MOSIS - Laboratorijska vežba 1

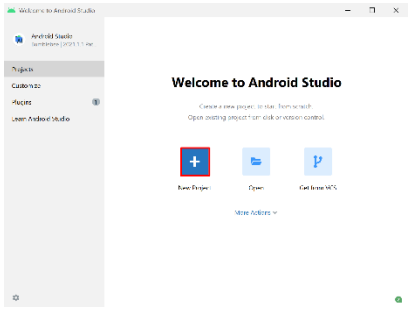
Laboratorijske vežbe iz predmeta MOSIS su koncipirane tako da prate računske vežbe i služe za proveru koncepata programiranja aplikacija namenjenih Android operativnom sistemu koji su na njima obrađeni. Imajući u vidu da je Android kompletan operativni system, teško je obraditi sve koncepte koji se prilikom programiranja aplikacija javljaju u toku jednog semestra, tako da će pažnja biti usmerena na one ključne dok se za naprednije mora konsultovati literatura (npr. ona preporučena na Moodle stranici kursa).

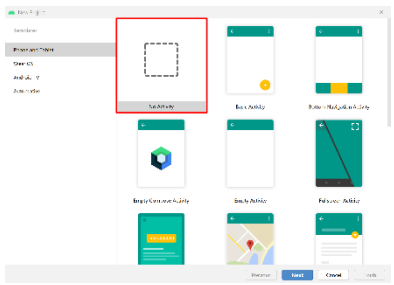
Laboratorijska vežba 1 je uvodna i namenjena je upoznavanju sa osnovnim konceptima programiranja za ***Android*** operativni sistem namenjen „pametnim telefonima“.

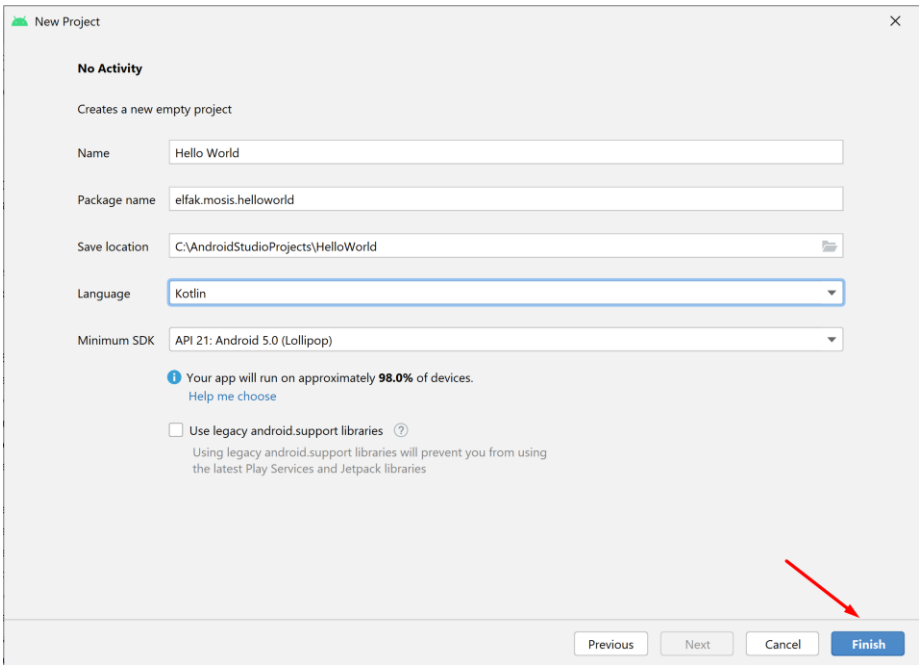
Ciljevi ove vežbe su:

1. Upoznavanje sa *Android Studio* razvojnim okruženjem
2. Kreiranje Hello World Android projekta
3. Upoznavanje sa strukturom i bitnim fajlovima android projekta
4. Kreiranje emulatora korišćenjem *AVD* menadžera
5. Pokretanje aplikacije
6. Provera i upoznavanje sa životnim ciklusom aplikacije

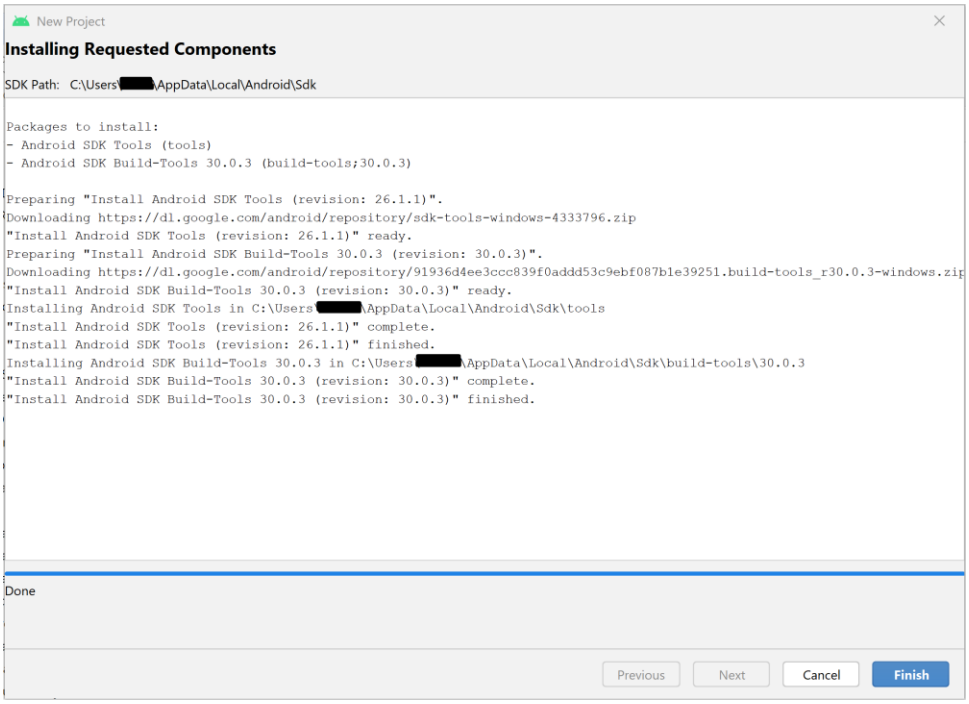
1. Upoznavanje sa *Android Studio* razvojnim okruženjem
    a. Preuzeti poslednju verziju Android Studio okruženje sa adrese (verzija u trenutku pisanja tutorijala Bumblebee 2021.1.1): https://developer.android.com/studio
    b. Nakon instalacije, pokrenuti *Android Studio* razvojno okruženje
    c. Na ekranu će biti prikazan starni ekran sa koga se mogu pokrenuti postojeći projekti ili kreirati novi.
2. Kreiranje *Hello World Android* projekta
    a. Kreirati novi projekat korišćenjem dugmeta *New project.*
    b. Izabrati opciju No Activity i nastaviti dalje.
    c. Upisati za ime aplikacije Hello World i ime paketa koji će biti korišćen koristiti a zatim izabrati programski jezik koji će biti korišćen za razvoj aplikacije. U ovom tutorijalu će biti korišćen Kotlin, kao moderna zamena za Java programski jezik koji je ranije dominantno korišćen za razvoj Android aplikacija. Potrebno je definisati i verziju Androida koja će biti korišćenja. Za ovu priliku je dovoljno zadržati podrazumevanu verziju koja pokriva 98% uređaja na tržištu a zatim kliknuti i kliknuti *Finish*.
    Application Name: *Hello World*
    Package name: *elfak.mosis.helloworld*



