J.J. Atamuradov, S.S. Salimov

MOBIL ILOVALAR YARATISH

O‘quv qo‘llanma

60610100 – Kompyuter ilmllari va dasturlash texnologiyalari
60610200 – Axborot tizimlari va texnologiyalari

Ushbu o‘quv qo‘llanma Oliy ta’lim, fan va innovatsiyalar vazirligi Buxoro davlat universitetining 2024-yil 29-apreldagi 281-soni buyrug‘iga asosan nashr etishga ruxsat berildi.

“KAMOLOT” nashriyoti
Buxoro - 2024

UO‘K: 004.658.2
KBK: 32.913
A90

Salimov Suhrobjon Sobirovich, Atamuradov Jamshid Jalilovich.
Mobil ilovalar yaratish. / [Matn]: o‘quv qo‘llanma / J.J.Atamuradov, S.S.Salimov - Buxoro : “BUXORO DETERMINANTI” MCHJning Kamolot nashriyoti, 2024. - 176 b.

ISBN: 978-9910-729-06-5

Mazkur o‘quv qo‘llanma bakalavriatura 60610100 - Kompyuter ilmlari va dasturlash texnologiyalari, 60610200 - Axborot texnologiyalari va tizimlari yo‘nalishi uchun O‘zbekiston Respublikasi oliy ta’lim fan va innovatsiyalar vazirligi, Buxoro davlat universitetining 2024 yil “29” apreldagi 281 -sonli buyrug‘i bilan tasdiqlangan “Mobil ilovalar yaratish” namunaviy fan dasturiga muvofiq tuzilgan.

Ushbu o‘quv qo‘llanmada talabalarga davlat ta’lim standartlariga mos bilim va ko‘nikmalarni hosil qilishni ta’minlaydi, dunyoqarash va tizimli fikrlashni shakllantirishga ko‘maklashadi, hamda shu sohadagi mutaxassislarga zamonaviy dasturlash texnologiyalari usullarini va talablarga javob beradigan yuqori sifatli dasturiy ta’minotni o‘rgatadi.

O‘quv qo‘llanma 3-bosqich Kompyuter ilmlari va dasturlash texnologiyalari, Axborot texnologiyalari va tizimlari yo‘nalishining talabalari uchun mo‘ljallangan bo‘lib unda mobil ilovalarni yaratishning konseptsiyalari, dasturiy injenering usullari, Android studio va ob’ektga yo‘naltirilgan tillar yordamida tizimli loyihalash usullarini amaliyotda tadbiq etish, korporativ ilovalarini ishlab chiqish, dasturiy komplekslarini loyihalashda kollektiv ishlab chiqish usullarini, modellashtirish tillaridan foydalangan holda predmet sohalarini tahlil etishni, diagrammallarni qurish va tizimli tahlil hamda loyihalash usullarini bilishi va ulardan foydalana olishi, dasturiy tizimlarni tizimli tahlil va loyihalashning asosiy prinsiplari va qoidalari; mobil dasturiy ilovalarni ishlab chiqishda zamonaviy dasturlash vositalarini tadbiq qilish bo‘yicha nazariy ma‘lumotlar berilgan.

Taqrizchilar:

T.R. Shafiyev - Buxoro davlat universiteti, t.f.f.d (Phd) dotsent.
N.N. Zaripov- Buxoro davlat pedagogika istituti, p.f.f.d dotsent.

MUNDARIJA

KIRISH

Mobil ilovalarni ishlab chiqish axborot texnologiyalarining tez rivojlanayotgan sohasidir. Buning bir qancha sabablari bor:

- juda ko‘p sonli turli xil mobil telefonlar va smartfonlarning paydo bo‘lishi;
- mobil Internet qamrovini faol oshirish;
- mobil qurilmalarning soddaligi va arzonligi.

Mobil qurilmalarning aksariyati ikkita operatsion tizimda (OS) ishlaydi - Android va iOS, taxminan 74% va 25% ni egallaydi. Shunga ko‘ra, boshqa barcha operatsion tizimlarning ulushi 1% dan oshmaydi. Agar mutlaq raqamlar haqida gapiradigan bo‘lsak, Android operatsion tizimiga asoslangan faol qurilmalar soni 3 milliardga, iOS operatsion tizimiga asoslanganlari esa 1 milliarddan oshadi.

