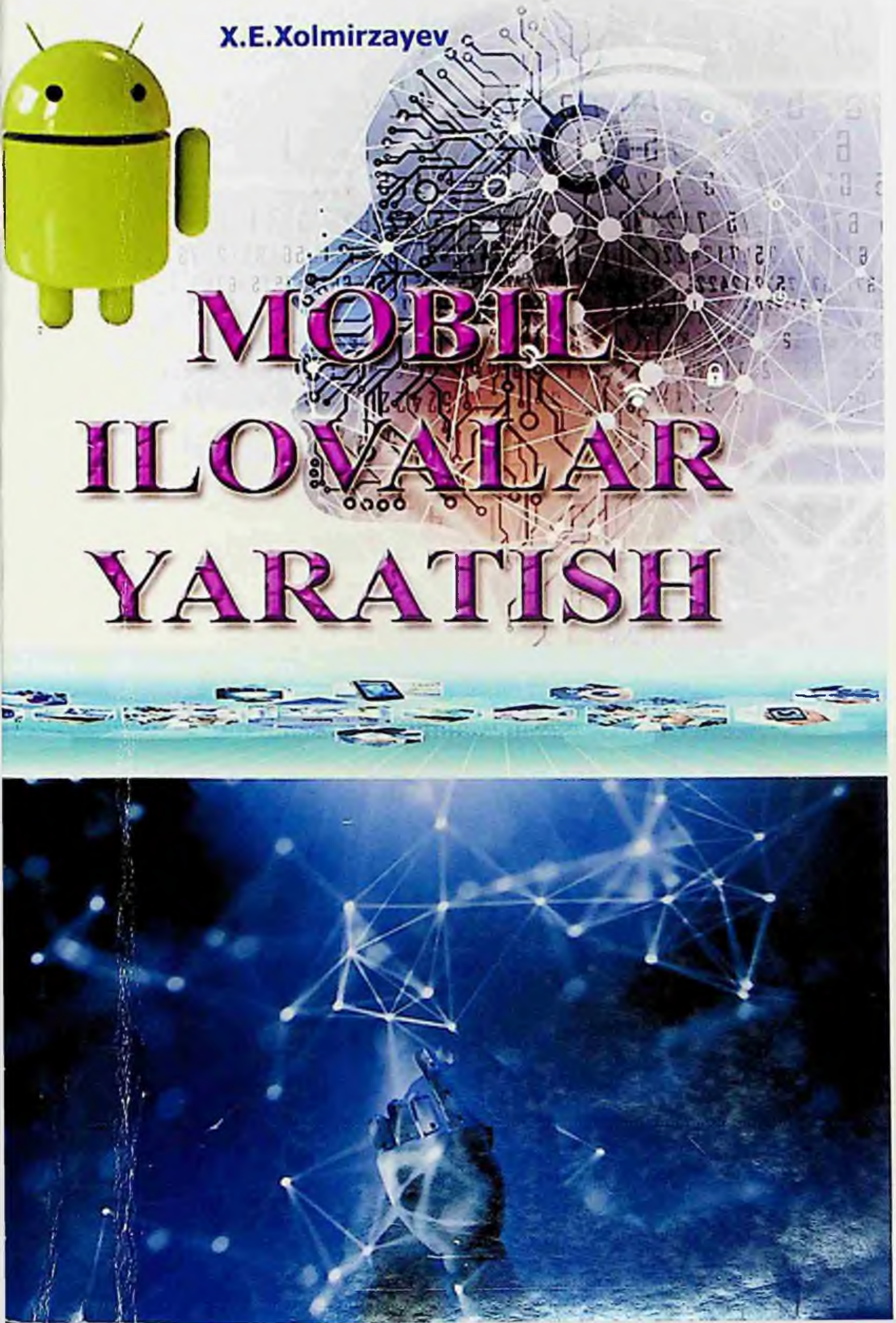
X.E.Xolmirzayev

# MOBIL ILOVALAR YARATISH



O'ZBEKISTON RESPUBLIKASI

OLIY TA`LIM, FAN VA INNOVATSIYALAR VAZIRLIGI

# "MOBIL ILOVALAR YARATISH"

O`QUV QO`LLANMA

60540200 – Amaliy matematika,

60610200 – Axborot tizimlari va texnologiyalari,

Namangan 2023

"Mobil ilovalar yaratish" nomli o`quv qo`llanma 60540200 – Amaliy matematika va 60610200 – Axborot tizimlari va texnologiyalari yo`nalishlari talabalariga "Mobil ilovalar yaratish" fanidan amaliy va ma'ruza mashg'ulotlar uchun mo`ljallangan bo`lib, fan va ishchi o'quv dasturi asosida yaratilgan. O`quv qo`llanmada berilgan har bir bob va mavzular asosida tuzuvchi tomonidan Android Studio dasturida yangi ilovalar yaratish va qayta ishlash haqida tushunchalar, shuningdek Java va XML tilida dasturlari keltirilgan.

**Tuzuvchi:**

H.E.Holmirzayev,

**Taqrizchilar:**

1. NamDU, Informatika kafedrasi dotsenti PhD. Sh.K.Boltibayev
2. TATU Urganch filiali, Raqamli ta'lim texnologiyasi kafedrasi
   dotsenti PhD. B.Yu.Palvanov

Mazkur o`quv qo`llanma O`zbekiston Respublikasi Oliy ta'lim, fan va innovatsiyalar vazirligining 2023 yil 27 martdagi 68-sonli buyrug`iga asosan O`zbekiston Respublikasi Vazirlar Mahkamasi tomonidan litsenziya berilgan nashriyotlarda nashr etishga ruhsat berrildi. Ro`yhatga olish raqami: № 68-358.

# MUNDARIJA

# I-BOB. ANDROID BILAN ISHLASHNI BOSHLASH

## 1.1. Kirish. Android Studio va Android SDK-ni o'rnatish.

So'nggi yillarda axborot texnologiyalarining jadal rivojlanishi planshetlar, smartfonlar, netbuklar va boshqa qurilmalar kabi ko'plab yangi qurilmalar va texnologiyalar paydo bo'lishiga olib keldi. Ular hayotimizga tobora mustahkamroq qo'shilib, odatiy holga aylanib bormoqda. Bugungi kunda bunday gadjetlar orasida etakchi platforma Android OS hisoblanadi.

Android turli xil qurilmalarda qo'llaniladi. Bular smartfonlar, planshetlar, televizorlar, aqlli soatlar va boshqa bir qator qurilmalar. Turli xil hisob-kitoblarga ko'ra, 2020 yilda ushbu operatsion tizimdan smartfon egalarining qariyb 85 foizi foydalanadi va Android smartfonlaridan foydalanuvchilarning umumiy soni dunyo bo'ylab 2,5 milliarddan ortiq odamni tashkil qiladi.

Android OS dasturchisi Andy Rubin tomonidan mobil telefonlar uchun operatsion tizim sifatida yaratilgan va dastlab Android Inc doirasida ishlab chiqilgan. Ammo 2005 yilda Google Android Inc.ni sotib oldi. Operatsion tizimni yangi kuch bilan rivojlantira boshlaydi. Android doimiy ravishda rivojlanib boradi va ishlab chiqarish vositalari va vositalari operatsion tizim bilan birga rivojlanib boradi. Hozirda (2020 yil noyabr) so'nggi versiyasi - Android 11.0, 2020 yil sentyabrda chiqdi:

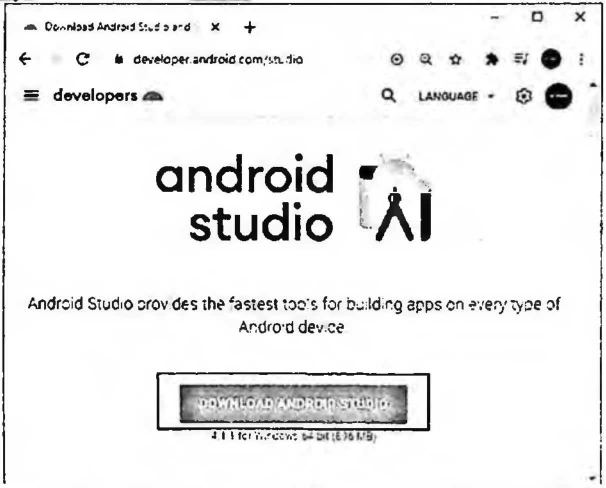
1-jadval. Android

| Versiyasi | Kod nomi | Ishlab chiqarilish sanasi | API darajasi |
|---|---|---|---|
| 12.0 | 12 | 4 oktyabr 2021 yil | 31 |
| 11.0 | 11 | 8 sentyabr 2020 yil | 30 |
| 10.0 | 10 | 3 sentyabr 2019 yil | 29 |
| 9.0 | Pie | 6 avgust 2018 yil | 28 |
| 8.1 | Oreo | 5 dekabr 2017 yil | 27 |
| 8.0 | Oreo | 21 avgust 2017 yil | 26 |
| 7.1 | Nougat | 4 oktyabr 2016 yil | 25 |
| 7.0 | Nougat | 2016 yil 22-avgust | 24 |
| 6.0 | Marshmallow | 2015 yil 5-oktabr | 23 |
| 5.1 | Lollipop | 9 mart 2015 yil | 22 |
| 5.0 | Lollipop | 2014 yil 3-noyabr | 21 |
| 4.4 | KitKat | 2013 yil 31 oktyabr | 19 |
| 4.3 | Jelly Bean | 2013 yil 24-iyul | 18 |
| 4.2 | Jelly Bean | 2012 yil 13-noyabr | 17 |
| 4.1 | Jelly Bean | 2012 yil 9-iyul | 16 |
| 4.0 | Ice Cream Sandwich | 2011 yil 16-dekabr | 15 |
| 2.3 | Gingerbread | 2010 yil 6-dekabr | 10 |

Rivojlanish uchun sizga nima kerak?

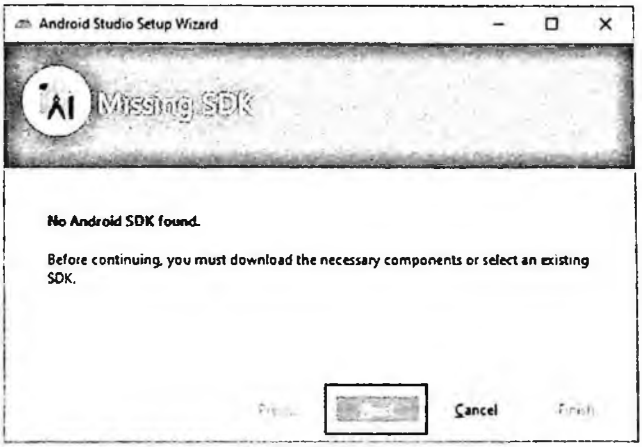
Shuni ta'kidlash kerakki, siz Android uchun turli xil ramkalar va dasturlash tillaridan foydalangan holda dasturlarni ishlab chiqishingiz mumkin. Shunday qilib, dasturlash tillari sifatida Java, Kotlin, Dart (Flutter framework), C ++, Python, C # (Xamarin platformasi) va boshqalar ishlatilishi mumkin. Ushbu qo'llanmada biz eng keng tarqalgan va ishlatiladigan Java tilidan foydalanamiz. Shuning uchun, ushbu qo'llanma yordamida Android uchun dasturlashni o'rganishdan oldin, hech bo'lmaganda Java tilining asoslarini o'zlashtirishingiz kerak.

**Rivojlanish vositalarini o'rnatish**. Android uchun turli xil rivojlanish muhiti mavjud. Tavsiya etilgan ishlab chiqish muhiti - bu Android ishlab chiqish uchun maxsus ishlab chiqilgan Android Studio. Shuning uchun biz undan foydalanamiz. O'rnatish faylini rasmiy veb-saytidan yuklab olishingiz mumkin: https://developer.android.com/studio:


Download Android Studio and
developer.android.com/studio
developers
LANGUAGE
android studio
Android Studio provides the fastest tools for building apps on every type of Android device
DOWNLOAD ANDROID STUDIO

1.1-rasm. Android Studio muhitini o`rnatuvchi faylini ko`chirib olish.

Android Studio muhitidan tashqari, rivojlanish uchun Android SDK deb nomlangan vositalar to'plami ham kerak bo'ladi. Masalan, oldin Android SDK o'rnatilmagan bo'lsa, u holda Android Studio bilan birinchi marta bog'langaningizda, Android SDK yo'qligi haqida xabar beradi.

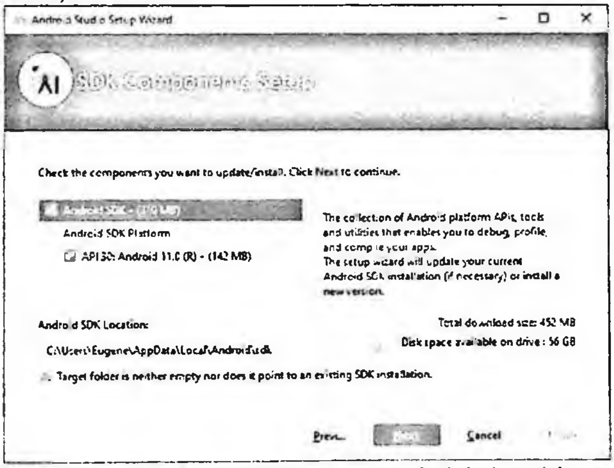


No Android SDK found.

Before continuing, you must download the necessary components or select an existing SDK.

1.2-rasm. Android SDK ni o`rnatish.

Android SDK-ni rasmiy saytdan alohida-alohida qo'lda yuklab olishimiz va o'rnatishimiz mumkin. Yoki biz buni to'g'ridan-to'g'ri Android Studio-dan amalga oshirishimiz mumkin. Shunday qilib, Next tugmachasini bosamiz. Keyingi ekranda bizdan Android SDK-ni so'nggi API versiyasi uchun yuklab olish talab qilinadi (bu holda Android 11):



Check the components you want to update/install. Click Next to continue.

Android SDK Platform

API 30: Android 11.0 (R) - (142 MB)

The collection of Android platform APIs, tools and utilities that enables you to debug, profile, and compile your apps.
The setup wizard will update your current Android SDK installation (if necessary) or install a new version.

Android SDK Location:

C:\Users\Eugene\AppData\Local\Android\sdk

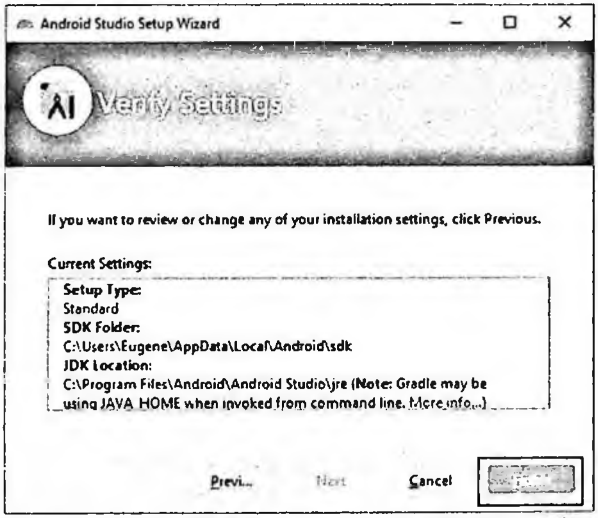
Total download size: 452 MB

Disk space available on drive : 56 GB

Target folder is neither empty nor does it point to an existing SDK installation.

1.3-rasm. Android SDK komponentalarini o`rnatish.

Bu yerda agar standart yo'l bizga mos kelmasa, biz Android SDK-ni o'rnatish uchun joyni belgilashimiz mumkin. Next tugmachasini bosing, so'ngra aniq nima o'rnatilishi haqida qisqacha ma'lumot berilgan oyna paydo bo'ladi(1.4-rasm):



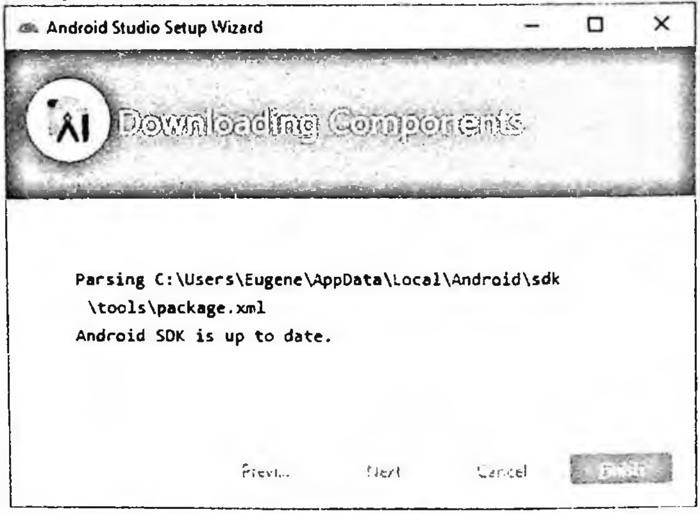
If you want to review or change any of your installation settings, click Previous.

Current Settings:

```
Setup Type:
Standard
SDK Folder:
C:\Users\Eugene\AppData\Local\Android\sdk
JDK Location:
C:\Program Files\Android\Android Studio\jre (Note: Gradle may be
using JAVA_HOME when invoked from command line. More info...)
```

1.4-rasm. Android SDK komponentalarini o`rnatish yakuni

Buning hammasini o'rnatish uchun Finish tugmachasini bosing.



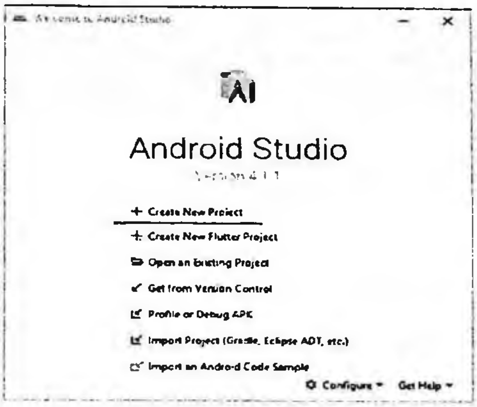
```
Parsing C:\Users\Eugene\AppData\Local\Android\sdk
 \tools\package.xml
Android SDK is up to date.
```

1.5-rasm. Android SDK komponentalarini ko`chirib olish va o`rnatish yakuni

O'rnatish tugagandan so'ng, **Finish** tugmasini bosing. Va biz dasturlarni yaratishni boshlashimiz mumkin.

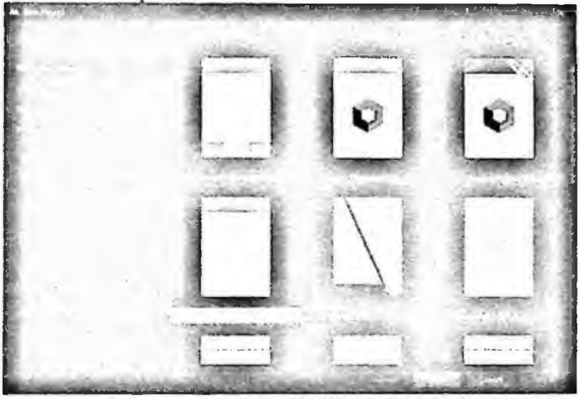
## 1.2. Android Studio-dagi birinchi loyiha.

Endi Android operatsion tizimi uchun birinchi Android Studio dasturini yarataylik. Keling, Android Studio-ni ochamiz va boshlang'ich ekranda Create New Project elementini tanlang:


Welcome to Android Studio
Android Studio
+ Create New Project
+ Create New Flutter Project
Open an Existing Project
Get from Version Control
Profile or Debug APK
Import Project (Gradle, Eclipse ADT, etc.)
Import an Android Code Sample
Configure
Get Help

1.6-rasm. Android Studio dastur umumiy ko`rinishi

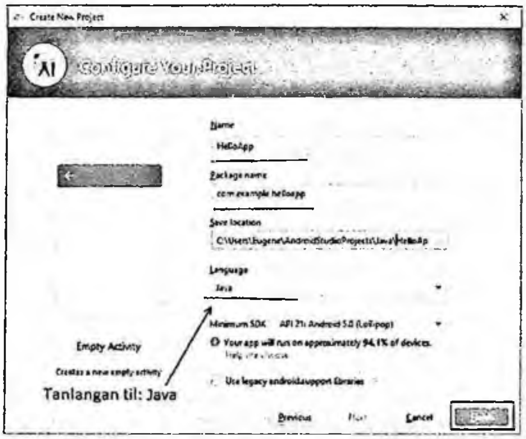
Loyihani yaratishda Android Studio birinchi navbatda bizdan loyiha shablonini tanlashni talab qiladi:



1.7-rasm. Loyiha shablonini tanlash

Android Studio turli xil vaziyatlar uchun bir qator shablonlarni taqdim etadi, ammo eng keng tarqalgani Basic Activity va Empty Activity. Bu ko'plab dasturlarni yaratish uchun eng qulay start-up shablonlari. Endi boshlang`ich holat bo'yicha Empty Activity shabloni tanlanadi (agar tanlanmagan bo'lsa, uni tanlang) va Next tugmasini bosing.

Shundan so'ng, yangi loyiha sozlamalari oynasi ko'rsatiladi:


Create New Project

Configure Your Project

Name
HelloApp

Package name
com.example.helloapp

Save location
C:\Users\Eugene\AndroidStudioProjects\Java\HelloAp

Language
Java

Minimum SDK API 21: Android 5.0 (Lollipop)

Your app will run on approximately 94.1% of devices.

Use legacy android.support libraries

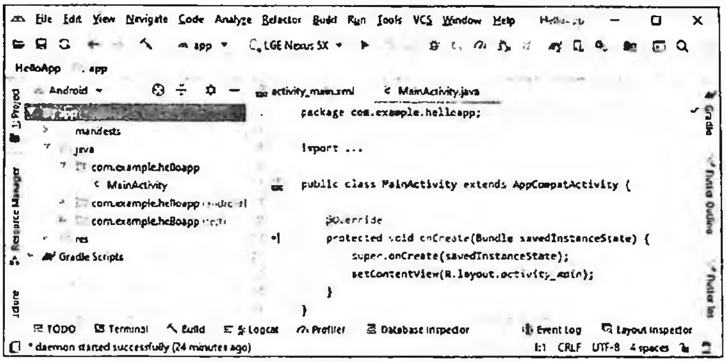
1.8-rasm. Loyiha shablonida tilni tanlash

Yangi loyihani yaratish oynasida biz uning dastlabki sozlamalarini o'rnatamiz:

- Ilova nomi **Name** maydoniga kiritiladi. **HelloApp** ismini nom sifatida ko'rsating.
- **Package Name** maydoni asosiy dastur sinfini joylashtiradigan paket nomini belgilaydi. Bunday holda, test loyihalari uchun bu qiymat juda muhim emas, shuning uchun **com.example.helloapp**-ni o'rnating.
- **Save Location** maydoni qattiq diskdagi loyiha fayllarining joylashishini belgilaydi. Boshlang`ich holat bo`yicha qoldirishimiz mumkin.
- **Language** maydonida biz **Java** dasturlash tili sifatida ko'rsatamiz (ehtiyot bo'ling, chunki boshlang`ich holat bo'yicha ushbu maydonda Kotlin dasturlash tili mavjud)
- **Minimum SDK** maydoni eng kichik qo'llab-quvvatlanadigan SDK versiyasini belgilaydi. Standart - **API 21: Android 5.0 (Lollipop)** ni tark etamiz, demak bizning dasturimiz Android 5.0 dan boshlab ishga tushirilishi mumkin, ya'ni 94% qurilmalar. Eski qurilmalar ishlay olmaydi.

Shuni esda tutish kerakki, SDK versiyasi qanchalik yuqori bo'lsa, qo'llab-quvvatlanadigan qurilmalar diapazoni shunchalik kichik bo'ladi.

Keyin **Finish** tugmachasini bosing va **Android Studio** yangi loyihani yaratadi:

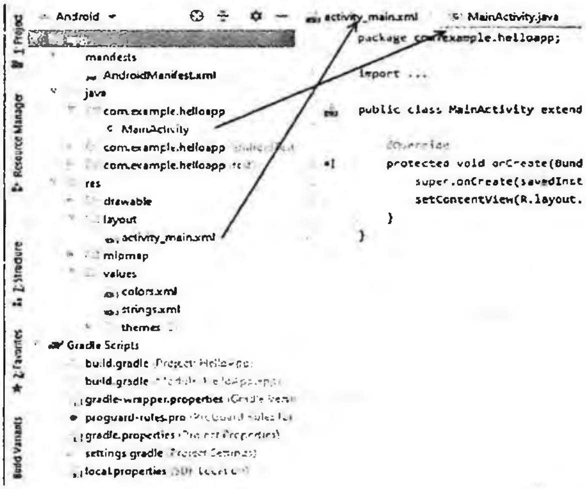


```
package com.example.helloapp;

import ...

public class MainActivity extends AppCompatActivity {

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

```
* daemon started successfully (24 minutes ago)                1:1  CRLF  UTF-8  4 spaces
```

1.9-rasm. Loyiha oynasi tuzilishi

Birinchidan, loyihaning boshlang`ich holat bo'yicha tuzilishini tezda ko'rib chiqamiz.



```
package com.example.helloapp;

import ...

public class MainActivity extend

    @Override
    protected void onCreate(Bund
        super.onCreate(savedInst
        setContentView(R.layout.
    }
}
```

1.10-rasm. Loyiha oynasi asosiy tashkil etuvchi papkasi.

Android loyihasi turli xil modullardan iborat bo'lishi mumkin. Odatiy bo'lib, biz loyihani yaratishda bitta modul yaratiladi - ilova. Modul uchta pastki papkaga ega:

- manifestlar: dastur konfiguratsiyasini tavsiflovchi va berilgan dastur tarkibiy qismlarining har birini belgilaydigan AndroidManifest.xml manifest faylini saqlaydi.
- java: java kod fayllarini alohida paketlarga tuzilgan holda saqlaydi. Shunday qilib, com.android.helloapp papkasida (uning nomi loyihani yaratish bosqichida ko'rsatilgan) sukut bo'yicha MainActivity.java fayli mavjud bo'lib, Java tilidagi kod mavjud bo'lib, u dastur boshlanganda sukut bo'yicha ishga tushirilgan MainActivity sinfini anglatadi.

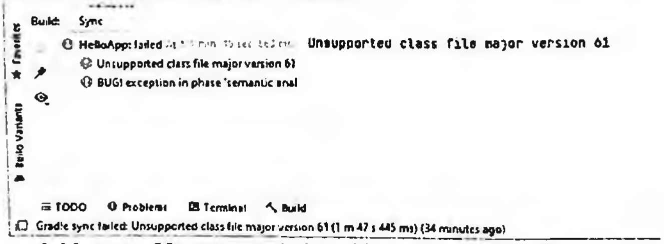
- res: dasturda ishlatiladigan resurslarni o'z ichiga oladi. Barcha manbalar pastki papkalarga bo'linadi.
    - drawable papkasi dasturda ishlatiladigan rasmlarni saqlash uchun;
    - layout papkasi grafik interfeysni belgilaydigan fayllarni saqlash uchun mo'ljallangan. Odatiy bo'lib, MainActivity sinfining interfeysini xml shaklida belgilaydigan activity_main.xml deb nomlangan fayl mavjud.
    - mipmap papkalarida turli xil ekran o'lchamlari bilan dastur piktogrammalarini yaratish uchun ishlatiladigan rasm fayllari mavjud.
    - values papkasida turli xil xml fayllar saqlanadi, ular tarkibida resurslar to'plami mavjud - ilovada ishlatiladigan har xil ma'lumotlar. Odatiy bo'lib, bu erda ikkita fayl va bitta papka mavjud:
        - colors.xml fayli dasturda ishlatiladigan ranglarning tavsifini saqlaydi
        - strings.xml fayli dasturda ishlatiladigan satr manbalarini o'z ichiga oladi
        - themes papkasida ikkita dastur mavzusi saqlanadi - yorug'lik (kunduzi) va qorong'i (tun) uchun

Alohida element, Gradle Scripts, dasturni yaratish uchun ishlatiladigan bir qator skriptlarni o'z ichiga oladi.

Ushbu butun tuzilishda Android Studio-da ochilgan va dastur mantig'ini o'z ichiga olgan va aslida dasturni bajarilishini boshlaydigan MainActivity.java faylini ajratib ko'rsatish kerak. Bundan tashqari, grafik interfeysni belgilaydigan activity_main.xml faylini tanlang - asosan dasturni yuklab olgandan keyin foydalanuvchi o'z smartfonida nimani ko'radi.

**Yuzaga kelishi mumkin bo`lgan muammolar.**

Ilova yaratish uchun Java dan foydalaniladi. Ilovani yaratish uchun esa Gradle infratuzilmadan foydalaniladi. Biroq, hozirda foydalanilayotgan Gradle versiyasi tanlangan standart JDK versiyasiga mos kelmasligi mumkin. Va bu holda Android Studio xatolarni ko'rsatishi mumkin.

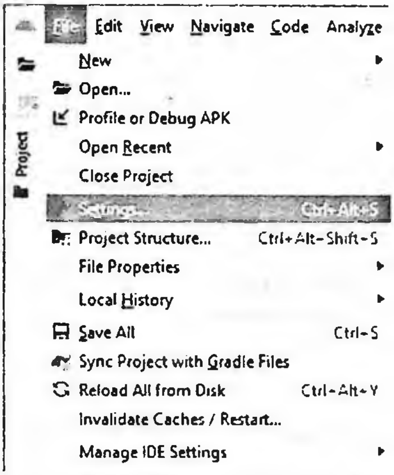


```
HelloApp: failed
  Unsupported class file major version 61
  BUG! exception in phase 'semantic anal
Unsupported class file major version 61
Gradle sync failed: Unsupported class file major version 61 (1 m 47 s 445 ms) (34 minutes ago)
```

1.11-rasm. Unsupported class file major version 61 xatosi

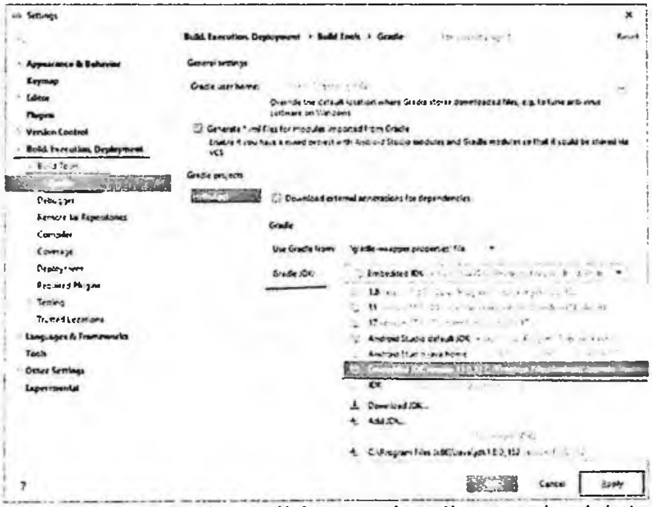
Bu xato JDK-ning 17-versiyasi Gradle ning joriy versiyasiga mos kelmasligini bildiradi. Quyi versiyadan foydalanishingiz kerak.

Ushbu muammoni hal qilish uchun File -> Settings dastur menyusiga o'tamiz (MacOS-da bu Android Studio -> Preferences)



1.12-rasm. Android Studio IDE Fayl menyusi

Keyin ochilgan sozlash oynasida **Build, Execution, Deployment -> Build Tools -> Gradle** menyu bandiga o'ting va keyin **Gradle JDK** maydonini toping, u yerda JDK versiyasini o'zgartiramiz. Bu 11 yoki undan yuqori versiya bo'lishi kerak. Qoidaga ko'ra, Android Studio bilan bir qatorda JDK ning qo'llab-quvvatlanadigan versiyasi ham o'rnatilgan - hozirda u JDK 11. Va uni JDK ro'yxatida tanlashingiz mumkin:



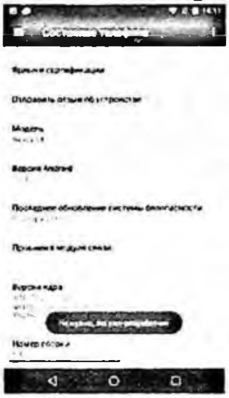
1.13-rasm. Gradle JDK ning qo'llab-quvvatlanadigan versiyasini o'rnatish

Android Studio bilan birga kelgan JDK versiyalarini tanlash uchun eng maqbul element o'rnatilgan JDK versiyasi deb ataladi. Skrinshotda ko'rib turganingizdek, bu 11-versiya, ammo Android Studio dasturining keyingi yangilanishlari bilan bu versiya o'zgarishi mumkin. O'zgarishlarni amalga oshirgandan so'ng, avval **Apply** tugmasini, keyin esa OK tugmasini bosing. Va keling, loyihani qayta boshlaymiz.

**Loyihani boshlash.** Oldingi mavzuda yaratilgan, allaqachon dastlabki funktsiyalarni o'z ichiga olgan standart loyiha. To'g'ri, bu funksionallik deyarli hech narsa qilmaydi, faqat "Hello_world!" qatorini namoyish etadi.

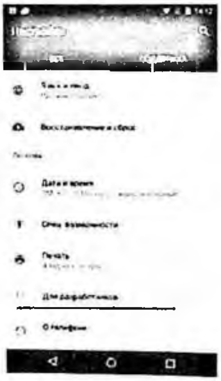
Ilovani ishga tushirish va sinovdan o'tkazish uchun biz emulyatorlardan yoki haqiqiy qurilmalardan foydalanishimiz mumkin. Ammo ideal holda haqiqiy qurilmalarda sinab ko'rish yaxshidir. Bundan tashqari, emulyatorlar katta apparat manbalarini talab qiladi va har bir kompyuter emulyatorlar talablarini bajara olmaydi. Va sinov uchun mobil qurilmadan foydalanish uchun sizga faqat kerakli drayverni o'rnatishingiz kerak bo'lishi mumkin.

**Telefonda ishlab chiquvchi rejimi.** Odatiy bo'lib, ishlab chiquvchi parametrlari smartfonlarda yashiringan. Ularni taqdim etish uchun siz Settings> About Phone (Настройки > О телефоне) -ga o'tishingiz kerak (Android 8-da u Settings> System> About Phone-da (Настройки > Система > О телефоне) va Build Number(Номер сборки)ni etti marta bosing.



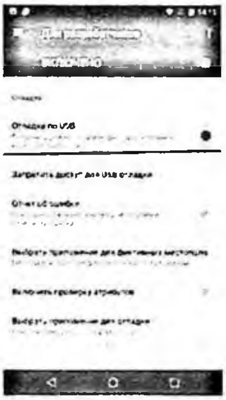
1.13-rasm. Mobil telefon holati

Oldingi ekranga qayting va u erda Developer options (Для разработчика) elementini ko'rasiz.



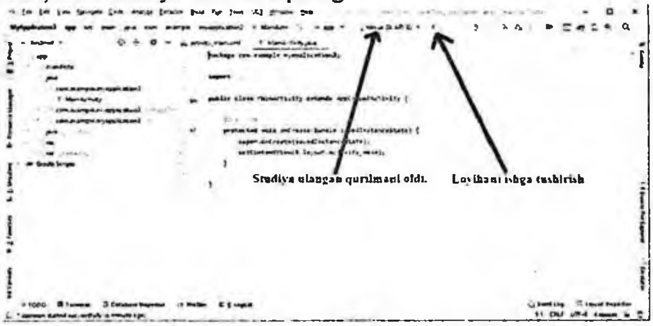
1.14-rasm. Mobil telefon: **Для разработчиков** bo`limi

**Для разработчиков** ga o'tamiz va USB orqali disk raskadrovka qilishni yoqamiz:



1.15-rasm. Mobil telefon: **USB tarmoq orqali ulanish** bo`limi

**Ilovani ishga tushirish.** Qurilmani Android OS bilan ulang (agar biz haqiqiy qurilmada sinovdan o'tayotgan bo'lsak) va asboblar panelidagi yashil strelka (ko`rsatkichni)ni bosib loyihani boshqaring.



1.16-rasm. Emulyator tanlash bo`limi

Qurilmani tanlang va OK tugmasini bosing. Ishga tushirgandan so'ng, biz dasturni qurilma ekranida ko'ramiz:

1.17-rasm. Dastur natijasi

## 1.3. Grafik interfeys yaratish

Oxirgi mavzuda biz Android Studio boshlang`ich holat bo'yicha taqdim etadigan va oddiygina Hello Android qatorini ko'rsatadigan oddiy dasturni qanday yaratishni ko'rib chiqdik. Lekin nima uchun biz ushbu aniq chiziqni namoyish qilamiz?

Nega biz bunday vizual interfeysni umuman yaratamiz?

Standart Android dasturining bajarilishi Android Studio-da boshlang`ich holat bo'yicha ochilgan MainActivity sinfidan boshlanadi:

```
package com.android.helloapp;
import androidx.appcompat.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);
   }
}
```

Ilovadagi har bir alohida ekran yoki sahifa **activity** kabi tushuncha bilan tavsiflanadi. Adabiyotda turli xil atamalardan foydalanish mumkin: ekran, sahifa, faoliyat. Bunday holda men "**activity**" atamasidan foydalanaman. Shunday qilib, agar biz dasturni qurilmada ishga tushiradigan bo'lsak, unda ekranda biz aslida ushbu interfeysni ifodalovchi ma'lum bir **activity**-ni ko'ramiz.

MainActivity sinfi asosan muntazam java sinfidir, uning boshida ushbu sinfning paketli ta'rifi mavjud:

```
package com.android.helloapp;
```

Keyingi - boshqa paketlardan sinflarni import qilish, ularning funktsional imkoniyatlari MainActivity-da ishlatiladi:

*import androidx.appcompat.app.AppCompatActivity;*

*import android.os.Bundle;*

Keyin haqiqiy sinf ta'rifi keladi:

*public class MainActivity extends AppCompatActivity*

Boshlang`ich holat bo`yicha MainActivity AppCompatActivity-dan meros qilib oladi, u yuqorida import direktivasi yordamida ulangan. AppCompatActivity sinfi asosan dasturning alohida ekranini (sahifasini) yoki uning ingl. Va MainActivity ushbu funktsiyalarning barchasini meros qilib oladi.

Boshlang`ich holat bo`yicha MainActivity faqat bitta onCreate() usulini o'z ichiga oladi, bu aslida barcha dastur interfeysini yaratadi:

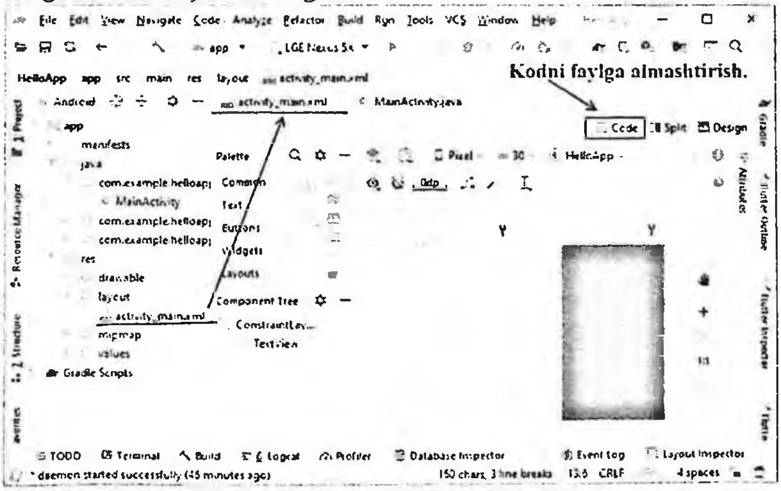
```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
}
```

GUI(Grafik interfeys o`lchami) belgilash resursi **setContentView()** uslubiga o'tkaziladi:

*setContentView(R.layout.activity_main);*

Aynan shu yerda MainActivity qanday vizual interfeysga ega bo'lishiga qaror qilinadi. Ammo bu holda R.layout.activity_main resursi nimani anglatadi?

Bu **res/layout** papkasidagi activity_main.xml fayli (printsipial ravishda resurs nomi fayl nomiga mos kelishini ko'rishingiz mumkin), bu Android Studio-da boshlang`ich holat bo'yicha ochilgan:



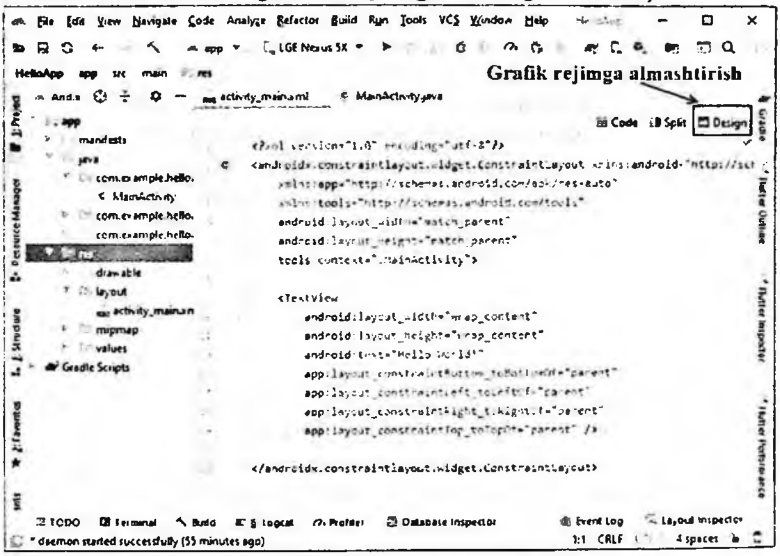
1.18-rasm. Dastur ishchi oynasi bo`limi

**Activity_main.xml fayli.** Android Studio sizga kod rejimida ham, grafik rejimida ham vizual interfeys bilan ishlashga imkon beradi. Shunday qilib, boshlang`ich holat bo'yicha fayl grafik rejimda ochiladi va biz dastur ekranining

25815

qanday ko'rinishini aniq ko'rishimiz mumkin. Hatto tugmachalar yoki matn maydonlari kabi asboblar panelidagi ba'zi boshqaruv elementlarini ham eskizlang.

Ammo biz fayl bilan kod rejimida ham ishlashimiz mumkin, chunki activity_main.xml - bu xml belgilariga ega oddiy matnli fayl. Kodga o'tish uchun grafik ko'rinish ustidagi **Code** tugmasini bosing. (Bundan tashqari, **Split** tugmasi yordamida siz ko'rish kodi + grafik dizaynerga o'tishingiz mumkin)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HelloApp app src main res
Grafik rejimga almashtirish
activity_main.xml
MainActivity.java
Code Split Design
app
manifests
java
com.example.hello
MainActivity
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
TODO Terminal Build Logcat Profiler Database Inspector Event Log Layout Inspector

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sch
    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">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

daemon started successfully (55 minutes ago)

1.19-rasm. Grafik rejimga almashtirish

Bu yerda, kod darajasida, activity_main.xml fayli quyidagi belgilashni o'z ichiga olganligini ko'rishimiz mumkin:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
 </androidx.constraintlayout.widget.ConstraintLayout>
```

Barcha interfeys **androidx.constraintlayout.widget.ConstraintLayout** konteyner elementi bilan ifodalanadi:

```
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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">
```

**ConstraintLayout** ichki elementlarni ekrandagi ma'lum joylarda joylashtirishga imkon beradi. **ConstraintLayout** elementi XML nom maydonlarining ta'rifidan boshlanadi:

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
```

Har bir nom maydoni quyidagicha ko'rsatilgan: xmlns: prefiks = "resurs_nomi". Masalan,

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

Resurs nomi (yoki URI – Uniform Resource Indicator) - "http://schemas.android.com/apk/res/android". Resurs android prefiksiga (xmlns: android) mos keladi.

Nima uchun bu ism maydonlari kerak? Har bir resurs yoki URI dasturda ishlatiladigan ba'zi bir funktsiyalarni belgilaydi, masalan, dasturni yaratish uchun zarur bo'lgan teglar va atributlarni taqdim etish.

- xmlns: android = "http://schemas.android.com/apk/res/android": Android platformasi tomonidan ta'minlanadigan, boshqaruv elementlariga qo'llaniladigan va ularning vizual xususiyatlarini aniqlaydigan (masalan, o'lcham, joylashishni aniqlash) asosiy xususiyatlarni o'z ichiga oladi.
- xmlns: app = "http://schemas.android.com/apk/res-auto": dastur ichida aniqlangan atributlarni o'z ichiga oladi
- xmlns: tools = "http://schemas.android.com/tools": Android Studio-da dizaynerlar rejimi bilan ishlash uchun ishlatiladi

Ushbu resurslar bilan ishlashni osonlashtirish uchun prefikslar qo'llaniladi. Masalan, biz quyidagilarni ko'ramiz:

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
```

android: layout_width konteynerning kengligini belgilaydi. Ushbu atribut (layout_width) "http://schemas.android.com/apk/res/android" resursida joylashgan. Va bu resurs android prefiksiga bog'langanligi sababli, ushbu resursning prefiksi undan oldin nuqta bilan ajratilgan atributga murojaat qilish uchun ko'rsatiladi.

Android: layout_weight atributi "match_parent" dir. Bu shuni anglatadiki, (ConstraintLayout) elementi konteynerning butun kengligi bo'ylab cho'ziladi (qurilma ekrani).

Android: layout_height = "match_parent" atributi konteyner balandligini belgilaydi va "http://schemas.android.com/apk/res/android" da belgilanadi. "Match_parent" qiymati ConstraintLayout konteynerning butun uzunligiga (qurilma ekrani) qadar cho'zilishini bildiradi.

tools: context atributi qaysi activity (dastur ekrani) sinfining joriy interfeys ta'rifi bilan bog'liqligini aniqlaydi. Bunday holda, bu MainActivity sinfidir. Bu Android Studio-ga dastur ko'rinishini dizayner rejimida oldindan ko'rishni tashkil qilishda yordam beradi.

**TextView.** Matn maydoni **android: text** atributi yordamida matnni o'rnatadi.

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

- android: layout_width vidjetning kengligini belgilaydi. Wrap_content qiymati vidjetni konteynerda ko'rsatish uchun etarli hajmga o'rnatadi.
- android: layout_height vidjetning balandligini belgilaydi. Wrap_content qiymati, kenglikni o'rnatishga o'xshash, vidjet balandligini konteynerda ko'rsatish uchun etarli darajada o'rnatadi
- android: text TextView-da ko'rsatiladigan matnni o'rnatadi (bu holda "Hello World!" qatori)
- app: layout_constraintLeft_toLeftOf = "parent": elementning chap chegarasi ConstraintLayout konteynerining chap tomoniga to'g'ri kelishini bildiradi.

Shuni esda tutingki, ushbu atribut nomlar fazosida ilova prefiksi bilan, ya'ni "http://schemas.android.com/apk/res-auto" bilan belgilanadi.

- app: layout_constraintTop_toTopOf = "parent": elementning yuqori chegarasi ConstraintLayout konteynerining yuqori qismiga to'g'ri kelishini bildiradi.
- app: layout_constraintRight_toRightOf = "parent": elementning o'ng chegarasi ConstraintLayout konteynerining o'ng tomoniga to'g'ri kelishini bildiradi.
- app: layout_constraintBottom_toBottomOf = "parent": elementning pastki chegarasi ConstraintLayout konteynerining pastki qismiga to'g'ri kelishini bildiradi.

Shuni ta'kidlash kerakki, so'nggi to'rtta atributlar TextView-ning ekranda markazlashishiga olib keladi.

Shunday qilib, dastur ishga tushganda, birinchi navbatda MainActivity sinfi ishga tushiriladi, bu esa activity_main.xml faylidan GUI sifatida belgilanishni o'rnatadi. Va ushbu belgi tarkibida bir nechta matnni ifodalovchi TextView elementi bo'lganligi sababli, biz uning matnini smartfon ekranida ko'rib chiqamiz.

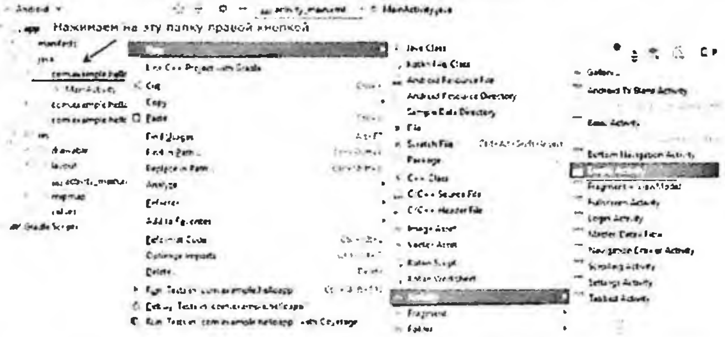
## 1.4. Birinchi ilovangizni yaratish va Activity-ni qo'shish.

Oldingi mavzularda biz birinchi loyihani yaratdik, ammo butun loyiha faqat boshlang`ich holat bo'yicha qo'shilgan ibtidoiy koddan iborat edi. Endi dasturni o'zimiz aniqlaymiz, bu ham juda sodda bo'ladi. Deylik, biz dasturning bir sahifasiga ba'zi ma'lumotlarni kiritamiz va tugmachani bosish orqali avval kiritilgan ma'lumotlarni ko'rsatadigan dasturning boshqa sahifasiga o'tish bo'ladi.

Shunday qilib, ilgari yaratilgan loyihani olaylik (yoki yangisini yarating).

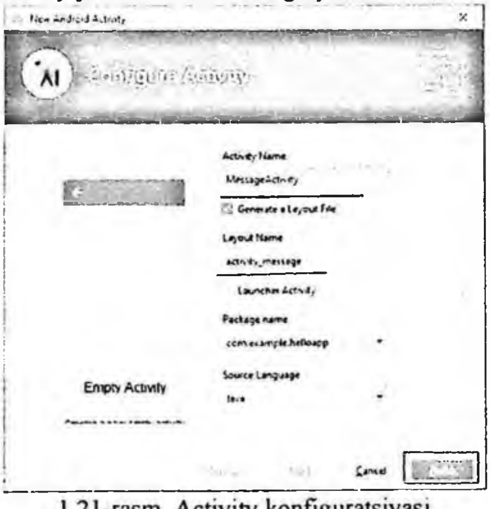
***Yangi Activity qo'shish.***

Boshlang`ich holat bo'yicha bizda allaqachon bir activity mavjud - MainActivity sinfi. Endi boshqasini qo'shaylik. Buning uchun MainActivity sinfi joylashgan papkada loyiha tuzilmasiga o'ng tugmasini bosing va keyin kontekst menyusida New-> Activity-> Empty Activity-ni tanlang:



1.20-rasm. Yangi Activity qo'shish

Bu yangi activity yaratish uchun dialog oynasini ochadi:


New Android Activity

Configure Activity

Activity Name: MessageActivity

☑ Generate a Layout File

Layout Name: activity_message

Launcher Activity

Package name: com.example.helloapp

Source Language: Java

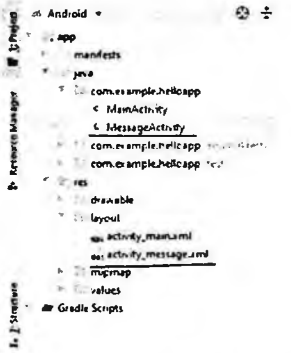
1.21-rasm. Activity konfiguratsiyasi

Ushbu oynada, Activity Name maydoniga MessageActivity-ni kiriting. Shundan so'ng, Layout Name maydonida activity_message avtomatik ravishda o'rnatilishi kerak (agar o'rnatilmagan bo'lsa, biz ushbu maydonga activity_message-ni kiritamiz). Qolgan maydonlarda standart qiymatlarni qoldiring:

- Package Name - MainActivity o'z ichiga olgan paket bilan bir xil nomga ega bo'lishi kerak
- Source Language Java bo'lishi kerak

Keyin Finish tugmachasini bosing.

Shundan so'ng MainActivity papkasida yangi activity - MessageActivity fayllari paydo bo'lishi kerak:



1.22-rasm. MessageActivity yangi activity fayllari

Bundan tashqari, MesageActivity uchun GUI tavsifini ifodalovchi res / layout katalogida activity_message.xml fayli paydo bo'lishi kerak.

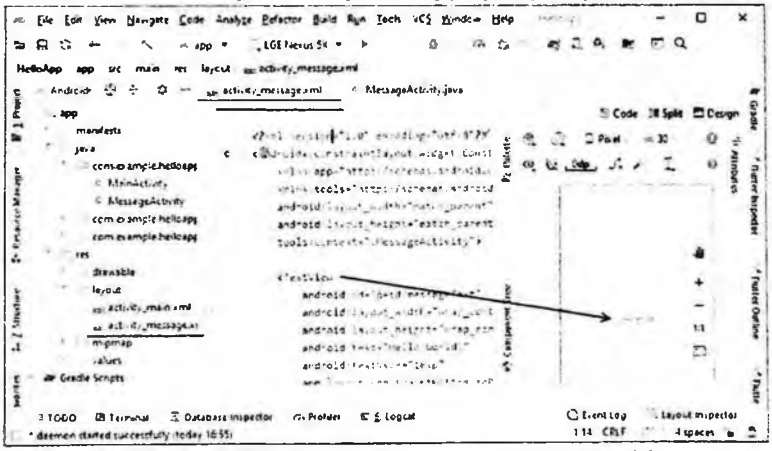
Birinchidan, activity_message.xml faylini oching va uning kodini quyidagicha o'zgartiring:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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=".MessageActivity">
  <TextView
    android:id="@+id/messageText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="18sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

*</androidx.constraintlayout.widget.ConstraintLayout>*

MessageActivity interfeysi ta'rifi faqat bitta satrni ekranga ko'rsatadigan TextView elementini o'z ichiga oladi. Unda o'rnatilgan xususiyatlarni ko'rib chiqing:

- android: id = "@ + id / messageText" - TextView-da java kodidagi TextView-ga murojaat qilishimiz mumkin bo'lgan "messageText" identifikatori mavjud. @ Belgisi XML tahlil qiluvchiga atribut satrining qolgan qismini identifikator sifatida ishlatishini aytadi. Va "+" belgisi shuni anglatadiki, agar he___ader qiymatiga ega bo'lgan i_d element uchun aniqlanmagan bo'lsa, u holda uni aniqlash kerak.
- android: layout_width = "wrap_content" - matn maydonining kengligi barcha tarkibni ekranda joylashtirish uchun etarli bo'ladi
- android: layout_height = "wrap_content" - TextView-ning balandligi uning tarkibidagi barcha tarkibga mos keladigan darajada bo'ladi.
- android: textSize = "18sp" - TextView-dagi matn balandligi 18 birlik (sp qiymati shrift balandligini o'rnatish uchun ishlatiladi)
- a_pp: layout_constraintBottom_toBottomOf = "parent" - TextView-ning pastki qismi ConstraintLayout konteynerining pastki qismiga to'g'ri keladi
- a_pp: layout_constraintLeft_toLeftOf = "parent" - TextView-ning chap chegarasi ConstraintLayout konteynerining chap tomoniga tekislanadi
- a_pp: layout_constraintRight_toRightOf = "parent" - TextView-ning o'ng qirrasi ConstraintLayout konteynerining o'ng tomoniga tekislanadi
- a_pp: layout_constraintTop_toTopOf = "parent" - TextView-ning yuqori tomoni ConstraintLayout konteynerining yuqori qismiga to'g'ri keladi



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.Const
    xmlns:app="http://schemas.android.c
    xmlns:tools="http://schemas.android
    android:layout_width="match_parent"
    android:layout_height="match_parent
    tools:context=".MessageActivity">

    <TextView
        android:id="@+id/messageText"
        android:layout_width="wrap_cont
        android:layout_height="wrap_con
        android:text="Hello World!"
        android:textSize="18sp"
        [illegible]
```

1.23-rasm. XML da yangi komponent kodini yaratish

Ma'lumotlarni qabul qilish. MessageActivity ning mohiyati shundaki, u biron bir matnli xabarni qabul qiladi va uni ekranda namoyish etadi (rasmiy ravishda yuqorida ko'rsatilgan TextView elementida). Qanday qilib MessageActivity-ga (va boshqa har qanday activity-da) boshqa activity-dan uzatiladigan ba'zi ma'lumotlarni olishimiz mumkin?

MessageActivity sinfining kodini olaylik. Odatiy bo'lib, quyidagicha ko'rinadi:

```
package com.example.helloapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MessageActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_message);
    }
}
```

Va buni quyidagicha o'zgartiraylik:

```
package com.example.helloapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;
public class MessageActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_message);
        // Ushbu activity-ni ishga tushirgan Intent ob'ektini oliadi.
        Intent intent = getIntent();
        // intent ob'ektidan xabarni olish
        String message = intent.getStringExtra("message");
        // Biz TextView-ni id orqali olamiz
        TextView messageText = (TextView) findViewById(R.id.messageText);
        // biz TextView uchun matnni o'rnatdik.
        messageText.setText(message);
    }
}
```

Xuddi MainActivity-da bo'lgani kabi (va boshqa activity-da), joriy activity-ning yaratilishi onCreate () usulida amalga oshiriladi. Barcha activity sinflari onCreate usulini amalga oshirishi kerak, chunki tizim yangi ac_tivity yaratilganda uni chaqiradi. Aynan shu usulda yangi activity ob'ekti setContentView usuli yordamida o'rnatiladi va bu yerda komponentlarning dastlabki konfiguratsiyasi sodir bo'ladi.

Har bir Activity ob'ekti Intent ob'ekti tomonidan chaqiriladi. Intent kodini getIntent usuli yordamida Java kodida olishimiz mumkin:

*Intent intent = getIntent();*

Ammo niyat biz uchun o'z-o'zidan muhim emas, chunki u orqali ma'lumotlar ushbu activity-ga o'tkaziladi. Ushbu ma'lumotlar har xil turlarni ifodalashi mumkin, ammo biz bu erda magistral MainActivity-ga uzatilishini taxmin qilamiz. Intent ob'ektidan satrini olish uchun getStringExtra() usuli quyidagicha nomlanadi:

```
String message = intent.getStringExtra("message");
```

Ma'lumotlar kaliti usulga o'tkaziladi. Ya'ni, uzatilgan ma'lumotlarning har bir elementi kalit-qiymat formatida namoyish etiladi. Biz qiymatni kalit orqali olishimiz mumkin. Va bu holda, kalit "message" bo'ladi. Va bu kalit uchun qiymat String message o'zgaruvchisiga o'tadi.

MessageActivity-ga uzatilgan ma'lumotlarni qabul qilib, biz uni ekran ekranida ko'rish uchun TextView-ga o'tkazamiz. Buni amalga oshirish uchun avval TextView-ni id tomonidan topamiz va keyin TextView-da ko'rsatilgan matnni o'rnatadigan setText() usulini chaqiramiz:

```
TextView messageText = (TextView) findViewById(R.id.messageText);
messageText.setText(message);
```

Shunday qilib, MessageActivity unga yuborilgan xabarni qabul qiladi va TextView-ga chiqaradi. Keling, ushbu activity-ni MainActivity-dan qanday chaqirishni va ushbu xabarni qanday uzatishni ko'rib chiqamiz.

## 1.5. Ikkinchi Activity ishga tushirish jarayoni

Oxirgi mavzuda biz ba'zi ma'lumotlarni tashqaridan qabul qiladigan MessageActivity-ni aniqladik. Endi MessageActivity-ni boshlaydigan va unga ba'zi ma'lumotlarni uzatadigan MainActivity-dagi kodni aniqlaymiz.

**Interfeys ta'rifi.** Activity_main.xml faylini quyidagicha o'zgartiramiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">
   <EditText
     android:id="@+id/editText"
     android:layout_width="0dp"
     android:layout_height="wrap_content"
     android:layout_marginStart="16dp"
     android:layout_marginTop="16dp"
     android:hint="Введите текст"
     app:layout_constraintRight_toLeftOf="@+id/button"
     app:layout_constraintLeft_toLeftOf="parent"
     app:layout_constraintTop_toTopOf="parent" />
  <Button
     android:id="@+id/button"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_marginEnd="16dp"
```

```
        android:layout_marginStart="16dp"
        android:text="Отправить"
        android:onClick="sendMessage"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/editText" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu erda biz ConstraintLayout-ga ikkita element qo'shdik: E_ditText (matn kiritish maydoni) va B_utton (tugma). Keling, ular nimani anglatishini alohida ko'rib chiqaylik.

MessageActivity-ga uzatiladigan matnni kiritish uchun EditText elementi qo'shilgan:

```
<EditText
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:hint="Введите текст"
        app:layout_constraintRight_toLeftOf="@+id/button"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Shunday qilib, biz quyidagi atributlarni aniqladik:

- android: id: biz ob'ektga murojaat qilishimiz mumkin bo'lgan noyob vidjet identifikatorini taqdim etadi
- android: layout_width elementning kengligini belgilaydi. Bunday holda, qiymat "0dp" dir. Darhaqiqat, kenglik ConstraintLayout konteyneridan keyin keladigan qo'shimcha atributlar asosida o'rnatiladi.
- android: layout_height idishni balandligini belgilaydi. Wrap_content qiymati vidjet uchun hajmini idishda ko'rsatish uchun etarli qilib belgilaydi
- android: layout_marginStart: konteynerning chap chetidan ofsetni o'rnatadi. Bunday holda, 16 birlik
- android: layout_marginTop: konteynerning yuqori chetidan ofsetni o'rnatadi. Bunday holda, 16 birlik
- android: hint: matn maydoniga sinov maslahatini o'rnatadi
- app: layout_constraintRight_toLeftOf: EditText elementning chap tomonida joylashganligini bildiradi, uning id qiymati qiymat sifatida ko'rsatilgan. Shunday qilib, bu holda "@ + id / button" qiymati bizning tugmachamizning identifikatorini anglatadi. Ya'ni, EditText-ning o'ng tomoni Button-ning chap tomoniga to'g'ri keladi.
- app: layout_constraintLeft_toLeftOf = "parent": elementning chap chegarasi ConstraintLayout konteynerining chap tomoni bo'ylab harakatlanishini belgilaydi (yuqoridagi to'plam to'ldirilgan holda)

- app: layout_constraintTop_toTopOf = "parent": elementning yuqori chegarasi ConstraintLayout konteynerining yuqori qismidan o'tishini belgilaydi (yuqoridagi to'plamga muvofiq)

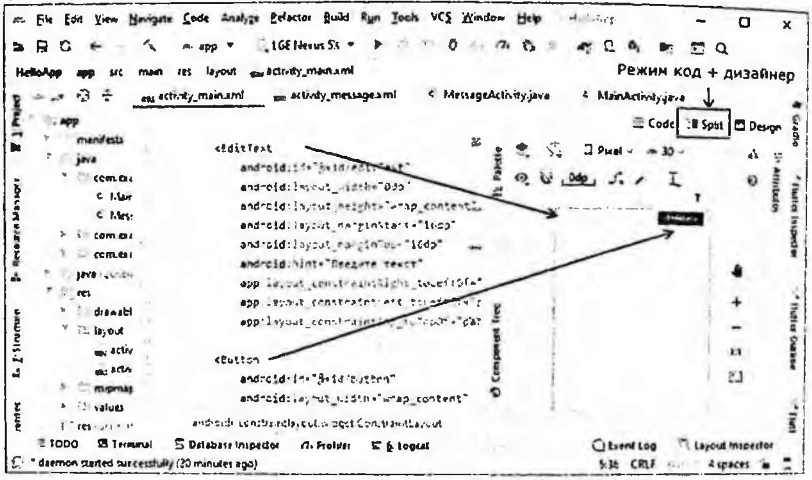
Endi tugmachaning kodini - MessageActivity-ni ishga tushiradigan Button elementini ko'rib chiqamiz.

