

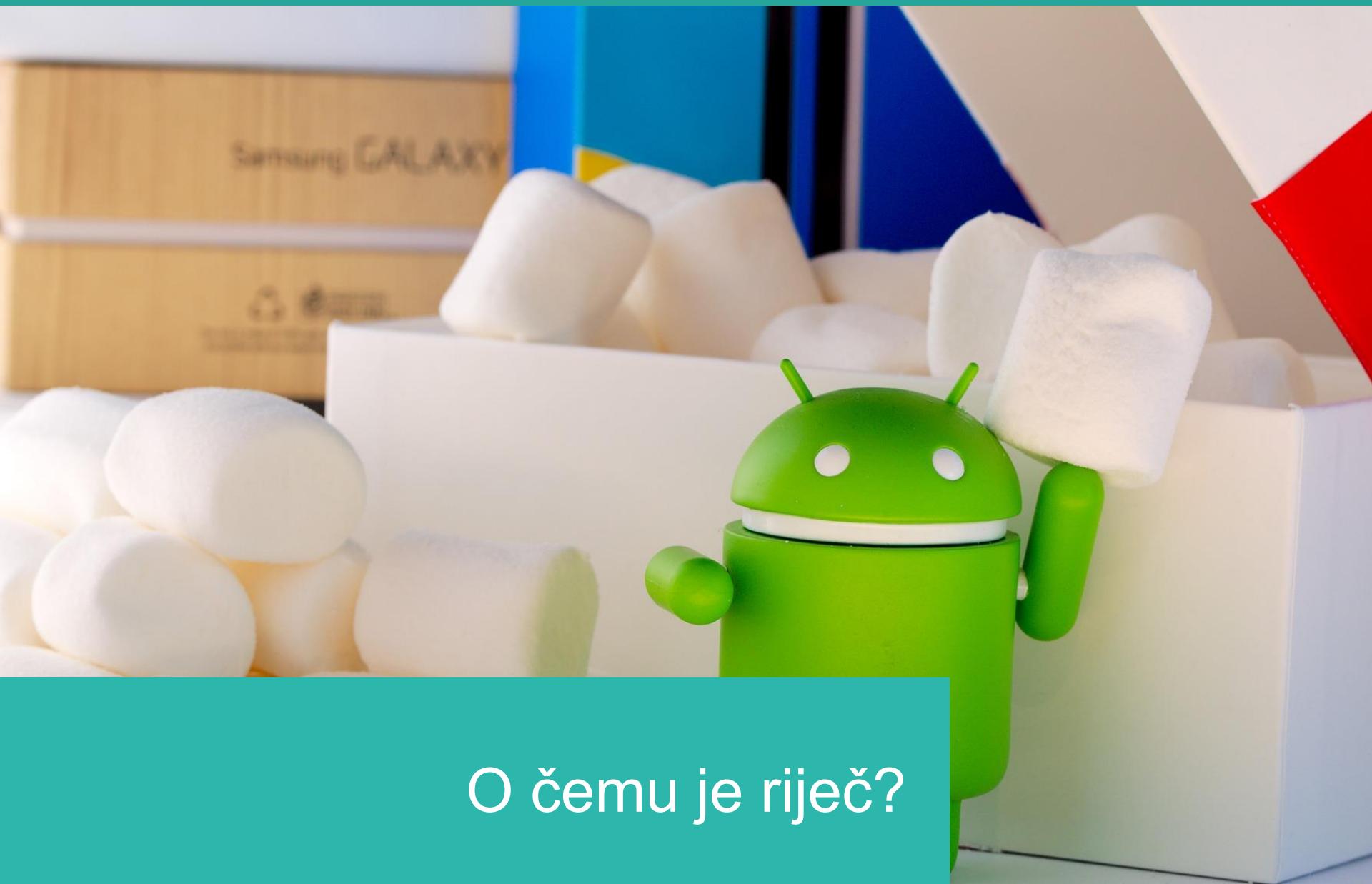


## A spoonful of desserts

Android Developer Akademija – predavanje 3

READ

SHARE



O čemu je riječ?

# ≡ Android OS



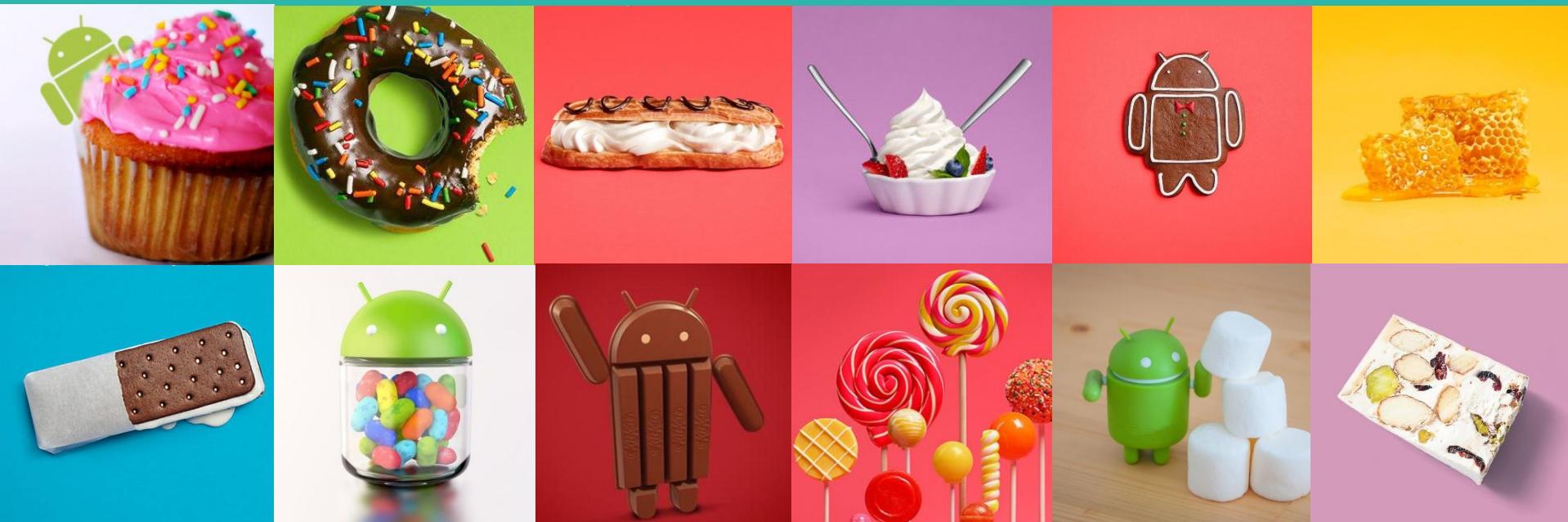
Mobilni OS zasnovan na Linux kernelu koji pogoni više od milijardu različitih uređaja

Preko 3 300 000 aplikacija samo na Play Storeu

Najavljen 2007., na tržištu od 2008.

Orijentiran na otvorenost i aplikacije (za razliku od ranog iOS-a)

# Slatkiši



Google nije stvorio Android

U osnovi je zamišljen kao OS za digitalne kamere

Verzija 0.9 imala je podršku za horizontalni homescreen, Lollipop nije

„I don't understand the impact they are going to have” – MS marketing tim

Britanska tvrtka lansirala je Nexus One u svemir u svrhu upravljanja satelitom

# ☰ Flagship



2008



# ☰ Kratka i dinamična povijest

## 2008

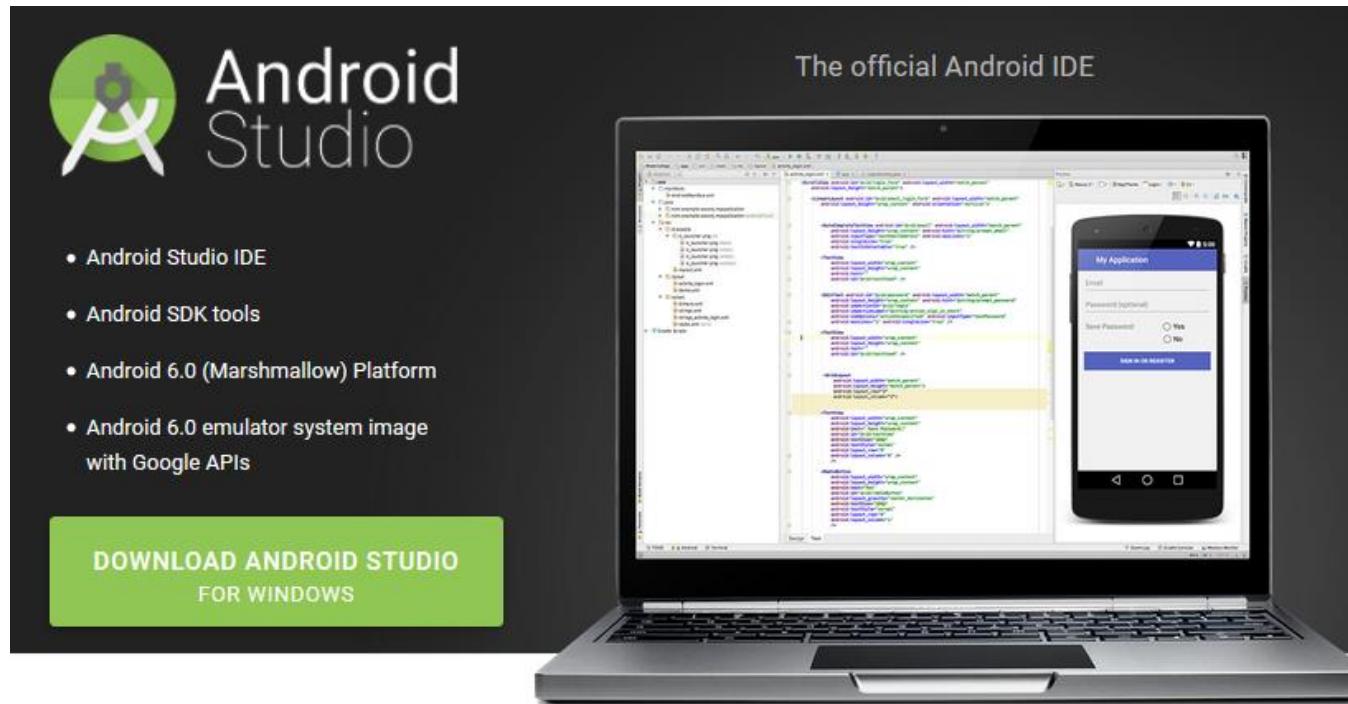


2015



# Što je potrebno?

# ☰ Osnovni alati



## Alati

IDE: Android Studio (dolazi s SDK)

Emulator (AVD, Genymotion itd.) ili uređaj

Java JDK

Gimp i/ili Corel i/ili Inkscape i/ili...

