

# JAVA

## Android

# Přehled

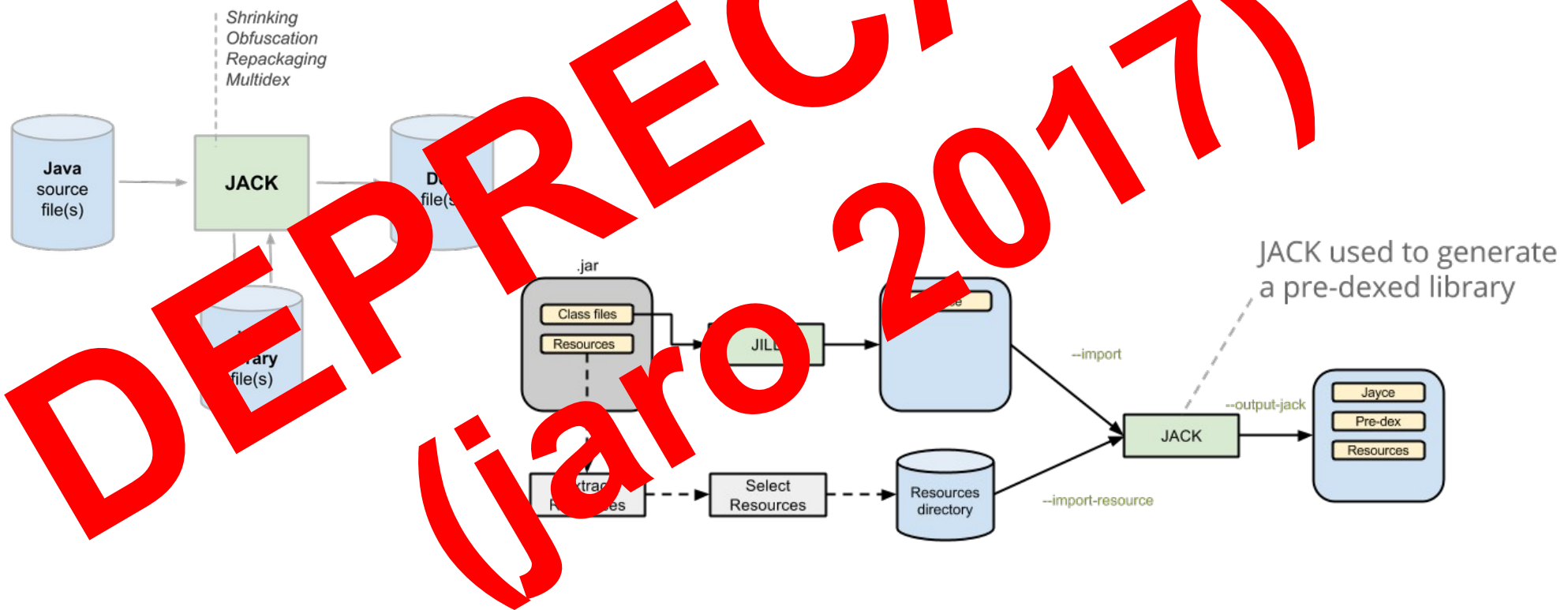
- kompletní platforma pro mobilní zařízení
  - založená na Linuxu
- původně vyvíjen firmou Android, Inc.
- 2005 – koupeno Googlem
- 2007 – Open Handset Alliance
  - Google, výrobci HW, výrobci SW,...
- <http://developer.android.com/>
  - dokumentace
  - tutoriály
  - nástroje
    - SDK – základní nástroje
    - Android Studio – IDE, založeno na IntelliJ IDEA
  - ...

