

# OSNOVE RAZVOJA WEB I MOBILNIH APLIKACIJA



---

**LV5:**

Uvod u Android

---

# 1. Laboratorijska vježba 1

## 1.1. Uvod

U ovoj je vježbi prikazano kreiranje prve aplikacije za Android sustav. Razmotreni su njeni osnovni dijelovi, od osnovnih gradivnih elemenata do korisničkog sučelja (engl. *user interface*, UI). Kroz primjere je pokazana izgradnja UI-ja u XML opisnom jeziku (engl. *eXtensible Markup Language*) korištenjem različitih kontrola, povezivanje događaja poput klika na gumb s kodom koji ga obrađuje te pokretanje aplikacije.

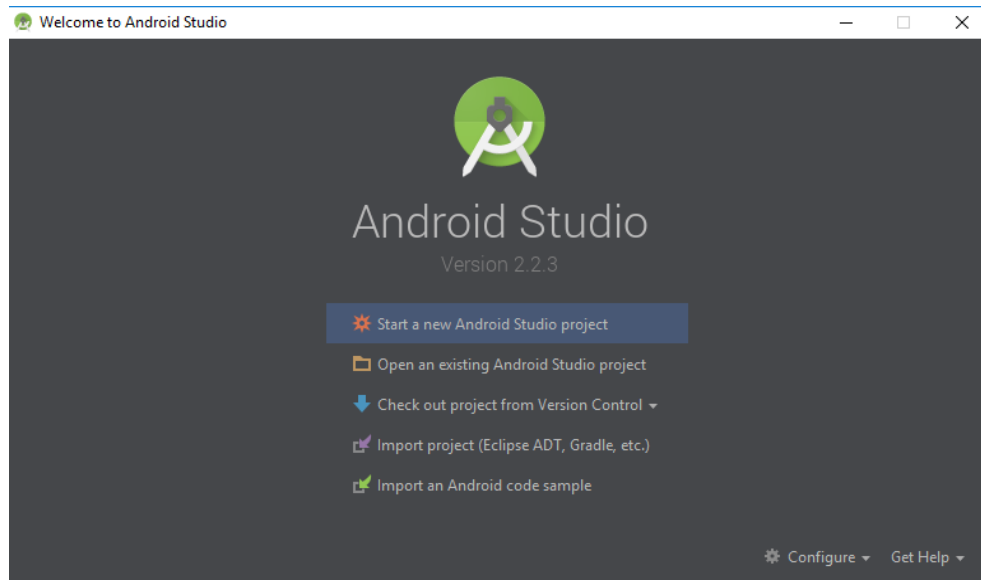
### 1.1.1. Potrebna predznanja

- 🤖 Osnove programiranja
  - Kolegiji Programiranje I, Programiranje II, Algoritmi i strukture podataka
- 🤖 Osnove objektno orijentiranog programiranja
  - Kolegij Objektno orijentirano programiranje
- 🤖 Osnove Java programskog jezika
  - H. Schildt, Java, A Beginner's Guide, 5th Edition
  - B. Eckel, Thinking in Java, 4th edition
  - <http://docs.oracle.com/javase/tutorial/>

### 1.1.2. Korisna predznanja

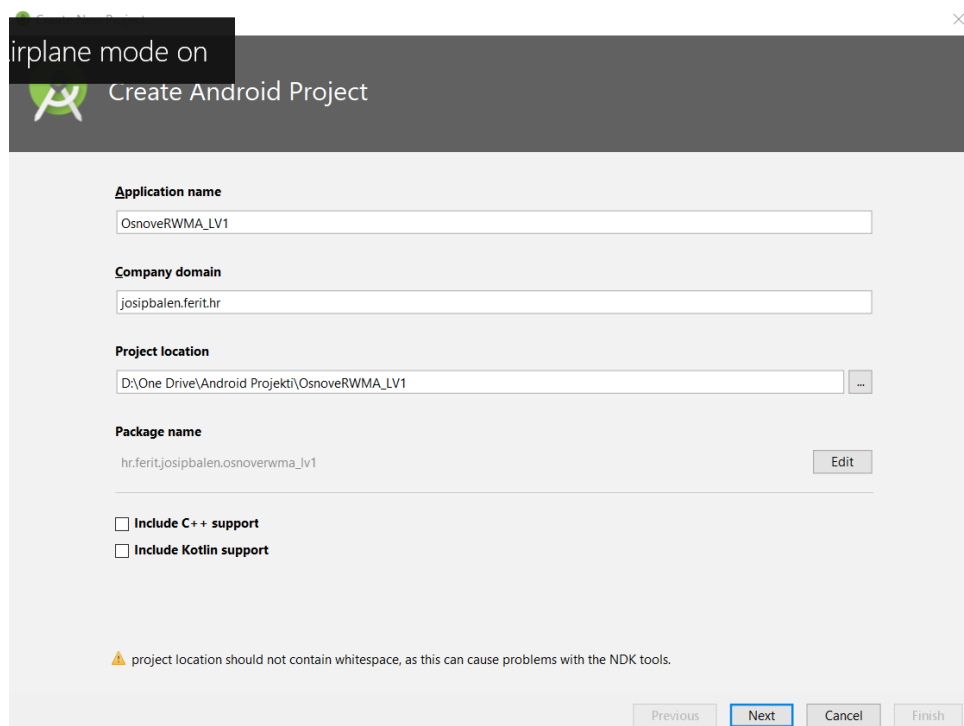
- 🤖 Input Controls, <http://developer.android.com/guide/topics/ui/controls.html>
- 🤖 Layouts, <http://developer.android.com/guide/topics/ui/declaring-layout.html>
- 🤖 Resursi, <http://developer.android.com/guide/topics/resources/index.html>
- 🤖 App manifest, <http://developer.android.com/guide/topics/manifest/manifest-intro.html>
- 🤖 Activity lifecycle, <http://developer.android.com/training/basics/activity-lifecycle>
- 🤖 Begginer's guide (I/O), <https://www.youtube.com/watch?v=yqCj83leYRE>
- 🤖 UI design patterns, <https://www.youtube.com/watch?v=M1ZBjICRfz0>
- 🤖 Design for UI Devs, <https://www.youtube.com/watch?v=Jl3-lzlzOJI>

## 1.2. Kreiranje Android projekta



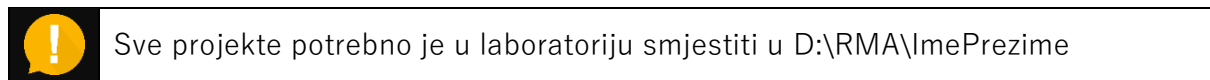
Slika 1.1. Stvaranje novog projekta

Kreiranje novog Android projekta moguće je obaviti iz glavnog izbornika kod pokretanja Android studija, kako je prikazano slikom 1.1. Nakon toga korisniku se prikazuje izbornik nalik onome na slici 1.2., koji omogućuje unos različitih detalja vezanih uz projekt. Prvenstveno se

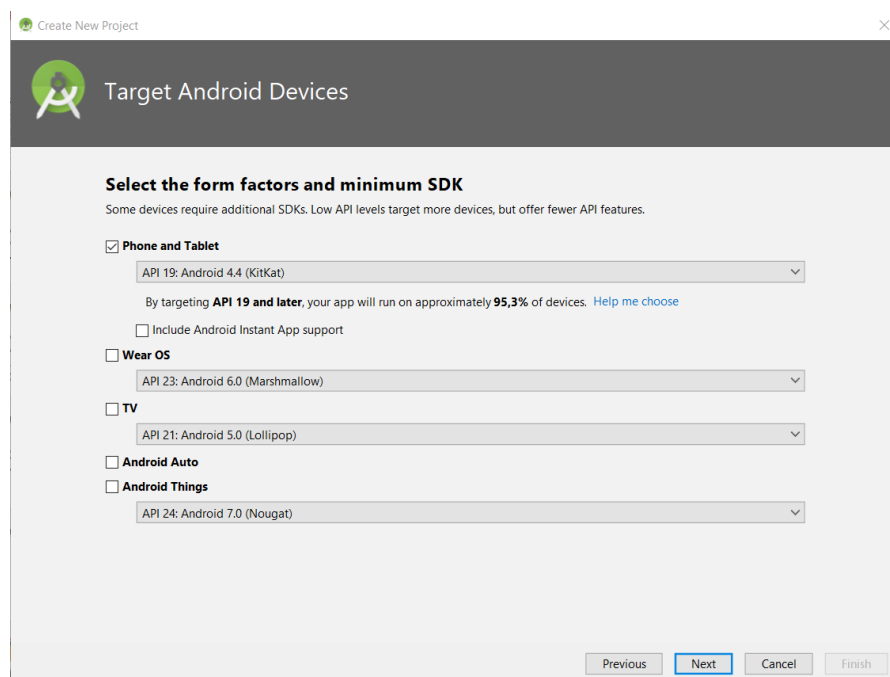


Slika 1.2. Stvaranje novog projekta

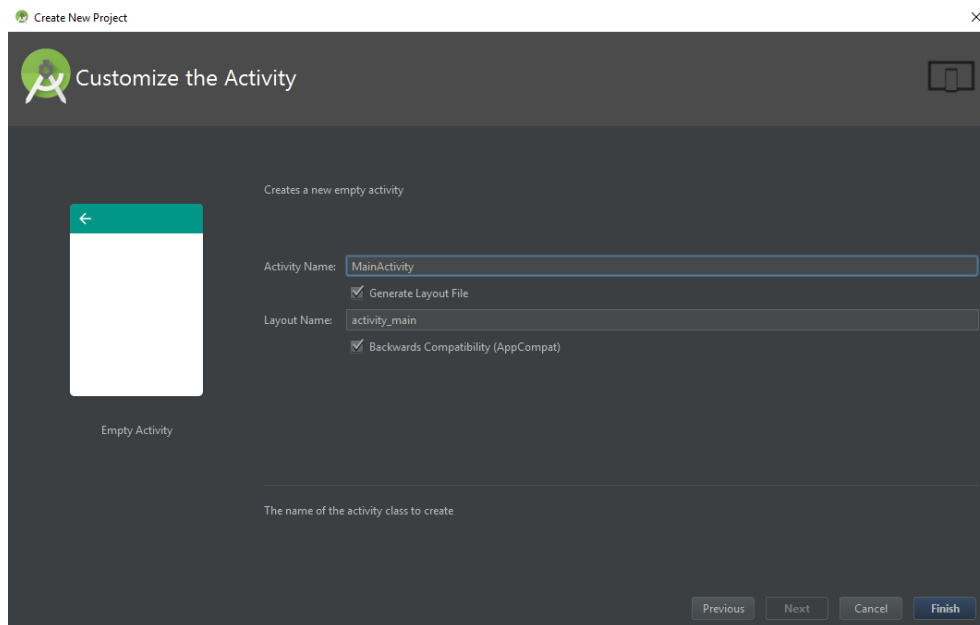
ovdje radi o nazivu aplikacije, nizu znakova koji će korisniku biti prikazani ispod ikone na uređaju, kao i na trgovini aplikacijama. Važno je reći kako ime aplikacije ne mora biti jedinstveno. Ono što mora biti jedinstveno je „Package name“. Naime, kod je kod Java projekata organiziran u pakete, a paket mora biti jedinstven kako bi bilo moguće razlikovati projekte, odnosno aplikacije. Ime paketa određuje se stoga upravo kao obrnuta domena, s ciljem olakšavanja održavanja jedinstvenosti. Ukoliko ne posjedujete domenu, za učenje ovo nije problem. Možete unijeti bilo koje ime paketa (npr. *com.example.josip*), a kasnije, ukoliko se odlučite na distribuciju možete razmišljati o jedinstvenosti. Za potrebe ovih vježbi rabit će se **imeprezime.ferit.hr**. Ovim načinom CD autora ovih redaka bit će **josipbalen.ferit.hr**, dok će ime paketa biti **hr.ferit.josipbalen.osnoverwma\_lv1**. Za potrebe ovih vježbi nije potrebno uključivati podršku za C++, a lokaciju projekta nužno je postaviti na **D:** disk, u mapu **RMA**, u mapu vlastitog imena.