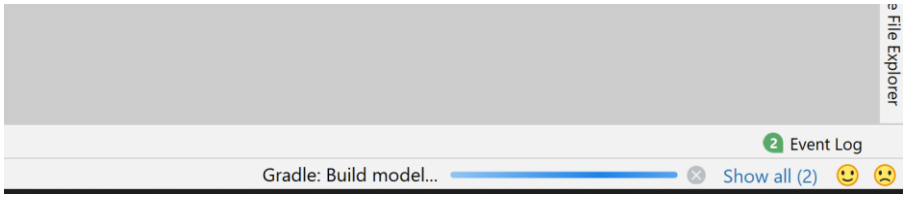


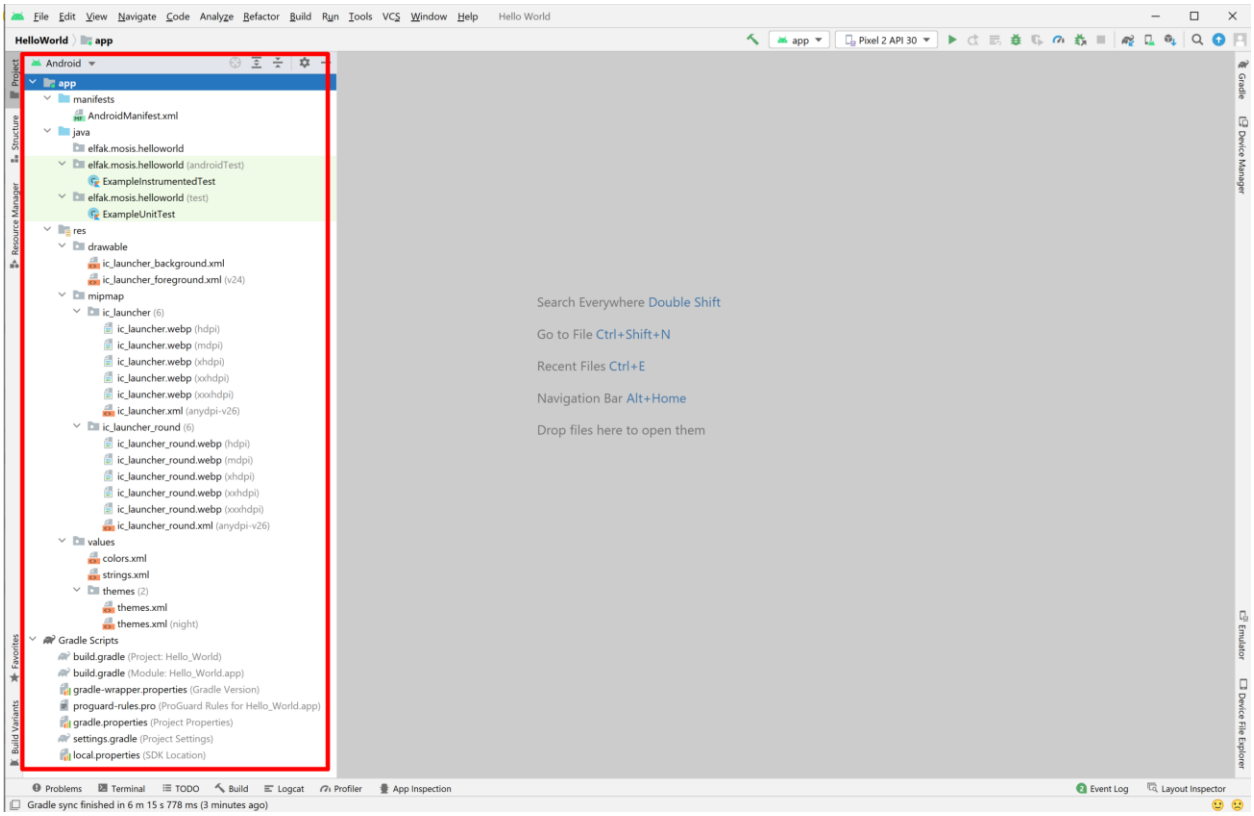


d. U skladu sa unetim izborima, Android Studio će probaviti odgovarajuće biblioteke i delove SDK za razvoj. Nakon što su svi potrebni moduli preuzeti i instalirani kliknuti na dugme *Finish*.



e. Nakon što je kreiranje projekta završeno, biće prikazano okruženje sa kreiranim projektom. Pre nego što okruženje bude spremno za prikaz hijerarhije fajlova kreiranog Android projekta, sačekati dok Gradle, podsistem za pribavljanje biblioteka ne instalira sve potrebne module za projekat. Progres se može pratiti u donjem desnom uglu razvojnog okruženja.