# Java vs. Android

- ...je to Java nebo ne...?
  - ano i ne
    - záleží na „úhlu pohledu“
- programy se (primárně) píší v Javě
- pak se přeloží do byte-kódu (.class)
- ten se přeloží do Dalvik byte-kódu (.dex)
  - jiný než Java byte-kód
- ten se vykonává na
  - Dalvik Virtual Machine  $\leq$  Android 4.4
    - jiná než Java Virtual Machine
  - ART Virtual Machine  $\geq$  Android 5
    - jiná než Java Virtual Machine

# Java vs. Android

- jaro 2016 – změna od Android N
  - Jack and Jill tool chain
  - přímá kompilace z Java do DEX

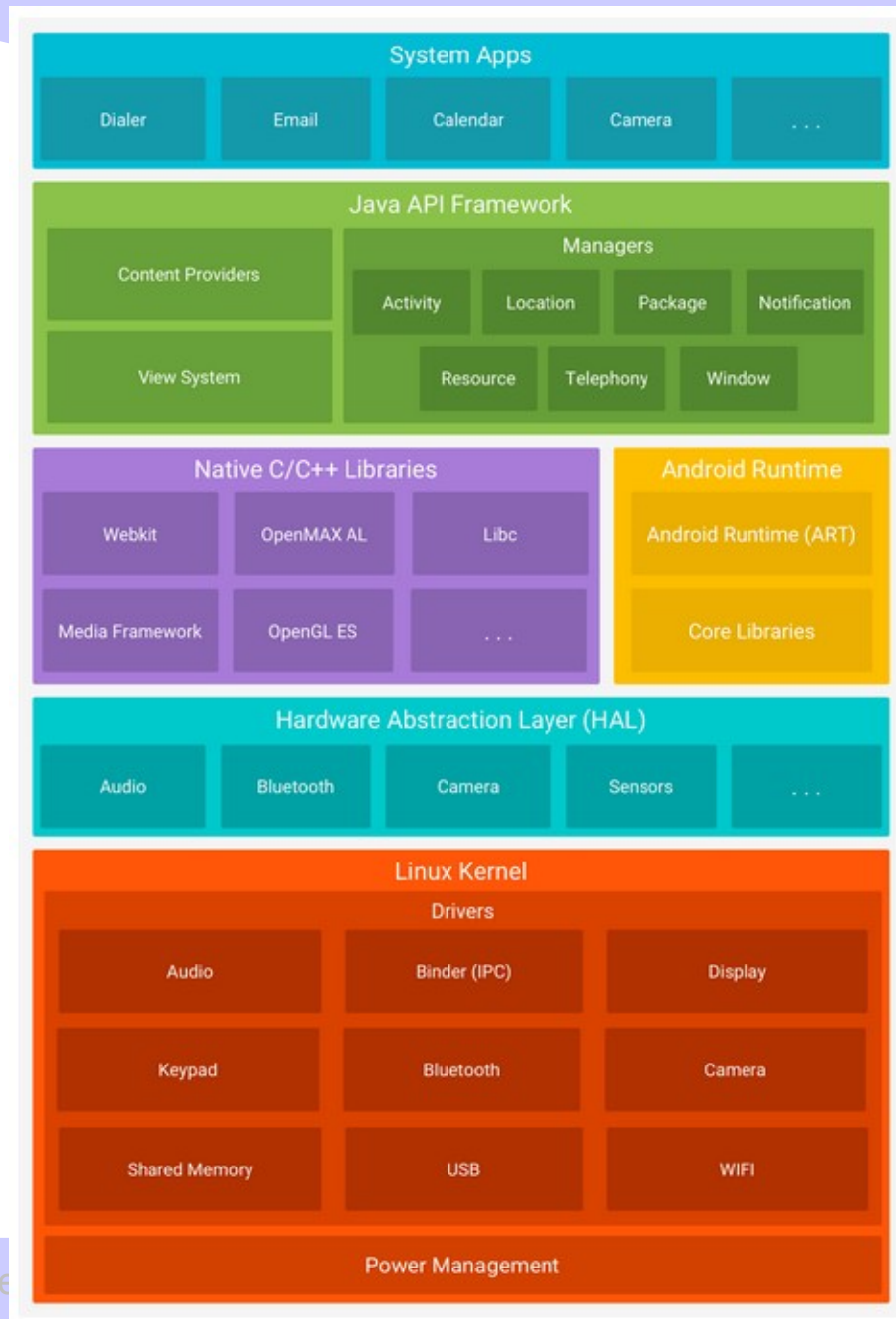


zdroj obrázků: <https://source.android.com/source/jack.html>

# Java vs. Android

- z Javy se používá
  - jazyk
    - se stejnou syntaxí a sémantikou
  - část API ze standardní knihovny