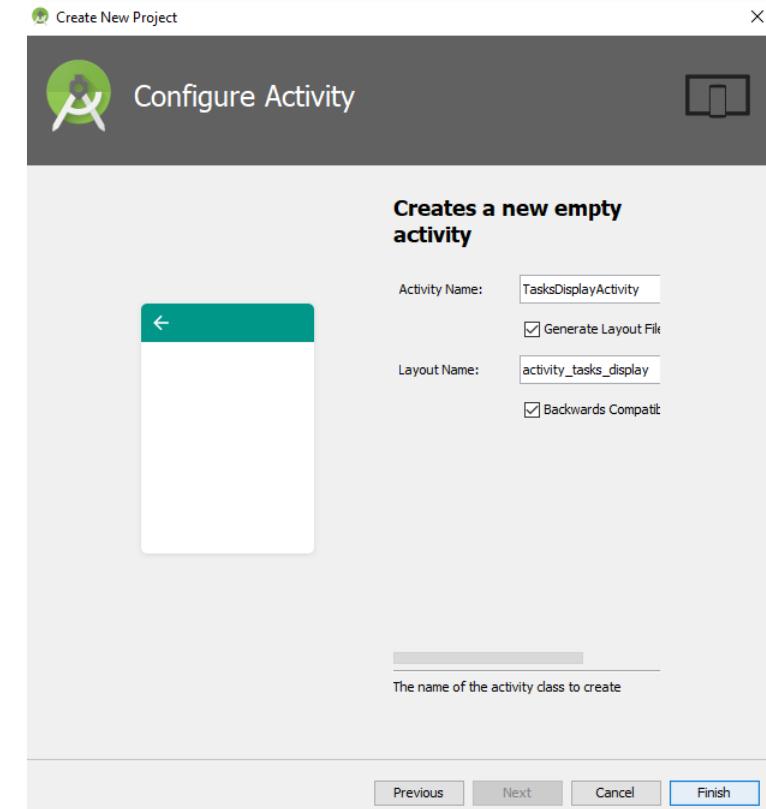
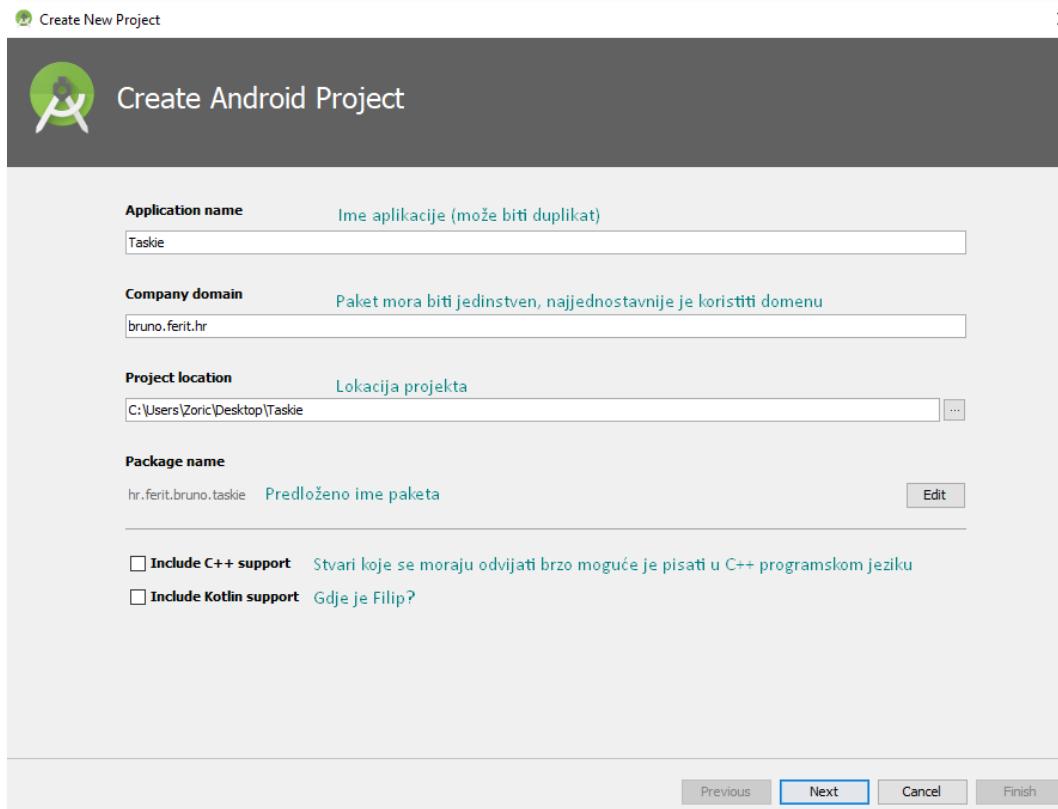
..... (<https://developer.android.com/sdk/index.html>)

..... (<https://www.genymotion.com/>)

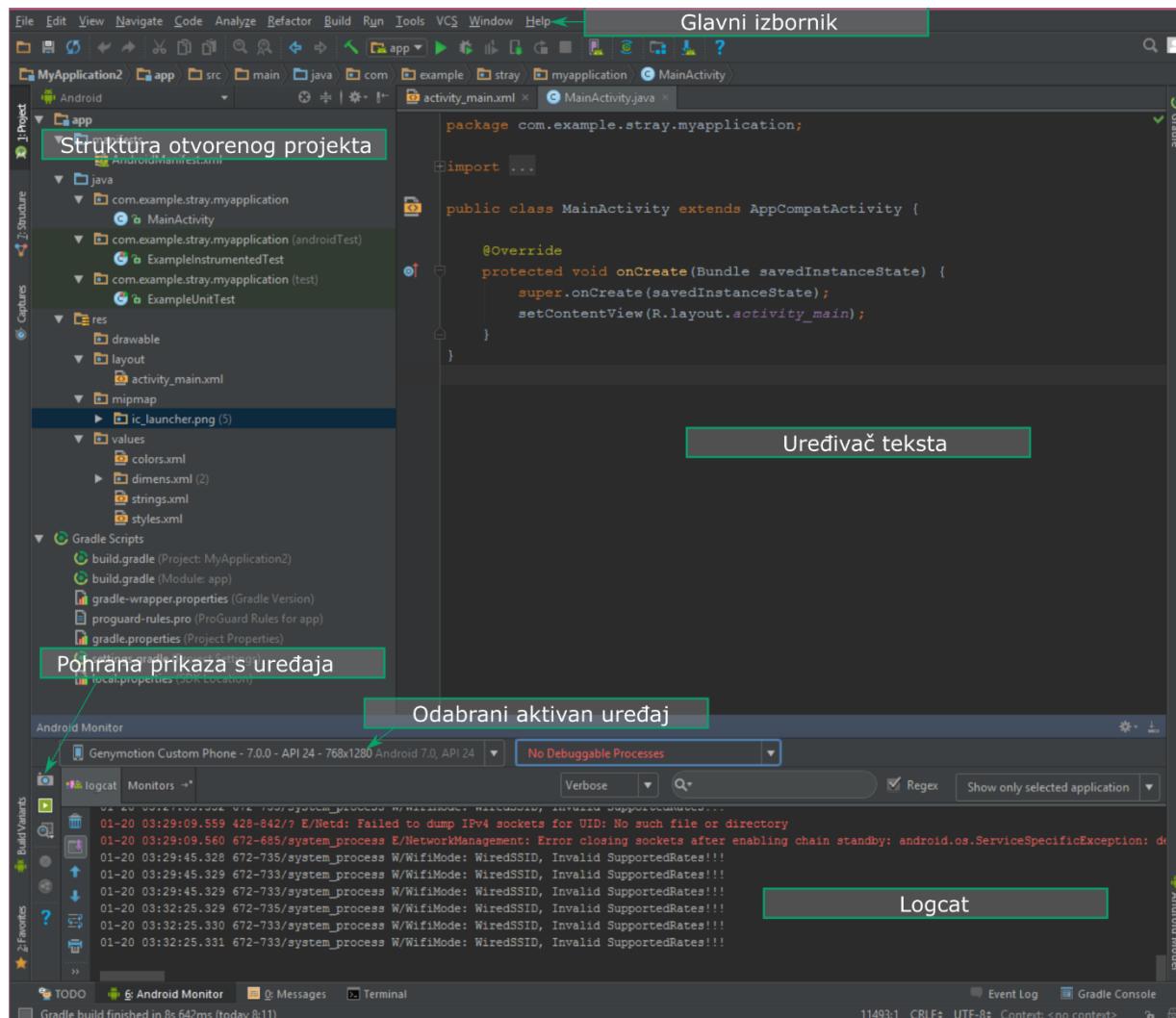
..... (<http://www.oracle.com/technetwork/java/javase/downloads>)

..... (<http://www.gimp.org/> / <https://inkscape.org/>)

# Android studio i projekt



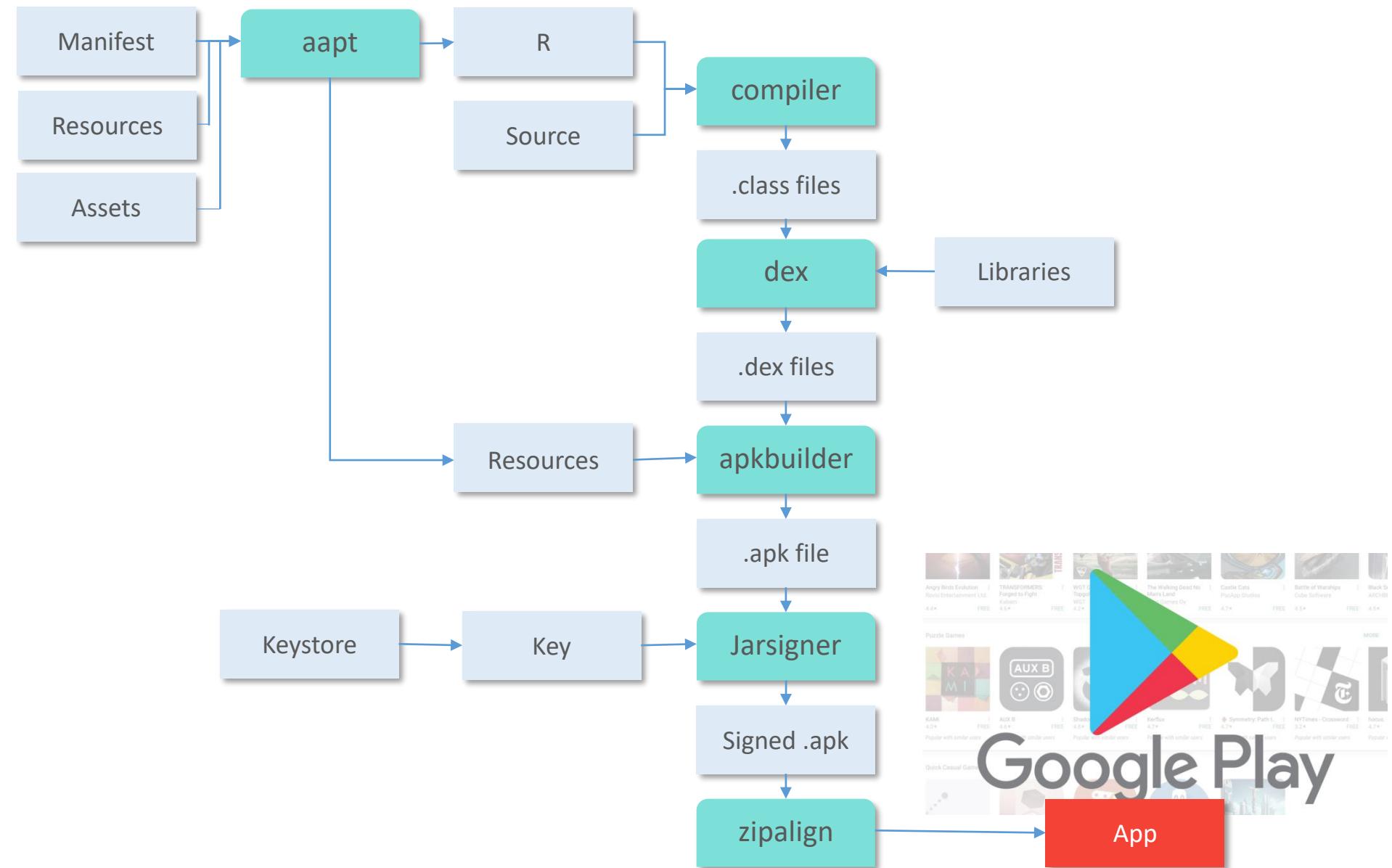
Stvaranje projekta kroz čarobnjak – postavljanje osnovnih informacija o projektu  
 Odabir ciljane platforme (Telefon, Wear, Tablet, Auto, Things) i njene inačice ([INFO](#)).  
 Dodavanje Activitya (izbor među nekoliko u galeriji predložaka)  
 Postavljanje informacija o Activityu