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="16dp"
    android:layout_marginStart="16dp"
    android:text="Отправить"
    android:onClick="sendMessage"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toRightOf="@+id/editText" />
```

Tugma elementi quyidagi atributlarni qo'llaydi:

- android: id: tugma id - tugma.
- android: layout_width - tugma kengligi wrap_content-ga o'rnatiladi, shuning uchun tugma uning matnini ko'rsatish uchun etarli bo'lgan kenglikka ega bo'ladi.
- android: layout_height - wrap_content qiymati vidjetni matnni ko'rsatish uchun etarli hajmga o'rnatadi
- android: layout_marginStart: elementning shartli chap chegarasidan chuqquni o'rnatadi - bu holda u EditText matn maydonining o'ng chekkasidir. Bunday holda, bu 16 birlik
- android: layout_marginTop: konteynerning yuqori chetidan ofsetni o'rnatadi. Bunday holda, 16 birlik
- android: text: tugma sinovini belgilaydi
- android: onClick: tugmani bosish uchun ishlov beruvchini o'rnatadi. Atributga berilgan "sendMessage" qiymati bu bog'liq faoliyat sinfida aniqlangan usulning nomi (bu holda, MainActivity). Ya'ni, tugma bosilganda sendMessage usuli chaqiriladi, biz uni yanada aniqlaymiz.
- app: layout_constraintLeft_toRightOf = "@ + id / editText": Tugmaning chap tomonini EditText-ning o'ng chegarasiga o'rnatilishini o'rnatadi. = "@ + Id / editText" qiymati tugmachaning pastki chegarasi elementning pastki chegarasi bilan id editText, ya'ni matn maydoniga to'g'ri kelishini bildiradi.
- app: layout_constraintTop_toTopOf = "parent": elementning yuqori chegarasi to'ldirilgan holda ConstraintLayout konteynerining yuqori qismidan o'tishini belgilaydi.
- app: layout_constraintRight_toRightOf: elementning o'ng chegarasi ConstraintLayout konteynerining o'ng tomoni bo'ylab harakatlanishini bildiradi.

Agar biz dizayner rejimiga o'tsak, masalan split rejim (kod + dizayner), unda biz quyidagi tartibni ko'ramiz:



```
<EditText
    android:id="@+id/editText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:hint="Введите текст"
    app:layout_constraintRight_toLeftOf="[illegible]
    app:layout_constraintLeft_toLeftOf="[illegible]
    app:layout_constraintTop_toTopOf="par

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
androidx.constraintlayout.widget.ConstraintLayout
```

1.24-rasm. Dastur dizayni va kodi

**Tugmani bosish bilan ishlov berish.** Shunday qilib, yuqorida biz tugmachani bosish bilan ishlashni aniqladik: android: onClick = "sendMessage". Biz tugmachani bosganimizda MessageActivity ishga tushiriladigan sendMessage usuli ishga tushirilishini taxmin qilamiz. Ushbu usulni qo'shish uchun MainActivity sinfining kodini o'zgartiramiz:

```
package com.example.helloapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   // Tugmani bosish bilan ishlash usuli
   public void sendMessage(View view) {
      // tugmachani bosgandan so'ng amalga oshiriladigan harakatlar
      // Yangi Activity-ga chaqirish uchun Intent ob'ektini yarating
      Intent intent = new Intent(this, MessageActivity.class);
      // Joriy Ac_tivity-da matn maydonini olish
      EditText editText = (EditText) findViewById(R.id.editText);
      // Berilgan matn maydonining matnini oling
      String message = editText.getText().toString();
```

```
            // PutExtra xususiyatidan foydalanib, ob'ekt qo'shing - birinchi
parametr kalit,
            // ikkinchi parametr - bu ob'ektning qiymati
            intent.putExtra("message", message);
            // activity ishga tushiradi
            startActivity(intent);
        }
    }
```

Tugmachani bosish qayta ishlanmasi - sendMessage() usuli parametr sifatida bosilgan tugmachani aks ettiruvchi View ob'ektini olishi kerak:

*public void sendMessage(View view) {*

Ikkinchi activity-ni boshlash uchun Intent ob'ekti kerak. Intent ob'ekti bajarilishi kerak bo'lgan ba'zi bir dastur vazifalarini ifodalaydi (masalan, activity boshlash):

*Intent intent = new Intent(this, MessageActivity.class);*

Ushbu ob'ektning konstruktori ikkita parametrni oladi:

- Birinchi parametr kontekstni ifodalaydi - Context ob'ekti (bu erda this kalit so'zi ishlatiladi, chunki MainActivity sinfi Context sinfining pastki sinfi)
- Ikkinchi parametr - bu I_ntent ob'ektini o'tkazadigan komponentlar sinfi. Avvalgi mavzuga qo'shilgan MessageActivity sinfining ob'ekti xuddi shunday ishlaydi.

SendMessage() usuli ichida biz EditText elementini olish va kiritilgan matnni intent ob'ektiga o'tkazish uchun findViewById usulidan foydalanamiz:

*EditText editText = (EditText) findViewById(R.id.editText);*
*String message = editText.getText().toString();*

Keyin matn maydonidan olingan matn boshlanadigan activity-ga uzatiladi:

*intent.putExtra("message", message);*

"message" parametri uzatilayotgan ma'lumotlarning kalitini bildiradi. Ya'ni, biz yangi a_ctivity-ga juda ko'p ma'lumotlarni uzatishimiz mumkin va ular chegaralanishi uchun ular uchun kalit o'rnatiladi. Men sizning e'tiboringizni shu erda, biz MessageActivity-da ushbu ma'lumotlarni oladigan kalit yordamida uzatamiz:

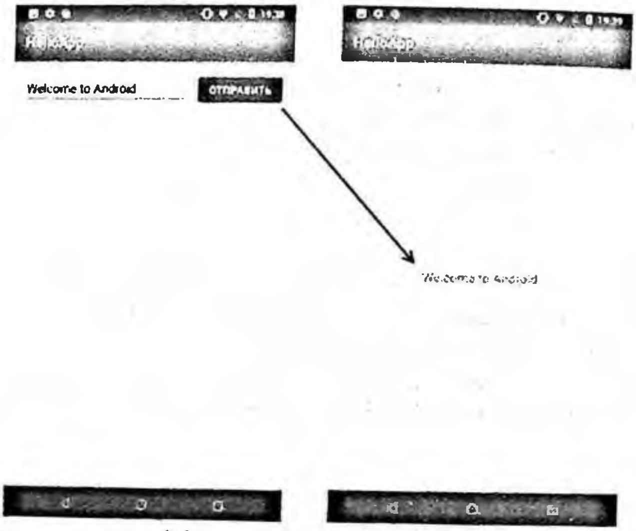
*String message = intent.getStringExtra("message");*

activity-ni boshlash uchun startActivity() usulini chaqirishingiz va parametr sifatida Intent ob'ektini o'tkazishingiz kerak:

*startActivity(intent);*

Ushbu usulni chaqirgandan so'ng tizim signal oladi va Intent ob'ekti tomonidan aniqlangan yangi Activity-ni boshlaydi.

Endi loyihani ishga tushiramiz va dasturni ishga tushirgandan so'ng biz tugmachali matn maydonini ko'ramiz, ammo matnni kiritib, tugmachani bosgandan so'ng, avval kiritilgan ma'lumotlarni ko'rsatadigan yangi activity - MessageActivity ishga tushiriladi:



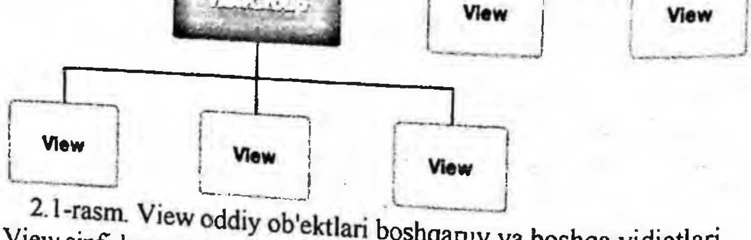
1.25-rasm. Dastur natijasi

# II-BOB. INTERFEYS ASOSLARI

## 2.1. Interfeys dizayniga kirish

Grafik foydalanuvchi interfeysi - bu android.view.View va android.view.ViewGroup ob'ektlarining iyerarxiyasi. Har bir ob'yekt V_iewGroup bolada View moslamalarini o'z ichiga olgan va joylashtirgan idishni aks ettiradi. Xususan, konteynerlarga RelativeLayout, LinearLayout, GridLayout, ConstraintLayout va boshqa bir qator elementlar kiradi.

View oddiy ob'ektlari boshqaruv va boshqa vidjetlarni, masalan foydalanuvchi dastur bilan o'zaro aloqada bo'lgan tugmachalar, matn qutilari va boshqalarni aks ettiradi:



2.1-rasm. View oddiy ob'ektlari boshqaruv va boshqa vidjetlari

View sinfidan meros qilib olingan vizual elementlarning aksariyati, masalan tugmachalar, matn qutilari va boshqalar android.widget paketida joylashgan.

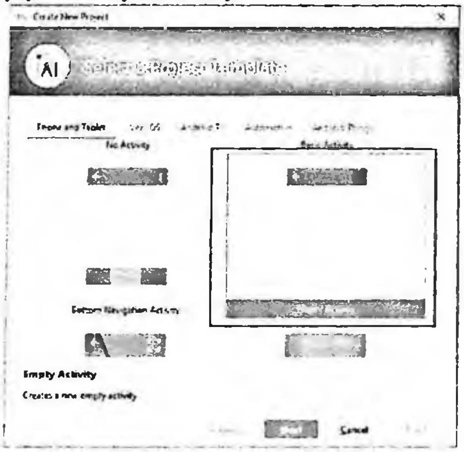
Vizualni belgilashda biz uchta strategiyaga egamiz:

- Java kodida dasturiy ta'minotni yarating
- XML-da interfeys elementlarini e'lon qiling
- Ikkalasining kombinatsiyasi - XML-da asosiy belgilash elementlarini aniqlang va qolganlarini ish vaqtida qo'shing

Dastlab, birinchi strategiyani - Java kodidagi interfeysni aniqlashni ko'rib chiqamiz.

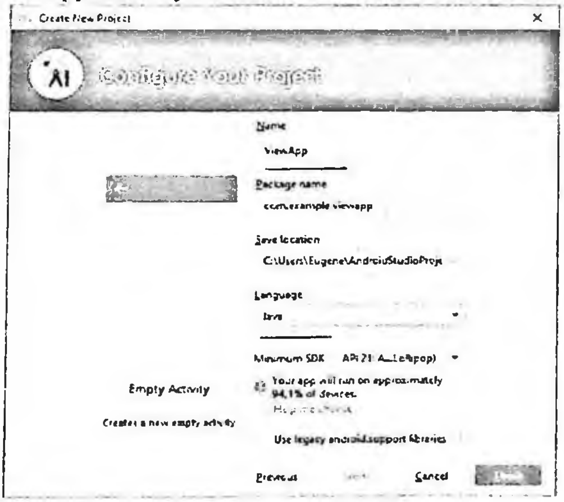
**Java kodida interfeys yaratish**

Vizual elementlar bilan ishlash uchun yangi loyiha yarataylik. Loyiha shabloni sifatida Empty Activity-ni tanlaymiz:



2.2-rasm. Yangi proyekt uchun qolip tanlash bo`limi

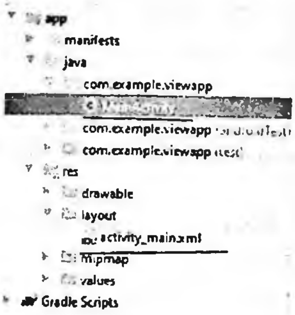
Buni ViewsApp deb ataymiz:



Name
ViewsApp
Package name
com.example.viewsapp
Save location
C:\Users\Eugene\AndroidStudioProj
Language
Java
Minimum SDK API 21: Android 5.0 (Lollipop)
Your app will run on approximately 94,1% of devices.
Use legacy android.support libraries

2.3-rasm. Yangi proyekt konfiguratsiyasi

Va loyihani yaratgandan so'ng, vizual interfeys yaratishda bizni qiziqtiradigan ikkita asosiy fayl MainActivity sinfi va activity_main.xml faylida ushbu faoliyat uchun interfeys ta'rifi.



2.4-rasm. Yangi proyekt strukturasi

MainActivity sinfidagi eng oddiy interfeysni aniqlaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // TextView yaratish
    TextView textView = new TextView(this);
    // matnni TextView-ga sozlash
    textView.setText("Hello Android!");
    // matn balandligini sozlash
    textView.setTextSize(22);
    // activity uchun vizual interfeysni sozlash
    setContentView(textView);
  }
}
```

Java kodida vidjetlarni yaratishda ularning konstruktoridan foydalaniladi, unga ushbu vidjetning konteksti, aniqrog'i android.content.Context ob'ekti, ya'ni hozirgi MainActivity sinfi kiradi.

*TextView textView = new TextView(this);*

Bu erda butun interfeys matnni namoyish qilish uchun mo'ljallangan TextView elementi bilan ifodalanadi. TextView-ning turli xil xususiyatlarini odatda s_et bilan boshlanadigan usullar yordamida sozlash mumkin. Masalan, bu holda setText() usuli matnni maydonga, setTextSize () esa shrift balandligini o'rnatadi.

Elementni dastur interfeysi sifatida o'rnatish uchun Activity kodi setContentView () usulini chaqiradi, u ingl.

Agar biz dasturni ishga tushiradigan bo'lsak, quyidagi vizual interfeysga ega bo'lamiz:

2.5-rasm. Dastur natijasi

Xuddi shunday, biz yanada murakkab o'zaro ta'sirlarni yaratishimiz mumkin. Masalan, ConstraintLayout ichida joylashgan TextView:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     ConstraintLayout constraintLayout = new ConstraintLayout(this);
     TextView textView1 = new TextView(this);
     textView1.setText("Hello Android!");
     textView1.setTextSize(26);
     // parametr parametrlari va elementning joylashishini o'rnatish
     ConstraintLayout.LayoutParams          layoutParams          =          new
ConstraintLayout.LayoutParams
          (ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
    // chapga tekislash ConstraintLayout
    layoutParams.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;
    // yuqori tekislash ConstraintLayout
    layoutParams.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;
    // textView uchun parametrlarni sozlash
    textView.setLayoutParams(layoutParams);
    // TextView-ni ConstraintLayout-ga qo'shing
    constraintLayout.addView(textView);
    // ildiz sifatida
    setContentView(constraintLayout);
```

Har bir konteyner uchun undagi elementni qo'shish va joylashtirish bo'yicha aniq harakatlar farq qilishi mumkin. Bunday holda, konteynerlar ConstraintLayout sinfidir, shuning uchun elementning joylashuvi va hajmini aniqlash uchun ConstraintLayout.LayoutParams ob'ekti yaratilishi kerak. (LinearLayout uchun bu mos ravishda LinearLayout.LayoutParams va RelativeLayout, RelativeLayout.LayoutParams va boshqalar uchun) Ushbu ob'ekt ikkita parametr bilan boshlangan: kenglik va balandlik. Kenglik va balandlikni belgilash uchun element o'lchamlarini ekrandagi tarkibiga mos ravishda o'rnatadigan ViewGroup.LayoutParams.WRAP_CONTENT doimiysidan foydalanishingiz mumkin.

Keyinchalik, joylashishni aniqlash belgilanadi. O'rnatilishi mumkin bo'lgan xususiyatlar to'plami konteyner turiga qarab farq qilishi mumkin. Shunday qilib, kod satri

*layoutParams.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;*

elementning chap chegarasi konteynerning chap chegarasiga to'g'ri kelishini bildiradi.

Kod satri quyidagicha:

*layoutParams.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;*

elementning yuqori chegarasi idishning yuqori qismiga to'g'ri kelishini bildiradi. Natijada, element ConstraintLayout-ning yuqori chap burchagiga joylashtiriladi.

Ushbu qiymatlarning barchasini ma'lum bir element (TextView) uchun o'rnatish uchun ViewGroup.LayoutParams ob'ekti (yoki uning merosxo'rlaridan biri, masalan, ConstraintLayout.LayoutParams) uning setLayoutParams() uslubiga o'tkaziladi.

*textView.setLayoutParams(layoutParams);*

Android.view.ViewGroup (RelativeLayout, LinearLayout, GridLayout, ConstraintLayout va boshqalar) dan meros bo'lib o'tgan barcha konteyner sinflarida bekor qilingan addView usuli mavjud (android.view.View child), bu konteynerga yana bir element qo'shish imkonini beradi - oddiy vidjet turi TextView yoki boshqa konteyner. Va bu holda, ushbu usul orqali TextView ConstraintLayout-ga qo'shiladi:

*constraintLayout.addView(textView);*

Shunga qaramay, men ma'lum bir konteyner uchun aniq harakatlar farq qilishi mumkinligini ta'kidlayman, ammo qoida tariqasida ularning barchasi uchta bosqich bilan tavsiflanadi:

- ViewGroup.LayoutParams ob'ektini yaratish va uning xususiyatlarini sozlash
- ViewGroup.LayoutParams ob'ektini setLayoutParams () uslubiga o'tkazish
- Konteyner ob'ektining addView () uslubiga qo'shish uchun elementni yuborish

Shunga o'xshash yondashuvdan foydalanishimiz mumkin bo'lsa-da, xml fayllarida vizual interfeysni aniqlash va faoliyat sinfidagi barcha tegishli mantiqlarni aniqlash eng maqbuldir. Shunday qilib, biz interfeys va dastur mantig'i o'rtasidagi farqga erishamiz, ularni ishlab chiqish va keyinchalik ularni o'zgartirish osonroq bo'ladi. Va buni keyingi mavzuda ko'rib chiqamiz.

## 2.2. XML faylida interfeysni aniqlash. Layout fayllari

Qoida tariqasida, Android loyihalarida vizual interfeysni aniqlash uchun maxsus xml fayllari ishlatiladi. Ushbu fayllar belgilash manbalari bo'lib, vizual interfeys ta'rifini XML sifatida saqlaydi. Ushbu yondashuv veb-saytlarni yaratishga o'xshaydi, bu erda interfeys HTML fayllarida va dastur mantig'i javascript kodida aniqlanadi.

XML fayllarida foydalanuvchi interfeysini e'lon qilish dastur interfeysini koddan ajratadi. Ya'ni java-ni o'zgartirmasdan interfeys ta'rifini o'zgartirishimiz mumkin. Masalan, dastur turli xil monitor yo'nalishlari, har xil moslama o'lchamlari, har xil tillar va boshqalar uchun XML fayllaridagi belgilanishni belgilashi mumkin. Bundan tashqari, XML-da formatlashni e'lon qilish interfeys tuzilishini tasavvur qilishni osonlashtiradi va disk raskadrovka qilishni osonlashtiradi.

Grafik interfeysning maket fayllari loyihada res/layout katalogida joylashgan. Odatiy bo'lib, bo'sh ac_tivity bilan loyiha yaratganingizda, allaqachon quyidagicha ko'rinishi mumkin bo'lgan bitta formatlash resurs fayli mavjud.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    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">
     <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Hello World!"
       app:layout_constraintBottom_toBottomOf="parent"
       app:layout_constraintLeft_toLeftOf="parent"
       app:layout_constraintRight_toRightOf="parent"
       app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Fayl interfeysni tashkil etuvchi barcha grafik elementlarni va ularning atributlarini belgilaydi. XML-da belgilashni yaratishda ba'zi qoidalarga amal qilish kerak: Har bir belgilash fayli View yoki ViewGroup ob'ektini ko'rsatishi kerak bo'lgan bitta ildiz elementini o'z ichiga olishi kerak.

Bunday holda, ildiz elementi TextView elementini o'z ichiga olgan ConstraintLayout elementidir.

Odatda, ildiz elementi foydalanishga mo'ljallangan XML nom maydonlarining ta'rifini o'z ichiga oladi. Masalan, ConstraintLayout-dagi standart kodda biz quyidagi atributlarni ko'rishimiz mumkin:

*xmlns:android="http://schemas.android.com/apk/res/android"* *xmlns:app="http://schemas.android.com/apk/res-auto"*

*xmlns:tools="http://schemas.android.com/tools"*

Har bir nom maydoni quyidagicha ko'rsatilgan: xmlns: prefiks = "resurs nomi". Masalan, ichida

*xmlns:android="http://schemas.android.com/apk/res/android"*

Resurs nomi (yoki URI – Uniform Resource Indicator) - "http://schemas.android.com/apk/res/android". Va manba android prefiksiga (xmlns: android) mos keladi. Ya'ni, prefiks orqali biz ushbu nom maydonining funksionalligiga murojaat qilishimiz mumkin.

Har bir nom maydoni dasturda ishlatiladigan ba'zi bir funktsiyalarni belgilaydi, masalan, dasturni yaratish uchun zarur bo'lgan teglar va atributlarni taqdim etish.

xmlns: android = "http://schemas.android.com/apk/res/android": Android platformasi tomonidan ta'minlanadigan, boshqaruv elementlariga taalluqli va ularning vizual xususiyatlarini belgilaydigan (masalan, o'lcham, joylashishni aniqlash) asosiy atributlarni o'z ichiga oladi. Masalan, ConstraintLayout kodi "http://schemas.android.com/apk/res/android" nom maydonidan quyidagi atributdan foydalanadi:

*android:layout_width="match_parent"*

xmlns: app = "http://schemas.android.com/apk/res-auto": dastur ichida aniqlangan atributlarni o'z ichiga oladi. Masalan, TextView kodida:

*app:layout_constraintBottom_toBottomOf="parent"*

xmlns: tools = "http://schemas.android.com/tools": Android Studio-da dizaynerlar rejimi bilan ishlash uchun ishlatiladi

Bu eng keng tarqalgan ismlar. Va odatda har bir ildiz elementi (faqat ConstraintLayout bo'lishi shart emas) ularni o'z ichiga oladi. Ammo, agar siz Android Studio-da grafik dizaynerdan foydalanishni rejalashtirmasangiz va butunlay xml kodida ishlashni xohlasangiz, unda "http://schemas.android.com/tools" nom maydonida hech qanday ma'no yo'q va uni olib tashlash mumkin.

Tuzilganida, har bir XML formatlash fayli View resursiga yig'iladi. Belgilangan resursni yuklash Activity.onCreate usulida amalga oshiriladi. Joriy activity ob'ekti uchun belgilashni o'rnatish uchun setContentView() uslubiga parametr sifatida havolani belgilash manbasiga o'tkazing.

*setContentView(R.layout.activity_main);*

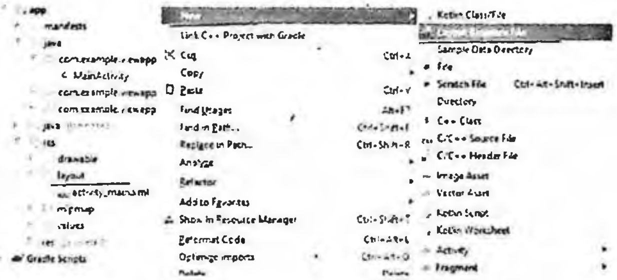
Java kodidagi manbaga havola olish uchun R.layout.[resurs nomi] iborasidan foydalanish kerak. Layout resurs nomi fayl nomi bilan bir xil bo'ladi, shuning uchun activity_main.xml faylini vizual interfeys manbai sifatida ishlatish uchun MainActivity sinfida quyidagi kodni belgilashingiz mumkin:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // interfeysni activity_main.xml faylidan yuklash
        setContentView(R.layout.activity_main);
    }
}
```

layout faylni qo'shilmoqda.

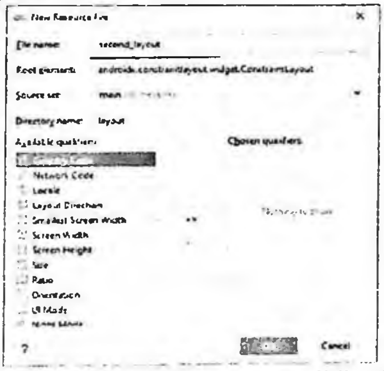
Ammo bizda bir nechta turli xil tartib manbalari bo'lishi mumkin. Odatda, har bir alohida Faoliyat sinfi o'z layout-dan foydalanadi. Yoki bir vaqtning o'zida bitta Activity sinfi uchun bir nechta turli xil layout fayllaridan foydalanish mumkin.

Masalan, loyihaga yangi interfeys belgilash fayli qo'shilsin. Buning uchun sichqonchaning o'ng tugmasi bilan res/layout papkasini bosing va paydo bo'lgan menyuda New->Layout->Resource File bandini tanlang:



2.6-rasm. Loyihaga yangi resurs faylini qo`shish

Shundan so'ng, maxsus oynada sizdan layout fayli uchun nom va ildiz elementini ko'rsatishingiz so'raladi:



File name: second_layout
Root element: androidx.constraintlayout.widget.ConstraintLayout
Source set: main
Directory name: layout

2.7-rasm. layout fayli uchun nom va ildiz elementi

Second_layout-ni nom sifatida ko'rsatamiz. Boshqa barcha sozlamalarni sukut bo'yicha qoldiring:

- Root elementi maydoni ildiz elementini bildiradi. Odatiy bo'lib, androidx.constraintlayout.widget.ConstraintLayout.
- Source set maydoni yangi faylni qaerga joylashtirishni belgilaydi. Odatiy bo'lib, bu main - dasturni ishlab chiqishda biz aslida ishlaydigan loyiha maydoni.

- Directory main maydoni yangi fayl aslida joylashtirilgan oldingi variantda tanlangan katalog ichidagi papkani belgilaydi. Interfeysi belgilanadigan fayllar uchun sukut layout hisoblanadi.

Shundan so'ng res / layout papkasiga second_layout.xml yangi fayli qo'shiladi, biz u bilan activity_main.xml bilan bir xil tarzda ishlashimiz mumkin. Xususan, second_layout.xml faylini ochamiz va tarkibini quyidagicha o'zgartiramiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/header"
    android:text="Welcome to Android"
    android:textSize="26sp"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

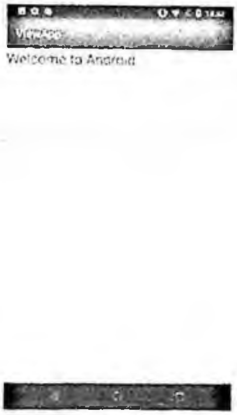
U quyidagi atributlarga ega bo'lgan TextView matn qutisini belgilaydi:

- android: id - bu element identifikatori, bu orqali biz unga kodda murojaat qilishimiz mumkin. Android: id = "@ + id / header" da @ belgisi XML tahlil qiluvchisiga atribut satrining qolgan qismini identifikator sifatida ishlatishini aytadi. + Belgisi shuni anglatadiki, agar elementda sarlavha qiymati bilan identifikatsiyalangan identifikator bo'lmasa, u aniqlanishi kerak.
- android: text - element matni - "Welcome to Android" qatori ekranda ko'rinadi.
- android: textSize - shrift balandligi (bu erda 26 birlik)
- android:layout_width - elementning kengligi. "match_parent" qiymati element ConstraintLayout konteynerining butun kengligi bo'ylab cho'zilishini bildiradi
- android: layout_height - elementning balandligi. "match_parent" qiymati element ConstraintLayout konteynerining to'liq balandligiga qadar cho'zilishini bildiradi

Ushbu faylni MainActivity sinfida GUI ta'rifi sifatida ishlatamiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.second_layout);
  }
}
```

Interfeys fayli second_layout.xml deb nomlangan, shuning uchun sukut bo'yicha u uchun R.layout.second_layout resursi yaratiladi. Shunga ko'ra, uni ishlatish uchun biz setContentView usuliga o'tamiz. Natijada, biz ekranda quyidagilarni ko'ramiz:



2.8-rasm. Dastur natijasi

**Vizuallarni kodda olish va boshqarish.** Yuqorida keltirilgan TextView elementi juda muhim xususiyatga ega - id elementi. Ushbu identifikator sizga Java kodidan xml faylida aniqlangan elementga murojaat qilish imkonini beradi. Masalan, MainActivity sinfiga o'tamiz va uning kodini o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // interfeys sifatida second_layout.xml faylini o'rnating
    setContentView(R.layout.second_layout);
    // textView elementini oling
    TextView textView = (TextView) findViewById(R.id.header);
    // matnni qayta o'rnating
    textView.setText("Hello from Java!");
  }
}
```

SetContentView () usuli, formatlashni second_layout.xml faylidan o'rnatadi.

Ta'kidlash kerak bo'lgan yana bir muhim narsa - bu Visual TextView-ni olish. Biz uning kodida android: id atributini aniqlaganimiz uchun uni ushbu id orqali olishimiz mumkin.

id tomonidan ma'lumotlar olish uchun Activity sinfida findViewById () usuli mavjud. Resurs identifikatori ushbu usulga R.id [element_identifikatori] shaklida uzatiladi. Ushbu usul View ob'ektini qaytaradi - barcha elementlar uchun asosiy sinf ob'ekti, shuning uchun usul natijasi hali TextView turiga chiqarilishi kerak.

Keyin biz ushbu element bilan biror narsa qilishimiz mumkin, bu holda biz uning matnini o'zgartiramiz.

Bundan tashqari, muhim bo'lgan narsa, ushbu vizual element aniqlangan setContentView uslubida belgilash o'rnatilgandan so'ng olinadi.

Agar biz loyihani olib boradigan bo'lsak, TextView-da yangi matn paydo bo'lishini ko'ramiz:

ViewApp
Hello from Java!

2.9-rasm. Dastur natijasi

## 2.3. O'lcham.

Android dasturlarini ishlab chiqishda biz har xil o'lchov turlaridan foydalanishimiz mumkin:

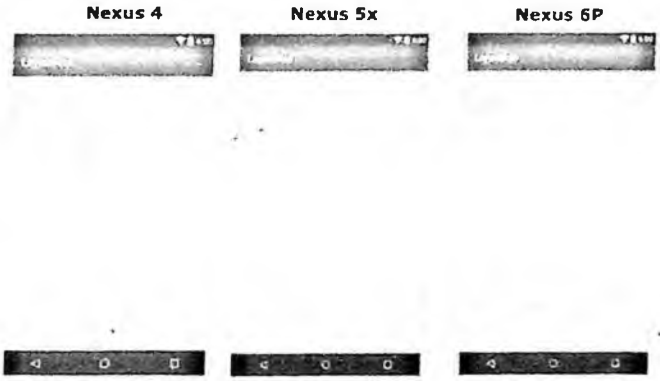
- px: joriy ekranning piksellari. Biroq, bu o'lchov birligi tavsiya etilmaydi, chunki tashqi ko'rinishning haqiqiy namoyishi qurilmaga qarab farq qilishi mumkin; har bir qurilmada dyuym uchun ma'lum bir piksellar to'plami mavjud, shuning uchun ekrandagi piksellar soni ham o'zgarishi mumkin
- dp: (device-independent pixels) zichlikdan mustaqil piksellar. 160 dpi (dyuymga nuqta) ekranning fizik zichligiga asoslangan mavhum o'lchov birligi. Bunday holda, 1dp = 1px. Agar ekran o'lchami 160 dpi dan katta yoki kichik bo'lsa, 1dp ko'rsatishda qo'llaniladigan piksellar soniga mos ravishda ko'paytiriladi yoki kamayadi. Masalan, 240dpi ekranda 1dp = 1,5px, 320dpi ekranda 1dp = 2px. Dp dan fizik piksellar sonini olishning umumiy formulasi: px = dp * (dpi / 160)
- sp: (scale-independent pixels) mustaqil piksellarni masshtablash. Foydalanuvchini sozlash imkonini beradi. Shriftlar bilan ishlash uchun tavsiya etiladi.
- pt: 1/72 dyuym, ekranning fizik o'lchamlari asosida
- mm: millimetr
- in: dyuym

Foydalanish uchun afzal birliklar dp. Buning sababi Android mobil qurilmalari dunyosi piksellar sonini va ekran o'lchamlari jihatidan juda parchalangan. Va dyuymdagi piksel zichligi qanchalik baland bo'lsa, shunga mos ravishda ko'proq piksel biz uchun mavjud bo'ladi:

1 inch
1 inch
1 inch
72 dpi
150 dpi
300 dpi

2.10-rasm. Mobil qurilmalar uchun pikselIar tasvirlanishi

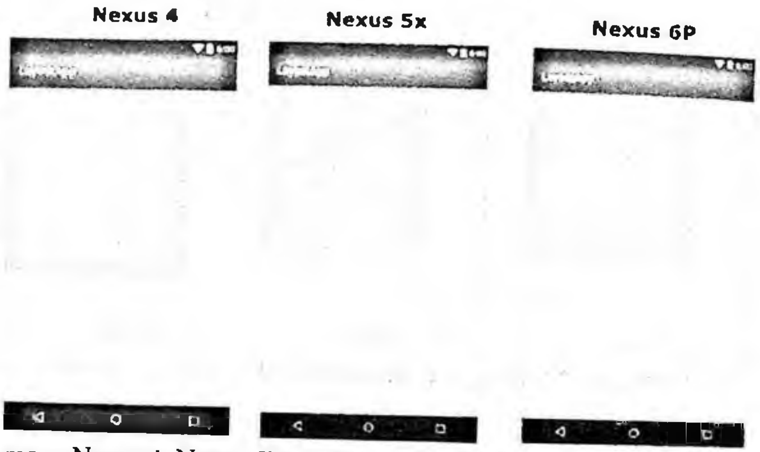
Standart fizik piksellardan foydalangan holda, biz elementning o'lchamlari, shuningdek, qurilmaning piksel zichligiga qarab juda katta farq qilishi mumkin degan muammoga duch kelamiz. Masalan, Nexus 4, Nexus 5X va Nexus 6P ekranining har xil xususiyatlariga ega bo'lgan 3 ta qurilmani olamiz va 300px x 300px kvadratni namoyish qilamiz:



2.11-rasm. Nexus 4, Nexus 5X va Nexus 6P ekranining har xil xususiyatlari

Bir holda, kenglikdagi kvadrat 40% ni, boshqasida - kenglikning uchdan bir qismini, uchinchisida - 20% ni egallaydi.

Endi biz tomonlari 300x300 bo'lgan kvadratni ham olamiz, ammo endi fizik piksellar o'rniga dp birliklaridan foydalanamiz:



2.12-rasm. Nexus 4, Nexus 5X va Nexus 6P ekranining har xil xususiyatlari o`zgarishi

Endi kvadratning kattaligi barcha qurilmalarda yanada uyg'unroq ko'rinadi.

Ishni o'lchamlari bilan soddalashtirish uchun barcha o'lchamlar bir necha guruhga bo'linadi:

- ldpi (low): ~120dpi
- mdpi (medium): ~160dpi
- hdpi (high): ~240dpi (ushbu guruh Nexus One kabi eski qurilmani o'z ichiga oladi)
- xhdpi (extra-high): ~320dpi (Nexus 4)
- xxhdpi (extra-extra-high): ~480dpi (Nexus 5/5X, Samsung Galaxy S5)
- xxxhdpi (extra-extra-extra-high): ~640dpi (Nexus 6/6P, Samsung Galaxy S6)

***O'lchamlarni o'rnatish.*** O'lchash bilan bog'liq asosiy muammo ularni Java kodida o'rnatish bilan bog'liq. Masalan, ba'zi usullar device-independent pixels o'rniga fizik piksellarni oladi. Bunday holda siz qiymatlarni bir birlik turidan boshqasiga o'zgartirishingiz kerak bo'lishi mumkin. Buning uchun TypedValue.applyDimension() usuli talab qilinadi, bu uchta parametrni oladi:

*public static float applyDimension(int unit,*
*float value,*
*android.util.DisplayMetrics metrics)*

Unit parametri piksel qiymatini oladigan birlik turini aks ettiradi. Birlik turi TypedValue doimiylaridan biri tomonidan tavsiflanadi:

- COMPLEX_UNIT_DIP - dp yoki ekran zichligi mustaqil piksellar
- COMPLEX_UNIT_IN - dyuym yoki dyuym
- COMPLEX_UNIT_MM - mm yoki millimetr
- COMPLEX_UNIT_PT - pt yoki ball
- COMPLEX_UNIT_PX - piksel yoki fizik piksel
- COMPLEX_UNIT_SP - sp yoki o'lchovdan mustaqil piksellar (scale-independent pixels)

Value parametri konvertatsiya qilinadigan qiymatni bildiradi

Metrik parametrlari konversiyani amalga oshirishi kerak bo'lgan o'lchov haqida ma'lumot beradi.

Natijada, usul konvertatsiya qilingan qiymatni qaytaradi. Mavhum misolni ko'rib chiqaylik. Masalan, biz 60dp dan oddiy fizik piksellarni olishimiz kerak:

```
int valueInDp = 60;
int valueInPx = (int) TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP, valueInDp,
getResources().getDisplayMetrics());
```

Uchinchi argument - bu getResources (). GetDisplayMetrics () usuliga qo'ng'iroq, bu sizga joriy moslama bilan bog'liq o'lchov haqida ma'lumot olish imkonini beradi. Natijada biz 60dp dan piksellar sonini olamiz.

## 2.4.Elementlarning kengligi va balandligi.

Ilovada foydalanadigan barcha vizual elementlar odatda konteynerlar yordamida ekranda joylashtiriladi. Android-da bunday konteynerlar RelativeLayout, **LinearLayout, GridLayout, TableLayout, ConstraintLayout, FrameLayout** kabi sinflardir. Ularning barchasi elementlarni boshqacha tarzda tartibga soladi va boshqaradi, ammo vizual komponentlarning joylashuvida ba'zi umumiy fikrlar mavjud, biz hozir ko'rib chiqamiz.

Joylashtirish imkoniyatlari konteyner ichidagi elementlarni tartibga solish uchun ishlatiladi. Ularni xml fayliga o'rnatish uchun **layout_** prefiksidan boshlanadigan atributlardan foydalaniladi. Xususan, ushbu parametrlar o'lchamlarni o'rnatish uchun ishlatiladigan **layout_height** va **layout_width** atributlarini o'z ichiga oladi va quyidagi variantlardan birini ishlatishi mumkin:

- **match_parent** qiymati (**ConstraintLayout** tashqari barcha konteynerlar uchun) yoki **0dp** (**ConstraintLayout** uchun) yordamida idishni to'liq kengligi yoki balandligiga cho'zish.
- **wrap_content** qiymatidan foydalanib tarkibidagi barcha tarkibni o'z ichiga oladigan darajada mos keladigan elementni cho'zish
- Elementning aniq o'lchamlari, masalan 96 dp.

**match_parent**. Qiymatni **match_parent**-ga o'rnatish elementning idishning butun kengligi yoki balandligi bo'ylab cho'zilishiga imkon beradi. Shuni ta'kidlash kerakki, bu qiymat **ConstraintLayout**dan tashqari barcha konteynerlarga tegishli. Masalan, **TextView**ni **LinearLayout** konteynerining to'liq kengligi va balandligiga tortamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="Hello World!"
```

```
        android:textSize="30sp"
        android:background="#e0e0e0" />

</LinearLayout>
```

Bu holda **LinearLayout** bo'lgan yuqori darajadagi konteyner balandligi va kengligi uchun **match_parent** qiymatiga ega, ya'ni **activity** uchun butun maydonni to'ldiradi - odatda butun ekran.

**TextView** shunga o'xshash atributlarni ham qabul qiladi. **Android: layout_width = "match_parent"** kenglik bo'ylab, **android: layout_height = "match_parent"** vertikal ravishda cho'ziladi. Aniqlik uchun **TextView** fonni ifodalovchi android: background atributidan foydalanadi va bu holda elementni **"#e0e0e0"** rang bilan ranglaydi, shunda biz uning egallagan maydonini ko'rishimiz mumkin.

Hello World!

2.13-rasm. Dastur natijasi

**match_parent** qiymati **LinearLayout** yoki **RelativeLayout** va ularning elementlari kabi deyarli barcha o'rnatilgan konteynerlarda ishlatilishi mumkinligini unutmang. Biroq, **match_parent-ni** **ConstraintLayout** ichidagi elementlarga qo'llash tavsiya etilmaydi. **ConstraintLayoutdagi "match_parent"** o'rniga, elementni gorizontal yoki vertikal ravishda cho'zish uchun **0dp** dan foydalanishingiz mumkin:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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">

   <TextView
      android:layout_width="0dp"
      android:layout_height="0dp"
      android:text="Hello World!"
      android:textSize="30sp"
      android:background="#e0e0e0"
```

```
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Shuni ta'kidlash kerakki, **ConstraintLayout**ning o'zi ham **layout_width** va **android: layout_height** atributlarida **"match_parent"** qiymatidan foydalangan holda ekranning kengligi va balandligiga cho'zilgan, ammo bu qiymat ichki elementlar uchun tavsiya etilmaydi.

**ConstraintLayout** o'lchamlarini o'rnatishda ba'zi o'ziga xos xususiyatlarga ega bo'lganligi sababli, **ConstraintLayout**dagi elementlarning o'lchamlari bilan ishlash quyidagi mavzulardan birida batafsilroq tavsiflanadi.

**wrap_content. wrap_content** qiymati ekrandagi element tarkibiga mos keladigan kenglik yoki balandlik zarurligini belgilaydi:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="30sp"
    android:background="#ffcdd2"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
  />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu erda **TextView** o'z matniga mos keladigan darajada cho'zilgan.

Hello World!

2.14-rasm. Dastur natijasi

**Aniq qiymatlarni o'rnatish.** Bundan tashqari, aniq qiymatlarni o'rnatishimiz mumkin:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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">

    <TextView
       android:layout_height="90dp"
       android:layout_width="150dp"
       android:text="Hello World!"
       android:textSize="30sp"
       android:background="#e0e0e0"
       app:layout_constraintLeft_toLeftOf="parent"
       app:layout_constraintTop_toTopOf="parent"
       />

  </androidx.constraintlayout.widget.ConstraintLayout>
```

Hello
World!

2.15-rasm. Dastur natijasi

Shu bilan bir qatorda, siz tarkibning kengligiga mos ravishda cho'zish va balandlik uchun aniq qiymatlarni belgilash kabi bir nechta qiymatlarni birlashtira olasiz:

```
<TextView
    android:layout_height="80dp"
    android:layout_width="wrap_content"
    android:text="Hello World!"
    android:textSize="30sp"
    android:background="#e0e0e0"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
/>
```

Agar wrap_content qiymati kenglik va uzunlikni o'rnatish uchun ishlatilsa, biz **minWidth/maxWidth** va **minHeight/maxHeight** atributlari yordamida qo'shimcha va minimal qiymatlarni cheklashimiz mumkin:

```
    <TextView
        android:minWidth="200dp"
        android:maxWidth="250dp"
        android:minHeight="100dp"
        android:maxHeight="200dp"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Hello World!"
        android:textSize="30sp"
        android:background="#e0e0e0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
    />
```

Bunday holda, **TextView** kengligi matnni o'z ichiga olishi uchun etarli bo'ladi, lekin **maxWidth** qiymatidan va **minWidth** qiymatidan kam bo'lmasligi kerak. Xuddi shu narsa balandlikni belgilash uchun ham amal qiladi.

**Dasturlashtiriladigan kenglik va balandlikni sozlash.** Agar element, masalan, xuddi shu **TextView** java kodida yaratilgan bo'lsa, unda balandlik va kenglikni o'rnatish uchun **setLayoutParams()** usulidan foydalanish mumkin. Shunday qilib, **MainActivity** kodini o'zgartiraylik:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ConstraintLayout constraintLayout = new ConstraintLayout(this);
        TextView textView = new TextView(this);
        textView.setText("Hello Android");
        textView.setTextSize(26);

 // elementning o'lchamlari va holatini o'rnating
      ConstraintLayout.LayoutParams          layoutParams          =          new
 ConstraintLayout.LayoutParams
            (ConstraintLayout.LayoutParams.WRAP_CONTENT,
 ConstraintLayout.LayoutParams.WRAP_CONTENT);
        // эквивалент app:layout_constraintLeft_toLeftOf="parent"
        layoutParams.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;
        // эквивалент app:layout_constraintTop_toTopOf="parent"
        layoutParams.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;
       // textView uchun parametrlarni o'rnating
        textView.setLayoutParams(layoutParams);
        // TextView-ni ConstraintLayout-ga qo'shing
        constraintLayout.addView(textView);
        setContentView(constraintLayout);
    }
}
```

**ViewGroup.LayoutParams** ob'ekti **setLayoutParams()** uslubiga o'tkaziladi. Ushbu ob'ekt ikkita parametr bilan boshlangan: kenglik va balandlik.

Kenglik va balandlikni aniqlash uchun **ViewGroup.LayoutParams.WRAP_CONTENT** yoki **ViewGroup.LayoutParams.MATCH_PARENT** doimiylaridan birini ishlatishingiz mumkin (**LinearLayout** yoki **RelativeLayout** holatlarida).

Hello Android

2.16-rasm. Dastur natijasi

Shuningdek, biz aniq qiymatlarni berishimiz yoki qiymat turlarini birlashtirishimiz mumkin:

```
ConstraintLayout.LayoutParams layoutParams = new
ConstraintLayout.LayoutParams
        (ConstraintLayout.LayoutParams.WRAP_CONTENT, 200);
```

## 2.5. Ichki va tashqi fiksirlash.

Belgilash parametrlari elementning tashqi chegaralaridan konteyner chegaralariga qadar ham, elementning o'zi ichida ham uning chegaralari va tarkibi o'rtasida indentlarni o'rnatishga imkon beradi.

**Padding**. To'ldirishni o'rnatish uchun **android:padding** atributidan foydalaniladi. U konteynerning to'rt tomonidan to'ldirishni o'rnatadi. Siz quyidagi atributlarni qo'llash orqali konteynerni faqat bir tomonidan to'ldirishingiz mumkin: **android: paddingLeft**, **android: paddingRight**, **android: paddingTop** va **android: paddingBottom**.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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"
  android:padding="50dp"
  tools:context=".MainActivity">
```

```
<TextView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="Hello World!"
    android:textSize="30sp"
    android:background="#e0e0e0"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

**ConstraintLayout** konteynerida faqat bitta 50 ta birlik mavjud. Ichki **TextView** konteynerning yuqori chap burchagida joylashgan (**app:layout_constraintLeft_toLeftOf = "parent"** va **app: layout_constraintTop_toTopOf = "parent"** atributlari tufayli). ... Shuning uchun **TextView** boshlang'ich nuqtadan (**ConstraintLayout** konteynerining yuqori chap burchagi) pastga va chapga 50 birlikka o'tadi. Bunga qo'shimcha ravishda, agar element konteynerning pastki yoki o'ng chegarasiga ulashgan bo'lsa, xuddi shu plomba o'ng va pastki qismlarga tegishlidir.

50
50
Hello World!

2.17-rasm. Ichki va tashqi fiksirlash

Bir chuqurchaga o'rnatish.

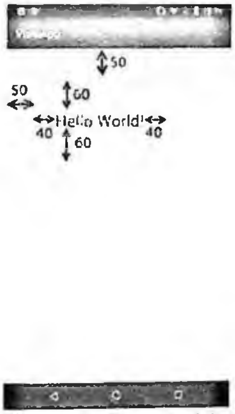
*android:padding="50dp"*

To'rt qatorni belgilash bilan bir xil bo'ladi.

```
android:paddingTop="50dp"
android:paddingLeft="50dp"
android:paddingBottom="50dp"
android:paddingRight="50dp"
```

Xuddi shunday, siz boshqa elementlarda ham indentlarni o'rnatishingiz mumkin. Masalan, **TextView** 60 ta ichki tarkibning (ya'ni matnning) yuqori va pastki qismini va 40 birlikning chap va o'ng qirralarini o'rnatamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    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"
    android:padding="50dp"
    tools:context=".MainActivity">
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:paddingTop="60dp"
        android:paddingLeft="40dp"
        android:paddingRight="40dp"
        android:paddingBottom="60dp"
        android:text="Hello World!"
        android:textSize="30sp"
        android:background="#e0e0e0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```



2.18-rasm. android:paddingLeft va android:paddingRight atributlari

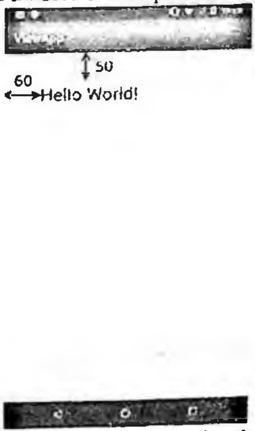
Shuni ta'kidlash kerakki, **android:paddingLeft** va **android:paddingRight** atributlari o'rniga dasturni chap tomonga yo'naltirilgan tillar uchun ishlashga moslashtirish uchun maxsus ishlab chiqilgan **android:paddingStart** va **android:paddingEnd** atributlaridan foydalanishingiz mumkin. va o'ng tomonga yo'naltirish (arabcha, forscha).

**Margin. Layout_margin** atributi chekkalarni o'rnatish uchun ishlatiladi. Ushbu atributda faqat bitta tomondan indent o'rnatishga imkon beruvchi modifikatsiyalar mavjud: **android:layout_marginBottom, android:layout_marginTop,** **android:layout_marginLeft** va

**android:layout_marginRight** (navbati bilan pastki, yuqori, chap va o'ng chegaralardagi kirishlar):

```
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">
  <TextView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_marginTop="50dp"
    android:layout_marginLeft="60dp"
    android:text="Hello World!"
    android:textSize="30sp"
    android:background="#e0e0e0"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
  />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu yerda TextView ConstraintLayout-ning ikki tomonidan (60 birlik chapga va yuqori qismdan 50 birlik) kiruvchi chiziqlarni o'rnatadi:



2.19-rasm. Ilovaga komponentaning joylashuv konstuktorligi

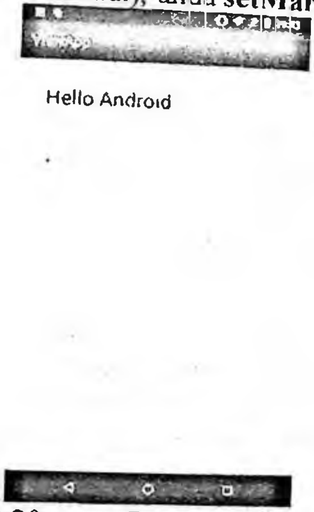
Dasturiy jihatdan indenting

Elementlarning to'ldirilishini dasturiy ravishda o'rnatish uchun setPadding (left, top, right, bottom) usuli chaqirilib, unga har bir tomon uchun to'rtta qiymatlar kiritiladi. Bundan tashqari getPaddingLeft(), getPaddingTop(), getPaddingRight() va getPaddingBottom() usullaridan foydalanib to'ldirishni alohida o'rnatishingiz mumkin.

Chegaralarni o'rnatish uchun foydalanilayotgan konteyner uchun LayoutParams ob'ektini amalga oshirishingiz kerak. Va keyin ushbu LayoutParams ob'ektida setMargins (left, top, right, bottom) usulini chaqiring:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      ConstraintLayout constraintLayout = new ConstraintLayout(this);
      TextView textView = new TextView(this);
      // matn maydonining rangini sozlash
      textView.setBackgroundColor(0xFFE0E0E0);
      // matn maydonining matnini sozlash
      textView.setText("Hello Android");
      // matn hajmini belgilash.
      textView.setTextSize(30);
      ConstraintLayout.LayoutParams layoutParams = new
ConstraintLayout.LayoutParams
            (ConstraintLayout.LayoutParams.WRAP_CONTENT ,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
      // tashqi to'ldirishni sozlash
      layoutParams.setMargins(60, 50, 60, 50);
      // konteynerni yuqori chap burchagida joylashtirish
      // app:layout_constraintLeft_toLeftOf="parent" ekvivalent
      layoutParams.leftToLeft =
ConstraintLayout.LayoutParams.PARENT_ID;
      // app:layout_constraintTop_toTopOf="parent" ekvivalent
      layoutParams.topToTop =
ConstraintLayout.LayoutParams.PARENT_ID;
      // o'lchamlarini o'rnating.
      textView.setLayoutParams(layoutParams);
      // to'ldirishni o'rnatish
      textView.setPadding(40,40,40,40);
      // TextView-ni ConstraintLayout-ga qo'shish.
      constraintLayout.addView(textView);
      setContentView(constraintLayout);
   }
}
```

Bu holda **TextView** elementi **ConstraintLayout** tipidagi konteynerga qo'shilganligi sababli, uni joylashtirish uchun **ConstraintLayout.LayoutParams** ob'ekti ishlatiladi (mos ravishda **LinearLayout** uchun **LinearLayout.LayoutParams** bo'ladi), unda **setMargins()** usuli chaqiriladi.


Hello Android

2.20-rasm. Dastur natijasi

Ammo oxirgi skrinshotga qarasangiz, shuni ko'rishingiz mumkinki, chekka joylar fayllar jadvalidagi oldingi misoldagi kabi o'rnatilgandek, aslida, ekranda, biz chekkalarni ko'rayapmiz butunlay boshqacha qadriyatlar. Haqiqat shundaki, **setPadding()** va **setMargins()** usullari piksellarda qiymatlarni qabul qiladi, dp birliklari esa Layout faylida ishlatilgan. Va kodingizda dp dan foydalanish uchun siz transformatsiyalarni bajarishingiz kerak:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.util.TypedValue;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ConstraintLayout constraintLayout = new ConstraintLayout(this);
    TextView textView = new TextView(this);
    textView.setBackgroundColor(0xFFE0E0E0);
    textView.setText("Hello Android!");
    textView.setTextSize(30);
    // biz 50 dp uchun piksel bilan ofset olamiz.
    int margin50inDp = (int) TypedValue.applyDimension(          50.
         TypedValue.COMPLEX_UNIT_DIP,
getResources().getDisplayMetrics());
    // biz 60 dp uchun piksel bilan ofset olamiz
    int margin60inDp = (int) TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,                                60,
getResources().getDisplayMetrics());
            // biz 40 dp uchun piksel bilan ofset olamiz
            int padding40inDp = (int) TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,                                40,
getResources().getDisplayMetrics());
            ConstraintLayout.LayoutParams      layoutParams       =       new
ConstraintLayout.LayoutParams
                (ConstraintLayout.LayoutParams.WRAP_CONTENT                   ,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
            // chekkalarni o'rnatish
            layoutParams.setMargins(margin60inDp,                margin50inDp,
margin60inDp, margin50inDp);
            // ConstraintLayout-ning chap yo'nalishiga tekislash.
            layoutParams.leftToLeft                                          =
ConstraintLayout.LayoutParams.PARENT_ID;
            // ConstraintLayout-ning yuqori chegarasiga moslashtirish.
            layoutParams.topToTop                                            =
ConstraintLayout.LayoutParams.PARENT_ID;
            // o`lchamlarni o`rnatish
            textView.setLayoutParams(layoutParams);
            // to'ldirishni o'rnatish
            textView.setPadding(padding40inDp,                padding40inDp,
padding40inDp, padding40inDp);
            // TextView-ni ConstraintLayout-ga qo'shing
            constraintLayout.addView(textView);
            setContentView(constraintLayout);
        }
    }
```

```
Hello Android
```

2.21-rasm. Dastur natijasi

## 2.6. Cheklovlar(ConstraintLayout).

**ConstraintLayout** sizga moslashuvchan va miqyoslanadigan vizual interfeyslarni yaratishga imkon beradigan konteynerni ifodalaydi.

Elementni **ConstraintLayout** ichida joylashtirish uchun cheklovlarni **(constraints)** belgilashingiz kerak. Cheklovlarning bir nechta turlari mavjud. Xususan, pozitsiyani ma'lum bir elementga nisbatan belgilash uchun quyidagi cheklovlar qo'llaniladi:

- layout_constraintLeft_toLeftOf: Chap chegara boshqa elementning chap chegarasiga nisbatan joylashtirilgan
- layout_constraintLeft_toRightOf: chap chegara boshqa elementning o'ng chegarasiga nisbatan joylashtirilgan
- layout_constraintRight_toLeftOf: o'ng chegara boshqa elementning chap chegarasiga nisbatan joylashgan
- layout_constraintRight_toRightOf: o'ng chegara boshqa elementning o'ng chegarasiga nisbatan joylashtirilgan
- layout_constraintTop_toTopOf: yuqori chegara boshqa elementning yuqori chegarasiga nisbatan joylashtirilgan
- layout_constraintTop_toBottomOf: yuqori chegara boshqa elementning pastki chegarasiga nisbatan joylashtirilgan
- layout_constraintBottom_toBottomOf: pastki chegara boshqa elementning pastki chegarasiga nisbatan joylashtirilgan
- layout_constraintBottom_toTopOf: pastki chegara boshqa elementning yuqori chegarasiga nisbatan joylashtirilgan
- layout_constraintBaseline_toBaselineOf: asosiy element boshqa elementning boshlang'ich chizig'iga nisbatan joylashtirilgan
- layout_constraintStart_toEndOf: element boshqa element tugagan joyda boshlanadi
- layout_constraintStart_toStartOf: element boshqa element boshlanadigan joyda boshlanadi
- layout_constraintEnd_toStartOf: element boshqa element boshlanadigan joyda tugaydi
- layout_constraintEnd_toEndOf: element boshqa element tugaydigan joyda tugaydi

Ehtimol, so'nggi to'rtta xususiyat haqida, elementning boshi yoki oxiri nimani anglatishini tushunmaslik bor edi. Haqiqat shundaki, ba'zi tillar (masalan, arab yoki fors tillari) o'ng tomonga yo'naltirilgan, ya'ni belgilar Evropa tillarida bo'lgani kabi chapdan o'ngga emas, chapdan o'ngga qarab harakatlanadi. Va hozirgi yo'nalishga qarab - chap yoki o'ng qo'l - elementning boshi va oxiri qaerda joylashganligi aniq o'zgaradi. Masalan, chap tomonga yo'naltirilgan holda boshlanish chap tomonda, oxir esa o'ng tomonda bo'ladi, shuning uchun layout_constraintStart_toEndOf atributi aslida layout_constraintLeft_toRightOf atributi bilan bir xil bo'ladi. Va o'ng tomonga yo'naltirish uchun - layout_constraintRight_toLeftOf atributi, chunki boshi o'ngda, oxiri chapda.

Har bir cheklash elementning joylashishini gorizontal yoki vertikal ravishda belgilaydi. Va elementning ConstraintLayout-dagi o'rnini aniqlash uchun siz kamida bitta gorizontal va bitta vertikal cheklovni belgilashingiz kerak.

Joylashuv ContentLayout konteynerining o'zi chegaralariga nisbatan (bu holda, cheklash **parent** qiymatiga ega) yoki ConstraintLayout ichidagi boshqa har qanday elementga nisbatan bo'lishi mumkin, keyin ushbu elementning id cheklov qiymati sifatida ko'rsatiladi.

Eng oddiy misol:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!"
      android:textSize="30sp"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bunday holda, TextView elementi ikkita cheklovga ega: biri gorizontal chiziq uchun (app: layout_constraintLeft_toLeftOf = "parent"), ikkinchisi vertikal chiziq uchun (app: layout_constraintTop_toTopOf = "parent"). Ikkala cheklovlar ham ConstraintLayout konteyneriga nisbatan o'rnatiladi, shuning uchun ular parent, ya'ni ConstraintLayout-ga o'rnatiladi.

app:layout_constraintLeft_toLeftOf="parent" cheklovi TextView-ning chap chegarasini konteynerning chap chegarasiga o'rnatadi.

app:layout_constraintTop_toTopOf = "parent" cheklovi TextView-ning yuqori chegarasini konteynerning yuqori chegarasida o'rnatadi.

Natijada, **TextView** konteynerning yuqori chap burchagida joylashgan bo'ladi.

Hello World!

2.22-rasm. Dastur natijasi

Ushbu cheklov atributlarining barchasi app prefiksiga qo'shiladigan "http://schemas.android.com/apk/res-auto" nom maydonidan kelib chiqqanligini unutmang.

Agar siz boshqa elementga nisbatan cheklov o'rnatishingiz kerak bo'lsa, unda ushbu elementning id-ni belgilashingiz kerak:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">
  <EditText
    android:id="@+id/editText"
    android:layout_width="180dp"
    android:layout_height="wrap_content"
    android:hint="Введите Email"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
  <Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Отправить"
    app:layout_constraintLeft_toRightOf="@id/editText"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu yerda **EditText** bosh konteyner **ConstraintLayout**-ga nisbatan ikkita cheklovga o'rnatiladi, shuning uchun cheklovlar **parent**ga o'rnatiladi va **EditText** konteynerning chap va yuqori qismiga to'g'ri keladi. **Button** tugmachasining yuqori chegarasi, shuningdek, konteynerning yuqori chegarasiga to'g'ri keladi. Ammo tugmachaning chap chegarasi **EditText**ning o'ng chegarasi bilan tekislangan. Buning uchun **EditText**ning **id** atribut qiymati sifatida qabul qilinadi:

*app:layout_constraintLeft_toRightOf="@id/editText"*

2.21-rasm. Dastur natijasi

Xuddi shu tarzda, biz kerakli joylashishni aniqlash uchun siz atributlarning turli xil birikmalarini tuzishingiz mumkin. Masalan, tugma kodini o'zgartiraylik:

```
<Button
  android:id="@+id/button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Jo`natish"
  app:layout_constraintLeft_toRightOf="@id/editText"
  app:layout_constraintTop_toBottomOf="@id/editText" />
```

Bunday holda, tugmachaning yuqori chegarasi **EditText**ning pastki chegarasi bilan tekislanadi.

Bundan tashqari, biz ikkala elementni bir-biriga nisbatan joylashtira olamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">
  <EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Введите Email"
    app:layout_constraintRight_toLeftOf="@+id/button"
    app:layout_constraintTop_toTopOf="parent" />
  <Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Jo`natish"
    app:layout_constraintLeft_toRightOf="@id/editText"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**Markazni joylashtirish.** Agar elementni vertikal ravishda konteynerning o'rtasiga qo'yishingiz kerak bo'lsa, unda siz bir nechta atributlardan foydalanishingiz kerak.