# Struktura platformy



zdroj: <https://developer.android.com/guide/platform>

# Odbočka: nativní aplikace

- programy lze psát i v C/C++
  - není to primární způsob
  - nutno stáhnout oddělené NDK
    - SDK podporuje jen programy v „Javě“
  - podpora ARM, MIPS a x86 procesorů

# Kotlin & Android

- Kotlin
  - staticky typovaný jazyk běžící na Java virtual machine
  - vyvíjený JetBrains
- druhý oficiální jazyk pro vývoj pro Android
  - od května 2017



# Problém – „roztříštěnost“

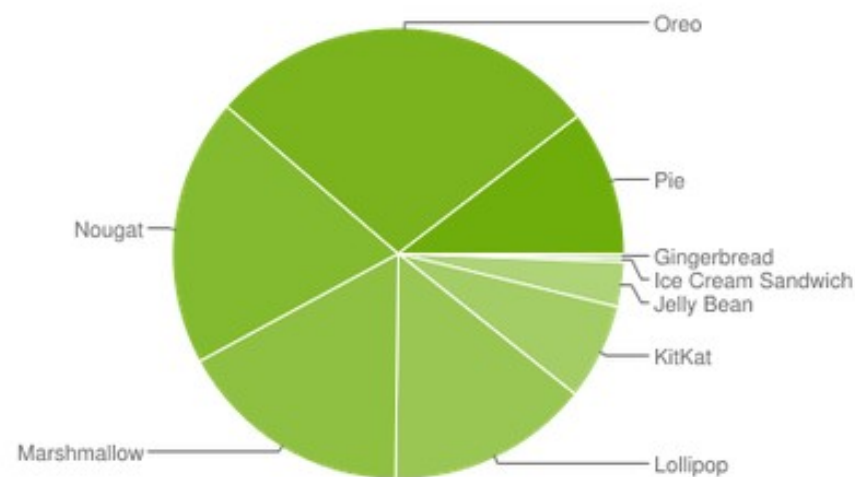
- softwarová i hardwarová
- softwarová
  - mnoho používaných verzí systému
    - nová API
    - deprecated API
    - různá doporučení, jak vyvíjet aplikace
- hardwarová
  - stovky různých zařízení s Androidem s různými vlastnostmi
    - velikost displeje, hustota displeje, (ne)přítomnost senzorů, (ne)přítomnost HW tlačítek,.....

# Různé verze Androidu

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,8%
4.2 Jelly Bean	17	99,2%
4.3 Jelly Bean	18	98,4%
4.4 KitKat	19	98,1%
5.0 Lollipop	21	94,1%
5.1 Lollipop	22	92,3%
6.0 Marshmallow	23	84,9%
7.0 Nougat	24	73,7%
7.1 Nougat	25	66,2%
8.0 Oreo	26	60,8%
8.1 Oreo	27	53,5%
9.0 Pie	28	39,5%
10. Android 10	29	8,2%