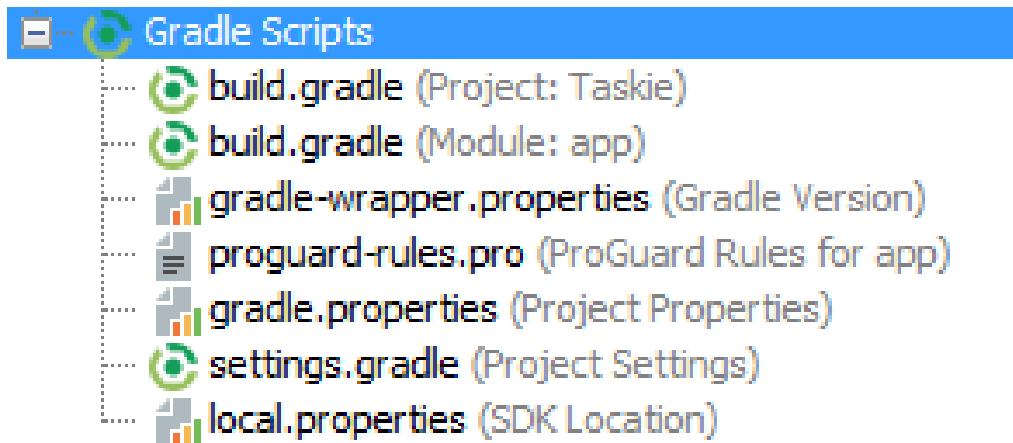
Kod prve instalacije SDK instalirati po podrazumijevanim postavkama, a kasnije pod Tools→Android→SDK Manager postaviti prema potrebama

SDK smjestiti van direktorija AS-a.

# ≡ Build process – from source to store



# ☰ Gradle



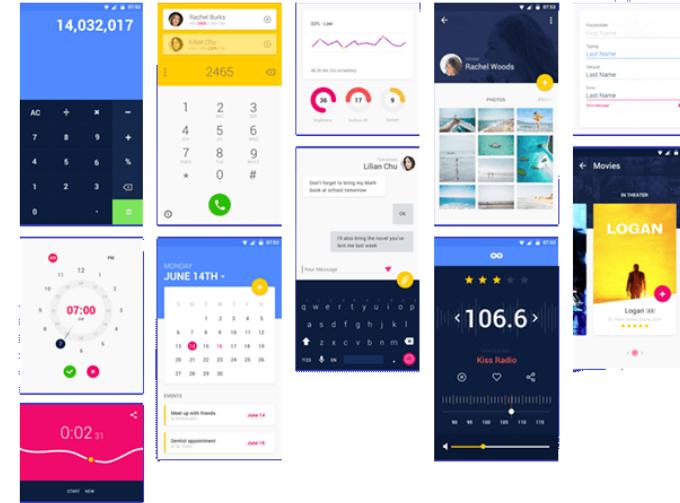
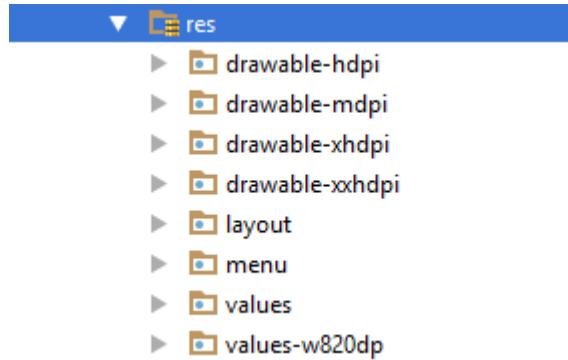
- Alat otvorenog koda za automatizaciju build procesa
- Široke mogućnosti za postavljanje koraka izgradnje aplikacije koje uključuju lako dodavanje vanjskih biblioteka, različite varijante aplikacije (npr. Plaćena/besplatna) i brojne druge stvari
- Programski jezik Groovy
- Dvije „glavne“ datoteke, na razini projekta i na razini modula
- Detaljnije: [gradle for Android](#)

# ☰ Manifest

- Sadrži ga svaka aplikacija
- Sadrži metapodatke o aplikaciji uključujući
  - Ime aplikacijskog paketa
  - Sve komponente aplikacije (*Activity, Broadcast receiver, Service, Content Providers*)
  - Dozvole aplikacije
  - HW/SW zahtjeve (primjerice OpenGL ili kamera)
- Više: [AndroidManifest.xml](#)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="hr.ferit.bruno.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

# ☰ Resources



- Svi resursi u Android aplikacijama se postavljaju kao vanjski elementi (eksternaliziraju se)
- Resursi uključuju sve od slika, stringova, boja, stilova, dimenzija do izgleda korisničkog sučelja
- Sustav će sam na temelju postavki uređaja učitati odgovarajuće resurse unutar aplikacije
- Resursi su dostupni u kodu preko R klase koja se automatski generira
- Svaki resurs jednoznačno je određen zadanim identifikatorom ili u slučaju vrijednosti poput stringova imenom
- U R klasi generira se konstantna cjelobrojna vrijednost koja predstavlja resurs

# ≡ Strings

- Stringovi se nalaze u mapi values, datoteci strings.xml
- Svaki je string označen imenom preko kojega mu se pristupa
- Moguće im je pristupiti preko R.string u kodu ili preko @string/ u xml-u
- Kada se aplikacija prevodi, daje se samo druga inačica strings.xml datoteke



Primjer – activity\_main.xml

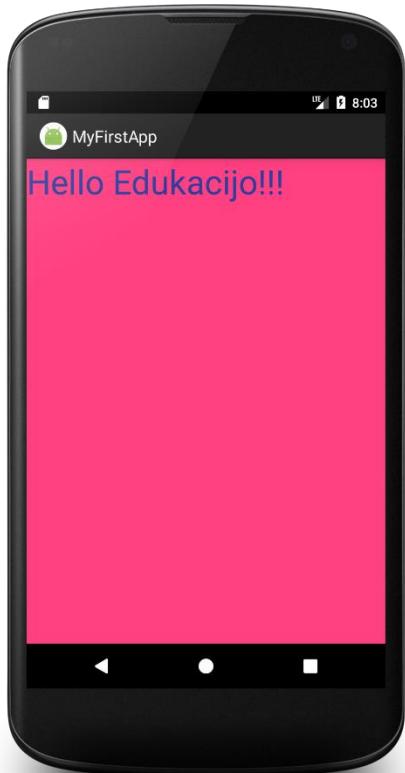
```
<!-- U XMLu -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

Primjer – strings.xml

```
<resources>
    <string name="app_name">Hello World</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">!Bok edukacijo!</string>
</resources>
```

# Colors

- Boje se nalaze u mapi values, datoteci uobičajeno nazvanoj colors.xml
- Svaka je boja označena imenom preko kojega joj se pristupa
- Moguće im je pristupiti preko R.color u kodu ili preko @color/ u xml-u
- Podržani oblici: #RGB | #ARGB | #RRGGBB | #AARRGGBB



Primjer – colors.xml

```
<resources>
    <color name=„colorBackground”>#FFFFFF</color>
    <color name=„colorForeground”>#000000</color>
</resources>
```

Primjer – activity\_main.xml

```
<!-- U XMLu -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorBackground">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:textColor="@color/colorForeground"/>
</LinearLayout>
```

# ☰ Dimensions

- Dimenzijs se nalaze u mapi values, datoteci nazvanoj npr dimens.xml
- Svaka je dimenzija označena imenom preko kojega joj se pristupa
- Moguće im je pristupiti preko R.dimen u kodu ili preko @dimen/ u xml-u
- Podržani oblici: dp, sp, pt, px, mm, in (a koriste se isključivo dp i sp)



Primjer – dimens.xml

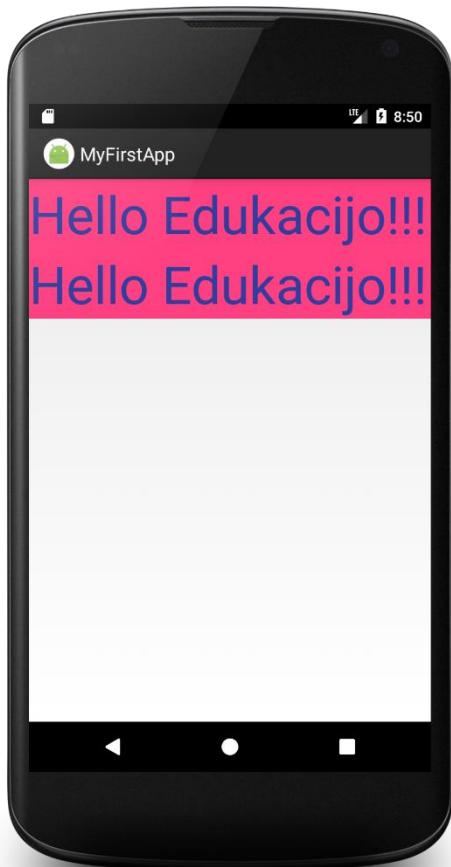
```
<resources>
    <dimen name="sizeTitleText">36sp</dimen>
    <dimen name="big_font">50sp</dimen>
    <dimen name="left_margin">10dp</dimen>
</resources>
```

Primjer – activity\_main.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorBackground">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:textSize="@dimen/big_font"
        android:layout_marginLeft="@dimen/left_margin"
        android:textColor="@color/colorForeground"/>
</LinearLayout>
```

# ☰ Styles

- Stilovi se nalaze u mapi values, datoteci nazvanoj npr styles.xml
- Svaki je stil označen imenom preko kojega joj se pristupa
- Moguće im je pristupiti preko R.style u kodu ili preko @style/ u xml-u
- Vrlo su korisni ako se isti izgled primjenjuje na puno različitih kontrola, podržavaju nasljeđivanje



Primjer – styles.xml

```
<resources>
    <style name="TextBoxCustom">
        <item name="android:background">@color/colorBackground</item>
        <item name="android:textColor">@color/colorForeground</item>
        <item name="android:textSize">@dimen/big_font</item>
    </style>
</resources>
```