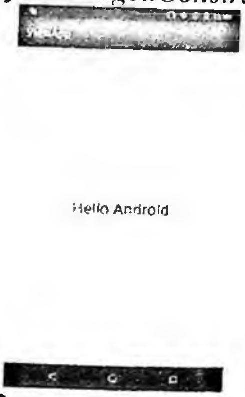
```
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
```

Agar elementni gorizontal ravishda konteynerning o'rtasiga qo'yishingiz kerak bo'lsa, unda siz quyidagi juft atributlardan foydalanishingiz kerak.

```
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```

Shunga ko'ra, konteyner markazida vertikal va gorizontal joylashish uchun yuqoridagi to'rtta atributning barchasi qo'llanilishi kerak:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello Android"
      android:textSize="30sp"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```


Hello Android

2.22-rasm. Dastur natijasi

Shift(Sdvig). Agar elementlar markazlashtirilgan bo'lsa, ConstraintLayout ularni gorizontal va vertikal ravishda almashtirishga imkon beradi. Gorizontal siljish uchun layout_constraintHorizontal_bias atributi va vertikal siljish uchun layout_constraintVertical_bias atributi qo'llaniladi. Ular suzuvchi nuqta raqamini 0 dan 1 gacha qabul qilishadi. Odatiy qiymati 0,5 (markazlashtirilgan). Masalan:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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">
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top TextView"
        android:textSize="30sp"
        android:background="#e0e0e0"
        app:layout_constraintHorizontal_bias="0.2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom TextView"
        android:textSize="30sp"
        android:background="#e0e0e0"
        app:layout_constraintHorizontal_bias=".9"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Birinchi TextView konteynerning chap chegarasidan 20% ga siljiydi (sukut bo'yicha 0,5, shuning uchun 0,2 qiymati elementni chapga siljitadi). Ikkinchi TextView konteynerning chap chegarasidan 90% ofset qilingan. Masalan, 1 qiymati element o'ng chegaraga, 0 qiymati chapga siljiganligini bildiradi.

2.23-rasm. Dastur natijasi

Layout_constraintVertical_bias atributi xuddi shunday ishlaydi, vertikal shakliga keltiradi.

## 2.7. ConstraintLayoutdagi elementlarning o'lchamlari.

ConstraintLayout-da o'lchamlarni uchta usuli mavjud:

Aniq o'lchamlarni o'rnatish, masalan 123dp;

Vidjetning tarkibiga mos keladigan hajmini belgilaydigan **wrap_content** qiymati;

Java kodidagi "**match_constraint**" qiymatiga teng bo'lgan 0dp qiymati. Bunday holda, elementning o'lchamlari uning uchun belgilangan cheklovlar asosida o'rnatiladi. Odatiy bo'lib, element barcha mavjud joylarni egallaydi.

Biz o'lchamlarning uch turini ham qo'llaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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">
   <TextView
      android:layout_width="160dp"
      android:layout_height="wrap_content"
      android:text="Top TextView"
      android:textSize="30sp"
      android:background="#e0e0e0"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent"/>
   <TextView
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:text="Center TextView"
      android:textSize="30sp"
      android:background="#e0e0e0"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintBottom_toBottomOf="parent"/>
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Bottom TextView"
      android:textSize="30sp"
      android:background="#e0e0e0"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu erda uchta **TextView** elementlari yaratilgan. Ularning barchasi gorizontal ravishda markazlashtirilgan, ammo vertikal ravishda konteynerning yuqori va pastki qismida va markazda joylashgan. Barcha uchta **TextViews**-ning balandlig

**wrap_content**ga o'rnatilgan, ya'ni uchta element ham tarkibga mos keladigan balandlikni egallaydi:

*android:layout_height="wrap_content"*

Biroq, har bir element o'z kengligi sozlamalariga ega. Yuqori **TextView**ning aniq hajmi 160 birlikka o'rnatildi:

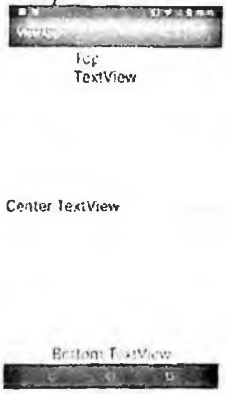
*android:layout_width="160dp"*

**TextView** markazi "0dp" ga o'rnatildi, shunda element sukut bo'yicha barcha bo'sh joyni egallaydi (bu holda u gorizontal ravishda cho'ziladi):

*android: layout_width = "0dp"*

**TextView** pastki qismi "**wrap_content**" ga o'rnatildi, ya'ni element tarkibini o'z ichiga olishi uchun kerakli kenglikni oladi:

*android:layout_width="wrap_content"*



2.24-rasm. Dastur natijasi

Shuni ta'kidlash kerakki, **ConstraintLayout**da ichki o'rnatilgan vidjetlarda **match_parent** qiymatidan foydalanish tavsiya etilmaydi, bu vidjetga barcha mavjud joylarni egallashga imkon beradi. Buning o'rniga 0dp yoki "**MATCH_CONSTRAINT**" dan foydalanish tavsiya etiladi - boshqa cheklovlar bilan birgalikda ular kerakli effekt beradi. Shunday qilib, konteynerning kengligiga cho'zish uchun quyidagi atributlar qo'llaniladi:

*android:layout_width="0dp"*
*app:layout_constraintLeft_toLeftOf="parent"*
*app:layout_constraintRight_toRightOf="parent"*

Va konteyner balandligi bo'ylab cho'zish uchun quyidagi atributlar qo'llaniladi:

*android:layout_height="0dp"*
*app:layout_constraintTop_toTopOf="parent"*
*app:layout_constraintBottom_toBottomOf="parent"*

Masalan, **TextView**ni konteynerning to'liq uzunligi va kengligiga cho'zish:

```
<TextView
  android:layout_width="0dp"
  android:layout_height="0dp"
  android:text="Hello Android"
  android:textSize="30sp"
```

```
android:background="#e0e0e0"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent"/>
```

Minimal va maksimal o'lchamlar.

Bir qator atributlar maksimal va minimal o'lchamlarni belgilaydi:

➢ **layout_constraintWidth_min** va **layout_constraintHeight_min**: mos ravishda minimal kenglik va balandlikni ifodalaydi
➢ **layout_constraintWidth_max** va **layout_constraintHeight_max**: mos ravishda maksimal kenglik va balandlikni ifodalaydi

Ular qiymat sifatida **dp** yoki **wrap** qiymatini (wrap_contentga o'xshash) oladi. Masalan:

```
<TextView
   android:layout_width="260dp"
   android:layout_height="wrap_content"
   android:text="Hello Android"
   android:textSize="30sp"
   android:background="#e0e0e0"
   app:layout_constraintHeight_max="200dp"
   app:layout_constraintWidth_max="200dp"
   app:layout_constraintHeight_min="wrap"
   app:layout_constraintWidth_min="wrap"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintRight_toRightOf="parent"
   app:layout_constraintTop_toTopOf="parent"
   app:layout_constraintBottom_toBottomOf="parent"/>
```

Garchi bu holda TextView kengligi 260dp ga o'rnatilgan bo'lsa-da, maksimal kengligi 200dp ga o'rnatilganligi sababli, haqiqiy kengligi 200dp dan oshmaydi.

**Foiz o'lchamlari**. **layout_constraintWidth_percent** atributi elementning kengligini mavjud gorizontal bo'shliqqa foiz sifatida belgilaydi. Xuddi shunday **layout_constraintHeight_percent** atributi mavjud bo'lgan vertikal bo'shliqning foizini belgilaydi.

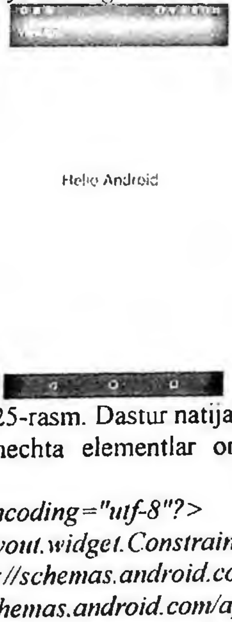
Ulardan foydalanish uchun quyidagi shartlar bajarilishi kerak:

➢ Hajmni o'rnatish uchun mos keladigan atribut (**android: layout_width** - agar biz kenglikni o'rnatgan bo'lsak yoki **android: layout_height** - agar biz balandlikni foiz sifatida belgilasak) **MATCH_CONSTRAINT** yoki **0dp** bo'lishi kerak
➢ Shuningdek, kenglikni o'rnatishda **app:layout_constraintWidth_default = "percent"** atributini va balandlikni o'rnatishda **app:layout_constraintHeight_default = "percent"** xususiyatini o'rnatishingiz kerak.

**layout_constraintWidth_percent** va **layout_constraintHeight_percent** atributlari qiymat sifatida **0** dan **1** gacha bo'lgan kasr sonini oladi.

Masalan, **TextView** vertikal ravishda 25% va gorizontal ravishda 50% egallaydi deylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">
  <TextView
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:text="Hello Android"
    android:textSize="30sp"
    android:background="#e0e0e0"
    app:layout_constraintWidth_default="percent"
    app:layout_constraintHeight_default="percent"
    app:layout_constraintWidth_percent="0.5"
    app:layout_constraintHeight_percent="0.25"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```



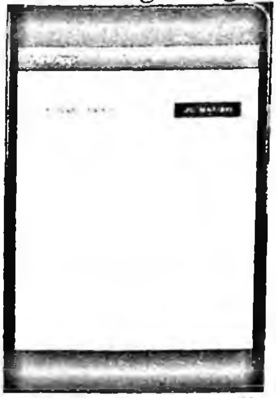
2.25-rasm. Dastur natijasi

Yana bir misol, bir nechta elementlar orasidagi bo'shliqni mutanosib ravishda ajratish:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">
    <EditText
      android:id="@+id/editText"
      android:hint="Введите Email"
      android:layout_height="wrap_content"
      android:layout_width="0dp"
      app:layout_constraintWidth_default="percent"
      app:layout_constraintWidth_percent="0.66"
      app:layout_constraintRight_toLeftOf="@+id/button"
      app:layout_constraintTop_toTopOf="parent" />
    <Button
      android:id="@+id/button"
      android:text="Отправить"
      android:layout_height="wrap_content"
      android:layout_width="0dp"
      app:layout_constraintWidth_default="percent"
      app:layout_constraintWidth_percent="0.33"
      app:layout_constraintLeft_toRightOf="@id/editText"
      app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bunday holda, **EditText** 66% keng va tugma 33% keng bo'ladi:



2.27-rasm. Dastur natijasi

**Balandlikning kenglikka nisbatini o'rnatish.**

ConstraintLayout shuningdek, elementlarning balandligini kenglik/kenglikka nisbatan balandlikni belgilashga imkon beradi. Buning uchun **layout_constraintDimensionRatio** atributidan foydalaniladi. Qiymat sifatida u **Width: Height** shaklidagi nisbatni oladi, masalan, 1: 0,5 - bu erda 1 raqami kenglikni, 0,5 esa balandlikni bildiradi. Ya'ni, kenglik balandlikdan ikki baravar ko'p bo'ladi. Ammo bir vaqtning o'zida 0dp (**MATCH_CONSTRAINT**) bitta o'lchov uchun o'rnatilishi kerak. Masalan:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:text="Hello Android"
    android:textSize="30sp"
    android:background="#e0e0e0"
    app:layout_constraintDimensionRatio="1:0.6"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bunday holda, **TextViewning** kengligi uning tarkibi uchun zarur bo'lgan har qanday bo'ladi va balandligi kenglikning 60% ni tashkil qiladi.

2.28-rasm. Dastur natijasi

Agar ikkala kenglik va balandlik 0dp ga o'rnatilgan bo'lsa, unda bu holda tizim barcha cheklovlarga javob beradigan eng katta o'lchamni tanlaydi va unga nisbatan boshqa o'lchamning qiymatini belgilaydi. Hisoblash amalga oshiriladigan o'lchovni belgilash uchun siz W (kenglik) yoki H (balandlik) belgisini belgilashingiz mumkin. Masalan:

```
<TextView
  android:layout_width="0dp"
  android:layout_height="0dp"
```

```
    android:text="Hello Android"
    android:textSize="30sp"
    android:background="#e0e0e0"
    app:layout_constraintDimensionRatio="W, 1:4"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"/>
```

Bunday holda, kenglik balandlikdan 4 baravar kam bo'ladi.

## 2.8. ConstraintLayoutdagi elementlarning zanjirlari.

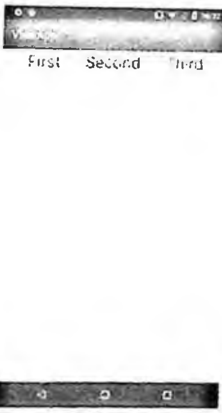
**ConstraintLayout** sizga elementlarning ketma-ket joylashishini gorizontal yoki vertikal ravishda yoki **Android** zanjirlar yoki zanjirlar deb ataydigan narsalarni tartibga solishga imkon beradi. Biz bir elementning boshqasiga nisbatan joylashishini zanjirga bog'lashimiz va shu bilan bir qator elementlarni tashkil qilishimiz mumkin.

**Elementlarning gorizontal zanjiri.** Masalan, elementlar qatori gorizontal ravishda:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">
  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#efefef"
    android:text="First"
    android:textSize="30sp"
    app:layout_constraintRight_toLeftOf="@id/textView2"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
  <TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#e0e0e0"
    android:text="Second"
    android:textSize="30sp"
    app:layout_constraintLeft_toRightOf="@id/textView1"
    app:layout_constraintRight_toLeftOf="@id/textView3"
```

```
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#efefef"
        android:text="Third"
        android:textSize="30sp"
        app:layout_constraintLeft_toRightOf="@id/textView2"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Natijada, zanjir elementlari idishning butun kengligi bo'ylab teng ravishda cho'zilib ketadi:



2.29-rasm. Dastur natijasi

Elementlarning gorizontal zanjiri ikki omil orqali amalga oshiriladi:

- Birinchi element konteynerning chap chegarasiga nisbatan (**app: layout_constraintLeft_toLeftOf = "parent"**), oxirgi element konteynerning o'ng chegarasiga nisbatan joylashtiriladi (**app:layout_constraintRight_toRightOf = "parent"**).
- **app:layout_constraintLeft_toRightOf** va **app:layout_constraintRight_toLeftOf** atributlarini o'rnatib, biz bitta elementni ikkinchisining o'ngiga yoki chapiga joylashtiramiz.

Bundan tashqari, **ConstraintLayout** sizga zanjir ichidagi elementlarning holatini sozlash imkonini beradi. Buning uchun quyidagi qiymatlarni qabul qilishi mumkin bo'lgan **layout_constraintHorizontal_chainStyle** atributidan foydalaniladi:

**spread**: Yuqoridagi misolda bo'lgani kabi zanjir elementlarini zanjirning butun uzunligi bo'ylab teng ravishda uzatadigan standart qiymat.

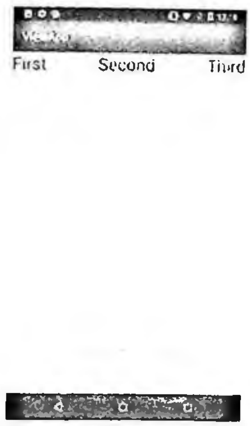
**spread_inside**: zanjirning birinchi va oxirgi elementlari konteyner chegaralariga qo'shni

**packed**: zanjir elementlari bir-biriga yaqin joylashtirilgan.

Masalan, **spread_inside** qiymatini qo'llaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    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">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#efefef"
        android:text="First"
        android:textSize="30sp"
        app:layout_constraintHorizontal_chainStyle="spread_inside"
        app:layout_constraintRight_toLeftOf="@id/textView2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#e0e0e0"
        android:text="Second"
        android:textSize="30sp"
        app:layout_constraintLeft_toRightOf="@id/textView1"
        app:layout_constraintRight_toLeftOf="@id/textView3"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#efefef"
        android:text="Third"
        android:textSize="30sp"
        app:layout_constraintLeft_toRightOf="@id/textView2"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bundan tashqari, bu holda zanjirning birinchi elementining atributini o'rnatish kifoya:



2.30-rasm. Dastur natijasi

**packed** qiymati:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    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">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#efefef"
        android:text="First"
        android:textSize="30sp"
        app:layout_constraintHorizontal_chainStyle="packed"
        app:layout_constraintRight_toLeftOf="@id/textView2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#e0e0e0"
        android:text="Second"
        android:textSize="30sp"
        app:layout_constraintLeft_toRightOf="@id/textView1"
        app:layout_constraintRight_toLeftOf="@id/textView3"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
       android:id="@+id/textView3"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:background="#efefef"
       android:text="Third"
       android:textSize="30sp"
       app:layout_constraintLeft_toRightOf="@id/textView2"
       app:layout_constraintRight_toRightOf="parent"
       app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

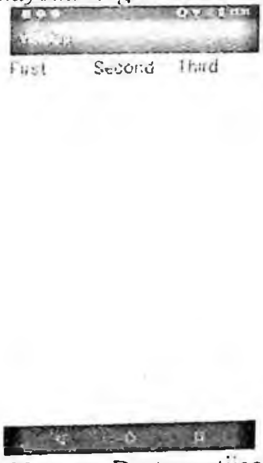
FirstSecondThird

2.31-rasm. Dastur natijasi

**Element og'irligi**. Shunisi e'tiborga loyiqki, yuqorida elementlar ularning tarkibi uchun zarur bo'lgan kenglikka o'rnatildi. Ammo biz kenglikni nolga o'rnatgan bo'lsak ham, ular orasidagi bo'shliqlarni yaratmasdan elementlar zanjir bo'ylab teng ravishda taqsimlanadi.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    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">
    <TextView
       android:id="@+id/textView1"
       android:layout_width="0dp"
       android:layout_height="wrap_content"
       android:background="#efefef"
       android:text="First"
```

```
        android:textSize="30sp"
        app:layout_constraintRight_toLeftOf="@id/textView2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="#e0e0e0"
        android:text="Second"
        android:textSize="30sp"
        app:layout_constraintLeft_toRightOf="@id/textView1"
        app:layout_constraintRight_toLeftOf="@id/textView3"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="#efefef"
        android:text="Third"
        android:textSize="30sp"
        app:layout_constraintLeft_toRightOf="@id/textView2"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



```
First        Second     Third
```

2.32-rasm. Dastur natijasi

Bunday holda, **app:layout_constraintHorizontal_chainStyle** atributining qiymati ahamiyatga ega emas, chunki barcha elementlar butun zanjir bo'ylab cho'zilgan.

Biroq, bu xatti-harakatlar sizga mos kelmasligi mumkin, masalan, biz bi element boshqasidan ikki baravar katta bo'lishini xohlaymiz. Va bu holda bi **layout_constraintHorizontal_weight** atributidan foydalanishimiz mumki Shunga qaramay, elementlar uchun og'irliklarni ishlatganda ular nol kengligi bo'lisl kerakligini yodda tutish kerak:

```
        <?xml version="1.0" encoding="utf-8"?>
        <androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
            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">
            <TextView
                android:id="@+id/textView1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:background="#efefef"
                android:text="First"
                android:textSize="30sp"
                app:layout_constraintHorizontal_weight="1"
                app:layout_constraintRight_toLeftOf="@id/textView2"
                app:layout_constraintLeft_toLeftOf="parent"
                app:layout_constraintTop_toTopOf="parent" />
            <TextView
                android:id="@+id/textView2"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:background="#e0e0e0"
                android:text="Second"
                android:textSize="30sp"
                app:layout_constraintHorizontal_weight="2"
                app:layout_constraintLeft_toRightOf="@id/textView1"
                app:layout_constraintRight_toLeftOf="@id/textView3"
                app:layout_constraintTop_toTopOf="parent" />
            <TextView
                android:id="@+id/textView3"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:background="#efefef"
                android:text="Third"
                android:textSize="30sp"
                app:layout_constraintHorizontal_weight="1"
                app:layout_constraintLeft_toRightOf="@id/textView2"
                app:layout_constraintRight_toRightOf="parent"
```

*app:layout_constraintTop_toTopOf="parent" />*
*</androidx.constraintlayout.widget.ConstraintLayout>*

**layout_constraintHorizontal_weight** atributi qiymat sifatida raqamni oladi - elementning vazni. Shunday qilib, bu holda birinchi elementning og'irligi 1 ga, ikkinchisining og'irligiga 2 ga, uchinchisining og'irligiga 1 ga teng bo'ladi, shuning uchun idishning butun kengligi shartli ravishda 1+2+1 = 4 qismga bo'linadi, shundan birinchi va uchinchi elementlar bitta qismni, ikkinchisi esa 2 qismni oladi, ya'ni ikkinchi element birinchi va uchinchi elementlardan ikki baravar katta bo'ladi.

First
Second
Third

2.33-rasm. Dastur natijasi

Aslida biz "wrap_content" kengligi yoki "0dp" dan tashqari ma'lum bir qiymatga ega elementlarni qoldirishimiz mumkin, faqat bu holda ular konteyner maydonini taqsimlashda qatnashmaydi va bunday elementning og'irligi rol.

**Vertikal zanjir.** Vertikal zanjir hosil qilish uchun ikkita shart bajarilishi kerak:

- Birinchi element konteynerning yuqori qismiga (**app:layout_constraintTop_toTopOf = "parent"**), oxirgi element konteynerning pastki qismiga (**app:layout_constraintBottom_toBottomOf = "parent"**) nisbatan hizalanadi.
- **app:layout_constraintBottom_toTopOf** va **app:layout_constraintBottom_toTopOf** atributlarini o'rnatib, biz bir elementni ikkinchisining ustiga qo'yamiz.

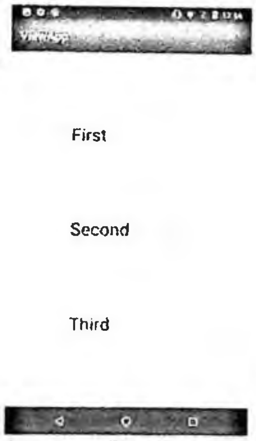
Zanjir ichidagi elementlarning holatini sozlash uchun quyidagi qiymatlarni olishi mumkin bo'lgan **layout_constraintVertical_chainStyle** atributi qo'llaniladi:

- **spread:** zanjir elementlarini zanjirning butun uzunligi bo'ylab teng ravishda uzatadigan standart qiymat
- **spread_inside:** zanjirning birinchi va oxirgi elementlari konteyner chegaralariga qo'shni
- **packed:** zanjir elementlari bir-biriga qo'shni.

Masalan, standart qiymati spr_ead bo'lgan vertikal zanjir:

*<?xml version="1.0" encoding="utf-8"?>*
*<androidx.constraintlayout.widget.ConstraintLayout*
*xmlns:android="http://schemas.android.com/apk/res/android"*

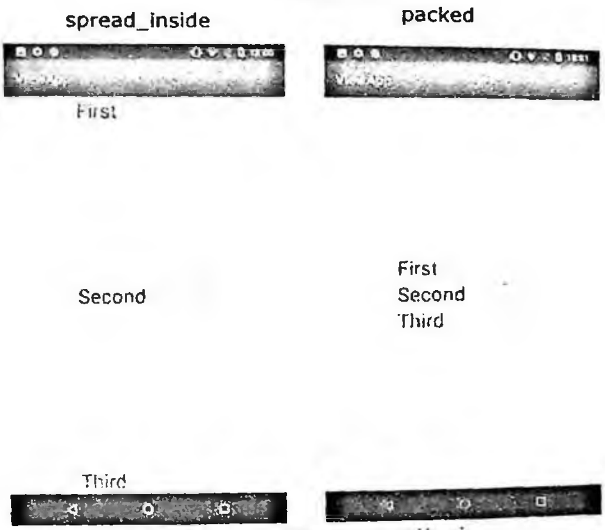
```
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<TextView
   android:id="@+id/textView1"
   android:layout_width="200dp"
   android:layout_height="wrap_content"
   android:background="#efefef"
   android:text="First"
   android:textSize="30sp"
   app:layout_constraintBottom_toTopOf="@id/textView2"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintRight_toRightOf="parent"
   app:layout_constraintTop_toTopOf="parent" />
<TextView
   android:id="@+id/textView2"
   android:layout_width="200dp"
   android:layout_height="wrap_content"
   android:background="#e0e0e0"
   android:text="Second"
   android:textSize="30sp"
   app:layout_constraintTop_toBottomOf="@id/textView1"
   app:layout_constraintBottom_toTopOf="@id/textView3"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintRight_toRightOf="parent" />
<TextView
   android:id="@+id/textView3"
   android:layout_width="200dp"
   android:layout_height="wrap_content"
   android:background="#efefef"
   android:text="Third"
   android:textSize="30sp"
   app:layout_constraintTop_toBottomOf="@id/textView2"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintRight_toRightOf="parent"
   app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



2.34-rasm. Dastur natijasi

Elementlarning o'mini o'zgartirish uchun **layout_constraintVertical_chainStyle** atributini zanjirning birinchi elementiga qo'llash kifoya:

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:background="#efefef"
    android:text="First"
    android:textSize="30sp"
    app:layout_constraintVertical_chainStyle="spread_inside"
    app:layout_constraintBottom_toTopOf="@id/textView2"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

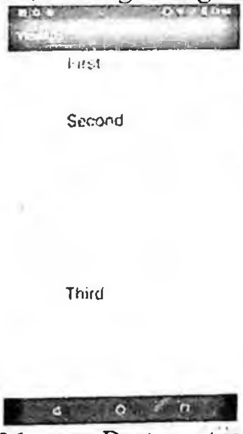


2.35-rasm. Dastur natijasi

Va xuddi vertikal zanjirdagi gorizontal yo'nalishda bo'lgani kabi, **layout_constraintVertical_weight** atributi yordamida element og'irliklaridan foydalanishingiz mumkin. Elementning og'irligini o'rnatish uchun balandligi 0dp ga o'rnatilishi kerak

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    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">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="200dp"
        android:layout_height="0dp"
        android:background="#efefef"
        android:text="First"
        android:textSize="30sp"
        app:layout_constraintVertical_weight="1"
        app:layout_constraintBottom_toTopOf="@id/textView2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="200dp"
        android:layout_height="0dp"
        android:background="#e0e0e0"
        android:text="Second"
        android:textSize="30sp"
        app:layout_constraintVertical_weight="3"
        app:layout_constraintTop_toBottomOf="@id/textView1"
        app:layout_constraintBottom_toTopOf="@id/textView3"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="200dp"
        android:layout_height="0dp"
        android:background="#efefef"
        android:text="Third"
        android:textSize="30sp"
        app:layout_constraintVertical_weight="2"
        app:layout_constraintTop_toBottomOf="@id/textView2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu holda elementlarning umumiy og'irligi 1 + 3 + 2 = 6. Shuning uchun idishning butun balandligi 6 qismga bo'linadi, ulardan birinchi element 1 qismdan, ikkinchisi 3 qismdan va uchinchisi - uning vazniga mos ravishda 2 qism.



2.36-rasm. Dastur natijasi

## 2.9. ConstraintLayout dasturiy yaratish va joylashishni aniqlash.

Java kodida konteyner yaratish uchun shu nomdagi **ConstraintLayout** sinfidan foydalaniladi, ob'ektni yaratish uchun elementning kengligi va balandligi uchun qiymatlar konstruktorga beriladi:

*ConstraintLayout.LayoutParams layoutParams = new ConstraintLayout.LayoutParams*

*(ConstraintLayout.LayoutParams.WRAP_CONTENT ConstraintLayout.LayoutParams.WRAP_CONTENT);*

Birinchi parametr elementning kengligini, ikkinchisi balandlikni o'rnatadi. **ConstraintLayout.LayoutParams.WRAP_CONTENT** element tarkibini ko'rsatish uchun kerak bo'lganda o'lchamlarini belgilaydi. **ConstraintLayout.LayoutParams.WRAP_CONTENT**-ga qo'shimcha ravishda, **ConstraintLayout.LayoutParams.MATCH_CONSTRAINT** doimiysidan foydalanishingiz mumkin, bu layout_width va layout_height atributlarida "0dp" qiymatidan foydalanishga o'xshaydi va elementni konteynerning kengligi yoki balandligiga cho'zadi.

Siz aniq o'lchamlardan ham foydalanishingiz mumkin, masalan:

*ConstraintLayout.LayoutParams layoutParams = new ConstraintLayout.LayoutParams*
*(ConstraintLayout.LayoutParams.MATCH_CONSTRAINT, 200);*

**ConstraintLayout.LayoutParams** sinfi **ConstraintLayout** ichidagi joylashishni sozlash uchun ishlatiladi. Bu juda ko'p funktsiyaga ega. Bunday holda, faqat elementning joylashishini belgilashga imkon beradigan maydonlarni ko'rib chiqing:

- baselineToBaseline: elementning boshlang'ich chizig'ini identifikatori xususiyatga berilgan boshqa elementning boshlang'ich chizig'iga moslashtiradi.
- bottomToBottom: Elementning pastki qismini boshqa elementning pastki qismiga tekislang.
- bottomToTop: Elementning pastki chegarasini boshqa elementning yuqori chegarasi bilan tekislaydi.
- leftToLeft: Elementning chap chegarasini boshqa elementning chap chegarasi bilan tekislaydi.
- leftToRight: Elementning chap chegarasini boshqa elementning o'ng chegarasi bilan tekislaydi.
- rightToLeft: Elementning o'ng chegarasini boshqa elementning chap chegarasi bilan tekislaydi.
- rightToRight: Elementning o'ng chegarasini boshqa elementning o'ng chegarasi bilan tekislaydi.
- startToEnd: elementning boshlanishini boshqa elementning oxiri bilan tekislaydi.
- startToStart: Elementning boshlanishini boshqa elementning boshiga moslashtiradi.
- topToBottom: Elementning yuqori chegarasini boshqa elementning pastki chegarasi bilan tekislaydi.
- topToTop: Elementning yuqori chegarasini boshqa elementning yuqori chegarasi bilan tekislaydi.
- endToEnd: elementning uchini boshqa elementning uchi bilan tekislaydi.
- endToStart: Elementning uchini boshqa elementning boshiga moslashtiradi.

Qiymat sifatida ushbu maydonlar joylashishni aniqlashga nisbatan elementning **id**(identifikatorini) oladi. Agar tartib **ConstraintLayout** konteyneriga nisbatan bo'lsa, u holda **ConstraintLayout.LayoutParams.PARENT_ID** qo'llaniladi.

Eng oddiy misolni ko'rib chiqaylik:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.activity_main);
    ConstraintLayout constraintLayout = new ConstraintLayout(this);
    TextView textView = new TextView(this);
    // matn maydonini matnini sozlash
    textView.setText("Hello Android");
    // matn hajmini sozlash
    textView.setTextSize(30);
    ConstraintLayout.LayoutParams       layoutParams       =       new
ConstraintLayout.LayoutParams
            (ConstraintLayout.LayoutParams.WRAP_CONTENT              ,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
    // konteynerni yuqori chap burchagida joylashtirish
    // app:layout_constraintLeft_toLeftOf="parent" ga ekvivalent
    layoutParams.leftToLeft                                          =
ConstraintLayout.LayoutParams.PARENT_ID;
    // app:layout_constraintTop_toTopOf="parent" ga ekvivalent
    layoutParams.topToTop                                            =
ConstraintLayout.LayoutParams.PARENT_ID;
    // o'lchamlarni o'rnatish
    textView.setLayoutParams(layoutParams);
    // TextView-ni ConstraintLayout-ga qo'shish
    constraintLayout.addView(textView);
    setContentView(constraintLayout);
  }
}
```

Bunday holda, kenglik va balandlik uchun **ConstraintLayout.LayoutParams.WRAP_CONTENT** qiymati element tarkibini ko'rsatish uchun zarur bo'lgan hajmda bo'lishini ko'rsatadi.

Keyin elementning chap chegarasini konteynerning chap tomoniga tekislang:

*layoutParams.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;*

Ushbu parametr dasturni ishlatishda bir xil: **layout_constraintLeft_toLeftOf = "parent"** atributi.

Keyin elementning yuqori chegarasini idishning yuqori qismiga tekislaymiz

*layoutParams.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;*

Ushbu parametr **app:layout_constraintTop_toTopOf="parent"** atributidan foydalanish bilan bir xil.

Nihoyat, **ConstraintLayout.LayoutParams** ob'ektini **TextViewga** qo'llang:

*constraintLayout.addView(textView);*

Natijada, **TextView** elementi **ConstraintLayoutning** yuqori chap burchagida joylashgan bo'ladi:

ViewApp
Hello World!

2.37-rasm. Dastur natijasi

Yana bir misolni ko'rib chiqamiz - elementlarning bir-biriga nisbatan joylashishini belgilash:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        ConstraintLayout constraintLayout = new ConstraintLayout(this);
        EditText editText = new EditText(this);
        editText.setHint("E-mailni kiriting");
        editText.setId(View.generateViewId());
        Button button = new Button(this);
        button.setText("Jo`natish");
        button.setId(View.generateViewId());
        ConstraintLayout.LayoutParams    editTextLayout    =    new
ConstraintLayout.LayoutParams
            (ConstraintLayout.LayoutParams.WRAP_CONTENT                ,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        editTextLayout.leftToLeft                                     =
ConstraintLayout.LayoutParams.PARENT_ID;
        editTextLayout.topToTop                                       =
ConstraintLayout.LayoutParams.PARENT_ID;
        editTextLayout.rightToLeft = button.getId();
        editText.setLayoutParams(editTextLayout);
        constraintLayout.addView(editText);
        ConstraintLayout.LayoutParams    buttonLayout    =    new
ConstraintLayout.LayoutParams
            (ConstraintLayout.LayoutParams.WRAP_CONTENT                ,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        buttonLayout.leftToRight = editText.getId();
        buttonLayout.topToTop                                         =
ConstraintLayout.LayoutParams.PARENT_ID;
        button.setLayoutParams(buttonLayout);
        constraintLayout.addView(button);
        setContentView(constraintLayout);
    }
}
```

Bir elementni boshqasiga nisbatan joylashtirganda, biz ikkinchi element idini bilishimiz kerak. Agar element Java kodida aniqlangan bo'lsa, unda avval siz identifikator yaratishingiz kerak:

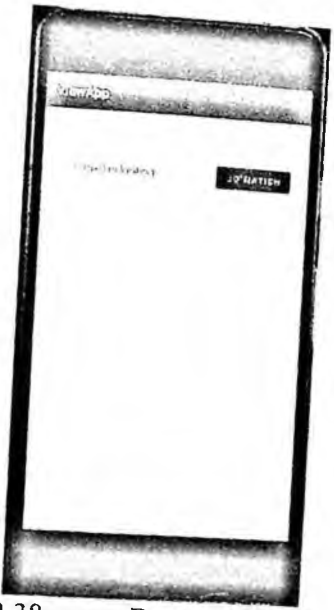
```
editText.setId(View.generateViewId());
button.setId(View.generateViewId());
```

Keyin joylashishni aniqlashni aniqlash uchun element identifikatorlarini qo'llashingiz mumkin. Shunday qilib, **EditText**ning chap chegarasi tugmachaning o'ng chegarasiga to'g'ri keladi:

```
editTextLayout.rightToLeft = button.getId();
```

Va tugmachaning o'ng chegarasi **EditText** chap chegarasiga to'g'ri keladi:

```
buttonLayout.leftToRight = editText.getId();
```



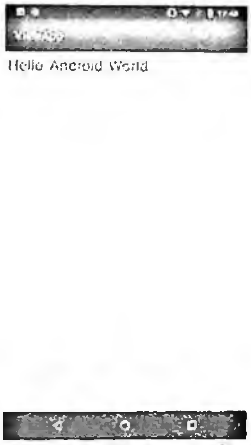
2.38-rasm. Dastur natijasi

## 2.10. LineerLayout konteyneri

**LinearLayout** konteyner - bu eng oddiy konteyner, barcha bolalarini bir yo'nalishda: gorizontal yoki vertikal ravishda joylashtiradigan **ViewGroup**. Barcha elementlar birin-ketin joylashgan. Belgilash yo'nalishi **android:orientation** atributidan foydalanib ko'rsatiladi.

Agar, masalan, maket yo'nalishi vertikal bo'lsa (**android:orientation="vertical"**), unda barcha elementlar ustunda joylashtirilgan - har bir satrda bitta element. Agar yo'nalish gorizontal bo'lsa (**android:orientation="horizontal"**), unda elementlar bitta qatorga joylashtirilgan. Masalan, elementlarni gorizontal qatorga joylashtiramiz:

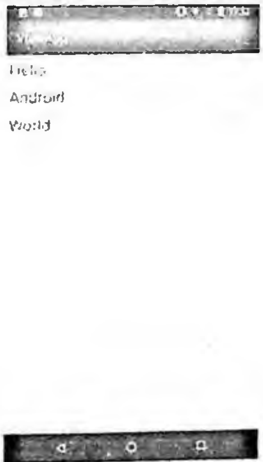
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="Hello"
        android:textSize="26sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="Android"
        android:textSize="26sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="World"
        android:textSize="26sp" />
</LinearLayout>
```



```
Hello Android World
```

2.39-rasm. Dastur natijasi

Agar biz **LinearLayout** uchun **android:orientation="vertical"** atributini ko'rsatgan bo'lsak, u holda elementlar vertikal ravishda joylashtirilgan bo'lar edi:



```
Hello
Android
World
```

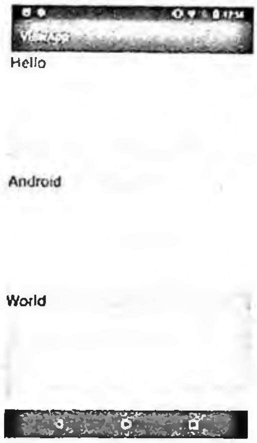
2.40-rasm. Dastur natijasi

**Elementning vazni. LinearLayout** elementning vazn xususiyatini qo'llab-quvvatlaydi, uni **android:layout_weight** xususiyati uzatadi. Ushbu xususiyat, ushbu elementning boshqa ob'ektlarga nisbatan bo'sh joyning qancha qismini olishini ko'rsatadigan qiymatni oladi. Masalan, agar android uchun bitta element 2 qiymatga ega bo'lsa: **layout_weight** xususiyati, ikkinchisi esa 1 qiymatga ega bo'lsa, u holda ular 3 ga qo'shiladi, shuning uchun birinchi element qolgan bo'shliqning 2/3 qismini egallaydi va ikkinchisi - 1/3.

Agar barcha elementlarda **android:layout_weight="1"** bo'lsa, unda barcha elementlar konteynerning butun maydoniga teng taqsimlanadi:

```
        <?xml version="1.0" encoding="utf-8"?>
        <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:orientation="vertical" >
          <TextView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:text="Hello"
            android:background="#e0e0e0"
            android:layout_weight="1"
            android:textSize="26sp" />
          <TextView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:background="#eeeeee"
            android:text="Android"
            android:layout_weight="1"
            android:textSize="26sp" />
          <TextView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:text="World"
            android:background="#bdbdbd"
            android:layout_weight="1"
            android:textSize="26sp" />
        </LinearLayout>
```

Bunday holda, **LinearLayout** vertikal yo'nalishga ega, shuning uchun barcha elementlar yuqoridan pastga qarab joylashtiriladi. Uch elementda har **android:layout_weight="1"** mavjud, shuning uchun barcha elementlarning og'irliklari yig'indisi 3 ga teng bo'ladi va har bir element **LinearLayout**da bo'sh joyning uchdan bir qismini oladi:



2.41-rasm. Dastur natijasi

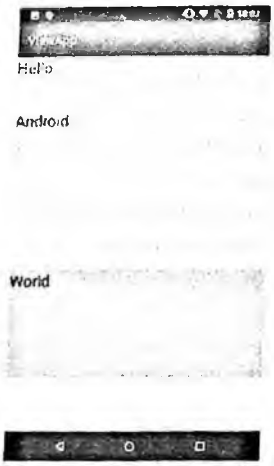
Bundan tashqari, bizda vertikal suyakka ega bo'lgani uchun **layout_height** xususiyatini 0dp ga o'rnatishimiz kerak. Agar **LinearLayout** gorizontal yo'nalishga ega bo'lsa, u holda layout_width xususiyati 0dp ga o'rnatilishi kerak edi.

Boshqa bir atribut **android:weightSum** barcha elementlarning og'irliklari yig'indisini ko'rsatishga imkon beradi. Masalan:

```
        <?xml version="1.0" encoding="utf-8"?>
        <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:orientation="vertical"
          android:weightSum="7">
          <TextView
             android:layout_width="match_parent"
             android:layout_height="0dp"
             android:text="Hello"
             android:background="#e0e0e0"
             android:layout_weight="1"
             android:textSize="26sp" />
          <TextView
             android:layout_width="match_parent"
             android:layout_height="0dp"
             android:background="#eeeeee"
             android:text="Android"
             android:layout_weight="3"
             android:textSize="26sp" />
          <TextView
             android:layout_width="match_parent"
             android:layout_height="0dp"
             android:text="World"
        android:background="#bdbdbd"
        android:layout_weight="2"
        android:textSize="26sp" />
</LinearLayout>
```

**LineerLayout** bu erda og'irliklarning yig'indisini 7 ga teng o'rnatadi, ya'ni butun vertikal bo'shliq (vertikal yo'nalishdan beri) shartli ravishda etti teng qismga bo'linadi.

Birinchi **TextView**ning og'irligi 1 ga teng, ya'ni etti qismdan faqat bittasini oladi. Ikkinchi **TextView**ning og'irligi 3 ga teng, ya'ni etti qismdan uch qismini egallaydi. Va uchinchisining vazni 2 ga teng. Umumiy miqdori 6. Ammo **LinearLayout** og'irlikni 7 ga o'rnatganligi sababli, bitta qism barcha elementlardan xoli bo'ladi.



2.42-rasm. Dastur natijasi

**Dasturiy jihatdan LinearLayout yaratish.** Java kodida LinearLayout yaratish:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.LinearLayout;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.activity_main);
    LinearLayout linearLayout = new LinearLayout(this);
    // gorizontal yo'nalish
    linearLayout.setOrientation(LinearLayout.HORIZONTAL);
    TextView textView = new TextView(this);
    textView.setText("Hello");
    textView.setTextSize(30);
```

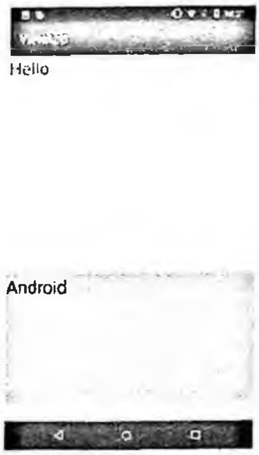
```
            // element uchun joylashishni aniqlash parametrlarini yaratish
            LinearLayout.LayoutParams          layoutParams          =          new
LinearLayout.LayoutParams
                (LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT);
            // chekkalarni o'rnatish
            layoutParams.setMargins(100, 100, 0, 0);
            textView.setLayoutParams(layoutParams);
            // добавляем элемент в LinearLayout
            linearLayout.addView(textView);
            setContentView(linearLayout);
        }
    }
```

2.43-rasm. Dastur natijasi

**LinearLayout.LayoutParams()** konstruktorining qo'shimcha versiyasi elementning og'irligini uchinchi parametr sifatida ko'rsatishga imkon beradi:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.LinearLayout;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    LinearLayout linearLayout = new LinearLayout(this);
    linearLayout.setOrientation(LinearLayout.VERTICAL);
    // birinchi matn maydoni
    TextView textView1 = new TextView(this);
    textView1.setText("Hello");
    textView1.setTextSize(30);
```

```
        // textView1 3 vaznga ega
        linearLayout.addView(textView1, new LinearLayout.LayoutParams
              (LinearLayout.LayoutParams.MATCH_PARENT, 0, 3));
        // ikkinchi matn maydoni
        TextView textView2 = new TextView(this);
        textView2.setText("Android");
        textView2.setBackgroundColor(0xFFBDBDBD);
        textView2.setTextSize(30);
        // textView2 - 2 vaznga ega
        linearLayout.addView(textView2, new LinearLayout.LayoutParams
              (LinearLayout.LayoutParams.MATCH_PARENT, 0, 2));
        setContentView(linearLayout);
    }
}
```

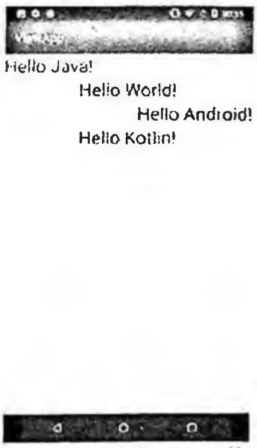


2.44-rasm. Dastur natijasi

**Layout_gravity**. **Layout_gravity** atributi **LinearLayout**ga nisbatan joylashishni aniqlashga imkon beradi. Buning uchun quyidagi qiymatlar kerak:

- **top:** Elementni konteynerning yuqori qismiga tekislang
- **bottom**: elementni konteynerning pastki qismiga tekislaydi
- **left**: elementni konteynerning chap chegarasiga tekislaydi
- **right**: elementni konteynerning o'ng chegarasiga tekislaydi
- **center_vertical**: Elementni markazga vertikal ravishda tekislaydi
- **center_horizontal**: Elementni gorizontal ravishda markazlashtiring
- **center**: element markazga joylashtirilgan
- **fill_vertical**: element vertikal ravishda cho'zilgan
- **fill_horizontal**: element gorizontal ravishda cho'zilgan
- **fill**: element butun konteyner maydonini to'ldiradi
- **clip_vertical**: elementning yuqori va pastki chegaralarini qisadi
- **clip_horizontal**: Elementning o'ng va chap chegaralarini kesadi
- **start**: element konteynerning boshida (yuqori chap burchakda) joylashgan
- **end**: element konteynerning oxirida joylashgan (yuqori o'ng burchak)

Masalan:
```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
  <TextView
    android:layout_gravity="left"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="Hello Java!"
    android:background="#e8eaf6"/>
  <TextView
    android:layout_gravity="center"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="Hello World!"
    android:background="#e8eaf6"/>
  <TextView
    android:layout_gravity="right"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="Hello Android!"
    android:background="#e8eaf6"/>
  <TextView
    android:layout_gravity="center"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="Hello Kotlin!"
    android:background="#e8eaf6"/>
</LinearLayout>
```
Bunday holda, birinchi **TextView** konteynerning chap tomonida (**android:layout_gravity="left"**), ikkinchisi **TextView** markazda (**android:layout_gravity="center"**), uchinchisi o'ng tomonda (**android:layout_gravity="right"**) va to'rtinchisi markazlashtirilgan (**android:layout_gravity="center"**).



```
Hello Java!
        Hello World!
                Hello Android!
        Hello Kotlin!
```

2.45-rasm. Dastur natijasi

Konteynerning yo'nalishini ko'rib chiqishga arziydi. Masalan, vertikal yo'nalishda barcha elementlar yuqoridan pastga qarab vertikal stekni ifodalaydi. Shuning uchun elementning vertikal joylashuvi bilan bog'liq qiymatlar (masalan, **top** yoki **bottom**) elementga hech qanday ta'sir qilmaydi. Bundan tashqari, gorizontal yo'naltirilganda, **LinearLayout** elementni gorizontal joylashtiradigan qiymatlarga ta'sir qilmaydi, masalan, **left** va **right**.

**layout_gravity** parametrini dasturiy ravishda sozlash uchun **LinearLayout.LayoutParams** ob'ektining g_ravity maydonini o'rnatishingiz kerak:

*LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams*
*(LinearLayout.LayoutParams.MATCH_PARENT, LinearLayout.LayoutParams.MATCH_PARENT);*
*// layout_gravity o'rnatish*
*layoutParams.gravity = Gravity.CENTER;*

Qiymat **Gravity** sinfining atribut qiymatlariga o'xshash doimiylaridan biriga o'tkaziladi.

## 2.11. RelativeLayout

**RelativeLayout** - bu **ViewGroup** ob'ektini ifodalaydi, u bolalarni belgilashdagi boshqa bolalarning holatiga nisbatan yoki **RelativeLayout**ning o'zi maydoniga nisbatan joylashtiradi. Nisbatan joylashishni aniqlash yordamida biz elementni o'ngga yoki markazga yoki ushbu konteyner ta'minlaydigan boshqa usulda joylashtirishimiz mumkin. **xml** faylida elementni o'rnatish uchun biz quyidagi atributlarni qo'llashimiz mumkin:

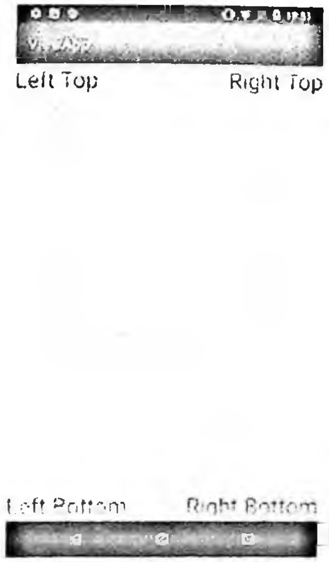
- **android:layout_above**: Belgilangan Id bilan element ustidagi elementni joylashtiradi
- **android:layout_below**: belgilangan id bilan element ostidagi elementni joylashtiradi

- **android:layout_toLeftOf**: belgilangan Id bilan elementning chap tomonida joylashgan
- **android:layout_toRightOf**: belgilangan Id bilan elementning o'ng tomonida joylashgan
- **android:layout_toStartOf**: Belgilangan Id bilan element boshlanadigan joriy elementning boshlanishini joylashtiradi
- **android:layout_toEndOf**: Belgilangan Idga ega element tugaydigan joriy elementning boshlanishini joylashtiradi
- **android:layout_alignBottom**: Elementni boshqa elementning pastki chegarasiga belgilangan Id bilan moslashtiradi
- **android:layout_alignLeft**: Belgilangan Id bilan elementni boshqa elementning chap chegarasiga tekislaydi
- **android:layout_alignRight**: Elementni boshqa elementning o'ng chegarasiga belgilangan Id bilan moslashtiradi
- **android:layout_alignStart**: elementni belgilangan Id bilan boshqa elementni boshlaydigan satrga moslashtiradi
- **android:layout_alignEnd**: Elementni belgilangan Id bilan boshqa element bilan tugaydigan qatorga tekislaydi
- **android:layout_alignTop**: Elementni boshqa elementning yuqori chegarasiga belgilangan Id bilan moslashtiradi
- **android:layout_alignBaseline**: Elementning boshlang'ich chizig'ini ko'rsatilgan element bilan boshqa elementning asosiy chizig'iga moslashtiradi
- **android:layout_alignParentBottom**: agar atribut true ga o'rnatilgan bo'lsa, u holda element konteynerning pastki qismiga siqiladi
- **android:layout_alignParentRight**: agar atribut true ga o'rnatilgan bo'lsa, u holda element konteynerning o'ng chetiga siqiladi
- **android:layout_alignParentLeft**: agar atribut true ga o'rnatilgan bo'lsa, element konteynerning chap chetiga siqib qo'yiladi
- **android:layout_alignParentStart**: agar atribut true-ga o'rnatilgan bo'lsa, u holda element konteynerning boshlang'ich chetiga bosiladi (chap tomon yo'naltirilgan matn yo'nalishi bilan, chap chekka)
- **android:layout_alignParentEnd**: agar atribut true-ga o'rnatilgan bo'lsa, u holda element konteynerning so'nggi chetiga siqiladi (matnning chap tomoni bilan - o'ng qirrasi bilan)
- **android:layout_alignParentTop**: agar atribut true bo'lsa, u holda element konteynerning yuqori chegarasiga bosiladi
- **android:layout_centerInParent**: agar atribut true bo'lsa, u holda element asosiy konteynerda joylashgan bo'ladi
- **android:layout_centerHorizontal**: true ga o'rnatilganda, elementni gorizontal ravishda markazga tekislaydi
- **android:layout_centerVertikal**: true ga o'rnatilganda, elementni markazga vertikal ravishda tekislaydi

Masalan, **RelativeLayout** konteyneriga nisbatan joylashishni aniqlash:

*<?xml version="1.0" encoding="utf-8"?>*

```
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView android:text="Left Top"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:textSize="26sp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true" />
  <TextView android:text="Right Top"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:textSize="26sp"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true" />
  <TextView android:text="Left Bottom"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:textSize="26sp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true" />
  <TextView android:text="Right Bottom"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:textSize="26sp"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true" />
</RelativeLayout>
```

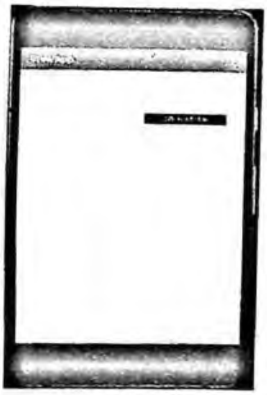


2.46-rasm. Dastur natijasi

Boshqa elementga nisbatan joylashish uchun biz ushbu elementning **idni** belgilashimiz kerak. Shunday qilib, **RelativeLayoutga** matnli maydon va tugmani qo'yamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <EditText
    android:id="@+id/edit_message"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"/>
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Отправить"
    android:layout_alignRight="@id/edit_message"
    android:layout_below="@id/edit_message"
    />
</RelativeLayout>
```

Bunday holda, EditText RelativeLayout markazida joylashgan bo'lib, tugma EditText ostiga qo'yilib, uning o'ng chegarasiga to'g'ri keladi:


2.47-rasm. Dastur natijasi

**Dasturiy jihatdan RelativeLayout yarating**. Dasturiy jihatdan Java kodid
**RelativeLayout** elementini yarataylik:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      RelativeLayout relativeLayout = new RelativeLayout(this);
      EditText editText = new EditText(this);
      editText.setId(EditText.generateViewId());
      Button button = new Button(this);
      button.setText("Jo'natish");
      // EditText uchun pozitsiya parametrlarini belgilash
      RelativeLayout.LayoutParams        editTextParams        =        new
RelativeLayout.LayoutParams(
            RelativeLayout.LayoutParams.MATCH_PARENT.
            RelativeLayout.LayoutParams.WRAP_CONTENT
      );
      // bosh konteynerning markaziy tekislanishi
      editTextParams.addRule(RelativeLayout.CENTER_IN_PARENT);
      // RelativeLayout-ga qo'shish
      relativeLayout.addView(editText, editTextParams);
      // B_utton uchun pozitsiya parametrlarini o'rnating
      RelativeLayout.LayoutParams        buttonParams        =        new
RelativeLayout.LayoutParams(
            RelativeLayout.LayoutParams.WRAP_CONTENT.
            RelativeLayout.LayoutParams.WRAP_CONTENT
            );
            // EditText maydonining o'ng va pastki qismlariga tekislang
            buttonParams.addRule(RelativeLayout.BELOW, editText.getId());
            buttonParams.addRule(RelativeLayout.ALIGN_RIGHT,
editText.getId());
            // добавляем в RelativeLayout
            relativeLayout.addView(button, buttonParams);
            setContentView(relativeLayout);
        }
    }
```

**RelativeLayout.LayoutParams** sinfi elementning konteynerdagi holatini belgilash uchun ishlatiladi. Kenglik va balandlik uchun qiymatlar konstruktor orqali o'rnatiladi. Masalan, **EditText** elementi uchun kenglik **MATCH_PARENT** va balandlik **WRAP_CONTENT** ga o'rnatiladi.

**AddRule()** usuli bilan elementni joylashtirish uchun qo'shimcha qoidalar qo'shishimiz mumkin. Ushbu usul joylashishni aniqlash parametrini ifodalovchi va atributga o'xshash parametr sifatida raqamli doimiyni oladi. Masalan, **android:layout_centerInParent** atributi **CENTER_IN_PARENT** doimiysiga, **android:layout_alignRight** atributi esa **ALIGN_RIGHT** doimiysiga mos keladi.

Shuni ta'kidlash kerakki, **EditTextda id** ni o'rnatish uchun kodni soddalashtirish uchun **generateViewId()**; usuli chaqiriladi, bu boshqarish uchun **id**ni dasturiy ravishda yaratishga imkon beradi.

Keyin tugma qoidalarini o'rnatishda **id** to'plami **addRule** uslubiga ikkinchi parametr sifatida uzatiladi:

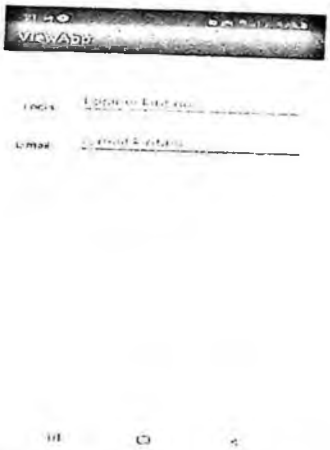
*buttonParams.addRule(RelativeLayout.BELOW, editText.getId());*

Shunday qilib, biz joylashishni qaysi elementga nisbatan belgilashimiz kerakligini ko'rsatamiz.

## 2.12. TableLayout konteyneri

**TableLayout** konteyner jadvalga o'xshash boshqaruvlarni ustunlar va qatorlarda tuzadi. **activity_main.xml** faylida **TableLayout** elementini aniqlaymiz, u ikkita qator va ikkita ustunni o'z ichiga oladi:

```
    <TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TableRow>
          <TextView
            android:layout_weight="0.5"
            android:text="Login"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
          <EditText
            android:layout_weight="1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </TableRow>
    <TableRow>
        <TextView
            android:layout_weight="0.5"
            android:text="E-mail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <EditText
            android:layout_weight="1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </TableRow>
</TableLayout>-
```



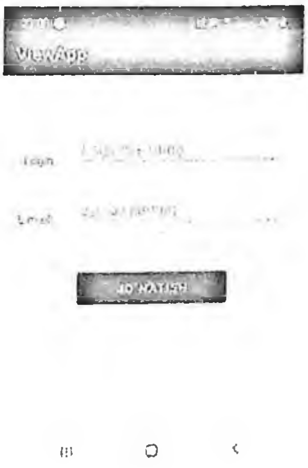
2.48-rasm. Dastur natijasi

**TableRow** elementi yordamida biz bitta qatorni hosil qilamiz. Belgilanish qancha ustun yaratish kerakligini qanday biladi? Android maksimal darajada bir xil darajadagi vidjetlar sonini topadi va bu raqam ustunlar sonini bildiradi. Masalan, bu holda bizda ikkita chiziq aniqlangan va ularning har biri ikkita elementga ega. Agar ulardan bittasida uchta vidjet bo'lsa, unda boshqa qatorda ikkita vidjet qolgan bo'lsa ham, uchta ustun bo'ladi.

Bundan tashqari, **TableRow** elementi **LinearLayout** sinfidan meros bo'lib o'tgan, shuning uchun biz unga **LinearLayout** bilan bir xil funktsiyalarni qo'llashimiz mumkin. Xususan, **android:layout_weight** atributi ketma-ket narsalar uchun joyni aniqlash uchun ishlatiladi.

Agar elementni bir qator ustunlar bo'ylab cho'zish kerak bo'lsa, biz elementni qancha ustunlarga cho'zish kerakligini ko'rsatadigan layout_span atributidan foydalanib uzaytiramiz:

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TableRow>
      <TextView
        android:textSize="22sp"
        android:text="Login"
        android:layout_width="100dp"
        android:layout_height="wrap_content" />
      <EditText
        android:textSize="22sp"
        android:layout_width="200dp"
        android:layout_height="wrap_content" />
    </TableRow>
    <TableRow>
      <TextView
        android:textSize="22sp"
        android:text="E-mail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
      <EditText
        android:textSize="22sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    </TableRow>
    <TableRow>
      <Button
        android:text="Jo`natish"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_span="2"/>
    </TableRow>
</TableLayout>
```



2.49-rasm. Dastur natijasi

Shuningdek, **android:layout_weight="1"** atributini o'rnatib, elementni butun qatoriga cho'zishingiz mumkin:

```
<TableRow>
   <Button
      android:text="Jo`natish"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_weight="1" />
</TableRow>
```

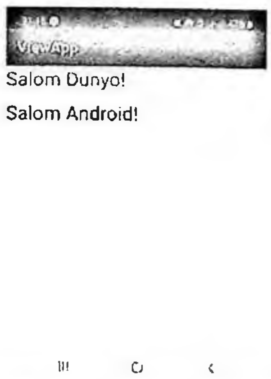
## 2.13. FrameLayout konteyneri

**FrameLayout** konteyner unda joylashtirilgan bitta vizual elementni namoyish qilish uchun mo'ljallangan. Agar biz bir nechta elementlarni joylashtirsak, unda ular bir-birining ustiga chiqadi. Shu bilan birga, **FrameLayout**da bir nechta elementlarni joylashtirishingiz mumkin.

Aytaylik, **FrameLayout**da ikkita **TextView** boshqaruv elementlarini joylashtirdik:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Salom Dunyo!"
      android:textSize="26sp"/>
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Salom Android!"
      android:textSize="26sp"
      android:layout_marginTop="50dp"/>
</FrameLayout>
```

Bu erda ikkala element ham bir joyda - **FrameLayout** konteynerining yuqori chap burchagida joylashtirilgan va bir-birini takrorlamaslik uchun, bu holda ikkinchi **TextView** 50 ta yuqori chegaraga o'rnatiladi.


Salom Dunyo!

Salom Android!

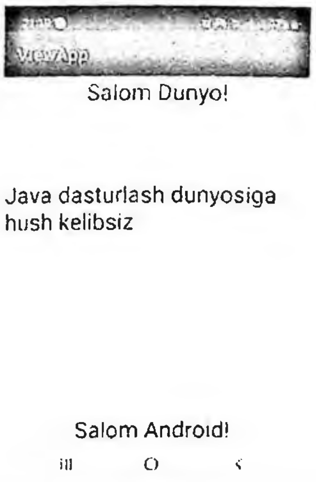
2.50-rasm. Dastur natijasi

Ko'pincha **FrameLayout**, aylantirishni ta'minlaydigan **ScrollView** kabi olingan konteynerlarni yaratish uchun ishlatiladi.

**FrameLayoutga** joylashtirilgan boshqaruv elementlari **android:layout_gravity** xususiyati yordamida joylashishni aniqlashlari mumkin:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="26sp"
    android:layout_gravity="center_horizontal" />
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Welcome to Java World"
    android:textSize="26sp"
    android:layout_gravity="center"/>
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello Android!"
    android:textSize="26sp"
    android:layout_gravity="bottom|center_horizontal"/>
</FrameLayout>
```

Qiymatni belgilashda biz bir qator qiymatlarni vertikal chiziq bilan ajratish orqali birlashtira olamiz: **bottom|center_horizontal.**


ViewApp

Salom Dunyo!

Java dasturlash dunyosiga
hush kelibsiz

Salom Android!

2.51-rasm. Dastur natijasi

**FrameLayout** dasturini **MainActivity** kodida dasturiy ta'minot bilan yarating:

```
    package com.example.viewapp;
    import androidx.appcompat.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.Gravity;
    import android.widget.FrameLayout;
    import android.widget.TextView;
    public class MainActivity extends AppCompatActivity {
      @Override
      protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        FrameLayout frameLayout = new FrameLayout(this);
        TextView textView = new TextView(this);
        textView.setText("Hello World!");
        FrameLayout.LayoutParams        layoutParams        =        new
FrameLayout.LayoutParams
            (FrameLayout.LayoutParams.WRAP_CONTENT,
FrameLayout.LayoutParams.WRAP_CONTENT);
        layoutParams.gravity    =    Gravity.CENTER_HORIZONTAL
Gravity.TOP;
        textView.setLayoutParams(layoutParams);
        textView.setTextSize(26);
        frameLayout.addView(textView);
        setContentView(frameLayout);
      }
    }
```

## 2.14. GridLayout konteyneri

**GridLayout** jadval ko'rinishini yaratishga imkon beradigan yana bir konteynerni taqdim etadi. **GridLayout** qatorlar to'plamidan iborat bo'lib, ularning har biri alohida katakchalardan iborat:

```
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="3"
    android:columnCount="3">
    <Button android:text="1" />
    <Button android:text="2" />
    <Button android:text="3" />
    <Button android:text="4" />
    <Button android:text="5" />
    <Button android:text="6" />
    <Button android:text="7" />
    <Button android:text="8" />
    <Button android:text="9" />
</GridLayout>
```

**android:rowCount** va **android:columnCount** atributlari navbati bilan qatorlar va ustunlar sonini o'rnatadi. Shunday qilib, bu holda biz 3 qator va 3 ustunni o'rnatdik. **GridLayout** ichki o'rnatilgan boshqaruv elementlarini satrlar bo'yicha avtomatik ravishda joylashtirishi mumkin. Shunday qilib, bizning holatimizda birinchi tugma birinchi katakka (birinchi qator, birinchi ustun), ikkinchi tugma ikkinchi katakka va hokazolarga o'tadi.

Bunday holda, ustunlar kengligi avtomatik ravishda eng keng elementning kengligiga o'rnatiladi.

2.52-rasm. Dastur natijasi

Biroq, biz ma'lum bir element uchun ustun va satr raqamini aniq belgilab olamiz va agar kerak bo'lsa, bir nechta ustunlar yoki qatorlar bo'ylab cho'zilib ketamiz. Buning uchun biz quyidagi atributlarni qo'llashimiz mumkin:

- **android: layout_column:** ustun raqami (noldan hisoblash)
- **android: layout_row:** satr raqami
- **android: layout_columnSpan:** elementga cho'zilgan ustunlar soni
- **android: layout_rowSpan:** element kengaytirilgan qatorlar soni

Masalan:

```
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="3"
    android:columnCount="3">
    <Button
      android:text="1"
      android:layout_column="0"
      android:layout_row="0" />
    <Button android:text="2"
      android:layout_column="1"
      android:layout_row="0"/>
    <Button android:text="3"
      android:layout_column="2"
      android:layout_row="0" />
    <Button android:text="4"
      android:layout_width="180dp"
      android:layout_columnSpan="2"/>
    <Button android:text="5"
      android:layout_height="100dp"
      android:layout_rowSpan="2"/>
    <Button android:text="6" />
    <Button android:text="7"/>
</GridLayout>
```

2.53-rasm. Dastur natijasi

## 2.15. ScrollView konteyneri

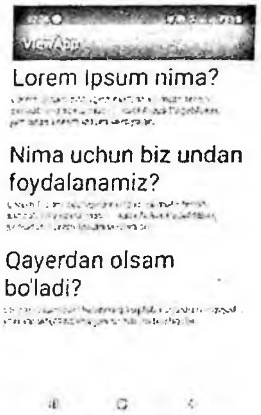
**ScrollView** konteynerida barcha elementlari birdaniga qurilmaning ekraniga sig'maydigan bunday interfeys uchun aylantirishni yaratish uchun mo'ljallangan. **ScrollView** faqat bitta elementni o'z ichiga olishi mumkin, shuning uchun agar biz bir nechta elementlarni joylashtirmoqchi bo'lsak, ular qandaydir idishga joylashtirilishi kerak.

Masalan, katta matnli **TextView** qatorini aniqlaylik:

```
<ScrollView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Lorem Ipsum nima?"
      android:textSize="34sp" />
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=" Lorem Ipsum oddiygina matbaa va matn terish
sanoatining soxta matni... kabi Aldus PageMaker, jumladan Lorem Ipsum
versiyalari."
      android:textSize="14sp"/>
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=" Nima uchun biz undan foydalanamiz?"
      android:layout_marginTop="16dp"
      android:textSize="34sp"/>
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=" Lorem Ipsum oddiygina matbaa va matn terish
sanoatining soxta matni... kabi Aldus PageMaker, jumladan Lorem Ipsum
versiyalari."
      android:textSize="14sp"/>
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=" Qayerdan olsam bo'ladi?"
```

```
            android:layout_marginTop="16dp"
            android:textSize="34sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=" Lorem Ipsum parchalarining ko'plab variantl
mavjud... yoki xarakterli bo'lmagan so'zlar va boshqalar."
            android:textSize="14sp"/>
    </LinearLayout>
</ScrollView>
```

**ScrollView**ga faqat bitta element joylashtirilishi mumkinligi sababli, barc **TextViews** **LinearLayout**ga joylashtirilgan. Agar ekran maydo **LinearLayout**ning barcha tarkibiga mos kelmasa, aylantirish imkoniyati pay bo'ladi:



2.54-rasm. Dastur natijasi

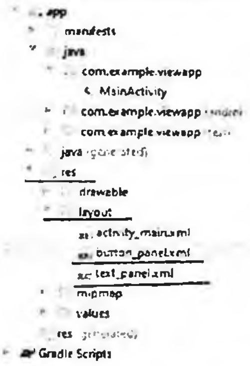
**MainActivity** kodida dasturiy jihatdan **ScrollView** yarating:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.ScrollView;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        ScrollView scrollView = new ScrollView(this);
        TextView textView = new TextView(this);
        textView.setText("Lorem Ipsum is simply dummy text of the printing and
typesetting industry...like Aldus PageMaker including versions of Lorem Ipsum.");
        textView.setLayoutParams(new ViewGroup.LayoutParams
                (ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            textView.setTextSize(26);
            scrollView.addView(textView);
            setContentView(scrollView);
        }
    }
```

### 2.16. Ichki layout konteyneri

Bitta **layout** boshqa **layout**ni o'z ichiga olishi mumkin. Buning uchun **include** elementi ishlatiladi.

Masalan, **res/layout** papkasiga ikkita layout faylini qo'shamiz, ular **text_panel.xml** va **button_panel.xml** deb nomlanadi:



```
app
    manifests
    java
        com.example.viewapp
            MainActivity
        com.example.viewapp (androidTest)
        com.example.viewapp (test)
    java (generated)
    res
        drawable
        layout
            activity_main.xml
            button_panel.xml
            text_panel.xml
        mipmap
        values
    res (generated)
Gradle Scripts
```

2.55-rasm. Loyiha strukturasi

**text_panel.xml** faylida quyidagi kodni aniqlang:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
  <TextView
    android:id="@+id/clicksText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="0 Clicks"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Aslida, bu faqat matnni ko'rsatish uchun **TextView** maydonini belgilaydi.

**button_panel.xml** faylida biz quyidagi belgilashni aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click"
    android:onClick="onClick"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu biz bosish paytida ishlaydigan tugmani belgilaydi.

Ilova interfeysini belgilaydigan asosiy belgilash fayli hanuz **activ_main.xml**. Keling, uni o'zgartiraylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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"
  android:padding="16dp"
  tools:context=".MainActivity">
  <include
    android:id="@+id/textView"
    layout="@layout/text_panel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/button"
    />
  <include
    android:id="@+id/button"
    layout="@layout/button_panel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**ConstraintLayout** yordamida butun interfeys vertikal stekka joylashtirilgan. **ConstraintLayout** ichidagi elementlar **text_panel.xml** va **button_panel.xml** fayllarining tarkibini qo'shadi. **Layout** atributi fayl nomini ko'rsatish uchun ishlatiladi.

Bu **include** elementi o'rniga to'g'ridan-to'g'ri fayllarning tarkibini qo'shganimiz bilan bir xil. Biroq, bu usul o'zining afzalliklariga ega. Masalan, belgilashning ba'zi qismlari, boshqaruv elementlari guruhi har xil **activityda** takrorlanishi mumkin. Va bu elementlarni yuz marta aniqlamaslik uchun ularni alohida **layout** faylga qo'yishingiz va ularni **include** yordamida qo'shishingiz mumkin.

**ConstraintLayout**ga qo'shilgandan so'ng, ushbu konteynerda ichki elementlarga qo'llaniladigan barcha standart atributlar **include** elementlariga qo'llanilishi mumkin, masalan, o'lchamlarni, pozitsiyani sozlash. Shuni ham ta'kidlash joizki, tashqi **layout**ni nafaqat **ConstraintLayout**ga, balki boshqa konteynerlarga ham qo'shishingiz mumkin (**LinearLayout, RelativeLayout** va boshqalar).

**MainActivity** kodini ham o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   int clicks = 0;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void onClick(View view){
      TextView clicksText = findViewById(R.id.clicksText);
      clicks++;
      clicksText.setText(clicks + " Clicks");
   }
}
```

**MainActivity**da biz joylashtirilgan **layout**dagi narsalarga murojaat qilishimiz mumkin. Masalan, biz tugmachani bosish uchun ishlov beruvchini o'rnatamiz, u erda bosilganda **TextView**dagi matnni o'zgartiramiz.