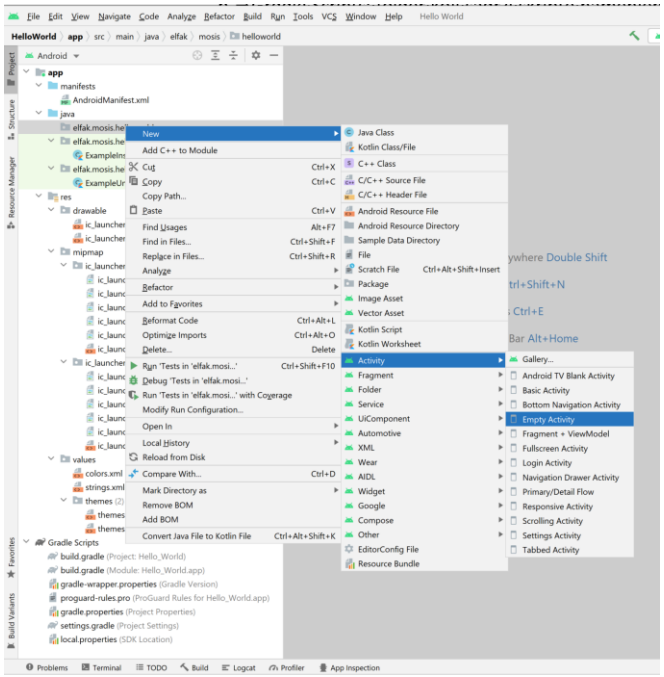
3. Detaljno proučiti kreiranu aplikaciju. Hijerarhija direktorijuma u *Android Studio*-u sadrži više foldera u kojima se nalaze delovi programskog koda, fajlovi koji opisuju korisnički interfejs u XML formatu kao i resurse koji će biti prikazivani u okviru aplikacije. Folderi od interesa su:
    a. *app*: folder koji sadrži fajlove namenjene izvornom kodu aplikacije i kao i resurse potrebne aplikaciji. Ovaj folder sadrži nekoliko potfoldera.
        i. *java*: sadrži hijerarhiju paketa i .java i/ili .kt klase koje se dodaju u projekat. Iako je izabran *Kotlin* kao programski jezik projekta, i dalje je zadrćan naziv foldera *java*. Ovo je i zbog toga što u Android projektu mogu egzistirati i jedan i drugi tip fajlova.
            1. *elfak.mosis.helloworld*: Sadrži sve klase projekta.
            2. *elfak.mosis.helloworld (androidTest)*: Sadrži klase u kojima se pišu testovi projekta. Neće biti od interesa u daljem toku laboratorijskih vežbi.
            3. *elfak.mosis.helloworld (Test)*: Takođe, projekat za pisanje *unit* testova.
        ii. *manifests*: folder u kome se nalazi AndroidManifest.xml fajl.

iii. *res*: ovaj folder će sadržati sve resurse koje će aplikacija koristiti. U startu sadrži tri osnovna foldera koje okruženje podrazumevano kreira. Svaki foldera može da ima više različitih verzija u zavisnosti od namene (multimedija, orijentacija ekrana, regionalna podešavanja jezika itd.)

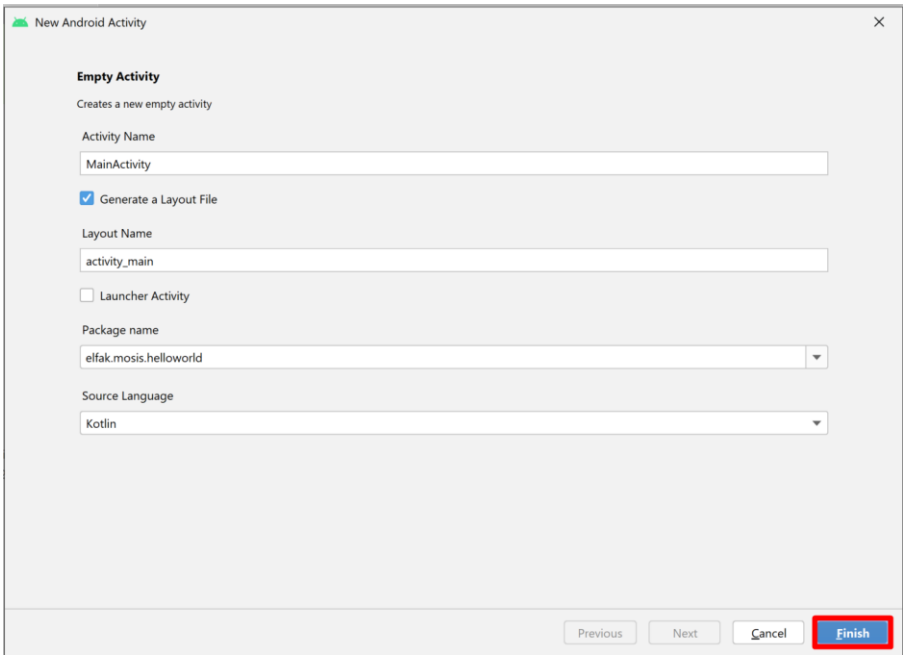
1. *drawable*: sadrži slike i animacije koje se koriste u aplikaciji (moguće je umesto jednog imati više ovakvih foldera u zavisnosti od rezolucija ekrana mobilnih telefona koje je potrebno podržati). Ovakvi folderi počinju rečju *drawable* i sadrže slike za ekrane različite gustine piksela) .
2. *mipmap*: sadrži folder u kome se nalaze ikone aplikacije namenjene različitim gustinama ekrana (*hdpi*, *mdpi*, *xhdpi* i *xxhdpi*).
3. values: sadrži value type resurse poput stringova, celih brojeva, nizova (sve konstante koje se koriste u kodu, preferira se njihovo navodjenje u resursima u odnosu na kod)
    a. Inicijalno sadrži colors.xml, strings.xml, folder sa themes.xml fajlovima u kome su definisani boje, stringovi i teme potrebne HelloWorld aplikaciji respektivno.

b. *Gradle Scripts*: folder koji sadrži skripte namenjene *Gradle build* sistemu. *Android Studio* sadrži *Gradle build* sistem u kome je moguće podešavati izvore koda (source control koji se koristi, biblioteke koje se preuzimaju sa servera itd.), izvore resursa, skinove aplikacije. Gradle skripte se programiraju korišćenjem *Groovy* skripti. U ovoj vežbi ćemo koristiti podrazumevana podešavanja za build aplikacije.

4. *Activity* je jedinstvena, fokusirana aktivnost koju korisnik može izvršavati. Koristi se za interakciju sa korisnikom. *Activity* služi za kreiranje prozora (podloge) na koju je moguće smestiti elemente korisničkog interfejsa sa kojima korisnik interreaguje. Dodati u aplikaciju *MainActivity.kt*.
    a. Desnim klikom kliknuti na package aplikacije i dodati *Empty Activity*.