Android va iOS uchun ilovalarni ishlab chiqish umumiy tamoyilga ega, va sezilarli farqlar mavjud. Avval umumiy tamoyillarni ko‘rib chiqaylik. Operatsion tizimdan qat'iy nazar, har qanday mobil ishlab chiquvchi buni bilishi kerak turli ekranlarga moslashuvchi tartib asosidagi interfeyslarni yaratish. Tarmoq so‘rovlarini yaratish va ulardan javob olish imkoniyati ham bir xil darajada muhimdir.

Ma'lumotlarni qayta ishlash qobiliyati ham foydali bo‘ladi, bu birinchi navbatda JSON formatiga tegishli (matn formati JavaScript-ga asoslangan ma'lumotlar almashinuvi), bu mijoz-server mobil ilovalari uchun de-fakto standart formatdir.

Ushbu qo‘llanma mobil qurilmalar uchun dasturiy ta'minotni ishlab chiqishni o‘rganish uchun mo‘ljallangan. Qo‘llanmaning asosiy maqsadi Android platformalarida mobil ilovalarni ishlab chiqish bilan tanishtirishdir. Kitobda mobil qurilmalar va texnologiyalar rivojlanishining qisqacha tarixi yoritilgan, mobil ilovalarning mavjud tasnifi keltirilgan va Java dasturlash tilidan foydalangan holda ularni rivojlantirishning asosiy bosqichlari ko‘rsatilgan. O‘quv qo‘llanma Java ilovalari bilan ishlash tajribasiga ega bo‘lgan foydalanuvchilar uchun mo‘ljallangan. O‘quvchi nafaqat mobil ilovalarni yaratish asoslari bilan tanishishi, balki ularni mavjud xizmatlarga moslashtirish xususiyatlarini o‘rganishi, shuningdek, zamonaviy muhitda yangi kontent uchun dastur kodlarini loyihalash va yozish bo‘yicha qimmatli ko‘rsatmalarga ega bo‘lishi mumkin. Internet texnologiyalari va turli axborot tizimlarida.

1. MOBIL OPERATSION TIZIMLARINING RIVOJLANISH TARIXI. ANDROID OPERATSION TIZIMI.

Reja:

1.1 Mobil ilovalar tarixi.

1.2 Android operatsion tizimi tarixi.

Mobil **operatsion tizim** smartfonlar, planshetlar, aqlli soatlar yoki boshqa noutbuk bo‘lmagan shaxsiy mobil hisoblash qurilmalari uchun ishlatiladigan operatsion tizimdir. Oddiy/mobil noutbuklar kabi kompyuterlar "mobil" bo‘lsa-da, ularda ishlatiladigan operatsion tizimlar odatda mobil hisoblanmaydi, chunki ular dastlab tarixan o‘ziga xos *mobil* funksiyalarga ega bo‘lmagan yoki kerak bo‘lmagan ish stoli kompyuterlari uchun mo‘ljallangan. Mobil va boshqa shakllarni ajratib turadigan bu chiziq so‘nggi yillarda ancha pasaydi, chunki yangi qurilmalar o‘tmishdagi apparatlardan farqli ravishda kichikroq va mobilroq bo‘lib qolgan.

Mobil operatsion tizimlar ish stoli kompyuteri operatsion tizimining xususiyatlarini mobil yoki qo‘lda foydalanish uchun foydali bo‘lgan boshqa funktsiyalar bilan birlashtiradi va odatda simsiz o‘rnatilgan modem va telefon va ma'lumotlar ulanishi uchun SIM-kartani o‘z ichiga oladi. 2018-yilning 1-choragida 123 milliondan ortiq smartfon sotilgan (eng yuqori ko‘rsatkich), ularning 60,2 foizi Android va 20,9 foizi iOS tizimida ishlaydi. 2012-yilda (1,56 milliard), 2023-yilda ham sotuvlar oshib bordi, aniqrog‘i 1,43 milliard , 53,32% Android . Androidning o‘zi mashhur ish stoli operatsion tizimi Microsoft Windows ga qaraganda ko‘proq sotadi va umuman smartfondan foydalanish (hatto planshetlarsiz ham) ish stolidan foydalanishdan ko‘proqdir.

Android operatsion tizimi haqida batafsil tanishib chiqamiz.