# Různé verze Androidu (-1 rok)

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	Pie	27	15.4%
9		28	10.4%

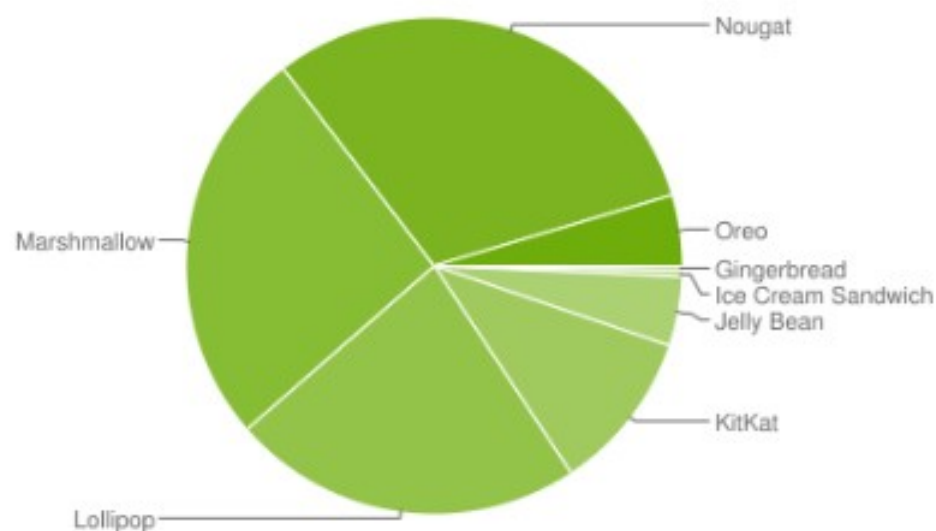


data k 7. 5. 2019

zdroj: <http://developer.android.com/about/dashboards/index.html>

# Různé verze Androidu (-2 rok)

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 k 16. 4. 2018

zdroj: <http://developer.android.com/about/dashboards/index.html>

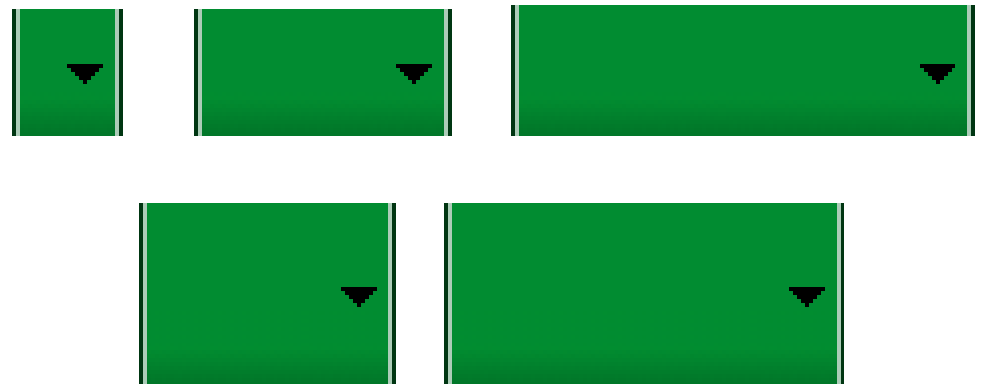
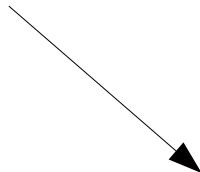
# Různé verze Androidu

- minimum SDK version
  - vlastnost aplikace (definovaná v manifestu)
  - minimální API level požadovaný, aby aplikace běžela
    - na nižší nepůjde nainstalovat
  - vždy by se měla specifikovat
    - implicitní hodnota = 1
- target SDK version
  - API level vůči kterému byla aplikace vyvíjena
  - systém nebude aplikovat žádné změny chování pro zachování kompatibility
  - implicitní hodnota = minSdkVersion
- maximum SDK version
  - neměla by se specifikovat
    - nové verze Androidu by měly být vždy zpětně kompatibilní

# Různá velikost/rozlišení displeje

- density-independent pixel
  - dp
  - $1\text{dp} = 160\text{px}/\text{dpi}$
- obrázky a „obrazovky“ v různých variantách
  - podle velikosti/hustoty
    - bude zmíněno později
- 9-patch PNG
  - „roztážitelné“ obrázky
  - přípona .9.png
  - PNG obrázek ve kterém okraje mají speciální význam
    - levý a horní okraj – kde se obrázek může roztahovat
    - pravý a dolní okraj – okraj obsahu (např. vnitřek tlačítka)
  - výroba – draw9patch program v SDK

# 9-patch PNG



zdroj obrázků: <http://developer.android.com/training/multiscreen/screensizes.html>

# Bezpečnost

- aplikace běží ve „sandboxu“
- implicitně aplikace „skoro“ nic nesmí
- oprávnění (permissions)
  - specifikována v manifestu
  - při instalaci aplikace systém oznámí uživateli všechna požadovaná oprávnění
  - příklady oprávnění
    - lokace (GPS)
    - bluetooth
    - telefonování
    - SMS/MMS
    - přístup k síti
    - ...