Idući zaslon nudi izbor platforme za koju se želi razvijati aplikacija. Android je od originalnog sustava za digitalne kamere prerastao u operacijski sustav za širok spektar najrazličitijih uređaja, od telefona do automobila. Ukoliko se želi razvijati za pametne telefone, odabire se prva opcija i određuje se najniža razina (verzija) Androida koja se želi podržati, kako je



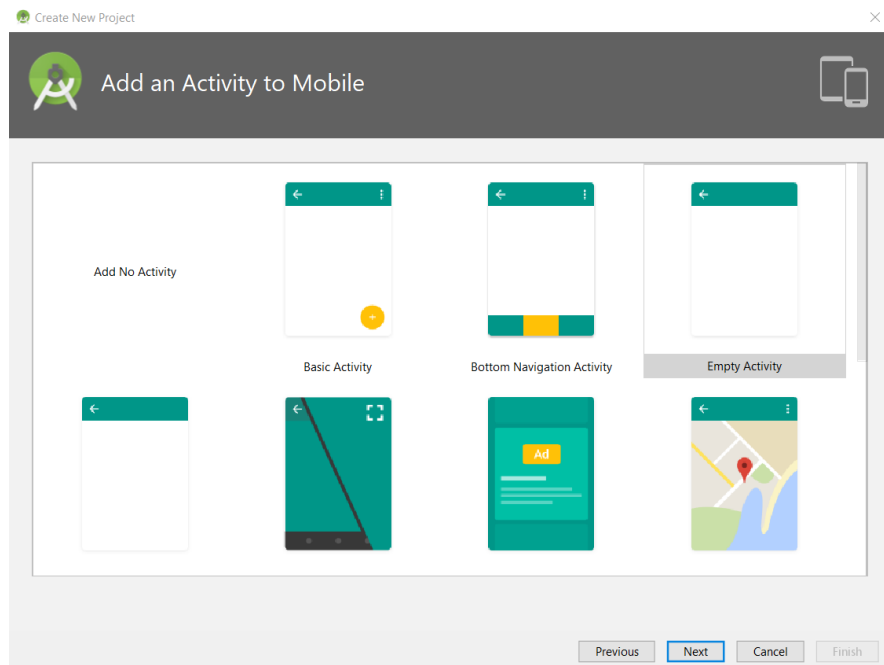
**Slika 1.3.** Stvaranje novog projekta – odabir platforme i inačice Androida



**Slika 1.4.** Stvaranje novog projekta – prazan Activity

prikazano slikom 1.3. Različite inačice Android sustava donosile su različite novitete koji u ranijim inačicama nisu u potpunosti podržani ili je potrebno na njih obratiti posebnu pozornost. Za potrebe ovih vježbi kao najniža inačica rabit će se API razine 21, odnosno najniža podržana verzija Androida bit će 5.0. (*Lollipop*). Ciljanjem navedene inačice pokriveno je otprilike 85.% uređaja, dok ostalima aplikacija neće biti vidljiva. Kod kreiranja novog projekta čarobnjak Android studija omogućuje dodavanje novog *Activitya*. *Activity* je nalik formi u klasičnom programiranju desktop aplikacija, a predstavlja jedan zaslon aplikacije. Ponuđeni su različiti predlošci, čak i mogućnost izostanka *Activitya*, a za potrebe ovih vježbi svi će novi projekti započinjati dodavanjem praznog *Activitya* (*Empty Activity*), kako je označeno slikom 1.4.

Dodavanjem praznog *Activitya*, IDE generira nužan kod (zapis u manifest datoteci, kreiranje nove klase itd.), no ipak je nužno unijeti neke osnovne informacije, kako je prikazano slikom 1.5. Naime, potrebno je reći naziv *Activitya* i naziv datoteke koja definira njegov izgled. Imenom *Activitya* određuje se ime klase koja predstavlja taj *Activity* i unutar koje će biti pisan kod, a s obzirom da Android sustav razdvaja funkcionalnost od prikaza, u datoteci koja definira izgled (*layout*) uporabom XML opisnog jezika bit će definirani svi vizualni elementi (kontrolne na korisničkom sučelju). Za sada je dovoljno ostaviti sve na podrazumijevanim postavkama. Klikom na *Finish* kreiran je (nakon kraćeg čekanja) prvi projekt koji ima osnovnu

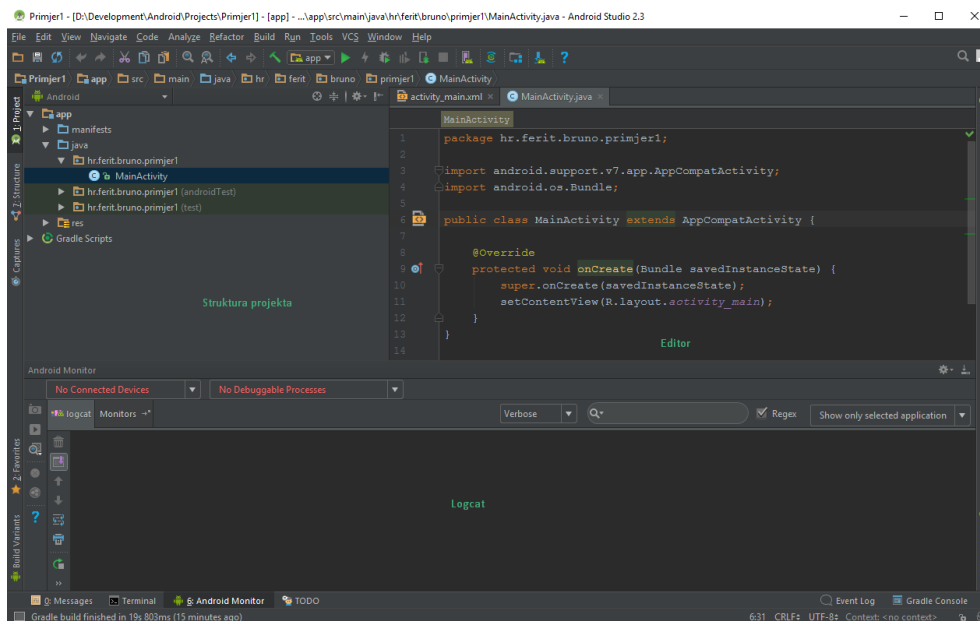


Slika 1.5. Stvaranje novog projekta – odabir predloška Activitya

funkcionalnost, ispisuje na zaslonu poruku „*Hello world*“. Izgled projekta u Android studiju prikazan je slikom 1.6.

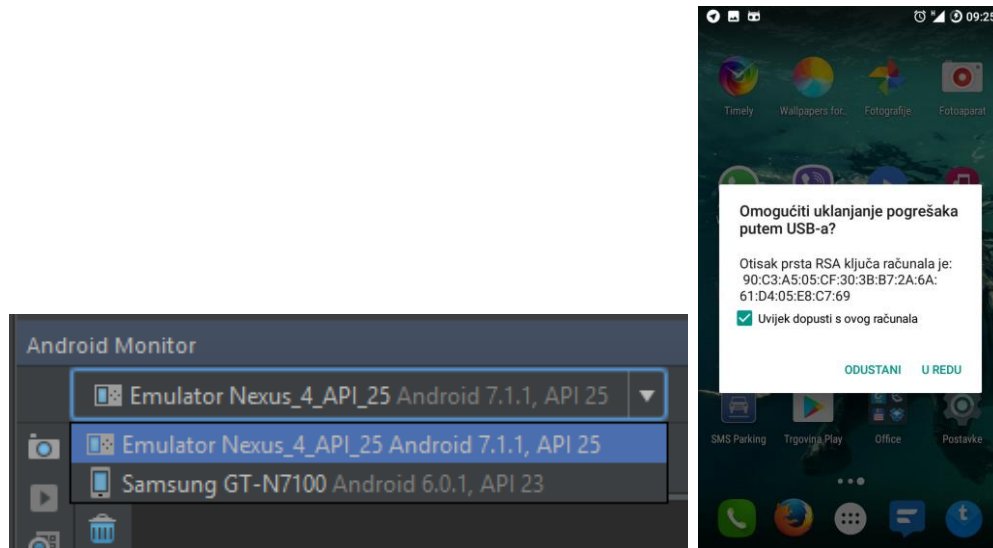
### 1.2.1. Pokretanje Android projekta

Da bi se Android projekt pokrenuo, potrebno je imati spojen i uključen uređaj. Ovdje može biti riječ ili o stvarnom ili o virtualnom uređaju. Svi spojeni uređaji dostupni preko ADB-a vidljivi



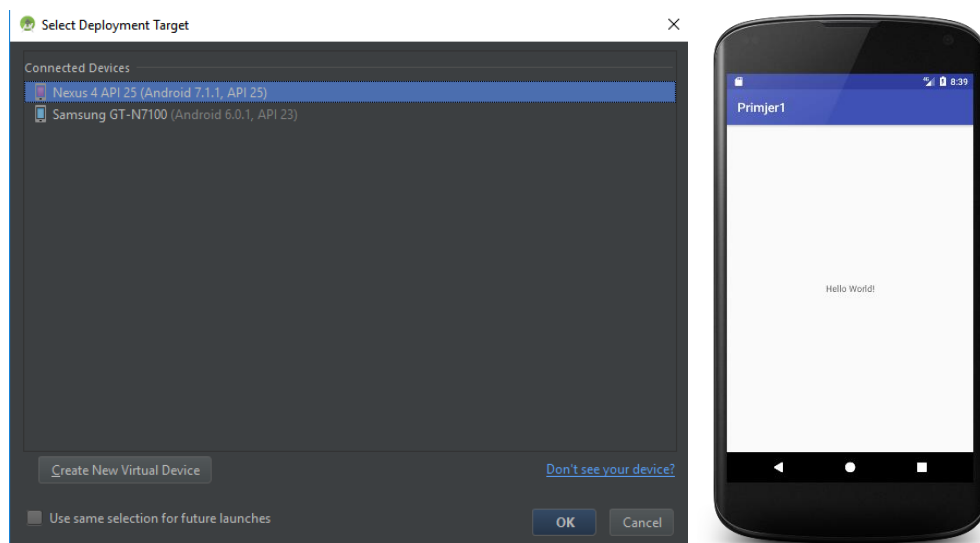
Slika 1.6. Stvaranje novog projekta – izgled projekta

su unutar Android monitora na dnu sučelja razvojnog okruženja, kako je prikazano slikom 1.7. Virtualni uređaji radit će bez ikakvih problema, dok je za korištenje fizičkih uređaja potrebno dati dozvolu na samom uređaju. Klikom na gumb u alatnoj traci ili pritiskom na kombinaciju tipku *Shift* i F10 pokreće se aplikacija i nudi se izbor uređaja na kojem ju se želi pokrenuti.