ViewApp
5 Clicks
CLICK

2.56-rasm. Dastur natijasi

Shu bilan birga, biz bir **layout** fayliga bir necha marta boshqa **layout** fayli qo'shishimiz mumkin. Buning uchun avvalo button_panel.xml faylini quyidagicl o'zgartiramiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:background="#3F51B5"
   android:paddingTop="10dp"
   android:paddingBottom="10dp">
   <Button
      android:id="@+id/clickBtn"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Va **activity_main.xml** faylini o'zgartiring:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   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"
   android:padding="16dp"
   tools:context=".MainActivity">
```

```
    <include
        layout="@layout/text_panel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />
    <include layout="@layout/button_panel"
        android:id="@+id/plus_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/minus_button"/>
    <include layout="@layout/button_panel"
        android:id="@+id/minus_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="36dp"
        app:layout_constraintLeft_toRightOf="@id/plus_button"
        app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

**button_panel.xml** fayli endi ikki marta qo'shildi. Ushbu faylni qo'shganda har bir **include** elementiga ma'lum bir **id** belgilanishi muhimdir. Ushbu **id** orqali qaysi element haqida gap ketayotganini bilib olamiz.

**MainActivity**ni ham o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    int clicks = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        View plusButtonView = findViewById(R.id.plus_button);
        View minusButtonView = findViewById(R.id.minus_button);
        TextView clicksText = findViewById(R.id.clicksText);
        Button plusButton = plusButtonView.findViewById(R.id.clickBtn);
        Button minusButton = minusButtonView.findViewById(R.id.clickBtn);
        plusButton.setText("+");
        minusButton.setText("-");
        plusButton.setOnClickListener(v -> {
            clicks++;
            clicksText.setText(clicks + " Clicks");
        });
        minusButton.setOnClickListener(v -> {
            clicks–;
            clicksText.setText(clicks + " Clicks");
        });
    }
}
```

Bu erda biz avval **id** tomonidan individual **include** elementlarini olamiz. Keyin, ushbu elementlar ichida biz tugmachani olamiz. Shundan so'ng biz tugma uchun har qanday matnni o'rnatib, chertish hodisasi ishlovchisini osib qo'yishimiz mumkin. Shunday qilib, ikkala tugmachaning harakati boshqacha bo'ladi.

ViewApp
2 Clicks

2.57-rasm. Dastur natijasi

### 2.17. Gravity va element ichida joylashishni aniqlash.

**Gravity**. **gravity** atributi vizual element tarkibidagi joylashishni belgilaydi. U quyidagi qiymatlarni qabul qilishi mumkin:

- o **top**: buyumlar yuqori qismga joylashtirilgan
- o **bottom**: elementlar pastki qismga joylashtirilgan
- o **left**: elementlar chap tomonda joylashgan
- o **right**: buyumlar idishning o'ng tomoniga joylashtirilgan
- o **center_vertical**: elementlarni markazga vertikal ravishda tekislaydi
- o **center_horizontal**: elementlarni gorizontal ravishda markazlashtiring
- o **center**: buyumlar markazlashtirilgan
- o **fill_vertical**: element vertikal ravishda cho'zilgan
- o **fill_horizontal**: element gorizontal ravishda cho'zilgan
- o **fill**: element butun konteyner maydonini to'ldiradi
- o **clip_vertical**: elementlarning yuqori va pastki chegaralarini qisadi
- o **clip_horizontal**: Elementlarning o'ng va chap chegaralarini kesadi

- **start**: element idishning boshida (yuqori chap burchakda) joylashgan
- **end**: element idishning oxirida joylashgan (yuqori o'ng burchak)

Masalan, matnni **TextView**ning pastki qismiga qo'yamiz:

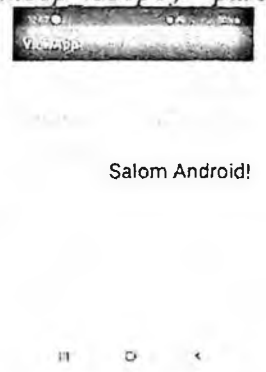
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">
  <TextView
    android:gravity="bottom"
    android:layout_width="0dp"
    android:layout_height="200dp"
    android:text="Hello Android!"
    android:textSize="30sp"
    android:background="#e8eaf6"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2.58-rasm. Dastur natijasi

Agar kerak bo'lsa, biz ulami quvur bilan ajratish orqali qiymatlami birlashtira olamiz:

```
<TextView
  android:gravity="bottom|right"
  android:layout_width="0dp"
  android:layout_height="200dp"
  android:text="Hello Android!"
  android:textSize="30sp"
  android:background="#e8eaf6"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent"
```

```
app:layout_constraintTop_toTopOf="parent" />
```



2.59-rasm. Dastur natijasi

**gravity dasturini o'rnatish**. Element uchun **gravity** parametrini o'rnatish uchun **setGravity()** usulini chaqiring. Parametr sifatida **Gravity** sinfining konstantalaridan biri atributning qiymatlariga o'xshash bo'lgan metodga o'tkaziladi (bundan tashqari, ismlar katta harflar bilan):

```
        package com.example.viewapp;
        import androidx.appcompat.app.AppCompatActivity;
        import androidx.constraintlayout.widget.ConstraintLayout;
        import android.os.Bundle;
        import android.view.Gravity;
        import android.widget.TextView;
        public class MainActivity extends AppCompatActivity {
           @Override
           protected void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              ConstraintLayout constraintLayout = new ConstraintLayout(this);
              TextView textView = new TextView(this);
              textView.setText("Hello Android!");
              textView.setTextSize(30);
              textView.setBackgroundColor(0xffe8eaf6);
              // gravity ni o'rnatish
              textView.setGravity(Gravity.CENTER);
              // kengligi va balandligini o'rnatish
              ConstraintLayout.LayoutParams      layoutParams      =      new
ConstraintLayout.LayoutParams
                 (ConstraintLayout.LayoutParams.MATCH_CONSTRAINT, 200);
              layoutParams.leftToLeft
ConstraintLayout.LayoutParams.PARENT_ID;
              layoutParams.rightToRight
ConstraintLayout.LayoutParams.PARENT_ID;
              layoutParams.topToTop
ConstraintLayout.LayoutParams.PARENT_ID;
```

```
            layoutParams.bottomToBottom
ConstraintLayout.LayoutParams.PARENT_ID;
            textView.setLayoutParams(layoutParams);
            constraintLayout.addView(textView);
            setContentView(constraintLayout);
        }
    }
```

ViewApp
Hello Android!

2.60-rasm. Dastur natijasi

Bir nechta qiymatlarni birlashtirish uchun vertikal chiziqdan ham foydalanishingiz mumkin:

*textView.setGravity(Gravity.BOTTOM | Gravity.CENTER);*

# III-BOB. ASOSIY BOSHQARUV ELEMENTLARI

## 3.1. TextView elementi

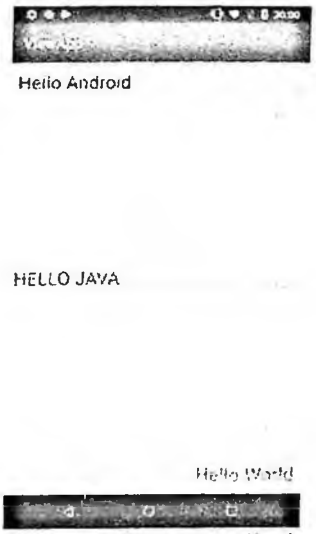
**TextView** elementi matnni ekranda oddiy namoyish qilish uchun mo'ljallangan. Bu shunchaki matnni tahrirlash qobiliyatisiz ko'rsatadi. Uning ba'zi bir asosiy xususiyatlari:

- **android:text**: element matnini o'rnatadi
- **android:textSize**: matn balandligini belgilaydi, sp balandlik uchun birlik sifatida ishlatiladi
- **android:background**: elementning fon rangini o'n oltinchi belgida rang sifatida yoki rang manbai sifatida belgilaydi
- **android:textColor**: matn rangini belgilaydi
- **android:textAllCaps**: agar true bo'lsa, matndagi barcha belgilarni katta harflar bilan yozadi
- **android:textDirection**: matn yo'nalishini belgilaydi. Odatiy yo'nalish chapdan o'ngga, lekin **rtl** qiymati yo'nalishni o'ngdan chapga o'rnatish uchun ishlatilishi mumkin
- **android:textAlignment**: Matnning hizalanishini belgilaydi. U quyidagi qiymatlarni olishi mumkin:
  - **center**: markazga tekislash
  - **textStart**: chapga tekislangan
  - **textEnd**: o'ng tomonga tekislangan
  - **viewStart**: matn chapdan o'ngga yo'naltirilganda, chapga, o'ngdan chapga yo'naltirilganda, o'ngga tekislanadi
  - **viewEnd**: matn chapdan o'ngga yo'naltirilganda, o'ngga, matn o'ngdan chapga yo'naltirilganda, chapga tekislanadi
- **android:fontFamily**: shrift turini o'rnatadi. U quyidagi qiymatlarni olishi mumkin:
  - **monospace**
  - **serif**
  - **serif-monospace**
  - **sans-serif**
  - **sans-serif-quyultirilgan**
  - **sans-serif-smallcaps**
  - **sans-serif-nur**
  - **casual**
  - **cursive**
  - **cursive**

Masalan, uchta matn maydonini aniqlaymiz:

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

```
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:layout_margin="10dp"
        android:text="Hello Android "
        android:fontFamily="sans-serif"
        android:textSize="26sp"
        android:background="#ffebee"
        android:textColor="#f44336"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:layout_margin="10dp"
        android:text="Hello Java"
        android:textAllCaps="true"
        android:textSize="26sp"
        android:background="#ede7f6"
        android:textColor="#7e57c2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:layout_margin="10dp"
        android:text="Hello World"
        android:textAlignment="textEnd"
        android:textSize="26sp"
        android:background="#e8eaf6"
        android:textColor="#5c6bc0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```



3.1-rasm. Dastur natijasi

Elementni kodga o'rnatish ham sodda. Masalan, element yarataylik va un
namoyish qilaylik:

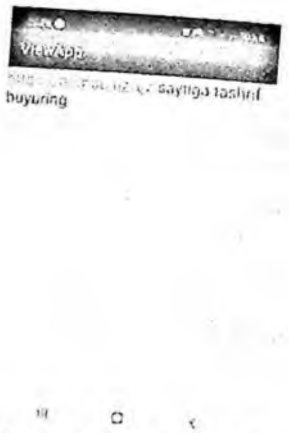
```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.graphics.Typeface;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      ConstraintLayout constraintLayout = new ConstraintLayout(this);
      TextView textView = new TextView(this);
      // fon rangini o`rnatish
      textView.setBackgroundColor(0xffe8eaf6);
      // matn rangini o`rnatish
      textView.setTextColor(0xff5c6bc0);
      // barcha harflarni katta harflar bilan yozing
      textView.setAllCaps(true);
      // matnni tekislashni markazga qo'ying
      textView.setTextAlignment(TextView.TEXT_ALIGNMENT_CENTER
      // matnni o'rnating
      textView.setText("Hello Android!");
      // shriftni o`rnatish
      textView.setTypeface(Typeface.create("casual", Typeface.NORMAL)
      // matn balandligini o'rnating
      textView.setTextSize(26);
```

```
            ConstraintLayout.LayoutParams layoutParams = new
ConstraintLayout.LayoutParams
                (ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
            layoutParams.leftToLeft =
ConstraintLayout.LayoutParams.PARENT_ID;
            layoutParams.topToTop =
ConstraintLayout.LayoutParams.PARENT_ID;
            textView.setLayoutParams(layoutParams);
            constraintLayout.addView(textView);
            setContentView(constraintLayout);
        }
    }
```

3.2-rasm. Dastur natijasi

Ba'zan ekranda biron bir harakat amalga oshirilishini bosish orqali havolani yoki telefonni ko'rsatish kerak bo'ladi. Buning uchun **android: autoLink** atributi **TextView**da aniqlanadi:

```
<TextView
  android:text="https://namdu.uz/uz saytiga tashrif buyuring "
  android:textSize="21sp"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:autoLink="web|email"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintTop_toTopOf="parent"/>
```



[illegible] saytiga tashrif buyuring

3.3-rasm. Dastur natijasi

**android: autoLink** bir nechta qiymatlarni olishi mumkin:

- **none**: barcha havolalarni o'chiradi
- **web**: barcha veb-havolalarni o'z ichiga oladi
- **email**: elektron pochta manzillariga havolalarni o'z ichiga oladi
- **phone**: telefon raqamlariga havolalarni o'z ichiga oladi
- **map**: xaritaga havolalarni o'z ichiga oladi
- **all**: yuqoridagi barcha havolalarni o'z ichiga oladi

Ya'ni, android: **autoLink = "web"** ni o'rnatishda, agar matnda url-manzil haqida so'z bor bo'lsa, unda ushbu manzil ajratib ko'rsatiladi va uni bosganingizda veb-brauzerga o'tasiz, u ushbu manzildagi sahifa. Oldinga chiziq bilan biz quyidagi holatdagi kabi shartlarni birlashtira olamiz: **android: autoLink = "web | email"**

## 3.2. EditText elementi

**EditText** elementi **TextView** sinfining qism sinfidir. Shuningdek, u matn maydonini taqdim etadi, ammo endi matnni kiritish va tahrirlash imkoniyati mavjud. Shunday qilib, **EditText**da biz **TextView**dagi kabi barcha xususiyatlardan foydalanishimiz mumkin.

**TextView** mavzusida yoritilmagan atributlardan **android:hint** atributiga e'tibor qaratish lozim. Agar u **EditText** elementi bo'sh bo'lsa, ko'rsatma sifatida ko'rsatiladigan matnni o'rnatishga imkon beradi. Bundan tashqari, biz kirish uchun klaviaturani o'rnatishga imkon beradigan **android:inputType** atributidan foydalanishimiz mumkin. Xususan, uning qadriyatlari orasida quyidagilarni ajratish mumkin:

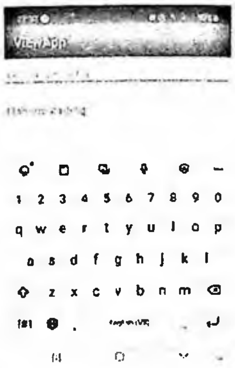
- **text**: bitta qatorli matnni kiritish uchun oddiy klaviatura
- **textMultiLine**: ko'p qatorli matn qutisi
- **textEmailAddress**: @ belgisiga ega oddiy klaviatura elektron pochtani kiritishga yo'naltirilgan
- **textUri**: Internet-manzillarni kiritishga yo'naltirilgan / belgili oddiy klaviatura
- **textPassword**: parolni kiritish uchun klaviatura
- **textCapWords**: matn terish paytida kiritilgan so'zning birinchi belgisi katta harf, qolganlari kichik harf

- **number**: raqamli klaviatura
- **phone**: telefon uslubidagi klaviatura
- **date**: sana kiritish uchun klaviatura
- **time**: vaqtni kiritish uchun klaviatura
- **datetime**: sana va vaqtni kiritish uchun klaviatura

**EditText**dan foydalanish:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <EditText
    android:id="@+id/name"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Введите имя"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintRight_toRightOf="parent"/>
  <EditText
    android:id="@+id/message"
    android:layout_marginTop="16dp"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:hint="Введите сообщение"
    android:inputType="textMultiLine"
    android:gravity="top"
    app:layout_constraintTop_toBottomOf="@+id/name"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu erdagi birinchi maydon odatdagi bir qator, ikkinchisi esa ko'p qatorli maydon. Matnni ikkinchi maydonda yuqoriga tekislash uchun **android:gravity="top"** atributi qo'shimcha ravishda o'rnatiladi.


3.4-rasm. Dastur natijasi

**EditText** elementining xususiyatlaridan biri, shuningdek, kiritilgan belgilarni foydalanuvchi kirishi bilan qayta ishlash qobiliyatidir. Buning uchun **activity_main.xml** faylida quyidagi belgini belgilang:

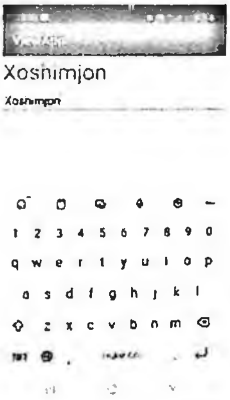
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/textView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:textSize="34sp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"/>
  <EditText
    android:id="@+id/editText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Введите имя"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**EditText**ga kiritilgan belgilar darhol **TextView** boshqaruvida paydo bo'ladi deb taxmin qilinadi. Buning uchun biz **MainActivity** kodini ham o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      EditText editText = (EditText) findViewById(R.id.editText);
      editText.addTextChangedListener(new TextWatcher() {
         public void afterTextChanged(Editable s) {}
         public void beforeTextChanged(CharSequence s, int start,
                                  int count, int after) {
         }
         public void onTextChanged(CharSequence s, int start, int before, int
count) {
            TextView textView = (TextView) findViewById(R.id.textView);
            textView.setText(s);
         }
      });
   }
}
```

**AddTextChangedListener()** usuli yordamida **EditText** elementiga matn kiritishni tinglovchi, **TextWatcher** ob'ekti qo'shiladi. Uni ishlatish uchun biz uchta usulni amalga oshirishimiz kerak, ammo aslida matn o'zgarganda chaqiriladigan **onTextChanged** usulini amalga oshirish biz uchun etarli. Kiritilgan matn bu usulga **CharSequence** parametri sifatida uzatiladi. Usulning o'zida biz shunchaki bu matnni **TextView** boshqaruviga o'tkazamiz.

Natijada, **EditText**ga yozishda barcha belgilar **TextView**da ham ko'rsatiladi:



3.5-rasm. Dastur natijasi

## 3.3. Button elementi

Odatda ishlatiladigan elementlardan biri tugmalar bo'lib, ular **android.widget.Button** sinfi bilan ifodalanadi. Tugmalarning asosiy xususiyati bu bosish orqali foydalanuvchi bilan o'zaro aloqada bo'lish qobiliyatidir.

Tugmalarga o'rnatilishi mumkin bo'lgan ba'zi bir asosiy atributlar:

- **text**: matnni tugmachaga o'rnatadi
- **textColor**: tugma ustiga matn rangini o'rnatadi
- **background**: tugmaning fon rangini belgilaydi
- **textAllCaps**: true-ga o'rnatilganda, matnni katta harflar bilan o'rnatadi. Odatiy bo'lib, true qiymati ishlatiladi.
- **onClick**: tugmachani bosish moslamasini o'rnatadi

Shunday qilib, kodni **activity_main.xml** da quyidagicha o'zgartiraylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textSize="34sp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
    <EditText
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Введите имя"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Kiritish"
        android:onClick="sendMessage"
        app:layout_constraintTop_toBottomOf="@+id/editText"
        app:layout_constraintLeft_toLeftOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**android:onClick** atributi yordamida **java** kodida tugmani bosish bilan ishlaydigan usulni o'rnatishingiz mumkin. Shunday qilib, yuqoridagi misolda bu

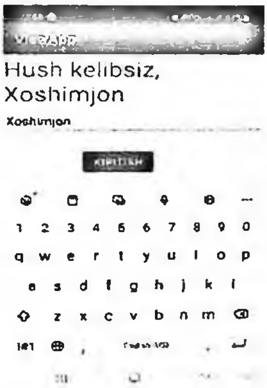
**sendMessage** usuli. Endi **MainActivity** kodiga o'tamiz va unda quyidagi usulni yozamiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);  }
   // Tugmani bosish bilan ishlov berish
   public void sendMessage(View view) {
      TextView textView = (TextView) findViewById(R.id.textView);
      EditText editText = (EditText) findViewById(R.id.editText);
      textView.setText("Hush kelibsiz, " + editText.getText());  }
}
```

Xitni boshqarish usulini yaratishda quyidagi fikrlarni ko'rib chiqing:

- Usul **public** modifikatori bilan e'lon qilinishi kerak
- **void** qiymatini qaytarishi kerak
- **View** ob'ektini parametr sifatida oling. Ushbu **View** ob'ekti bosilgan tugmani aks ettiradi

Bunday holda, tugmachani bosgandan so'ng, **TextView**da **EditText**dan matn ko'rsatiladi.


Hush kelibsiz,
Xoshimjon

3.6-rasm. Dastur natijasi

Shunga o'xshash misol **MainActivity** kodida to'liq:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.view.View;
```

```
    import android.widget.Button;
    import android.widget.EditText;
    import android.widget.TextView;
    public class MainActivity extends AppCompatActivity {
        EditText editText;
        TextView textView;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            //setContentView(R.layout.activity_main);
            ConstraintLayout constraintLayout = new ConstraintLayout(this);
            textView = new TextView(this);
            textView.setId(View.generateViewId());
            ConstraintLayout.LayoutParams      textViewLayout      =      new
ConstraintLayout.LayoutParams(
                    ConstraintLayout.LayoutParams.MATCH_CONSTRAINT,
ConstraintLayout.LayoutParams.WRAP_CONTENT
            );
            textViewLayout.topToTop                                        =
ConstraintLayout.LayoutParams.PARENT_ID;
            textViewLayout.leftToLeft                                      =
ConstraintLayout.LayoutParams.PARENT_ID;
            textViewLayout.rightToRight                                    =
ConstraintLayout.LayoutParams.PARENT_ID;
            textView.setLayoutParams(textViewLayout);
            constraintLayout.addView(textView);
            editText = new EditText(this);
            editText.setId(View.generateViewId());
            editText.setHint("Введите имя");
            ConstraintLayout.LayoutParams      editTextLayout      =      new
ConstraintLayout.LayoutParams(
                    ConstraintLayout.LayoutParams.MATCH_CONSTRAINT.
ConstraintLayout.LayoutParams.WRAP_CONTENT    );
            editTextLayout.topToBottom = textView.getId();
            editTextLayout.leftToLeft                                      =
ConstraintLayout.LayoutParams.PARENT_ID;
            editTextLayout.rightToRight                                    =
ConstraintLayout.LayoutParams.PARENT_ID;
            editText.setLayoutParams(editTextLayout);
            constraintLayout.addView(editText);
            Button button = new Button(this);
            button.setText("Kiritish");
            ConstraintLayout.LayoutParams      buttonLayout      =      new
ConstraintLayout.LayoutParams(
                    ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT  );
            buttonLayout.topToBottom = editText.getId();
            buttonLayout.leftToLeft
ConstraintLayout.LayoutParams.PARENT_ID;
            button.setLayoutParams(buttonLayout);
            constraintLayout.addView(button);
            button.setOnClickListener(new View.OnClickListener() {
              public void onClick(View v) {
                // Bosilganda qayta ishlash
                textView.setText("Hush kelibsiz, " + editText.getText());
              }
            });
            setContentView(constraintLayout);
        }
    }
```

Tugmachani dasturiy ravishda yaratishda biz uni tinglash uchun **View.OnClickListener**ni aniqlay olamiz va bosish bilan ishlash uchun uning **onClick** usulidan foydalanamiz:

```
button.setOnClickListener(new View.OnClickListener() {
  public void onClick(View v) {
    // Chertishni boshqarish
  }
});
```

## 3.4. Kalkulyator dasturi

Tartibning ba'zi asoslarini va **TextView**, **EditText** va **Button** kabi elementlarni bilib, siz allaqachon ozmi-ko'pmi to'liq dastur yaratishingiz mumkin. Bunday holda biz oddiy kalkulyatorni qilamiz.

Buning uchun yangi loyiha yarating va **activity_main.xml** faylida quyidagi interfeysni aniqlang:

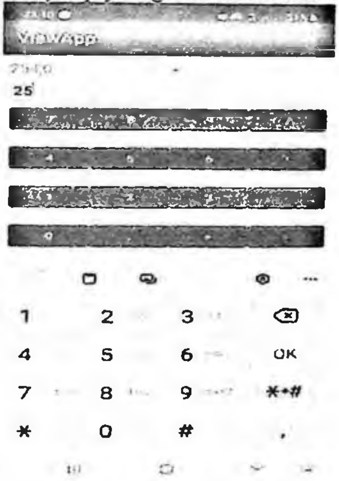
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="8dp">
<!-- natija maydoni -->
  <TextView
    android:id="@+id/resultField"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintHorizontal_weight="1"
    android:textSize="18sp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toLeftOf="@+id/operationField"/>
<!-- operatsiya belgisi maydoni -->
<TextView
    android:id="@+id/operationField"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintHorizontal_weight="1"
    android:textSize="18sp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toRightOf="@+id/resultField"
    />
<!-- raqam kiritish maydoni -->
<EditText
    android:id="@+id/numberField"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:inputType="phone"
    app:layout_constraintTop_toBottomOf="@+id/resultField"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"/>
<LinearLayout
    android:id="@+id/firstButtonPanel"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    app:layout_constraintTop_toBottomOf="@+id/numberField"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent">
    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="7"
        android:onClick="onNumberClick"/>
    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="8"
        android:onClick="onNumberClick"/>
    <Button
        android:layout_weight="1"
```

```
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:text="9"
      android:onClick="onNumberClick"/>
    <Button
      android:layout_weight="1"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:text="/"
      android:onClick="onOperationClick"/>
  </LinearLayout>
  <LinearLayout
    android:id="@+id/secondButtonPanel"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    app:layout_constraintTop_toBottomOf="@+id/firstButtonPanel"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent">
    <Button
      android:layout_weight="1"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:text="4"
      android:onClick="onNumberClick"/>
    <Button
      android:layout_weight="1"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:text="5"
      android:onClick="onNumberClick"/>
    <Button
      android:layout_weight="1"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:text="6"
      android:onClick="onNumberClick"/>
    <Button
      android:layout_weight="1"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:text="*"
      android:onClick="onOperationClick"/>
  </LinearLayout>
  <LinearLayout
```

```
android:id="@+id/thirdButtonPanel"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginTop="16dp"
app:layout_constraintTop_toBottomOf="@+id/secondButtonPanel"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent">
<Button
  android:layout_weight="1"
  android:layout_width="0dp"
  android:layout_height="wrap_content"
  android:text="1"
  android:onClick="onNumberClick"/>
<Button
  android:layout_weight="1"
  android:layout_width="0dp"
  android:layout_height="wrap_content"
  android:text="2"
  android:onClick="onNumberClick"/>
<Button
  android:layout_weight="1"
  android:layout_width="0dp"
  android:layout_height="wrap_content"
  android:text="3"
  android:onClick="onNumberClick"/>
<Button
  android:layout_weight="1"
  android:layout_width="0dp"
  android:layout_height="wrap_content"
  android:text="-"
  android:onClick="onOperationClick"/>
</LinearLayout>
<LinearLayout
  android:id="@+id/forthButtonPanel"
  android:layout_width="0dp"
  android:layout_height="wrap_content"
  android:layout_marginTop="16dp"
  app:layout_constraintTop_toBottomOf="@+id/thirdButtonPanel"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent">
  <Button
    android:layout_weight="1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="0"
```

```
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="."
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="+"
            android:onClick="onOperationClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="="
            android:onClick="onOperationClick"/>
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Natijada, butun interfeys quyidagicha ko'rinadi:



3.7-rasm. Dastur natijasi

Ildizni joylashtiruvchi konteyner **ConstraintLayout** elementini aks ettiradi. Uning yuqori qismida ikkita **TextView** matn maydonlari aniqlangan: bittasi hisob-kitoblar natijasini ko'rsatish uchun, ikkinchisi esa amaldagi belgini ko'rsatish uchun.

Keyin raqamlarni kiritish uchun **EditText** elementi keladi.

Va keyin tugmachalarning gorizontal qatorlari bo'lgan to'rtta LineerLayout elementlari mavjud. Barcha tugmalar konteyner ichida teng joy egallashi uchun ularga **android:layout_weight="1"** va **android:layout_width="0dp"** atributlari o'rnatilgan.

Bundan tashqari, raqamli tugmalar uchun **onNumberClick** usuli bosish moslamasi sifatida o'rnatiladi va operatsion belgilari bo'lgan tugmalar uchun **onClick** atributi **onOperationClick** uslubiga ishora qiladi.

Endi **MainActivity** sinfini o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   TextView resultField; // natija chiqarish uchun matn maydoni
   EditText numberField;  // raqam kiritish maydoni
   TextView operationField;    // operatsiya belgisini ko'rsatish uchun matn
maydoni
   Double operand = null;  // operatsiya operandasi
   String lastOperation = "="; // oxirgi amallar
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // id bo'yicha barcha maydonlarni activity_main.xml-dan oling
      resultField =(TextView) findViewById(R.id.resultField);
      numberField = (EditText) findViewById(R.id.numberField);
      operationField = (TextView) findViewById(R.id.operationField);
   }
   // vaziyatni saqlab qolish
   @Override
   protected void onSaveInstanceState(Bundle outState) {
      outState.putString("OPERATION", lastOperation);
      if(operand!=null)
         outState.putDouble("OPERAND", operand);
      super.onSaveInstanceState(outState);
   }
   // ilgari saqlangan holatni qaytarib olish
   @Override
   protected void onRestoreInstanceState(Bundle savedInstanceState) {
      super.onRestoreInstanceState(savedInstanceState);
      lastOperation = savedInstanceState.getString("OPERATION");
      operand= savedInstanceState.getDouble("OPERAND");
      resultField.setText(operand.toString());
      operationField.setText(lastOperation);
```

```
    }
    // raqamli tugmani bosish bilan ishlash
    public void onNumberClick(View view){
        Button button = (Button)view;
        numberField.append(button.getText());
        if(lastOperation.equals("=") && operand!=null){
            operand = null;
        }
    }
    // operatsion tugmachasini bosish bilan ishlash
    public void onOperationClick(View view){
        Button button = (Button)view;
        String op = button.getText().toString();
        String number = numberField.getText().toString();
        // если введенно что-нибудь
        if(number.length()>0){
            number = number.replace(',', '.');
            try{
                performOperation(Double.valueOf(number), op);
            }catch (NumberFormatException ex){
                numberField.setText("");
            }
        }
        lastOperation = op;
        operationField.setText(lastOperation);
    }
    private void performOperation(Double number, String operation){
        // agar operand ilgari o'rnatilmagan bo'lsa (birinchi operatsiyani
kiritishda)
        if(operand ==null){
            operand = number;
        }
        else{
            if(lastOperation.equals("=")){
                lastOperation = operation;
            }
            switch(lastOperation){
                case "=":
                    operand =number;
                    break;
                case "/":
                    if(number==0){
                        operand =0.0;
                    }
                    else{
```

```
                    operand /=number;
                }
                break;
            case "*":
                operand *=number;
                break;
            case "+":
                operand +=number;
                break;
            case "-":
                operand -=number;
                break;
        }
    }
    resultField.setText(operand.toString().replace('.', ','));
    numberField.setText("");
  }
}
```

Keling, ushbu kodni tahlil qilaylik. Birinchidan, **onCreate()** usulida biz barcha maydonlarni **activity_main.xml** dan olamiz, ularning matni o'zgaradi:

```
resultField =(TextView) findViewById(R.id.resultField);
numberField = (EditText) findViewById(R.id.numberField);
operationField = (TextView) findViewById(R.id.operationField);
```

Amaliyot natijasi **Double** turini ifodalovchi **operand** o'zgaruvchisiga o'tadi va operatsiya belgisi **lastOperation** o'zgaruvchiga o'tadi:

```
Double operand = null;
String lastOperation = "=";
```

Aksincha manzara, yoki aksincha, uchun vertikal kommutatsiya qachon buyon, biz kiritilgan barcha ma'lumotlar yo'qotishi mumkin, shuning uchun, deb emas, balki, uni yo'qotish, biz **onSaveInstanceState()** usuli uni saqlash va orqa **onRestoreInstanceState()** usuli uni olish.

Raqamli tugmachani bosganingizda **onNumberClick** usuli chaqiriladi, biz kiritilgan maydon yoki vergulni numberField maydoniga matnga qo'shamiz:

```
Button button = (Button)view;
numberField.append(button.getText());
if(lastOperation.equals("=") && operand!=null){
    operand = null;
}
```

Bundan tashqari, agar oxirgi operatsiya natija bo'lsa (tenglik belgisi), biz operand o'zgaruvchisini tiklaymiz.

**OnOperationClick** usulida operatsiya belgisi bo'lgan tugmani bosish qayta ishlanadi:

```
Button button = (Button)view;
String op = button.getText().toString();
String number = numberField.getText().toString();
```

```
if(number.length()>0){
  number = number.replace(',', '.');
  try{
    performOperation(Double.valueOf(number), op);
  }catch (NumberFormatException ex){
    numberField.setText("");
  }
}
lastOperation = op;
operationField.setText(lastOperation);
```

Bu yerda biz ilgari kiritilgan raqamni va kiritilgan operatsiyani olamiz va ularni **performOperation()** uslubiga o'tkazamiz. Usul shunchaki mag'lubiyat emas, balki Ikkala raqam orqali uzatilganligi sababli, biz satrni chlo ga aylantirishimiz kerak. Va nazariy jihatdan raqamli bo'lmagan belgilar kiritilishi mumkinligi sababli, konvertatsiya paytida yuzaga kelishi mumkin bo'lgan istisnoga erishish uchun **try...catch** konstruktsiyasi ishlatiladi.

Bundan tashqari, **javadagi Doubledagi** butun son va kasr qismlarining ajratuvchisi nuqta bo'lganligi sababli, vergulni nuqta bilan almashtirishimiz kerak, chunki biz vergulni ajratuvchi sifatida ishlatamiz deb taxmin qilinadi.

Va **performOperation()** usulida biz haqiqiy operatsiyani bajaramiz. Birinchi operatsiyaga kirganimizda, operand hali o'rnatilmagan bo'lsa, biz shunchaki operandni o'rnatamiz:

```
if(operand ==null){
  operand = number;
}
```

Ikkinchi va keyingi operatsiyalarni kiritishda biz **operand** operandiga va sonli maydonga kiritilgan ikkinchi raqamga **lastOperation** o'zgaruvchisida saqlanadigan oldingi amalni qo'llaymiz. Olingan operatsiya natijasini **operand** o'zgaruvchisiga saqlang.

## 3.5. Qalqib chiquvchi oynalar. Toast

Oddiy bildirishnomalarni yaratish uchun **Android Toast** sinfidan foydalanadi. Aslida, **Toast** bir muncha vaqt ko'rsatiladigan ba'zi bir matnli qalqib chiquvchi menyuni taqdim etadi.

**Toast** ob'ekti **xml** formatlash kodida, masalan, **activity_main.xml**da yaratib bo'lmaydi. **Toast**dan faqat **java** kodida foydalanish mumkin.

Shunday qilib, **activity_main.xml** formatlash faylidagi tugmani aniqlaylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp">
  <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click"
        android:onClick="onClick"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

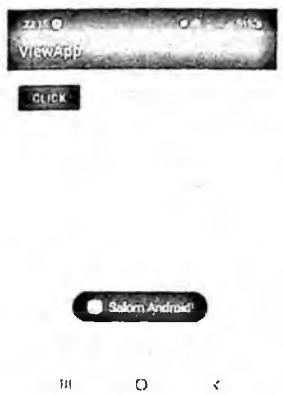
Tugmachani bosish moslamasi mavjud - **onClick** usuli. Buni **MainActivity** kodida aniqlaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import android.view.Gravity;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  public  void onClick(View view){
    Toast        toast        =        Toast.makeText(this,        "Salom
Android!",Toast.LENGTH_LONG);
    toast.show();
  }
}
```

Ishlov beruvchida qalqib chiquvchi menyu ko'rsatiladi. Uni yaratish uchun **Toast.makeText()** usuli qo'llaniladi, unga uchta parametr: joriy kontekst (joriy **activity** ob'ekti), ko'rsatilgan matn va oynani ko'rsatish vaqti beriladi.

Oynani ko'rsatish vaqti sifatida biz butun son qiymatidan foydalanishimiz mumkin - millisekundalar soni yoki o'rnatilgan **Toast.LENGTH_LONG** (3500 millisekundalar) va **Toast.LENGTH_SHORT** (2000 millisekundlar).

Oynaning o'zini ko'rsatish uchun **show()** usuli deyiladi:



3.8-rasm. Dastur natijasi

Odatiy bo'lib, oyna interfeysning pastki qismida, o'rtada markazda ko'rsatiladi. Lekin **setGravity()** va **setMargin()** usullari yordamida oynaning joylashishini sozlashimiz mumkin. Shunday qilib, **onClick** usulini o'zgartiraylik:

```
public void onClick(View view){
    Toast toast = Toast.makeText(this, "Salom Android!",
Toast.LENGTH_LONG);
    toast.setGravity(Gravity.TOP, 0,160);
    toast.show();
}
```

**setGravity** usulining birinchi parametri **Toast** konteynerining qaysi qismiga joylashtirilishini belgilaydi, ikkinchi va uchinchi parametrlar mos ravishda ushbu holatdan gorizontal va vertikal chekkalarni o'rnatadi:

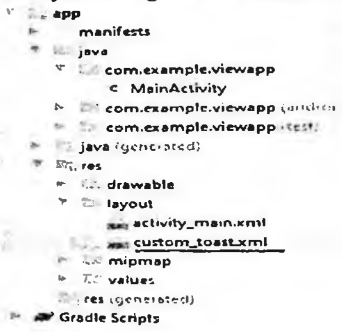
3.9-rasm. Dastur natijasi

**setMargin()** usuli ikkita parametrni oladi: konteynerning chap chegarasidan chegara konteynerning kengligidan foizga va yuqori chegaradan konteyner uzunligidan foizga.

**Toast uchun interfeysni aniqlash.** Android qalqib chiquvchi menyularning vizual interfeysini sozlashni qo'llab-quvvatlaydi. Buning uchun **res/layout** papkasida yangi **layout custom_toast.xml** faylini loyihaga qo'shing:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
```

```
android:id="@+id/toast_layout"
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#555"
android:padding="16dp"
>
<TextView android:id="@+id/text"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textColor="#81C784"
   android:textSize="30sp"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintRight_toRightOf="parent"
   app:layout_constraintTop_toTopOf="parent"
   app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



```
app
    manifests
    java
        com.example.viewapp
            MainActivity
        com.example.viewapp (androidTest)
        com.example.viewapp (test)
    java (generated)
    res
        drawable
        layout
            activity_main.xml
            custom_toast.xml
        mipmap
        values
    res (generated)
Gradle Scripts
```

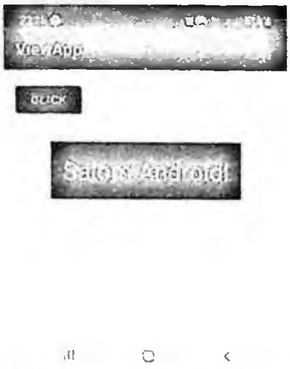
3.10-rasm. Loyiha strukturasi

**MainActivity** kodida **Toast** displeyini o'rnatamiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
```

```
    }
    public  void onClick(View view){
        LayoutInflater inflater = getLayoutInflater();
        View layout = inflater.inflate(R.layout.custom_toast.
                (ViewGroup) findViewById(R.id.toast_layout));
        TextView text = (TextView) layout.findViewById(R.id.text);
        text.setText("Hello Android!");
        Toast toast = new Toast(getApplicationContext());
        toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
        toast.setDuration(Toast.LENGTH_LONG);
        toast.setView(layout);
        toast.show();
    }
}
```

Bu yerda **LayoutInflater** ob'ekti yordamida tugmani bosish orqali biz **custom_toast.xml** faylidan yuqorida belgilangan interfeysni yuklaymiz. Keyin biz oynaning displeyini, uning matnini o'matamiz va ekranning markazida qalqib chiquvchi menyu oynasini ko'rsatamiz.



3.11-rasm. Dastur natijasi

## 3.6. Snackbar elementi

**Snackbar** elementi **Toast**ga o'xshashdir: shuningdek, qalqib chiquvchi menyu xabarlarni ko'rsatishga imkon beradi, ammo endi xabarlar ekranning kengligiga mos ravishda uzatiladi.

**Snackbar**dan foydalanish uchun, **Snackbar** paydo bo'ladigan tugmani bosib, **activity_main.xml** fayliga tugma ta'rifini qo'shing:

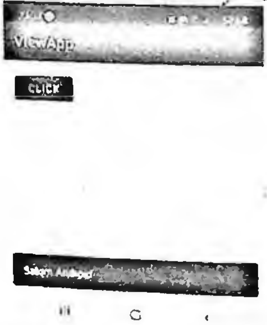
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click"
        android:onClick="onClick"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Tugma bu yerda, xabar ko'rsatiladigan joyga bosish orqali aniqlanadi.
Va shuningdek, **MainActivity** sinfini o'zgartiring:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import com.google.android.material.snackbar.Snackbar;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public  void onClick(View view){
        Snackbar.make(view, "Hello Android", Snackbar.LENGTH_LONG)
            .show();
    }
}
```

**Snackbar make()** usuli yordamida yaratiladi, u uchta parametrdan iborat: ochilgan xabar biriktirilgan **View** ob'ekti, xabarning o'zi satr sifatida va xabar qancha vaqt ko'rsatilishini ko'rsatuvchi parametr. Oxirgi parametr raqamli qiymat bo'lishi mumkin - millisekundlar soni yoki uchta konstantadan biri: **Snackbar.LENGTH_INDEFINITE** (noma'lum vaqt davomida ko'rsatish), **Snackbar.LENGTH_LONG** (uzoq aks ettirish) yoki **Snackbar.LENGTH_SHORT**(qisqa aks ettirish).

Yaratgandan so'ng, **Snackbar show** usuli yordamida ko'rsatiladi:



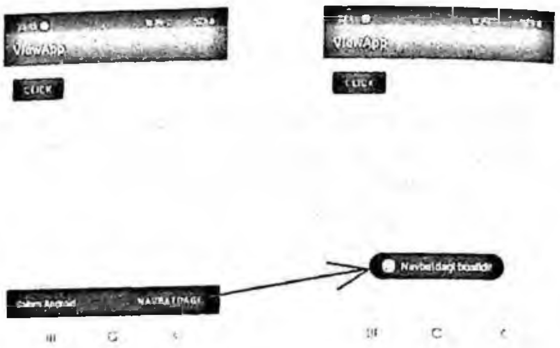
3.12-rasm. Dastur natijasi

Shu bilan birga, **Toast**dan farqli o'laroq, biz xabarning joylashuviga ta'sir qila olmaymiz, u ekranning pastki qismida ko'rsatiladi va butun pastki qismini egallaydi.

**Voqealar boshqaruvchisini biriktirish. Snackbar** foydalanuvchi xabarga qandaydir munosabat bildirishi uchun vidjetga harakat qo'shishga imkon beradi. Masalan, **MainActivity** kodini quyidagicha o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import com.google.android.material.snackbar.Snackbar;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public  void onClick(View view){
      Snackbar snackbar = Snackbar.make(view, "Salom   Android",
Snackbar.LENGTH_LONG);
         snackbar.setAction("Navbatdagi...", new View.OnClickListener (){
            @Override
            public void onClick(View v) {
               Toast toast = Toast.makeText(getApplicationContext(), " Keyingi
bosildi!",Toast.LENGTH_LONG);
               toast.show();
            }
         });
      snackbar.show();
   }
}
```

**Snackbar** harakat qo'shish uchun **setAction()** usulidan foydalanadi. Birinchi parametr foydalanuvchi bosishi mumkin bo'lgan xabardagi tugma matnini aks ettiradi - bu holda **"Next ..."**. Ikkinchi parametr **View.OnClickListener** interfeysining bajarilishini anglatadi (xuddi shu tugmani bosish uchun ishlatiladi). **OnClick()** usulida biz aslida xabardagi tugma bosilganda chaqiriladigan amallarni bajaramiz. Bunday holda, soddalik uchun biz shunchaki tostni **Toast** ob'ekti sifatida namoyish etamiz.



3.13-rasm. Dastur natijasi

**Vizual ko'rinishni sozlash. Snackbar**ning bir qator usullari tashqi ko'rinishini sozlash imkonini beradi:

- **setTextColor()**: matn rangini belgilaydi
- **setBackgroundTint()**: fon rangini belgilaydi
- **setActionTextColor()**: Qalqib chiquvchi xabarda tugma matnining rangini belgilaydi

```
snackbar.setTextColor(0XFF81C784);
snackbar.setBackgroundTint(0XFF555555);
snackbar.setActionTextColor(0XFF0277BD);
```

## 3.7. Checkbox elementi

**Checkbox** elementlari - belgilanadigan yoki belgilanmagan qutilar. Belgilash katakchalari bir nechta qiymatlardan bir nechta tanlovga imkon beradi. Shunday qilib, **activity_main.xml** formatlash faylida **CheckBox** elementini aniqlaymiz:

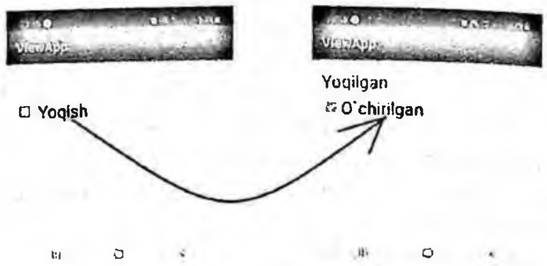
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp">
    <TextView android:id="@+id/selection"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="26sp"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent"/>
    <CheckBox android:id="@+id/enabled"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=" Yoqish "
      android:textSize="26sp"
```

```
        android:onClick="onCheckboxClicked"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/selection"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

**android:onClick** atributi, oddiy tugmachalarda bo'lgani kabi, katakchani bosish uchun ishlov beruvchini o'rnatishga imkon beradi. **MainActivity** kodida klik ishlov beruvchisini aniqlaylik:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  public void onCheckboxClicked(View view) {
    // Biz katakchani olamiz
    CheckBox checkBox = (CheckBox) view;
    TextView selection = (TextView) findViewById(R.id.selection);
    // Agar berilgan katakcha belgilansa oling
    if(checkBox.isChecked()) {
      selection.setText("Yoqilgan");
      checkBox.setText("O`chirilgan");
    }
    else {
      selection.setText("Yoqilgan");
      checkBox.setText("O`chirilgan");
    }
  }
}
```

Bosilgan tasdiqlash qutisi parametr sifatida **onCheckboxClicked** bosish ishlov beruvchisiga uzatiladi. Tekshirish moslamasi har **checkBox** tugmachasini bosganda ishga tushiriladi. Ya'ni, katakchani belgilaganimizda ham, katakchani olib tashlaganimizda ham. **IsChecked()** usuli yordamida tasdiqlash qutisi tanlanganligini bilib olishingiz mumkin - bu holda usul **true** qiymatini qaytaradi.



3.14-rasm. Dastur natijasi

Shu kabi bir nechta katakchalardan foydalanish mumkin:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp">
  <TextView android:id="@+id/selection"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="26sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
  <CheckBox android:id="@+id/java"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Java"
    android:textSize="26sp"
    android:onClick="onCheckboxClicked"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/selection"/>
  <CheckBox
    android:id="@+id/kotlin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="4dp"
    android:layout_marginTop="24dp"
    android:onClick="onCheckboxClicked"
    android:text="Kotlin"
    android:textSize="26sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/java" />
  <CheckBox
    android:id="@+id/cplus"
    android:layout_width="wrap_content"
```

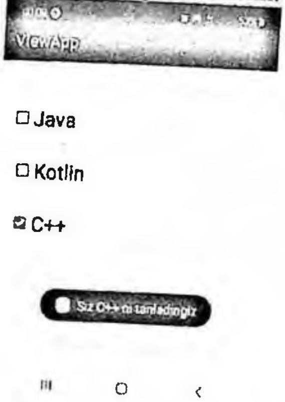
```
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            android:layout_marginTop="24dp"
            android:onClick="onCheckboxClicked"
            android:text="C++"
            android:textSize="26sp"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/kotlin" />
    </androidx.constraintlayout.widget.ConstraintLayout>
```

Har bir katakchada o'z chertish moslamasi bo'lishi mumkin. Yoki bu holatda bo'lgani kabi, siz ham qilishingiz mumkin. Bunday holda, biz **switch...case** konstruktsiyasi yordamida **java** kodidagi bir nechta katakchalarni boshqarishimiz mumkin

```
    package com.example.viewapp;
    import androidx.appcompat.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.CheckBox;
    import android.widget.TextView;
    import android.widget.Toast;
    public class MainActivity extends AppCompatActivity {
      @Override
      protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
      }
      public void onCheckboxClicked(View view) {
        // Bayroqchani olamiz
        CheckBox checkBox = (CheckBox) view;
        // Agar berilgan katakcha belgilansa oling
        boolean checked = checkBox.isChecked();
        TextView selection = (TextView) findViewById(R.id.selection);
        // Qaysi qutilar belgilanganligini ko'rib chiqamiz
        switch(view.getId()) {
          case R.id.java:
            if (checked)
              Toast.makeText(this,      "Siz      Java      ni      tanladingiz",
Toast.LENGTH_LONG).show();
            break;
          case R.id.kotlin:
            if (checked)
              Toast.makeText(this,"Siz      Kotlin      ni      tanladingiz",
Toast.LENGTH_LONG).show();
            break;
          case R.id.cplus:
            if (checked)
```

```
                    Toast.makeText(this, "Siz        C++        ni        tanladingiz"
Toast.LENGTH_LONG).show();
                    break;
                default:
                    selection.setText("");
            }
        }
    }
```

**switch...case** konstruktsiyasidan foydalanib, siz bosilgan bayroqning idni olishingiz va tegishli amallarni bajarishingiz mumkin.



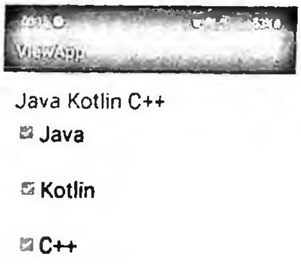
3.15-rasm. Dastur natijasi

To'g'ri, agar biz faqat tanlangan katakchadan matnni olishimiz kerak bo'lsa, unda **switch** konstruktsiyasidan foydalanish shart emas, chunki biz butun kodni quyidagicha qisqartiramiz:

```
public void onCheckboxClicked(View view) {
    // Biz katakchani olamiz
    CheckBox language = (CheckBox) view;
    // Agar berilgan katakcha belgilansa oling
    TextView selection = (TextView) findViewById(R.id.selection);
    if(language.isChecked())
        selection.setText(language.getText());
}
```

Biroq, bu holda, muammo qoladi: matn maydonida faqat bitta tanlangan element ko'rsatiladi. Ikkala tanlangan elementni ko'rsatish uchun **MainActivity** kodini o'zgartiraylik:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onCheckboxClicked(View view) {
        // Biz bayroqlarni olamiz
        CheckBox java = (CheckBox) findViewById(R.id.java);
        CheckBox kotlin = (CheckBox) findViewById(R.id.kotlin);
        String selectedItems = "";
        if(java.isChecked())
            selectedItems +=java.getText() + " ";
        if(kotlin.isChecked())
            selectedItems +=kotlin.getText();
        TextView selection = (TextView) findViewById(R.id selection);
        selection.setText(selectedItems);
    }
}
```


ViewApp

Java Kotlin C++

☑ Java

☑ Kotlin

☑ C++

3.16-rasm. Dastur natijasi

**OnCheckedChangeListener**. **OnCheckedChangeListener**dan foydalanish katakchaning o'zgarishini kuzatishning muqobil usulini taqdim etadi. Belgilash katagiga belgi qo'yganimizda yoki belgini olib tashlaganimizda, ushbu tinglovchi ishga tushiriladi. Masalan, quyidagi **checkbox**ni aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <TextView android:id="@+id/selection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

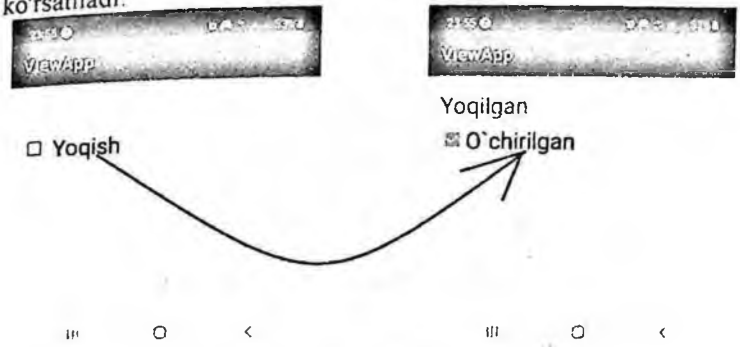
```
            android:textSize="26sp"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toTopOf="parent"/>
        <CheckBox android:id="@+id/enabled"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Yoqish"
            android:textSize="26sp"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/selection"/>
    </androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity kodida holatni o'zgartirish moslamasini ulang:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView selection = (TextView) findViewById(R.id.selection);
        CheckBox enableBox = (CheckBox) findViewById(R.id.enabled);
        enableBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
                if(isChecked) {
                    selection.setText("Yoqilgan");
                    buttonView.setText("O`chirilgan");
                }
                else {
                    selection.setText("Yoqilgan");
                    buttonView.setText("O`chirilgan");
                }
            }
        });
    }
}
```

**OnCheckedChangeListener** asosiy **CompoundButton** sinfida aniqlanadi va bitta **onCheckedChanged** usulini belgilaydi. Ushbu **buttonView** usulining birinchi parametri o'zgartirilgan **CheckBox** bayrog'ining o'zi. Va ikkinchi parametr **isChecked** katagiga belgi qo'yilganligini bildiradi.

Belgilash oynasining holati o'zgarganda, tegishli bildirishnoma ochilgan oynada ko'rsatiladi:



3.17-rasm. Dastur natijasi

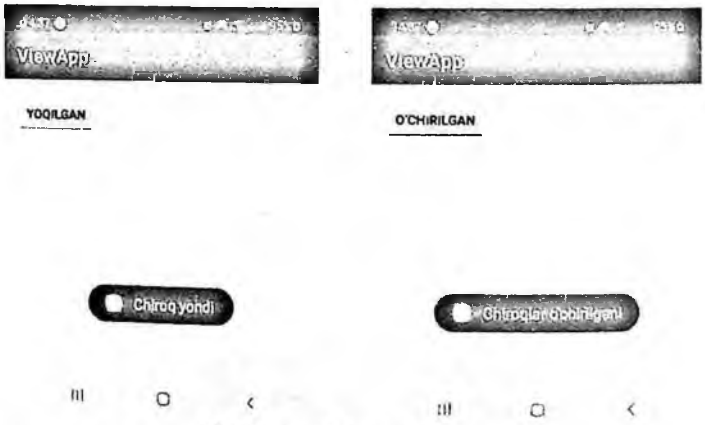
## 3.8. ToggleButton elementi

**ToggleButton, CheckBox** elementi singari, ikkita holatda bo'lishi mumkin: tekshirilgan va belgilanmagan va har bir holat uchun biz o'z matnimizni alohida o'rnatamiz. Masalan, quyidagi **ToggleButton** elementini aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp">
  <ToggleButton
    android:id="@+id/toggle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Yoqilgan"
    android:textOff="O'chirilgan"
    android:onClick="onToggleClicked"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**Android:textOn** va **android:textOff** atributlari tugma matnini mos ravishda belgilangan va belgilanmagan holatlarda o'rnatadi. Va boshqa tugmalar singari biz ham **onClick** hodisasi yordamida elementni bosish bilan shug'ullanishimiz mumkin. Bunday holda, biz **Activity** sinfida voqea ishlovchilarini aniqlaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import android.widget.ToggleButton;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void onToggleClicked(View view) {
      // tugmasi yoqilgan
      boolean on = ((ToggleButton) view).isChecked();
      if (on) {
         // yoqilgan bo'lsa, harakatlar
         Toast.makeText(this,          "          Chiroq          yondi          "
Toast.LENGTH_LONG).show();
      } else {
         // Agar o'chirib qo'yilgan bo'lsa, harakatlar
         Toast.makeText(this,          "          Chiroqlar          o'chirilgan!",
Toast.LENGTH_LONG).show();
      }
   }
}
```



3.18-rasm. Dastur natijasi

**Java** kodida **ToggleButton** elementini yaratish:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import android.widget.ToggleButton;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        ConstraintLayout layout = new ConstraintLayout(this);
        ConstraintLayout.LayoutParams        layoutParams        =        new
ConstraintLayout.LayoutParams
                (ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        ToggleButton toggleButton = new ToggleButton(this);
        toggleButton.setTextOff("O'chirilgan");
        toggleButton.setTextOn("Yoqilgan");
        toggleButton.setText("O'chirilgan");
        toggleButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                boolean on = ((ToggleButton) view).isChecked();
                if (on) {
                    Toast.makeText(getApplicationContext(),    "Chiroq    yondi",
Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(getApplicationContext(),      "      Chiroqlar
o'chirilgan!", Toast.LENGTH_LONG).show();
                }
            }
        });
        layoutParams.leftToLeft
ConstraintLayout.LayoutParams.PARENT_ID;
        layoutParams.topToTop
ConstraintLayout.LayoutParams.PARENT_ID;
        layout.addView(toggleButton);
        setContentView(layout);
    }
}
```

## 3.9. RadioButton elementi

Qutilarga o'xshash funktsionallik **RadioButton** sinfi tomonidan taqdim etiladigan radio tugmalari bilan ta'minlanadi. Ammo katakchalardan farqli o'laroq, biz radio tugmalari guruhida bir vaqtning o'zida faqat bitta radio tugmachani tanlashimiz mumkin.

Tanlash uchun radio tugmalar ro'yxatini yaratish uchun avval barcha radio tugmachalarni o'z ichiga olgan **RadioGroup** ob'ekti yaratilishi kerak:

```
<?xml version="1.0" encoding="utf-8"?>
```

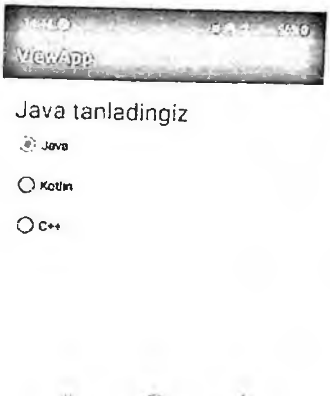
```
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp">
   <TextView android:id="@+id/selection"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="26sp"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent"/>
    <RadioGroup
      android:id="@+id/radios"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/selection"
      >
      <RadioButton android:id="@+id/java"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Java"
         android:onClick="onRadioButtonClicked"/>
      <RadioButton android:id="@+id/kotlin"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Kotlin"
         android:onClick="onRadioButtonClicked"/>
   </RadioGroup>
</androidx.constraintlayout.widget.ConstraintLayout>
```

**RadioGroup** sinfi **LinearLayout**dan olinganligi sababli, biz ro'yxatning vertikal yoki gorizontal yo'nalishini belgilashimiz mumkin, shu qatorda unga nafaqat radio tugmachalarini, balki boshqa moslamalarni ham qo'shishimiz mumkin, masalan, tugma yoki **TextView**.

**MainActivity** sinfida biz radio tugmachalarini tanlash bilan ishlashni aniqlaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.RadioButton;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
public void onRadioButtonClicked(View view) {
    // agar belgilash katakchasi belgilansa
    boolean checked = ((RadioButton) view).isChecked();
    TextView selection = (TextView) findViewById(R.id.selection);
    // Biz bosilgan kalitni olamiz
    switch(view.getId()) {
        case R.id.java:
            if (checked){
                selection.setText("Java tanladingiz");
            }
            break;
        case R.id.kotlin:
            if (checked){
                selection.setText("Kotlin tanladingiz");
            }
            break;
    }
}
}
```


Java tanladingiz

3.19-rasm. Dastur natijasi

**OnCheckedChangeListener**. Har bir alohida tugmachani bosish bilan bir qatorda, biz **OnCheckedChangeListener**ni butun **RadioGroup**ga kalitlari bilan osib qo'yishimiz va undagi sekin urishlarni qayta ishlashimiz mumkin. Buning uchun **android:onClick** atributlarini radio tugmachalarini belgilashdan olib tashlang va **RadioGroup** elementi uchun **id**ni aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:padding="16dp">
<TextView android:id="@+id/selection"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textSize="26sp"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintTop_toTopOf="parent"/>
 <RadioGroup
   android:id="@+id/radios"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintTop_toBottomOf="@+id/selection">
   <RadioButton android:id="@+id/java"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Java" />
   <RadioButton android:id="@+id/kotlin"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Kotlin" />
 </RadioGroup>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Keyin **MainActivity** kodida **OnCheckedChangeListener** tinglovchisini **RadioGroup** ob'ektiga belgilab qo'yamiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.RadioGroup;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // RadioGroup ob'ektini olamiz
      RadioGroup radGrp = (RadioGroup)findViewById(R.id.radios);
      // tanlash holatini almashtirishni qayta ishlash.
      radGrp.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(RadioGroup arg0, int id) {
          TextView selection = (TextView) findViewById(R.id.selection);
          switch(id) {
            case R.id.java:
              selection.setText("Java ni tanladingiz");
              break;
            case R.id.kotlin:
              selection.setText("Kotlinni tanladingiz");
              break;
            default:
              break;
          }
        }});
    }
}
```

**RadioGroup.OnCheckedChangeListener** **RadioGroup** ob'ekti va tanlangan radio tugmachasining **id**dan uzatiladigan **onCheckedChanged()** usulini belgilaydi. Keyinchalik, biz idni tekshirib, biroz ishlov beramiz.

## 3.10. DatePicker elementi

DatePicker sana tanlashni anglatadi. Uning atributlari orasida quyidagilar mavjud:

- ✓ android: calendarTextColor: taqvim matni rangi
- ✓ android: calendarViewShown: taqvim ko'rinishining ko'rsatilishini ko'rsatadi
- ✓ android: datePickerMode: sana tanlash rejimini o'rnatadi
- ✓ android: dayOfWeekBackground: hafta tanlagichining kun rangini belgilaydi
- ✓ android: endYear: oxirgi ko'rsatilgan yilni belgilaydi
- ✓ android: firstDayOfWeek: haftaning birinchi kunini belgilaydi
- ✓ android: headerBackground: tanlangan sana qatori uchun fon rangini belgilaydi
- ✓ android: maxDate: maksimal namoyish kunini mm / dd / yyyy formatida o'rnatadi
- ✓ android: minDate: minimal ko'rish kunini mm / dd / yyyy formatida o'rnatadi
- ✓ android: spinnersShown: spinner vidjetda ko'rsatilishini ko'rsatadi
- ✓ android: startYear: ko'rsatiladigan yilni belgilaydi
- ✓ android: yearListSelectorColor: yil tanlagich uchun rangni belgilaydi

**DatePicker** usullari quyidagilarni o'z ichiga oladi:

**int getDayOfMonth()**: tanlangan kunning sonini qaytaradi

**int getMonth()**: tanlangan oy raqamini qaytaradi (0 dan 11 gacha)

**int getYear()()**: tanlangan yil raqamini qaytaradi

**void init(int year, int monthOfYear, int dayOfMonth, DatePicker.OnDateChangedListener onDateChangedListener)**: boshlanish sanasini belgilaydi. Oxirgi parametr tanlangan sanani o'zgartirish uchun tinglovchini o'rnatadi

**void setOnDateChangedListener (DatePicker.OnDateChangedListener onDateChangedListener)**: tanlangan sanani o'zgartirish uchun tinglovchini o'rnatadi

**void setFirstDayOfWeek (int firstDayOfWeek)**: haftaning birinchi kunini belgilaydi

**void updateDate (int year, int month, int dayOfMonth)**: tanlangan sanani dasturiy ravishda yangilang

**DatePicker** elementi **activity_main.xml** da aniqlansin:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp">
   <TextView android:id="@+id/dateTextView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="26sp"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
    <DatePicker android:id="@+id/datePicker"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/dateTextView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Uning xatti-harakatlarini boshqarish uchun ba'zi **DatePicker** usullarini qo'llaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.DatePicker;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      TextView                              dateTextView
(TextView)findViewById(R.id.dateTextView);
      DatePicker                              datePicker
(DatePicker)this.findViewById(R.id.datePicker);
         // Noldan boshlab oy. Ko'rish uchun 1 qo'shing.
```

```
            datePicker.init(2020,           02,           01,           new
DatePicker.OnDateChangedListener() {
                @Override
                public  void  onDateChanged(DatePicker  view,  int  year,  int
monthOfYear, int dayOfMonth) {
                    // Oylar noldan hisoblanadi. Ko'rish uchun 1 qo'shing.
                    dateTextView.setText("Дата: " + view.getDayOfMonth() + "/" +
                        (view.getMonth() + 1) + "/" + view.getYear());
                    // muqobil yozuv
                    //  dateTextView.setText("Дата:  "  +  dayOfMonth  +  "/"  +
(monthOfYear + 1) + "/" + year);
                }
            });
        }
    }
```

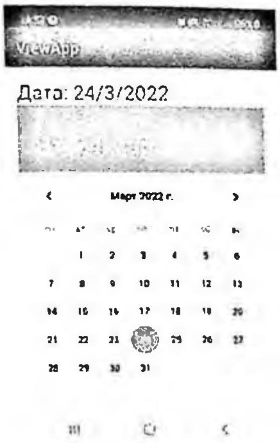
**DatePicker.init()** usulidan foydalanish; standart sanani - 2020 yil 1 martni belgiladik, chunki oylar noldan hisoblanadi. Bundan tashqari, so'nggi parametr, **DatePicker.OnDateChangedListener** ob'ekti, sana tanlagich bilan ishlashni o'rnatadi. Har safar foydalanuvchi sanani tanlaganida, **DatePicker.OnDateChangedListener** ning **onDateChanged()** usuli yonadi. Ushbu usul to'rtta parametrni oladi - **view** (DatePicker elementi), **year** (tanlangan yil), **monthOfYear** (tanlangan oy), **dayOfMonth** (tanlangan kun).