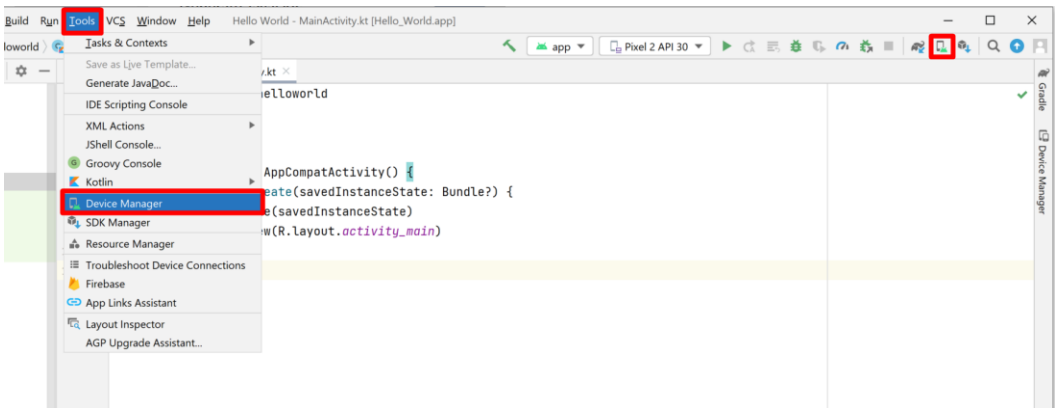


b. Prihvatiti podrazumevani unos i kliknuti dugme *Finish*.

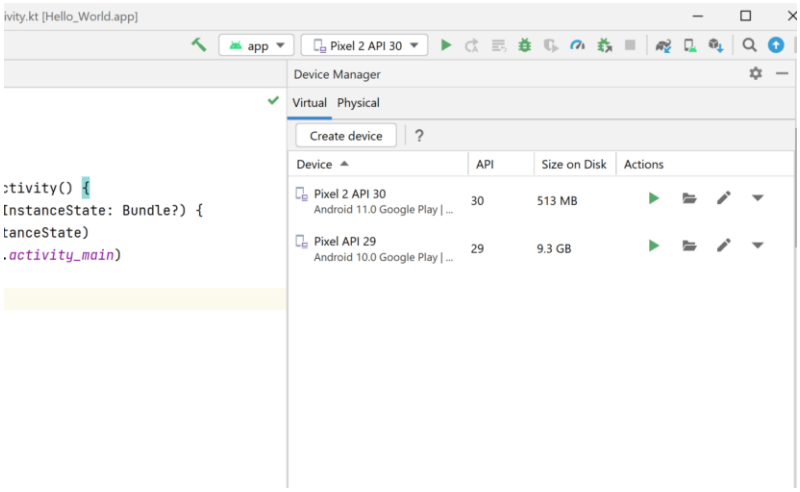


c. Primetiti da je dodata klasa *MainActivity.kt* u *package* aplikacije kao i novi potfolder *layout* sa *activity_main.xml* fajlom. Potfolder *layout* sadrži sve xml opise korisničkog interfejsa i može imati svoje pandane za različite veličine ekrana i/ili orijentacije (layout-large, layout-xlarge-land itd.). Korisnički interfejs *Activity*-a je moguće menjati direktno iz dizajnerskog prikaza ili menjanjem njegovog xml opisa u respektivnom xml fajlu.

5. Pre pokretanja aplikacije potrebno je napraviti emulator, ovo nije potrebno vršiti svaki put već samo onda kada je potrebno kreirati emulator sa odgovarajućim karakteristima projekta.
   a. Potrebno je startovati *Device Manager* (*Tools > Device Manager* ili izaberati ikonu toolbar-a). Alternativno, moguće je i startovati predefinisani uređaj klikom na dugme *play*.



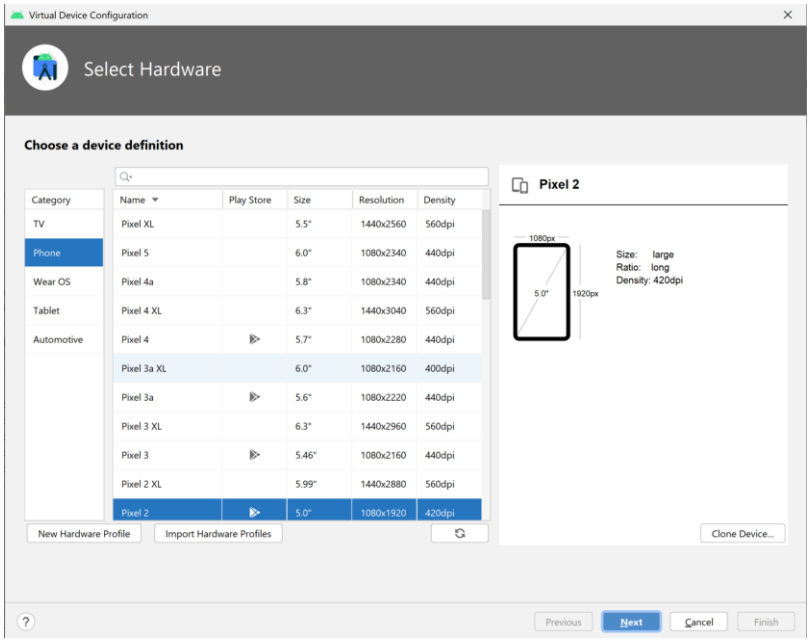
b. U desnom delu ekrana su redom su prikazani unapred kreirani emulatori.



c. Startovanje kreiranog emulatora se vrši klikom na *play* dugme.

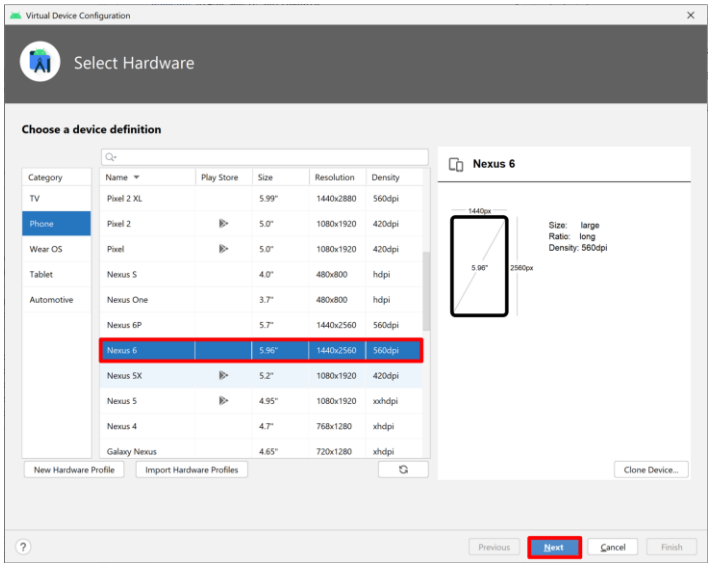
d. Kreiranje novog uređaja je moguće izborom dugmeta *Create Device* iz zaglavlja.