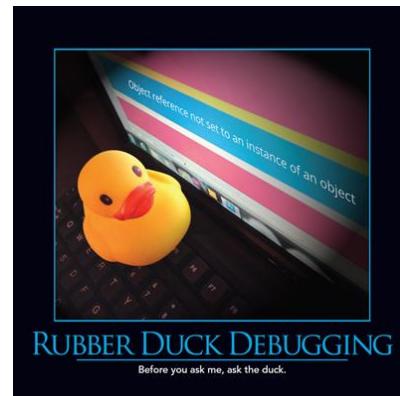
Primjer – activity\_main.xml

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        style="@style/TextBoxCustom"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        style="@style/TextBoxCustom"/>
</LinearLayout>
```

# ≡ Drawables

- Grafički elementi se nalaze u mapi drawables-????, gdje upitnici predstavljaju gustoću zaslona
- Svaki je drawable označen imenom preko kojega mu se pristupa
- Moguće im je pristupiti preko R.drawable u kodu ili preko @drawable/ u xml-u
- Podržani formati su .jpg, .gif, .png i .9.png a moguće ih je definirati i u XML-u (oblici, gradijenti itd.). Novije inačice Android sustava pružaju podršku i za vektorske formate.

image.png



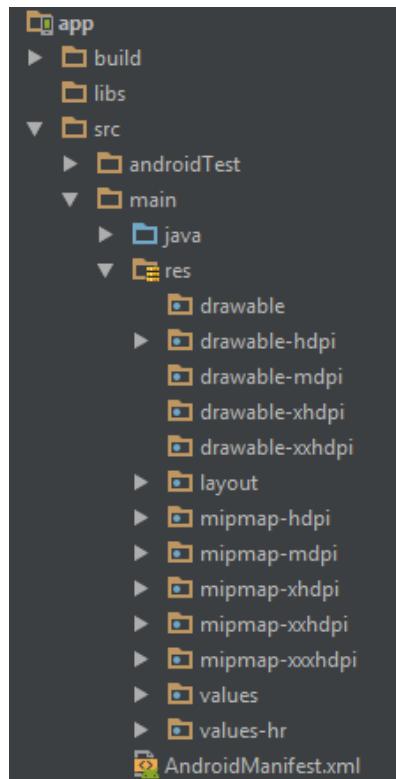
mdpi	- 100 x 100
hdpi	- 150 x 150
xhdpi	- 200 x 200
xxhdpi	- 300 x 300
xxxhdpi	- 400 x 400



## Primjer – activity\_main.xml

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="@color/colorBackground">  
    <TextView  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:text="@string/hello_world"  
        android:textSize="@dimen/big_font"  
        android:layout_marginLeft="@dimen/left_margin"  
        android:textColor="@color/colorForeground"  
        android:drawableBottom="@drawable/image"/>  
    </LinearLayout>
```

- Android platforma eksternalizira sve resurse, alternativni resursi smještaju se u mape s kvalifikatorima u imenu, primjerice values-hr
- Podrazumijevani resursi su u values mapi, a dobro je da budu na engleskom jeziku
- <https://developer.android.com/guide/topics/resources/providing-resources.html>



### Primjer – strings.xml u values

```
<resources>
    <string name="text_greeting">Good afternoon my friends!</string>
</resources>
```

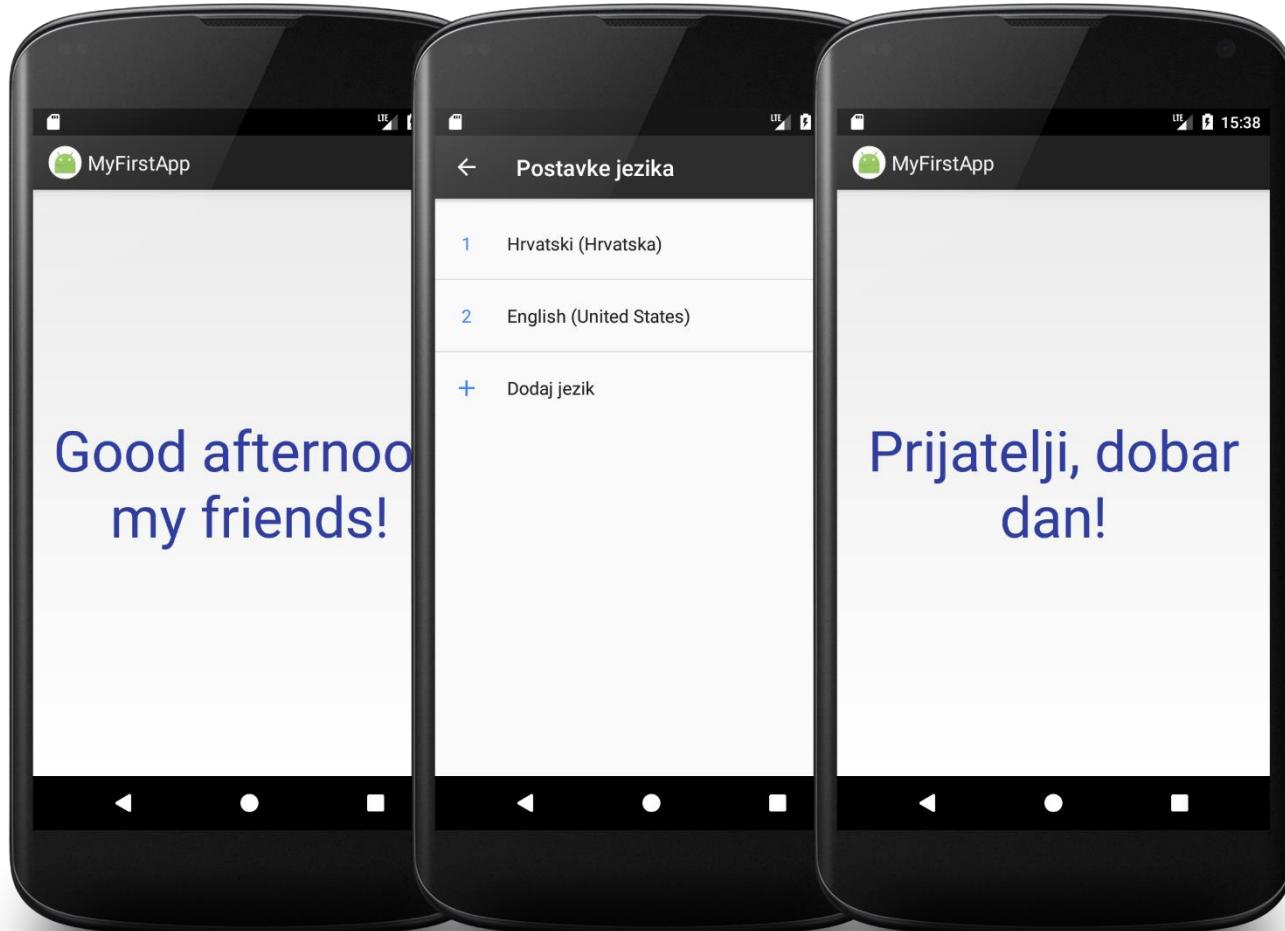
### Primjer – strings.xml u values-hr

```
<resources>
    <string name="text_greeting">Prijatelji, dobar dan!</string>
</resources>
```