Keyin tanlangan kun, oy va yilni olishimiz mumkin. Bundan tashqari, siz uchun **onDateChanged** usuli parametrlaridan ham, **DatePicker**ning o'zi usullaridan foydalanishingiz mumkin

Tanlovdan oldingi dastlabki holat 2022 yil 1 martga o'rnatiladi.

3.20-rasm. Dastur natijasi

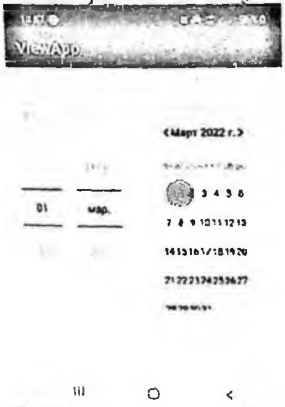
Tasodifiy bilan sana tanlash (2022 yil 24-mart):



3.21-rasm. Dastur natijasi

**DatePicker** dastlabki holat bo'yicha taqvim rejimida ko'rsatiladi, lekin biz **android:datePickerMode** atributidan foydalangan holda boshqa rejimni qo'shish uchun foydalanamiz.

```
<DatePicker android:id="@+id/datePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:datePickerMode="spinner"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/dateTextView" />
```



3.22-rasm. Dastur natijasi

Bunday holda, spinner taqvimning o'ng tomonida ko'rsatiladi. Agar biz taqvimni umuman ko'rsatishni xohlamasak, unda **android:calendarViewShown = "false"** ni o'rnatamiz.

```
<DatePicker android:id="@+id/datePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:datePickerMode="spinner"
    android:calendarViewShown="false"
```

```
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/dateTextView" />
```

## 3.11. TimePicker elementi

**TimePicker** vaqtni tanlaydigan vidjetni taqdim etadi, u vaqtni 24 soatlik yoki 12 soatlik formatda aks ettirishi mumkin.

**TimePicker** atributlari orasida vaqt rejimini ajratib ko'rsatish kerak, bu esa displey rejimiga imkon beradi va ikkita qiymatdan birini qabul qilishi mumkin: **clock** (soat ko'rinishidagi displey) va **spinner** (spinner ko'rinishidagi displey).

**TimePicker** usullari quyidagilarni o'z ichiga oladi:

- int getHour (): soatni qaytaradi (24 soatlik formatda)
- int getMinute (): daqiqalarni qaytaradi
- boolean is24HourView (): agar 24 soatlik format ishlatilsa, true qiymatini qaytaradi
- void setHour (int hour): TimePicker uchun soatni belgilaydi
- void setIs24HourView (Boolean is24HourView): 24 soatlik formatni o'rnatadi
- void setMinute (int mi_nute): daqiqalarni o'rnatadi
- void setOnTimeChangedListener (TimePicker.OnTimeChangedListener onTimeChangedListener): vaqt o'zgaruvchisini TimePicker-ga TimePicker.OnTimeChangedListener sifatida o'rnatadi

**TimePicker**ni **activity_main.xml**da aniqlaymiz:

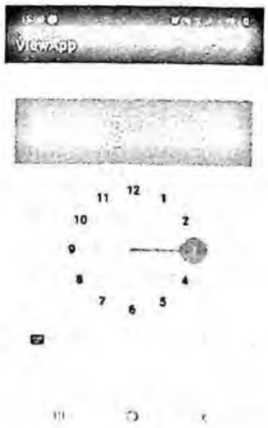
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <TextView android:id="@+id/timeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/timeTextView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Uning xatti-harakatlarini boshqarish uchun ba'zi **TimePicker** usullarini qo'llaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
```

```
        import android.os.Bundle;
        import android.widget.TextView;
        import android.widget.TimePicker;
        public class MainActivity extends AppCompatActivity {
           @Override
           protected void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.activity_main);
              TextView                              timeTextView
(TextView)findViewById(R.id.timeTextView);
              TimePicker                            timePicker
(TimePicker)this.findViewById(R.id.timePicker);
              timePicker.setOnTimeChangedListener(new
TimePicker.OnTimeChangedListener() {
                 @Override
                 public void onTimeChanged(TimePicker view, int hourOfDay, int
minute) {
                    timeTextView.setText("Minut: " + hourOfDay + ":" + minute);
                    //yoki shunday ko`rinishda
                    // timeTextView.setText("Soat: " + view.getHour() + ":" +
view.getMinute());
                 }
              });
           }
        }
```

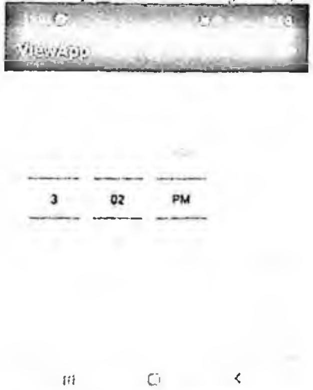
Vaqtni o'zgartiruvchi tinglovchini **TimePicker**ga qo'shish uchun **TimeOtTimeChangedListener()** usulidan foydalaning, u **TimePicker.OnTimeChangedListener** ob'ekti orqali uzatiladi. Unda **TimePicker**dagi vaqt o'zgarganda chaqiriladigan bitta usul mavjud, **onTimeChanged()**. Ushbu usul uchta parametrni o'z ichiga oladi - **TimePicker** o'zi, **hourOfDay** - soat to'plami va **minute** - o'matilgan daqiqalar. Bunday holda, biz faqat tanlangan vaqt qiymatini **TextView**ga o'tkazamiz.



3.22-rasm. Dastur natijasi

Odatiy bo'lib, **TimePicker** **"clock"** yoki soat rejimida ko'rsatiladi. Keling, "spinner" rejimini qo'llaymiz:

```
<TimePicker android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="spinner"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/timeTextView" />
```



3.23-rasm. Dastur natijasi

## 3.12. SeekBar slayderi

**SeekBar** elementi slayder vazifasini bajaradi, ya'ni joriy belgini o'zgartirishimiz mumkin bo'lgan bo'linmalar shkalasi.

Uning atributlari orasida quyidagilar mavjud:

**android:max**: maksimal qiymatni belgilaydi

**android:min**: minimal qiymatni belgilaydi

**android:progress**: minimal va maksimal oralig'idagi joriy qiymatni belgilaydi

**SeekBar**ni boshqarish uchun quyidagi usullar ajratilgan bir qator usulla belgilanadi:

**void setProgress(int progress)**: slaydemíng joriy qiymatini belgilaydi

**void setMin(int min)**: minimal qiymatni belgilaydi

**void setMax(int max):** maksimal qiymatni o'rnatadi

**void incrementProgressBy(int diff)**: joriy qiymatni diff ga oshiradi

**int getMax()**: maksimal qiymatni qaytaradi

**int getMin()**: minimal qiymatni qaytaradi

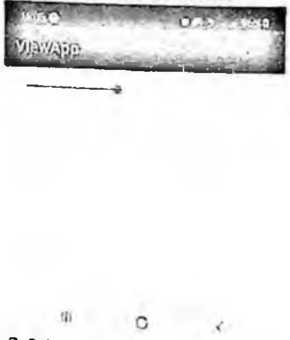
**int getProgress()**: joriy qiymatni qaytaradi

**void setOnSeekBarChangeListener(SeekBar.OnSeekBarChangeListener** SeekBardagi qiymatni o'zgartirish uchun tinglovchini o'rnatadi

**layout** belgisida **SeekBar**ni aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp">
   <SeekBar
      android:id="@+id/seekBar"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:progress="20"
      android:max="50"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**android:progress** atributi slaydemíng joriy qiymati sifatida 20 raqamini, **android:max** atributi esa mumkin bo'lgan maksimal qiymatni - 50 raqamini o'rnatadi. Natijada biz quyidagi elementni olamiz:



3.24-rasm. Dastur natijasi

Endi biz **setOnSeekBarChangeListener()** usulidan foydalanamiz, bu slayder qiymatini o'zgartirish uchun voqea ishlovchilarini o'rnatishga imkon beradi. Shunday qilib, **layout** faylida quyidagi kodni aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp">
  <TextView android:id="@+id/seekBarValue"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:textSize="26sp"
     app:layout_constraintLeft_toLeftOf="parent"
     app:layout_constraintTop_toTopOf="parent" />
  <SeekBar
     android:id="@+id/seekBar"
     android:layout_width="0dp"
     android:layout_height="wrap_content"
     android:progress="20"
     android:max="50"
     app:layout_constraintRight_toRightOf="parent"
     app:layout_constraintLeft_toLeftOf="parent"
     app:layout_constraintTop_toBottomOf="@+id/seekBarValue" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu o'zgartirilganda slayderning joriy qiymatini ko'rsatadigan **TextView** elementini belgilaydi.

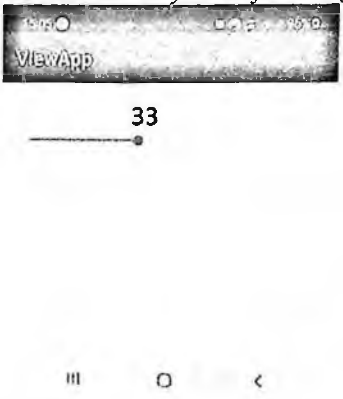
Endi **MainActivity** kodini o'zgartiraylik:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.TimePicker;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    SeekBar seekBar = (SeekBar) findViewById(R.id.seekBar);
    TextView textView = (TextView) findViewById(R.id.seekBarValue);
    seekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
       @Override
```

```
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                textView.setText(String.valueOf(progress));
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
    }
}
```

**setOnSeekBarChangeListener()** usuli **SeekBar.OnSeekBarChangeListener** ob'ekti orqali uzatiladi, bu sizga uchta ishlov berish usulini o'rnatishga imkon beradi:

- **onProgressChanged**: slayder shkala bo'ylab tortilganda yong'in chiqadi. Metodga o'tgan **progress** parametri slayderning yangi qiymatini olishga imkon beradi, bu holda u ekranda ko'rsatish uchun **TextView**ga uzatiladi.
- **onStartTrackingTouch**: slayder shkala bo'ylab siljiy boshlaganda yonadi
- **onStopTrackingTouch**: shkala bo'ylab siljitish tugagandan so'ng yonadi



3.25-rasm. Dastur natijasi

Slayderning joriy qiymatini **getProgress()** usuli yordamida ham olishimiz mumkin:

```
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        textView.setText(String.valueOf(seekBar.getProgress()));
    }
```

# IV-BOB. RESURSLAR

## 4.1. Resurslarbilan ishlash

Android dasturidagi resurs interfeysni belgilash fayli kabi fayl yoki oddiy satr kabi ba'zi bir qiymatdir. Ya'ni, resurslar ikkala belgilash fayllari, alohida satrlar, ovozli fayllar, rasm fayllari va boshqalar. Barcha resurslar loyihada res katalogida joylashgan. Loyihada aniqlangan turli xil resurs turlari uchun res katalogida kichik kataloglar yaratiladi. Qo'llab-quvvatlanadigan pastki kataloglar:

- **animator/**: _ xml fayllarining animatsiyasini belgilaydigan xml
- **anim/**: tween animatsiyasini belgilaydigan xml fayllari
- **color/**: ranglar ro'yxatini belgilaydigan xml fayllari
- **drawable/**: Grafik fayllar (.png, .jpg, .gif)
- **mipmap/**: Turli ekran o'lchamlari uchun dastur ikonkalari uchun ishlatiladigan grafik fayllar
- **layout/**: dasturning foydalanuvchi interfeysini belgilaydigan xml fayllari
- **menu/**: dastur menyusini belgilaydigan xml fayllari
- **raw/**: mavjud bo'lgan holda saqlanadigan turli xil fayllar
- **values/**: dasturda ishlatiladigan turli xil qiymatlarni o'z ichiga olgan xml fayllari, masalan, string resurslari
- **xml/**: o'zboshimchalik bilan xml fayllari
- **font/**: shrift ta'riflari va .ttf, .otf yoki .ttc kengaytmalari yoki **<font-family>** elementini o'z ichiga olgan xml fayllari.

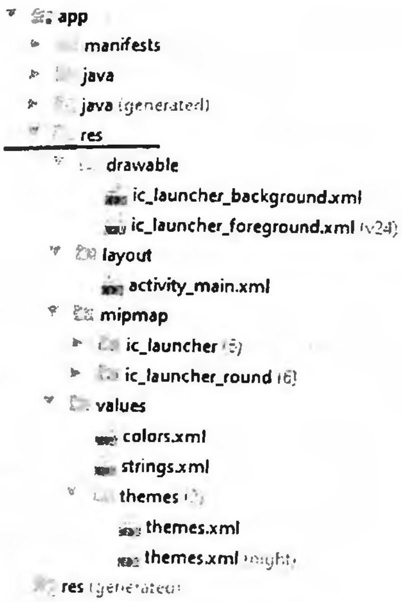
Umuman olganda biz quyidagi turdagi resurslarni aniqlay olamiz:

4.1-jadval-Resurslar jadvali

| Resurs | Loyiha katalogi | Fayl | Fayl elementi |
|---|---|---|---|
| Satrlar | /res/values/ | strings.xml | <string> |
| Plurals | /res/values/ | strings.xml | <plurals> |
| Satrlar massivi | /res/values/ | strings.xml yoki arrays.xml | <string-array> |
| Boolean tipidagi mantiqiy qiymatlar | /res/values/ | bools.xml | <bool> |
| Ranglar | /res/values/ | colors.xml | <color> |
| Ranglar ro`yhati | /res/color/ | Произвольное название | <selector> |
| O`lchamlar (Dimensions) | /res/values/ | dimens.xml | <dimen> |
| Identifikatorlar ID | /res/values/ | ids.xml | <item> |
| Butun sonlar | /res/values/ | integers.xml | <integer> |
| Butun sonlar massivi | /res/values/ | integers.xml | <integer-array> |
| Grafik fayllar | /res/drawable/ | jpg va png kengaytmalari bo'lgan fayllar | - |

| | | | |
|---|---|---|---|
| Tween-animatsiyasi | /res/anim/ | Tasodifiy bilan nomlangan xml fayli | <set>, <alpha>, <rotate>, <scale>, <translate> |
| Kadrlar bo'yicha animatsiya | /res/drawable/ | Tasodifiy bilan nomlangan xml fayli | <animation-list> |
| Animatsiya xususiyati | /res/animator/ | Tasodifiy bilan nomlangan xml fayli | <set>, <objectAnimator>, <valueAnimator> |
| Menyular | /res/menu/ | Tasodifiy bilan nomlangan xml fayli | <menu> |
| XML-fayllar | /res/xml/ | Tasodifiy bilan nomlangan xml fayli | |
| Binar(Ikkilik) va matn reusrslar | /res/raw/ | Multimedia fayllari (mp3, mp4), matnli va boshqa fayllar | |
| Grafik interfeys (GUI) o`lchamlari | /res/layout/ | Tasodifiy bilan nomlangan xml fayli | |
| Стили и темы | /res/values/ | styles.xml, themes.xml | <style> |

Masalan, biz odatiy ravishda yaratilgan standart **Android Studio** loyihasini olsak, u yerda **res** katalogida turli xil resurslar uchun bir nechta papkalar mavjudligini ko'rishimiz mumkin:



```
app
    manifests
    java
    java (generated)
    res
        drawable
            ic_launcher_background.xml
            ic_launcher_foreground.xml (v24)
        layout
            activity_main.xml
        mipmap
            ic_launcher (6)
            ic_launcher_round (6)
        values
            colors.xml
            strings.xml
            themes (2)
                themes.xml
                themes.xml (night)
    res (generated)
```

4.1-rasm Loyiha strukturasi

Odatiy bo'lib, bu erda **Android**da ishlatiladigan barcha turdagi resurslar uchun kataloglar mavjud, ammo agar kerak bo'lsa, biz kerakli katalogni **res** papkasiga qo'shib, keyin unga resursni joylashtiramiz.

Loyiha kompilyatsiya qilinganida, barcha resurslar to'g'risidagi ma'lumotlar maxsus **R.jar** fayliga qo'shiladi, undan keyin resurslar bilan ishlashda foydalaniladi.

**Resurslardan foydalanish.** Resurslarga kirishning ikkita usuli mavjud: resurs faylida va xml faylida.

**Koddagi resurslarga havola.** Ushbu yozuvdagi resurs turi **R.java** faylida aniqlangan bo'shliqlardan biriga (ichki sinflarga) tegishli bo'lib, ular xml da o'zlarining tegishli turlariga ega:

- R.drawable (u **xml drawable**dagi turiga mos keladi)
- R.id (id)
- R.layout (layout)
- R.string (string)
- R.attr (attr)
- R.plural (plurals)
- R.array (string-array)

Masalan, **MainActivity** kodidagi GUI(grafik interfeys) sifatida **activity_main.xml** resurssini o'rnatish uchun **onCreate()** usulida shunday satr mavjud:

*setContentView(R.layout.activity_main);*

**R.layout.activity_main** ifodasi orqali biz **activity_main.xml** resursiga murojaat qilamiz, bu yerda **layout** - resurs turi va **activity_main** - resurs nomi.

Xuddi shunday, biz boshqa resurslarni ham olishimiz mumkin. Masalan, **app_name** resursi **res/values/strings.xml** da aniqlanadi:

```
<resources>
  <string name="app_name">ViewApp</string>
</resources>
```

Ushbu resurs satrga ishora qiladi. Java kodida berilgan resursga havola olish uchun biz **R.string.app_name** iborasidan foydalanishimiz mumkin.

Xml faylida kirish. Ko'pincha xml faylidagi resursga murojaat qilish kerak, masalan, vizual interfeysni belgilaydigan faylda, masalan, activity_main.xml da. Xml fayllaridagi resurslarga havolalar quyidagi rasmiylashtirilgan shaklga ega:

**@ [paket_nomi:] resurs_tipi / resurs_nomi**

- ✓ **paket_nomi** resurs joylashgan paket nomini bildiradi (agar resurs bitta paketda bo'lsa, ixtiyoriy)
- ✓ **resurs_tipi** resurs turi uchun **R** sinfida belgilangan subklassni aks ettiradi
- ✓ **resurs_nomi** - resurs faylining kengaytmasi bo'lmagan nomi yoki XML elementidagi **android: name** atributining qiymati (oddiy qiymatlar uchun).

Masalan, biz **TextView**ga **strings.xml** faylida resurs sifatida aniqlangan qatorni chiqarishimiz mumkin:

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/app_name" />
```

Bunday holda, **text** xususiyati **app_name** satrli resursining qiymatini uning qiymati sifatida oladi.

**getResources usuli. Activity** sinfida resurslarni olish uchun **android.content.res.Resources** ob'ektini qaytaradigan **getResources()** usulidan foydalanishimiz mumkin. Ammo resursni o'zi olish uchun, natijada olingan **Resources** ob'ektidagi usullardan birini chaqirishimiz kerak. Uning ba'zi usullari:

- ➢ **getString()**: qatorni strings.xml faylidan raqamli **id** bo'yicha qaytaradi
- ➢ **getDimension()**: raqamli qiymatni qaytaradi - **dimen** resursi
- ➢ **getDrawable()**: Grafik faylini **Drawable** ob'ekti sifatida qaytaradi
- ➢ **getBoolean()**: **boolean** qiymatini qaytaradi
- ➢ **getColor()**: rangning ta'rifini qaytaradi
- ➢ **getColorStateList()**: **ColorStateList** ob'ekti - ranglar to'plamini qaytaradi
- ➢ **getFont()**: shrift ta'rifini **Typeface** ob'ekti sifatida qaytaradi
- ➢ **getFloat()**: **float** qiymatini qaytaradi
- ➢ **getLayout()**: **layout** fayli bilan bog'langan **XmlResourceParser** ob'ektini qaytaradi

Bu ba'zi bir usullar. Ammo ularning barchasi parametr sifatida olinadigan resursning identifikatorini oladi. Keling, ularning arizasini tezda ko'rib chiqamiz. Resursi sifatida bir xil **res/values/strings.xml** faylini olaylik, bu shunday ko'rinishda bo`ladi:

```
<resources>
  <string name="app_name">ViewApp</string>
```

```
</resources>
```

**MainActivity** kodini o'zgartiraylik:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     //setContentView(R.layout.activity_main);
     // values/strings.xml faylidan resurslarni olish
     String app_name = getResources().getString(R.string.app_name);
     TextView textView = new TextView(this);
     textView.setTextSize(30);
     textView.setText(app_name);
     setContentView(textView);
  }
}
```

Bu yerda **getResources()** usuli yordamida biz barcha resurslarni olamiz va keyin ularni grafik elementlarning xususiyat qiymatlarini belgilashda foydalanamiz. Ilovani ishga tushirishda biz olingan resurslarning qo'llanilishini ko'rib chiqamiz:

ViewApp

4.2-rasm Dastur natijasi

Xuddi shunday, biz boshqa resurslarni dasturiy ravishda olishimiz va ulardan ilovada foydalanishimiz mumkin. Shunga qaramay, shuni ta'kidlash kerakki, bu holda **getResources()** usulidan foydalanishimiz va umuman resurs olish uchun har qanday aniq harakatlarni bajarishimiz shart emas, chunki **TextView** sinfining

**setText()** usuli matnni to'g'ridan-to'g'ri sozlashni qo'llab-quvvatlaydi. resurs identifikatori:

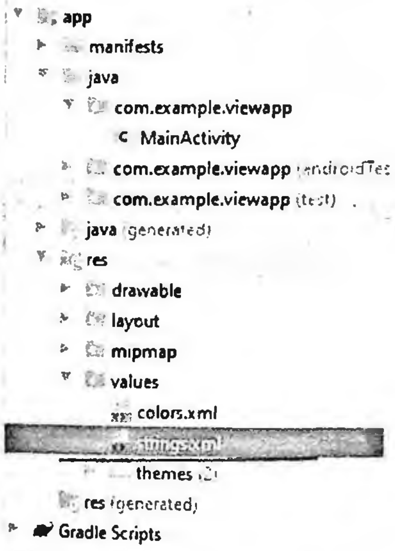
*textView.setText(R.string.app_name);*

## 4.2. Satr resurslari

Satr resurslar dasturning muhim tarkibiy qismlaridan biridir. Biz ulardan dastur nomini, turli xil matnlarni, masalan, tugmalar matni va boshqalarni ko'rsatishda foydalanamiz.

Satr resurslari bo'lgan XML fayllari loyihada **res/values** papkasida joylashgan. Odatiy bo'lib, satr resurslari **strings.xml** faylida joylashgan bo'lib, ular quyidagicha ko'rinishi mumkin:

```
<resources>
  <string name="app_name">ViewApp</string>
</resources>
```



4.3-rasm Loyiha strukturasi strings.xml fayli o`zgartirish

Oddiy shaklda ushbu fayl bitta "**app_name**" resurssini belgilaydi, bu dastur nomini o'rnatadi, biz uni qurilma ekranidagi dastur sarlavhasida ko'rib turibmiz. Ammo har qanday satrli resurslarni tabiiy ravishda aniqlashimiz mumkin. Har bir alohida resurs string elementi yordamida aniqlanadi va uning nomi atributida resurs nomi mavjud.

Keyin dasturda, kod fayllarida biz ushbu resurslarga murojaat qilishimiz mumkin:

*R.string.app_name*

Masalan, Java kodida:

*String application_name = getResources().getString(R.string.app_name);*

Yoki xml faylida:

*@string/app_name*

Masalan, **res/values/strings.xml** faylini quyidagicha o'zgartiramiz:

```
<resources>
  <string name="app_name">ViewApp</string>
  <string name="message">Hello Android!</string>
</resources>
```

"Hello Android!" Qiymatiga ega bo'lgan xabar resursi qo'shildi.

Endi biz **activity_main.xml** faylida resursdan foydalanamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/message"
    android:textSize="30sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**@string/message** ifodasidan foydalanib, qiymatni **android:text** atributiga resursdan o'tkazing.

Hello Android!

4.4-rasm Dastur natijasi

Xuddi shunday, biz resursdan **Activity** kodida foydalanishimiz mumkin:

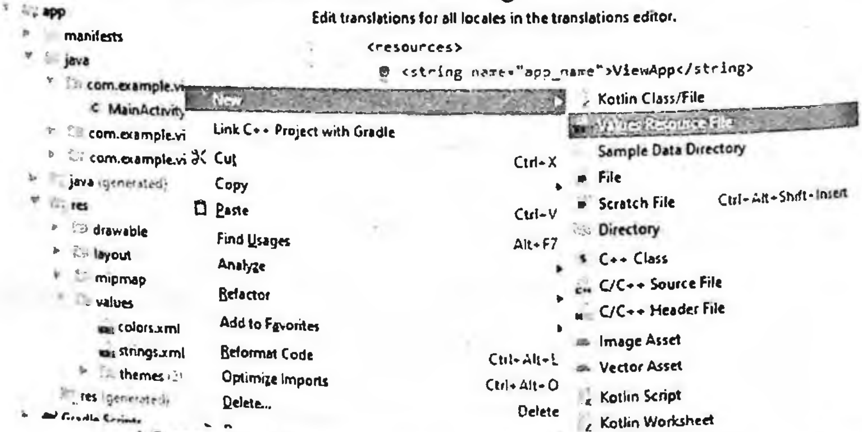
```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // textView elementini olamiz
        TextView textView = (TextView) findViewById(R.id.textView);
        // unga matnni qayta o`rnatamiz
        textView.setText(R.string.message);
    }
}
```

Agar biz printsipial ravishda **Java** kodida resursni olishimiz kerak bo'ls (**TextView**da matnni o'rnatish shart emas), unda bu holda **getResources()** usulida foydalanishimiz mumkin.

*String message = getResources().getString(R.string.message);*

Standart **strings.xml** fayli satr resurslari uchun ishlatilgan bo'lsada, ishla chiquvchilar loyihaning **res/values** katalogiga qo'shimcha resurs fayllarir qo'shishlari mumkin. Bunday holda, faylning tuzilishini kuzatish kifoya: **<resources>** ildiz tuguniga ega bo'lishi va bir yoki bir nechta **<string** elementlariga ega bo'lishi kerak.

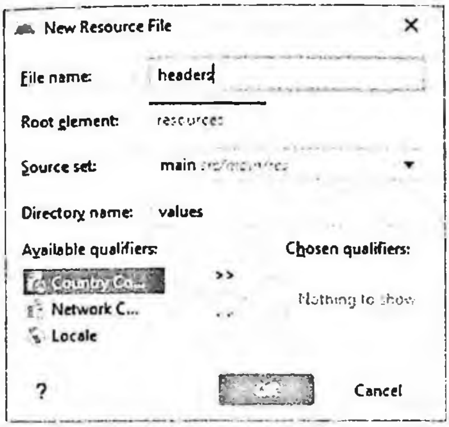
Shunday qilib, **res / values** papkasini o'ng tugmasini bosing va paydo bo'lgan ro'yxatdan **New->Value Resource File** ni tanlang:



4.5-rasm Loyiha resursiga yangi qiymat faylini qo`shish

Shuni ta'kidlash kerakki, ushbu turdagi fayllar **res/values** papkasiga qo'shiladigan har qanday turdagi resurslar uchun odatiy bo'ladi.

Shundan so'ng, bizdan fayl nomini aniqlash so'raladi:


New Resource File

File name: headers

Root element: resources

Source set: main

Directory name: values

Available qualifiers: Country Co... Network C... Locale

Chosen qualifiers: Nothing to show

4.6-rasm Loyihaning yangi resurs fayliga yangi nom berish

Masalan, **headers** (fayl nomi tasodifiy bilan) deb nomlaylik va boshqa barcha maydonlar uchun biz standart qiymatlarni qoldiramiz. **res/values** papkasiga yangi **headers.xml** fayli qo'shiladi. Undagi bir nechta resurslarni aniqlaylik:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="welcome"> Xush kelibsiz </string>
  <string name="click_button"> tugmachani bosing </string>
</resources>
```

Keyin biz **Activity** kodida yoki **layout**da belgilangan resurslardan foydalanishimiz mumkin.

**Satrlarni formatlash.** Android sizga string formatidagi resurslarga formatlashni qo'llash imkonini beradi. Masalan, **string.xml** faylini o'zgartiramiz:

```
<resources>
  <string name="app_name">ViewApp</string>
  <string name="message">Hello Android!</string>
  <string name="welcome_message">Hush kelibsiz %1Ss! Ayni paytda
%2Sd : %3Sd</string>
</resources>
```

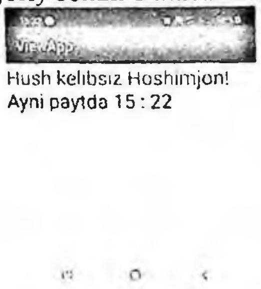
Uchinchi resurs, **welcome_message**, formatlangan satrni aks ettiradi. Masalan, unda% 1 $ s,% 2 $ d va% 3 $ d kabi belgilar mavjud. Ular nimani anglatadi? % 1 $ s bu birinchi argument ekanligini va "s" belgisi bu argument satrni anglatishini bildiradi. % 2 $ d ikkinchi argumentni ifodalaydi va "d" uning butun son bo'lishini bildiradi. Xuddi shu tarzda,% 3 $ d bu uchinchi raqam ekanligini ko'rsatib, butun sonni bildiradi.

Java kodidagi resursni oling

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TextView;
import java.util.Calendar;
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        Resources res = getResources();
        String userName = "Hoshimjon";
        Calendar calendar = Calendar.getInstance();
        int hour = calendar.get(Calendar.HOUR_OF_DAY);
        int minute = calendar.get(Calendar.MINUTE);
        String text = getString(R.string.welcome_message, userName, hour,
minute);
        TextView textView = new TextView(this);
        textView.setText(text);
        textView.setTextSize(28);
        setContentView(textView);
    }
}
```

**getString(R.string.welcome_message, userName, hour, minute)** usuli **welcome_message** resurssini oladi va uni keyingi parametrlar sifatida qiymat argumentlariga uzatadi. Birinchi satr argumenti uchun **userName** o'zgaruvchisi ishlatiladi va ikkinchi va uchinchi argumentlar uchun biz **Calendar** sinfi yordamida olingan soat va daqiqalarning joriy sonini o'tkazamiz.

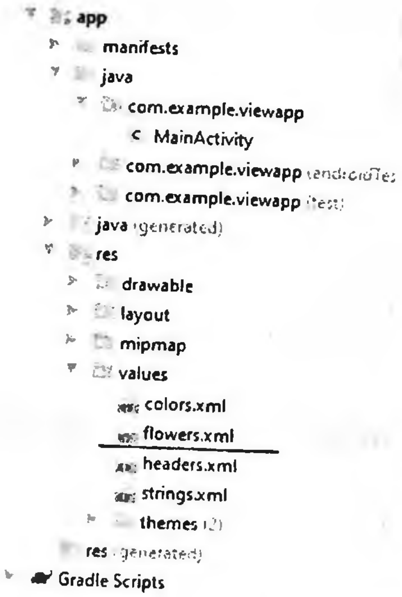


```
Hush kelibsiz Hoshimjon!
Ayni paytda 15 : 22
```

4.7-rasm Dastur natijasi

**Plurals resurslari**. **Plurals** satrlarning yana bir turini anglatadi. Bu buyumlar sonini tavsiflash uchun mo'ljallangan. Bu nima uchun? Masalan, ismni olaylik: u ko'pincha ishlatilgan songa qarab sonni o'zgartiradi: 1 gul, 2 gul, 5 gul. Bunday holatlar uchun p_lurals resursidan foydalaniladi.

Keling, bir misolni ko'rib chiqaylik. **res/values** papkasiga yangi resurs qo'shaylik. Keling, uni **flowers** deb ataymiz:



4.8-rasm Loyiha strukturasida flowers.xml fayli

Uning tarkibini quyidagicha o'zgartiraylik:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <plurals name="flowers">
    <item quantity="one">%d цветок</item>
    <item quantity="few">%d цветка</item>
    <item quantity="many">%d цветков</item>
  </plurals>
</resources>
```

Resursni aniqlash uchun **<plurals>** elementidan foydalaniladi, buning uchun nom atributi mavjud bo'lib, u o'zboshimchalik bilan nomni qiymat sifatida qabul qiladi va keyinchalik ushbu resursga murojaat qiladi.

Satr satrlarning o'zi voris elementlari tomonidan kiritiladi **<item>** elementlari. Ushbu elementda ushbu satr ishlatilishini ko'rsatadigan qiymatga ega bo'lgan miqdor atributi mavjud. Ushbu atribut quyidagi qiymatlarni qabul qilishi mumkin:

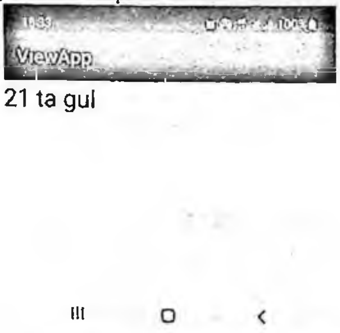
- **zero**: 0 o'lchamdagi miqdor uchun satr
- **one**: satrga 1 miqdor (rus tilida - 1 bilan tugaydigan barcha miqdorlarni ko'rsatish uchun, 11dan tashqari)
- **two**: 2 o'lchamdagi miqdor uchun satr
- **few**: oz miqdordagi satr
- **many**: katta sonlar uchun satr
- **other**: boshqa barcha holatlar

Bunday holda, ko'p narsa o'ziga xos tilga bog'liq. Va tizimning o'zi ma'lum bir raqam uchun qanday qiymat olish kerakligini aniqlashga imkon beradi.

Ushbu resursdan faqat java kodida foydalanish mumkin. Shuning uchun biz **MainActivity** kodini o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      //setContentView(R.layout.activity_main);
      String rose = getResources().getQuantityString(R.plurals.flowers, 21,
21);
      TextView textView = new TextView(this);
      textView.setText(rose);
      textView.setTextSize(26);
      setContentView(textView);
   }
}
```

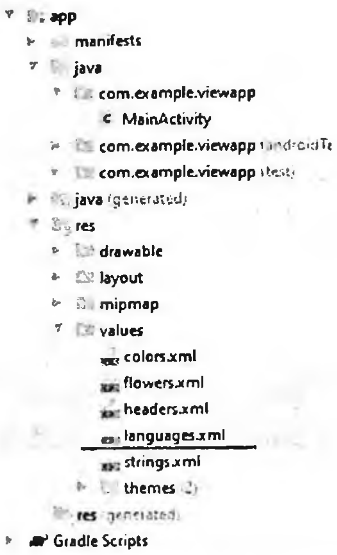
**getQuantityString** usuli yordamida biz resurs qiymatini olamiz. Birinchi parametr - bu resurs identifikatori. Ikkinchi parametr - bu qiymat. buning uchun kerakli qatorni topmoqchisiz. Uchinchi parametr -% d to'ldiruvchisi o'rniga kiritiladigan qiymat. Ya'ni, biz 21 raqami uchun satrni olamiz.



21 ta gul

4.9-rasm. Dastur natijasi

**String array**. Satr resursning yana bir turi - **string-array** yoki satrlar massivi. Masalan, **res/values** papkasiga **languages.xml** deb nomlangan yangi fayl qo'shamiz:



4.10-rasm Loyiha strukturasida languages.xml fayli

Unda quyidagi kod bo'lishi kerak:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="languages">
    <item>Java</item>
    <item>Kotlin</item>
    <item>Dart</item>
  </string-array>
</resources>
```

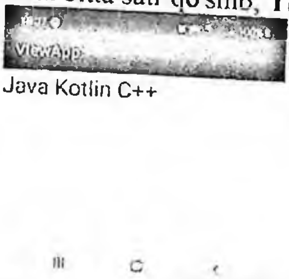
Resurs **<string-array>** elementi yordamida aniqlanadi. Bu aslida satrlar to'plamini belgilaydi. Va har bir alohida satr **<item>** elementi yordamida o'rnatiladi

**MainActivity.java** faylida ushbu manbadan qiymatlarni olish uchun kodni aniqlaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.activity_main);
    Resources res = getResources();
    String[] languages = res.getStringArray(R.array.languages);
```

```
        String allLangs = "";
        for (String lang: languages) {
            allLangs += lang + " ";
        }
        TextView textView = new TextView(this);
        textView.setText(allLangs);
        textView.setTextSize(28);
        setContentView(textView);
    }
}
```

**getStringArray** usuli yordamida biz resurslarni satrlar massiviga kiritamiz, so'ngra sikl yordamida massivdan bitta satr qo'shib, **TextView**ga o'tkazamiz.



Java Kotlin C++

4.11-rasm. Dastur natijasi

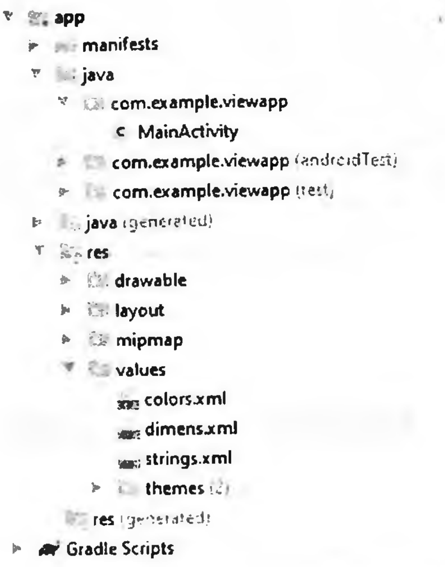
## 4.3. Dimension resurslari

O'lcham har qanday ixtiyori nom bilan nomlangan fayldagi **res/values** papkasida joylashgan bo'lishi kerak. Resursni aniqlash uchun umumiy sintaksis quyidagicha:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="resurs_nomi">o'lchamidan foydalanish</dimen>
</resources>
```

Boshqa manbalar singari, **dimension** manbai **<resources>** ildiz elementida aniqlanadi. **<dimen>** yorlig'i manbani bildiradi va qabul qilingan o'lchov birliklaridan birida (dp, sp, pt, px, mm, in) o'lchov qiymatini oladi. O'lchash haqida avvalgi maqolalardan birida - o'lchamlarni aniqlashda batafsilroq muhokama qilingan.

Shunday qilib, keling, Android studiyasidagi **res/values** papkasiga yangi **Values Resources File** elementini qo'shaylik, biz uni **dimens.xml** deb ataymiz.


```
app
  manifests
  java
    com.example.viewapp
      MainActivity
    com.example.viewapp (androidTest)
    com.example.viewapp (test)
  java (generated)
  res
    drawable
    layout
    mipmap
    values
      colors.xml
      dimens.xml
      strings.xml
      themes (2)
  res (generated)
Gradle Scripts
```

4.12-rasm Loyiha strukturasida dimens.xml fayli

Unda quyidagi tarkibni aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <dimen name="horizontal_margin">64dp</dimen>
  <dimen name="vertical_margin">32dp</dimen>
  <dimen name="text_size">32sp</dimen>
</resources>
```

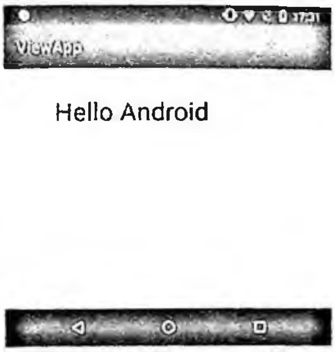
To'ldirish uchun ikkita, masalan, 64dp va 32dp ushlab turadigan horizontal_margin va **vertical_margin** va shrift balandligi, 32sp saqlanadigan **text_size** manbasini belgilaydi. Resurs nomlari tasodifiy bilan bo'lishi mumkin.

Resursni **activity_main.xml** faylida ishlatamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello Android"
    android:background="#eaeaea"
    android:layout_marginTop="@dimen/vertical_margin"
    android:layout_marginLeft="@dimen/horizontal_margin"
```

```
        android:textSize="@dimen/text_size"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**dimension** resurslari ularning qiymati sifatida raqamli qiymatni talab qiladigan vizual atributlar uchun ishlatiladi. Masalan, **android:layout_height** yoki **android:textSize** xususiyati. Resursni xml formatida olish uchun **"@dimen/"** dan keyin resurs nomi ko'rsatiladi.



Hello Android

4.13-rasm. Dastur natijasi

**Java** kodi manbalarni olish uchun **Resources** sinfining **getDimension()** usulidan foydalanadi. Masalan, shunga o'xshash vizual interfeysni **Java** kodida aniqlaylik:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.content.res.Resources;
import android.os.Bundle;
import android.util.TypedValue;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        // biz resurslarni olamiz
        Resources resources = getResources();
        float textSize = resources.getDimension(R.dimen.text_size);
        int                              hMargin
(int)resources.getDimension(R.dimen.horizontal_margin);
        int vMargin = (int)resources.getDimension(R.dimen.vertical_margin);
        ConstraintLayout constraintLayout = new ConstraintLayout(this);
        ConstraintLayout.LayoutParams        layoutParams        =        new
ConstraintLayout.LayoutParams
```

```
                (ConstraintLayout.LayoutParams.WRAP_CONTENT
ConstraintLayout.LayoutParams.WRAP_CONTENT);
            layoutParams.leftToLeft
ConstraintLayout.LayoutParams.PARENT_ID;
            layoutParams.topToTop
ConstraintLayout.LayoutParams.PARENT_ID;
            TextView textView = new TextView(this);
            textView.setText("Hello Android");
            textView.setBackgroundColor(0XFFEAEAEA);
            // shrift hajmini resurs bo'yicha o'rnating
            textView.setTextSize(TypedValue.COMPLEX_UNIT_PX, textSize);
            // resurslarga muvofiq to'ldirishni o'rnating
            layoutParams.setMargins(hMargin, vMargin, hMargin, vMargin);
            textView.setLayoutParams(layoutParams);
            constraintLayout.addView(textView);
            setContentView(constraintLayout);
        }
    }
```

**getDimension()** usuli yordamida **dimen** resurslarini dasturiy ravishda olishda, manba faylidagi (32sp, 64dp) qiymatlarni aniqlagan bo'lsak ham, qiymatlarni fizik piksellarda qaytarishini yodda tuting. Ko'pgina hollarda, bu hech qanday qiyinchilik tug'dirmaydi, chunki **Java**dagi aksariyat usullar aniq piksellarni qabul qiladi, emas, balki dp yoki boshqa birliklar, masalan, chekkalarni o'rnatadigan **setMargins()** usuli.

Biroq, **setTextSize()** usuli aniq **sp**ni qabul qiladi, shuning uchun qo'shimcha parametrdan foydalanib, bu holda biz **sp** emas, balki jismoniy piksellarni anglatishini belgilashingiz kerak:

```
textView.setTextSize(TypedValue.COMPLEX_UNIT_PX, textSize);
```

Yoki, muqobil ravishda, **TypedValue** sinfidan foydalanib, fizik piksellarni **sp** yoki boshqa o'lchov birligiga dasturiy ravishda o'zgartiring.

## 4.4. Color resurslari va ranglarni sozlash.

**Android** dasturida siz rang resurslarini (**Color**) belgilashingiz mumkin. Ular **res/values** yo'li ostida faylda saqlanishi kerak va satr resurslari singari **<resources>** yorlig'iga kiritilgan. Shunday qilib, sukut bo'yicha, eng sodda loyihani yaratishda, **color.xml** fayli **res/values** papkasiga qo'shiladi:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="purple_200">#FFBB86FC</color>
   <color name="purple_500">#FF6200EE</color>
   <color name="purple_700">#FF3700B3</color>
   <color name="teal_200">#FF03DAC5</color>
   <color name="teal_700">#FF018786</color>
   <color name="black">#FF000000</color>
   <color name="white">#FFFFFFFF</color>
```

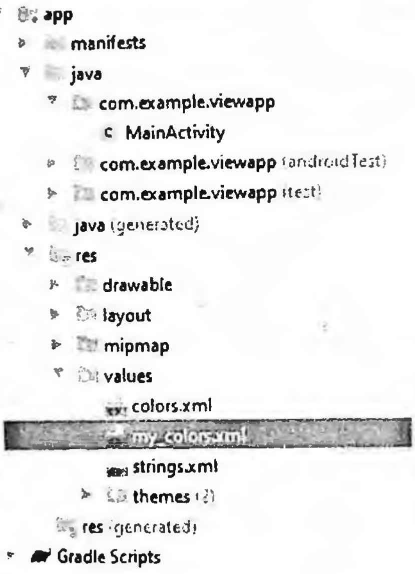
*</resources>*

Rang **<color>** elementi yordamida aniqlanadi. Uning nomi atributi dasturda ishlatiladigan rang nomini belgilaydi va o'n oltinchi raqam rang qiymatidir.

Rang resurslarini aniqlash uchun quyidagi formatlardan foydalanish mumkin:

- #RGB (# F00 - 12-bitli qiymat)
- #ARGB (# 8F00 - alfa kanali qo'shilgan 12-bitli qiymat)
- #RRGGBB (# FF00FF - bu 24-bitli qiymat)
- #AARRGGBB (# 80FF00FF - alfa kanali qo'shilgan 24-bitli qiymat)

Ushbu faylga tegmaslik yoki buzilmasligi uchun biz yangi resurs faylimizni aniqlaymiz va buning uchun **res/values** papkasiga **my_colors.xml** deb nomlanadigan yangi resurs faylini qo'shamiz.



```
app
  manifests
  java
    com.example.viewapp
      MainActivity
    com.example.viewapp (androidTest)
    com.example.viewapp (test)
  java (generated)
  res
    drawable
    layout
    mipmap
    values
      colors.xml
      my_colors.xml
      strings.xml
    themes (2)
  res (generated)
Gradle Scripts
```

4.14-rasm Loyiha strukturasida my_colors.xml fayli

**my_colors.xml** faylini unga ikkita rang qo'shish uchun o'zgartiraylik:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="textViewBackColor">#A0EAE1</color>
  <color name="textViewFontColor">#00695C</color>
</resources>
```

Ranglarni **activity_main.xml** faylida qo'llaymiz:

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

```
<TextView
   android:id="@+id/textView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Hello Android"
   android:background="@my_colors/textViewBackColor"
   android:textColor="@my_colors/textViewFontColor"
   android:textSize="32sp"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**android:textColor** atributi **TextView**dagi matn rangini, **android:background** atributi esa **TextView**ning fonini belgilaydi. Ular rangni qiymat sifatida ishlatishadi, masalan, xuddi shu o'n oltilik formatida. Rangni o'zi olish uchun resurs nomi "@**color/**" dan keyin ko'rsatiladi.

Hello Android

4.15-rasm. Dastur natijasi

Shuningdek, **MainActivity** kodingizda rang resurslaridan foydalanishingiz mumkin:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        // biz resurslarni olamiz
        Resources resources = getResources();
        int textColor = resources.getColor(R.color.textViewFontColor, null);
        int                              backgroundColor                         =
resources.getColor(R.color.textViewBackColor, null);
        ConstraintLayout constraintLayout = new ConstraintLayout(this);
        ConstraintLayout.LayoutParams         layoutParams          =          new
ConstraintLayout.LayoutParams
                (ConstraintLayout.LayoutParams.WRAP_CONTENT
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        layoutParams.leftToLeft
ConstraintLayout.LayoutParams.PARENT_ID;
        layoutParams.topToTop
ConstraintLayout.LayoutParams.PARENT_ID;
        TextView textView = new TextView(this);
        textView.setText("Hello Android");
        textView.setTextSize(32);
        // color resurslaridan foydalanish
        textView.setTextColor(textColor);
        textView.setBackgroundColor(backgroundColor);
        textView.setLayoutParams(layoutParams);
        constraintLayout.addView(textView);
        setContentView(constraintLayout);
    }
}
```

Resurslardan rangni olish uchun ikkita parametrni o'z ichiga olgar **resources.getColor()** usuli qo'llaniladi. Birinchi parametr - siz rang olishni istagar manbaning identifikatori. Ikkinchi parametr mavzuni anglatadi. Ammo bu hold mavzu muhim bo'lmaganligi sababli, biz ushbu parametr uchun **null** qiymatin beramiz.

Iltimos, iltimos, yuqorida ko'rsatilgan ikkita parametrga ega bo'lgan **Resources.getColor** () usuli minimal Android versiyasi kamida Android 6.0 (yoki Android 23) bo'lsa, foydalanish mumkin. Biroq, minimal Android versiyasi pastroq, keyin siz bitta parametr bilan eskirgan versiyadan foydalanishingiz mumkin:

```
int textColor = resources.getColor(R.color.textViewFontColor);
// o'rniga
//int textColor = resources.getColor(R.color.textViewFontColor, null);
```

# V-BOB. Activity

## 5.1. Activity va dasturning hayot aylanishi.

**Android** dasturida vizual interfeys yaratishning asosiy komponenti **activity** hisoblanadi. Ko'pincha, **activity** alohida ekran yoki dastur oynasi bilan bog'liq bo'lib, derazalar o'rtasida biridan ikkinchisiga o'tishda sodir bo'ladi. Ilovada bir yoki bir nechta ishlash qobiliyati bo'lishi mumkin. Masalan, bo'sh **Activity** bilan loyiha yaratishda, loyihaga sukut bo'yicha bitta **Activity** sinfi, **MainActivity** qo'shiladi: undan dastur ishlay boshlaydi:

```
public class MainActivity extends AppCompatActivity {
    // sinf tarkibi
}
```

Barcha **activity** ob'ektlari **android.app.Activity** sinfining ob'ektlari bo'lib, u barcha **activity** uchun asosiy funktsiyalarni o'z ichiga oladi. Oldingi mavzudagi dasturda biz ushbu sinf bilan to'g'ridan-to'g'ri ishlamadik va **AppCompatActivity** sinfidan meros qolgan **MainActivity**. Biroq, **AppCompatActivity** o'zi to'g'ridan-to'g'ri bo'lmasa ham, asosiy **Activity** sinfidan oladi.

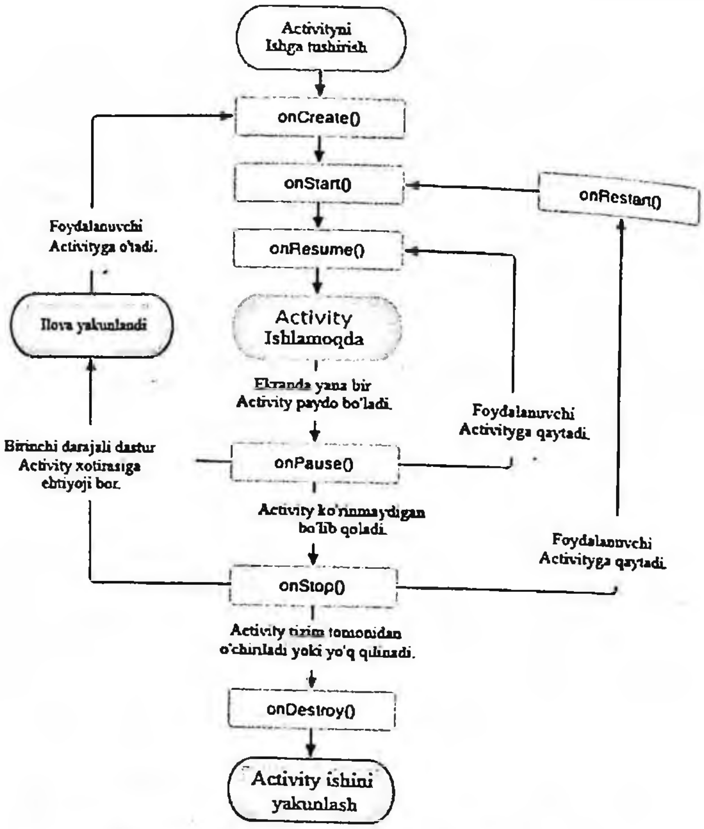
**Amaliy hayot aylanishi.** Barcha Android dasturlari qat'iy belgilangan hayot aylanishiga ega. Agar foydalanuvchi dasturni ishga tushirsa, tizim ushbu dasturga yuqori ustuvorlikni beradi. Har bir dastur alohida jarayon sifatida ishlaydi, bu tizim ba'zi jarayonlarga boshqalarnikiga qaraganda yuqori ustuvorlikni berishga imkon beradi. Shu tufayli, masalan, ba'zi Android dasturlari bilan ishlashda sizga kiruvchi murojaatlarni bloklamaslik mumkin. Ilova bilan ishlash tugagandan so'ng tizim barcha bog'liq resurslarni chiqaradi va dasturni past ustuvorlik toifasiga o'tkazadi va uni yopadi.

Ilovadagi barcha **activity** ob'ektlari tizim tomonidan **back stack** deb nomlangan **activity** to'plami sifatida boshqariladi. Yangi **activity** ishga tushirilganda, u stackning yuqori qismiga qo'yiladi va yangi **activity** paydo bo'lguncha qurilma ekranida ko'rsatiladi. Joriy **activity** o'z ishini tugatgandan so'ng (masalan, foydalanuvchi dasturni tark etadi), keyin u stekdan o'chiriladi va avval stekda ikkinchisi bo'lgan **activity** qayta tiklanadi.

Boshlangandan so'ng, **activity** tizim tomonidan qayta ishlanadigan va bir nechta qayta murojaatlar bo'lgan bir qator voqealarni boshdan kechiradi:

```
protected void onCreate(Bundle saveInstanceState);
protected void onStart();
protected void onRestart();
protected void onResume();
protected void onPause();
protected void onStop();
protected void onDestroy();
```

Ushbu qayta murojaatlarning barchasi o'rtasidagi munosabatlar sxematik tarzda quyidagicha ifodalanishi mumkin.



Activityni Ishga tushirish
onCreate()
onStart()
onRestart()
Foydalanuvchi Activityga o'tadi.
onResume()
Ilova yakunlandi
Activity Ishlamoqda
Ekranda yana bir Activity paydo bo'ladi.
Foydalanuvchi Activityga qaytadi.
Birinchi darajali dastur Activity xotirasiga ehtiyoji bor.
onPause()
Activity ko'rinmaydigan bo'lib qoladi.
Foydalanuvchi Activityga qaytadi.
onStop()
Activity tizim tomonidan o'chiriladi yoki yo'q qilinadi.
onDestroy()
Activity ishini yakunlash

5.1-rasm. Activity ni ishga tushirish sxemasi

**onCreate(). onCreate** - bu **activity** bilan boshlanadigan birinchi usul. Ushbu usulda **activity Created** holatiga o'tadi. Ushbu usul **activity** sinfida aniqlangan bo'lishi kerak. Bu yerda dastlabki **activity** sozlamalari o'rnatiladi. Xususan, vizual interfeys ob'ektlari yaratiladi. Ushbu usul saqlangan bo'lsa, oldingi faoliyat holatini o'z ichiga olgan **Bundle** ob'ektini oladi. Agar **activity** qayta yaratilsa, u holda bu ob'ekt **null** bo'ladi. Agar **activity** allaqachon yaratilgan bo'lsa, lekin to'xtatilgan holatda bo'lsa, **bundle** ichida **activity** bilan bog'liq ma'lumotlar mavjud.

**OnCreate()** usuli bajarilgandan so'ng, **activity Started** holatiga o'tadi va tizim **onStart()** usulini chaqiradi.

**OnStart. OnStart()** usuli uskuna ekranida **activity** ko'rsatishga tayyorlanadi. Qoida tariqasida, ushbu usulni bekor qilish kerak emas va barcha ishlar ichki kod orqali amalga oshiriladi. **Activity** usuli tugagandan so'ng, u ekranda ko'rsatiladi, **onResume** usuli chaqiriladi va **activity** davom ettiriladi.

**onResume. OnResume** usuli chaqirilganda, **activity Resumed** holatiga o'tadi va qurilma ekranida aks etadi va foydalanuvchi u bilan o'zaro aloqada bo'lishi mumkin. Va **activity** o'zi shu holatida, masalan, boshqa **activity** ga o'tish sababli yoki shunchaki qurilma ekrani o'chib qolganligi sababli, diqqatni yo'qotguncha qoladi.

**onPause**. Agar foydalanuvchi boshqa **activity** ga o'tishga qaror qilsa, tizim onPause usulini chaqiradi va **activity Paused** holatiga o'tadi. Ushbu usulda siz foydalaniladigan resurslarni bo'shatishingiz, jarayonlarni to'xtatib qo'yishingiz, masalan, audio, animatsiyalarni ijro etishingiz, kamerani to'xtatishingiz (agar u ishlatilayotgan bo'lsa) va hk.

Shuni esda tutish kerakki, bu holatda hali ham ekranda ko'rinadigan bo'lib qoladi va bu usulning ishlashiga juda oz vaqt ajratiladi, shuning uchun siz bu erda hech qanday ma'lumotni saqlamasligingiz kerak, ayniqsa, bu tarmoqqa kirishni talab qilsa, masalan, Internet orqali ma'lumotlarni yuborish yoki ma'lumotlar bazasiga kirish - bu harakatlar eng yaxshi **onStop()** usulida amalga oshiriladi.

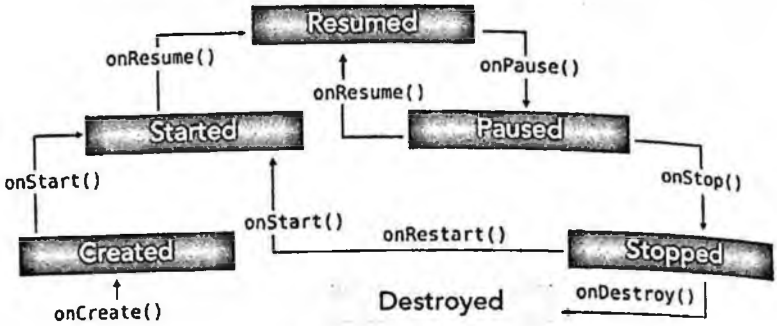
Ushbu usulni amalga oshirgandan so'ng, **activity** ko'rinmaydi, ekranda ko'rinmaydi, lekin u hali ham faol. Agar foydalanuvchi ushbu **activity**-ga qaytishga qaror qilsa, u holda tizim yana **onResume** usulini chaqiradi va ekranda yana **activity** paydo bo'ladi.

To'satdan tizim faol dasturlarning ishlashi uchun ko'proq xotira kerakligini ko'rsa, ishning yana bir varianti paydo bo'lishi mumkin. Va tizimning o'zi ko'rinmas va fonda joylashgan **activity** ishini to'liq bajarishi mumkin. Shu bilan bir qatorda, foydalanuvchi **back** (Orqaga) tugmachasini bosishi mumkin. Bunday holda, onStop usuli **activity** deb nomlanadi. Ushbu usulda **activity Stopped** holatiga o'tadi. Bunday holatda, **activity** to'liq ko'rinmaydi. **onStop** usulida foydalanuvchi **activity** bilan o'zaro aloqada bo'lmagan paytda kerak bo'lmagan manbalarga ixtisoslashishingiz kerak. Bu erda ma'lumotlarni, masalan, ma'lumotlar bazasiga saqlash mumkin. Shu bilan birga, **Stopped** holatida, qurilma xotirasida **activity** qoladi, barcha interfeys elementlarining holati saqlanib qoladi. Masalan, **EditText** matn maydoniga biron bir matn kiritilgan bo'lsa, u holda **activity** qayta tiklanib, **Resumed** holatiga o'tgandan so'ng, biz matn maydonida avval kiritilgan matnni yana ko'ramiz. Agar **onStop** usulini chaqirgandan so'ng foydalanuvchi oldingi **activity**-ga qaytishga qaror qilsa, u holda tizim onRestart usulini chaqiradi. Agar **activity** o'z ishini umuman tugatgan bo'lsa, masalan, dastur yopilishi sababli, u holda **onDestroy()** usuli chaqiriladi.

**onDestroy**. **Activity onDestroy** usulini chaqirish bilan tugaydi, bu tizim konfiguratsiya sabablari (masalan, ekranning aylanishi yoki ko'p oynali rejimda) sababli **activity**ni o'ldirishga qaror qilganda yoki **finish()** usuli chaqirilganda paydo bo'ladi.

Shuni ham yodda tutingki, ekran yo'nalishi o'zgarganda, tizim **activity** dan chiqadi va uni **onCreate** usulini chaqirib qayta yaratadi.

Umuman olganda, **activity** holatlari orasidagi o'tish quyidagi sxema bilan ifodalanishi mumkin:



5.2-rasm. Activity holatlari orasidagi o'tish sxemasi

Keling, bir nechta vaziyatlarni ko'rib chiqaylik. Agar biz **Activity** bilan ishlasak, keyin boshqa dasturga o'tsak yoki Home tugmachasini bosgan bo'lsak, unda **Activity** uchun quyidagi usullar zanjiri chaqiriladi: **onPause -> onStop**. **Activity Stopped** holatidadir. Agar foydalanuvchi **Activity**ga qaytishga qaror qilsa, quyidagi usullar zanjiri deyiladi: **onRestart -> onStart -> onResume.**

Yana bir holat, agar foydalanuvchi **Back (Orqaga)** tugmachasini bossa, u holda quyidagi zanjir **onPause -> onStop -> onDestroy** chaqiriladi. Natijada, **Activity** yo'q qilinadi. Agar biz to'satdan vazifa menejeri orqali **Activity**ga qaytishni yoki dasturni qayta ochishni xohlasak, u holda **activity onCreate -> onStart->onResume** usullari orqali qayta tiklanadi

**Hayotiy tsiklni boshqarish.** Ushbu hayot aylanish jarayonlarini tegishli usullarni bekor qilish orqali boshqarishimiz mumkin. Buning uchun **MainActivity** sinfini oldingi bobdan oling va uni quyidagicha o'zgartiring:

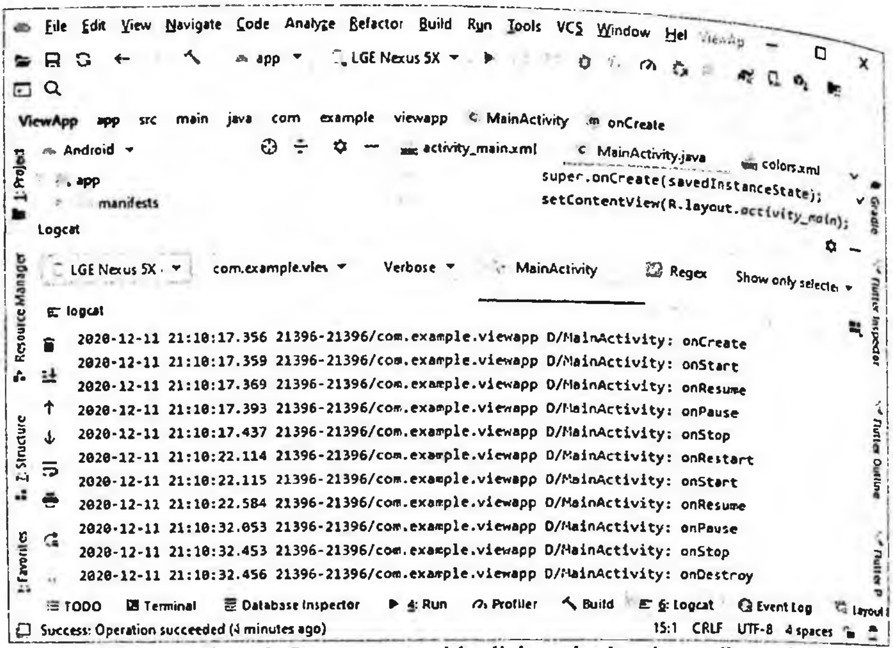
```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends AppCompatActivity {
   private final static String TAG = "MainActivity";
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(TAG, "onCreate");
   }
   @Override
   protected void onDestroy(){
      super.onDestroy();
      Log.d(TAG, "onDestroy");
   }
   @Override
   protected void onStop(){
      super.onStop();
```

```
        Log.d(TAG, "onStop");
    }
    @Override
    protected void onStart(){
        super.onStart();
        Log.d(TAG, "onStart");
    }
    @Override
    protected void onPause(){
        super.onPause();
        Log.d(TAG, "onPause");
    }
    @Override
    protected void onResume(){
        super.onResume();
        Log.d(TAG, "onResume");
    }
    @Override
    protected void onRestart(){
        super.onRestart();
        Log.d(TAG, "onRestart");
    }
}
```

Voqealarni ro'yxatdan o'tkazish uchun bu erda **android.util.Log** sinfi ishlatiladi.

Bunday holda, hayot aylanishining barcha asosiy usullari qayta ishlanadi. Barcha ishlov berish **Log.d()** usuli chaqiruviga qisqartiriladi, unga **TAG** yuboriladi - tasodifiy satr qiymati va **Android Studio** pastki qismidagi **Logcat** konsolida ko'rsatiladigan satr, disk raskadrovka ma'lumotlari sifatida ishlaydi. Agar ushbu konsol sukut bo'yicha yashiringan bo'lsa, biz unga menyu elementi orqali o'tishimiz mumkin View-> Tool Windows -> Logcat.

Ilova ishga tushgach, biz **Logcat** oynasida disk raskadrovka ma'lumotlarini ko'rishimiz mumkin, bu esa **activity** hayot davri usullarida aniqlanadi:


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app
LGE Nexus 5X
ViewApp app src main java com example viewapp MainActivity onCreate
Android
activity_main.xml
MainActivity.java
colors.xml
app
manifests
Logcat
LGE Nexus 5X
com.example.view
Verbose
MainActivity
Regex
Show only selected
logcat
TODO Terminal Database Inspector 4: Run Profiler Build 6: Logcat Event Log Layout
Success: Operation succeeded (4 minutes ago)
15:1 CRLF UTF-8 4 spaces

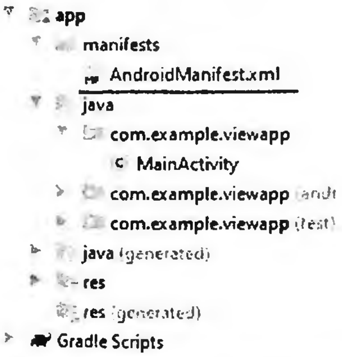
```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
```

```
2020-12-11 21:10:17.356 21396-21396/com.example.viewapp D/MainActivity: onCreate
2020-12-11 21:10:17.359 21396-21396/com.example.viewapp D/MainActivity: onStart
2020-12-11 21:10:17.369 21396-21396/com.example.viewapp D/MainActivity: onResume
2020-12-11 21:10:17.393 21396-21396/com.example.viewapp D/MainActivity: onPause
2020-12-11 21:10:17.437 21396-21396/com.example.viewapp D/MainActivity: onStop
2020-12-11 21:10:22.114 21396-21396/com.example.viewapp D/MainActivity: onRestart
2020-12-11 21:10:22.115 21396-21396/com.example.viewapp D/MainActivity: onStart
2020-12-11 21:10:22.584 21396-21396/com.example.viewapp D/MainActivity: onResume
2020-12-11 21:10:32.053 21396-21396/com.example.viewapp D/MainActivity: onPause
2020-12-11 21:10:32.453 21396-21396/com.example.viewapp D/MainActivity: onStop
2020-12-11 21:10:32.456 21396-21396/com.example.viewapp D/MainActivity: onDestroy
```

5.3-rasm. Ilovada Logcat oynasida disk raskadrovka ma'lumotlari

## 5.2. AndroidManifest.xml manifest fayli.

Har bir dasturda **AndroidManifest.xml** manifest fayli mavjud. Ushbu fayl dastur haqida muhim ma'lumotlarni - nomi, versiyasi, piktogramma, dastur qanday ruxsatlardan foydalanishini belgilaydi, ishlatilgan barcha **activity** sinflarini, xizmatlarini va boshqalarni ro'yxatdan o'tkazadi. Ushbu faylni loyihada **manifests** papkasida topish mumkin:



```
app
  manifests
    AndroidManifest.xml
  java
    com.example.viewapp
      MainActivity
    com.example.viewapp (andr
    com.example.viewapp (test)
  java (generated)
  res
  res (generated)
Gradle Scripts
```

5.4-rasm. Loyiha strukturasida AndroidManifest.xml fayli

Manifest fayli quyidagicha ko'rinishi mumkin:

*<?xml version="1.0" encoding="utf-8"?>*
*<manifest xmlns:android="http://schemas.android.com/apk/res/android"*
  *package="com.example.viewapp">*
  *<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/Theme.ViewApp">
        <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>
```

Ildiz darajasi elementi manifest tugunidir. Bunday holda, faqat dasturlar to'plami aniqlanadi - **package = "com.example.viewapp"**. Aslida bu standart manifest faylining ta'rifi. Har holda, dasturlar to'plami farq qilishi mumkin, bo'sh **activity** bilan loyiha yaratishda tarkibning qolgan qismi bir xil bo'ladi.