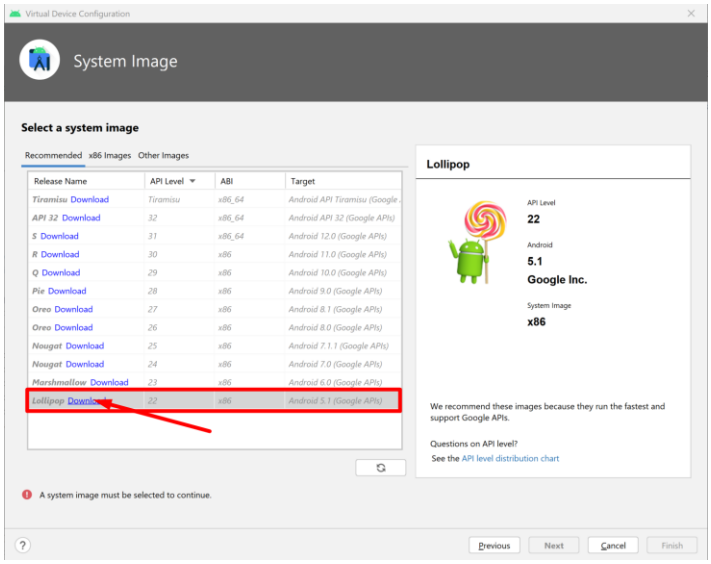
e. Izabrati jedan od unapred preipremljenih hardverskh profila ili kreirati uređaj po želji klikom na dugme *New Hardware Profile*.

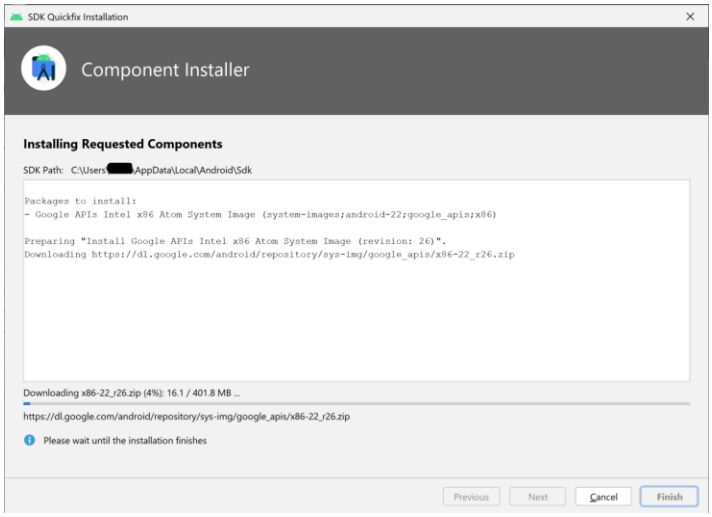


f. Kao primer, izaberati jedan profil i kliknuti na dugme *Next.*

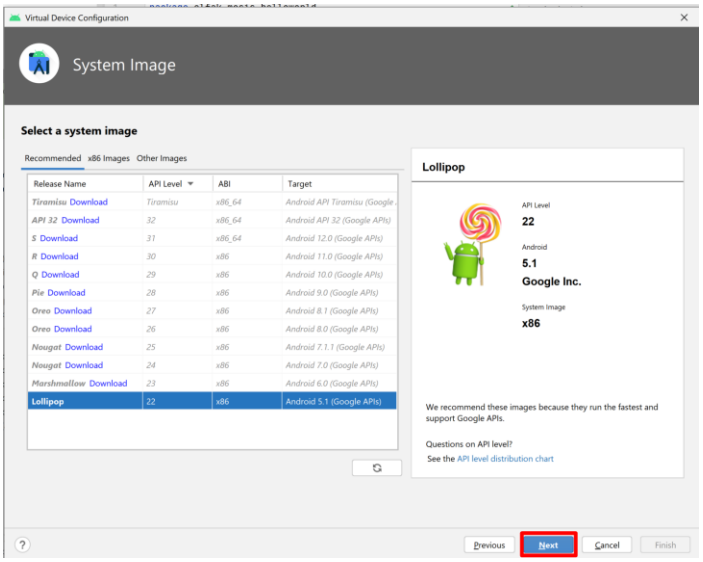
g. Izaberati verziju androida koju je potrebno koristiti za testiranje. Ukoliko ta verzija ne postoji na lokalnoj verziji *Android SDK*, potrebno ju je skinuti sa Internet-a klikom na dugme *Download.* Prilikom instalacije je potrebno saglasiti se sa odgovarajućim licencama za korišćenje.


Virtual Device Configuration
Select Hardware
Choose a device definition
Category
TV
Phone
Wear OS
Tablet
Automotive
Name
Play Store
Size
Resolution
Density
Pixel 2 XL 5.99" 1440x2880 560dpi
Pixel 2 5.0" 1080x1920 420dpi
Pixel 5.0" 1080x1920 420dpi
Nexus S 4.0" 480x800 hdpi
Nexus One 3.7" 480x800 hdpi
Nexus 6P 5.7" 1440x2560 560dpi
Nexus 6 5.96" 1440x2560 560dpi
Nexus 5X 5.2" 1080x1920 420dpi
Nexus 5 4.95" 1080x1920 xxhdpi
Nexus 4 4.7" 768x1280 xhdpi
Galaxy Nexus 4.65" 720x1280 xhdpi
New Hardware Profile
Import Hardware Profiles
Nexus 6
1440px
2560px
5.96"
Size: large
Ratio: long
Density: 560dpi
Clone Device...
Previous
Next
Cancel
Finish