Android – bu Linux yadrosining o‘zgartirilgan versiyasiga va boshqa ochiq kodli dasturlarga asoslangan mobil operatsion tizim (32-bit va 64-bit), asosan smartfon va planshetlar kabi sensorli mobil qurilmalar uchun mo‘ljallangan. Android Open Handset Alliance deb nomlanuvchi dasturchilar konsorsiumi tomonidan ishlab chiqilgan, garchi uning eng keng tarqalgan versiyasi birinchi navbatda Google tomonidan ishlab chiqilgan. U 2007 yil noyabr oyida taqdim etilgan

bo‘lib, birinchi tijorat Android qurilmasi HTC Dream 2008 yil sentyabr oyida taqdim etilgan.

Asosiysi, operatsion tizim Android Open Source Project (AOSP) sifatida tanilgan va asosan Apache Litsenziyasi ostida litsenziyalangan bepul va ochiq kodli dasturiy ta'minotdir (FOSS). Biroq, aksariyat qurilmalar Google tomonidan ishlab chiqilgan xususiy Android versiyasida ishlaydi, u oldindan o‘rnatilgan qo‘shimcha xususiy yopiq manbali dasturiy ta'minot bilan birga, ayniqsa, Google Chrome kabi asosiy ilovalarni o‘z ichiga olgan Google Mobile Services (GMS), raqamli tarqatish platformasi Google Play, va tegishli Google Play xizmatlarini ishlab chiqish platformasi.

Android 2011-yildan beri smartfonlar va 2013-yildan beri planshetlar bo‘yicha dunyo bo‘ylab eng ko‘p sotiladigan OT hisoblanadi. 2021-yil may holatiga ko‘ra uning oylik uch milliarddan ortiq faol foydalanuvchisi bor edi, bu dunyodagi har qanday operatsion tizimning eng katta o‘rnatilgan bazasi va 2021-yil yanvar oyida Google Play do‘konida 3 milliondan ortiq ilovalar mavjud edi. 2023-yil 4-oktabrda chiqarilgan Android 14 eng so‘nggi versiya bo‘lib, yaqinda chiqarilgan Android 12.1/12L buklanadigan telefonlar, planshetlar, ish stoli o‘lchamli ekranlar va Chromebook qurilmalariga xos yaxshilanishlarni o‘z ichiga oladi.

Android uchun asosiy apparat platformasi ARM (ARMv7 va ARMv8-A arxitekturalari) bo‘lib, x86 va x86-64 arxitekturalari Androidning keyingi versiyalarida ham rasman qo‘llab-quvvatlanadi. 2012 yildan boshlab Intel protsessorli Android qurilmalari, jumladan telefonlar va planshetlar paydo bo‘la boshladi. 64-bitli platformalar uchun qo‘llab-quvvatlanar ekan, Android dastlab 64-bitli x86-da, keyin esa ARM64- da ishlay boshladi. RISC-V arxitekturasi uchun operatsion tizimning norasmiy eksperimental porti 2021-yilda chiqarilgan.

Android qurilmalari ko‘plab ixtiyoriy apparat komponentlarini o‘z ichiga oladi, jumladan, harakatsiz yoki videokameralar, GPS, orientatsiya sensorlari, maxsus o‘yin boshqaruvlari, akselerometrlar, giroskoplar, barometrlar, magnitometrlar, yaqinlik sensorlari, bosim sensorlari, termometrlar va sensorli ekranlar.

Google har yili Android relizlarini yangi qurilmalarga o‘rnatish va mavjud qurilmalarni yangilash uchun taqdim etadi. Eng so‘nggi versiya Android 15.

Quyidagi jadvalda Android operatsion tizimining so‘ngi 8 ta versiyalari haqida ma’lumotlar keltirilgan.

Nomi	Ichki kod nomi	Versiyasi	Ishlab chiqarilish sanasi	So‘ngi xavfsizlik tuzatish sanasi	GooglePlay xizmatlarining so‘ngi versiyasi(chiqarish sanasi)
Android Pie	Pistachio Ice Cream	9	Avgust, 2018	Yanvar 2022	Mart 2024
Android 10	Quince Tart	10	Sentyabr, 2019	Fevral 2023	Mart 2024
Android 11	Red Velvet Cake	11	Sentyabr, 2020	Fevral 2024	Mart 2024
Android 12	Snow Cone	12	Oktyabr, 2021	Mart 2024	Mart 2024
Android 12L	Snow Cone v2	12.1	Mart, 2022	Mart 2024	Mart 2024
Android 13	Tiramisu	13	Avgust, 2022	Mart 2024	Mart 2024
Android 14	Upside Down Cake	14	Oktyabr, 2023	Mart 2024	Mart 2024
Android 15	Vanilla Ice Cream	15	Mart, 2022	Mart 2024	Mart 2024

Nazorat savollari.