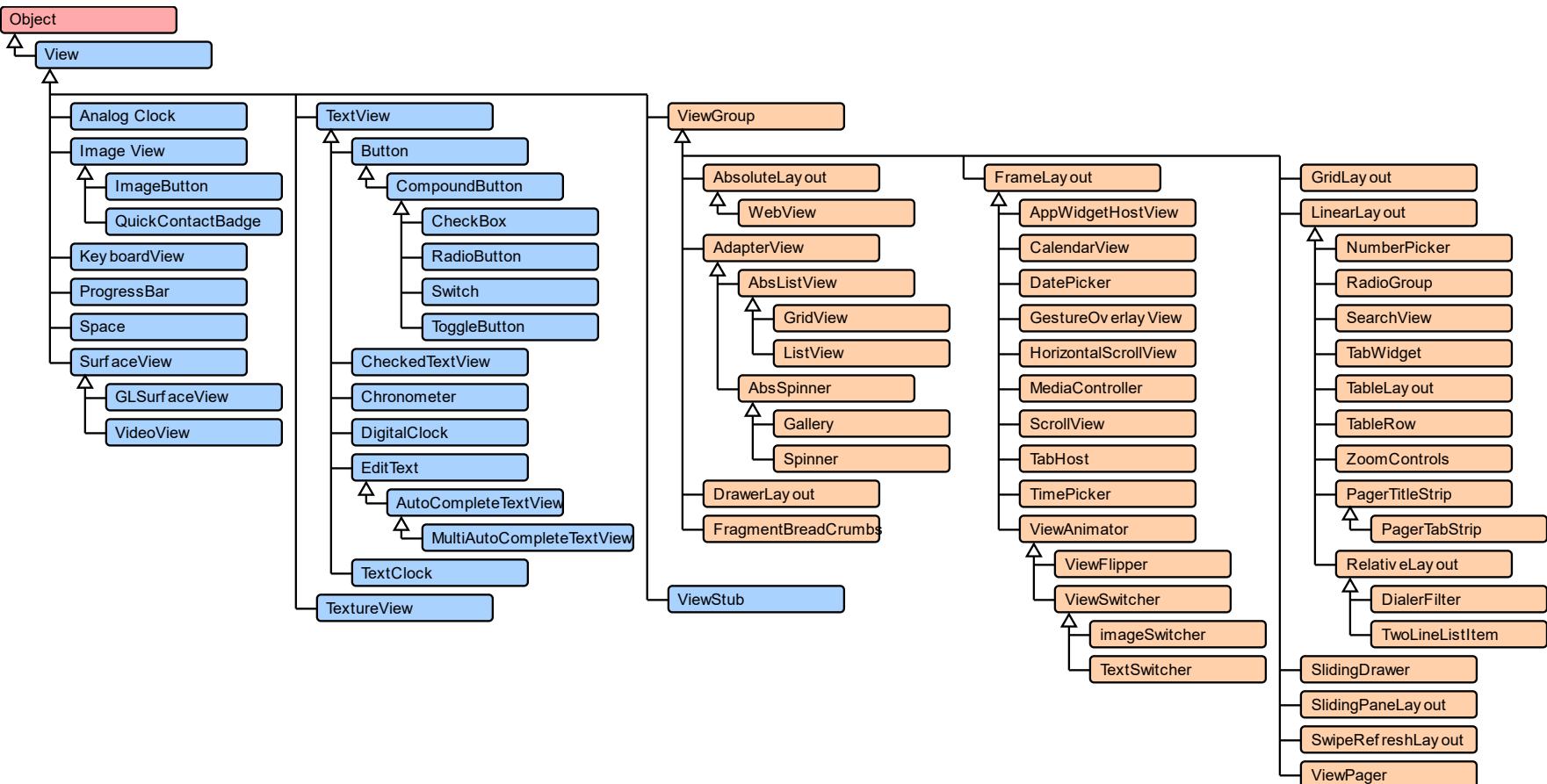
### Primjer – activity\_main.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/text_greeting"
        android:gravity="center"
        android:textSize="@dimen/big_font"
        android:textColor="@color/colorForeground"/>
</LinearLayout>
```

- Ako se u sustavu sada promijeni jezik na hrvatski, tada se automatski učitavaju stringovi iz values-hr
- Ukoliko neki string nije specifično definiran za odabrani locale, tada se koristi defaultna vrijednost
- Osim teksta, često se pružaju alternativni resursi za layout, primjerice kod korištenja na televiziji ili u arapskim zemljama gdje je orijentacija teksta drugačija.



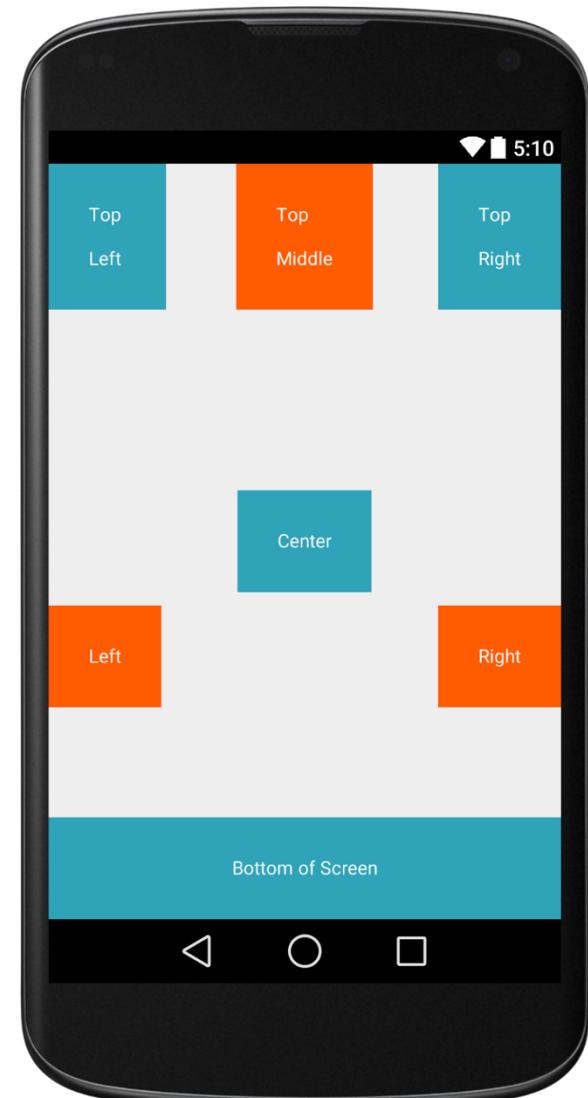
# Pogled iz druge perspektive



- `View` je osnovna klasa za elemente korisničkog sučelja
- Osim osnovnih elemenata, iz nje su izvedeni i svi upravitelji layoutima
- Moguće je naslijediti neku od postojećih klasa ili samu `View` klasu kako bi se izradili specifični elementi korisničkog sučelja

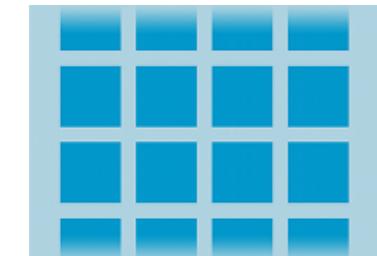
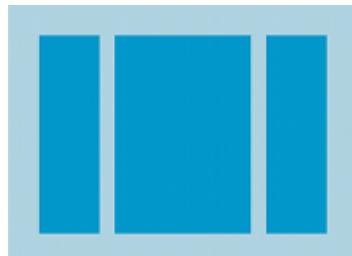
# ≡ Layout

- Layout resursi definiraju izgled korisničkog sučelja
- Smješteni su u mapu res/layout
- Imaju jedan korijenski element koji je ili neka UI kontrola (View) ili upravitelj kontrolama (ViewGroup) koji omogućuje smještanje kontrola unutar sebe
- Kako bi se moglo pristupati ovim elementima potrebno je definirati id atribut u XMLu
- Nužno je definirati visinu i širinu svakog elementa u layoutu, a najčešće se to postiže korištenjem
  - **match\_parent** → Popunjava širinu/visinu roditelja (nadređenog elementa)
  - **wrap\_content** → Omata se oko sadržaja – poprima širinu/visinu sadržaj
- Uz ove konstante rabe se i vrijednosti izražene u dp



# ≡ Layout manager

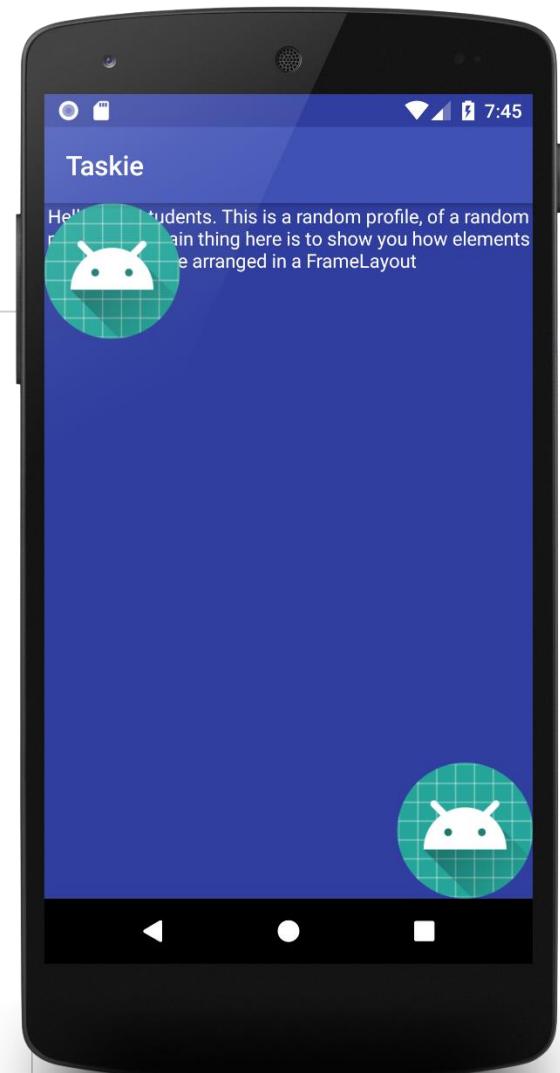
- Layout manager je naziv za objekte klase koje omogućuju grupiranje kontrola za izgradnju korisničkih sučelja
- Prilikom prevodenja, layout se prevede u View resurs kojeg je potrebno iskoristiti za postavljanje korisničkog sučelja, ovo se postiže pozivom setContentView() metode u onCreate() metodi Activitya
- Predaje se id resursa kao parametar
- Moguće ih je ugniježđivati i tako postići složena sučelja, iako pretjerano grijezđenje nije preporučljivo
- Svaki od Layouta dizajniran je da dobro skalira, a preferirani način njihova definiranja je u XML datoteci
- FrameLayout, LinearLayout, RelativeLayout, ConstraintLayout



# FrameLayout

- Jedan od najjednostavnijih layout mangera
- Slaže djecu unutar sebe jedno iznad drugog, s tim da je na dnu onaj view koji smo prvi dodali u layout
- Uobičajeno se drži samo jedno dijete unutar ovakvog layouta

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimaryDark"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@color/colorTextNormal"
        android:text="@string/tvProfileDescription"
        android:gravity="center"/>
    <ImageView
        android:layout_width="@dimen/avatarWidth"
        android:layout_height="@dimen/avatarHeight"
        android:src="@mipmap/ic_launcher_round"
        />
    <ImageView
        android:layout_width="@dimen/avatarWidth"
        android:layout_height="@dimen/avatarHeight"
        android:layout_gravity="right|bottom"
        android:src="@mipmap/ic_launcher_round"
        />
</FrameLayout>
```

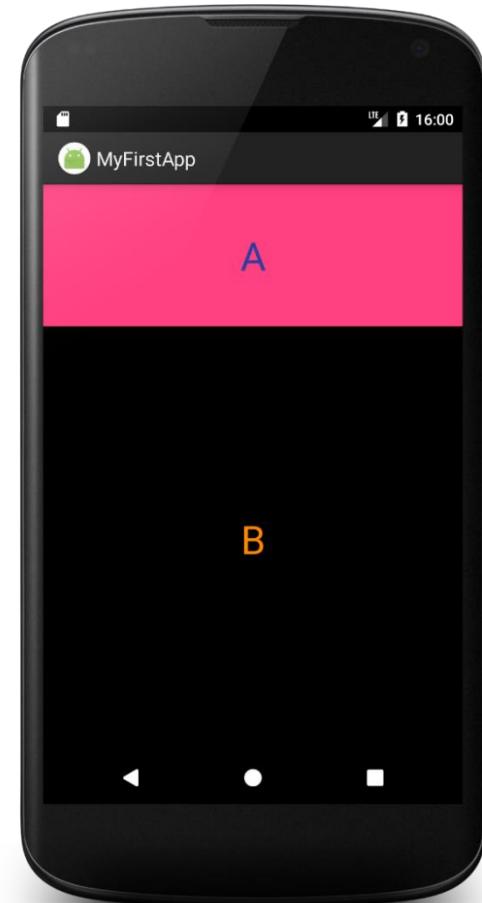


# ≡ LinearLayout

- LinearLayout grupira View kontrole u jednom smjeru, vertikalno ili horizontalno
- Sva su djeca unutar ovog layout naslagana jedno uz drugog
- Orijentacija se kontrolira s android:orientation i nužno ju je navesti
- Podržana je težina pojedinog djeteta koja omogućuje popunjavanje u ovisnosti o dostupnom prostoru

## Layout – Activity

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <TextView  
        android:layout_width="match_parent"  
        android:layout_height="0dp"  
        android:layout_weight="1"  
        android:text="A"  
        android:gravity="center"  
        android:textSize="@dimen/sizeTitleText"  
        android:textColor="@color/colorForeground"  
        android:background="@color/colorBackground"/>  
    <TextView  
        android:layout_width="match_parent"  
        android:layout_height="0dp"  
        android:layout_weight="3"  
        android:text="B"  
        android:gravity="center"  
        android:textSize="@dimen/sizeTitleText"  
        android:textColor="@android:color/holo_orange_dark"  
        android:background="@android:color/darker_gray"/>  
</LinearLayout>
```



# ≡ RelativeLayout

- RelativeLayout grupira View kontrole u odnosu na druge kontrole ili roditelja
- Da bi se navedeno postiglo, svakoj kontroli se mora dati id atribut na temelju kojeg je moguće slagati kontrole ispod drugih, pored drugih i slično
- Moguće je kontrole slagati i u odnosu na roditelja, primjerice centrirajući ih u roditelju

### Layout Activity

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="A"
        android:gravity="center"
        android:textSize="@dimen/sizeTitleText"
        android:textColor="@color/colorForeground"
        android:background="@color/colorBackground"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="B"
        android:gravity="center"
        android:textSize="@dimen/sizeTitleText"
        android:textColor="@android:color/holo_orange_dark"
        android:background="@android:color/black"/>
</RelativeLayout>
```



# ☰ TextView

- Kontrola za prikaz tekstualnog sadržaja
- Moguće kontrolirati izgled fonta, boju, veličinu, poravnanje...
- <https://developer.android.com/reference/android/widget/TextView.html>