Ilova qatlami parametrlarining aksariyati **application** elementi tomonidan belgilanadi. Atributlar yordamida bir qator sozlamalar ko'rsatilgan. Quyidagi atributlar sukut bo'yicha qo'llaniladi:

**android: allowBackup** dasturning zaxira nusxasini yaratilishini belgilaydi. **android:allowBackup = "true"** zaxira nusxasini yaratishga imkon beradi.

**android:icon** dastur belgisini o'rnatadi. Agar android qiymati: **icon="@mipmap/ic_launcher"** bo'lsa, dastur belgisi **res/mipmap** katalogidan olinadi

**android:roundIcon** dastur uchun yumaloq belgini o'rnatadi. Shuningdek, **res/mipmap** katalogidan olingan

**android:label** mobil qurilmada ko'rsatiladigan dastur nomini dasturlar ro'yxatida va sarlavhasida o'rnatadi. Bunday holda, u string resurslarida saqlanadi - **android:label="@string/app_name"**.

**android: supportRtl** turli xil **RTL API**-lardan foydalanish mumkinligini belgilaydi - o'ng tomonga yo'naltirilgan matn yo'nalishi bilan ishlash uchun maxsus **API** (masalan, arab yoki fors tillari uchun).

**android:theme** dastur uchun mavzuni belgilaydi. Mavzular keyinroq batafsil muhokama qilinadi, ammo hozircha mavzu dasturning umumiy uslubini belgilashini bilish kifoya. **"@style/Theme.ViewApp"** "**Theme.ViewApp**"mavzusini **res/values/themes** katalogidan oladi

Ichki o'rnatilgan **activity** elementlari dasturda ishlatiladigan barcha harakatlarni belgilaydi. Bunday holda, dasturda faqat bitta **activity** mavjudligini ko'rishingiz mumkin - **MainActivity**.

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

**MainActivity**dagi **intent-filter** elementi ushbu **activity** qanday ishlatilishini belgilaydi. Xususan, **actionandroid:name="android.intent.action.MAIN"** tugunidan foydalanib, ushbu **activity** dasturga kirish nuqtasi bo'lishi va tashqaridan hech qanday ma'lumot olmasligi kerak.

**Category android:name="android.intent.category.LAUNCHER"** elementi. **MainActivity** dastur boshlanganda ko'rsatiladigan boshlang'ich ekranni namoyish etishini bildiradi.

Manifest faylida ko'plab atributlarga ega bo'lgan ko'plab elementlar bo'lishi mumkin. Va barcha mumkin bo'lgan elementlarni va ularning xususiyatlarini hujjatlarda topish mumkin. Bu erda foydalanishning ba'zi bir misollari keltirilgan.

**Versiyasini aniqlash.** Ilova versiyasini va uning kodini aniqlash uchun manifest elementining atributlaridan foydalanish mumkin:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.viewapp"
    android:versionName="1.0"
    android:versionCode="1">
<!-- qolgan tarkibi -->
</manifest>
```

**android:versionName** atributi foydalanuvchiga ko'rsatiladigan versiya raqamini va dastur bilan ishlashda qaysi foydalanuvchilarni boshqarishini ko'rsatadi.

**android:versionCode** atributi ichki foydalanish uchun versiya raqamini anglatadi. Ushbu raqam faqatgina ilovaning bitta versiyasi versiyasi pastroq bo'lgan versiyasiga nisbatan yangi ekanligini ko'rsatadi. Ushbu raqam foydalanuvchilarga ko'rsatilmaydi.

Agar xohlasak, biz manbadagi versiyani ham aniqlay olamiz va bu erda manbaga murojaat qilishimiz mumkin.

**SDK versiyasini o'rnatish.** Android sdk versiyasini boshqarish uchun manifest faylida **<uses-sdk>** elementi aniqlanadi. U quyidagi atributlardan foydalanishi mumkin:

- **minSdkVersion**: minimal qo'llab-quvvatlanadigan SDK versiyasi
- **targetSdkVersion**: optimal versiya
- **maxSdkVersion**: maksimal versiya

Versiya API raqami bilan belgilanadi, masalan Jelly Beans 4.1 versiyasi 16, Android 11 versiyasi 30:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.viewapp"
    android:versionName="1.0"
    android:versionCode="1">
    <uses-sdk                                   android:minSdkVersion="22"
android:targetSdkVersion="30" />
```

```
<!--qolgan tarkibi-->
</manifest>
```

**Ruxsatlarni sozlash**. Ba'zan ilovaga kontaktlar, kamera va boshqalar kabi ba'zi manbalarga kirish uchun ruxsat kerak. Ilova bir xil kontaktlar ro'yxati bilan ishlashi uchun manifest faylida tegishli ruxsatlar o'rnatilishi kerak. Ruxsatlarni o'rnatish uchun **<uses-permission>** elementi ishlatiladi:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.viewapp">
  <uses-permission
android:name="android.permission.READ_CONTACTS" />
  <uses-permission          android:name="android.permission.CAMERA"
android:maxSdkVersion="30" />
<!-- qolgan tarkibi-->
</manifest>
```

**android: name** atributi ruxsat nomini belgilaydi: bu holda kontaktlar ro'yxatini o'qish va kameradan foydalanish. Ixtiyoriy ravishda, sdk-ning maksimal versiyasini **android: maxSdkVersion** atributi orqali o'rnatishingiz mumkin, bu API raqamini oladi.

**Turli xil qarorlarni qo'llab-quvvatlash.** Android qurilmalari dunyosi juda parchalangan, ham kichik ekranli gadjetlar, ham katta keng ekranli televizorlar mavjud. Va ba'zi ekran o'lchamlari uchun dasturdan foydalanishni cheklashingiz kerak bo'lgan holatlar mavjud. Buning uchun manifest faylida **<supports-screens>** elementi aniqlanadi:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.viewapp">
  <supports-screens
    android:largeScreens="true"
    android:normalScreens="true"
    android:smallScreens="false"
    android:xlargeScreens="true" />
<!-- qolgan tarkibi-->
</manifest>
```

Ushbu element to'rtta xususiyatni oladi:

- ✓ android: largeScreens - diagonali 4,5 dan 10 gacha bo'lgan ekranlar
- ✓ android: normalScreens - diagonali 3 dan 4,5 gacha bo'lgan ekranlar
- ✓ android: smallScreens - diagonali 3 dan kam bo'lgan ekranlar
- ✓ android: xlargeScreens - 10 dan kattaroq ekranlar

Agar atribut true-ga o'rnatilsa, u holda ekran tegishli ekran o'lchamlari bilan qo'llab-quvvatlanadi

**Yo'nalishni o'zgartirishni taqiqlash**. Ilova, gadjetning holatiga qarab, landshaft va portret yo'nalishida bo'lishi mumkin. Bu har doim ham qulay emas. Gadjetning aylanishidan qat'i nazar, biz dasturni faqat bitta yo'nalishni ishlata olamiz. Buning uchun **android:screenOrientation** atributi manifest faylidagi kerakli activity uchun o'rnatilishi kerak. Masalan, albom oriyentatsiyasini o'chirib qo'yamiz:

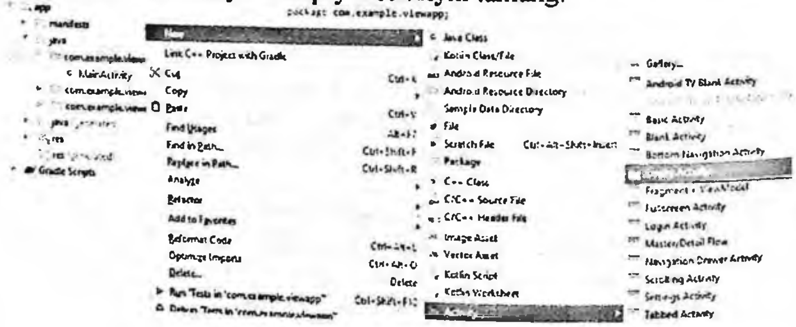
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.viewapp"
  android:versionName="1.0"
  android:versionCode="1" >
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.ViewApp">
    <activity android:name=".MainActivity"
      android:screenOrientation="portrait">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category
android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

**android:screenOrientation="portrait"** bu faollik faqat portret yo'nalishda bo'lishini bildiradi. Agar siz faqat landshaft yo'nalishini o'rnatishingiz kerak bo'lsa, unda **android:screenOrientation ="landscape"** qiymatidan foydalanishingiz kerak

## 5.3. Intentga kirish. Activity ishga tushirilmoqda.

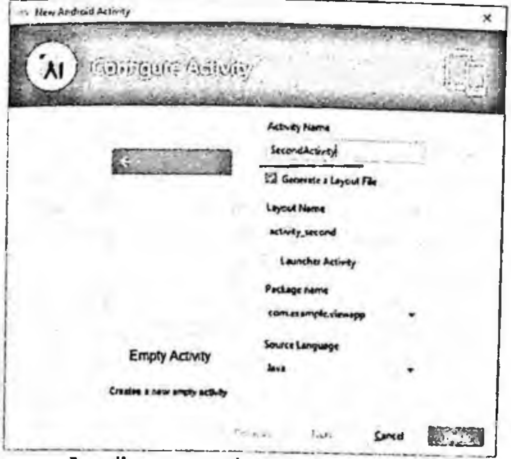
Turli xil ob'yektlar o'rtasidagi aloqa uchun asosiy sinf **android.content.Intent**. Bu dastur bajarishi kerak bo'lgan vazifani anglatadi.

**Intent** bilan ishlash uchun yangi **Activity** sinfini qo'shaylik. Buning uchun **MainActivity** sinfi joylashgan papkani o'ng tugmasini bosing va keyin kontekst menyusida **New-> Activity-> Empty Activity**ni tanlang:



5.5-rasm. Loyihaga yangi Activity sinfini qo'shish

Biz yangi **Activity** sinfga **SecondActivity** nomini beramiz va boshqa barcha sozlamalarni sukut bo'yicha qoldiramiz:



New Android Activity

Configure Activity

Activity Name: SecondActivity

☑ Generate a Layout File

Layout Name: activity_second

Launcher Activity

Package name: com.example.viewapp

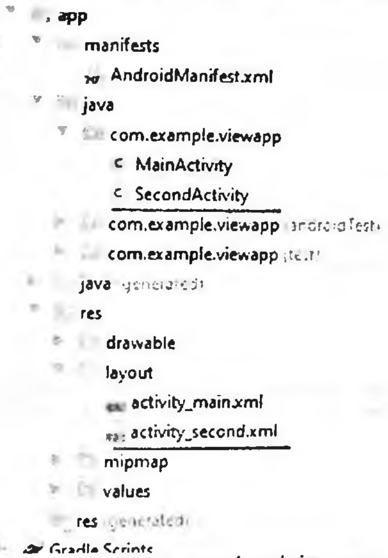
Source Language: Java

Empty Activity

Creates a new empty activity

5.6-rasm. Loyihaga yangi Activity sinfiga nom berish

Bundan keyin loyihaga yangi **Activity** - **SecondActivity** qo'shiladi:



```
app
  manifests
    AndroidManifest.xml
  java
    com.example.viewapp
      MainActivity
      SecondActivity
    com.example.viewapp (androidTest)
    com.example.viewapp (test)
  java (generated)
  res
    drawable
    layout
      activity_main.xml
      activity_second.xml
    mipmap
    values
  res (generated)
Gradle Scripts
```

5.7-rasm. Loyiha strukturasida yangi activity_second.xml fayli

Shundan so'ng, **AndroidManifest.xml** manifest faylida biz quyidagi qatorlarni topishimiz mumkin:

```
<activity android:name=".SecondActivity"></activity>
<activity
  android:name=".MainActivity">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
```

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

Barcha ishlatiladigan **activity** sinflari **<activity>** elementi yordamida **AndroidManifest.xml** faylida tavsiflanishi kerak. Har bir bunday element sinf nomini **activity**ga o'rnatadigan kamida bitta **android:name** atributini o'z ichiga oladi.

Biroq, mohiyatan **activity** - bu **Activity** sinfidan yoki uning avlodlaridan meros bo'lib o'tadigan standart java sinflari. Shuning uchun **Android Studio**da o'rnatilgan shablonlar o'rniga biz oddiy sinflarni qo'shib, keyin **Activity** sinfidan meros olishimiz mumkin. Ammo, bu holda siz **manifest** fayliga **activity** haqidagi ma'lumotlarni qo'lda qo'shishingiz kerak bo'ladi.

Bundan tashqari, **MainActivity** uchun **intent-filter** elementida **intent** filtri aniqlanadi. Unda "**android.intent.action.MAIN**" qiymatiga ega bo'lgan harakat elementi dasturga kirishning asosiy nuqtasini aks ettiradi. Ya'ni, **MainActivity** asosiy bo'lib qoladi va standart dastur tomonidan ishga tushiriladi.

**SecondActivity** uchun shunchaki uning loyihada ekanligi ko'rsatilgan va buning uchun **intent** filtrlari o'rnatilmagan. **SecondActivity**ni **MainActivity**dan boshlash uchun **startActivity()** usulini chaqirishingiz kerak:

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```

**Intent** ob'ekti **startActivity** uslubiga parametr sifatida uzatiladi. Uni yaratish uchun **Intent** konstruktorda ikkita parametrni oladi: bajarish konteksti (bu holda, hozirgi **MainActivity** ob'ekti) va **Intent** ob'ekti foydalanadigan va topshiriqqa uzatilgan ma'lumotlarni ifodalovchi sinf (aslida **activity** sinfi) , biz uni ishga tushiramiz).

Endi bir **Activity**dan ikkinchisiga o'tishni amalga oshirishni ko'rib chiqamiz. Buning uchun **activity_main.xml** faylidagi tugmani belgilang (ya'ni **MainActivity** interfeysida):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/navButton"
        android:textSize="20sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Перейти к SecondActivity"
        android:onClick="onClick"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

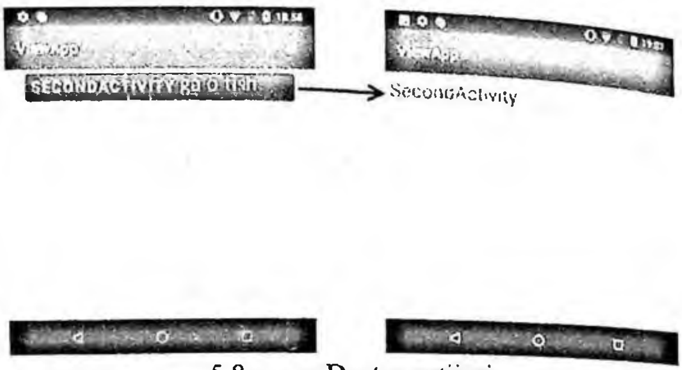
Va biz **MainActivity** sinfidagi tugmachani bosish uchun ishlov beruvchini aniqlaymiz, u yangi **Activity**ga o'tish uchun ishlatiladi:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void onClick(View view) {
      Intent intent = new Intent(this, SecondActivity.class);
      startActivity(intent);
   }
}
```

**SecondActivity** klik ishlov beruvchisida ishga tushiriladi.
Keyin **SecondActivity** kodini o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class SecondActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      //setContentView(R.layout.activity_second);
      TextView textView = new TextView(this);
      textView.setTextSize(20);
      textView.setPadding(16, 16, 16, 16);
      textView.setText("SecondActivity");
      setContentView(textView);
   }
}
```

Ilovani ishga tushiramiz va bitta **Activity**dan boshqasiga o'tamiz:



5.8-rasm. Dastur natijasi

## 5.4. Ma'lumotlarni Activity o'rtasida uzatish. Serializatsiya.

Ikki **Activity** o'rtasida ma'lumotlarni uzatish uchun **Intent** ob'ekti ishlatiladi. **putExtra()** usuli orqali siz kalitni va unga bog'liq qiymatni qo'shishingiz mumkin.

Masalan, "hello" tugmachasi bilan "Hello World" satrini **SecondActivityga** o'tish.

```
// SecondActivity-ni boshlash uchun Intent ob'ektini yarating
Intent intent = new Intent(this, SecondActivity.class);
// ob'ektni "hello" tugmachasi va "Hello World" qiymati bilan o'tkazish
intent.putExtra("hello", "Hello World");
// SecondActivity ni ishga tushirish
startActivity(intent);
```

Ma'lumotlarni uzatish uchun **putExtra()** usuli qo'llaniladi, bu sizga eng oddiy turdagi ma'lumotlarni - **String, int, float, double, long, short, byte, char,** ushbu turdagi massivlar yoki ob'ekt sifatida uzatishga imkon beradi. Serializable interfeysi.

**SecondActivityni** yuklashda yuborilgan ma'lumotlarni olish uchun siz ob'ekt tugmachasini olgan **get()** usulidan foydalanishingiz mumkin:

```
Bundle arguments = getIntent().getExtras();
String name = arguments.get("hello").toString();   // Hello World
```

Yuborilayotgan ma'lumotlar turiga qarab, biz ularni olganda **Bundle** ob'ektida bir qator usullardan foydalanishimiz mumkin. Ularning barchasi parametr sifatida ob'ekt kalitini oladi. Ulardan asosiylari:

- get(): Object tipidagi qiymatni qaytaradigan umumiy usul. Shunga ko'ra, ushbu qiymatni olish sohasi kerakli turga o'tkazilishi kerak
- getString (): String turidagi ob'ektni qaytaradi
- getInt (): int qiymatini qaytaradi
- getByte (): byte turidagi qiymatni qaytaradi
- getChar (): char qiymatini qaytaradi
- getShort (): short tipidagi qiymatni qaytaradi
- getLong (): long turidagi qiymatni qaytaradi
- getFloat (): float turidagi qiymatni qaytaradi
- getDouble (): double qiymatini qaytaradi
- getBoolean (): boolean turidagi qiymatni qaytaradi

➢ getCharArray (): char ob'ektlarining massivini qaytaradi
➢ getIntArray (): int ob'ektlarining massivini qaytaradi
➢ getFloatArray (): float ob'yektlarining massivini qaytaradi
➢ getSerializable (): Serializable interfeysi ob'ektini qaytaradi

Bizning loyihamizda ikkita faoliyat belgilangan: MainActivity SecondActivity.

SecondActivity kodida biz ma'lumotlarni olishni aniqlaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class SecondActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView textView = new TextView(this);
    textView.setTextSize(26);
    textView.setPadding(16, 16, 16, 16);
    Bundle arguments = getIntent().getExtras();
    if(arguments!=null){
      String name = arguments.get("name").toString();
      String company = arguments.getString("company");
      int age = arguments.getInt("age");
      textView.setText("Name: " + name + "\nCompany: " + company +
          "\nAge: " + age);
    }
    setContentView(textView);
  }
}
```

Bunday holda, **SecondActivity**da biz **Bundle** ob'ektidan barcha ma'lumotlarni olamiz va **TextView** matn maydonida namoyish qilamiz. Ushbu **activity** uchta element - **name** va **company** tugmachalari bilan ikkita satr va **price** tugmachasi bilan raqam o'tkaziladi deb taxmin qilinadi.

Keling, ma'lumotlarning **SecondActivity**ga uzatilishini aniqlaylik. Masalan, **activity_main.xml** faylida **MainActivity** uchun quyidagi interfeysni aniqlaylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/nameLabel"
    android:layout_width="0dp"
    android:layout_height="20dp"
    android:text="Name:"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
  <EditText
    android:id="@+id/name"
    android:layout_width="0dp"
    android:layout_height="40dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/nameLabel"/>
  <TextView
    android:id="@+id/companyLabel"
    android:layout_width="0dp"
    android:layout_height="20dp"
    android:text="Company:"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/name"/>
  <EditText
    android:id="@+id/company"
    android:layout_width="0dp"
    android:layout_height="40dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/companyLabel" />
  <TextView
    android:id="@+id/ageLabel"
    android:layout_width="0dp"
    android:layout_height="20dp"
    android:text="Age:"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/company"/>
  <EditText
    android:id="@+id/age"
    android:layout_width="0dp"
    android:layout_height="40dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/ageLabel"/>
  <Button
    android:id="@+id/btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

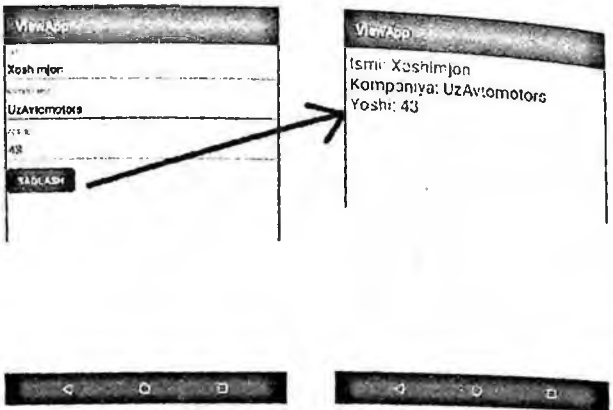
```
        android:onClick="onClick"
        android:text="Save"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/age"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Ma'lumotlarni kiritish uchun uchta matn maydonini va tugmani belgilaydi. **MainActivity** sinfida biz quyidagi tarkibni aniqlaymiz:

```
package com.example.viewapp:
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  public void onClick(View v) {
    EditText nameText = findViewById(R.id.name);
    EditText companyText = findViewById(R.id.company);
    EditText ageText = findViewById(R.id.age);
    String name = nameText.getText().toString();
    String company = companyText.getText().toString();
    int age = Integer.parseInt(ageText.getText().toString());
    Intent intent = new Intent(this, SecondActivity.class);
    intent.putExtra("name", name);
    intent.putExtra("company", company);
    intent.putExtra("age", age);
    startActivity(intent);
  }
}
```

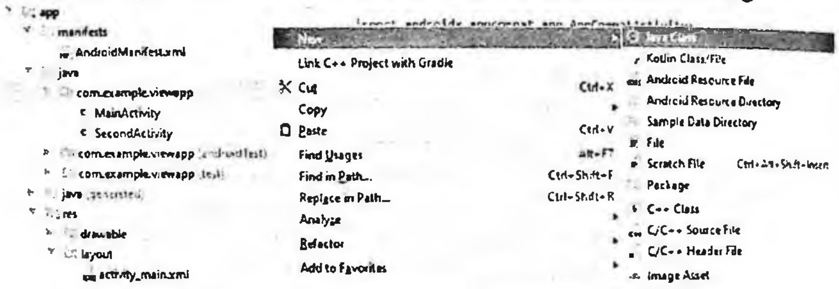
Tugmachani bosish moslamasida biz **EditText** matn maydonlariga kiritilgan ma'lumotlarni olamiz va **putExtra()** usuli yordamida **Intent** ob'ektiga uzatamiz. Keyin **SecondActivity**ni ishga tushiramiz.

Natijada, tugmani bosganingizda, matn maydonlariga kiritilgan ba'zi ma'lumotlarni oladigan **SecondActivity** ishga tushiriladi.



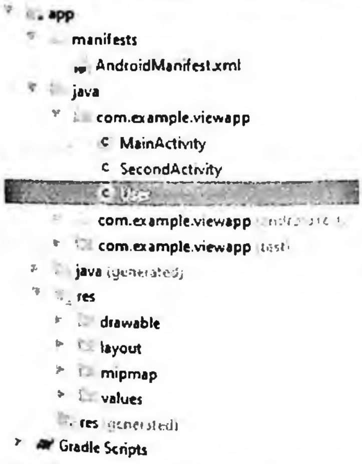
5.9-rasm. Dastur natijasi

**Murakkab ob'ektlarni o'tkazish.** Yuqoridagi misolda oddiy ma'lumotlar raqamlar, satrlar uzatildi. Ammo biz yanada murakkab ma'lumotlarni uzatishimiz mumkin. Bunday holda ketma-ketlashtirish mexanizmi qo'llaniladi. Buning uchun **MainActivity** va **SecondActivity** sinflari joylashgan paketlar papkasini bosing, o'ng tugmasini bosing va kontekst menyusida **New -> Java Classni** tanlang:



5.10-rasm. Loyihaga yangi Java sinfini qo`shish

Keling, yangi sinfni **User** deb ataymiz - u foydalanuvchini namoyish etsin.



5.11-rasm. Loyiha strukturasida User.java sinfi

User sinfida quyidagi kod mavjud bo'lsin:

```
package com.example.viewapp;
import java.io.Serializable;
public class User implements Serializable {
   private String name;
   private String company;
   private int age;
   public User(String name, String company, int age){
      this.name = name;
      this.company = company;
      this.age = age;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getCompany() {
      return company;
   }
   public void setCompany(String company) {
      this.company = company;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
}
```

Shuni ta'kidlash kerakki, ushbu sinf **Serializable** interfeysini amalga oshiradi. Endi **MainActivity** kodini o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
```

```
    public void onClick(View v) {
        EditText nameText = findViewById(R.id.name);
        EditText companyText = findViewById(R.id.company);
        EditText ageText = findViewById(R.id.age);
        String name = nameText.getText().toString();
        String company = companyText.getText().toString();
        int age = Integer.parseInt(ageText.getText().toString());
        User user = new User(name, company, age);
        Intent intent = new Intent(this, SecondActivity.class);
        intent.putExtra(User.class.getSimpleName(), user);
        startActivity(intent);
    }
}
```

Endi uchta tarqoq ma'lumotlar o'rniga bitta **User** ob'ekti uzatiladi. **User.class.getSimpleName()** usulining natijasi kalit sifatida ishlatiladi, bu sinf nomini asosan qaytaradi.

Va **SecondActivity** sinfini o'zgartiramiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_second);
        TextView textView = new TextView(this);
        textView.setTextSize(26);
        textView.setPadding(16, 16, 16, 16);
        Bundle arguments = getIntent().getExtras();
        User user;
        if(arguments!=null){
            user                    =                    (User)
arguments.getSerializable(User.class.getSimpleName());
            textView.setText("Name: " + user.getName() + "\nCompany: " +
user.getCompany() +
                    "\nAge: " + String.valueOf(user.getAge()));
        }
        setContentView(textView);
    }
}
```

**getSerializable**() usuli ma'lumotlar olish uchun ishlatiladi, chunki User sinfi **Serializable** interfeysini amalga oshiradi. Shunday qilib, biz turli xil ma'lumotlar to'plamining o'rniga bitta bitta ob'ektni o'tkaza olamiz.

## 5.5. Parcelable interfeysini

Ob'ektlarni seriyalash qobiliyati to'g'ridan-to'g'ri Java tilining ramkasi tomonidan ta'minlanadi. Biroq, Android shuningdek **Parcelable** interfeysini taqdim etadi, bu asosan ob'ektlarni **Serializable** singari seriyalashga imkon beradi, ammo Android uchun yanada optimallashtirilgan. Va shunga o'xshash **Parcelable** ob'yektlari ikkita **activity** o'rtasida o'tkazilishi yoki boshqa usulda ishlatilishi mumkin.

Masalan, oldingi satrda ma'lumotlar ketma-ketlikni ishlatadigan **User** ob'yektlari sifatida **activity** o'rtasida uzatilgan. Endi **User** sinfiga **Parcelable** interfeysini qo'llasin:

```
package com.example.viewapp;
import android.os.Parcel;
import android.os.Parcelable;
public class User implements Parcelable {
    private String name;
    private String company;
    private int age;
    public static final Creator<User> CREATOR = new Creator<User>() {
        @Override
        public User createFromParcel(Parcel source) {
            String name = source.readString();
            String company = source.readString();
            int age = source.readInt();
            return new User(name, company, age);
        }
        @Override
        public User[] newArray(int size) {
            return new User[size];
        }
    };
    public User(String name, String company, int age){
        this.name = name;
        this.company = company;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getCompany() {
        return company;
    }
    public void setCompany(String company) {
        this.company = company;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public int describeContents() {
        return 0;
    }
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeString(company);
        dest.writeInt(age);
    }
}
```

**android.os.Parcelable** interfeysi ikkita usulning bajarilishini o'z ichiga oladi: **descriptionContents()** va **writeToParcel()**. Birinchi usul tarkibni tavsiflaydi va ba'zi bir sonlarni qaytaradi. Ikkinchi usul **Parcel** ob'ektiga **User** ob'ekti tarkibini yozadi.

Ob'ekt ma'lumotlarini **Parcel**ga yozish uchun bir qator usullardan foydalaniladi, ularning har biri ma'lum bir ma'lumot turiga mo'ljallangan. Asosiy usullar:

- writeString ()
- writeInt ()
- writeFloat ()
- writeDouble ()
- writeByte ()
- writeLong ()
- writeIntArray ()
- writeValue () (Object tipidagi ob'ektni yozadi)
- writeParcelable () (Parcelable turidagi ob'ektni yozadi)

Bundan tashqari, **Parcelable** ob'ekti **Creator <User>** ob'ektini ifodalovchi statik **CREATOR** maydonini o'z ichiga olishi kerak. Bundan tashqari, ushbu ob'ekt ikkita usulni amalga oshiradi. Ular asl foydalanuvchi ob'ektlarining avvalgi seriyalangan ma'lumotlarini yaratish uchun kerak.

Shunday qilib, **newArray()** usuli **User** ob'ekti massivini yaratadi.

**CreateFromParcel** usuli **Parcel**dan yangi **User** ob'ektini yaratadi. Ya'ni, bu usul **writeToParcel** uslubiga teskari. Ma'lumotlarni **Parcel**dan olish uchun ma'lum turdagi ma'lumotlarni o'qish uchun **readString(), readInt(), readParcelable()** va boshqalar kabi usullardan foydalaniladi.

Bundan tashqari, **CreateFromParcel**dagi ma'lumotlar **Parcel** ob'ektidan **writeToParcel** usulida ushbu ob'ektga qo'shilish tartibida o'qilishi muhimdir.

Aytaylik, **SecondActivity** deb nomlangan **activity**da biz **User** ob'ektini olamiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class SecondActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView textView = new TextView(this);
    textView.setTextSize(26);
    textView.setPadding(16, 16, 16, 16);
    Bundle arguments = getIntent().getExtras();
    User user;
    if(arguments!=null){
      user = arguments.getParcelable(User.class.getSimpleName());
      textView.setText("Name: " + user.getName() + "\nCompany: " +
user.getCompany() +
              "\nAge: " + String.valueOf(user.getAge()));
    }
    setContentView(textView);
  }
}
```

**Parcelable**ni **activity**ga o'tkazishda **getParcelable()** usuli ishlatiladi. Va hech qanday turdagi kasting talab qilinmaydi.

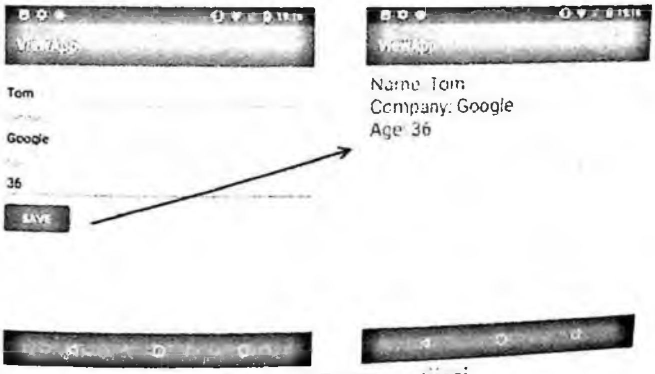
**Parcelable** transferini sinab ko'rish uchun **activity_main.xml** faylida **MainActivity** uchun eng oddiy interfeysni aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/nameLabel"
    android:layout_width="0dp"
    android:layout_height="20dp"
    android:text="Name:"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
  <EditText
```

```
    android:id="@+id/name"
    android:layout_width="0dp"
    android:layout_height="40dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/nameLabel"/>
<TextView
    android:id="@+id/companyLabel"
    android:layout_width="0dp"
    android:layout_height="20dp"
    android:text="Company:"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/name"/>
<EditText
    android:id="@+id/company"
    android:layout_width="0dp"
    android:layout_height="40dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/companyLabel" />
<TextView
    android:id="@+id/ageLabel"
    android:layout_width="0dp"
    android:layout_height="20dp"
    android:text="Age:"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/company"/>
<EditText
    android:id="@+id/age"
    android:layout_width="0dp"
    android:layout_height="40dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/ageLabel"/>
<Button
    android:id="@+id/btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="Save"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/age"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

**MainActivity** kodida biz **SecondActivity**ga ma'lumotlarni uzatishni aniqlaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  public void onClick(View v) {
    EditText nameText = findViewById(R.id.name);
    EditText companyText = findViewById(R.id.company);
    EditText ageText = findViewById(R.id.age);
    String name = nameText.getText().toString();
    String company = companyText.getText().toString();
    int age = Integer.parseInt(ageText.getText().toString());
    User user = new User(name, company, age);
    Intent intent = new Intent(this, SecondActivity.class);
    intent.putExtra(User.class.getSimpleName(), user);
    startActivity(intent);
  }
}
```



Name: Tom
Company: Google
Age: 36

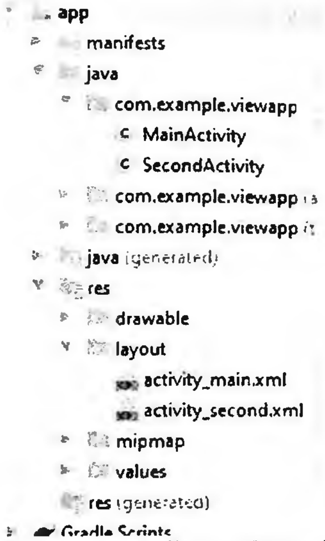
5.12-rasm. Dastur natijasi

### 5.6. Activity natijasini olish

Oxirgi mavzuda siz yangi **Activity**ni qanday chaqirishni va unga ba'zi lumotlarni uzatishni ko'rdingiz. Ammo biz nafaqat ma'lumotni ishga tushirilgan

**activity**ga o'tkazibgina qolmay, balki undan qandaydir ish natijalarini kutishimiz mumkin.

Masalan, loyihamizda ikkita **activity** mavjud deylik: **MainActivity** va **SecondActivity**. Va har bir **activity** o'z interfeys fayliga ega: navbati bilan **activity_main.xml** va **activity_second.xml**.



```
app
  manifests
  java
    com.example.viewapp
      MainActivity
      SecondActivity
    com.example.viewapp
    com.example.viewapp
  java (generated)
  res
    drawable
    layout
      activity_main.xml
      activity_second.xml
    mipmap
    values
  res (generated)
Gradle Scripts
```

5.13-rasm. Loyiha strukturasi

Oxirgi satrda **startActivity()** usuli yordamida yangi a_ctivity chaqirdik. Boshlangan faoliyat natijasini olish uchun **startActivityForResult(Intent** intent, **int requestCode)** usulidan foydalanishingiz kerak. Ushbu usul ikkita parametrni oladi: **Intent** boshlangan **activity**ga ma'lumotlarni uzatadi va ikkinchi parametr, **requestCode**, butun sonli so'rov kodiga ishora qiladi. Keling, uning ishlatilishini misol bilan ko'rib chiqaylik.

Shunday qilib, **activity_main.xml** faylida quyidagi interfeysni aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/textView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Укажите возраст"
    android:textSize="22sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
  <EditText
    android:id="@+id/age"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"/>
  <Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="Отправить"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/age"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Ma'lumotlarni kiritish uchun bu erda **EditText** elementi va yuborish tugmasi aniqlangan. **MainActivity** sinfida ikkinchi **activity** ishga tushirilishini aniqlaylik:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  static final String AGE_KEY = "AGE";
  static final String ACCESS_MESSAGE="ACCESS_MESSAGE";
  private static  final int REQUEST_ACCESS_TYPE=1;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  public void onClick(View view) {
    // Kiritilgan yosh qiymatini olish
    EditText ageBox = (EditText) findViewById(R.id.age);
    String age = ageBox.getText().toString();
    Intent intent = new Intent(this, SecondActivity.class);
    intent.putExtra(AGE_KEY, age);
    startActivityForResult(intent, REQUEST_ACCESS_TYPE);
  }
  @Override
  public void onActivityResult(int requestCode, int resultCode, Intent data){
    TextView textView = (TextView) findViewById(R.id.textView);
```

```
        if(requestCode==REQUEST_ACCESS_TYPE){
            if(resultCode==RESULT_OK){
                String                    accessMessage
data.getStringExtra(ACCESS_MESSAGE);
                textView.setText(accessMessage);
            }
            else{
                textView.setText("Kirishda xatolik bo`ldi");
            }
        }
        else{
            super.onActivityResult(requestCode, resultCode, data);
        }
    }
}
```

OnClick(.) tugmachasini bosish moslamasida biz matn maydoniga kiritilgan yoshni olamiz, uni AGE_KEY tugmachasi bilan Intent ob'ektiga qo'shamiz va startActivityForResult usuli yordamida **SecondActivity**ni ishga tushiramiz. Bundan tashqari, raqamli so'rov kodi REQUEST_ACCESS_TYPE doimiyligini anglatadi. Natijada qaysi raqamni o'tkazish juda muhim emas, ammo bu raqamdan foydalanib, biz SecondActivity-dan olingan javobni qayta ishlashimiz mumkin, ayniqsa, har xil vaziyatlarda bir nechta raqamli so'rov kodlari ishlatilgan bo'lsa.

Natijani SecondActivitydan olish va qayta ishlash uchun onActivityResult usulini bekor qilishingiz kerak. Ushbu usul uchta parametrni oladi:

- requestCode: startActivityForResult uchun ikkinchi parametr sifatida yuborilgan so'rovning raqamli kodi
- resultCode: raqamli natija kodi. Odatda, natijada o'rnatilgan RESULT_OK va RESULT_CANCELED doimiylaridan foydalaniladi.
- data: SecondActivity-dan MainActivity-ga yuborilgan ma'lumotlar

Bu holda onActivityResult () usuli olingan ma'lumotlarni TextView elementiga chiqaradi. Keyin, SecondActivity ga o'tamiz va activity_second.xml faylidagi tugmalar to'plamini aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/ageView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
  <Button
    android:id="@+id/button1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Kirish ochiq"
    android:onClick="onButton1Click"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/ageView"/>
  <Button
    android:id="@+id/button2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text=" Kirishni rad etish"
    android:onClick="onButton2Click"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button1"/>
  <Button
    android:id="@+id/button3"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text=" Yoshi noto'g'ri "
    android:onClick="onButton3Click"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button2" />
  <Button
    android:id="@+id/cancel"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Отмена"
    android:onClick="onCancelClick"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button3" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**SecondActivity** sinfida biz ushbu tugmalar uchun ishlov beruvchilarni aniqlaymiz:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
public class SecondActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_second);
      Bundle extras = getIntent().getExtras();
      if (extras != null) {
         TextView ageView = (TextView) findViewById(R.id.ageView);
         String age = extras.getString(MainActivity.AGE_KEY);
         ageView.setText("Yoshi: " + age);
      }
   }
   public void onCancelClick(View v) {
      setResult(RESULT_CANCELED);
      finish();
   }
   public void onButton1Click(View v) {
      sendMessage("Kirish uchun ruxsat berilgan ");
   }
   public void onButton2Click(View v) {
      sendMessage("Ruxsat berilmadi");
   }
   public void onButton3Click(View v) {
      sendMessage("Noto'g'ri yosh ");
   }
   private void sendMessage(String message){
      Intent data = new Intent();
      data.putExtra(MainActivity.ACCESS_MESSAGE, message);
      setResult(RESULT_OK, data);
      finish();
   }
}
```

Uchta tugma yuborilgan javobni yuboradigan sendMessage() usulini chaqiradi. Bu onActivityResult usulida MainActivity oladigan xabar bo'ladi.

Natijani qaytarish uchun **setResult()** usulini chaqirishingiz kerak, u ikkita parametrdan iborat:

raqamli natija kodi

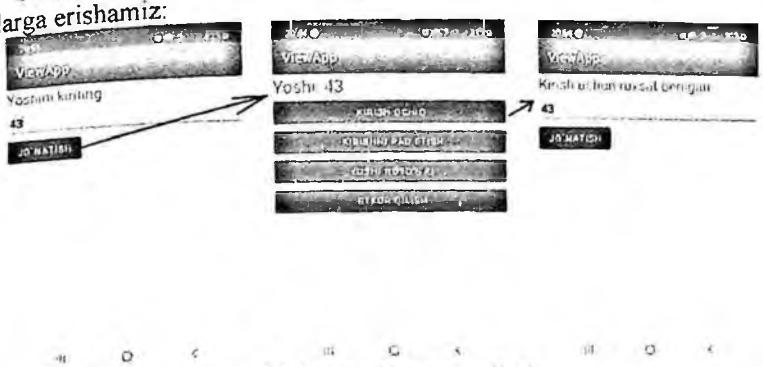
ma'lumotlar yuborildi

SetResult () usulini chaqirgandan so'ng, **finish** usulini chaqirishingiz kerak, bu esa joriy activity ni yo'q qiladi.

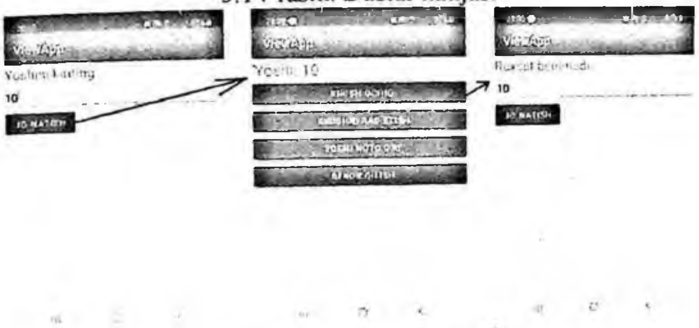
Bitta tugma onCancelClick () ishlov beruvchisini chaqiradi, unda faqat natija kodi - RESULT_CANCELED setResultga uzatiladi.

Ya'ni, nisbatan gapirganda, biz SecondActivity da MainActivity ga kiritilgan yoshni olamiz va ma'lum bir tugmachani bosib xabarning ba'zi natijalarini qaytaramiz.

SecondActivity da bosilgan tugmachaga qarab, biz MainActivity da har xil natijalarga erishamiz:



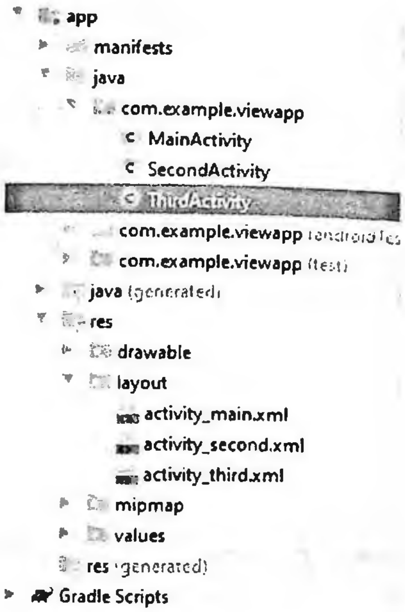
5.14-rasm. Dastur natijasi



5.15-rasm. Dastur natijasi

## 5.7. Activity o'rtasidagi o'zaro bog'liqlik

Oldingi mavzularda Intent ob'ekti yordamida activity va yangi activity boshlash davrlarini yoritdik. Endi bitta dasturda activity o'rtasidagi o'zaro ta'sirning ba'zi xususiyatlarini ko'rib chiqamiz. Aytaylik, bizda uchta activity bor: MainActivity, SecondActivity va ThirdActivity.



5.16-rasm. Loyiha strukturasi 3-Activity "activity_third.xml" fayli

Intentdan foydalanib, masalan, MainActivity tugmachasini bosib SecondActivityni ishga tushiradi:

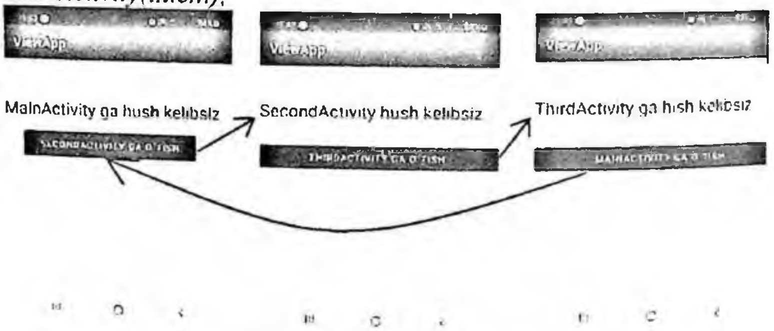
*Intent intent = new Intent(this, SecondActivity.class);*
*startActivity(intent);*

SecondActivityda ThirdActivityni ishga tushiradigan tugma mavjud:

*Intent intent = new Intent(this, ThirdActivity.class);*
*startActivity(intent);*

FirstActivityda birinchi activity - MainActivityga qaytadigan tugma mavjud

*Intent intent = new Intent(this, MainActivity.class);*
*startActivity(intent);*



5.17-rasm. Dastur natijasi

Agar biz barcha ketma-ketlikni ketma-ket ishga tushiradigan bo'lsak: asosiy MainActivitydan SecondActivityni ishga tushiramiz, SecondActivity-ThirdActivitydan, natijada biz quyidagi activity to'plamiga ega bo'lamiz:

*ThirdActivity*
*SecondActivity*
*MainActivity*

Agar bundan keyin biz MainActivity-dan ThirdActivity-ga murojaat qilmoqchi bo'lsak, u holda startActivity () usuli yangi MainActivity ob'ektini ishga tushiradi (va mavjudiga qaytmaydi) va stek allaqachon shunday ko'rinadi:

MainActivity
ThirdActivity
SecondActivity
MainActivity

Ya'ni, bizda MainActivityning ikkita mustaqil nusxasi bo'ladi. Agar mavjud bo'lgan holatga o'tishni istasak, bu holat istalmagan. Va bu momentni hisobga olish kerak.

Agar biz **Back(Orqaga)** tugmachasini bosgan bo'lsak, u holda stackning yuqori qismida joylashgan joriy activity stackdan chiqariladi va oldingi activity stackning yuqori qismida joylashgan bo'lib, o'z ishini davom ettiradi. Va shunday qilib, Back(Orqaga) tugmachasini ishlatib, stekdagi oldingi activityga o'tishimiz mumkin. Masalan, yuqoridagi holatda, agar biz Orqaga tugmachasini bosgan bo'lsak, u holda stakning yuqori qismidagi MainActivity o'z ishini tugatadi va ThirdActivity ekranda ko'rsatila boshlaydi.

ThirdActivity
SecondActivity
MainActivity

Biroq, ba'zida activity o'rtasidagi o'tishni boshqarish kerak bo'ladi. Masalan, bu holda, biz birinchi ishga tushirilgan va stekning pastki qismida joylashgan MainActivityga o'tish o'rniga, uchinchi tugmachadagi tugmachani bosganimizda MainActivityning yangi nusxasini ishga tushirishni xohlamaymiz. Keling, Android bizga qanday imkoniyatlar yaratishini ko'rib chiqaylik.

**activity stack boshqarish.** Stekni biron bir faoliyatdan boshqarish uchun Android **Intent** sinfida belgilangan bayroqlar - doimiylardan foydalanishni taklif qiladi. Muayyan bayroqni qo'llash bizga ma'lum bir activity uchun stekdagi holatni ma'lum bir tarzda o'zgartirishga imkon beradi.

Masalan, avvalgi vazifani olaylik, qachonki ThreeActivitydagi tugmachani bosgandan so'ng, MainActivityning yangi nusxasi ishga tushirilsa. Ammo biz yangisini ishga tushirishni xohlamaymiz, balki mavjudiga o'ting.

MainActivity
ThirdActivity
SecondActivity
MainActivity

Ushbu vaziyatdan chiqish uchun biz **Intent.FLAG_ACTIVITY_REORDER_TO_FRONT** bayrog'idan foydalanishingiz mumkin:

```
Intent intent = new Intent(this, MainActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
startActivity(intent);
```

**Intent.FLAG_ACTIVITY_REORDER_TO_FRONT** bayroqchasi agar u allaqachon stakda bo'lsa, stackning yuqori qismiga o'tish uchun activity harakatini amalga oshiradi. Va bu holda, ThirdActivitydan MainActivityga o'tgandan so'ng, stek shunday bo'ladi:

MainActivity
ThirdActivity
SecondActivity

Agar biz Back tugmachasini ishlatib orqaga qaytgandek, shunchaki ThirdActivity-dan MainActivityga o'tishimiz kerak bo'lsa, unda bi **Intent.FLAG_ACTIVITY_CLEAR_TOP** v **Intent.FLAG_ACTIVITY_SINGLE_TOP** bayroqlarini ishlatamiz:

```
        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(intent);
```

**Intent.FLAG_ACTIVITY_CLEAR_TOP** bayrog'i boshlanadiganidan tashqari barcha activityni o'chiradi (agar u allaqachon stakda bo'lsa). Va **Intent.FLAG_ACTIVITY_SINGLE_TOP** bayrog'i shuni ko'rsatadiki, agar stackning yuqori qismida allaqachon boshlash kerak bo'lgan activity mavjud bo'lsa, u ishlamaydi (keyin u to'plamda faqat bitta shaklda mavjud bo'lishi mumkin).

Bunday holda, ThirdActivitydan MainActivityga o'tgandan so'ng, stek to'liq tozalanadi va u erda faqat MainActivity qoladi.

Yana bitta bayroq - **Intent.FLAG_ACTIVITY_NO_HISTORY** ishga tushirilgan activityni stakka saqlamaydi. Masalan, SecondActivityni ishga tushirishda biz uni stakda saqlashni xohlamaymiz:

```
        Intent intent = new Intent(this, SecondActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
        startActivity(intent);
```

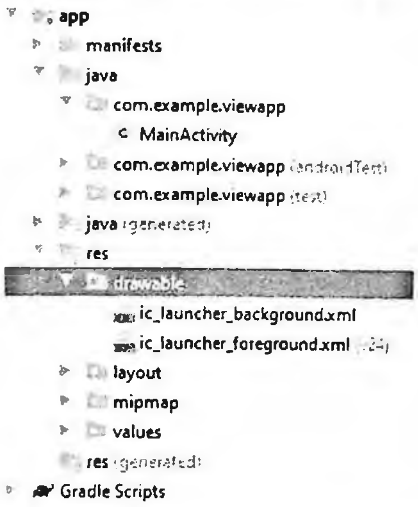
Bunday holda, MainActivity -> SecondActivity -> ThirdActivity zanjiri bo'ylab harakatlanayotganda stek quyidagi ko'rinishga ega bo'ladi:

MainActivity
ThirdActivity

# VI-BOB. TASVIRLAR BILAN ISHLASH

## 6.1. Rasm resurslari

Rasm fayllari resurslarning eng keng tarqalgan resurslaridan biridir. Android quyidagi fayl formatlarini qo'llab-quvvatlaydi: *.png*(afzal qilingan), *.jpg*(maqbul), *.gif*(eskirgan). **res/drawable** papkasi loyihadagi grafik fayllar uchun sukut bo'yicha allaqachon yaratilgan. Odatiy bo'lib, u allaqachon bir nechta fayllarni o'z ichiga oladi - bir nechta piktogramma fayllari:



```
app
  manifests
  java
    com.example.viewapp
      MainActivity
    com.example.viewapp (androidTest)
    com.example.viewapp (test)
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
    mipmap
    values
  res (generated)
Gradle Scripts
```

6.1-rasm. Loyiha strukturasida drawable papkasi

Ushbu papkaga grafik fayllarni qo'shishda Android ularning har biri uchun Drawable resursini yaratadi. Shundan so'ng biz Java kodida resursga quyidagicha kirishimiz mumkin:

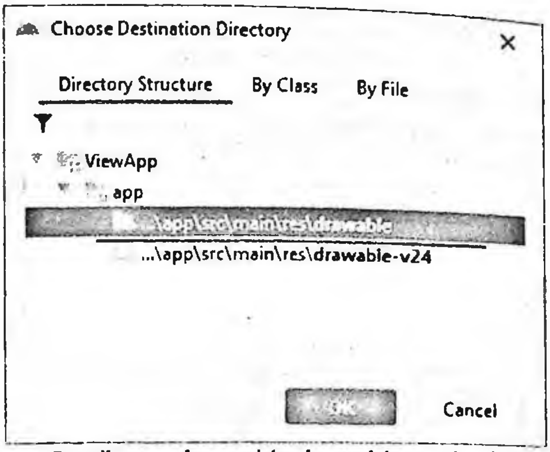
*R.drawable.fayl_nomi*

Yoki xml kodida:

*@[Paket_nomi:]drawable/fayl_nomi*

Masalan, **res/drawable** papkasida loyihaga rasm faylini qo'shamiz. Buning uchun qattiq diskdagi png yoki jpg kengaytmasi bo'lgan faylni nusxa oling va **res/drawable** papkasiga joylashtiring (loyihaga nusxalash uchun oddiy Copy-Paste ishlatiladi)

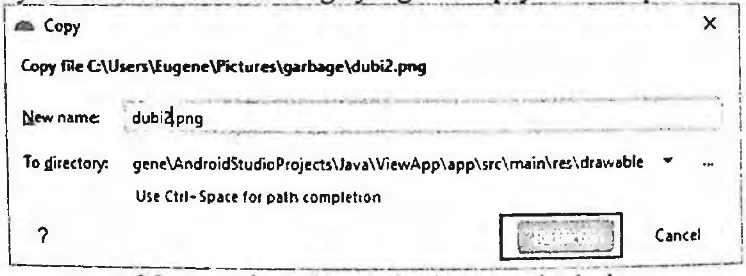
Keyin bizdan papkani tanlash talab qilinadi - **drawable** yoki **drawable-24**. Oddiy rasm fayllarini qo'shish uchun **drawable**ni tanlang:


Choose Destination Directory
Directory Structure
By Class
By File
ViewApp
app
...\app\src\main\res\drawable
...\app\src\main\res\drawable-v24
OK
Cancel

6.2-rasm. Loyiha strukturasida drawable papkasini tanlash.

Bu erda darhol rasm fayli dasturga qo'shilishini va shu bilan uning hajmini oshirishni o'ylash kerak. Bundan tashqari, katta rasmlar ishlashga salbiy ta'sir qiladi. Shuning uchun kichik va optimallashtirilgan (siqilgan) grafik fayllardan foydalanish yaxshiroqdir. Shunga qaramay, shuni ta'kidlash kerakki, ushbu papkaga qo'shilgan barcha rasm fayllari loyihani tuzish paytida **aapt** yordam dasturi yordamida avtomatik ravishda optimallashtirilishi mumkin. Bu sifatni yo'qotmasdan fayl hajmini kamaytirishga imkon beradi.

Faylni nusxalashda bizdan unga yangi nom qo'yish talab qilinadi.



Copy file C:\Users\Eugene\Pictures\garbage\dubi2.png

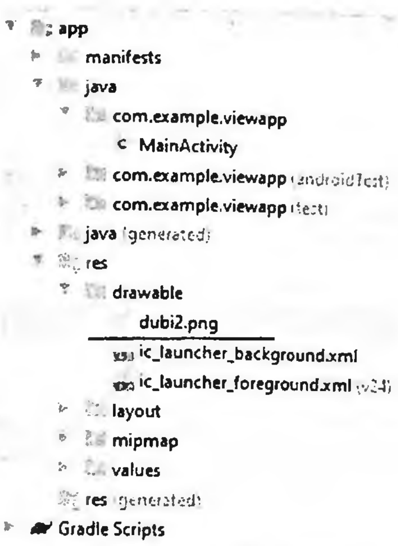
New name: dubi2.png

To directory: gene\AndroidStudioProjects\Java\ViewApp\app\src\main\res\drawable

Use Ctrl+Space for path completion

6.2-rasm. drawable papkasiga rasm joylash

Siz fayl nomini o'zgartirishingiz yoki uni qanday bo'lsa, shunday qoldirishingiz mumkin. Mening vaziyatimda fayl **dubi2.png** deb nomlanadi. Va keyin Refactor tugmachasini bosing. Va bundan keyin biz tanlagan rasm fayli drawable papkasiga qo'shiladi.



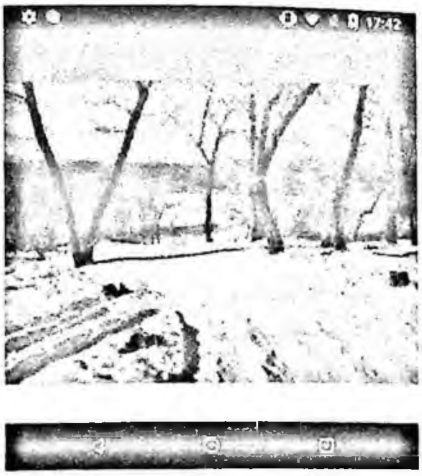
```
app
  manifests
  java
    com.example.viewapp
      MainActivity
    com.example.viewapp (androidTest)
    com.example.viewapp (test)
  java (generated)
  res
    drawable
      dubi2.png
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
    mipmap
    values
  res (generated)
Gradle Scripts
```

6.3-rasm. Loyiha strukturasida drawable papkasi

Android-da rasmlar bilan ishlash uchun turli xil elementlardan oydalanishingiz mumkin, ammo **ImageView** to'g'ridan-to'g'ri rasmlarni namoyish ilish uchun mo'ljallangan. Shuning uchun biz **activity_main.xml** faylini uyidagicha o'zgartiramiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/dubi2"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bunday holda, **android:src** atributi ImageViewda faylni ko'rsatish uchun elementga o'rnatiladi. Uning qiymati kengaytirilgan holda fayl nomi bilan bir xil bo'lgan grafik resurs nomini belgilaydi. Shundan so'ng, Previewda yoki Android Studioda dizaynerlik rejimida siz rasmning dasturini yoki dasturni ishga tushirishda ko'rishingiz mumkin:



6.4-rasm. Dastur natijasi

Agar biz java kodida ImageViewni yaratib, resursdan koddan foydalansak, u holda activity quyidagicha ko'rinishi mumkin:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.widget.ImageView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      //setContentView(R.layout.activity_main);
      ConstraintLayout constraintLayout = new ConstraintLayout(this);
      ImageView imageView = new ImageView(this);
      // biz resursdan foydalanamiz
      imageView.setImageResource(R.drawable.dubi2);
      ConstraintLayout.LayoutParams        layoutParams        =        ne
ConstraintLayout.LayoutParams
            (ConstraintLayout.LayoutParams.WRAP_CONTENT
ConstraintLayout.LayoutParams.WRAP_CONTENT);
         layoutParams.leftToLeft
ConstraintLayout.LayoutParams.PARENT_ID;
         layoutParams.topToTop
ConstraintLayout.LayoutParams.PARENT_ID;
         imageView.setLayoutParams(layoutParams);
         constraintLayout.addView(imageView);
         setContentView(constraintLayout);
      }
   }
```

Bunday holda, drawable resursi to'g'ridan-to'g'ri imageView.setImageResource() usuliga uzatiladi va shu bilan tasvir o'rnatiladi. Natijada, biz bir xil natijaga erishamiz.

*imageView.setImageResource(R.drawable.dubi2);*

Shu bilan birga, resursni ishlatishdan oldin uni qandaydir tarzda qayta ishlash yoki boshqa stsenariylarda ishlatish kerak bo'lishi mumkin. Bunday holda, avval uni Drawable ob'ekti sifatida olishimiz va undan keyin vazifalarimiz uchun foydalanishimiz mumkin:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.res.ResourcesCompat;
import android.content.res.Resources;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.widget.ImageView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.activity_main);
    ConstraintLayout constraintLayout = new ConstraintLayout(this);
    ImageView imageView = new ImageView(this);
    Resources res = getResources();
    Drawable drawable = ResourcesCompat.getDrawable(res,
R.drawable.dubi2, null);
    // biz resursdan foydalanamiz
    imageView.setImageDrawable(drawable);
    ConstraintLayout.LayoutParams layoutParams = new
ConstraintLayout.LayoutParams
        (ConstraintLayout.LayoutParams.WRAP_CONTENT ,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
    layoutParams.leftToLeft =
ConstraintLayout.LayoutParams.PARENT_ID;
    layoutParams.topToTop =
ConstraintLayout.LayoutParams.PARENT_ID;
    imageView.setLayoutParams(layoutParams);
    constraintLayout.addView(imageView);
    setContentView(constraintLayout);
  }
}
```

Resursni olish uchun Resources ob'ekti, resurs identifikatori va mavzusi uzatiladigan ResourcesCompat.getDrawable() usuli qo'llaniladi. Bunday holda, mavzu biz uchun muhim emas, shuning uchun biz unga null qiymatini beramiz. Resurs **Drawable** ob'ekti sifatida qaytariladi:

*Drawable drawable = ResourcesCompat.getDrawable(res, R.drawable.dubi2, null);*

Keyin, masalan, setImageDrawable() usuli orqali resursni ImageView ob'ektiga o'tkazishingiz mumkin.

*imageView.setImageDrawable(drawable);*

## 6.2. ImageView elementi

Avvalgi mavzuda siz ImageView boshqaruvidan foydalanib tasvirlarni qanday ko'rsatishni ko'rgansiz. Keling, ushbu element bilan ishlash bo'yicha qo'shimcha fikrlarni ko'rib chiqaylik.