1. Android operatsion tizimi haqida nimani bilasiz?
2. Android operatsion tizimi birinchi navbatda qaysi qurilmaga o‘rnatildi?
3. Android operatsion tizimining dasturiy tomoni haqida izoh bering.
4. Android operatsion tizimining apparat platformalarini tavsiflang.

2. DASTURLASH MUHITI VA UNI SOZLASH. MAXSUS INSTRUMENTAL DASTURIY VOSITALARNI O‘RNATISH VA SOZLASH. EMULYATORLARDAN FOYDALANISH.

Reja:

2.1 Dasturlash muxiti o‘rnatish va uni sozlash.

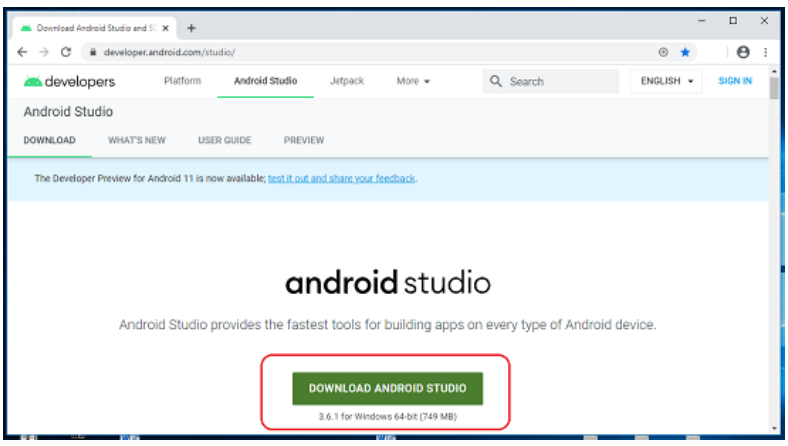
2.2 Maxsus instrumental dasturiy vositalarini urnatish va sozlash.

Android Studio - bu JetBrains IntelliJ IDEA dasturiy ta'minoti asosida qurilgan va Androidni rivojlantirish uchun maxsus ishlab chiqilgan Google Android operatsion tizimi uchun rasmiy integratsiyalashgan ishlab chiqish muhiti (IDE). Uni Windows, macOS va Linux asosidagi operatsion tizimlarda yuklab olish mumkin. Quyida mazkur muhitni o‘rnatish ketma ketligini ko‘rib o‘tamiz.

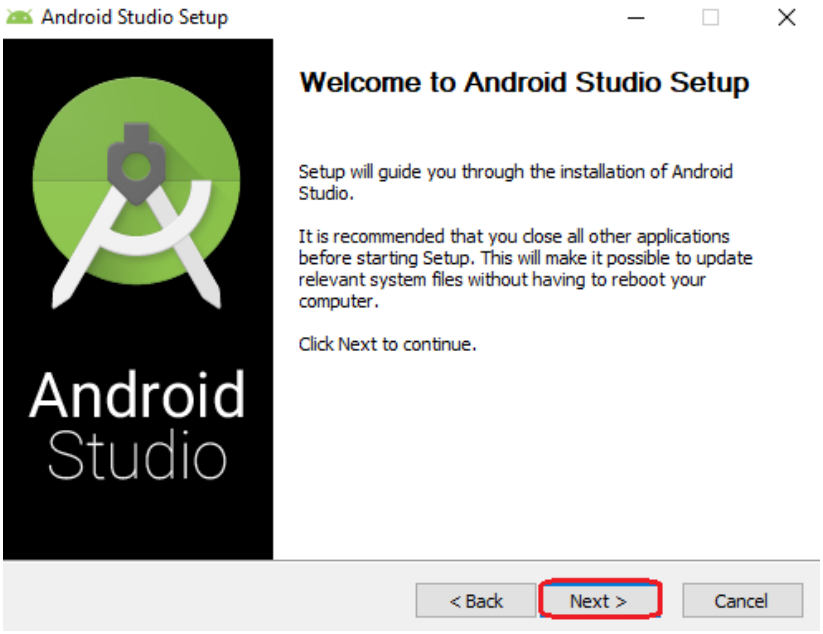
Android Studio-ni yuklab olish va o‘rnatishdan oldin quyidagi talablar muhim ahamiyatga ega.