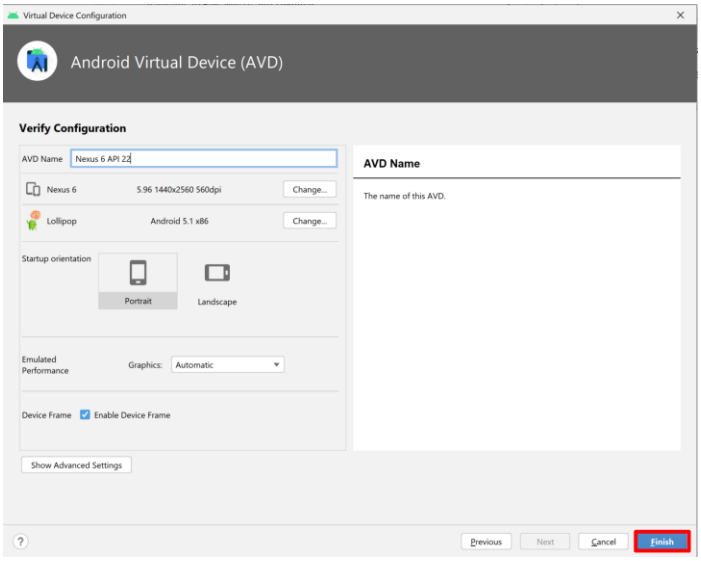

Virtual Device Configuration
System Image
Select a system image
Recommended
x86 Images
Other Images
Release Name
API Level
ABI
Target
Tiramisu Download
API 32 Download
S Download
R Download
Q Download
Pie Download
Oreo Download
Oreo Download
Nougat Download
Nougat Download
Marshmallow Download
Lollipop Download
Lollipop
API Level
22
Android
5.1
Google Inc.
System Image
x86
We recommend these images because they run the fastest and support Google APIs.
Questions on API level?
See the API level distribution chart
A system image must be selected to continue.
Previous
Next
Cancel
Finish


SDK Quickfix Installation
Component Installer
Installing Requested Components
SDK Path: C:\Users\[redacted]\AppData\Local\Android\Sdk
Packages to install:
- Google APIs Intel x86 Atom System Image (system-images;android-22;google_apis;x86)
Preparing "Install Google APIs Intel x86 Atom System Image (revision: 26)".
Downloading https://dl.google.com/android/repository/sys-img/google_apis/x86-22_r26.zip
Downloading x86-22_r26.zip (4%): 16.1 / 401.8 MB ...
https://dl.google.com/android/repository/sys-img/google_apis/x86-22_r26.zip
Please wait until the installation finishes
Previous
Next
Cancel
Finish

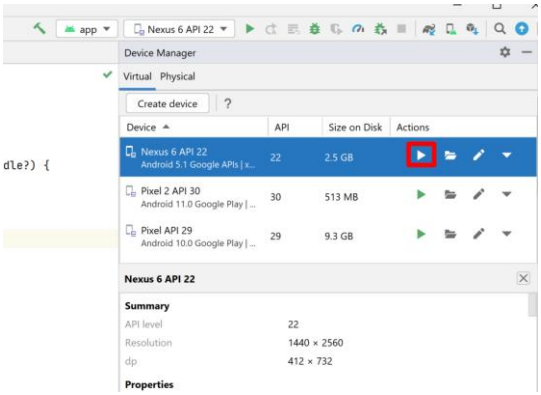
h. Nakon završetka preuzimanja potrebno je izabrati verziju koja je preuzeta a zatim kliknuti na dugme *Next*.



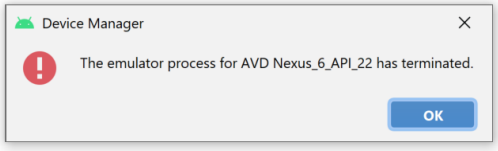
i. Potvrditi podešavanja koja su unapred izabrana i kliknuti na dugme *Finish*.



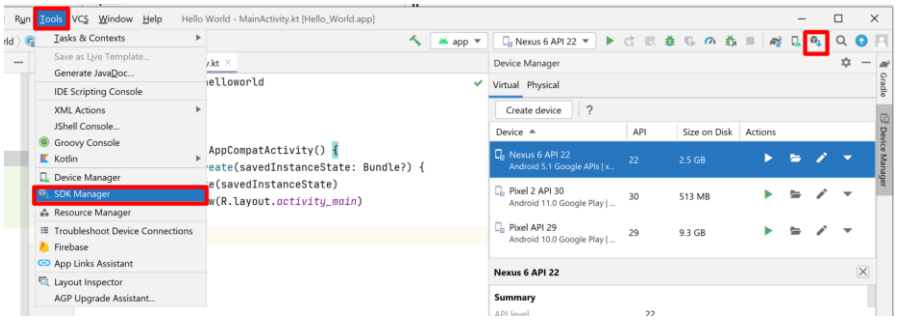
j. Rezultat kreiranja emulatora će biti prikazan u listi. Pokrenuti emulator klikom na dugme *play*.



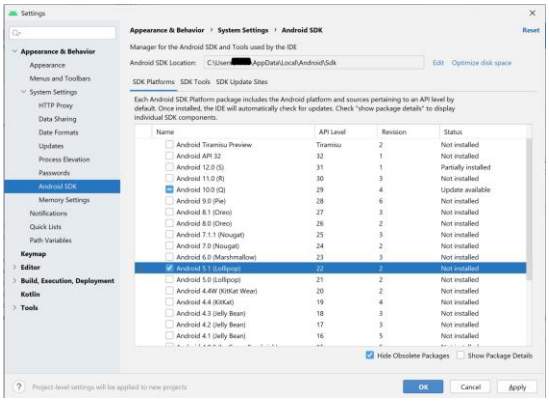
k. Ukoliko emulator sa datim podešavanjima nije pokrenut, kao što je prikazano na slici, potrebno je instalirati ciljanu Android verziju korišćenjem *SDK Manager* aplikacije.



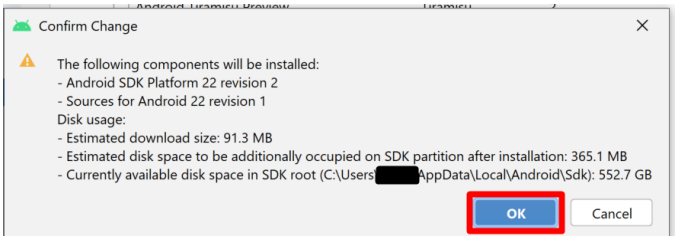
l. Startovati *SDK Manager* (*Tools > SDK Manager* ili izaberite ikonu toolbar-a).



m. Izbrati instalaciju Android verzije 22 čekiranjem odgovarajuće stavke a zatim kliknuti na dugme. *Install*.