# Struktura aplikace

- Activities
  - komponenty UI
  - vstupní bodu do aplikace
- Views
  - elementy uživatelského rozhraní
- Intents
  - asynchronní zprávy
- Services
  - služby bez UI běžící dlouhodobě na pozadí
- Content providers
  - zpřístupnění dat jiným aplikacím
- Broadcast Intent Receivers
  - poslouchání na broadcasty (např. oznámení o nízkém stavu baterie)
- (HomeScreen) Widgets
  - interaktivní komponenty na „ploše“

# Vytvoření projektu

- z IDE
  - New project...
- dříve i z příkazové řádky
  - nástroj **android**
  - deprecated

# Vytvoření projektu

- „parametry“ projektu
  - Application Name
    - lidsky čitelné jméno
  - Package Name
    - „kořenový“ balíček, slouží i jako identifikátor aplikace
    - nutno dodržovat konvenci pro pojmenování
  - Target (min SDK version)
    - není to přímo API level
    - příkaz **android list**
      - seznam všech dostupných targetů

# Struktura projektu

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

# Struktura projektu

- AndroidManifest.xml
  - popis aplikace
    - komponenty
    - požadavky
    - ...

```
<?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>
```

# Struktura projektu

- **res/** – zdroje (resources)
  - typ podadresáře v adresáři **res**
    - drawable
      - obrázky
      - ...
    - values
      - řetězce
      - ...
    - layouts
      - obrazovky
  - třída R
    - generovaná třída
    - obsahuje identifikátory zdrojů
      - jako statické atributy
      - používají se v kódu

# Struktura projektu

- zdroje mohou mít varianty
  - určují se podle přípon
    - drawable-hdpi, drawable-ldpi, drawable-mdpi
      - obrázky pro vysoké, nízké, střední rozlišení displaye
    - další přípony
      - land, port – orientace displaye
      - cs, en, fr, ... – jazyk zařízení
      - small, normal, large – velikost displaye
      - ...
    - přípony lze kombinovat
    - př:
      - res/values-de/
      - res/values-cs/
      - res/drawable-cs/
      - res/drawable-en-rUK/

# Spuštění aplikace

- v emulátor
  - IDE – Menu Tools-> AVD manager
- na skutečném zařízení
  - připojeném přes USB
  
- přeložení
  - `gradlew assembleDebug`
- nainstalování (do emulátoru/na zařízení)
  - `adb install`  
`app/build/outputs/MyFirstApp-debug.apk`