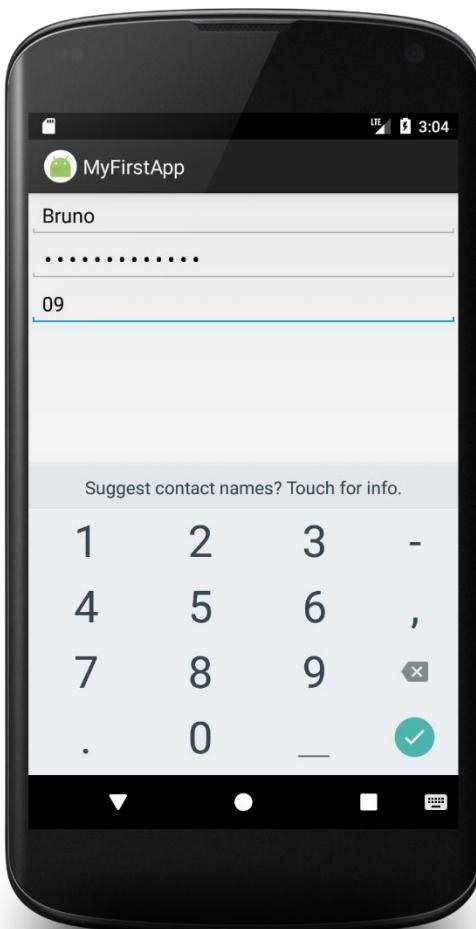


Primjer – activity\_main.xml

```
<LinearLayout  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <TextView  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:text="@string/hello_world"  
        style="@style/TextBoxCustom"/>  
    <TextView  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:text="@string/hello_world"  
        style="@style/TextBoxCustom"/>  
</LinearLayout>
```

# EditText

- Kontrola za unos tekstualnog sadržaja
- Moguće kontrolirati tipkovnicu koja se nudi korisniku, izgled teksta itd.
- <https://developer.android.com/reference/android/widget/EditText.html>



Primjer activity\_main.xml

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <EditText  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:hint="@string/hint_Username"  
        android:inputType="textCapWords"/>  
    <EditText  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:hint="@string/hint_Password"  
        android:inputType="textPassword"/>  
    <EditText  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:hint="@string/hint_Phone"  
        android:inputType="numberDecimal"/>  
/<LinearLayout>
```

# ☰ ImageView

- Kontrola za prikaz grafičkog sadržaja
- Podržani su rasterski formati i u novijim inačicama vektorski
- Slike se postavljaju u drawable mapu, odgovarajuće razlučivosti u odgovarajuće podmape
- <https://developer.android.com/reference/android/widget/ImageView.html>



Primjer activity\_main.xml

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <ImageView  
        android:layout_width="200dp"  
        android:layout_height="200dp"  
        android:layout_gravity="center"  
        android:src="@drawable/image"  
        android:contentDescription="@string/image_desc"  
        android:scaleType="fitCenter"/>  
</LinearLayout>
```

# ≡ Button

- Kontrola za korisnički unos
- Definira se metoda koja će biti pozvana klikom na gumb
- <https://developer.android.com/reference/android/widget/Button.html>



Primjer - activity\_main.xml

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <Button  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/button_text"  
        android:layout_gravity="center"  
        android:onClick="doSomeWork"/>  
</LinearLayout>
```

# ≡ Activities

- Jedan od osnovnih gradivnih elemenata Android aplikacije
- U osnovi ekran koji aplikacija predstavlja korisniku
- Ekvivalent je formi (engl. form) u programiranju desktop aplikacija



**Primjer – Activity**

```
package hr.ferit.brunozoric.myfirstapp;

import android.app.Activity;
import android.os.Bundle;

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

**Layout – Activity**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name"/>
</LinearLayout>
```

# Vezanje

- Do sada su na sučelje Activitya dodavani različiti elementi korisničkog sučelja poput TextViewa, ImageViewa i slično. Da bi im se pristupilo iz programskog koda, potrebno je najprije dohvatiti referencu na njih
- Riječ je o objektima koje Android „napuše“ iz prema uputama koje su dane u XML datoteci
- Ovo obavlja metoda setContentView ili sami možemo iskoristiti klasu LayoutInflater kako bismo iz XML datoteke kreirali neki View objekt
- Da bi se dohvatila referenca na objekt koji je smješten na korisničkom sučelju, potrebno je osloniti se na metodu findViewById()
- Toj metodi predaje se id View objekta, a njen je zadatak proći kroz cijelu strukturu na korisničkom sučelju i pronaći objekt s baš tim identifikatorom
- Razmislite o imenima!

The screenshot shows the GitHub search interface. At the top, there's a navigation bar with 'Explore', 'Gist', 'Blog', and 'Help'. Below it is a search bar containing the word 'foo'. On the left, there are two tabs: 'Repositories' (14,105 results) and 'Code' (38,900,125 results, which is highlighted). A message in the center says 'We've found 38,900,125 code results'.

```
byte me;  
Exception up; throw up;  
String me_along;  
Object strongly;  
String cheese;  
getEven();  
getAround();  
foo; data; a; acc; accynd;
```

## Primjer - MainActivity.java

```
public class MainActivity extends Activity implements View.OnClickListener {
    private static final String TAG = "Bruno";
    Button bActivityListener, bAnonInnerListener;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        this.initialize();
    }

    private void initialize(){
        this.bActivityListener = (Button) this.findViewById(R.id.bDisplayUserName);
        this.bAnonInnerListener = (Button) this.findViewById(R.id.bDisplayUserEmail);
    }
}
```

## Primjer – activity\_main.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/bDisplayUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/bAnonInnerListenerText"/>
    <Button
        android:id="@+id/bDisplayUserEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/bActivityListenerText"/>
</LinearLayout>
```

# ☰ Tko osluškuje osluškivače?

- Do sada je korišten onClick atribut za obradu klika na gumb, no ovim načinom kod se miješa sa definiranjem sučelja
- U pozadini se generira kod kojeg ne vidimo, a nad kojim nemamo kontrolu
- Ponekad je korisnije / prikladnije povezivanje interakcije korisnika sa sučeljem i koda obaviti samostalno, dok kod iole složenijih interakcija drugačiji pristup nije ni moguć (primjerice, ne postoji atribut za dugi pritisak na gumb)
- Za definiranje ponašanja kod interakcije sa korisničkim sučeljem koriste se sučelja (*interface*) poput onClickListener ili onLongClickListener (ili vlastita)
- Dovoljno je implementirati metode koje nalaže sučelje te preko identifikatora kontrole dodijeliti joj odgovarajući objekt, instancu klase koja sučelje i implementira
- Dva česta pristupa su da se napiše anonimna unutarnja klasa koja implementira sučelje (retrolambda / lambda u Java 8) ili da sam Activity ponudi implementaciju sučelja

## Primjer 72. – MainActivity.java

```
public class MainActivity extends Activity implements View.OnClickListener {
    private static final String TAG = "Bruno";
    Button bActivityListener, bAnonInnerListener;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        this.initialize();
    }

    private void initialize() {
        this.bActivityListener = (Button) this.findViewById(R.id.bActivityListener);
        this.bAnonInnerListener = (Button) this.findViewById(R.id.bAnonInnerListener);
        this.bActivityListener.setOnClickListener(this);
        this.bAnonInnerListener.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                logThings("onClick in an anonymous inner class.");
            }
        });
    }

    public void onClick(View view) {
        logThings("onClick method in MainActivity");
    }

    private void logThings(String message) {
        Log.d(TAG, message);
    }
}
```

## Primjer 72. – activity\_main.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/bAnonInnerListener"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/bAnonInnerListenerText"/>
    <Button
        android:id="@+id/bActivityListener"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/bActivityListenerText"/>
</LinearLayout>
```

# Biblioteke

- Do sada smo se u izradi aplikacija oslanjali isključivo na kod dostupan kroz Android SDK
- Ipak, postoji ogromna količina koda dostupnog kroz različite biblioteke (kod koji se može koristiti dinamički tijekom izvođenja, nije potrebno prevodenje)
- Kako bi se koristile biblioteke, potrebno ih je dodati u projekt što je moguće postići na različite načine (primjerice korištenjem repozitorija) uporabom gradle build skripti
- Dodaje se implementation naredba u build.gradle Module: app
- Koristan resurs predstavljaju stranice <http://gradleplease.appspot.com/>
- Najbolje je popratiti upute koje uobičajeno daju autori svake od biblioteka
- Važnije biblioteke za android aplikacije:
  - Butterknife
  - Picasso / Glide
  - Retrofit / Volley
  - Gson / Jackson / Moshi
  - ORMLite / GreenDAO / Realm / Room
  - AChartEngine
  - Zxing
  - Dagger2
  - ... (milijun i jedna druga, vidi ([www.android-arsenal.com](http://www.android-arsenal.com))

# ☰ Butterknife



- Butterknife <http://jakewharton.github.io/butterknife/>
- Biblioteka za izbjegavanje dosadnog findViewById()
- Povezivanje kontrola na sučelju s metodama za obradu događaja
- Koristi anotacije poput @BindView

## Primjer 79. – build.gradle Module: app

```
dependencies {  
    ...  
  
    implementation 'com.jakewharton:butterknife:8.8.1'  
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.8.1'  
}
```

# ☰ Gradle and library versions

- Iako kod koji je potrebno umetnuti u Gradle build skriptu na prethodnom slideu radi bez ikakvih problema (ako nemate staru verziju build toolsa 😊), to nije najbolji način za dodavanje zavisnosti
- Inačicu biblioteke najbolje je izdvojiti u ekstra svojstva projekta
- U skripti koja se odnosi na modul tada se oslanja na vanjska svojstva što znači da je vrlo lako kontrolirati inačice biblioteka i osvježavati ih u slučaju promjene

## Primjer - build.gradle

```
ext{  
    butterknifeVersion = "8.8.1"  
}
```

## Primjer – build.gradle Module: app

```
dependencies {  
    ...  
  
    implementation "com.jakewharton:butterknife:$butterknifeVersion"  
    annotationProcessor "com.jakewharton:butterknife-compiler:butterknifeVersion"  
}
```

# ☰ Ne volim anonimne unutarnje klase. - Đ. Balašević

- Butterknife omogućuje pisanje metoda koje će biti pozvane u slučaju događaja poput klika na neki element korisničkog sučelja
- Umjesto da se definira anonimna unutarnja klasa koja implementira odgovarajuće sučelje, Butterknife dopušta označavanje metoda anotacijama poput @OnClick
- Uz anotaciju se navodi identifikator View objekta za čiji se događaj ta funkcija želi prijaviti
- U slučaju postojanja potrebe da se ista metoda poziva za različite View objekte, moguće ih je navesti u listi u obliku @OnClick({R.id.b1, R.id.b2})
- Osim View objekata, moguće je vezati i resurse

I THINK GROWN-UPS JUST  
ACT LIKE THEY KNOW  
WHAT THEY'RE DOING.



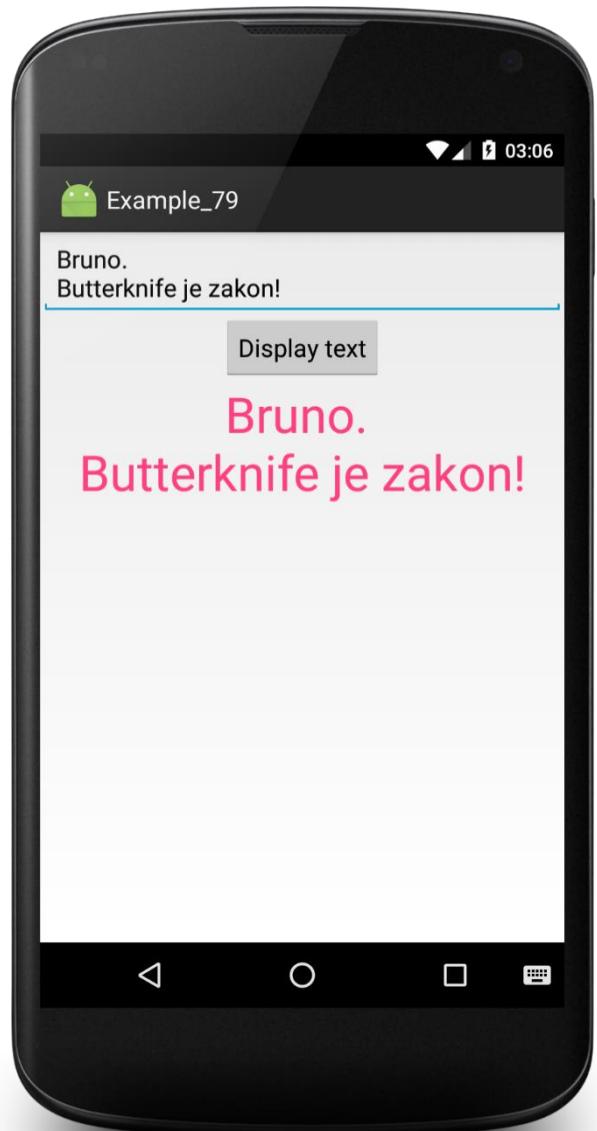
## Primjer – MainActivity.java

