run in one thread
- events are also served in this thread
  - “main” thread / UI thread
- similarly as in Swing

- UI is not “thread-safe”
  - manipulations with UI perform in the “main” thread
  - the “main” thread should not be blocked

- helper methods
  - Activity.runOnUiThread(Runnable)
  - View.post(Runnable)
  - View.postDelayed(Runnable, long)
- AsyncTask
  - similar to SwingWorker

Dialogs

```
public class ADialogFragment extends DialogFragment {
  @Override
  public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder =
                     new AlertDialog.Builder(getActivity());
    builder.setMessage("message")
           .setPositiveButton("OK",
              new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog,
                                                   int id) {
                 . . .
               }
             })
            .setNegativeButton("Cancel",
               new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog,
                                                   int id) {
                 . . .
               }
             });
      return builder.create();   }   }
```

Dialogs

- showing a dialog

```
ADialogFragment aDialog =
                       new ContactDialogFragment();
aDialog.show(getFragmentManager(), "dialog");
```

Dialogs – deprecated way

```
@Override
protected Dialog onCreateDialog(int id) {
  switch (id) {
    case DIALOG_SHOW_CONTACT: {
      return new AlertDialog.Builder(this).setTitle("XXX").
         setMessage("Message").setCancelable(true).
                     setPositiveButton("OK", null).create();
    }
  }
  return null;
}

@Override
protected void onPrepareDialog(int id, Dialog dialog) {
  switch (id) {
    case DIALOG_SHOW_CONTACT: {
      if (pickedContact != null) {
       ((AlertDialog) dialog).setMessage("YYY"));
      }
}}}
```

called just once

“user-defined” constant

called before each showing

Dialogs – deprecated way

- showDialog(DIALOG_SHOW_CONTACT);
  - showing a dialog