Slika 1.7. Odabir uređaja

Ovakvim pokretanjem koristi se podrazumijevana konfiguracija za pokretanje, dok je istu moguće urediti ili pak kreirati potpuno novu korištenjem *Run→EditConfigurations* izbornika. Slikom 1.8. prikazan je izbornik uređaja, a uređaj je moguće učiniti podrazumijevanim tako da



Slika 1.8. Pokrenuta aplikacija

svako iduće pokretanje aplikacije bude na istom uređaju. Aplikacija u radu prikazana je istom slikom.

Iako se cjelokupni postupak izrade aplikacije čini vrlo jednostavan, u pozadini se odvija velik broj koraka koje IDE prikriva, obavi i njima ne opterećuje programera. Sustav za izgradnju (engl. *build system*) odgovoran je za sve od prevođenja izvornog koda do pakiranja aplikacije, a čini to uporabom različitih alata kojima upravlja (prevoditelj, povezičak i sl.) te skripti za izgradnju. Sav izvorni kod prevodi se u Java class datoteke koje se zatim pretvaraju u .dex datoteke (Dalvik executable) uporabom alata *dx*. Ova se datoteka zatim zajedno sa svim resursima pakira uporabom alata AAPT pakiraju u .apk datoteku. Ova se datoteka u slučaju *debug* inačice potpisuje takozvanim *debug* ključem što omogućuje da ju se uporabom alata *adb* postavi i instalira na uređaj. Od verzije Androida 5.0 koristi se ART, pa postoje neke razlike u ranije opisanom postupku.



Više informacija o gradleu moguće je pronaći na

<https://rominirani.com/announcing-gradle-tutorial-series-5fd134223bf8#.gwyelvyc1>

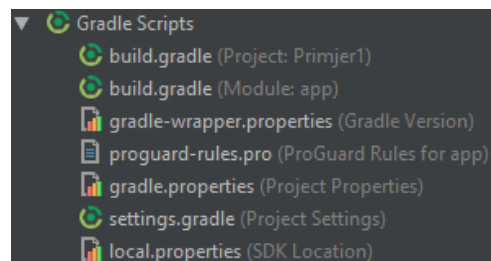
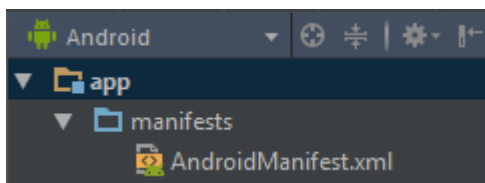
### 1.3. Manifest

Jedna od temeljnih datoteka unutar Android aplikacije jest Manifest datoteka, koja se nalazi unutar *app→manifests* mape u strukturi projekta. Riječ je o datoteci pisanoj XML opisnim jezikom koja sadrži metapodatke o aplikaciji, definira strukturu aplikacije, sve aktivnosti, servise, pružatelje sadržaja, dozvole, definira ikonu, verziju aplikacije i slično. Izgled osnovne manifest datoteke dan je primjerom 1. Pregledom datoteke uočava se osnovni manifest element koji govori da se radi o manifest datoteci. Gotovo svaki gradivni element aplikacije mora biti prijavljen i definiran u ovoj datoteci.



#### Primjer 1. – Manifest datoteka

Prikaz manifest datoteke, njena smještaja u strukturi projekta te njena sadržaja za osnovnu aplikaciju s jednim Activityem.





## Gradle skripta:

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.0"
    defaultConfig {
        applicationId "brunozoric.ferit.hr.primjer1"
        minSdkVersion 15
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
    }
}
```

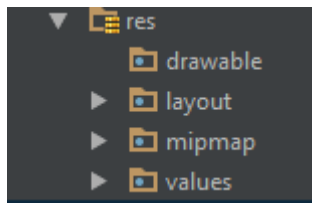
## AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="hr.ferit.bruno.primjer1">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="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>
```

Unutar oznake manifest smještaju se nužne informacije. Prvenstveno se ovoj oznaci pridodaje atribut *package* koji predstavlja ID aplikacije kao i mjesto gdje će biti smješten generirani kod, poput primjerice R klase s resursima. Važno je još jednom naglasiti kako na trgovini Play ne mogu postojati dvije aplikacije koje imaju isti ID aplikacije. Unutar oznaka manifest moguće je navesti i minimalnu podržanu verziju SDK kao i ciljanu verziju. Kod Android studija potonji se detalji navode u *gradle build* skripti. U istoj skripti navode se trenutna inačica aplikacije, i to na dva načina. Ime verzije predstavlja korisniku čitljiv niz znakova, dok kod verzije predstavlja cjelobrojnu vrijednost koja se ne prikazuje korisnicima, ali ju primjerice trgovina *Play* rabi za provjeru i ažuriranje aplikacije. Informacije o verziji mogu se navesti i u manifest datoteci. U manifest datoteci nalazi se potom oznaka *application* unutar koje se definira tema, ikona, te se registrira svaka komponenta aplikacije. U trenutnoj inačici postoji samo jedan *Activity*, a uočljivo je kako je dano ime (*.MainActivity*) te je uporabom *intent filtera* navedeno da se ova aktivnost pokreće prilikom pokretanja aplikacije. U slučaju da se nove komponente dodaju uporabom čarobnjaka Android studija, manifest datoteka bit će uglavnom automatski ažurirana.

## 1.4. Resursi

Kod razvoja aplikacija za Android platformu, kod se u pravilu razdvaja od izgleda, slika i drugih podataka koji se zajednički nazivaju resursima. Ovo razdvajanje u značajnoj mjeri olakšava izmjene, ali i prilagodbe različitim uređajima, tržištima i slično, jer je vrlo lako imati više različitih inačica istog resursa. Ove prilagodbe sustav će odraditi bez zahtjeva za dodatnim pisanjem koda ili intervencijama programera. Resursi za aplikaciju smješteni su u *res* mapi unutar projekta, a pakiraju se u .apk datoteku prilikom njena generiranja.



Slika 1.9. Resursi unutar Android aplikacije

### 1.4.1. Izgled

Izgled Android aplikacija definira se resursima koji se nazivaju *layout*. Riječ je o datotekama napisanim uporabom XML opisnog jezika unutar kojih se definiraju elementi korisničkog sučelja. Unutar aplikacije se ovi elementi „napuhuju“ (engl. *Inflate*) te se na temelju njih stvaraju objekti poput gumba, polja za unos i sl. sa svojstvima definiranim u *layout* datoteci. Svaki layout sadrži jedan korijenski element, koji mora biti ili *View* ili *ViewGroup* (derivat *View* klase, implementiraju *ViewManager* sučelje) objekt. U korijenski se element mogu zatim ugnijezditi drugi elementi, odnosno moguće je izgraditi hijerarhiju koja sačinjava UI. Čest slučaj predstavlja korištenje nekog objekta *ViewGroup* kao korijenskog elementa, s obzirom da je riječ o takozvanim *layout mangerima*. Odnosno, oni nude mogućnost smještanja i organizacije drugih *View* objekata unutar sebe.

#### 1.4.1.1. View

*View* je osnovna klasa koja predstavlja UI kontrolu. Iz nje su izvedene brojne korisne kontrole poput *TextViewa*, *EditTexta*, *ImageViewa* i slično. Osnovne i najčešće korištene klase koje su derivati *Viewa* bit će dane u nastavku, dok je za potpunu listu i opis potrebno pogledati u dokumentaciju. Kod dodavanja bilo kakvih *View* objekata u *layout* datoteku, obavezno je navesti njihove dimenzije. Dimenzije se mogu dati eksplicitno ili putem dviju vrijednosti koje ih prilagođavaju roditelju ili sadržaju, a riječ je o *match\_parent* i *wrap\_content*. Detaljnije će ovo biti prikazano u nastavku. Dodatno svojstvo koje se u pravilu dodjeljuje jest *id*. Uporabom


ovog svojstva moguće je programski pristupiti elementu sučelja, ali i unutar jednog resursa referencirati drugi (primjerice slika na gumbu), a dodaje se tako da se u XML-u svojstvo *id* postavi na „*@+id/some\_custom\_id*“.



Više informacija o *View* klasi i njenim derivatima moguće je pronaći na <https://developer.android.com/reference/android/view/View.html>

### a) **TextView**

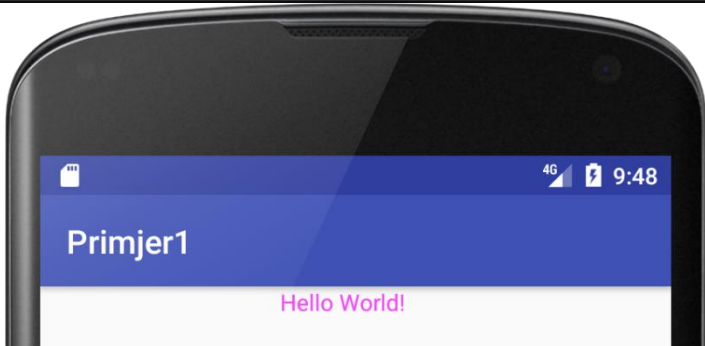
*TextView* je klasa koja predstavlja UI kontrolu za prikaz tekstualnog sadržaja koji se ne može izravno uređivati (zapravo može, ali samo u ograničenoj mjeri i to mu nije osnovna namjena). Ekvivalent je labeli iz desktop aplikacija. Sadrži brojna svojstva, među kojima je ključna *text*, a primjer njegova korištenja dan je primjerom 2.

Primjer 2. – TextView

U XML datoteku dodaje se oznaka `<TextView>`. Istu je moguće zatvoriti odgovarajućom zatvarajućom oznakom, ili koristiti `/>`. Parametrima *layout\_width* i *layout\_height* definiraju se dimenzije koje govore da širina treba biti jednaka širini roditelja (*View* objekta u koji je smješten *TextView*), dok se visina treba prilagoditi visini sadržaja. Svojstvo *gravity* govori da se tekst treba smjestiti u središte kontrole, dok svojstvo *text* određuje sadržaj. Ovakav način navođenja sadržaja nije dobar, već se tekst treba izdvojiti kao zaseban resurs, što će biti kasnije pokazano. **Napomena:** Ovaj *TextView* smješten je unutar linearnog *layouta*, što će biti pojašnjeno u kasnijem primjeru.

layout:


```
<TextView
    android:layout width="match parent"
    android:layout height="wrap content"
    android:text="Hello World!"
/>
```



### b) **EditText**

*EditText* je klasa koja nasljeđuje *TextView*, a riječ je o UI kontroli za unos alfanumeričkih znakova. Moguće ga je postaviti da dozvoljava samo određenu vrstu unosa (primjerice, samo

brojeve), da prikazuje zamaskirani unos lozinke (\*\*\*\*), da prikazuje prijedloge (engl. *hint*) itd. Primjer korištenja *EditText* kontrole dan je primjerom 3. Važna svojstva su *hint* koji definira naputak korisniku te *inputType* koji govori kakav će biti tip unosa podataka što će primjerice utjecati na prikaz ili na oblik tipkovnice koja će se ponuditi korisniku.