```
public class MainActivity extends Activity {

    // Use annotations for Views, no longer findViewById
    @BindView(R.id.bDisplayText) Button bDisplayText;
    @BindView(R.id.etTextEntry) EditText etTextEntry;
    @BindView(R.id.tvDisplay) TextView tvDisplay;

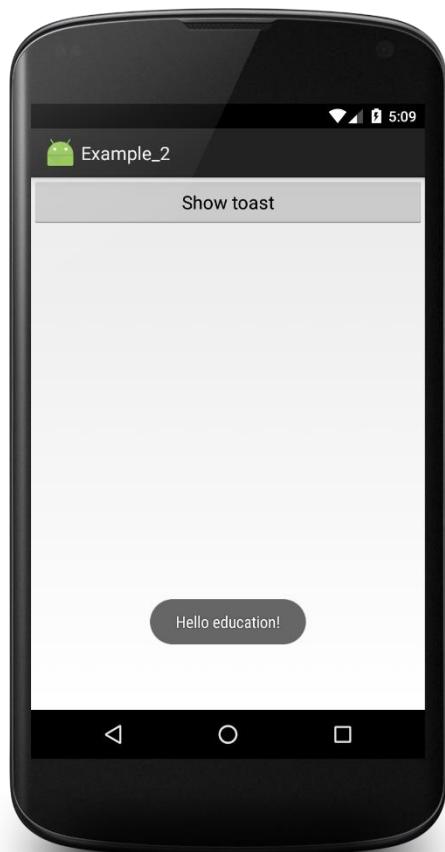
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
    }

    // Easy way to set event listeners
    @OnClick(R.id.bDisplayText)
    public void displayText() {
        String message = this.etTextEntry.getText().toString();
        tvDisplay.setText(message);
    }
}
```



# ☰ Toasts. Ne. Nisu za jesti. Ni piti.

- Toast je malena obavijest koja se pojavljuje u obliku pop-up prozora kako bi korisniku pružila informaciju o određenom događaju, a zauzima samo prostor nužan za prikaz obavijesti te ne podržava interakciju
- Kreira se objekt klase Toast kojemu se postavljaju parametri i na kojem se poziva metoda show() kako bi se prikazao



Primjer 70. – MainActivity.java

```
public class MainActivity extends Activity {

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

    public void showToast(View view) {
        String message = "Hello education!";
        Toast toast = Toast.makeText(this, message, Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

Primjer 70. – activity\_main.xml

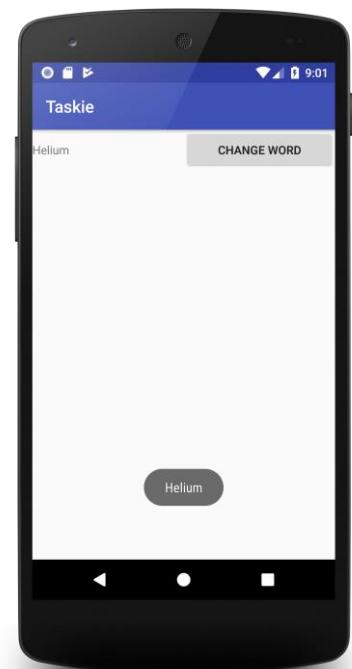
```
<Button
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/bShowToastText"
    android:onClick="showToast"/>
```

# Trebam string. Što sad?

- Resursima je korisno pristupati iz drugih resursa (primjerice, referencirati string vrijednost iz layout datoteke, referencirati boju iz druge boje i slično)
- Također je korisno resursima pristupati iz koda
- Kako bi se dohvatio identifikator, rabi se R klasa i preko odgovarajućeg tipa se pristupa konstantama koje predstavljaju resurse
- Određenim je metodama (primjerice, BitmapFactory.decodeResource(...)) potrebno predati Resources objekt kao argument. U ovom slučaju moguće je koristiti getResources() metodu koja je deklarirana u Contextu i moguće ju je pozvati preko nekog Context objekta (poput Activitya).

## Primjer – MainActivity.java

```
@OnClick(R.id.bChangeWord)
public void changeWord() {
    int messageId =
    mRandomGenerator.nextBoolean() ? R.string.firstWord :
    R.string.secondWord;
    String message = getResources().getString(messageId);
    tvWordDisplay.setText(message);
    // Could have done this:
    // tvWordDisplay.setText(messageId);
    // not the point here.
    displayToast(message);
}
```



# Logging

- Česta potreba kod izrade Android aplikacija je bilježenje stanja varijabli, prikaz određenih bilješki tijekom izvođenja i slično
- Log klasa omogućuje upravo to, korištenjem različitih metoda moguće je prikazivati bilješke u LogCat alatu pri čemu bilješke imaju različite razine važnosti
- Svaka metoda traži dva parametra, oznaku i poruku. Oznake se uobičajen definiraju kao konstante



## ◦ Prioriteti:

- |              |                           |
|--------------|---------------------------|
| ◦ Log.v();   | - verbose                 |
| ◦ Log.i();   | - info                    |
| ◦ Log.d();   | - debug                   |
| ◦ Log.w();   | - warning                 |
| ◦ Log.e();   | - error                   |
| ◦ Log.wtf(); | - what a terrible failure |

## Primjer – MainActivity.java

```
public class MainActivity extends Activity {

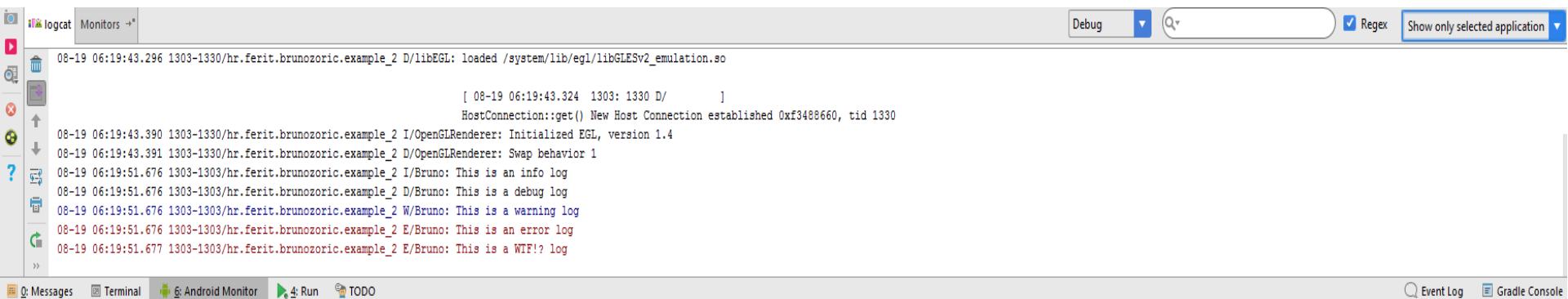
    private static final String TAG = "Bruno";

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

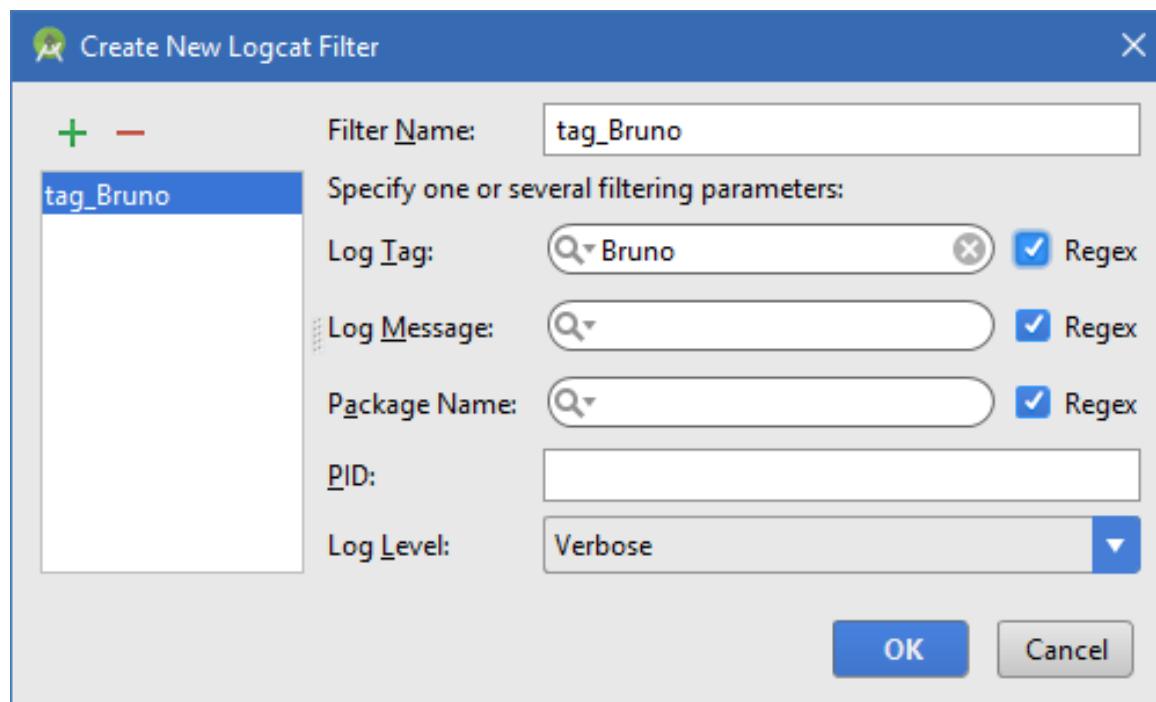
    public void logThings(View view) {
        Log.i(TAG, "This is an info log");
        Log.d(TAG, "This is a debug log");
        Log.w(TAG, "This is a warning log");
        Log.e(TAG, "This is an error log");
        Log.wtf(TAG, "This is a WTF!? log");
    }
}
```

## Primjer – activity\_main.xml

```
<Button
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/bLogThings"
    android:onClick="logThings"
    />
```



- Poruke u LogCatu moguće je filtrirati prema oznaci ili razini poruke, paketu, sadržaju poruke i PID-u



The Logcat tool window shows a list of log entries from the application 'hr.ferit.brunozoric.example\_2'. The entries are:

```
08-19 06:19:51.676 1303-1303/hr.ferit.brunozoric.example_2 I/Bruno: This is an info log
08-19 06:19:51.676 1303-1303/hr.ferit.brunozoric.example_2 D/Bruno: This is a debug log
08-19 06:19:51.676 1303-1303/hr.ferit.brunozoric.example_2 W/Bruno: This is a warning log
08-19 06:19:51.676 1303-1303/hr.ferit.brunozoric.example_2 E/Bruno: This is an error log
08-19 06:19:51.677 1303-1303/hr.ferit.brunozoric.example_2 E/Bruno: This is a WTF! log
```