ImageView elementining ba'zi asosiy atributlari:

- **android:cropToPadding**: true ga o'rnatilganda, tasvir o'rnatilgan maydonga muvofiq kesiladi.
- **android:scaleType**: tasvirni ImageView chegaralariga nisbatan qanday kattalashtirishni belgilaydi

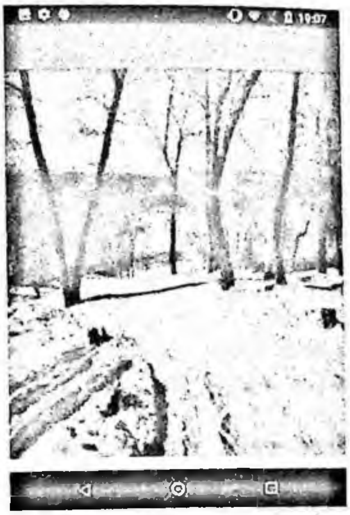
  Hisoblash qiymatlaridan biri o'lchov parametrlarini o'rnatish uchun ishlatiladi:
  - CENTER: Rasm markazlashtirilmagan holda markazlashtirilgan
  - CENTER_CROP: kenglik va balandlik orasidagi nisbatni saqlab turish uchun tasvir markazlashtirilgan va masshtablangan. Agar uning bir qismi ekranga to'g'ri kelmasa, u kesiladi
  - CENTER_INSIDE: tasvir kenglik va balandlik orasidagi nisbatni saqlab turish uchun markazlashtirilgan va masshtablangan, lekin kengligi va balandligi ImageView kengligi va balandligidan katta bo'lishi mumkin emas.
  - FIT_CENTER: Rasm kattalashtirilgan va markazlashtirilgan
  - FIT_START: Rasm element boshida joylashtiriladi (portret uchun yuqoriga va manzara uchun chapga)
  - FIT_END: tasvir elementning oxirigacha joylashtirilgan va joylashtirilgan (portret uchun pastda, manzara uchun o'ngda)
  - FIT_XY: Tasvir kengligi va balandligi orasidagi nisbatni saqlamasdan, butun ImageView maydonini to'ldirmasdan o'lchanadi.
  - MATRIX: tasvir matritsasi yordamida masshtablanadi
- **android:src**: rasm manbasi
- **android:alpha**: shaffoflikni o'rnatadi (qiymati 0,0 dan - to'liq shaffofdan 1,0 ga - to'liq ko'rinadigan)
- **android:tint**: tasvirni yopish uchun ishlatiladigan rang
- **android:tintMode**: tasvirlar ustki qatlamiga qo'llaniladigan rejim
- ImageView sinfining ba'zi asosiy usullari:
- **Drawable getDrawable()**: berilgan ImageView bilan bog'liq Drawable manbasini qaytaradi (yoki ImageView uchun manba o'rnatilmagan bo'lsa n_ull)

- **ImageView.ScaleType getScaleType()**: ImageView.ScaleType sonini qaytaradi, bu tasvirni ImageView boshqaruvi chegaralariga nisbatan qanday o'lchanganligini ko'rsatadi.
- **void setImageDrawable(Drawable drawable)**: Drawable obyekti yordamida tasvir manbasini o'rnatadi
- **void setImageResource (int resId)**: Drawable resurs identifikatori yordamida tasvir manbasini o'rnatadi
- **void setImageURI (Uri uri)**: tasvir manbasini ushbu manbaning Uri yordamida o'rnatadi
- **void setScaleType(ImageView.ScaleType scaleType)**: tasvirning o'lchamini o'rnatadi
- **void setImageAlpha (int alpha)**: tasvirning shaffofligini o'rnatadi - qiymat 0,0 dan 1,0 gacha

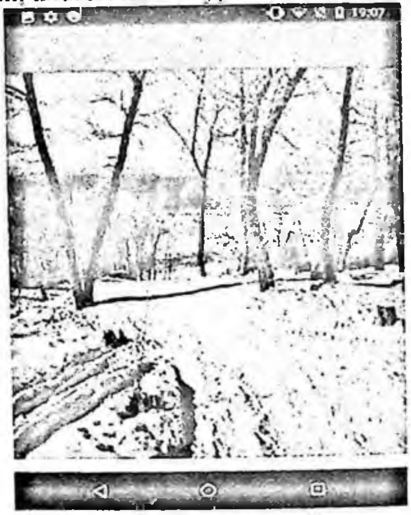
Masalan, activity_main.xml faylida android: scaleType atributi uchun FIT_XY qiymatini o'rnatish:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/dubi2"
    android:scaleType="fitXY"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Natijada, rasm vertikal va gorizontal ravishda cho'ziladi:


6.5-rasm. Dastur natijasi

Taqqoslash uchun, android:scaleType ="center" bilan o'xshash misol:


6.6-rasm. Dastur natijasi

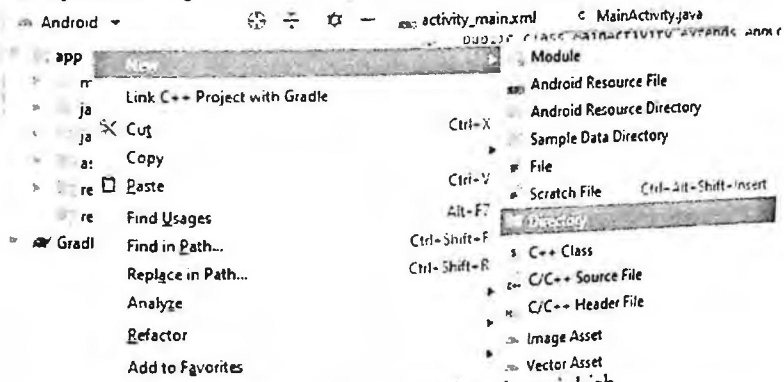
Java kodidagi o'xshash misol:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.widget.ImageView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        ConstraintLayout constraintLayout = new ConstraintLayout(this);
        ImageView imageView = new ImageView(this);
        imageView.setImageResource(R.drawable.dubi2);
        // o'lchov(mas'shtab)ni sozlash
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        ConstraintLayout.LayoutParams          layoutParams        =        new
ConstraintLayout.LayoutParams
                (ConstraintLayout.LayoutParams.WRAP_CONTENT                    ,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        layoutParams.leftToLeft                                                =
ConstraintLayout.LayoutParams.PARENT_ID;
        layoutParams.topToTop                                                  =
ConstraintLayout.LayoutParams.PARENT_ID;
        imageView.setLayoutParams(layoutParams);
        constraintLayout.addView(imageView);
        setContentView(constraintLayout);
    }
}
```

## 6.3. assets papkasidagi rasmlar

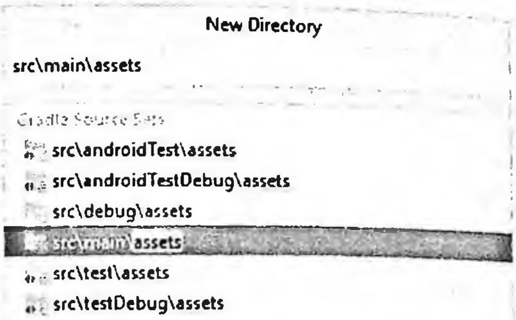
O'tgan mavzularda loyihadagi rasmlar res/drawables papkasida resurs sifatida joylashtirilgan va ImageView boshqaruvida aks ettirilgan. Biroq, rasmlarni bu papkaga joylashtirish printsipial jihatdan shart emas. Fayllarni assets papkasida ham joylashtirish mumkin. Keling, bunday rasmli fayllar bilan qanday ishlashni ko'rib chiqaylik.

Birinchidan, assets papkasini loyihaga qo'shing. Buning uchun Android Studio -da **app** katalogini bosing va paydo bo'ladigan kontekst menyusida **New->Directory** ni tanlang:



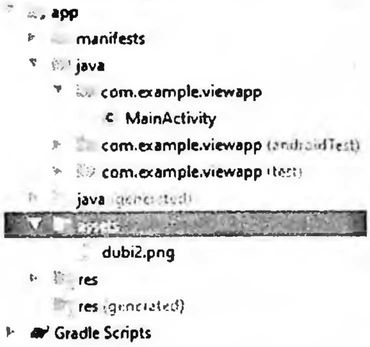
6.7-rasm. Loyihada yangi papka qo'shish

Keyin, paydo bo'lgan oynada **src\main\assets** bandini tanlang va uni loyihaga qo'shish uchun Enter ni bosing:



6.8-rasm. Loyihada yangi assets papka tarkibi

Keling, ushbu papkaga rasm qo'shamiz:



6.9-rasm. Loyihaga yangi assets papkasi

ImageView elementi **activity_main.xml** faylida aniqlansin:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <ImageView
      android:id="@+id/image"
      android:layout_width="0dp"
      android:layout_height="0dp"
      android:layout_margin="16dp"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

assets papkasidagi tasvirni **MainActivity** dagi ImageView elementiga yuklang:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.widget.ImageView;
import java.io.IOException;
import java.io.InputStream;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ImageView imageView = (ImageView) findViewById(R.id.image)
    String filename = "dubi2.png";
    InputStream inputStream = null;
    try{
      inputStream = getApplicationContext().getAssets().open(filename);
      Drawable drawable = Drawable.createFromStream(inputStream,
null);
      imageView.setImageDrawable(drawable);
      imageView.setScaleType(ImageView.ScaleType.FIT_XY);
    }
    catch (IOException e){
      e.printStackTrace();
    }
    finally {
      try{
        if(inputStream!=null)
          inputStream.close();
      }
      catch (IOException ex){
        ex.printStackTrace();
      }
    }
  }
}
```

Faylni yuklash uchun getApplicationContext().getAssets().Open(filename) ifodasi yordamida InputStream ni olish kerak.

Drawable.createFromStream (inputStream, null) ga murojaat qilish kirish oqimidan Drawable ob'ektini chiqaradi.

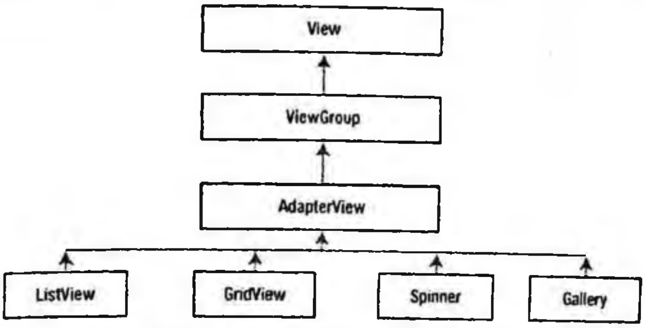
ImageView.setImageDrawable(d) usuli DrawVable ni ImageView ga yuklaydi.

6.10-rasm. Dastur natijasi

# VII-BOB. ADAPTERLAR VA RO'YXATLAR

## 7.1. ListView va ArrayAdapter

Android ro'yxatlarni aks ettiruvchi elementlarning keng palitrasini taqdim etadi. Ularning barchasi **android.widget.AdapterView** sinfidan meros bo'lib qolgan. Bu ListView, GridView, Spinner kabi vidjetlar. Ular boshqa boshqaruv elementlari uchun konteyner vazifasini o'tashi mumkin.



7.1-rasm. Boshqaruv elementlari uchun konteynerlar

Ro'yxatlar bilan ishlashda biz uchta komponent bilan ishlaymiz. Birinchidan, bu ekrandagi ro'yxatni (ListView, GridView) ifodalovchi va ma'lumotlarni ko'rsatadigan vizual element yoki vidjet. Ikkinchisi - bu ma'lumot manbai - massiv, ArrayList ob'ekti, ma'lumotlar bazasi va boshqalar. Uchinchidan, bu adapter - ma'lumotlar manbasini ro'yxat vidjetiga ulaydigan maxsus komponent.

Eng oddiy va eng keng tarqalgan ro'yxat elementlaridan biri **ListView** vidjetidir. Keling, ushbu adapterlardan biri - **ArrayAdapter** sinfidan foydalanib, ListView -ni ma'lumotlar manbasiga bog'lashni ko'rib chiqaylik.

**ArrayAdapter** sinfi - bu ma'lumotlar majmuasini TextView elementlari to'plami bilan bog'laydigan juda oddiy adapter, masalan, ListView tuzilishi mumkin. Ya'ni, bu holda ma'lumotlar manbai ob'ektlar majmuasidir. ArrayAdapter har bir ob'ektda toString () usulini satrga o'tkazishga chaqiradi va natijada paydo bo'lgan qatorni TextView elementiga o'rnatadi.

Keling, bir misolni ko'rib chiqaylik. Shunday qilib, ilovaning belgilanishi quyidagicha ko'rinishi mumkin:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <ListView
    android:id="@+id/countriesList"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </ListView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Shuningdek, u ob'ektlar ro'yxatini ko'rsatadigan ListView elementini belgilaydi. Keling, activity kodiga o'tamiz va ListView ni ArrayAdapter orqali ba'zi ma'lumotlar bilan bog'laymiz:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
public class MainActivity extends AppCompatActivity {
    // ro'yxatga havola qilish uchun ma'lumotlar to'plami
    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // ListView elementini oling
        ListView countriesList = (ListView) findViewById(R.id.countriesList);
        // adapter yarating
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
            android.R.layout.simple_list_item_1, countries);
        // ro'yxat uchun adapterni o'rnating
        countriesList.setAdapter(adapter);
    }
}
```

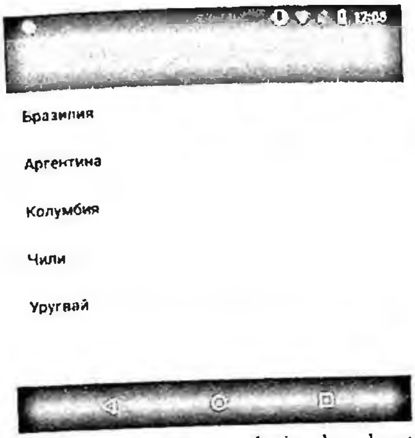
Bu erda, birinchi navbatda, id tomonidan ListView elementini olamiz va keyin unga adapter yaratamiz.

Adapterni yaratish uchun quyidagi ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, countries) konstruktori ishlatilgan.

- **this**: joriy faoliyat obyekti
- **android.R.layout.simple_list_item_1**: ramka sukut bo'yicha taqdim etadigan ro'yxatni belgilash fayli. U Android SDK papkasida platforms/[android-version_number]/data/res/layout yo'lida joylashgan. Agar biz ro'yxatning standart belgilashidan qoniqmasak, biz o'zimizni yaratishimiz mumkin, keyin kodda id ni kerakli belgining id ga o'zgartiramiz.
- **countries**: ma'lumotlar majmuasi. Bu erda qatorni ko'rsatish shart emas, bu ArrayList<T> bo'lishi mumkin.

Nihoyat, ListView uchun adapterni setAdapter() usuli yordamida o'rnatishingiz kerak.

Natijada biz quyidagi displeyni olamiz:


Бразилия

Аргентина

Колумбия

Чили

Уругвай

7.2-rasm. Boshqaruv elementlari uchun konteynerlar

## 7.2. string-array resursi va ListView

Oldingi mavzu, ListView -da ArrayAdapter yordamida qatorlar qatorini qanday ko'rsatishni o'z ichiga olgan. Bunday holda, satrlar qatori Java kodida dasturiy jihatdan aniqlangan. Biroq, bunday qatorlar qatorini xml faylida resurs sifatida saqlash ancha qulayroq bo'lardi.

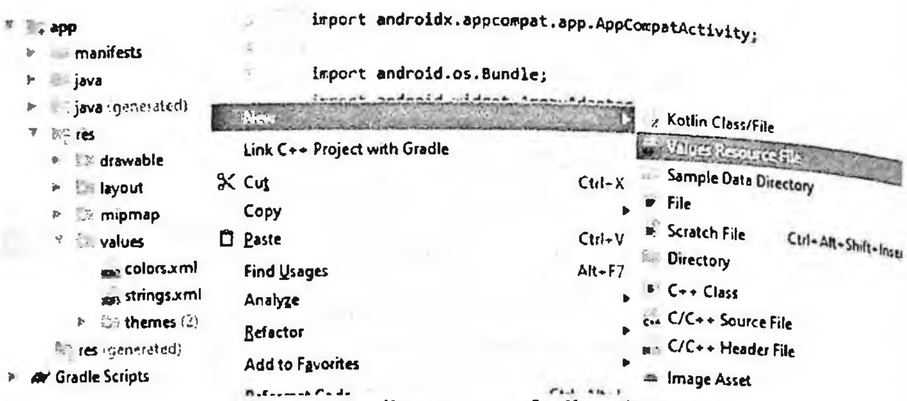
Satrli massiv resurslari **string-array** tipidagi elementni ifodalaydi. Ular ixtiyoriy nomli xml faylidagi **res/values** katalogida joylashishi mumkin.

Satrli massiv ta'riflari quyidagi sintaksisga ega:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="имя_массива_строк">
    <item>элемент</item>
  </string-array>
</resources>
```

Satrlar massivi <string-array> elementi yordamida ko'rsatiladi, uning nom atributi ixtiyoriy qiymatga ega bo'lishi mumkin va keyinchalik bu massivga murojaat qilish uchun ishlatiladi. Massivning barcha elementlari <item> qiymatlar to'plamini ifodalaydi

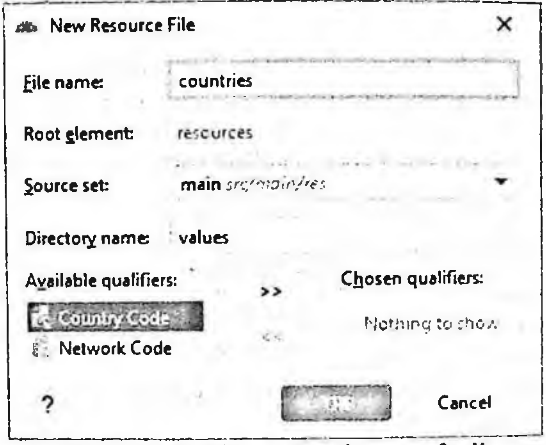
Masalan, **res/values** papkasiga yangi fayl qo'shamiz. Buning uchun ushbu katalogni o'ng tugmasini bosing va paydo bo'lgan menyudan **New->Value Resource file** elementini tanlang:



```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
```

7.3-rasm. Loyihaga resurs fayllar qiymatlari

Ko'rsatilgan oynada faylni countries deb nomlaymiz:



File name: countries
Root element: resources
Source set: main src/main/res
Directory name: values

7.4-rasm. Loyihaga yangi resurs fayli

Faylni res/values papkasiga qo'shgandan so'ng, uning mazmunini quyidagicha o'zgartiramiz:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string-array name="countries">
      <item>Бразилия</item>
      <item>Аргентина</item>
      <item>Колумбия</item>
      <item>Чили</item>
      <item>Уругвай</item>
   </string-array>
</resources>
```

**activity_main.xml** belgilash faylida ListView elementining ta'rifi qoladi:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ListView
      android:id="@+id/countriesList"
      android:layout_width="0dp"
      android:layout_height="0dp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent">
    </ListView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Keling, MainActivity kodidagi resurs va ListView ni bog'laylik:

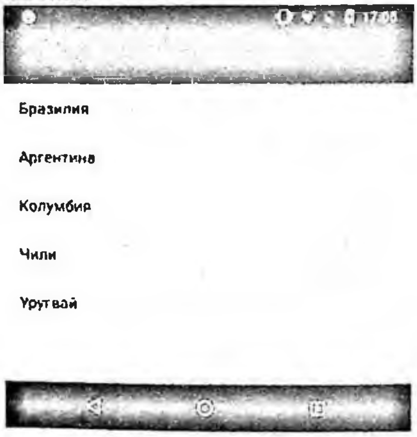
```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // ListView elementini oling
      ListView countriesList = (ListView) findViewById(R.id.countriesList);
      // manba olamiz
      String[]                              countries                              =
getResources().getStringArray(R.array.countries);
      // adapter yarating
      ArrayAdapter<String> adapter = new ArrayAdapter(this,
            android.R.layout.simple_list_item_1, countries);
      // ro'yxat uchun adapterni o'rnating
      countriesList.setAdapter(adapter);
   }
}
```

Java kodida manba olish uchun R.array.resurs_nomlanishi ifodasi ishlatiladi.

Lekin biz ListView ga dasturlar qatorini qo'shishimiz shart emas. Bu elementda string-array manbasini qiymat sifatida qabul qiladigan **entries** atributi mavjud:

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

```
    android:layout_height="match_parent">
    <ListView
        android:id="@+id/countriesList"
        android:entries="@array/countries"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </ListView>
</androidx.constraintlayout.widget.ConstraintLayout>
```
Bunday holda, biz MainActivity kodini standartga qisqartirishimiz mumkin:
```
package com.example.listapp;
import androidx.appcompat.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);
    }
}
```
Va natija bir xil bo'ladi:


```
Бразилия
Аргентина
Колумбия
Чили
Уругвай
```

7.5-rasm. Dastur natijasi

## 7.3. ListView da elementni tanlash

Oldingi mavzularda ma'lumotlarni ListViewga qanday yuklash va uni ma'lumotlar resursiga bog'lash mumkinligi ko'rib chiqilgan. ListView elementlar ro'yxatini ko'rsatish bilan bir qatorda, elementni tanlash va uning tanlovini boshqarishga imkon beradi. Keling, buni qanday qilishni ko'rib chiqaylik. **activity_main.xml** faylida quyidagi belgilashni aniqlaylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/selection"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
  <ListView
    android:id="@+id/countriesList"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintTop_toBottomOf="@+id/selection"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent">
  </ListView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Keling, ListView ni ma'lumotlar resurssiga bog'laymiz va unga ro'yxat elementini bosish uchun tinglovchini biriktiramiz:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // TextView elementini oling
    TextView selection = (TextView) findViewById(R.id.selection);
```

```
            // ListView elementini oling
            ListView countriesList = (ListView) findViewById(R.id.countriesList);
            // adapter yarating
            ArrayAdapter<String> adapter = new ArrayAdapter(this,
                 android.R.layout.simple_list_item_1, countries);
            // ro'yxat uchun adapterni o'rnating
            countriesList.setAdapter(adapter);
            // ro'yxatga tinglovchini qo'shing
            countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
               @Override
               public void onItemClick(AdapterView<?> parent, View v, int
position, long id)
               {
                  // pozitsiya bo'yicha biz tanlangan elementni olamiz
                  String selectedItem = countries[position];
                  // TextView boshqaruvining matnini sozlash
                  selection.setText(selectedItem);
               }
            });
         }
      }
```

Shunday qilib, setAdapter usuli ListView elementini ma'lum bir adapterga bog'laydi. Keyin, **OnItemClickListener** tinglovchisi ro'yxat elementini tanlash bilan shug'ullanadi. Bu tinglovchining bitta **onItemClick** usuli bor, uning yordamida biz tanlangan element va unga tegishli ma'lumotlarni olishimiz mumkin. Shunday qilib, u quyidagi parametrlarni oladi:

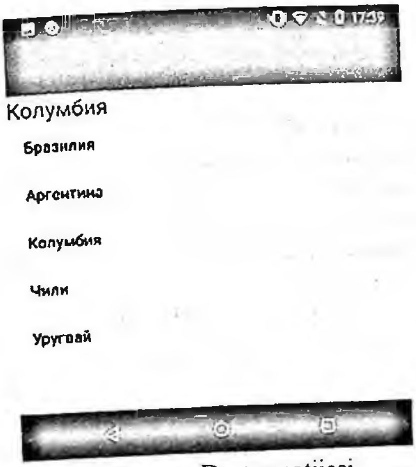
- **parent**: bosilgan AdapterView (bu holda bu bizning ListView)
- **view**: vidjetni AdapterView ichiga bosdi
- **position**: AdapterView ichidagi bosilgan vidjet indeksi
- **id**: bosilgan elementning qator identifikatori

Ushbu parametrlardan foydalanib, biz tanlangan elementni turli yo'llar bilan olishimiz mumkin.

Masalan, bu holda, satrlar qatoridagi element indeksiga mos keladigan bosilgan vidjet indeksini olsak, biz mos keladigan elementni massivlar qatoriga o'rnatamiz va shu tariqa uning matnini olamiz:

```
      countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
         @Override
         public void onItemClick(AdapterView<?> parent, View v, int position,
long id)
         {
            // pozitsiya bo'yicha biz tanlangan elementni olamiz
            String selectedItem = countries[position];
            // TextView boshqaruvining matnini sozlash
```

```
    selection.setText(selectedItem);
  }
});
```


Колумбия

Бразилия

Аргентина

Колумбия

Чили

Уругвай

7.6-rasm. Dastur natijasi

Shuningdek, biz tanlangan elementni AdapterViewdan olishimiz mumkin, u birinchi parametr sifatida beriladi - AdapterView <?>parent. Shunday qilib, bu holda, biz bilamizki, AdapterViewdagi har bir element aslida satr yoki String ob'ektini ifodalaydi, shuning uchun bu holda biz tanlangan elementni shunday olishimiz mumkin:

```
    countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
      @Override
      public void onItemClick(AdapterView<?> parent, View v, int position,
long id)
      {
        // tanlangan elementni oling
        String selectedItem = (String)parent.getItemAtPosition(position);
        // TextView boshqaruvining matnini sozlash
        selection.setText(selectedItem);
      }
    });
```

GetItemAtPosition usuli tanlangan elementni indeks bo'yicha qaytaradi. Agar biz Java kodida yaratilgan mavvisli qatorni ma'lumotlar resursi sifatida ishlatmasak, masalan, xml faylida ko'rsatilgan <string-array> resursidan foydalansak, bu tegishli bo'lishi mumkin.

Uchinchidan, biz ikkinchi parametr – View v sifatida berilgan tanlangan elementdan foydalanishimiz mumkin. Shunday qilib, bu holda, adapter markirovka turi sifatida **android.R.layout.simple_list_item_1** resursidan foydalanadi, ya'ni tanlangan element berilgan matnni ko'rsatadigan **TextView**ni ifodalaydi. Shuning uchun, bu holda, biz tanlangan elementni ham shunday olishimiz mumkin:

```
        countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position,
long id)
            {
              // tanlangan elementni oling
              TextView textView = (TextView) v;
              String selectedItem = (String)textView.getText();
              // установка текста элемента TextView
              selection.setText(selectedItem);
              // yoki shunday
              // selection.setText(textView.getText());
            }
        });
```

**Ro'yxatda bir nechta variant.** Ba'zida odatdagidek bir nechta elementni tanlash kerak bo'ladi, lekin bir nechta. Buning uchun, birinchi navbatda, **android:choiceMode="multipleChoice"** atributini ro'yxat belgilarida o'rnatish kerak:

```
        <?xml version="1.0" encoding="utf-8"?>
        <androidx.constraintlayout.widget.ConstraintLayout
          xmlns:android="http://schemas.android.com/apk/res/android"
          xmlns:app="http://schemas.android.com/apk/res-auto"
          android:layout_width="match_parent"
          android:layout_height="match_parent">
          <TextView
            android:id="@+id/selection"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:textSize="22sp"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
          <ListView
            android:id="@+id/countriesList"
            android:choiceMode="multipleChoice"
            android:layout_width="0dp"
            android:layout_height="0dp"
            app:layout_constraintTop_toBottomOf="@+id/selection"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent">
          </ListView>
        </androidx.constraintlayout.widget.ConstraintLayout>
```

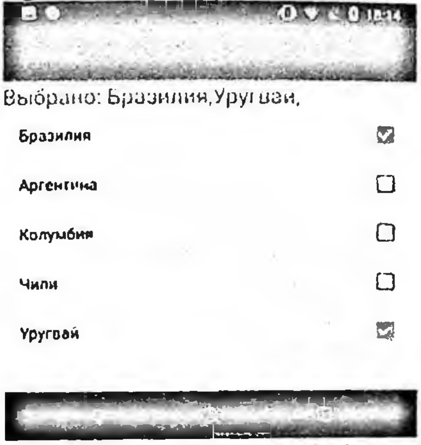
Keling, MainActivity kodida ro'yxat elementlarini tanlashni aniqlaylik:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.util.SparseBooleanArray;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // TextView elementini oling
      TextView selection = (TextView) findViewById(R.id.selection);
      // ListView elementini oling
      ListView countriesList = (ListView) findViewById(R.id.countriesList);
      // adapter yarating
      ArrayAdapter<String> adapter = new ArrayAdapter(this,
           android.R.layout.simple_list_item_multiple_choice, countries);
      // ro'yxat uchun adapterni o'rnating
      countriesList.setAdapter(adapter);
      // ro'yxatga tinglovchini qo'shing
      countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
        @Override
        public void onItemClick(AdapterView<?> parent, View v, int
position, long id)
        {
          SparseBooleanArray
selected=countriesList.getCheckedItemPositions();
          String selectedItems="";
          for(int i=0;i < countries.length;i++)
          {
            if(selected.get(i))
              selectedItems+=countries[i]+",";
          }
          // TextView boshqaruvining matnini sozlash
          selection.setText("Tanlangan: " + selectedItems);
        }
      });
    }
```

}

android.R.layout.simple_list_item_multiple_choice manbasi ko'p tanlovli ro'yxatni tuzish uchun framework tomonidan taqdim etilgan standart belgini ifodalaydi.

Va elementlarni tanlashda biz **SparseBooleanArray** ob'ektidagi barch tanlangan pozitsiyalarni olamiz, keyin biz butun massivni o'tamiz va aga massivdagi elementning o'rni SparseBooleanArrayda bo'lsa, ya'ni u belgilanga bo'lsa, biz satrga belgilangan element.



Выбрано: Бразилия,Уругвай,

Бразилия ✓

Аргентина ☐

Колумбия ☐

Чили ☐

Уругвай ✓

7.7-rasm. Dastur natijasi

## 7.4. ArrayAdapter va ListView ga qo'shish va olib tashlash

ListViewni adapter orqali ma'lumotlar manbaiga bog'lagandan so'ng, biz ma'lumotlar bilan ishlashimiz mumkin - faqat adapter orqali qo'shish, o'chirish, o'zgartirish. ListView faqat ma'lumotlarni ko'rsatish uchun mo'ljallangan.

Ma'lumotni boshqarish uchun biz adapter usullaridan yoki to'g'ridan -to'g'ri ma'lumotlar manbasidan foydalanishimiz mumkin. Masalan, ArrayAdapter sinfi ma'lumotlarni boshqarish uchun quyidagi usullarni taqdim etadi:

- **void add(T object)**: qator oxiriga ob'ekt elementini qo'shadi
- **void addAll(T ... items)**: barcha elementlarni massiv oxiriga qo'shadi
- **void addAll(Collection <? extends T> collection)**: massiv oxiriga collection elementlar to'plamini qo'shadi.
- **void clear()**: ro'yxatdagi barcha elementlarni olib tashlaydi
- **void insert (T object, int index)**: index indeksidagi qatorga object elementini qo'shadi.
- **void remove (T object)**: object elementini massivdan olib tashlaydi

Biroq, yuqoridagi usullar qo'llanilgandan so'ng, o'zgarishlar faqat ma'lumot manbai bo'lgan massivga ta'sir qiladi. O'zgarishlarni ListView bilan sinxronlashtirish uchun adapterning notifyDataSetChanged() usulini chaqiring.

Masalan, **activity_main.xml** faylida quyidagi elementlarni aniqlaylik:

*<?xml version="1.0" encoding="utf-8"?>*

*<androidx.constraintlayout.widget.ConstraintLayout*

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
      <EditText
        android:id="@+id/userName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintHorizontal_weight="4"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/add"
        app:layout_constraintTop_toTopOf="parent" />
      <Button
        android:id="@+id/add"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintHorizontal_weight="1"
        android:text="+"
        android:onClick="add"
        app:layout_constraintRight_toLeftOf="@+id/remove"
        app:layout_constraintLeft_toRightOf="@+id/userName"
        app:layout_constraintTop_toTopOf="parent"/>
      <Button
        android:id="@+id/remove"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintHorizontal_weight="1"
        android:text="-"
        android:onClick="remove"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/add"
        app:layout_constraintRight_toRightOf="parent" />
    <ListView
      android:id="@+id/usersList"
      android:layout_width="0dp"
      android:layout_height="0dp"
      android:choiceMode="multipleChoice"
      app:layout_constraintTop_toBottomOf="@+id/userName"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent">
    </ListView>
  </androidx.constraintlayout.widget.ConstraintLayout>
```

ListView elementlarni bir nechta tanlash imkoniyati bilan ro'yxatni ko'rsatish uchun mo'ljallangan. Qo'shish va o'chirish uchun ikkita tugma belgilangan. Ro'yxatga yangi ob'ekt kiritish uchun EditText maydonidan foydalaning.

Endi **MainActivity** sinfini o'zgartiraylik:

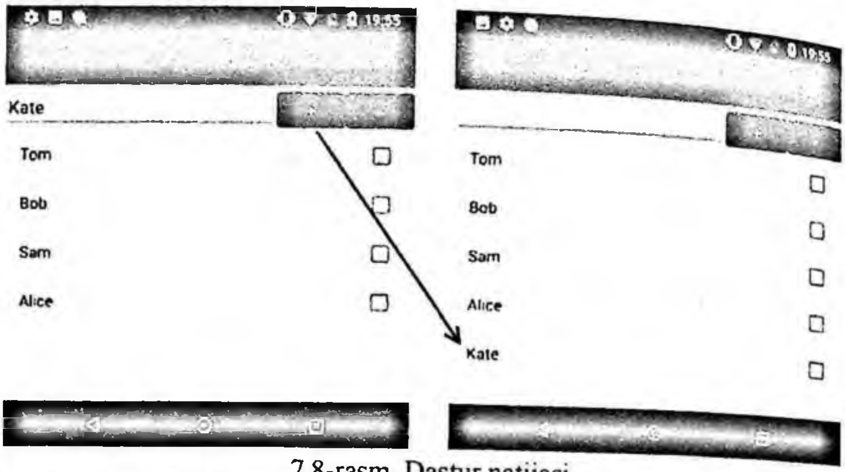
```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import java.util.ArrayList;
import java.util.Collections;
public class MainActivity extends AppCompatActivity {
  ArrayList<String> users = new ArrayList<String>();
  ArrayList<String> selectedUsers = new ArrayList<String>();
  ArrayAdapter<String> adapter;
  ListView usersList;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // добавляем начальные элементы
    Collections.addAll(users, "Tom", "Bob", "Sam", "Alice");
    // ListView elementini oling
    usersList = (ListView) findViewById(R.id.usersList);
    // adapter yaratamiz
    adapter          =          new          ArrayAdapter<String>(this,
android.R.layout.simple_list_item_multiple_choice, users);
    // ro'yxat uchun adapterni o'rnating
    usersList.setAdapter(adapter);
    // ro'yxatga qo'shish va tanlovni bekor qilish
    usersList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
      @Override
      public void onItemClick(AdapterView<?> parent, View v, int
position, long id)
      {
        // bosilgan elementni oling
        String user = adapter.getItem(position);
        if(usersList.isItemChecked(position))
          selectedUsers.add(user);
        else
          selectedUsers.remove(user);
      }
```

```
        });
    }
    public void add(View view){
        EditText userName = (EditText) findViewById(R.id.userName);
        String user = userName.getText().toString();
        if(!user.isEmpty()){
            adapter.add(user);
            userName.setText("");
            adapter.notifyDataSetChanged();
        }
    }
    public void remove(View view){
        // tanlangan elementlarni olish va o'chirish
        for(int i=0; i< selectedUsers.size();i++){
            adapter.remove(selectedUsers.get(i));
        }
        // oldindan o'rnatilgan barcha belgilarni olib tashlang
        usersList.clearChoices();
        // tanlangan ob'ektlar massivini tozalang
        selectedUsers.clear();
        adapter.notifyDataSetChanged();
    }
}
```

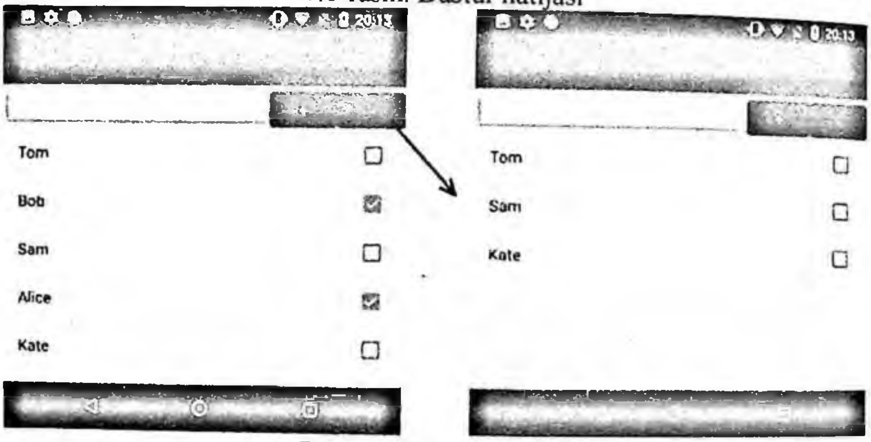
Qo'shish bilan hamma narsa nisbatan oddiy: biz kiritilgan satrni olamiz va **adapter.add()** usuli yordamida ro'yxatga qo'shamiz. **adapter.notifyDataSetChanged()** usuli qo'shilgandan so'ng ListView ni yangilash uchun chaqiriladi.

Va o'chirish uchun tanlangan elementlarni o'z ichiga olgan qo'shimcha Tanlangan Foydalanuvchilar ro'yxati tuziladi. Tanlangan elementlarni qabul qilish va ularni ro'yxatga qo'shish uchun **AdapterView.OnItemClickListener** tinglovchisi ishlatiladi, uning elementi tekshirilganda yoki belgilanmaganida, ya'ni element bosilganda chaqiriladigan onItemClick() usuli ishlatiladi.

O'chirish tugmasini bosgandan so'ng, tanlangan elementlar ro'yxatidan o'ting va ularning har biri uchun **adapter.remove()** usulini chaqiring.



7.8-rasm. Dastur natijasi



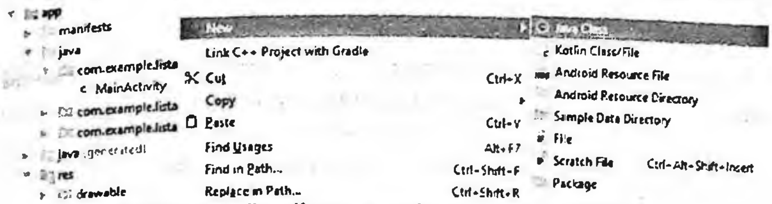
7.9-rasm. Dastur natijasi

### 7.5. Ro'yxatlarni kengaytirish va adapter yaratish

An'anaviy ListViews standart ArrayAdapters yordamida massivli satr bilan yaxshi ishlaydi. Biroq, biz tez -tez murakkab elementlar ro'yxatiga duch kelamiz, bu erda bitta element bitta chiziqni emas, balki bir nechta satrlarni, rasmlarni va boshqa komponentalarni ifodalaydi.

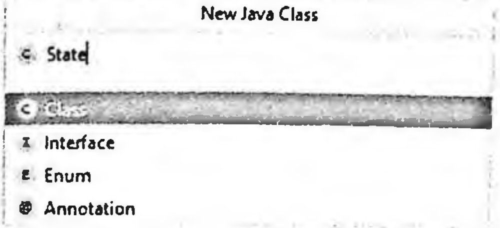
Murakkab ro'yxatni tuzish uchun biz ishlatilgan adapterlardan birini bekor qilishimiz kerak. Qoida tariqasida, ArrayAdapter ishlatiladi, shuning uchun biz uni bekor qilamiz.

Lekin birinchi navbatda, ma'lumotlar ro'yxatda ko'rsatiladigan modelni aniqlaylik. Buning uchun MainActivity sinfi joylashgan katalogga yangi sinf qo'shing. Buning uchun sichqonchaning o'ng tugmasi bilan ushbu katalogni bosing va menyudan **New -> Java Class** ni tanlang:



7.10-rasm. Loyihaga yangi Java sinfini qo`shish

Ko'rsatilgan oynada qo'shilgan sinf uchun State nomini ko'rsating.



7.11-rasm. Loyihaga yangi Java sinfiga nom berish

Qo'shgandan so'ng, State sinfini quyidagicha o'zgartiramiz:

```
package com.example.listapp;
public class State {
   private String name; // nomlash
   private String capital;  // poytaxt
   private int flagResource; // resurs bayroqchasi
   public State(String name, String capital, int flag){
      this.name=name;
      this.capital=capital;
      this.flagResource=flag;
   }
   public String getName() {
      return this.name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getCapital() {
      return this.capital;
   }
   public void setCapital(String capital) {
      this.capital = capital;
   }
   public int getFlagResource() {
      return this.flagResource;
   }
   public void setFlagResource(int flagResource) {
      this.flagResource = flagResource;
```

```
    }
}
```
Bu sinf ikkita satrli maydonni saqlaydi - shtat nomi va uning poytaxti, shuningdek drawable papkasidagi tasvir manbasini ko'rsatadigan raqamli maydon, shtat bayrog'i ko'rsatiladi.

Keyin, **res/layout** papkasiga yangi **list_item.xml** faylini qo'shing, u ro'yxatdagi bitta elementni belgilaydi:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="wrap_content">
  <ImageView
    android:id="@+id/flag"
    android:layout_width="70dp"
    android:layout_height="50dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toLeftOf="@+id/name"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent" />
  <TextView
    android:id="@+id/name"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:text="Название"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toRightOf="@+id/flag"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@-id/capital" />
  <TextView
    android:id="@+id/capital"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:text="Столица"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toRightOf="@+id/flag"
    app:layout_constraintTop_toBottomOf="@+id/name"
    app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```
Har bir elementda ImageView tasviri va shtat nomi va poytaxtini ko'rsatish uchun ikkita TextView komponentasi bo'ladi.

Shundan so'ng, MainActivity va State sinflari joylashgan katalogga yangi sinf qo'shing, biz buni **StateAdapter** deb ataymiz:

```
package com.example.listapp;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.List;
public class StateAdapter extends ArrayAdapter<State> {
   private LayoutInflater inflater;
   private int layout;
   private List<State> states;
   public StateAdapter(Context context, int resource, List<State> states) {
      super(context, resource, states);
      this.states = states;
      this.layout = resource;
      this.inflater = LayoutInflater.from(context);
   }
   public View getView(int position, View convertView, ViewGroup parent)
{
      View view=inflater.inflate(this.layout, parent, false);
      ImageView flagView = (ImageView) view.findViewById(R.id.flag);
      TextView nameView = (TextView) view.findViewById(R.id.name);
      TextView capitalView = (TextView) view.findViewById(R.id.capital);
      State state = states.get(position);
      flagView.setImageResource(state.getFlagResource());
      nameView.setText(state.getName());
      capitalView.setText(state.getCapital());
      return view;
   }
}
```

Bu erda ro'yxat bilan barcha o'zaro munosabatlar StateAdapter sinfid o'tadi. StateAdapter konstruktorida biz uchta parametrni asosiy sinf konstruktoriga berishimiz kerak:

- sinf ishlatiladigan kontekst. A_ctivity sinfi, qoida tariqasida, o'z vazifasini bajaradi.
- ListView da bitta elementni yaratish uchun ishlatiladigan interfeyslarni belgilash manbasi
- ListView da ko'rsatiladigan ob'ektlar to'plami

StateAdapter konstruktorida biz belgilash manbasini va ob'ektlar to'plamini olamiz va ularni alohida o'zgaruvchilarda saqlaymiz. Olingan belgilash manbasidan

View ob'ektini yaratish uchun sizga LayoutInflater ob'ekti kerak bo'ladi, u ham o'zgaruvchida saqlanadi.

- **getView()** usuli ro'yxat elementining ko'rinishini o'rnatadi. Bu usul uchta parametrni oladi:
- **position**: ko'rinish yaratilayotgan adapter ichidagi element o'rnini uzatadi
- **convertView**: ob'ektning eski ko'rinishi, agar mavjud bo'lsa, ListView tomonidan optimallashtirish maqsadida ishlatiladi
- **parent**: elementni ifodalovchi asosiy komponent

Bunday holda, LayoutInflater ob'ektidan foydalanib, biz ro'yxatdagi har bir alohida element uchun View ob'ektini yaratamiz:

*View view=inflater.inflate(this.layout, parent, false);*

Yaratilgan View ob'ektidan biz ImageView va TextView elementlarini id orqali olamiz:

*ImageView flagView = (ImageView) view.findViewById(R.id.flag);*
*TextView nameView = (TextView) view.findViewById(R.id.name);*
*TextView capitalView = (TextView) view.findViewById(R.id.capital);*

Bu list_item.xml faylida aniqlangan elementlar. Bu erda biz ularni olamiz.

Keyinchalik, position parametridan foydalanib, biz belgilash uchun yaratilgan State ob'ektini olamiz:

*State state = states.get(position);*

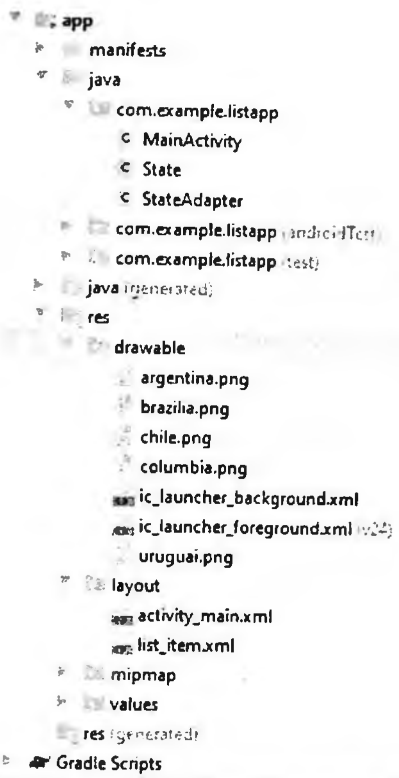
Keyin olingan ImageView va TextView elementlarini pozitsiya bo'yicha olingan State ob'ektidan to'ldiramiz:

*flagView.setImageResource(state.getFlagResource());*
*nameView.setText(state.getName());*
*capitalView.setText(state.getCapital());*

Va oxirida, State ob'ektini ko'rsatish uchun yaratilgan View elementi usuldan qaytariladi:

*return view;*

Rasmlardan foydalanish uchun *res/drawable* papkasiga bir nechta rasm qo'shing, menimcha, bu davlat bayroqlarining beshta tasviri. Natijada, loyiha quyidagicha ko'rinadi:



```
app
  manifests
  java
    com.example.listapp
      MainActivity
      State
      StateAdapter
    com.example.listapp (androidTest)
    com.example.listapp (test)
  java (generated)
  res
    drawable
      argentina.png
      brazilia.png
      chile.png
      columbia.png
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
      uruguai.png
    layout
      activity_main.xml
      list_item.xml
    mipmap
    values
  res (generated)
Gradle Scripts
```

7.12-rasm. Loyihada drawable papkasi tarkibi

active_main.xml faylida ma'lumotlar yuklanadigan ListViewni aniqlang:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <ListView
    android:id="@+id/countriesList"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Va MainActivity faylida, StateAdapterni ListViewga ulang:

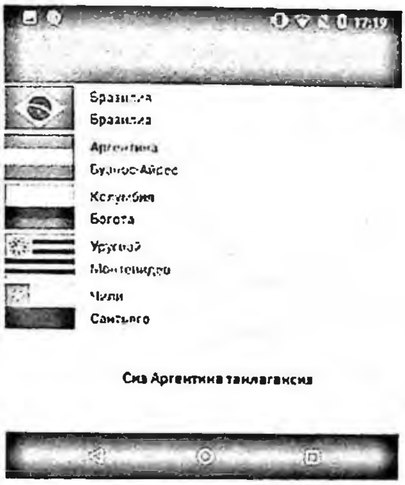
```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
```

```
        import android.os.Bundle;
        import android.view.View;
        import android.widget.AdapterView;
        import android.widget.ListView;
        import android.widget.Toast;
        import java.util.ArrayList;
        public class MainActivity extends AppCompatActivity {
           ArrayList<State> states = new ArrayList();
           ListView countriesList;
           @Override
           protected void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.activity_main);
              // ro'yxatning boshlanishi
              setInitialData();
              // ListView elementini oling
              countriesList = (ListView) findViewById(R.id.countriesList);
              // adapter yarating
              StateAdapter stateAdapter = new StateAdapter(this, R.layout.list_item,
states);
              // adapterni o'rnating
              countriesList.setAdapter(stateAdapter);
              // ro'yxatni tanlash tinglovchisi
              AdapterView.OnItemClickListener        itemListener        =        new
AdapterView.OnItemClickListener() {
                 @Override
                 public void onItemClick(AdapterView<?> parent, View v, int
position, long id) {
                    // tanlangan elementni oling
                    State selectedState = (State)parent.getItemAtPosition(position);
                    Toast.makeText(getApplicationContext(), "Был выбран пункт "
+ selectedState.getName(),
                          Toast.LENGTH_SHORT).show();
                 }
              };
              countriesList.setOnItemClickListener(itemListener);
           }
           private void setInitialData(){
              states.add(new State ("Бразилия", "Бразилиа", R.drawable.brazilia));
              states.add(new       State       ("Аргентина",       "Буэнос-Айрес",
R.drawable.argentina));
              states.add(new State ("Колумбия", "Богота", R.drawable.columbia));
              states.add(new       State       ("Уругвай",       "Монтевидео",
R.drawable.uruguai));
              states.add(new State ("Чили", "Сантьяго", R.drawable.chile));
        }
    }
```

Bu erda ma'lumotlar manbai - setInitialData usulida ma'lumotlarni oladigan ArrayList sinfi. Ro'yxatdagi har bir qo'shilgan State ob'ektiga shtat nomi, uning poytaxti va tasvir bayrog'i tasvirlangan res/drawable papkasidan uzatiladi.

Adapter yaratilganda, ilgari yaratilgan list_item.xml belgilash manbasi va holatlar ro'yxati unga uzatiladi:

*StateAdapter stateAdapter = new StateAdapter(this, R.layout.list_item, states);*


Бразилия
Бразилиа
Аргентина
Буэнос-Айрес
Колумбия
Богота
Уругвай
Монтевидео
Чили
Сантьяго

Сиз Аргентина танлагансиз

7.12-rasm. Dastur natijasi

## 7.6. Adapter va View Holder ni optimallashtirish

Oldingi mavzuda ob'ektlarning murakkab ro'yxatlari bilan ishlashga imkon beradigan maxsus adapter yaratildi:

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.List;
public class StateAdapter extends ArrayAdapter<State> {
    private LayoutInflater inflater;
    private int layout;
    private List<State> states;
    public StateAdapter(Context context, int resource, List<State> states) {
        super(context, resource, states);
        this.states = states;
        this.layout = resource;
```

```
            this.inflater = LayoutInflater.from(context);
        }
        public View getView(int position, View convertView, ViewGroup parent)
{
            View view=inflater.inflate(this.layout, parent, false);
            ImageView flagView = (ImageView) view.findViewById(R.id.flag);
            TextView nameView = (TextView) view.findViewById(R.id.name);
            TextView capitalView = (TextView) view.findViewById(R.id.capital);
            State state = states.get(position);
            flagView.setImageResource(state.getFlagResource());
            nameView.setText(state.getName());
            capitalView.setText(state.getCapital());
            return view;
        }
    }
```

Ammo bu adapterning bitta katta kamchiliklari bor - ListView ni aylantirishda, agar ro'yxatda juda ko'p ob'ektlar bo'lsa, u holda har bir element uchun, ko'rinishda, getView usuli qayta chaqiriladi, unda yangi View ob'ekt qayta yaratiladi. Shunga ko'ra, xotira iste'moli ortadi va ishlash pasayadi. Shuning uchun biz **getView** usulining kodini optimallashtiramiz:

```
    public View getView(int position, View convertView, ViewGroup parent) {
        if(convertView==null){
            convertView = inflater.inflate(this.layout, parent, false);
        }
        ImageView          flagView          =          (ImageView)
convertView.findViewById(R.id.flag);
        TextView          nameView          =          (TextView)
convertView.findViewById(R.id.name);
        TextView          capitalView          =          (TextView)
convertView.findViewById(R.id.capital);
        State state = states.get(position);
        flagView.setImageResource(state.getFlagResource());
        nameView.setText(state.getName());
        capitalView.setText(state.getCapital());
        return convertView;
    }
```

ConvertView parametri position pozitsiyasi bo'yicha ro'yxatdagi ob'ekt uchun ishlatiladigan View elementiga ishora qiladi. Agar View bu ob'ekt uchun allaqachon yaratilgan bo'lsa, unda convertView parametri biz ishlatadigan qiymatni o'z ichiga oladi.

Bunday holda, biz allaqachon yaratilgan ob'ektlarni qayta ishlatamiz va ishlashni oshiramiz, lekin bu kodni yanada optimallashtirish mumkin. Gap shundaki, elementlarni id orqali olish ham nisbatan qimmat operatsiya hisoblanadi. Shuning uchun, biz StateAdapter kodini quyidagicha o'zgartirib, yanada optimallashtiramiz:

```
    import android.content.Context;
```

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.List;
public class StateAdapter extends ArrayAdapter<State> {
  private LayoutInflater inflater;
  private int layout;
  private List<State> states;
  public StateAdapter(Context context, int resource, List<State> states) {
    super(context, resource, states);
    this.states = states;
    this.layout = resource;
    this.inflater = LayoutInflater.from(context);
  }
  public View getView(int position, View convertView, ViewGroup parent)

    ViewHolder viewHolder;
    if(convertView==null){
      convertView = inflater.inflate(this.layout, parent, false);
      viewHolder = new ViewHolder(convertView);
      convertView.setTag(viewHolder);
    }
    else{
      viewHolder = (ViewHolder) convertView.getTag();
    }
    State              state              =              states.get(position);
wHolder.imageView.setImageResource(state.getFlagResource());
    viewHolder.nameView.setText(state.getName());
    viewHolder.capitalView.setText(state.getCapital());
    return convertView;
  }
  private class ViewHolder {
    final ImageView imageView;
    final TextView nameView, capitalView;
    ViewHolder(View view){
      imageView = (ImageView)view.findViewById(R.id.flag);
      nameView = (TextView) view.findViewById(R.id.name);
      capitalView = (TextView) view.findViewById(R.id.capital);
    }
  }
}
```

Ishlatilgan ImageView va TextView elementlariga havolalarni saqlash uchun ViewHolder ichki xususiy sinfi aniqlanadi, u konstruktorda ImageView va TextView o'z ichiga olgan View ob'ektini oladi.

GetView usulida, agar convertView null bo'lsa (ya'ni, ilgari ob'ekt uchun hech qanday belgilash yaratilmagan bo'lsa), ViewHolder ob'ektini yarating, biz uni convertView yorlig'iga saqlaymiz:

*convertView.setTag(viewHolder);*

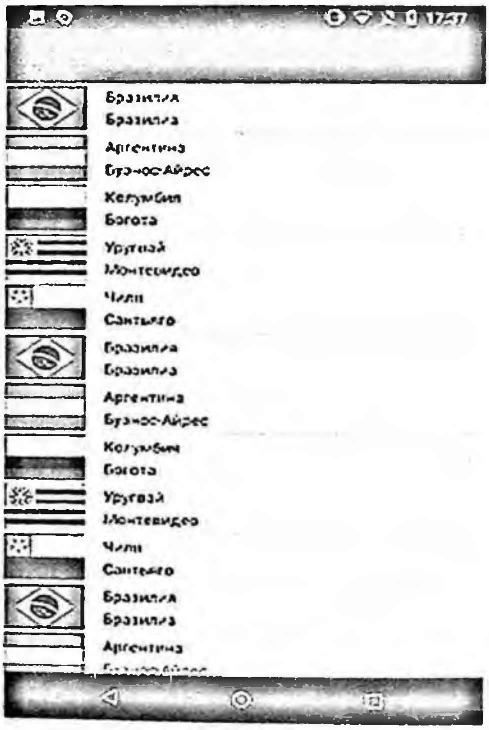
Agar ListViewdagi ob'ekt uchun belgilash allaqachon yaratilgan bo'lsa, biz ViewHolderni tegdan qaytaramiz:

*viewHolder = (ViewHolder) convertView.getTag();*

Keyin State ob'ektining qiymatlari ViewHolderdagi ImageView va TextView uchun ham o'rnatiladi:

*viewHolder.imageView.setImageResource(state.getFlagResource());*
*viewHolder.nameView.setText(state.getName());*
*viewHolder.capitalView.setText(state.getCapital());*

Va endi ListView, ayniqsa katta ro'yxatlar bilan, oldingi mavzuga qaraganda silliq va samaraliroq ishlaydi:



```
Бразилия
Бразилиа
Аргентина
Буэнос-Айрес
Колумбия
Богота
Уругвай
Монтевидео
Чили
Сантьяго
Бразилия
Бразилиа
Аргентина
Буэнос-Айрес
Колумбия
Богота
Уругвай
Монтевидео
Чили
Сантьяго
Бразилия
Бразилиа
Аргентина
```

7.13-rasm. Dastur natijasi

## 7.7. Tugmalar bilan murakkab ro'yxat

Avvalgi darslarda biz murakkab ma'lumotlarni ro'yxatlarda ko'rsatishga imkon beradigan maxsus adapterlarni ko'rib chiqdik. Keling, tugmalar kabi boshqa elementlarni ro'yxatlarga qanday qo'shishimiz va ularning voqealarini boshqarishni ko'rib chiqaylik.

Buning uchun biz birinchi navbatda quyidagi **Product** sinfini aniqlaymiz:

```
package com.example.listapp;
public class Product {
   private String name;
   private int count;
   private String unit;
   Product(String name, String unit){
      this.name = name;
      this.count=0;
      this.unit = unit;
   }
   public String getUnit() {
      return this.unit;
   }
   public void setCount(int count) {
      this.count = count;
   }
   public int getCount() {
      return count;
   }
   public void setName(String name){
      this.name = name;
   }
   public String getName(){
      return this.name;
   }
}
```

Bu sinfda mahsulot nomi, miqdori va o'lchov birligi saqlanadi. Va bu sinf ob'ektlari ro'yxatda ko'rsatiladi.

Buning uchun **res/layout** papkasiga yangi **list_item.xml** faylini qo'shing:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:padding="16dp" >
   <TextView
      android:id="@+id/nameView"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:textSize="18sp"
      app:layout_constraintHorizontal_weight="2"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/countView"
        app:layout_constraintTop_toTopOf="parent"/>
    <TextView
        android:id="@+id/countView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        app:layout_constraintHorizontal_weight="2"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/nameView"
        app:layout_constraintRight_toLeftOf="@+id/addButton"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/addButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="+"
        app:layout_constraintHorizontal_weight="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/countView"
        app:layout_constraintRight_toLeftOf="@+id/removeButton"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/removeButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="-"
        app:layout_constraintHorizontal_weight="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/addButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Mahsulot nomi va miqdorini ko'rsatish uchun ikkita matnli maydoni v mahsulotning bir birligini qo'shish va olib tashlash uchun ikkita tugma mavjud.

Endi **ProductAdapter** deb nomlangan adapter sinfini qo'shamiz:

```
package com.example.listapp;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
```

```
        import android.widget.TextView;
        import java.util.ArrayList;
        class ProductAdapter extends ArrayAdapter<Product> {
           private LayoutInflater inflater;
           private int layout;
           private ArrayList<Product> productList;
           ProductAdapter(Context context, int resource, ArrayList<Product>
products) {
              super(context, resource, products);
              this.productList = products;
              this.layout = resource;
              this.inflater = LayoutInflater.from(context);
           }
           public View getView(int position, View convertView, ViewGroup parent)
{
              final ViewHolder viewHolder;
              if(convertView==null){
                 convertView = inflater.inflate(this.layout, parent, false);
                 viewHolder = new ViewHolder(convertView);
                 convertView.setTag(viewHolder);
              }
              else{
                 viewHolder = (ViewHolder) convertView.getTag();
              }
              final Product product = productList.get(position);
              viewHolder.nameView.setText(product.getName());
              viewHolder.countView.setText(product.getCount() + " " +
product.getUnit());
              viewHolder.removeButton.setOnClickListener(new
View.OnClickListener() {
                 @Override
                 public void onClick(View v) {
                    int count = product.getCount()-1;
                    if(count<0) count=0;
                    product.setCount(count);
                    viewHolder.countView.setText(count + " " + product.getUnit());
                 }
              });
              viewHolder.addButton.setOnClickListener(new
View.OnClickListener() {
                 @Override
                 public void onClick(View v) {
                    int count = product.getCount()+1;
                    product.setCount(count);
                    viewHolder.countView.setText(count + " " + product.getUnit());
```

```
            }
        });
        return convertView;
    }
    private class ViewHolder {
        final Button addButton, removeButton;
        final TextView nameView, countView;
        ViewHolder(View view){
            addButton = (Button) view.findViewById(R.id.addButton);
            removeButton = (Button) view.findViewById(R.id.removeButton);
            nameView = (TextView) view.findViewById(R.id.nameView);
            countView = (TextView) view.findViewById(R.id.countView);
        }
    }
}
```

Bu erda har bir tugma uchun bosish moslamasi aniqlanadi, bunda biz mahsulot miqdorini bir marta kamaytiramiz yoki ko'paytiramiz, so'ngra tegishli matn maydoniga matnni qayta o'rnatamiz.