 Primjer 3. – EditText

U XML datoteku dodaje se oznaka `<EditText>`. Istu je moguće zatvoriti odgovarajućom zatvarajućom oznakom, ili koristiti `/>`. Parametrima *layout\_width* i *layout\_height* definiraju se dimenzije koje govore da širina treba biti jednaka širini roditelja (*View* objekta u koji je smješten *EditText*), dok se visina treba prilagoditi visini sadržaja. Svojstvo *hint* govori koji se tekst treba prikazati dok je kontrola prazna, dok svojstvo *inputType* određuje vrstu sadržaja. Kako je vidljivo sa slike, lozinka se ne prikazuje, dok je za unos telefonskog broja prikazana numerička tipkovnica. Kao i u prethodnom primjeru, kontrole su smještene u linerni *layout*.

layout:

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter password"
    android:inputType="textPassword"/>
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter phone number"
    android:inputType="numberDecimal"/>
```

### c) Button

Klasa *Button* predstavlja standardni gumb pritiskom na koji je moguće obaviti neku radnju. Stil gumba moguće je definirati unutar datoteke koja određuje stilove, ukoliko podrazumijevani

izgled iz bilo kojeg razloga nije zadovoljavajuće rješenje. Definiranje ponašanja moguće je postići na više različitih načina, neki od kojih su opisani u kasnijem poglavlju. Primjer korištenja gumba dan je primjerom 4.

 Primjer 4. – Button

U XML datoteku dodaje se oznaka `<Button>`. Istu je moguće zatvoriti odgovarajućom zatvarajućom oznakom, ili koristiti `/>`. Parametrima `layout_width` i `layout_height` definiraju se dimenzije koje govore da širina treba biti jednaka širini roditelja (*View* objekta u koji je smješten *Button*), dok se visina treba prilagoditi visini sadržaja. Svojstvo `text` govori koji se tekst treba prikazati na površini gumba, dok svojstvo `onClick` određuje metodu koja će biti pozvana pritiskom na gumb. Ova metoda mora imati točno određen potpis, a kao argument prima objekt *View* klase koji je generirao događaj. Kao i u prethodnom primjeru, kontrole su smještene u linerni *layout*.

layout:

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Destroy all humans"
    android:onClick="destroyAllHumans"
/>
```



#### d) *ImageView*

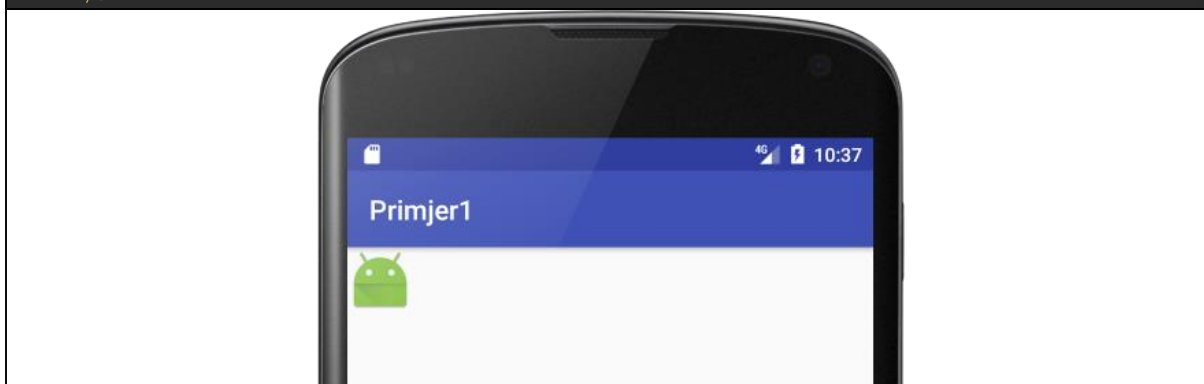
Klasa *ImageView* predstavlja UI kontrolu za prikaz slika na zaslonu. Nudi mogućnost prikaza slika iz različitih izvora, omogućuje njen prikaz u bilo kojem *layout* upravitelju te nudi brojne mogućnosti vezano uz prikaz slike. Ključan atribut u ovom slučaju je `src` koje govori iz kojeg se izvora dohvaća slika. Odabrana vrijednost u primjeru 5 govori da se prikazuje ikona aplikacije. Ona se nalazi u resursima, konkretno u mipmap mapi i zove se `ic_launcher`. Ovo se i navodi korištenjem `@` simbola i navođenjem identifikatora resursa, koji je u ovom slučaju njegovo ime.

 Primjer 5. – ImageView

U XML datoteku dodaje se oznaka `<ImageView>`. Istu je moguće zatvoriti odgovarajućom zatvarajućom oznakom, ili koristiti `/>`. Parametrima `layout_width` i `layout_height` definiraju se dimenzije koje govore da širina treba biti jednaka širini roditelja (*View* objekta u koji je smješten *ImageView*), dok se visina treba prilagoditi visini sadržaja. Svojstvo `src` govori koja se slika treba prikazati, dok svojstvo `contentDescription` određuje opis slike u slučaju da istu nije moguće prikazati. Svojstvo `scaleType` govori na koji način će slika biti skalirana u slučaju da njene dimenzije ne odgovaraju dimenzijama kontrole. Kao i u prethodnom primjeru, kontrole su smještene u linearni *layout*.

layout:

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:src="@mipmap/ic_launcher"
    android:contentDescription="The original app icon"
    android:scaleType="fitCenter"
/>
```




#### 1.4.1.2. ViewGroup

Klasa *ViewGroup* izvedena je iz klase *View*, a klase koje nju nasljeđuju su upravitelji *layouta*. Objekti ovih klasa omogućuju unutar sebe organiziranje i grupiranje drugih *View* (ili *ViewGroup*) objekata. Uz nekoliko opisanih klasa izvedenih iz klase *ViewGroup*, postoje i druge, dio kojih će biti prikazan u kasnijim vježbama (primjerice *RecyclerView*).

##### a) LinearLayout

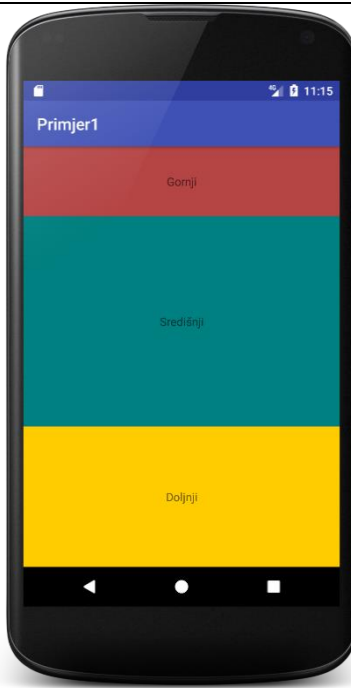
Linearni *layout* je jedan od najjednostavnijih i osnovnih *layouta*. Elementi se unutar ovog *layouta* slažu jedan ispod drugog, ako je orijentacija postavljena na vertikalnu, ili jedan pored drugoga, ako je orijentacija postavljena na horizontalnu. Ako je potrebno rasporediti elemente unutar ovog *layouta* tako da neki od njih zauzimaju određeni udio ekrana, to je moguće učiniti uporabom svojstva *weight*. Prikaz linearnog *layouta* i načina korištenja dan je primjerom 6.



Primjer 6. – Linearni layout

U XML datoteku dodaje se oznaka `<LinearLayout>`. Istu je moguće zatvoriti odgovarajućom zatvarajućom oznakom. Parametrima `layout_width` i `layout_height`

definišu se dimenzije koje govore da i širina i visina trebaju biti jednaki širini roditelja. S obzirom da je ovo korjenski element, on će zauzeti čitav ekran. Orijentacija je vertikalna, pa će svi elementi biti složeni jedan ispod drugoga. Svojstvo koje određuje težinu svakog elementa, i to tako da gornji zauzima  $1/(1+3+2)$  dijela ekrana, središnji  $3/6$ , a donji  $2/6$  dijelova ekrana. Kao što je jasno iz prethodno navedenog, svaki od elemenata uzima onoliko dijelova u ukupnoj sumi svih težina kolika je njegova težina.




layout:

```
<LinearLayout
    android:orientation="vertical"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Gornji"
        android:gravity="center"
        android:background="#b44545"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:text="Središnji"
        android:gravity="center"
        android:background="#008080"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:text="Doljni"
        android:gravity="center"
        android:background="#ffcc00"/>
</LinearLayout>
```

## b) RelativeLayout

Za razliku od linearnog, relativni *layout* omogućuje smještanje elemenata unutar sebe relativno u odnosu na druge elemente ili u odnosu na vlastite točke vezanja (primjerice, gore lijevo). Kako bi se pojedini elementi mogli smjestiti u odnosu na druge elemente koriste se njihovi identifikatori. Prikaz korištenja relativnog *layouta* dan je primjerom 7.



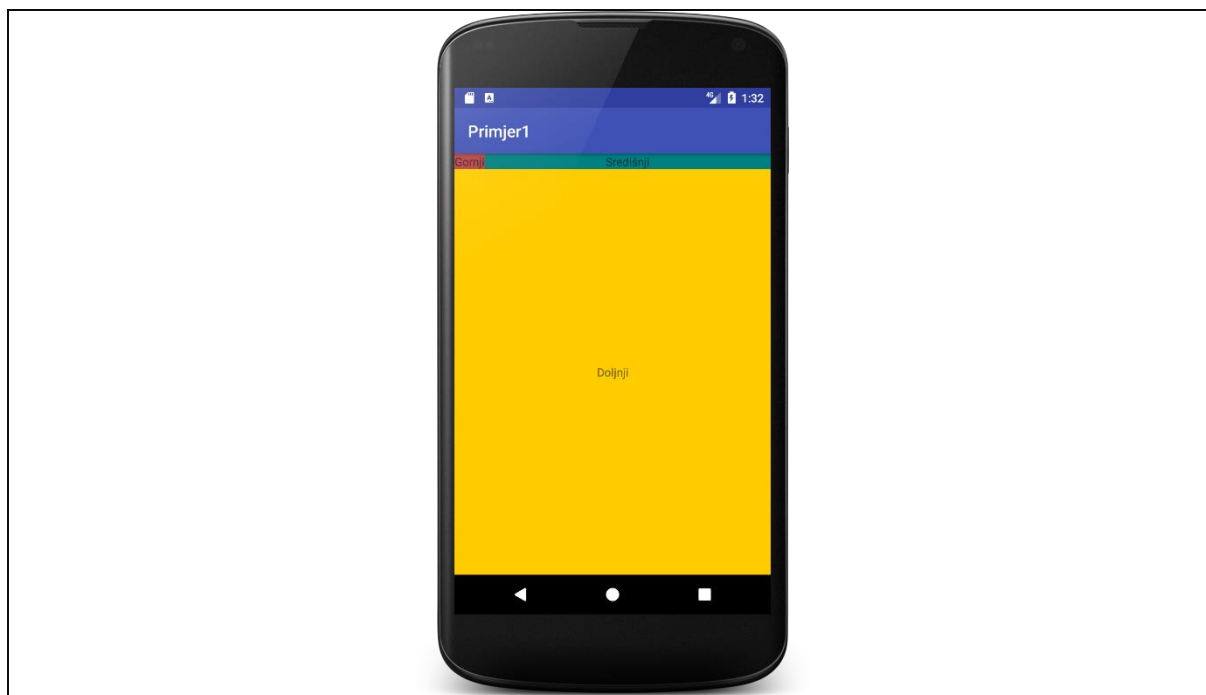
Primjer 7. – Relativni layout

U XML datoteku dodaje se oznaka `<RelativeLayout>`. Istu je moguće zatvoriti odgovarajućom zatvarajućom oznakom. Parametrima `layout_width` i `layout_height` definiraju se dimenzije koje govore da i širina i visina trebaju biti jednaki širini roditelja. S obzirom da je ovo korjenski element, on će zauzeti čitav ekran. Svakom se elementu zadaju točke roditelja u odnosu na koje se pozicionira, kao i odnosi prema drugim elementima. Da bi se navedeno postiglo, nužno je odrediti id svojstva za pojedine elemente.

Layout:

```
<RelativeLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/tvGornji"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:text="Gornji"
        android:gravity="center"
        android:background="#b44545"/>
    <TextView
        android:id="@+id/tvSredisnji"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/tvGornji"
        android:text="Središnji"
        android:gravity="center"
        android:background="#008080"/>
    <TextView
        android:id="@+id/tvDoljnji"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/tvSredisnji"
        android:text="Doljnji"
        android:gravity="center"
        android:background="#ffcc00"/>
</RelativeLayout>
```





### 1.4.2. Stringovi

Dobra praksa prilikom razvoja aplikacija jest razdvajanje stringova od koda i prikaza. Naime, česta je potreba distribuirati aplikaciju na različitim jezičnim područjima pa se stoga pojavljuje potreba za prevođenjem. Ako je sav tekst, uključujući onaj koji se pojavljuje na elementima UI-ja, izdvoji u posebnu datoteku, onda je dovoljno prevoditeljima dati samo tu datoteku i oni ne moraju brinuti o detaljima sučelja. Dodatno, programeri ne moraju voditi računa (osim u specifičnim situacijama) o učitavanju odgovarajuće skupine stringova, već će to učiniti sam sustav u ovisnosti o postavljenom *localeu*. U slučaju stringova, ali i drugih resursa poput boja, dimenzija i slika nije potrebno posebno dodavati id svojstvo, već se ime samog resursa rabi kao id. Primjer izdvajanja stringova u posebnu datoteku, ali i njihova referenciranja unutar *layouta* dan je primjerom 10.

</>

### Primjer 8. – Izdvajanje stringova kao resursa

Stringovi se izdvajaju kao resursi u strings.xml datoteku, smještenu u res/values mapu unutar hijerarhije projekta. Svaki je string smješten unutar oznaka `<string>` i `</string>`, a unutar otvarajuće oznake dodjeljeno mu je svojstvo *name*. Upravo ovo svojstvo predstavlja jedinstveni identifikator resursa

---

strings.xml

```
<resources>
  <string name="app_name">Primjer1</string>
  <string name="header message">Destroy all humans.</string>
</resources>
```

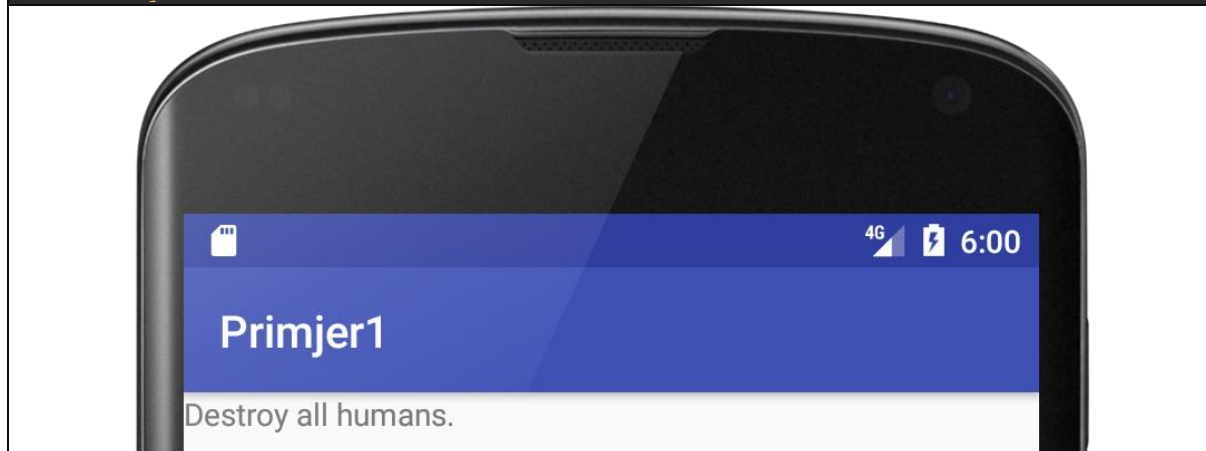
---

activity\_main.xml

```

<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/tvHeaderMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/header_message"/>
</LinearLayout>

```



### 1.4.3. Boje

Boje predstavljaju još jedan važan resurs koji je moguće izdvojiti. Ovim načinom moguće je rabiti boje prema smislenim imenima koje imaju značaj u kontekstu aplikacije, izbjegava se nečitak kod te je uvelike olakšana izmjena sheme boja u slučaju redizajna. Primjer korištenja boja dan je primjerom 11.



Moguće je pronaći i gotove xml datoteke s bojama, poput primjerice

<https://gist.github.com/kalehv/bae765c756e94455ed88>



### Primjer 9. – Izdvajanje boja kao resursa

Stringovi se izdvajaju kao resursi u colors.xml datoteku, smještenu u res/values mapu unutar hijerarhije projekta. Svaka je boja smještena unutar oznaka <color> i </color>, a unutar otvarajuće oznake dodjeljeno joj je svojstvo *name*. Upravo ovo svojstvo predstavlja jedinstveni identifikator resursa. Za kreirani projekt već su definirane neke boje, poput primjerice „colorPrimary“ i „colorAccent“, koje se koriste u resursnoj datoteci koja definira stil aplikacije.

colors.xml

```

<resources>
    <color name="colorBackground">#222d4a</color>
    <color name="colorNormalText">#f5c617</color>
</resources>

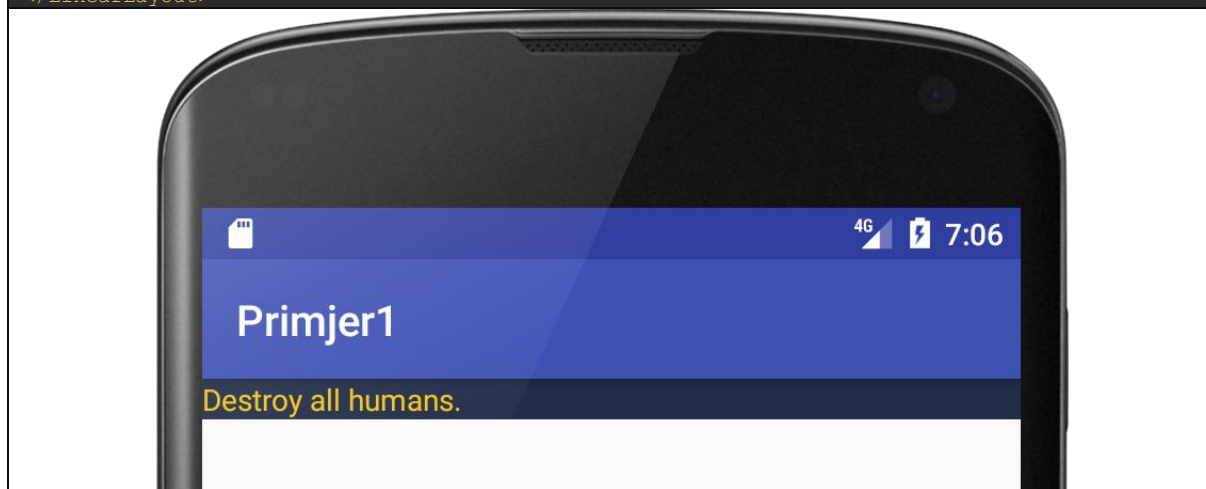
```

activity\_main.xml

```

<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/tvHeaderMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/header_message"
        android:background="@color/colorBackground"
        android:textColor="@color/colorNormalText"/>
</LinearLayout>

```



#### 1.4.4. Dimenzije

Dimenzije se u Android aplikacijama mogu definirati na nekoliko načina. Već su ranije spomenute dimenzije koje pojedini element prilagođavaju roditelju ili sadržaju, a riječ je o često korištenim izborima. Ukoliko se želi eksplicitno reći kolika je dimenzija nekog elementa, onda se to čini u jedinicama pod nazivom *density independent pixel* za sve vizualne elemente osim teksta, dok se za tekst koristi *scale independent pixel*. Iako je moguće rabiti i piksele (px) kao jedinicu, ovo se izbjegava i nikako ne preporučuje. Korištenje ranije navedenih jedinica osigurava da vizualni elementi izgledaju jednako na ekranima različitih gustoća piksela. Primjer korištenja ovih jedinica dan je primjerom 12.



#### Primjer 10. – Izdvajanje dimenzija kao resursa

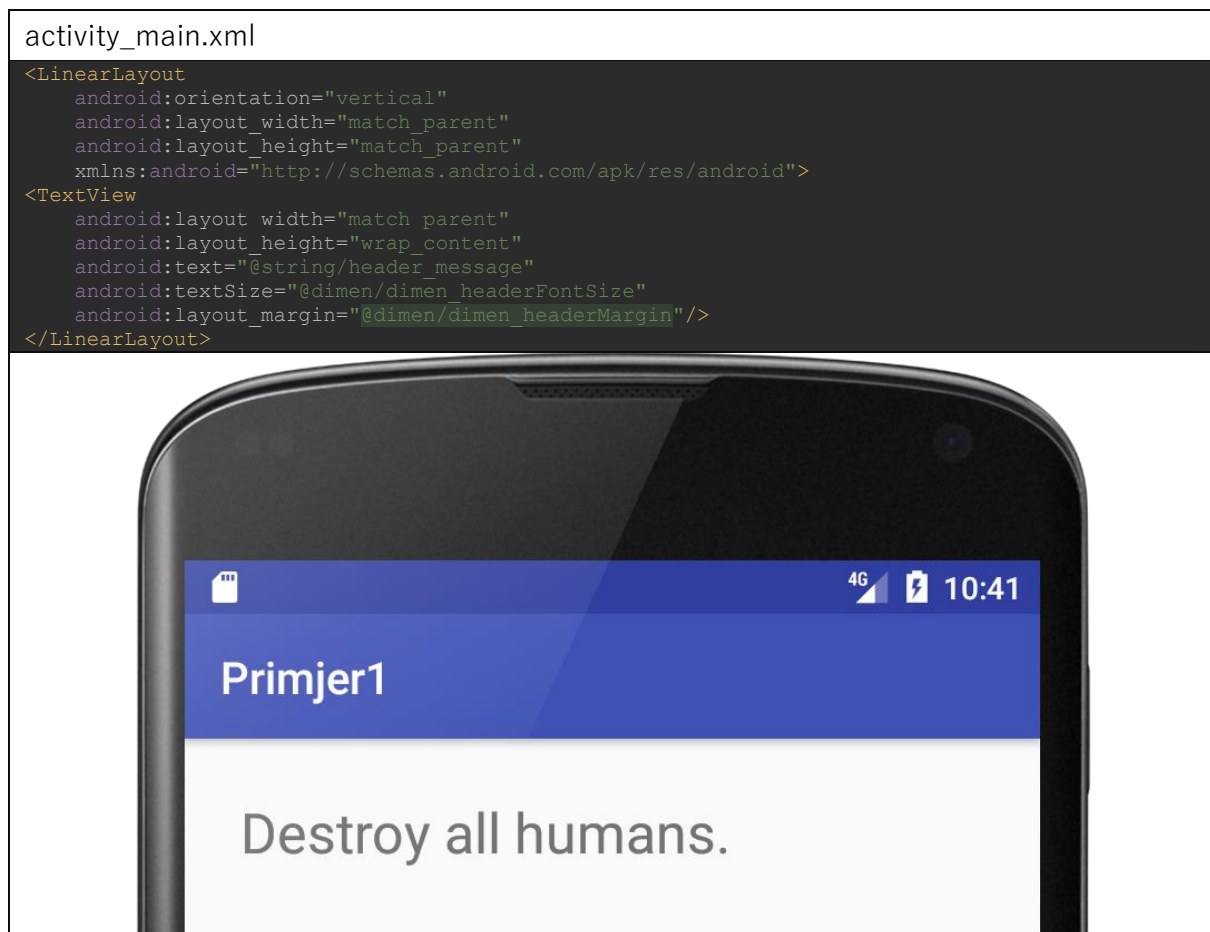
Dimenzije se izdvajaju kao resursi u xml datoteku proizvoljnog imena koju je nužno kreirati i smjestiti u res/values mapu unutar hijerarhije projekta. Svaka je dimenzija smještena unutar oznaka <dimen> i </dimen>, a unutar otvarajuće oznake dodjeljeno joj je svojstvo *name*. Upravo ovo svojstvo predstavlja jedinstveni identifikator resursa. Dimenzije se definiraju u dp i sp, ovisno o potrebi.

dimensions.xml