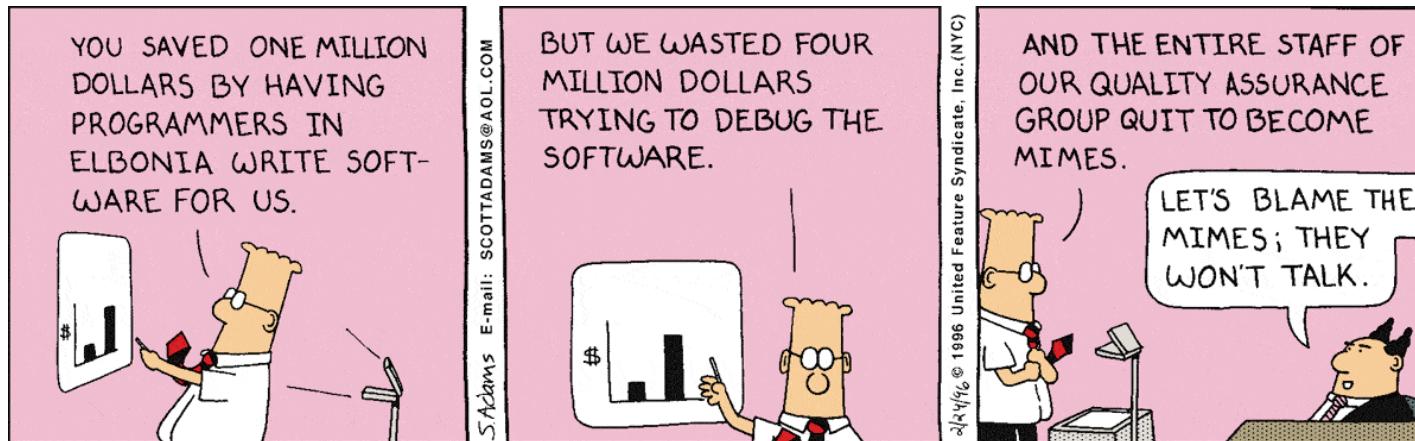
On the right side of the window, there is a filter configuration panel with the following options:

- Show only selected application
- Firebase
- No Filters
- Edit Filter Configuration
- tag\_Bruno

The 'tag\_Bruno' option is highlighted in blue.

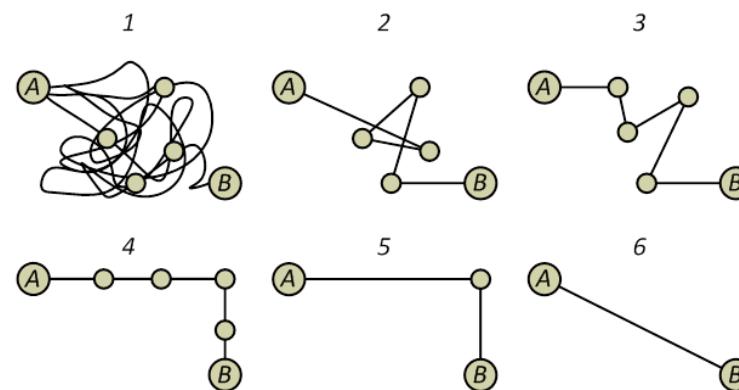
# ☰ Debugging

- Programiranje je, u osnovi, dodavanje bugova u prazan tekstualni dokument
- Bugovi su određeni problemi u programu koji ometaju njegov ispravan rad, a nužno ih je otkloniti
- Da bi se otklonili potrebno ih je prvo uočiti, zatim im pronaći uzroke u kodu te u konačnici prepraviti kod
- Problem:
  - *Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it. – B. Kernighan*
- Da bi bugove bilo lakše otkriti postoje određeni alati i koraci koji nam u tome mogu pomoći (npr. Lint, FindBugs, DDMS)



# ☰ Refaktoriranje (zašto volim Shift+F6)

- Kod nije isklesan u kamenu. Prilično ga je lako izmijeniti – ako imate pristojan IDE, niste skroz zabrljali dizajn
- Refaktoriranje je izmjena koda koji radi ispravno
- Ispravnost rada u idealnom svijetu proizlazi iz uspješno provedenih testova
- Cilj refaktoriranja je napisati kod čišćim
  - Kod ne pišete čitak računalu, pišete ga tako da budete čitak čovjeku (programeru koji dolazi iza vas, vjerojatno vama nakon 2 mjeseca)
  - Promjena imena
  - Osim čitkosti moguće je mijenjati dizajn (npr. zamijeniti ogroman switch polimorfizmom)
  - Moguće je raditi na performansama (npr. zamijeniti algoritam učinkovitijim)
- Više: Refactoring, Martin Fowler



# ≡ Work / Homework ☹

## Zadatak 1.

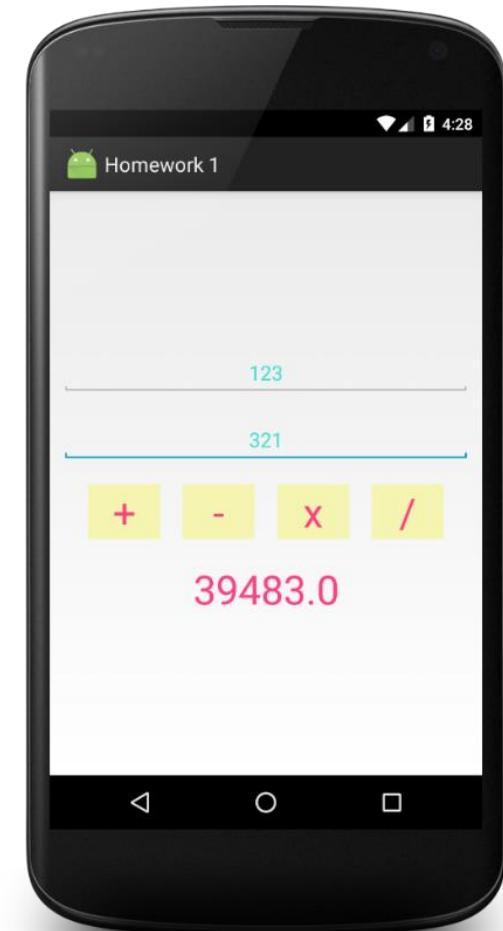
Debugirajte aplikaciju dostupnu na: [Aplikacija](#).

- Pronadite greške i ispravite ih
- Prepravite imena koja smatrate lošima
- Izvucite resurse iz koda
- Uvedite ButterKnife.

## Zadatak 2. Može li lakše?

Kreirajte aplikaciju koja omogućuje zbrajanje, oduzimanje, množenje i dijeljenje dva broja. Omogućiti unos brojeva s tipkovnice, a svaku od operacija definirajte na jednom gumbu.

- Izgled elemenata složiti u stilovima.
- Podržati hrvatski i engleski jezik
- Ispravno rukovati resursima
- U slučaju iznimke radi neispravnog unosa prikazati Toast poruku



# ☰ Work / Homework ☹

## Zadatak 4. Tko sam ja?

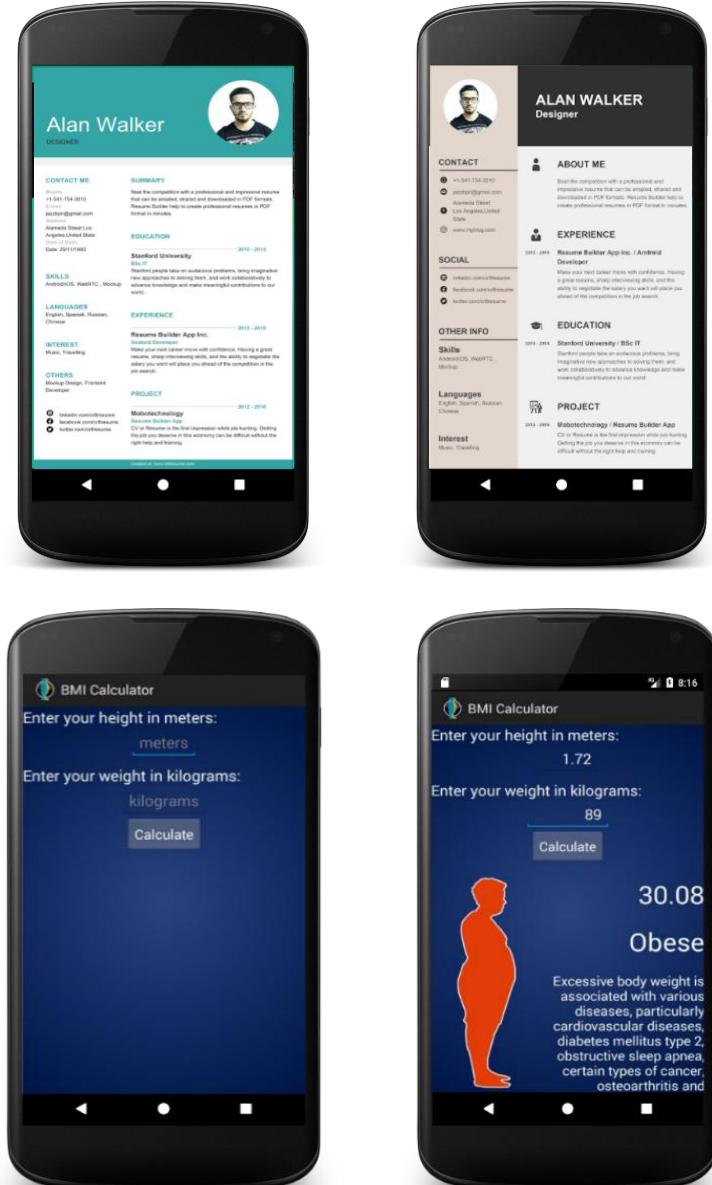
Koristeći sve naučeno, kreirajte aplikaciju s jednim Activityjem koja predstavlja Vaš CV (Dajte mašti na volju).

## Zadatak 4. Teške kosti.

Potrebno je kreirati aplikaciju za računanje indeksa težine. Ovaj se indeks računa prema izrazu:

$$\text{BMI} = \text{težina[kg]} / (\text{visina[m]})^2$$

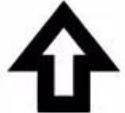
Nužno je iskoristiti dva polja za unos visine i težine, dok je rezultat potrebno prikazati u obliku slike, indeksa, naziva i kratkog opisa. Potrebno je dodati sve resurse koji se koriste u aplikaciji poput boja, dimenzija, stringova za svaki string u korisničkom sučelju, kao i za sve mogućnosti rezultata (Pothranjen, Zdrav, Debeo, Pretio). Potrebno je provjeravati korisnički unos, onemogućiti unos negativnih brojeva ili nula, slova, težine veće od 350kg i visine veće od 2.5m, kao i ekstrema u drugom smjeru. Također, kod za računanje BMI-ja ne treba biti u Activityu.



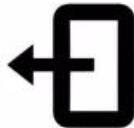
# In case of fire



1. git commit



2. git push



3. leave building

A spoonful of desserts

Android Developer Akademija – predavanje 3

READ

SHARE