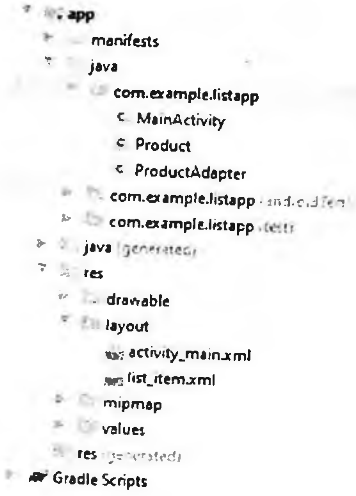
Keyin, **activity_main.xml** faylida ListView elementini belgilang:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ListView
        android:id="@+id/productList"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Va **MainActivity** sinfini o'zgartiraylik:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ListView;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ArrayList<Product> products = new ArrayList<Product>();
        if(products.size()==0){
          products.add(new Product("Картофель", "кг."));
          products.add(new Product("Чай", "шт."));
          products.add(new Product("Яйца", "шт."));
          products.add(new Product("Молоко", "л."));
          products.add(new Product("Макароны", "кг."));
        }
        ListView productList = (ListView) findViewById(R.id.productList);
        ProductAdapter    adapter    =    new    ProductAdapter(th
R.layout.list_item, products);
        productList.setAdapter(adapter);
    }
}
```

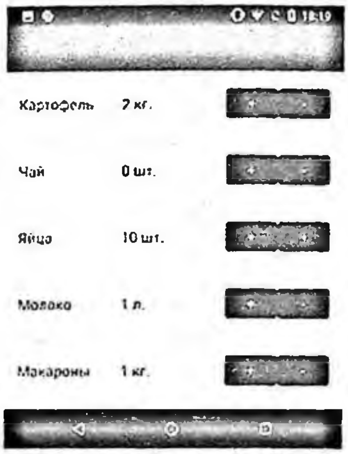
Natijada siz quyidagi loyihani olasiz:



```
app
  manifests
  java
    com.example.listapp
      MainActivity
      Product
      ProductAdapter
    com.example.listapp (androidTest)
    com.example.listapp (test)
  java (generated)
  res
    drawable
    layout
      activity_main.xml
      list_item.xml
    mipmap
    values
  res (generated)
Gradle Scripts
```

7.14-rasm. Loyihada java papkasi

Va ilovani ishga tushirgandan so'ng, biz tugmalar orqali mahsulot sonini nazorat qila olamiz:



```
Картофель   2 кг.
Чай         0 шт.
Яйцо        10 шт.
Молоко      1 л.
Макароны    1 кг.
```

7.15-rasm. Dastur natijasi

## 7.8. ListActivity sinfi

Ro'yxat elementlariga kirishni soddalashtirish uchun **ListActivity** sinfi ishlatiladi. ListActivity - bu Activitydan meros bo'lib o'tgan va ro'yxatlar bilan ishlash uchun maxsus mo'ljallangan sinf.

Shunday qilib, keling, bir misolni ko'rib chiqaylik. Birinchidan, **activity_main.xml** formatlash faylida ListView elementini aniqlaymiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <ListView
    android:id="@android:id/list"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

ListView identifikatori uchun deklaratsiyaga e'tibor bering: android:id="@android:id/list". ListActivity ro'yxatni tanib olish va o'zgartirish uchun shunga o'xshash deklaratsiya talab qilinadi.

ListViewdan tashqari, interfeyslarni belgilash fayli boshqa elementlarni ham o'z ichiga olishi mumkin. Ammo bu holda biz faqat ListView elementi bilan cheklanamiz.

Keyin, **MainActivity** sinfining kodini o'zgartiraylik:

```
package com.example.listapp;
```

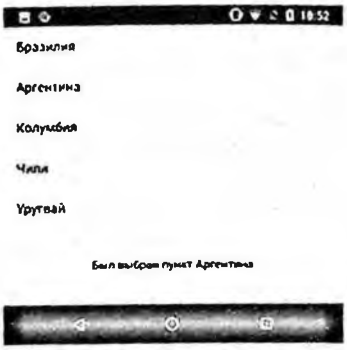
```
import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Toast;
public class MainActivity extends ListActivity {
   String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // adapter yarating
      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, countries);
      setListAdapter(adapter);
      AdapterView.OnItemClickListener        itemListener        =        new
AdapterView.OnItemClickListener() {
         @Override
         public  void  onItemClick(AdapterView<?>  parent,  View  v,  int
position, long id) {
            Toast.makeText(getApplicationContext(), "Был выбран пункт "
+
                  parent.getItemAtPosition(position).toString(),
Toast.LENGTH_SHORT).show();
         }
      };
      getListView().setOnItemClickListener(itemListener);
   }
}
```

MainActivity sinfi endi asosiy ListActivity sinfini kengaytiradi.

Bu erda, ListView misolida bo'lgani kabi, biz ArrayAdapter adapterini yaratamiz, faqat biz uni ListActivity tomonidan aniqlanadigan setListAdapter usuli orqali o'rnatamiz.

Keyinchalik, ro'yxat elementlarini tanlash bilan shug'ullanadigan OnItemClickListener tinglovchi obyekti yaratiladi. Uning yagona onItemClick usuli oldingi bo'limda muhokama qilingan usulga o'xshaydi, faqat bu erda biz tanlangan element matni bilan xabarni ko'rsatamiz.

Nihoyat, biz ListLiew ob'ektini qaytaradigan getListView() usulidan foydalanamiz. Va buning uchun biz yuqorida belgilangan tinglovchini o'rnatdik.


Бразилия

Аргентина

Колумбия

Чили

Уругвай

Был выбран пункт Аргентина

7.16-rasm. Dastur natijasi

## 7.9. Spinner ochiladigan ro'yxati

**Spinner**-ochiladigan ro'yxat. **activity_main.xml** formatlash faylida **Spinner** elementini aniqlaylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp">
   <Spinner
      android:id="@+id/spinner"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

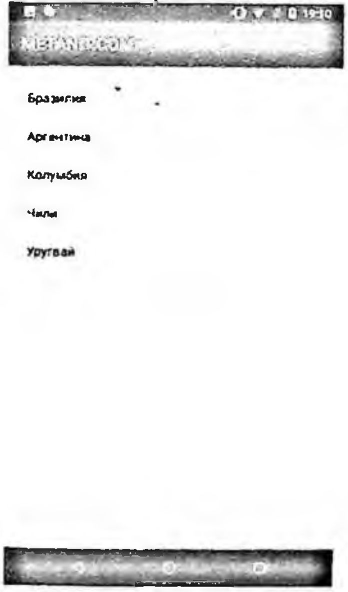
Ma'lumot resursi sifatida, ListViewga kelsak, Spinner oddiy ro'yxat yoki dasturiy ravishda yaratilgan massiv yoki string-array resurssi bo'lishi mumkin. Ma'lumot resursi bilan o'zaro aloqa ham adapter orqali o'tadi. Bunday holda, MainActivity kodida resurs dasturiy jihatdan massiv sifatida aniqlansin:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
public class MainActivity extends AppCompatActivity {

String[]countries={"Бразилия","Аргентина","Колумбия","Чили","Уругвай"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

```
            setContentView(R.layout.activity_main);
            Spinner spinner = (Spinner) findViewById(R.id.spinner);
            // Massivli satrlar va spinner standart belgilaridan foydalanib,
ArrayAdapter yarating
            ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, countries);
            //        Ob'ektni        tanlashda        belgilashni        belgilang
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
            // spinner elementiga adapter qo'llanilishi
            spinner.setAdapter(adapter);
        }
    }
```

ArrayAdapterni yaratishda ishlatiladigan android.R.layout.simple_spinner_item resursi ramka bilan ta'minlangan va ochiladigan ro'yxatni tuzish uchun standart belgidir.

Adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item) usuli ro'yxatning qo'shimcha vizual imkoniyatlarini o'rnatadi. Va usulga o'tgan android.R.layout.simple_spinner_dropdown_item resursi ochiladigan ro'yxatni ko'rsatish uchun ishlatiladi va platforma tomonidan ham ta'minlanadi.



7.17-rasm. Dastur natijasi

Ob'ektni tanlashni qayta ishlash. OnItemSelectedListenerdan, xususan, onItemSelected() usulidan foydalanib, biz ro'yxatdagi elementni tanlash bilan shug'ullana olamiz. Birinchidan, tanlangan elementni ko'rsatadigan interfeys belgisiga matn maydonini qo'shamiz:

```
        <?xml version="1.0" encoding="utf-8"?>
        <androidx.constraintlayout.widget.ConstraintLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <TextView
        android:id="@+id/selection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </TextView>
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/selection" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Va Spinner uchun **OnItemSelectedListener**ni aniqlash uchun MainActivity kodini o'zgartiraylik:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили", "Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView selection = (TextView) findViewById(R.id.selection);
        Spinner spinner = (Spinner) findViewById(R.id.spinner);
        // Massivli qatorlar va spinner standart belgilaridan foydalanib. ArrayAdapter yarating
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this. android.R.layout.simple_spinner_item. countries);
        // Ob'ektni tanlashda belgilashni belgilang
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dro
down_item);
        // spinner elementiga adapter qo'llanilishi
        spinner.setAdapter(adapter);
        AdapterView.OnItemSelectedListener itemSelectedListener = new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                // Tanlangan ob'ektni oling
                String item = (String)parent.getItemAtPosition(position);
                selection.setText(item);
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {
            }
        };
        spinner.setOnItemSelectedListener(itemSelectedListener);
    }
}
```

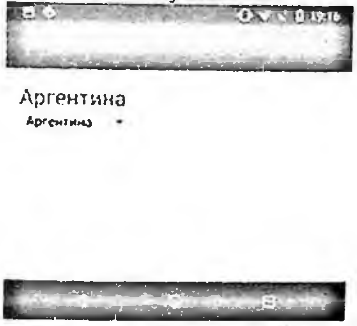
OnItemSelectedListenerning onItemSelected usuli to'rtta parametrni oladi:

- **parent**: ob'ekt tanlash hodisasi sodir bo'lgan Spinner obyekti
- **view**: Spinnera ichidagi tanlangan elementni ifodalovchi View obyekti
- **position**: adapterda tanlangan element indeksi
- **id**: tanlangan elementning satr identifikatori

Tanlangan elementning pozitsiyasini olganimizdan so'ng, biz uni ro'yxatda topamiz:

*String item = (String)parent.getItemAtPosition(position);*

Spinner sinfi OnItemSelectedListener tinglovchisini o'rnatish uchun **setOnItemSelectedListener** usulidan foydalanadi.


Аргентина
Аргентина

7.18-rasm. Dastur natijasi

## 7.10. AutoCompleteTextView avtomatik to'ldirish vidjeti

**AutoCompleteTextView** avtomatik to'ldirish imkoniyatiga ega bo'lgan EditText sinfiga asoslangan elementni ifodalaydi

Birinchidan, biz ushbu elementni belgilash resurssida e'lon qilamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp">
  <AutoCompleteTextView
    android:id="@+id/autocomplete"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:completionHint="Введите город"
    android:completionThreshold="1"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```

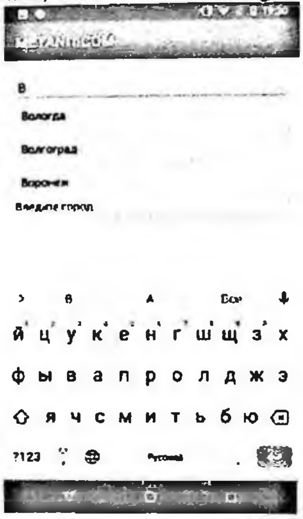
**android:completHint** atributi ro'yxatning pastki qismida ko'rsatiladigan sarlavhani o'rnatishga imkon beradi va **android:completThreshold** xususiyati ishga kirishi uchun avtomatik to'ldirish uchun qancha belgini kiritish kerakligini belgilaydi. Ya'ni, bu holda, bitta belgini kiritgandan so'ng, almashtirishlar ro'yxati paydo bo'lishi kerak.

ListView va Spinner elementlarida bo'lgani kabi, AutoCompleteTextView ma'lumotlar resurssiga adapter orqali ulanadi. Ma'lumot resursi yana massiv yoki ob'ektlar ro'yxati yoki satrli resurs bo'lishi mumkin.

Endi MainActivity sinfidagi vidjetga massiv satrlarni ulaylik:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
public class MainActivity extends AppCompatActivity {
  String[] cities = {"Москва", "Самара", "Вологда", "Волгоград",
"Саратов", "Воронеж"};
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // Belgilashda AutoCompleteTextView elementiga havolani olish
    AutoCompleteTextView autoCompleteTextView =
(AutoCompleteTextView) findViewById(R.id.autocomplete);
    // AutoCompleteTextView elementini avtomatik to'ldirish uchun
adapter yarating
```

```
        ArrayAdapter<String> adapter =
            new                                        ArrayAdapter<String>(this,
R.layout.support_simple_spinner_dropdown_item, cities);
        autoCompleteTextView.setAdapter(adapter);
    }
}
```
Matn maydoniga bitta harf kiritilgandan so'ng, avtomatik to'ldirish variantlari ro'yxati paydo bo'ladi, bu erda siz xohlagan harfni tanlashingiz mumkin:



7.19-rasm. Dastur natijasi

**MultiAutoCompleteTextView**. Bu vidjet AutoCompleteTextView boshqaruv funktsiyasini to'ldiradi. **MultiAutoCompleteTextView** sizga avtomatik to'ldirishni faqat bitta satr uchun emas, balki alohida so'zlar uchun ham ishlatishga imkon beradi. Masalan, agar so'z kiritilsa va undan keyin vergul qo'yilsa, vergul yoki boshqa ajratgichdan keyin yangi kiritilgan so'zlar uchun avtomatik to'ldirish ishlayveradi.

MultiAutoCompleteTextView AutoCompleteTextView bilan bir xil e'lon qilishga ega:

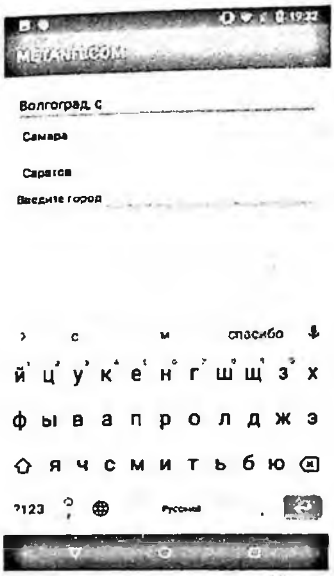
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp">
  <MultiAutoCompleteTextView
    android:id="@+id/autocomplete"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:completionHint="Введите город"
```

```
        android:completionThreshold="1"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Kodda MultiAutoCompleteTextViewni yoqish uchun siz ajratuvchi tokenni o'rnatishingiz kerak:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.MultiAutoCompleteTextView;
public class MainActivity extends AppCompatActivity {
  String[] cities = {"Москва", "Самара", "Вологда", "Волгоград",
"Саратов", "Воронеж"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // Belgilashda AutoCompleteTextView elementiga havolani olish
      MultiAutoCompleteTextView autoCompleteTextView =
(MultiAutoCompleteTextView) findViewById(R.id.autocomplete);
      // AutoCompleteTextView elementini avtomatik to'ldirish uchun
adapter yarating
      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
R.layout.support_simple_spinner_dropdown_item, cities);
      autoCompleteTextView.setAdapter(adapter);
      // ajratuvchi sifatida vergulni o'rnatish
      autoCompleteTextView.setTokenizer(new
MultiAutoCompleteTextView.CommaTokenizer());
    }
}
```

Bu o'rnatilgan vergulga asoslangan **CommaTokenizer()** ajratgichidan foydalanadi. Agar kerak bo'lsa, biz o'z ajratuvchilarimizni yaratishimiz mumkin.



7.20-rasm. Dastur natijasi

### 7.11. GridView elementi

GridView elementi jadval ko'rinishidagi tasvirlashni - qatorlar va ustunlar to'plamini ifodalaydi.

GridView ning asosiy atributlari:

- **android: columnWidth**: ustunlarning belgilangan kengligini o'rnatadi
- **android: gravity**: har bir hujayra ichidagi tarkibni moslashtirishni o'rnatadi
- **android: horizontalSpacing**: ustunlar orasidagi gorizontal oraliqni o'rnatadi
- **android: numColumns**: ustunlar sonini belgilaydi
- **android: stretchMode**: ustunlar qanday cho'zilishini va konteyner maydonini egallashini belgilaydi. U quyidagi qiymatlarni olishi mumkin:
    - columnWidth: Har bir ustun butun kengligi bo'ylab teng ravishda cho'zilgan. 2 ga teng
    - none: ustunlar cho'zilmagan. 0 ga teng
    - spacingWidth: ustunlar orasidagi bo'shliq hosil bo'ladi. 1 ga teng
    - spacingWidthUniform: Ustunlar orasiga tekis to'ldirishni ta'minlaydi. 3 ga teng
- **android: verticalSpacing**: chiziqlar orasidagi vertikal oraliqni o'rnatadi

GridView sinfining asosiy usullari:

- **int getColumnWidth()**: ustunlar kengligini qaytaradi
- **int getHorizontalSpacing()**: gorizontal to'ldirish hajmini qaytaradi
- **int getNumColumns()**: ustunlar sonini qaytaradi
- **int getStretchMode()**: panjara ichidagi bo'shliqni cho'zish rejimini qaytaradi
- **int getVerticalSpacing()**: vertikal to'ldirish hajmini qaytaradi
- **void setAdapter(ListAdapter adapter)**: adapterni ma'lumotlar resursiga ulanadigan qilib sozlash
- **void setColumnWidth(int columnWidth)**: ustunlar kengligini o'rnatadi

- **void setHorizontalSpacing(int horizontalSpacing)**: gorizontal to'ldirish hajmini belgilaydi
- **void setNumColumns(int numColumns)**: ustunlar sonini belgilaydi
- **void setStretchMode(int stretchMode)**: panjara ichidagi bo'sh joy uchun cho'zish rejimini o'rnatadi
- **void setVerticalSpacing(int verticalSpacing)**: vertikal to'ldirish hajmini belgilaydi
- **void setSelection(int pozitsiyasi)**: tanlangan elementni o'rnatadi

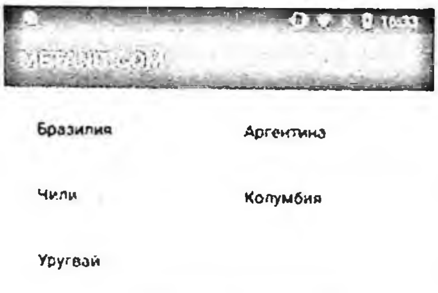
**activity_main.xml**da **GridView**ni aniqlaylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp">
  <GridView
    android:id="@+id/gridview"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:numColumns="2"
    android:verticalSpacing="16dp"
    android:horizontalSpacing="16dp"
    android:stretchMode="columnWidth"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bunday holda, biz panjara butun kengligi bo'ylab teng ravishda cho'zilgan 2 ustunga ega bo'lishini va hujayralar o'rtasida 16 dp gorizontal va vertikal chekkalari bo'lishini ko'rsatamiz.

Endi, ListViewda bo'lgani kabi, biz adapter bilan aloqa o'rnatishimiz kerak:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.GridView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
  String[] countries = { "Бразилия", "Аргентина", "Чили", "Колумбия", "Уругвай"};
  @Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // GridView elementini oling
    GridView countriesList = (GridView) findViewById(R.id.gridview);
    // adapter yarating
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, countries);
    countriesList.setAdapter(adapter);
    AdapterView.OnItemClickListener itemListener = new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            Toast.makeText(getApplicationContext(),"Вы выбрали "
                    + parent.getItemAtPosition(position).toString(),
                Toast.LENGTH_SHORT).show();
        }
    };
    countriesList.setOnItemClickListener(itemListener);
  }
}
```
GridView bosishni boshqarish uchun AdapterView.OnItemClickListener - dan foydalanadi.



METANIT.COM

```
Бразилия        Аргентина

Чили            Колумбия

Уругвай
```

7.21-rasm. Dastur natijasi

## 7.12. RecyclerView elementi

**RecyclerView** elementi ro'yxatlar bilan ishlashni optimallashtirish uchun mo'ljallangan va standart ListViewga qaraganda ko'proq ishlash afzalliklariga ega.

Ma'lumotni ko'rsatish uchun MainActivity sinfi joylashgan papkadagi loyihaga yangi Java sinfini qo'shing, biz uni State deb ataymiz:

```
package com.example.listapp;
public class State {
    private String name; // nomlanishi
    private String capital;  // poytaxt
    private int flagResource; // bayroqcha resursi
    public State(String name, String capital, int flag){
        this.name=name;
        this.capital=capital;
        this.flagResource=flag;
    }
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getCapital() {
        return this.capital;
    }
    public void setCapital(String capital) {
        this.capital = capital;
    }
    public int getFlagResource() {
        return this.flagResource;
    }
    public void setFlagResource(int flagResource) {
        this.flagResource = flagResource;
    }
}
```

State sinfida mamlakat nomi va poytaxtini saqlash uchun maydonlar, shuningdek, mamlakat bayrog'i tasvir resurssiga havola mavjud. Bunday holda, **res/drawable** papkasida ishlatilgan mamlakatlar uchun bayroqli rasm fayllari bo'lishi taxmin qilinadi.

Aytaylik, biz **RecyclerView** yordamida State ob'ektlari ro'yxatini ko'rsatmoqchimiz. Buning uchun **res/layout** papkasiga yangi list_item.xml faylini qo'shing:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/flag"
        android:layout_width="70dp"
        android:layout_height="50dp"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/name"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent" />
  <TextView
    android:id="@+id/name"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:text="Название"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toRightOf="@+id/flag"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/capital" />
  <TextView
    android:id="@+id/capital"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:text="Столица"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toRightOf="@+id/flag"
    app:layout_constraintTop_toBottomOf="@+id/name"
    app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Bu fayl bitta State obyektini chiqarish uchun belgini belgilaydi.

ListViewda bo'lgani kabi, **RecyclerView**da murakkab ob'ektlarni ko'rsatish uchun adapterni aniqlash kerak. Shuning uchun, keling, MainActivity va State sinfi joylashgan papkaga yangi Java sinfini qo'shaylik, biz buni **StateAdapter** deb ataymiz:

```
package com.example.listapp;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.recyclerview.widget.RecyclerView;
import java.util.List;
public class StateAdapter extends RecyclerView.Adapter<StateAdapter.ViewHolder>{
  private final LayoutInflater inflater;
  private final List<State> states;
  StateAdapter(Context context, List<State> states) {
    this.states = states;
    this.inflater = LayoutInflater.from(context);
  }
```

```
        @Override
        public    StateAdapter.ViewHolder    onCreateViewHolder(ViewGroup
parent, int viewType) {
            View view = inflater.inflate(R.layout.list_item, parent, false);
            return new ViewHolder(view);
        }
        @Override
        public  void  onBindViewHolder(StateAdapter.ViewHolder  holder,  int
position) {
            State state = states.get(position);
            holder.flagView.setImageResource(state.getFlagResource());
            holder.nameView.setText(state.getName());
            holder.capitalView.setText(state.getCapital());
        }
        @Override
        public int getItemCount() {
            return states.size();
        }
        public static class ViewHolder extends RecyclerView.ViewHolder {
            final ImageView flagView;
            final TextView nameView, capitalView;
            ViewHolder(View view){
                super(view);
                flagView = (ImageView)view.findViewById(R.id.flag);
                nameView = (TextView) view.findViewById(R.id.name);
                capitalView = (TextView) view.findViewById(R.id.capital);
            }
        }
    }
```

RecyclerView da ishlatiladigan adapter **RecyclerView.Adapter** abstrakt sinfidan meros bo'lishi kerak. Bu sinf uchta usulni belgilaydi:

- **onCreateViewHolder**: Ma'lumotlami birma -bir Phone ob'ektini saqlaydigan ViewHolder ob'ektini qaytaradi.
- **onBindViewHolder**: ViewHolder ob'ektini Phone ob'ektiga ma'lum bir pozitsiyada bog'laydi.
- **getItemCount**: ro'yxatdagi elementlar sonini qaytaradi

Ma'lumotlarni saqlash uchun adapter sinfi list_item.xml da belgilangan boshqaruv elementlaridan foydalanadigan statik ViewHolder sinfini belgilaydi.

Keling, **activ_main.xml** faylida RecyclerView elementini aniqlaylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list"
        android:layout_width="0dp"
        android:layout_height="0dp"
app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Shuni yodda tutish kerakki, RecyclerView **androidx.recyclerview.widget** paketida joylashgan va **Android Jetpack** asboblar to'plamiga kiradi, shuning uchun vidjetdan foydalanganda paketni hisobga olgan holda uning to'liq nomi ko'rsatiladi(printsipial jihatdan va ConstraintLayout haqida):

*<androidx.recyclerview.widget.RecyclerView ....*

RecyclerView uchun **app:layoutManager** atributi o'rnatilishi kerak, bu tartib menejeri turini ko'rsatadi. Layout menejeri **LayoutManager** sinfi bilan ifodalanadigan ob'ektni ifodalaydi. Odatiy bo'lib, RecyclerView kutubxonasi ushbu menejerning uchta dasturini taqdim etadi:

- **LinearLayoutManager**: elementlarni bitta ustunli ro'yxat sifatida tartibga soladi
- **GridLayoutManager**: Tarmoqdagi elementlarni ustunlar va qatorlar bilan tartibga soladi. Tarmoq elementlarni gorizontal (gorizontal panjara) yoki vertikal (vertikal panjara) tartibga solishi mumkin.
- **StaggeredGridLayoutManager**: GridLayoutManagerga o'xshash, lekin bir xil balandlikdagi (vertikal panjara uchun) va bir xil kenglikdagi (gorizontal panjara uchun) qatorning har bir elementi uchun o'rnatilishi shart emas.

Bunday holda, **androidx.recyclerview.widget.LinearLayoutManager** qiymatidan foydalanib, biz elementlar oddiy ro'yxatda joylashishini aniqlaymiz. E'tibor bering, LinearLayoutManager sinfi ham RecyclerView kutubxonasida joylashgan va shuning uchun qiymat ko'rsatilganda sinfning to'liq nomi uning to'plami nomi bilan ko'rsatiladi.

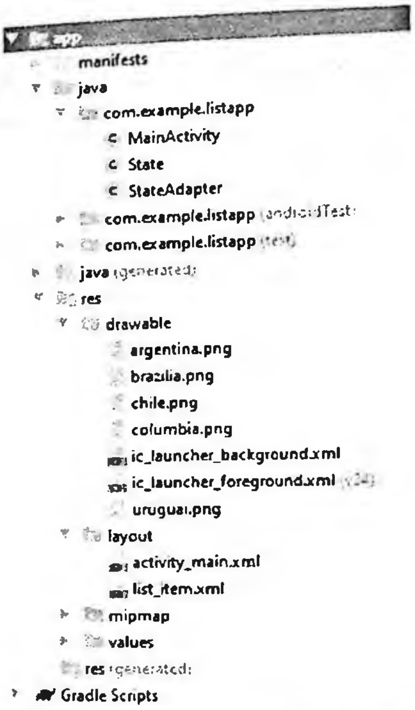
Va oxirida, **MainActivity** sinfini o'zgartiraylik:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;
import android.os.Bundle;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity {
    ArrayList<State> states = new ArrayList<State>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // ro'yxatning boshlanishi
        setInitialData();
        RecyclerView recyclerView = (RecyclerView) findViewById(R.id.list);
        // adapter yarating
        StateAdapter adapter = new StateAdapter(this, states);
        // ro'yxat uchun adapterni o'rnating
        recyclerView.setAdapter(adapter);
    }
    private void setInitialData(){
        states.add(new State ("Бразилия", "Бразилиа", R.drawable.brazilia));
        states.add(new State ("Аргентина", "Буэнос-Айрес",
R.drawable.argentina));
        states.add(new State ("Колумбия", "Богота", R.drawable.columbia));
        states.add(new State ("Уругвай", "Монтевидео",
R.drawable.uruguai));
        states.add(new State ("Чили", "Сантьяго", R.drawable.chile));
    }
}
```

SetInitialData() usuli dastlabki ma'lumotlar to'plamini o'rnatadi. Bu holda, biz res/drawables papkasida State moslamalari uchun bir qancha tasvir resurslari mavjudligini nazarda tutamiz.

ListView orqali ro'yxat ko'rsatilgandek, bu erda biz birinchi navbatda **RecyclerView** elementini olamiz, adapter yaratamiz va RecyclerView adapterini o'rnatamiz.

Nihoyat butun loyiha shunday bo'ladi:



```
▼ app
    manifests
  ▼ java
    ▼ com.example.listapp
        MainActivity
        State
        StateAdapter
    ► com.example.listapp (androidTest)
    ► com.example.listapp (test)
  ► java (generated)
  ▼ res
    ▼ drawable
        argentina.png
        brazilia.png
        chile.png
        columbia.png
        ic_launcher_background.xml
        ic_launcher_foreground.xml (v24)
        uruguai.png
    ▼ layout
        activity_main.xml
        list_item.xml
    ► mipmap
    ► values
    res (generated)
► Gradle Scripts
```

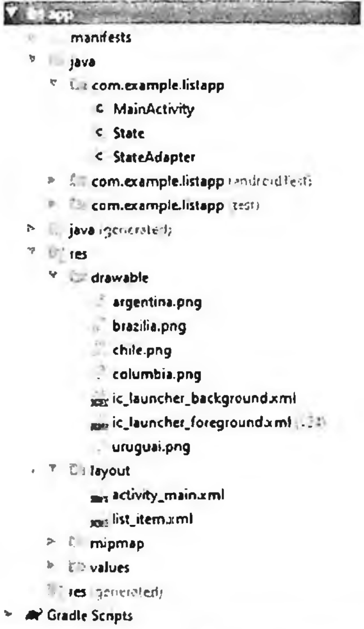
7.22-rasm. Loyihada RecyclerView adapterini o'rnatish

Natijada, RecyclerView ob'ektlar to'plamini ko'rsatadi:

7.23-rasm. Dastur natijasi

### 7.13. RecyclerViewda elementlarni tanlash

**RecyclerView** vidjeti bilan ishlaganda, muqarrar ravishda savol tug'iladi: RecyclerViewdagi elementni tanlash masalasi. Shuni ta'kidlash kerakki, ro'yxatlar bilan ishlashning boshqa turdagi vidjetlaridan farqli o'laroq, RecyclerView boshlang`ich bo'yicha ro'yxatdagi biror narsani bosish uchun tinglovchini belgilaydigan maxsus usulni ta'minlamaydi. Shuning uchun, butun infratuzilmani ishlab chiquvchi o'zi belgilashi kerak. Yaxshiyamki, aslida hamma narsa unchalik qiyin emas. Masalan, oldingi mavzudan loyihani olaylik:


```
app
  manifests
  java
    com.example.listapp
      MainActivity
      State
      StateAdapter
    com.example.listapp (androidTest)
    com.example.listapp (test)
  java (generated)
  res
    drawable
      argentina.png
      brazilia.png
      chile.png
      columbia.png
      ic_launcher_background.xml
      ic_launcher_foreground.xml
      uruguai.png
    layout
      activity_main.xml
      list_item.xml
    mipmap
    values
  res (generated)
Gradle Scripts
```

7.24-rasm. Loyihada RecyclerViewda elementlarni tanlash

Shunday qilib, loyihadagi ma'lumotlarni ko'rsatish uchun **State** sinfi mavjud bo'lib, u davlatni ifodalaydi:

```
package com.example.listapp;
public class State {
   private String name; // nomlanishi
   private String capital;  // poytaxt
   private int flagResource; // bayroqcha resursi
   public State(String name, String capital, int flag){
      this.name=name;
      this.capital=capital;
      this.flagResource=flag;
   }
   public String getName() {
      return this.name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getCapital() {
      return this.capital;
```

```
    }
    public void setCapital(String capital) {
        this.capital = capital;
    }
    public int getFlagResource() {
        return this.flagResource;
    }
    public void setFlagResource(int flagResource) {
        this.flagResource = flagResource;
    }
}
```

State sinfida mamlakat nomi va poytaxtini saqlash uchun maydonlar, shuningdek, mamlakat bayrog'i tasvir manbasiga havola mavjud. Bunday holda, **res/drawable** papkasida ishlatilgan mamlakatlar uchun bayroqli rasm fayllari bo'lishi taxmin qilinadi.

**res/layout** papkasida ro'yxatda bitta State ob'ektini ko'rsatish uchun quyidagi **list_item.xml** fayli aniqlangan:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/flag"
        android:layout_width="70dp"
        android:layout_height="50dp"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/name"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent" />
    <TextView
        android:id="@+id/name"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:text="Название"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/flag"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/capital" />
    <TextView
        android:id="@+id/capital"
```

```
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:text="Столица"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintLeft_toRightOf="@+id/flag"
            app:layout_constraintTop_toBottomOf="@+id/name"
            app:layout_constraintBottom_toBottomOf="parent" />
        </androidx.constraintlayout.widget.ConstraintLayout>
```

Keling, **StateAdapter** sinfiga o'tamiz va uning kodini quyidagicha ta'riflaymiz:

```
        package com.example.listapp;
        import android.content.Context;
        import android.view.LayoutInflater;
        import android.view.View;
        import android.view.ViewGroup;
        import android.widget.ImageView;
        import android.widget.TextView;
        import androidx.recyclerview.widget.RecyclerView;
        import java.util.List;
        public class StateAdapter extends
RecyclerView.Adapter<StateAdapter.ViewHolder>{
          interface OnStateClickListener{
            void onStateClick(State state, int position);
          }
          private final OnStateClickListener onClickListener;
          private final LayoutInflater inflater;
          private final List<State> states;
          StateAdapter(Context context, List<State> states, OnStateClickListener
onClickListener) {
            this.onClickListener = onClickListener;
            this.states = states;
            this.inflater = LayoutInflater.from(context);
          }
          @Override
          public StateAdapter.ViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
            View view = inflater.inflate(R.layout.list_item, parent, false);
            return new ViewHolder(view);
          }
          @Override
          public void onBindViewHolder(StateAdapter.ViewHolder holder, int
position) {
            State state = states.get(position);
        holder.flagView.setImageResource(state.getFlagResource());
        holder.nameView.setText(state.getName());
        holder.capitalView.setText(state.getCapital());
        holder.itemView.setOnClickListener(new View.OnClickListener(){
          @Override
          public void onClick(View v)
          {
            onClickListener.onStateClick(state, position);
          }
        });
      }
      @Override
      public int getItemCount() {
        return states.size();
      }
      public static class ViewHolder extends RecyclerView.ViewHolder {
        final ImageView flagView;
        final TextView nameView, capitalView;
        ViewHolder(View view){
          super(view);
          flagView = (ImageView)view.findViewById(R.id.flag);
          nameView = (TextView) view.findViewById(R.id.name);
          capitalView = (TextView) view.findViewById(R.id.capital);
        }
      }
    }
```

Bu yerda biz avvalgi mavzudagi kodga nisbatan qo'shilgan fikrlarga to'xtalaman.

Avvalo, biz klik hodisasi tinglovchisi uchun interfeysni aniqlashimiz kerak. Buning uchun StateAdapter sinfida interfeys aniqlanadi:

```
interface OnStateClickListener{
  void onStateClick(State state, int position);
}
```

Interfeys State obyekti tanlanganida chaqirilishi kerak bo'lgan va tanlangan State ob'ektini va uning ro'yxatdagi o'rnini oladigan onStateClick() usulini belgilaydi.

Keyingi nuqta, bu interfeys ob'ektini saqlash va konstruktorda uning qiymatini olish uchun o'zgaruvchining adapter sinfidagi ta'rifi:

```
private final OnStateClickListener onClickListener;
StateAdapter(Context context, List<State> states, OnStateClickListener onClickListener) {
    this.onClickListener = onClickListener;
    //.......................
}
```

Shunday qilib, adapter kodidan tashqari, biz har qanday tinglovchini o'matib, uni adapterga o'tkaza olamiz.

Uchinchi moment, **onBindViewHolder** usulidagi ro'yxatdagi elementni bosganda, tinglovchini chaqirish usuli:

```
public void onBindViewHolder(StateAdapter.ViewHolder holder, int position) {
    State state = states.get(position);
    holder.flagView.setImageResource(state.getFlagResource());
    holder.nameView.setText(state.getName());
    holder.capitalView.setText(state.getCapital());
    // ishlov berishni bosing
    holder.itemView.setOnClickListener(new View.OnClickListener(){
      @Override
      public void onClick(View v)
      {
        // tinglovchi usulini chaqiring, unga ma'lumotlarni uzating
        onClickListener.onStateClick(state, position);
      }
    });
}
```

**ViewHolder** sinfida ro'yxatdagi bitta ob'ekt uchun interfeysni ifodalovchi va aslida **View** obyekti bo'lgan **itemView** maydoni mavjud. Va bu ob'ektda setOnClickListener() usuli bor, u orqali siz standart **OnClickListener** klik tinglovchisini ulashingiz va interfeysimiz usulini onClick() usuliga qo'ng'iroq qilib, kerakli ma'lumotlarni - tanlangan State obyekti va uning ro'yxatdagi o'rnini berishingiz mumkin. .

Savol tug'ilishi mumkin, nima uchun bu erda darhol elementni bosish kerak? Nima uchun qo'shimcha interfeys, uning o'zgaruvchisi yaratish va uning usulini chaqirish kerak? Albatta, biz bu erda sekin urish bilan shug'ullanishga harakat qilishimiz mumkin, lekin bu yaxshi yoki odatiy amaliyot emas, chunki biz MainActivity sinfidagi bosish jarayonini u erda (yoki boshqa biror biridan) aniqlangan kod asosida aniqlashni xohlashimiz mumkin. tashqarida joylashtiring) ...

Xuddi shu vizual interfeys **activity_main.xml** faylida qoladi:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp">
  <androidx.recyclerview.widget.RecyclerView
    android:id="@+id/list"
    android:layout_width="0dp"
    android:layout_height="0dp"
app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
```

```
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Va oxirida, **MainActivity** sinfini o'zgartiraylik:

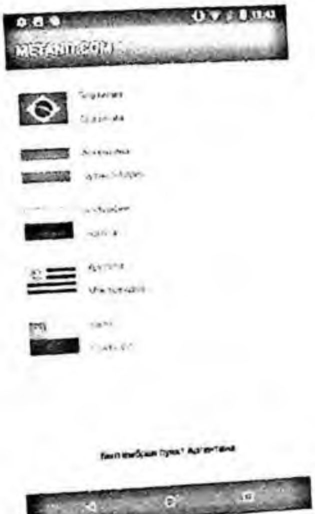
```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;
import android.os.Bundle;
import android.widget.Toast;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity {
   ArrayList<State> states = new ArrayList<State>();
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // ro'yxatning boshlanishi
      setInitialData();
      RecyclerView recyclerView = (RecyclerView) findViewById(R.id.list);
      // ro'yxatdagi elementni bosish uchun tinglovchini aniqlang
      StateAdapter.OnStateClickListener stateClickListener = new
StateAdapter.OnStateClickListener() {
         @Override
         public void onStateClick(State state, int position) {
            Toast.makeText(getApplicationContext(), "Был выбран пункт "
+ state.getName(),
                  Toast.LENGTH_SHORT).show();
         }
      };
      // adapter yarating
      StateAdapter adapter = new StateAdapter(this, states,
stateClickListener);
      // ro'yxat uchun adapterni o'rnating
      recyclerView.setAdapter(adapter);
   }
   private void setInitialData(){
      states.add(new State ("Бразилия", "Бразилиа", R.drawable.brazilia));
      states.add(new State ("Аргентина", "Буэнос-Айрес",
R.drawable.argentina));
      states.add(new State ("Колумбия", "Богота", R.drawable.columbia));
      states.add(new State ("Уругвай", "Монтевидео",
R.drawable.uruguai));
```

```
        states.add(new State ("Чили", "Сантьяго", R.drawable.chile));
    }
}
```

Adapter yaratilganda, MainActivity sinfida aniqlangan tinglovchi unga uzatiladi:

```
StateAdapter.OnStateClickListener     stateClickListener     =     new
StateAdapter.OnStateClickListener() {
    @Override
    public void onStateClick(State state, int position) {
        Toast.makeText(getApplicationContext(), "Был выбран пункт " +
state.getName(),
                Toast.LENGTH_SHORT).show();
    }
};
```

Bu faqat tanlangan ro'yxat elementi haqida qalqib chiquvchi xabami ko'rsatadi.



7.25-rasm. Dastur natijasi

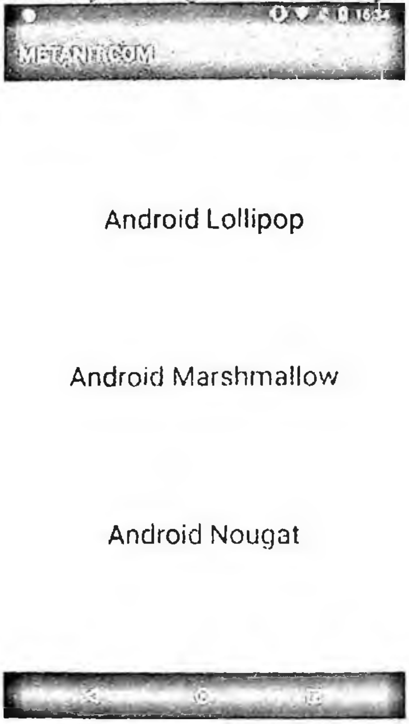
# VIII-BOB. MAVZULAR VA USLUBLAR

## 8.1. Uslublar

Biz elementni balandlik, kenglik, fon rangi, matn va boshqalarni belgilaydigan turli atributlar bilan sozlashimiz mumkin. Ammo agar bizda bir xil sozlamalardan foydalanadigan bir nechta element bo'lsa, biz bu sozlamalarni uslublarga birlashtirishimiz mumkin.

Masalan, bizda bir nechta TextView boshqaruv elementlari bor deylik:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/textView1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:textSize="28sp"
    android:textColor="#3f51b5"
    android:text="Android Lollipop"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/textView2"
    />
  <TextView
    android:id="@+id/textView2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:textSize="28sp"
    android:textColor="#3f51b5"
    android:text="Android Marshmallow"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/textView3"
    app:layout_constraintTop_toBottomOf="@+id/textView1"
    />
  <TextView
    android:id="@+id/textView3"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:textSize="28sp"
    android:textColor="#3f51b5"
    android:text="Android Nougat"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView2"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```



Android Lollipop

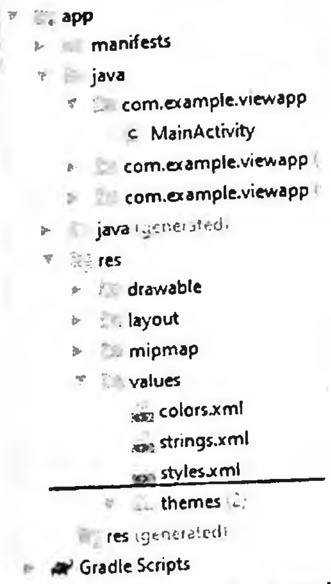
Android Marshmallow

Android Nougat

8.1-rasm. Dastur natijasi

Bu TextViewsning hammasi bir xil xususiyatlarga ega va masalan, agar biz matn rangini o'zgartirmoqchi bo'lsak, uni uchta TextViews uchun o'zgartirishimiz kerak bo'ladi. Bu yondashuv eng maqbul emas va eng maqbul yondashuv - **res/values** papkasida loyihada belgilangan uslublardan foydalanish.

Keling, **res/values** papkasidagi loyihaga **Value Resourse File** yangi elementini qo'shamiz, biz uni **style.xml** deb ataymiz:



8.2-rasm. Loyihada values papkasi

**styles.xml** faylida quyidagi tarkibni aniqlaylik:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="TextViewStyle">
      <item name="android:layout_width">0dp</item>
      <item name="android:layout_height">wrap_content</item>
      <item name="android:textColor">#3f51b5</item>
      <item name="android:textSize">28sp</item>
      <item name="android:gravity">center</item>
   </style>
</resources>
```

Bu erda TextViewStyle yangi uslubi, u TextView atributlari uchun qiymatlarni belgilash uchun itemsdan foydalanadi.

Uslub <style> elementi yordamida o'rnatiladi. name atributi uslub nomini bildiradi, undan keyin siz unga murojaat qilishingiz mumkin. Ixtiyoriy parent atributi berilgan uslub uchun ota -ona uslubini o'rnatadi, undan bola uslubi uning barcha xususiyatlarini oladi.

items vidjet uchun o'ziga xos xususiyatlarni o'rnatadi, u name atributining qiymati sifatida belgilanadigan xususiyat nomini oladi.

Endi uslubni qo'llang, activity_main.xml faylini o'zgartiring:

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

```
    <TextView
      android:id="@+id/textView1"
      style="@style/TextViewStyle"
      android:text="Android Lollipop"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintBottom_toTopOf="@+id/textView2"
      />
    <TextView
      android:id="@+id/textView2"
      style="@style/TextViewStyle"
      android:text="Android Marshmallow"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintBottom_toTopOf="@+id/textView3"
      app:layout_constraintTop_toBottomOf="@+id/textView1"
      />
    <TextView
      android:id="@+id/textView3"
      style="@style/TextViewStyle"
      android:text="Android Nougat"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/textView2"
      />
  </androidx.constraintlayout.widget.ConstraintLayout>
```

style="@style/TextViewStyle" ta'rifidan foydalanib, matn maydoni uslub ta'rifi bilan bog'langan. Yakuniy natija avvalgidek bo'ladi, faqat kod kamroq bo'ladi. Va agar biz ba'zi xususiyatlarni o'zgartirmoqchi bo'lsak, uslub ta'rifida kerakli item elementini o'zgartirish kifoya.

## 8.2. Mavzular

Alohida elementlarga individual uslublarni qo'llashdan tashqari, biz butun ilovani yoki activityni mavzu sifatida uslublashimiz mumkin. Mavzu odatda butun ilovaga, activity sinfiga yoki vidjet ierarxiyasiga tegishli bo'lgan atributlar to'plamini ifodalaydi.

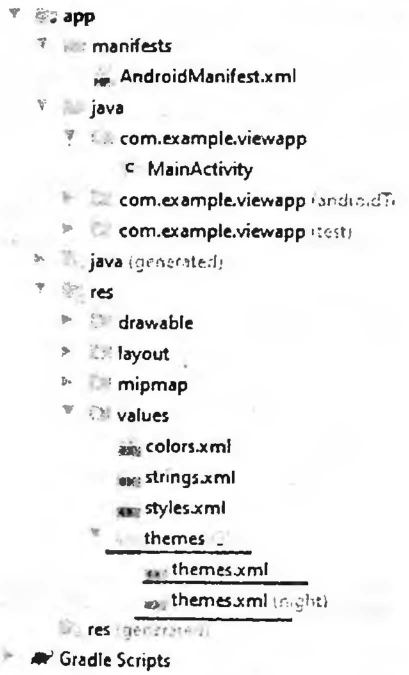
Biz o'zimiz mavzu yaratishimiz mumkin. Biroq, Android ilovani uslublash uchun oldindan belgilangan bir nechta mavzularni taqdim etadi, masalan, Theme.AppCompat.Light.DarkActionBar va boshqalar.

Odatiy bo'lib, ilova allaqachon mavzularni qo'llaydi. Shunday qilib, keling, **AndroidManifest.xml** faylini ochamiz. Unda biz ilovani ifodalovchi application elementining quyidagi ta'rifini ko'rishimiz mumkin:

```
  <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/Theme.ViewApp">
```

Mavzu android:theme atributi yordamida o'rnatiladi. Bunday holda, manba **Theme.ViewApp** deb nomlangan manba ishlatiladi. Odatiy bo'lib, mavzu fayllari **res/values** papkasida aniqlanadi. Xususan, shartli **themes** katalogini bu yerda topish mumkin, u boshlang`ich holat bo'yicha ikkita elementga ega: **themes.xml**:



```
app
  manifests
    AndroidManifest.xml
  java
    com.example.viewapp
      MainActivity
    com.example.viewapp (androidTest)
    com.example.viewapp (test)
  java (generated)
  res
    drawable
    layout
    mipmap
    values
      colors.xml
      strings.xml
      styles.xml
      themes
        themes.xml
        themes.xml (night)
  res (generated)
Gradle Scripts
```

8.3-rasm. Loyihada themes papkasi

Bir fayl ochiq mavzuni, ikkinchisi qorong'i mavzuni ifodalaydi. Masalan, themes.xml faylini engil mavzu bilan oching:

```
    <resources xmlns:tools="http://schemas.android.com/tools">
      <!-- Base application theme. -->
      <style                                    name="Theme.ViewApp"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item                        name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Bu erda biz **style** elementidan foydalangan holda uslub va uslub aniqlanganini ko'rishimiz mumkin. **parent** atributi joriy mavzuni barcha uslubiy xususiyatlarini olgan asosiy mavzuni ko'rsatadi. Ya'ni, "Theme.ViewApp" mavzusi boshqa mavzudan - "Theme.MaterialComponents.DayNight.DarkActionBar" dan foydalanadi. Bundan tashqari, u o'ziga xos uslublarni belgilaydi.

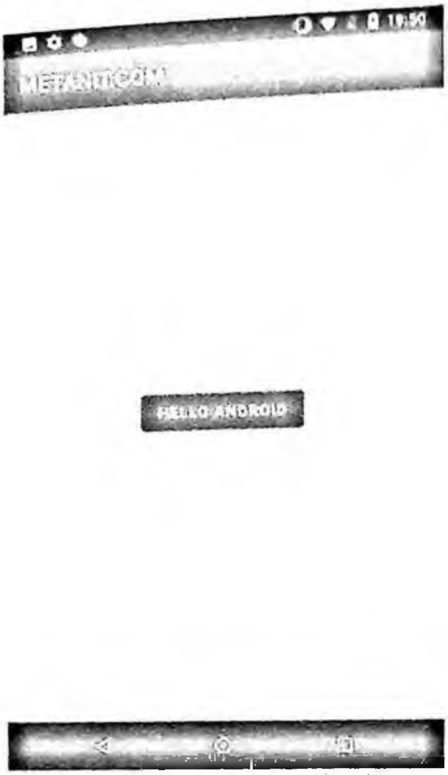
Bundan tashqari, bu erda atributlar uchun xarakteristikalargina emas, balki semantik ismlar ham aniqlanganini ko'rishingiz mumkin, masalan, colorPrimary, unga "@color/purple_500" manbasi xaritasi qo'yiladi.

Agar kerak bo'lsa, biz bu xususiyatlarni o'zgartirishimiz yoki mavzuga yangi uslubiy xususiyatlarni qo'shishimiz mumkin. Masalan, colorPrimary xususiyatining rangini o'zgartiramiz, u boshqa narsalar qatorida sarlavha va tugmaning fon rangi sifatida ishlatiladi:

```
<item name="colorPrimary">#1565C0</item>
```

Va sarlavha va tugma fonining standart rangi mos ravishda o'zgaradi:

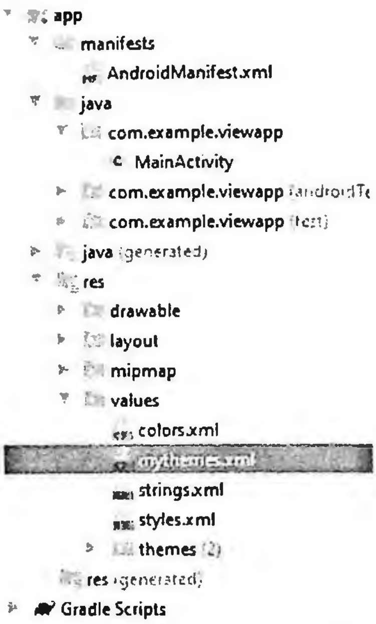
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello Android"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```



8.4-rasm. Dastur natijasi

**O'zingizning mavzuni yarating.** O'rnatilgan mavzularni ishlatish o'rniga, biz o'zimizni yaratishimiz mumkin. Buning uchun **res/values** papkasiga yangi **mythemes.xml** faylini qo'shing va undagi quyidagi tarkibni aniqlang:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="MyTheme" parent="Theme.AppCompat.Light">
    <item name="android:textColor">#FF018786</item>
    <item name="android:textSize">28sp</item>
  </style>
</resources>
```



8.5-rasm. Loyihada values papkasi

Shunday qilib, biz Theme.AppCompat.Light uslubidan meros bo'lib qolgan "MyTheme" uslubini yaratdik. Bu uslubda biz ikkita xususiyatni bekor qildik: shrift balandligi (textSize) 28sp va matn rangi (textColor) - # FF018786.

Endi bu uslubni **AndroidManifest.xml** faylidagi dastur mavzusi sifatida belgilaylik:

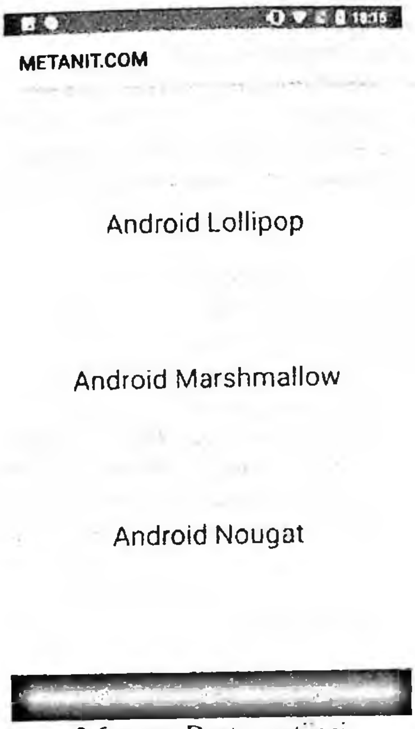
```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/MyTheme"><!-- dastur mavzusi -->
```

**activity_main.xml** da quyidagi belgilashga ega bo'laylik

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/textView1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
```

```
        android:gravity="center"
        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Marshmallow"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintTop_toBottomOf="@+id/textView1"
        />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Nougat"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Ko'rib turganingizdek, textSize va textColor atributlari TextView elementlari uchun o'rnatilmagan, biroq ular butun dunyoda bizning ilovamizga tegishli bo'lgan mavzuda aniqlanganligi sababli, TextView elementlari quyidagi uslubiy xususiyatlarga ega bo'ladi:


METANIT.COM

Android Lollipop

Android Marshmallow

Android Nougat

8.6-rasm. Dastur natijasi

**activity uchun mavzuni qo'llash**. Yuqoridagi mavzular butun dunyoda butun ilovaga tatbiq etilgan. Lekin siz ularni alohida Activity sinfiga qo'llashingiz mumkin. Buning uchun siz AndroidManifest manifest faylini to'g'rilashingiz kerak. Misol uchun:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.viewapp"
    android:versionCode="1"
    android:versionName="1.0">
    <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/Theme.ViewApp">
        <activity    android:name=".MainActivity"    android:theme="@style/
MyTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

<activity> elementining **android:theme** atributi MainActivityga qo'llaniladigan mavzuni bildiradi. Ya'ni, "Theme.ViewApp" mavzusi butun dunyo bo'ylab, "M_yTheme" esa MainActivity uchun qo'llaniladi.

**Vidjetlar ierarxiyasiga mavzuni qo'llash**. Mavzuni ishlatmoqchi bo'lgan elementga **android:theme** atributini o'rnatish orqali vidjet ierarxiyasiga mavzuni qo'llash ham mumkin. Masalan, ConstraintLayout va uning elementlariga mavzuni qo'llash:

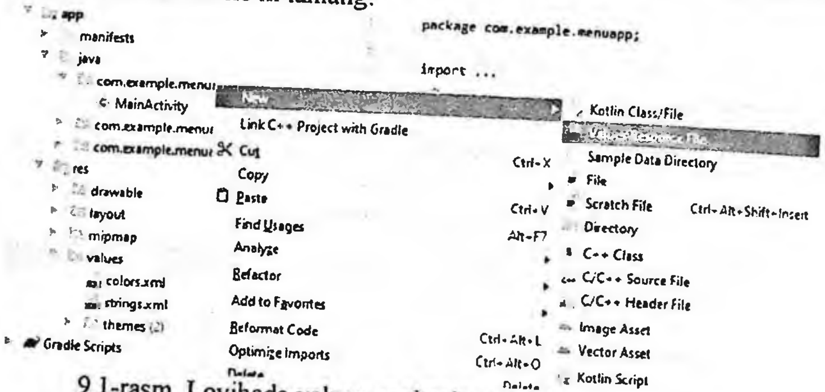
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:theme="@style/MyTheme">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

# IX BOB. Menyu

## 9.1. Menyu yaratish.

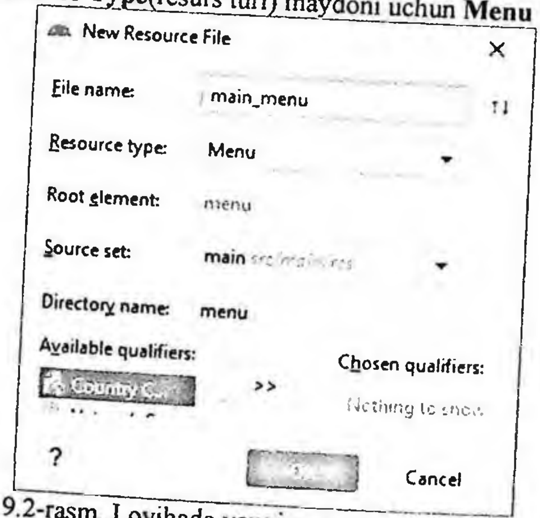
Ilovalar menyusi **android.view.Menu** sinfini ifodalaydi va har bir faoliyat shu turdagi ob'ekt bilan bog'liq. Android.view.Menu obyekti boshqa elementlar sonini o'z ichiga olishi mumkin, ular o'z navbatida pastki elementlarni saqlashi mumkin.

**Xmlda menyuning ta'rifi.** Menyu, masalan, interfeys yoki rasm fayllari ham resurs hisoblanadi. Ammo, Empty Activity bilan yangi loyiha yaratishda, boshlang`ich holat bo'yicha menyu resurslari yo'q, shuning uchun agar kerak bo'lsa, ularni qo'lda qo'shish kerak. Shunday qilib, loyihadagi menyu resurslarini aniqlash uchun res katalogidagi loyihani o'ng tugmasini bosing va ochilgan ro'yxatda **New - > Android Resource File** ni tanlang:



9.1-rasm. Loyihada values papkasiga yangi resurs qo`shish.

Keyin, paydo bo'lgan oynada, fayl nomi uchun **main_menu** nomini ko'rsating va **Resource Type**(resurs turi) maydoni uchun **Menu** ni tanlang:



New Resource File

File name: main_menu

Resource type: Menu

Root element: menu

Source set: main src/main/res

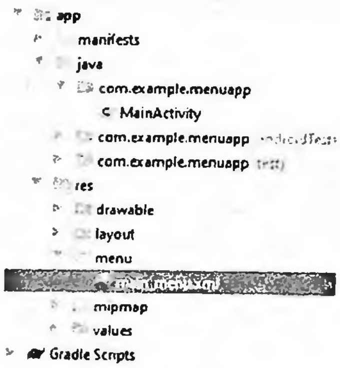
Directory name: menu

Available qualifiers: Country Code

Chosen qualifiers: Nothing to show

9.2-rasm. Loyihada yangi resursga nom berish.

Shundan so'ng, res katalogida menu ostki katalogi yaratiladi, unda main_menu.xml fayli bo'ladi.



9.3-rasm. Loyihada menu papkasi

Odatiy bo'lib, bu fayl bitta bo'sh menu elementini belgilaydi:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
</menu>
```

Keling, bir nechta fikrlarni belgilab, fayl mazmunini o'zgartiraylik:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="1"
        android:title="Настройки" />
    <item
        android:id="@+id/save_settings"
        android:orderInCategory="3"
        android:title="Сохранить" />
    <item
        android:id="@+id/open_settings"
        android:orderInCategory="2"
        android:title="Открыть" />
</menu>
```

<menu> tegi faylning ildiz tugunidir va bir yoki bir nechta <item> va <group> elementlaridan tashkil topgan menyuni belgilaydi.

<item> elementi menyu elementlaridan biri bo'lgan MenuItem ob'ektini ifodalaydi. Bu element ichki menyu yaratadigan <menu> ichki elementini o'z ichiga olishi mumkin.

<item> elementi tashqi ko'rinishi va xulq -atvorini belgilaydigan quyidagi atributlarni o'z ichiga oladi:

- **android:id**: menyu elementining yagona identifikatori, uni foydalanuvchi tanlaganida va id resurssini qidirish orqali topishga imkon beradi.
- **android:icon**: element uchun tasvirni o'rnatadigan drawable resursiga havola (android:icon="@drawable/ic_help")

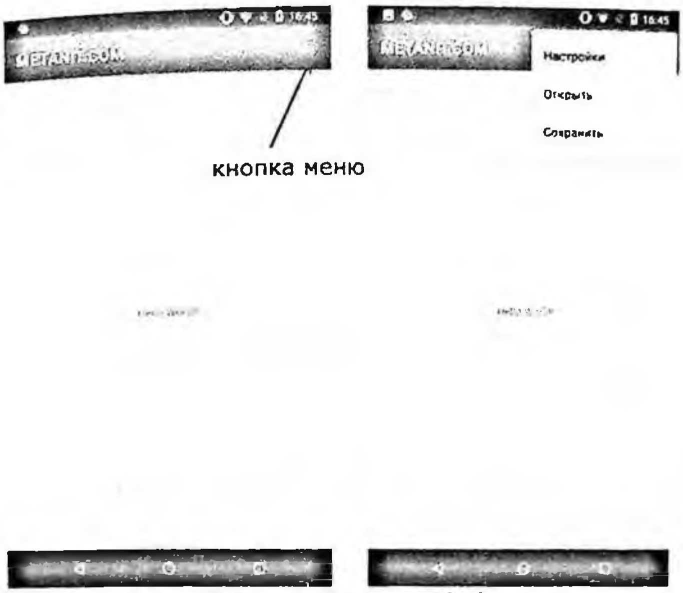
- **android:title**: element sarlavhasini o'z ichiga olgan satr resurssiga havola. Odatiy "Settings".
- **android:orderInCategory**: menyudagi elementlarning tartibi

**Menyuni elementlar bilan to'ldirish.** Biz uchta elementdan iborat menyuni aniqladik, lekin fayldagi elementlarning ta'rifi hali menyu yaratmagan. Bu faqat deklarativ tavsif. Uni ko'rsatish uchun biz uni Activity sinfida ishlatishimiz kerak. Buning uchun **onCreateOptionsMenu** usulini bekor qilish kerak. Shunday qilib, MainActivity sinfiga boramiz va uni shunday o'zgartiramiz:

```
package com.example.menuapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.main_menu, menu);
      return true;
   }
}
```

GetMenuInflater usuli inflate() usuli chaqiriladigan MenuInflater obyektini qaytaradi. Bu usul birinchi parametr sifatida xml dagi deklarativ menyu tavsifimizni ifodalovchi resurs oladi va men_u ob'ektini ikkinchi parametr sifatida to'ldiradi.

Standart dasturni ishga tushiramiz va yuqori o'ng burchakdagi menyu tugmasini bosamiz:



9.4-rasm. Dastur natijasi

**Menyuda bosishlarni boshqarish.** Agar menyu elementlaridan birini ossak, hech narsa bo'lmaydi. Menyuga amallarni bog'lash uchun, activity sinfida **nOptionsItemSelected**ni bekor qilishimiz kerak.

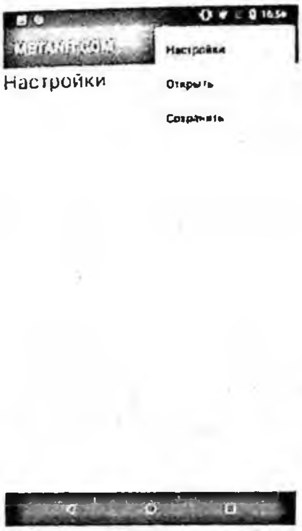
**activity_main.xml** faylida tanlangan menyu bandini ko'rsatish uchun 1=header bilan matn maydonini belgilang:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >
  <TextView
    android:id="@+id/selectedMenuItem"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="28sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity sinfini o'zgartiraylik:

```
package com.example.menuapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        TextView           headerView          =          (TextView)
findViewById(R.id.selectedMenuItem);
        switch(id){
            case R.id.action_settings :
                headerView.setText("Настройки");
                return true;
            case R.id.open_settings:
                headerView.setText("Открыть");
                return true;
            case R.id.save_settings:
                headerView.setText("Сохранить");
                return true;
        }
        //headerView.setText(item.getTitle());
        return super.onOptionsItemSelected(item);
    }
}
```

Qaysi menyu elementi tanlanganligini tushunish uchun avval unin identifikatorini olamiz int id=item.getItemId(). Keyin biz switch..cas konstruktsiyasidan o'tamiz va kerakli variantni tanlaymiz va tanlovga qarab ma'lun harakatlami bajaramiz - bu holda TextView matnini o'rnating.



9.5-rasm. Dastur natijasi

Ta'kidlash joizki, bu holda, agar bizning vazifamiz tanlangan menyu bandining matnini ko'rsatish bo'lsa, switch konstruktsiyasi o'rniga biz shunday yozishimiz mumkin edi:

*headerView.setText(item.getTitle());*

**Menyuni dasturiy ravishda yaratish.** Menyu elementlarini xml da belgilashdan tashqari, dasturiy ravishda menyuni ham yaratishingiz mumkin. Yangi menyu elementlarini qo'shish uchun **Menu** sinfining **add()** usulidan foydalaning.

Masalan, **MainActivity** kodini o'zgartiraylik:

```
package com.example.menuapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add("Настройки");
    menu.add("Открыть");
    menu.add("Сохранить");
    return true;
  }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        String title = item.getTitle().toString();
        TextView              headerView              =              (TextView)
findViewById(R.id.selectedMenuItem);
        headerView.setText(title);
        return super.onOptionsItemSelected(item);
    }
}
```

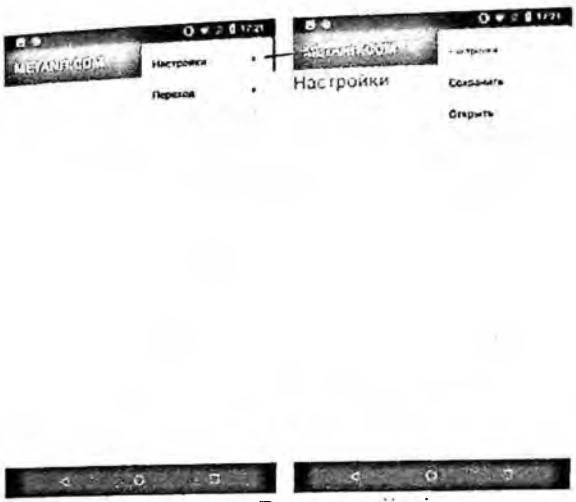
add() usulining ishlatilgan versiyasi menyu bandining nomini qabul qiladi.

## 9.2. Menyu va qism menyulardagi guruhlar.

**Qism menyu yaratish.** Menyu belgilash faylida qism menyu yaratish uchun ɜnu ichki elementini aniqlang:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_settings"
        android:title="Настройки">
        <menu>
            <item android:id="@+id/save_settings"
                android:title="Сохранить" />
            <item android:id="@+id/open_settings"
                android:title="Открыть" />
        </menu>
    </item>
    <item
        android:id="@+id/action_move"
        android:title="Переход">
        <menu>
            <item android:id="@+id/forward"
                android:title="Вперед" />
            <item android:id="@+id/back"
                android:title="Назад" />
        </menu>
    </item>
</menu>
```

Menyu bosilgandan so'ng, yuqori darajadagi elementlar ko'rsatiladi, uni osish orqali biz qism menyu ga o'tamiz:



9.6-rasm. Dastur natijasi

Menyuda guruhlar. group elementidan foydalanish menyu elementlarini guruhlarga ajratish imkonini beradi:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <group android:checkableBehavior="single">
    <item
      android:id="@+id/action_settings"
      android:title="Настройки"
      android:checked="true" />
    <item android:id="@+id/save_settings"
      android:title="Сохранить" />
    <item android:id="@+id/open_settings"
      android:title="Открыть" />
  </group>
</menu>
```

Guruh ta'rifida biz **android:checkableBehavior** atributini o'rnatishimiz mumkin. Bu atribut quyidagi qiymatlarni olishi mumkin: single (har bir element uchun radio tugmasi yaratiladi), all (har bir element uchun katakcha yaratiladi) va none.

Bunday holda, har bir element uchun radio tugmasi (vizual ravishda aylana) yaratiladi. Va tekshirilgan radio tugmasi birinchi element uchun o'rnatiladi (android:checked="true").

Interfeys belgilash faylida **activity_main.xml** shuningdek, matn maydonini aniqlashga ruxsat beradi:

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

```
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/selectedMenuItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="28sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Va MainActivity sinfida, tanlangan menyu bandining yonidagi radio tugmachasini tanlashini aniqlaylik:

```
package com.example.menuapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        TextView headerView = (TextView)
findViewById(R.id.selectedMenuItem);
        switch(id){
            case R.id.action_settings :
                headerView.setText("Настройки");
                return true;
            case R.id.open_settings:
                headerView.setText("Открыть");
                return true;
            case R.id.save_settings:
                headerView.setText("Сохранить");
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Настройки
Создать
Открыть

9.7-rasm. Dastur natijasi

**Menyu va qism menyularda guruhlarni dasturiy yaratish.** Bundan tashqari, guruhlar va qism menyular dasturiy jihatdan yaratilishi mumkin. Shunday qilib, **MainActivity** kodini o'zgartiraylik:

```
package com.example.eugene.menuapp;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0        // Группа
          ,1       // id
          ,0       //tartibi
          ,"Создать"); // sarlavhasi
    menu.add(0,2,1,"Открыть");
    menu.add(0,3,2,"Сохранить");
    return true;
  }
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
```

```
        TextView headerView = (TextView) findViewById(R.id.header);
        switch(id){
          case 1 :
            headerView.setText("Создать документ");
            return true;
          case 2:
            headerView.setText("Открыть документ");
            return true;
          case 3:
            headerView.setText("Сохранить документ");
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Bu erda ishlatilgan add() usulining versiyasi menyuga element qo'shib, quyidagi parametrlarni qabul qiladi: guruh raqami, id, menyudagi elementlarning tartibi va element sarlavhasi.

METANIT.COM
Создать докуме
Создать
Открыть
Сохранить

9.8-rasm. Dastur natijasi

**Foydalanilgan adabiyotlar**

1. Reto Meier. Ian Lake. Professional Android, Fourth Edition. Jhon Wiley & Sons, Inc,
2. https://metanit.com/java/android/ - Java da Android uchun dasturlash
3. Reto Meier. Ian Lake. Professional Android, Fourth Edition. Jhon Wiley & Sons, Inc, 2018.
4. Ian F. Darwin. Androir Cookbook. Problems and Solutions for Android Developers. O'Reil Media, 2017.
5. Билл Филипс, К.Стюарт., Книга «Android. Программирование для профессионалов. 3 издание», Питер 2017.