```

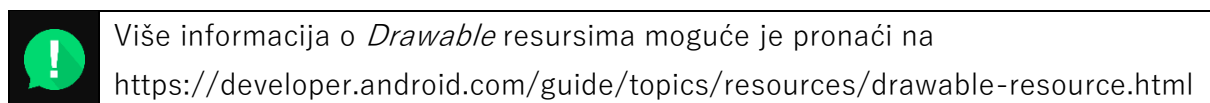
<resources>
    <dimen name="dimen_headerFontSize">25sp</dimen>
    <dimen name="dimen_headerMargin">25dp</dimen>
</resources>

```



#### 1.4.5. Slike

Drawables je zajednički naziv za slike koje se koriste u aplikaciji. Podržani rasterski formati su: JPEG, GIF, PNG i NinePatch (\*.9.png, rastezljivi PNG). Iako su podržani, dobra je praksa ne koristiti JPEG i GIF formate. Rasterske slike se uvijek kreiraju u nekoliko različitih veličina kako bi se podržale različite gustoće ekrana. Svaku sliku dobro je kreirati barem u 5 veličina i postaviti u odgovarajuće mape unutar hijerarhije projekta (mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi). U slučaju da za određenu rezoluciju nisu pruženi resursi, sustav će odabrati najbliži resurs i skalirati ga. Odnedavno je u Android studio uvedena i podrška za vektorsku grafiku, pa je kroz *Android Asset Studio* moguće kreirati grafičke resurse u vektorskom formatu. Dodatno, grafičke je elemente moguće kreirati i uporabom XML-a. Primjer korištenja slike koja dolazi kao dio Android SDK dan je primjerom 12.





Šalabahter za Android dizajnere

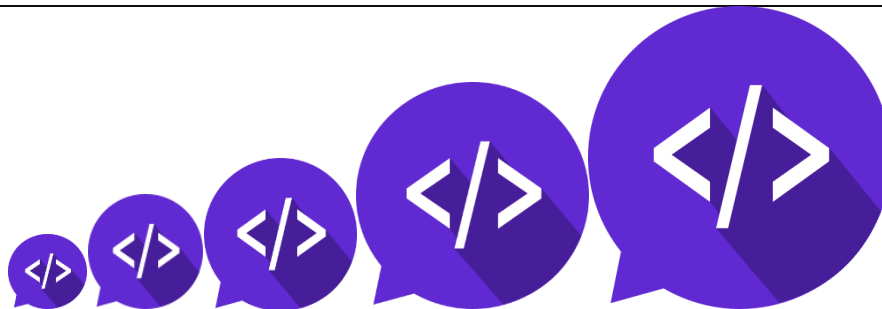
<https://possiblemobile.com/wp-content/uploads/2014/01/Android-Design-Cheat-Sheet.png>



## Primjer 11. – Korištenje drawable resursa

U programu za obradu slika GIMP kreirane su 4 inačice grafičkog resursa i postavljene su u odgovarajuće mape unutar hijerarhije projekta. Android će sam, već prema specifikaciji uređaja na kojemu se aplikacija pokreće odabrati i rabiti određeni grafički element. Uz vlastite grafičke elemente, kao dio SDK dolazi velik broj grafičkih elemenata, a prikazan je primjer korištenja i takvih elemenata. Kod kreiranja slike, mdpi je takozvani baseline, odnosno veličina koja se želi postići, hdpi je 1.5xmdpi, xhdpi je 2xmdpi, xxhdpi je 3xmdpi, a xxxhdpi je 4xmdpi.

Drawable: example.png



mdpi

hdpi

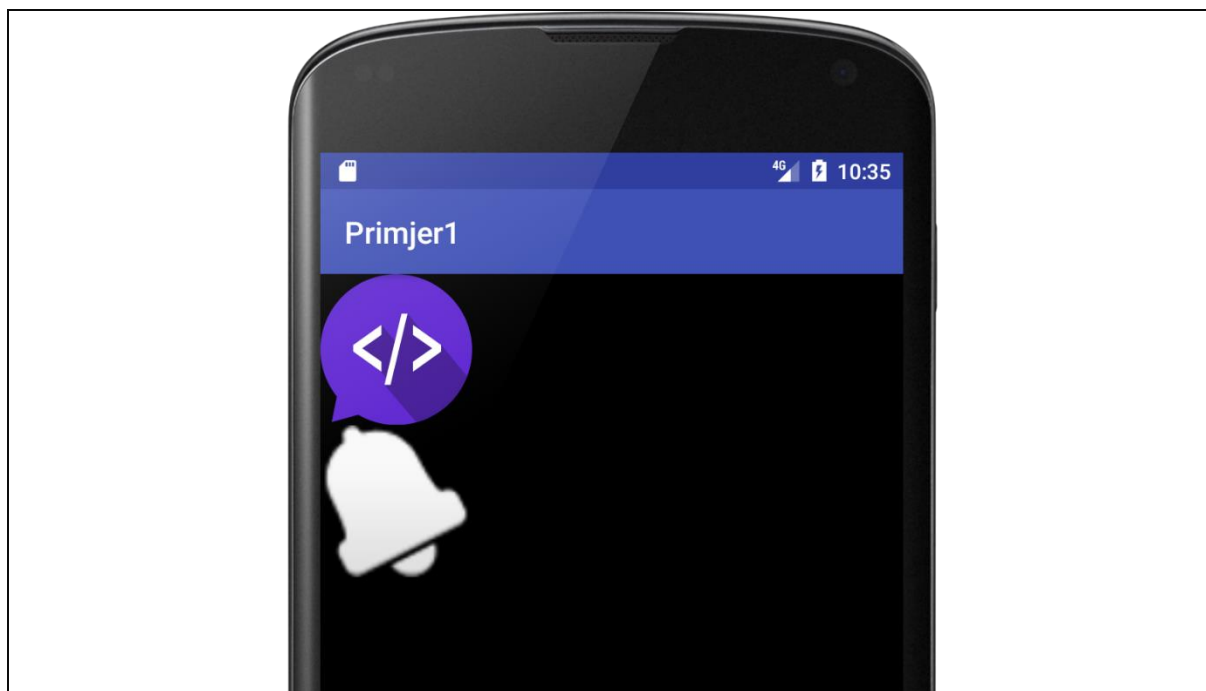
xhdpi

xxhdpi

xxxhdpi

activity\_main.xml

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/background_dark"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <ImageView
        android:id="@+id/ivCustomDrawable"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/example"/>
    <ImageView
        android:id="@+id/ivAndroidDrawable"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@android:drawable/ic_popup_reminder"/>
</LinearLayout>
```

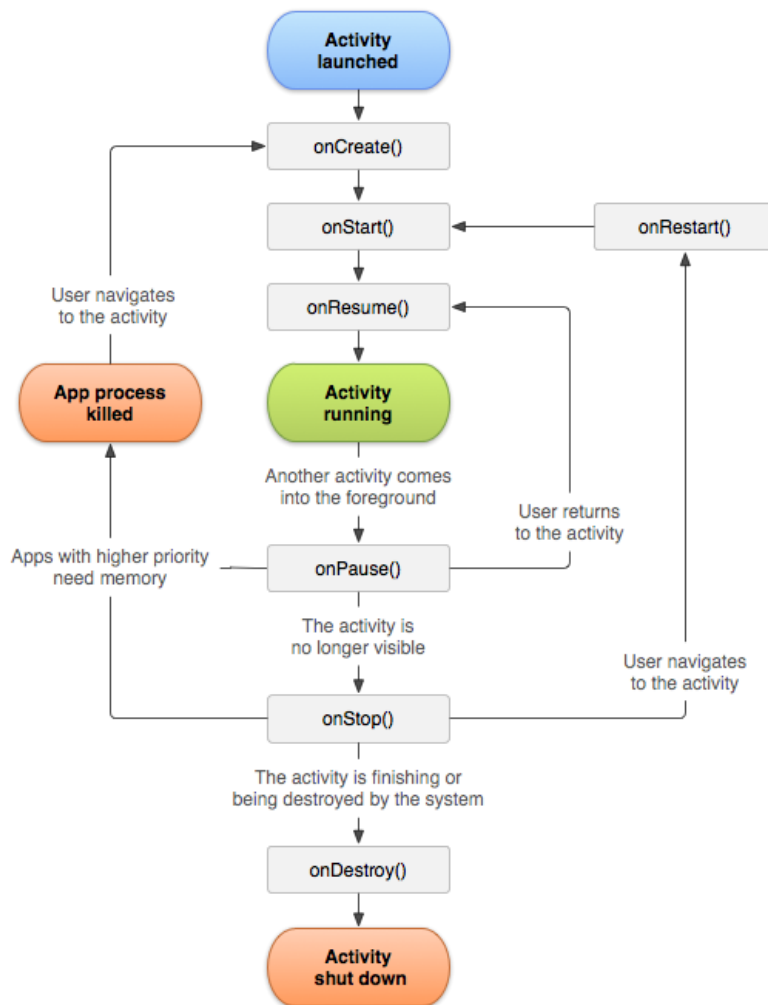


## 1.5. Kod

Sve do sada prikazano odnosilo se na baratanje XML datotekama i resursima. Iako sami nismo napisali niti jednu liniju koda, ostvareni rezultat je bila (donekle) funkcionalna Android aplikacija. Razlog tomu je odabir kod kreiranja projekta da se stvori i novi Activity, pri čemu je generirano nešto koda kako bi se korisniku prikazalo sučelje.