- **Operatsion tizim versiyasi:** Microsoft Windows 7/8/10/11 (32-bit yoki 64-bit).
- **Tezkor xotirasi (RAM):** Minimal 4 GB RAM va 8 GB RAM tavsiya etiladi.
- **Bo‘sh disk maydoni:** Minimal 2 GB va 4 GB tavsiya etiladi.
- **Minimal talab qilinadigan JDK versiyasi:** Java Development Kit (JDK) 8.
- **Minimal ekran o‘lchamlari:** 1280 * 800. o‘lchamlari.

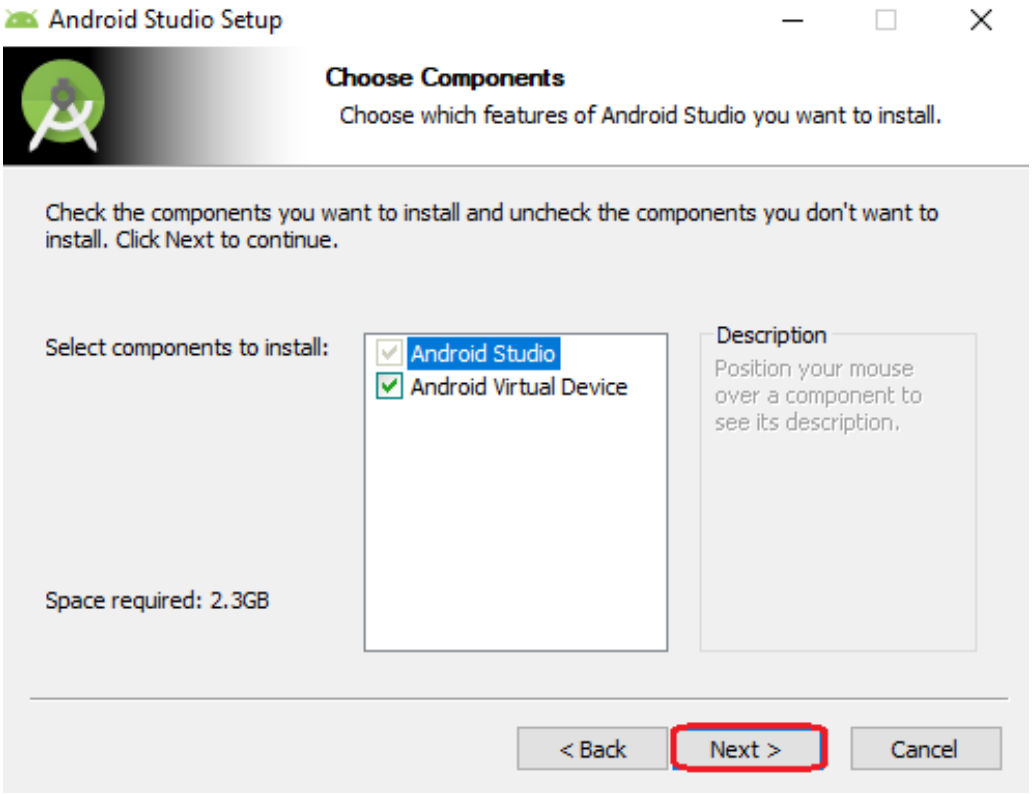
Android Studioni yuklab olish uchun veb-brauzeringizdagi rasmiy Android Studio veb-saytiga tashrif buyurib yuklab olishingiz mumkin.