# Aktivity

- potomek `android.app.Activity`
- okno aplikace
  - může sloužit i jako vstupní bod do aplikace
    - launcher
- vzhled se typicky popisuje jako xml soubor
  - v `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"/>
```

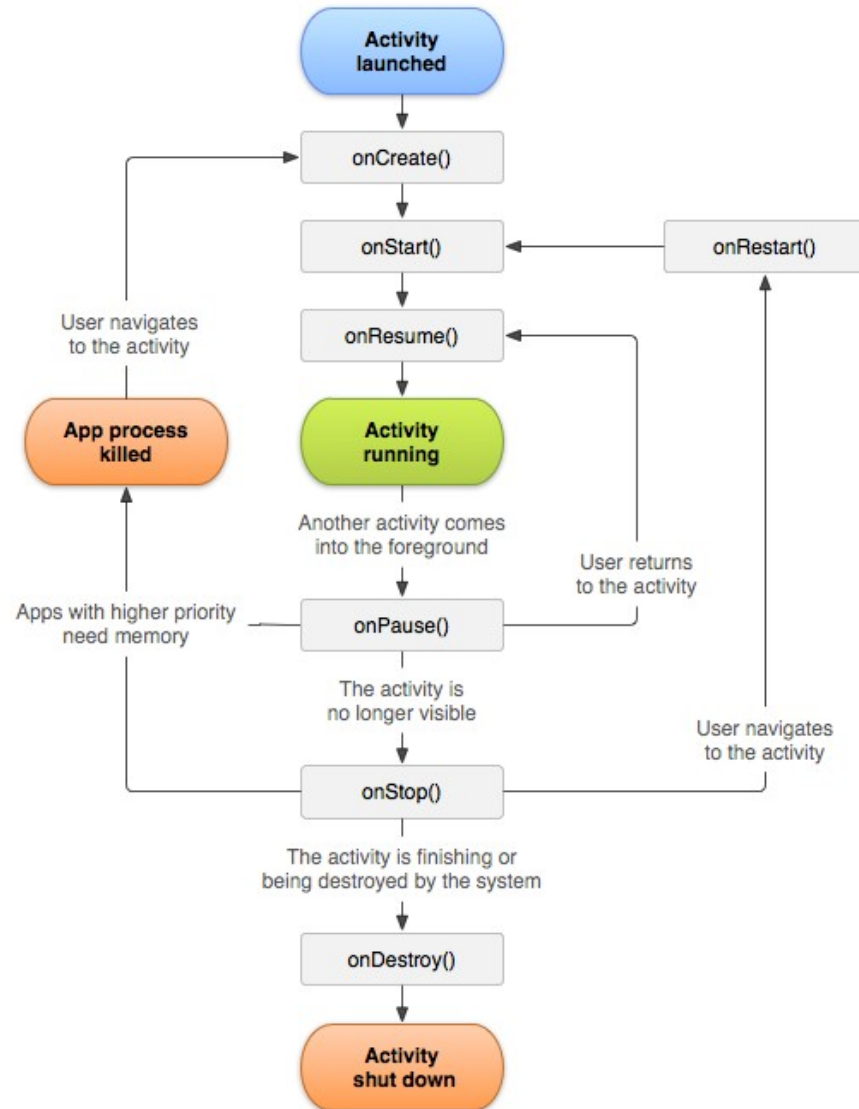
jednoznačné ID

reference

## res/values/strings.xml

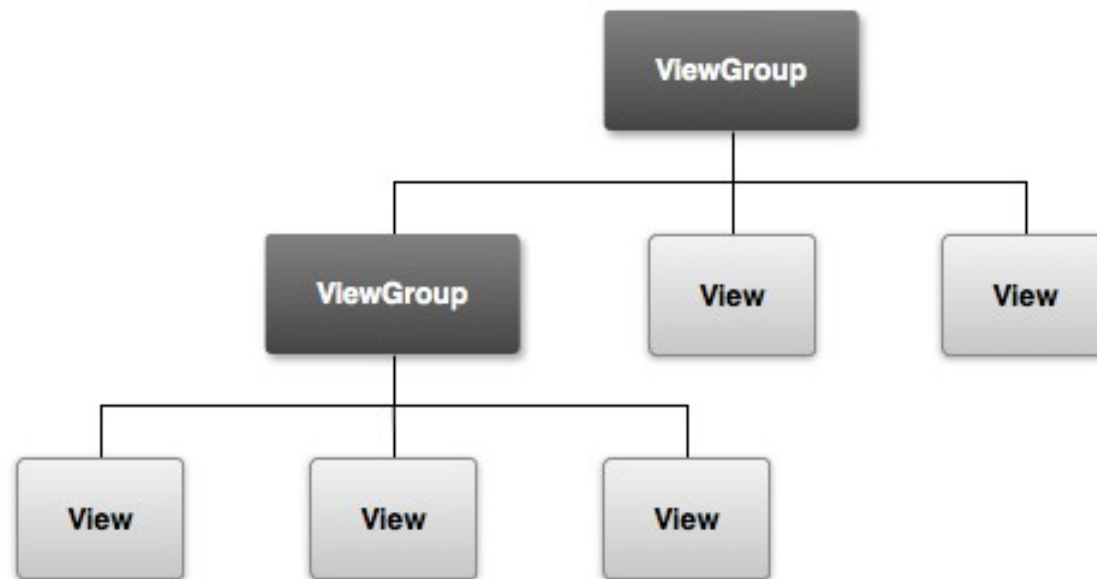
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, Android!
        I am a string resource!</string>
    <string name="app_name">Hello, Android</string>
</resources>
```

# Životní cyklus aktivity



zdroj: <https://developer.android.com/guide/components/activities/activity-lifecycle.html>