### 1.5.1. Activity

*Activity* je klasa koja predstavlja jedan zaslon aplikacije. Prvi *Activity* u ovoj je vježbi kreiran prilikom kreiranja projekta, a pri tome je automatski postavljen kao početni ekran koji se prikazuje pokretanjem aplikacije pritiskom na ikonu. Za razliku od klasičnih desktop aplikacija, Android aplikacije nemaju *main* metodu koja služi kao ulazna točka programa, već se životni ciklus aplikacije, odnosno pojedinih aktivnosti oslanja na određene *callback* metode. Ove se metode pozivaju primjerice kod stvaranja Activitya ili kod njegova gašenja, a metode životnog ciklusa su: *onCreate()*, *onStart()*, *onResume()*, *onPause()*, *onStop()*, *onRestart()* i *onDestroy()*. Životni je ciklus *Activitya* prikazan slikom 10. Budući da je sučelje odvojeno od koda, nužno ga je „napuhati“, odnosno nužno je iz XML opisa kreirati objekte koji predstavljaju elemente korisničkog sučelja. Ovo obavlja metoda *setContentView()* kojoj se kao argument daje resurs koji predstavlja sučelje za *Activity*.



Slika 1.10. Resursi unutar Android aplikacije (developer.android.com)



## Primjer 12. – Activity

Kreiran je osnovni *Activity*, a na sučelje su dodani po jedan *TextView* i *Button*. Kako je vidljivo iz prikaza koda, klasa koju smo kreirali zove se *MainActivity*, a pripada paketu *josipbalen.ferit.hr.osnoverwmalv1*. Ona nasljeđuje klasu *AppCompatActivity* kako bi se osigurala kompatibilnost sa starijim inačicama Androida, a sadrži samo jednu metodu, *onCreate()*. Ova se metoda poziva prilikom stvaranja *Activity*a, odnosno prilikom pokretanja aplikacije. *Bundle* objekt služi za pohranu prethodnog stanja sučelja, dok *setContentView()* metoda uzima kao argument resurs koji definira sučelje *Activity*a i iz XML-a stvara sve objekte sa zadanim svojstvima te omogućuje njihov prikaz na zaslonu.

strings.xml

```

<resources>
  <string name="app_name">Primjer1</string>
  <string name="bDestroyHumansText">Destroy all humans</string>
  <string name="tvHeaderMessageText">What should I do?</string>
</resources>

```

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:id="@+id/tvHeaderMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/tvHeaderMessageText"/>
    <Button
        android:id="@+id/bDestroyAllHumans"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/bDestroyHumansText"/>
</LinearLayout>

```

MainActivity.java:

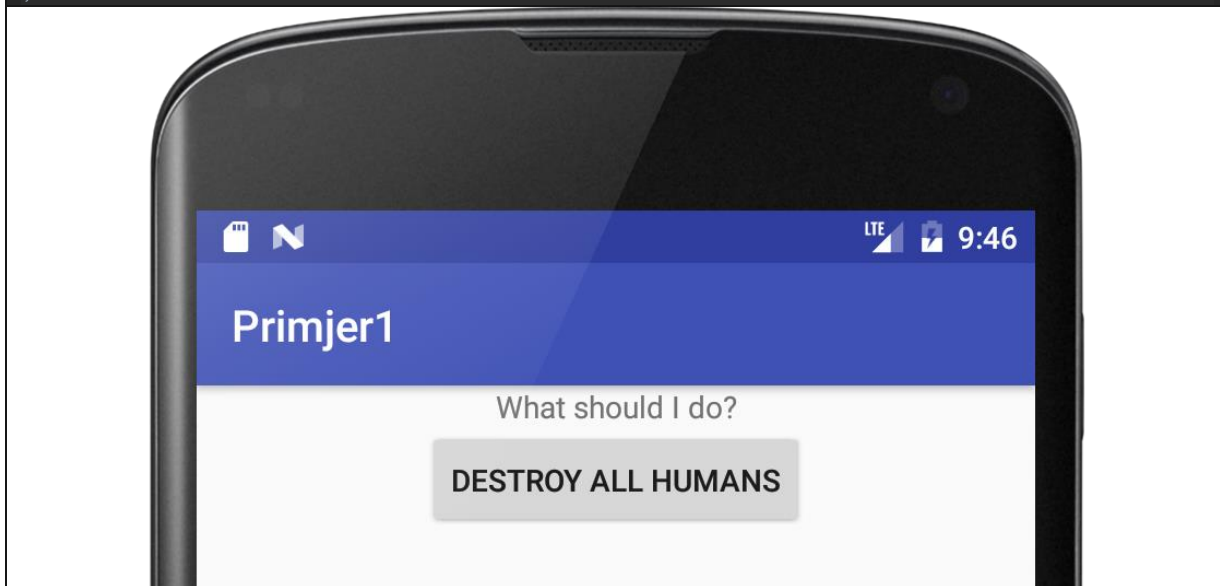
```

package brunozorric.ferit.hr.primjer1;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

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

```



### 1.5.2. Obrada događaja

Prethodni primjer pokazuje kako je gumb koji je smješten na sučelje nefunkcionalan. Iako je njegov izgled odgovarajući, a animacija klika očekivana – ništa se ne događa kod pritiska. Razlog tomu jest što je gumb nužno povezati s metodom koja će biti odgovorna za obavljanje posla, odnosno potrebno je oslušivati događaj i kada do njega dođe izvršiti određeni dio koda. Ovo se postiže implementacijom *onClickListener()* sučelja i postavljanjem objekta klase koja



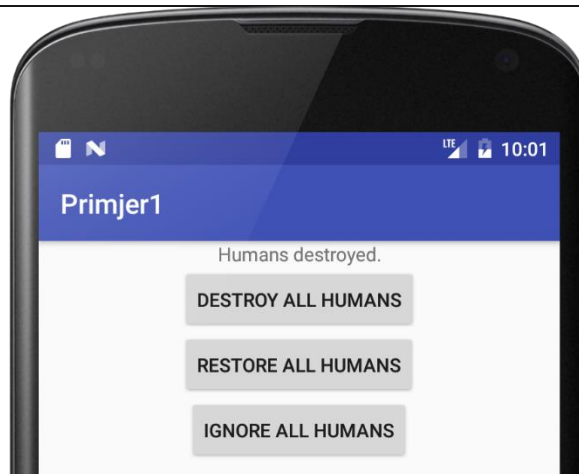
implementira navedeno sučelje kao osluškivača i to uporabom `setOnClickListener()` metode na gumbu. Za navedenu je potrebu najprije dohvatiti referencu na gumb iz koda što se postiže uporabom `findViewById()` metode kojoj se predaje id gumba, definiran u XML-u. Kako navedena metoda vraća objekt općenitije *View* klase, isti je nužno eksplicitno *castati* u objekt one klase kojoj on uistinu pripada – *Button*.



### Primjer 13. – Obrada događaja

Kreiran je osnovni *Activity*, a na sučelje su dodani po jedan *TextView* i *Button*. Dohvaća se referenca na gumb i postavlja se osluškivač, odnosno objekt neke klase koja implementira *onClickListener()* sučelje. To je moguće napraviti na tri načina:

- *Activity* implementira sučelje
  - `this.bDestroyAllHumans.setOnClickListener(this);`
  - Nakon toga poziva se `onClick()` metoda koja je automatski kreirana:
  - `public void onClick(View view){`
  - `this.tvHeaderMessage.setText(HUMANS_DESTROYED_MESSAGE);`
  - `}`
- Definiranje *OnClickListener* unutar poziva metode:
  - `this.bRestoreAllHumans.setOnClickListener(new View.OnClickListener() {`
  - `@Override`
  - `public void onClick(View view) {`
  - `tvHeaderMessage.setText(HUMANS_RESTORED_MESSAGE);`
  - `}`
  - `});`
  - `}`
- Definiranjem metode unutar XML-a koje će se pozvati prilikom klika na gumb:
  - `android:onClick="customOnClick"`
  - Metoda koja se poziva:
  - `public void customOnClick(View view){`
  - `this.tvHeaderMessage.setText(HUMANS_IGNORED_MESSAGE);`
  - `}`



strings.xml

```
<resources>
    <string name="app_name">Primjer1</string>
    <string name="bDestroyHumansText">Destroy all humans</string>
    <string name="bRestoreAllHumansText">Restore all humans</string>
    <string name="bIgnoreAllHumansText">Ignore all humans</string>
    <string name="tvHeaderMessageText">What should I do?</string>
</resources>
```

## MainActivity.java:

```
package brunozorric.ferit.hr.primjer1;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements View.OnClickListener{

    TextView tvHeaderMessage;
    Button bDestroyAllHumans, bRestoreAllHumans, bIgnoreAllHumans;

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

    private void initializeUI() {
        this.tvHeaderMessage = (TextView) findViewById(R.id.tvHeaderMessage);
        this.bDestroyAllHumans = (Button) findViewById(R.id.bDestroyAllHumans);
        this.bRestoreAllHumans = (Button) findViewById(R.id.bRestoreAllHumans);
        this.bIgnoreAllHumans = (Button) findViewById(R.id.bIgnoreAllHumans);
        this.bDestroyAllHumans.setOnClickListener(this); // Activity implements the interface
        this.bRestoreAllHumans.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                tvHeaderMessage.setText(R.string.bRestoreHumansText);
            }
        });
    }

    public void customOnClick(View view){
        this.tvHeaderMessage.setText(R.string.bIgnoreAllHumansText);
    }

    public void onClick(View view){
        this.tvHeaderMessage.setText(R.string.bDestroyAllHumansText);
    }
}
```

## 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:id="@+id/tvHeaderMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/tvHeaderMessageText"/>
    <Button
        android:id="@+id/bDestroyAllHumans"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/bDestroyHumansText"/>
    <Button
        android:id="@+id/bRestoreAllHumans"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/bRestoreAllHumansText"/>
    <Button
        android:id="@+id/bIgnoreAllHumans"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/bIgnoreAllHumansText"
        android:onClick="customOnClick"/>
</LinearLayout>

```

### 1.5.3. Preuzimanje teksta s *EditText* kontrole

Kako bi mogli preuzeti unesene vrijednosti u *EditText* kontroli potrebno je kreirati objekt tipa *EditText*, dohvatiti referencu na *EditText* iz koda što se postiže uporabom *findViewById()* metode kojoj se predaje id definiran u XML-u. Nakon toga se može tekst dobit pozivanjem metode *getText()* i pretvaranjem u *String* pomoću metode *toString()* kao što je prikazano ispod.