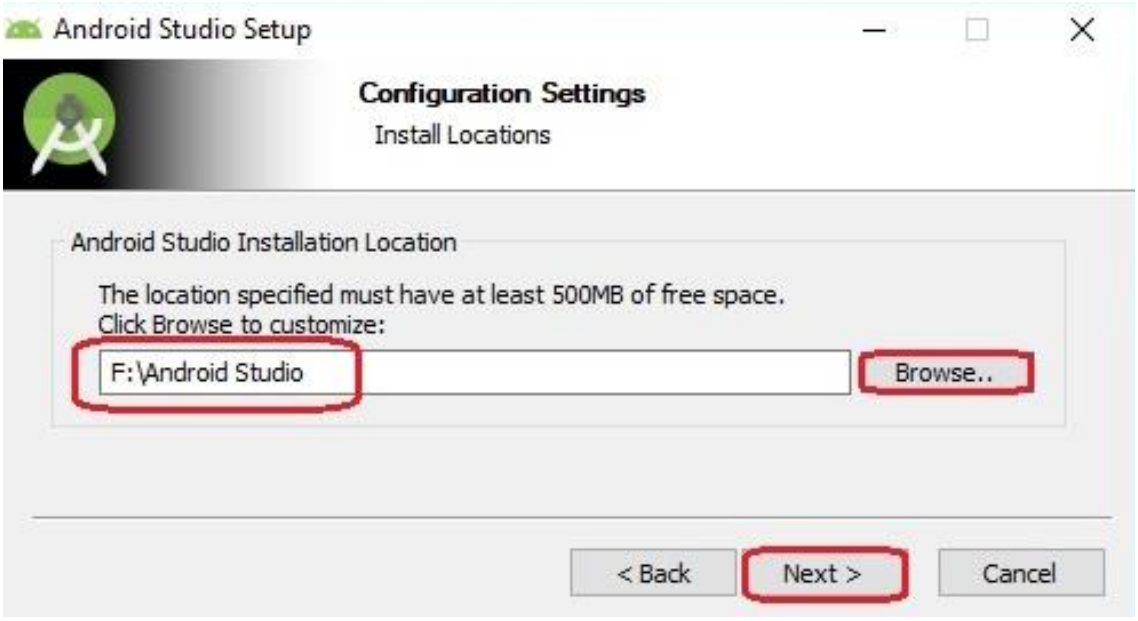
Yuklab olingan "Android Studio-ide.exe" faylini ikki marta bosing. Ekranda "Android Studio Setup" paydo bo‘ladi va davom etish uchun "Keyingi" tugmasini bosing.



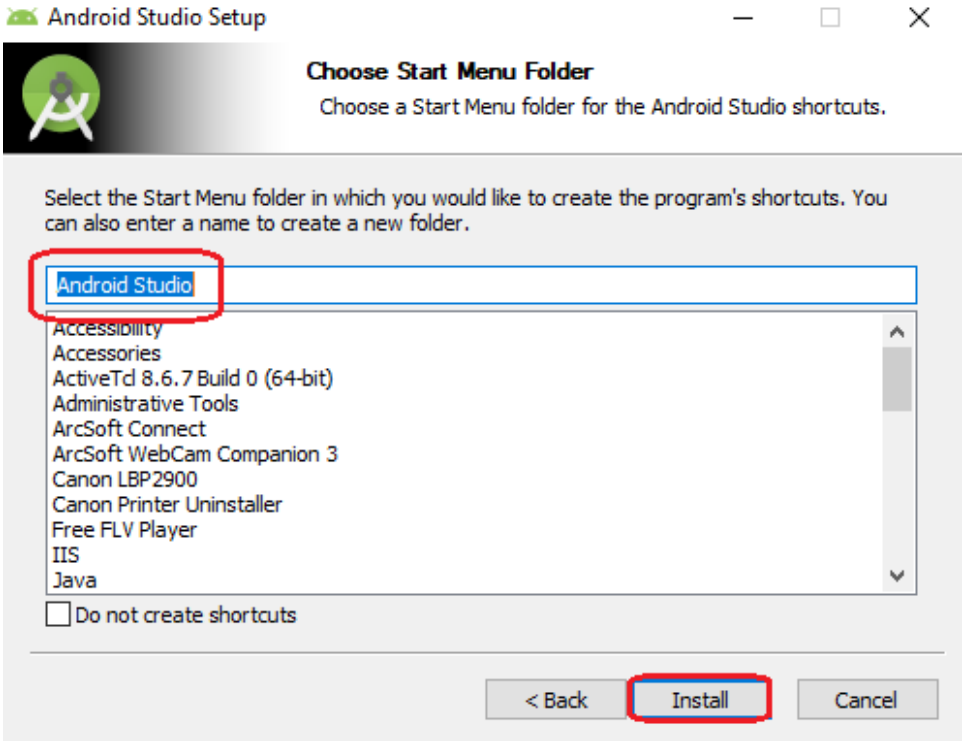
O‘rnatmoqchi bo‘lgan komponentlarni tanlang va "Keyingi" tugmasini bosing.