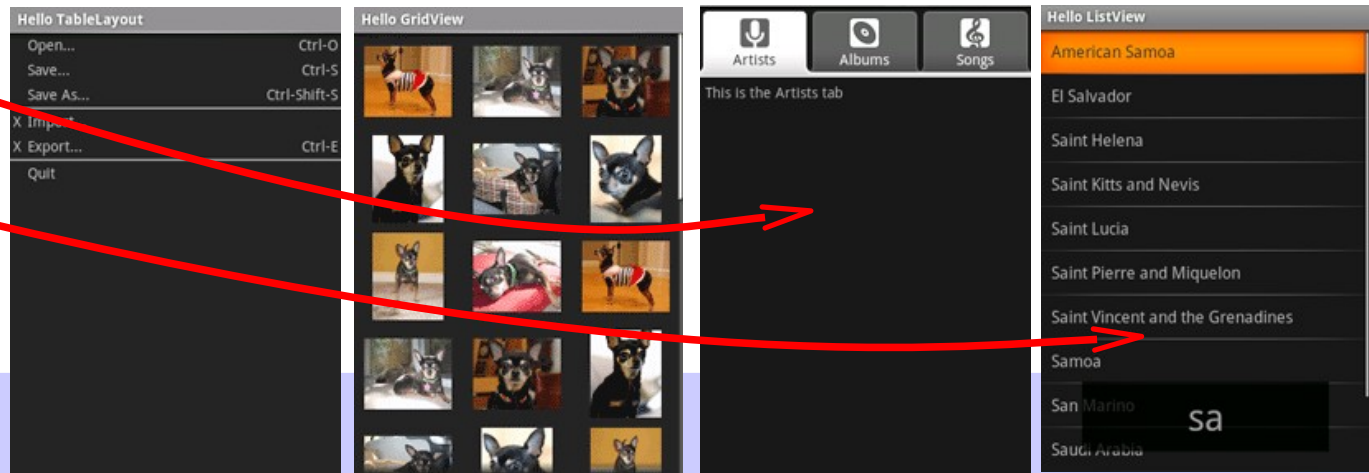
- podobně jako ve Swingu
- hierarchie objektů
  - potomci **View** a **ViewGroup**



zdroj: <https://developer.android.com/guide/topics/ui/declaring-layout>

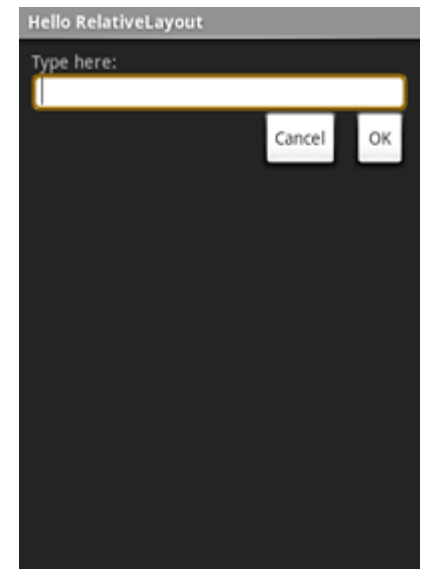
# ViewGroup ~ Layout

- potomci ViewGroup
- LinearLayout
  - skládá prvky „do řady“
    - `android:orientation="vertical"`
    - `android:orientation="horizontal"`
- RelativeLayout
  - určování polohy relativně vůči dalším prvkům
  - příklad na dalším slidu
- TableLayout
- GridLayout
- TabLayout
- ListView



# RelativeLayout příklad

```
<?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

- od Android 3.0
  - existuje „support library“, která přidává podporu i pro starší verze (od API level 4)
    - pozor na balíček  
`android.app.Fragment`  
`android.support.v4.app.Fragment`
- znovupoužitelná část uživatelského rozhraní
  - ~ „vnořená aktivita“ s vlastním layoutem a životním cyklem
- aktivita může zobrazovat několik fragmentů
- snadná tvorba UI pro různé typy displayů
  - telefon
  - tablet

# Používání fragmentů



zdroj: <http://developer.android.com/training/basics/fragments/fragment-ui.html>

# Používání 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);
    }
}
```