MainActivity.java:

```

public class MainActivity extends AppCompatActivity implements View.OnClickListener{

    EditText etInputText;

    private void initializeUI() {
        ...
        this.etInputText = (EditText) findViewById(R.id.etInputText);
        ...
    }
    ...
    public void onClick(View view) {
        int ageNo= Integer.parseInt(etInputText.getText().toString());
        this.tvHeaderMessage.setText("Your age is:"+ageNo);
    }
    ...
}

```

### 1.5.4. Upravljanje slikama

Slikama je moguće upravljati na različite načine, kroz XML ili direktno u kodu. U primjeru ispod je pokazano kako je moguće klikom na pojedinu sliku (ili neki drugi UI element) ju sakriti (INVISIBLE), ukloniti s ekrana (GONE) ili prikazati (VISIBLE).

MainActivity.java:

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
    ImageView ivPicture;

    private void initializeUI() {
        ...
        this.ivPicture = (ImageView) findViewById(R.id.ivPicture);
        ...
    }
    ...
    this.ivPicture.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view){
            ivPicture.setVisibility(View.INVISIBLE);
        }
    });
    ...
}
```

activity\_main.xml: `android:visibility="invisible"`

### 1.5.5. Toast poruke

Obavijesti se na Android platformi korisniku pružiti na različite načine, a jedan od najčešćih je korištenje kratkotrajnih tekstualnih poruka koje se nazivaju *Toast*. S ovim porukama ne postoji mogućnost interakcije, a kreiraju se korištenjem statičkih metoda *Toast* klase, kako je dano primjerom 16.



#### Primjer 14. – Toast poruka

Kako je uočljivo iz koda danog u klasi *MainActivity.java*, kreirana je posebna metoda za prikaz *Toast* poruka kojoj se predaje poruka u obliku objekta *String* klase. Metoda se poziva prilikom klika na gumb, a unutar iste se poziva *makeToast()* statička metoda. Kao argumenti daju joj se kontekst (*Activity* jest kontekst, pa je moguće dati *this* referencu), poruka te trajanje. Kao trajanje se koristi jedna od preddefiniranih konstantnih vrijednosti unutar *Toast* klase. U konačnici je nužno pozvati i metodu *show()* kako bi poruka uistinu i bila prikazana.

strings.xml

```
<resources>
    <string name="app_name">Primjer1</string>
    <string name="bDestroyHumansText">Destroy all humans</string>
    <string name="tvHeaderMessageText">What should I do?</string>
</resources>
```

## 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:id="@+id/tvHeaderMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/tvHeaderMessageText"/>
    <Button
        android:id="@+id/bDestroyAllHumans"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/bDestroyHumansText"/>
</LinearLayout>
```

## MainActivity.java

```
package brunozorric.ferit.hr.primjer1;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements View.OnClickListener{

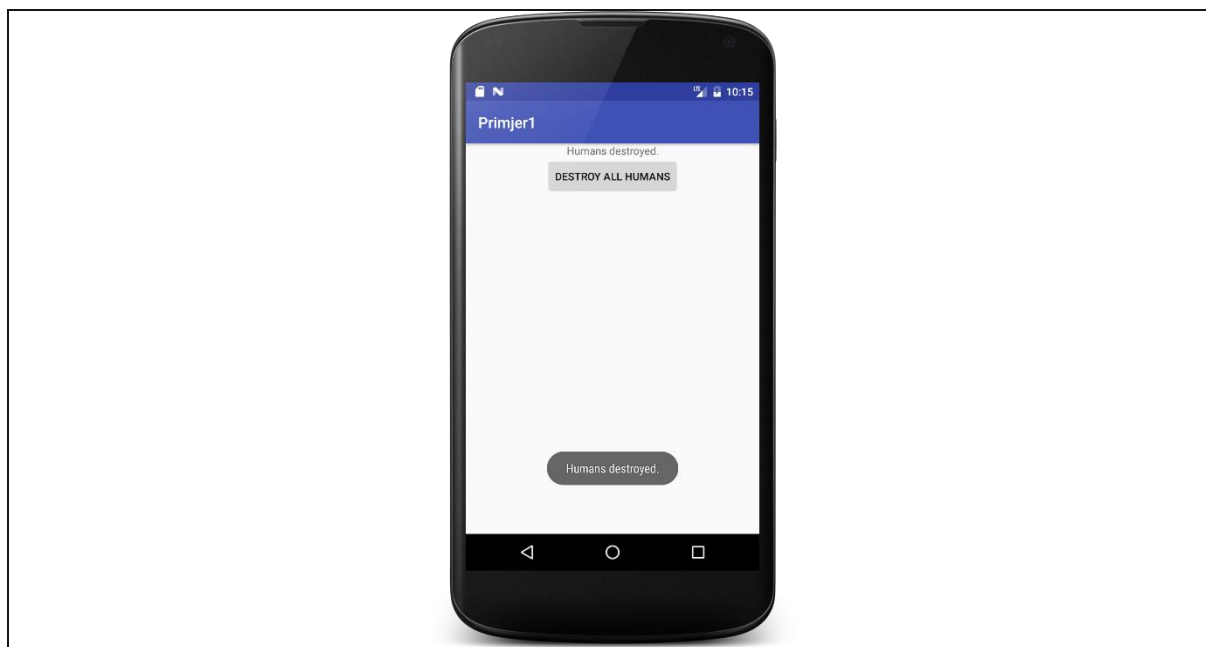
    private static final String HUMANS_DESTROYED_MESSAGE = "Humans destroyed.";
    TextView tvHeaderMessage;
    Button bDestroyAllHumans;

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

    private void initializeUI() {
        this.tvHeaderMessage = (TextView) findViewById(R.id.tvHeaderMessage);
        this.bDestroyAllHumans = (Button) findViewById(R.id.bDestroyAllHumans);
        this.bDestroyAllHumans.setOnClickListener(this); // Activity implements the interface
    }


    public void onClick(View view){
        this.tvHeaderMessage.setText(HUMANS_DESTROYED_MESSAGE);
        this.displayToast(HUMANS_DESTROYED_MESSAGE);
    }

    private void displayToast(String message) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }
}
```



### 1.5.6. Log poruke

Poruke *Toast* koriste se kada je nužno korisnika obavijestiti o nečemu. Kada je nužno programera obavijestiti o nečemu, one i nisu najbolji izbor. Pri razvoju Android aplikacija moguće je rabiti *Log* klasu te njene statičke metode koje omogućuju ispis prikaz poruka različite razine važnosti. Poruke se prikazuju u *logcatu*, a podržane su razine verbose - `Log.v()`, debug - `Log.d()`, info - `Log.i()`, warning - `Log.w()`, error - and `Log.e()` i what a terrible failure – `Log.wtf()`. Dobra je praksa definirati vlastite oznake (engl. *tag*) kako bi se mogli stvoriti filteri poruka.

	Primjer 15. – Log poruka
U bilo kojem trenutku moguće je iskoristiti neku od ranije navedenih metoda kako bi se zabilježila poruka. Kreirana je i vlastita oznaka kako bi se poruka razlikovala od ogromnog broja poruka koje sustav stalno generira.	
MainActivity.java	

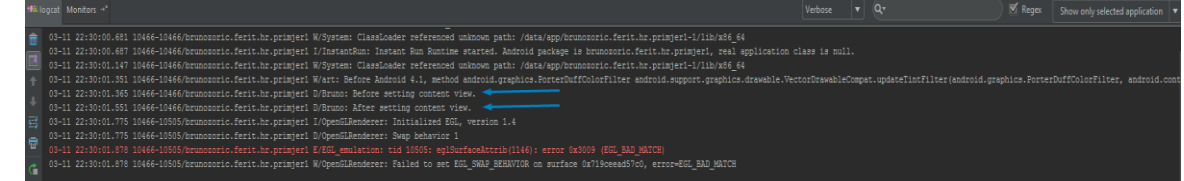
```
package brunozorric.ferit.hr.primjer1;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "Bruno";

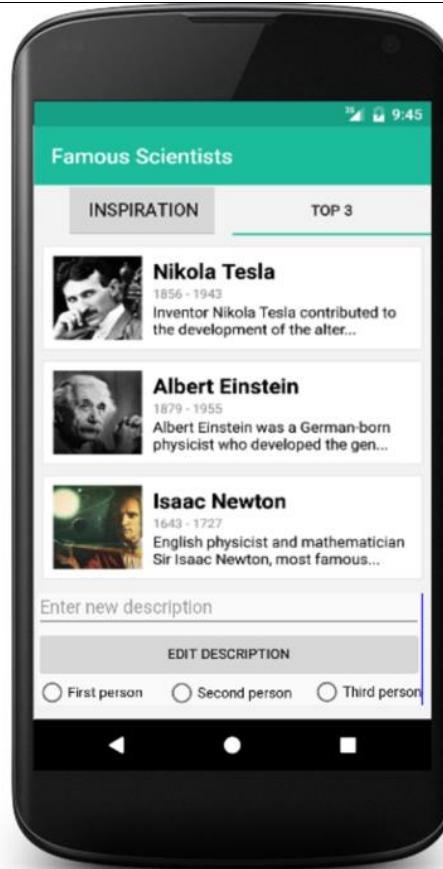
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, "Before setting content view.");
        setContentView(R.layout.activity_main);
        Log.d(TAG, "After setting content view.");
    }
}
```



## 1.6. Zadaća

	Zadaća 1. HaloOfFame - Aplikacija o inspirirajućim osobama/sportašima
	<p>Korištenjem naučenoga na ovoj vježbi, kreirajte jednostavnu aplikaciju koja nudi informacije o tri osobe/sportaša koji Vas inspiriraju i za koje smatrate da su imale važan utjecaj. Za svaku osobu potrebno je staviti sliku, datum rođenja/smrti i kratak životopis/statistiku. Definirati klik na sliku koji ispisuje neku činjenicu iz njegovog života) u obliku <i>Toast</i> poruke, što je moguće spremiti u XML datoteku u obliku polja stringova. Aplikaciju je potrebno testirati na barem dva uređaja (stvarna ili virtualna), te osmisлити ikonu. Potrebno je riješiti zadaću tijekom LV-a i pokazati ju asistentu koji će vam na temelju kvalitete i završenosti izrade dodijeliti bodove.</p>
	<ul style="list-style-type: none"><li>🤖 Kreirajte novi Android projekt s jednim Activityem</li><li>🤖 Sučelje definirati u XML-u, tako da sadrži jedan relativni <i>layout</i> u kojem su složeni ostali elementi.</li><li>🤖 Napisati metodu koja barata klikom na pojedinu sliku i potpuno ju uklanja s ekrana</li><li>🤖 Dodati jedan Button „Inspiration“ koji će u obliku <i>Toasta</i> ispisati <i>random</i> činjenicu iz života jednog od ta tri sportaša</li><li>🤖 Dodati RadioButton koji će označavati pojedinu osobu</li></ul>

- Dodati jedan *Button* i *EditText* u koji se može unijeti neki tekst i onda izmijeniti postojeći klikom na *Button* „Edit description“ u ovisnosti o odabranom *RadioButton*u



### Napomena:

Svaka je nadogradnja poželjna i dobrodošla. Korištenje znanja koja nadilaze opseg vježbi se potiče.