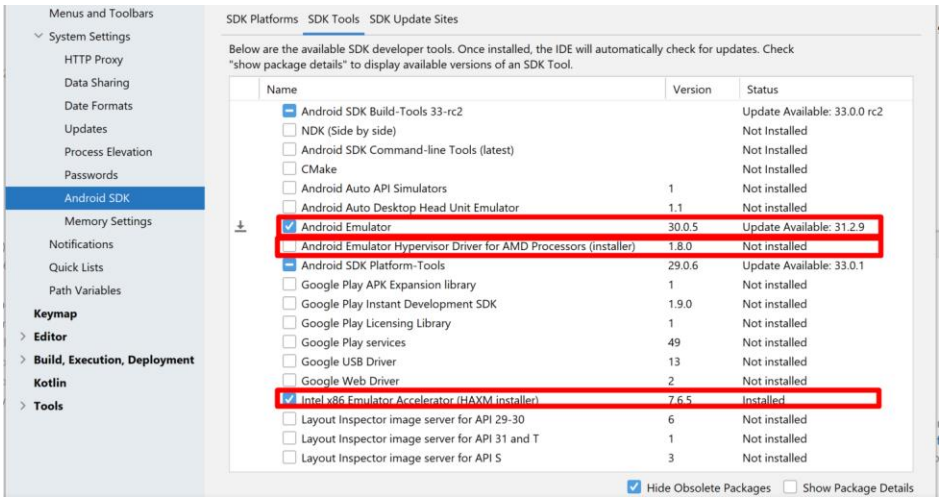
n. Potvrditi instalaciju



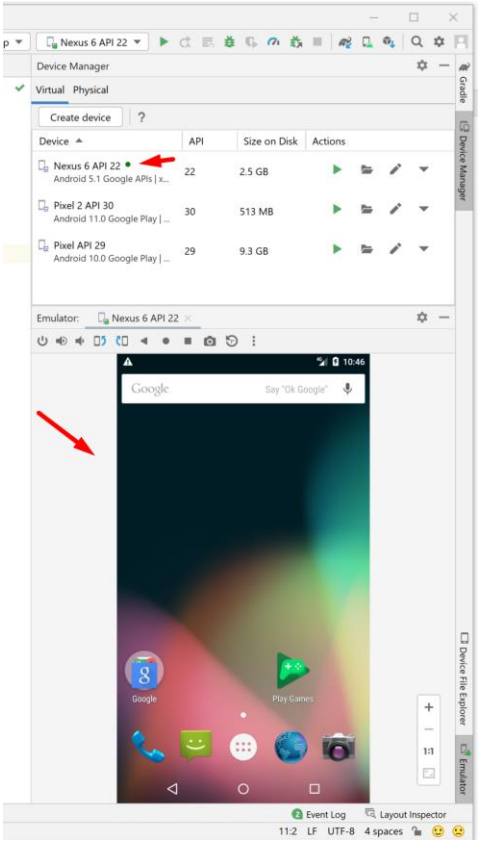
o. Ukoliko i nakon instaliranja odgovarajuće verzije korišćenjem *SDK Manager*-a emulator ne može da se pokrene, potrebno je restartovati *Android Studio.*

p. Pokrenuti kreirani emulator klikom na *play* dugme.

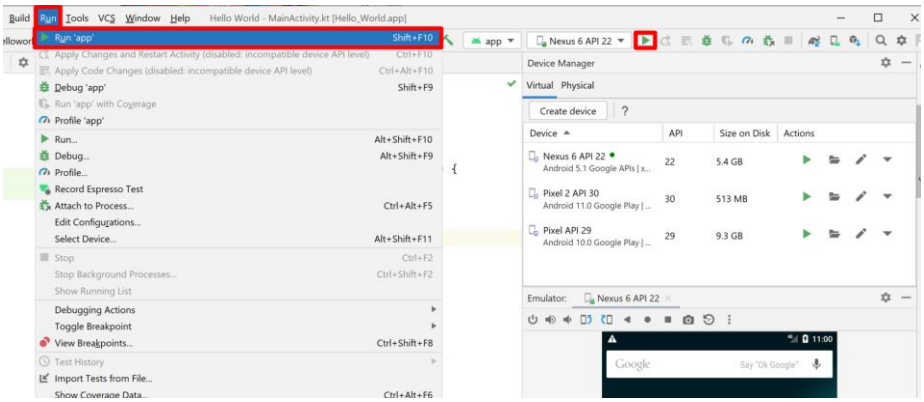
q. Ukoliko i dalje postoje problemi, potrebno je ažurirati odgovarajuće drajvere za virtuelizaciju na odgovarajućem procesoru kao i sam emulator iz *SDK Manager*-a.



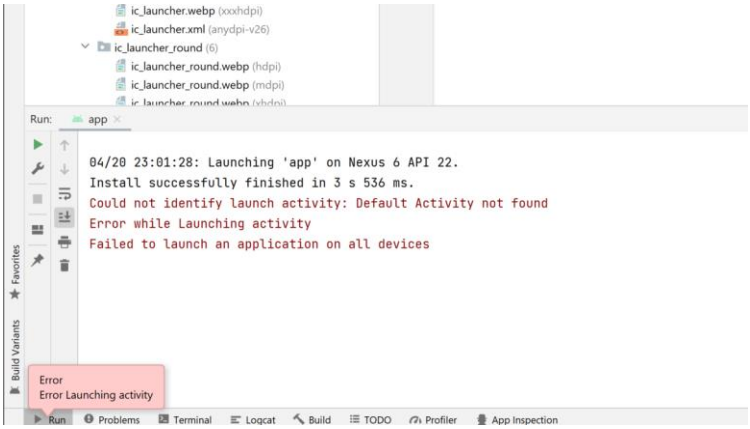
r. Emulatoru je potrebno vreme da se pokrene. Nemojte ga gasiti tokom razvoja. Svaki put kada budete pokrenuli aplikaciju iz okruženja, ona će biti ponovo učitana. Nakon pokretanja emulator je vidljiv u okruženju.



6. Pokrenuti aplikaciju izborom *Run -> Run* ili izaberite ikonicu iz *toolbar*-a.



a. Pošto je *MainActivity.kt* klasa dodata naknadno, okruženje je ne prepoznaje kao podrazumevani *Activity* aplikacije.