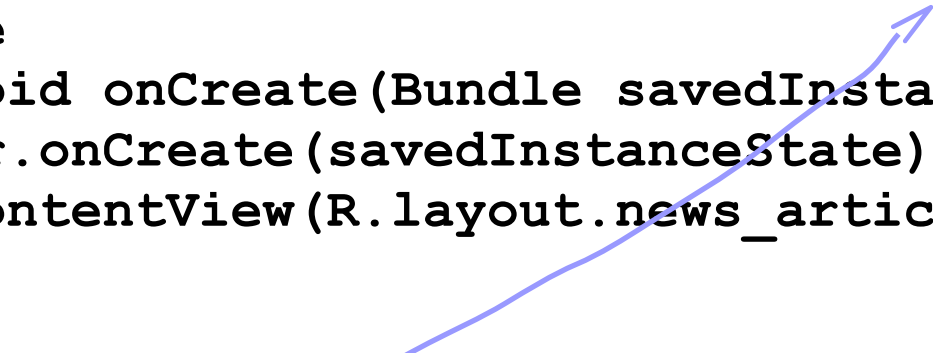
- res/layout-large/news\_articles.xml:

```
<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" .... />
```

# Používání fragmentů

- aktivita

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



- pokud je min API level 11, lze použít normální **Activity**

# Používání fragmentů

- předchozí příklad – pevné UI s dvěma fragmenty vhodné např. pro tablet
  - viz **large** přípona u layoutu
- pro střídání fragmentů (např. na telefonu) nutno manipulovat fragmenty z kódu
- res/layout/news\_articles.xml

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

  - prázdný layout – obsah přidáván z kódu
  - bez **large** přípony, tj. pro ostatní velikosti displejů

# Používání fragmentů

```
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();
        }
    }
}
```

# Používání fragmentů

- výměna zobrazeného fragmentu

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

# Intents

- komponenty aplikace (aktivity, služby a broadcast receivers) jsou aktivovány pomocí Intentů
  - „zprávy“
  - Intent – pasivní objekt
    - potomek `android.content.Intent`
    - položky
      - component name
      - action
        - řetězec
          - mnoho předdefinovaných
          - lze vytvořit vlastní
      - data
        - URI dat, se kterými se má pracovat
      - category
        - další informace o typu komponenty, která má na intent reagovat
      - extras



# Intents

- explicitní
  - se jménem cílové komponenty
  - typicky používané uvnitř aplikace
- implicitní
  - bez jména komponenty
  - typicky komunikace mezi aplikacemi
- intent filtry
  - které intenty komponenta může obsloužit
  - zapisuje se v manifestu

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

# Intents

- lze nastavit oprávnění reagování na intent
  - zapsáno v manifestu
  - schvalováno v okamžiku instalace
- i „systémové“ aplikace reagují na intenty
  - > lze si napsat vlastní „systémové“ aplikace
    - Mailer, SMS app, Homepage,...

# Intents – příklad

```
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

- zásobník spuštěných aktivit
  - aktivita reaguje na intent = vytvoří se nová instance a vloží se na zásobník
- uživatel komunikuje s aktivitou na vrcholu
- může existovat více tasků paralelně
  
- task ~ běžící aplikace

# Services

- služby běžící na pozadí
- potomci od `android.app.Service`
  - nespouštějí automaticky svoje vlákno!
- `IntentService`
  - potomek od `Service`
  - určeno pro služby reagující na intenty
  - již obsahuje správu vláken
  - stačí předefinovat `void onHandleIntent(Intent intent)`

# Vlákna

- aktivity aplikace se spouští v jednom vlákně
- události se také obsluhují v tomto vlákně
  - „main“ thread / UI thread
- obdobně jako Swing
  
- UI není „thread-safe“
  - manipulace s UI provádět v „main“ vlákně
  - neblokovat „main“ vlákno
  
- pomocné metody
  - `Activity.runOnUiThread(Runnable)`
  - `View.post(Runnable)`
  - `View.postDelayed(Runnable, long)`
- `AsyncTask`
  - obdoba `SwingWorkeru`

# Dialogy

```
public class AlertDialogFragment 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();    }    }
```

# Dialogy

- **zobrazení dialogu**

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



# Dialogy – starý způsob

voláno pouze jednou

```
@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;
}
```

“uživatelská”  
konstanta

voláno před  
každým zobrazením

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

# Dialogy – starý způsob

- `showDialog(DIALOG_SHOW_CONTACT);`
  - zobrazení dialogu



Verze prezentace AJ13.cz.2020.01

Tato prezentace podléhá licenci [Creative Commons Uveďte autora-Neužívejte komerčně 4.0 Mezinárodní License](https://creativecommons.org/licenses/by-nc/4.0/).