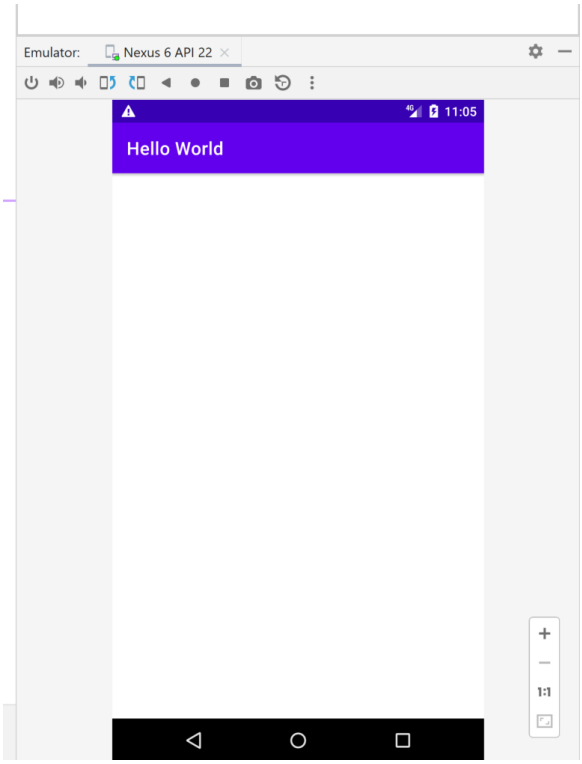
b. Potrebno je editovati fajl *AndroidManifest.xml*. Otvoriti *AndroidManifest.xml* i u delu gde je naveden *MainActivity* dodati kod kao na slici.

```
ty_main.xml   MainActivity.kt   AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="elfak.mosis.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Hello World"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.HelloWorld">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

c. Pokrenuti aplikaciju ponovo.

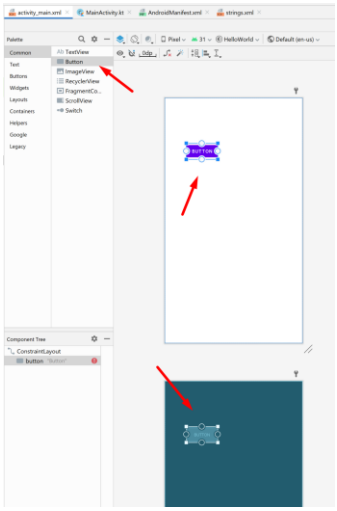
d. Aplikacija nakon pokretanja izgleda kao na slici:



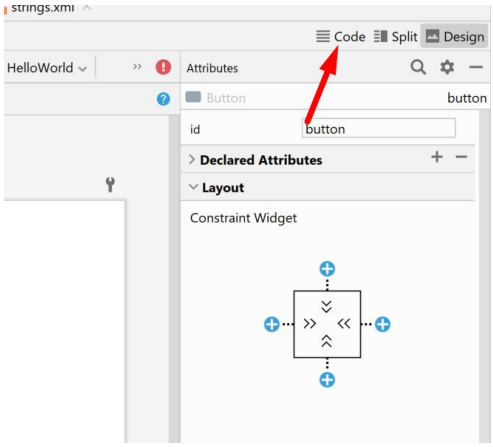
e. Proverite funkcionalnosti emulatora i aplikacije.

7. Upoznati se sa životnim ciklusom Android aplikacije:
   a. Otvoriti strings.xml i u njemu promeniti *app_name* resurs kao na slici. Dodati slog sa tekstom koji će biti iskorišćen na dugmetu koje će biti dodato.

   b. Otvoriti *activity_main.xml*. Uočiti da je inicijalno otvoren dizajner korisničkog interfejsa. Iz dizajnera dodajti *Button* element.



   c. Otvorite *activity_main.xml* fajl izborom *Code tab*-a u gornjem desnom zaglavlju.



d. Proučiti *activity_main.xml.* Izmeniti ime string resursa dugmeta.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.c
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/finish_button"
        tools:layout_editor_absoluteX="64dp"
        tools:layout_editor_absoluteY="135dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

e. Izmeniti *id* dodatog dugmeta u *finish_button*.

```
<Button
    android:id="@+id/finish_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/finish_button"
    tools:layout_editor_absoluteX="64dp"
    tools:layout_editor_absoluteY="135dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

f. U MainActivity.kt klasi dodati odgovarajući kod za *onClick* metod kreiranom dugmetu.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val finishButton: Button = findViewById(R.id.finish_button)
    }
}
```

g. Da bi kod bio funkcionalan, u klasu je potrebno uključiti odgovarajući paket. Najlakše je to postići klikom na tekst Button i kombinacijom tastera sa tastature Alt+Enter nakon čega treba kliknuti na *import* stavku.

h. Dodati *onClickListener* dugmetu.

i. Za svaku ključnu metodu vezanu za životni ciklus android aplikacije napraviti *Override* funkcije
(*onRestart(), onStart(), onResume(), onPause(), onStop(), onDestroy()*).

y_main.xml | MainActivity.kt | AndroidManifest.xml | strings.xml

```
package elfak.mosis.helloworld

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val finishButton: Button = findViewById(R.id.finish_button)
    }

    override fun onStart() {
        super.onStart()
    }

    override fun onRestart() {
        super.onRestart()
    }

    override fun onResume() {
        super.onResume()
    }

    override fun onPause() {
        super.onPause()
    }

    override fun onStop() {
        super.onStop()
    }

    override fun onDestroy() {
        super.onDestroy()
    }
}
```

j. Proveriti kako se aplikacija ponaša.

k. Uskladiti kod kao što je prikazano na slici. Svaki metod treba da ima poziv Toast.makeText(this, "*ime_funkcije_u_kojoj_se_nalazi*", Toast.LENGTH_SHORT).show(); i odgovarajući poziv metodi nadklase (Npr. super.onStart()).

```
package elfak.mosis.helloworld

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val finishButton: Button = findViewById(R.id.finish_button)
        finishButton.setOnClickListener({view -> finish()})
    }

    override fun onStart() {
        super.onStart()
        Toast.makeText( context: this,  text: "onStart", Toast.LENGTH_SHORT).show()
    }

    override fun onRestart() {
        super.onRestart()
        Toast.makeText( context: this,  text: "onRetart", Toast.LENGTH_SHORT).show()
    }

    override fun onResume() {
        super.onResume()
        Toast.makeText( context: this,  text: "onResume", Toast.LENGTH_SHORT).show()
    }

    override fun onPause() {
        Toast.makeText( context: this,  text: "onPause", Toast.LENGTH_SHORT).show()
        super.onPause()
    }

    override fun onStop() {
        Toast.makeText( context: this,  text: "onStop", Toast.LENGTH_SHORT).show()
        super.onStop()
    }

    override fun onDestroy() {
        Toast.makeText( context: this,  text: "onDestroy", Toast.LENGTH_SHORT).show()
        super.onDestroy()
    }
}
```

l. Pratite dešavanja u aplikaciji u emulatoru nakon pozivanja telefonskog broja, pritiska na Back dugme itd.