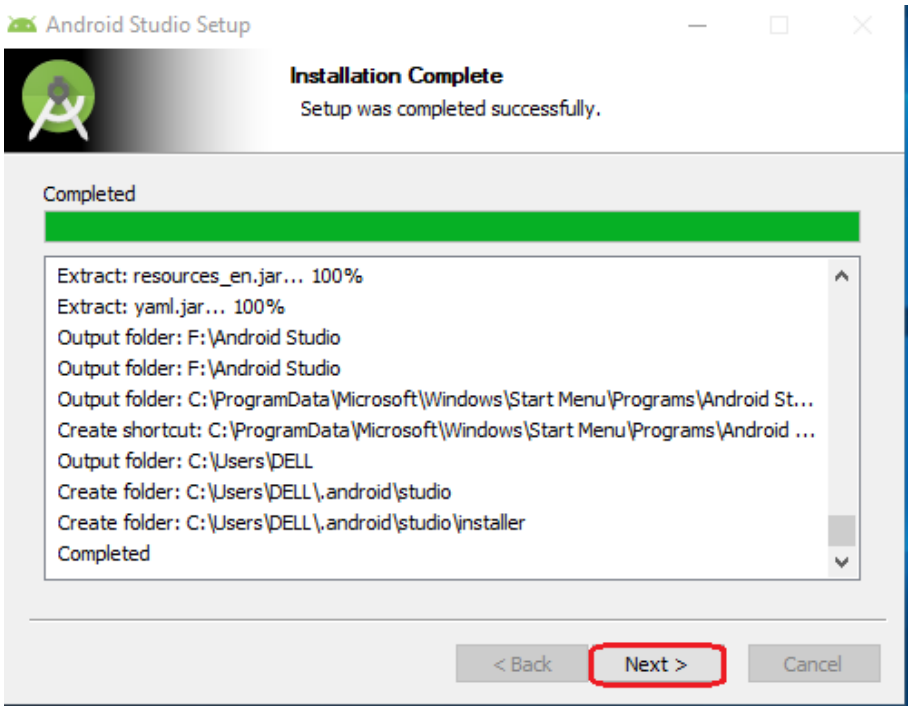
Endi, Android Studio-ni o‘rnatmoqchi bo‘lgan joyni belgilaymiz va davom etish uchun "Keyingi" tugmasini bosing.



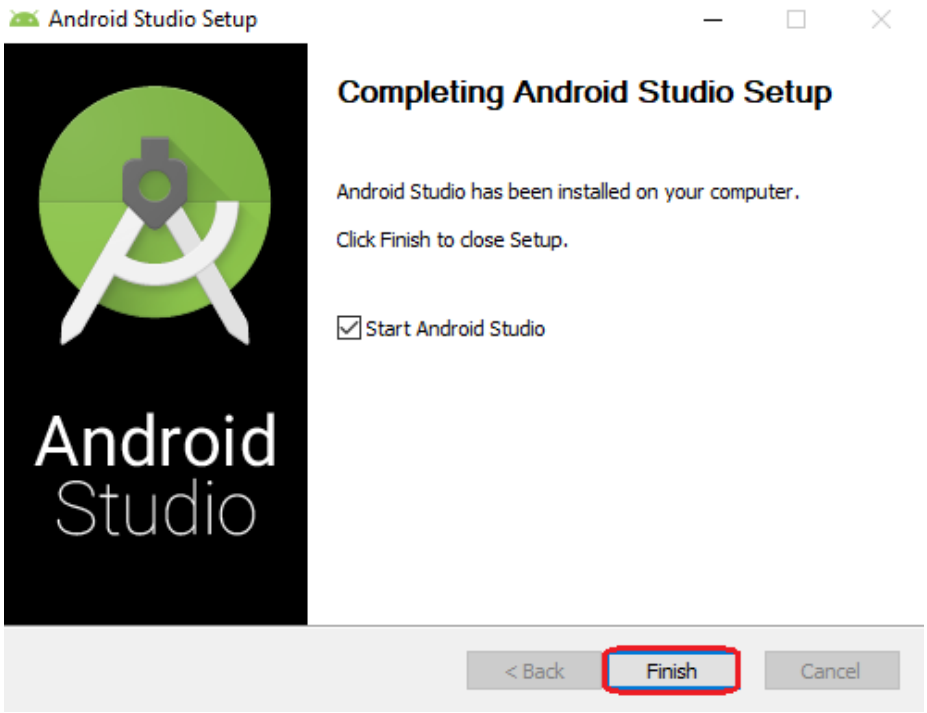
"Android Studio" yorlig'i uchun boshlash menyusi papkasini tanlang va davom etish uchun "O‘rnatish" tugmasini bosing.



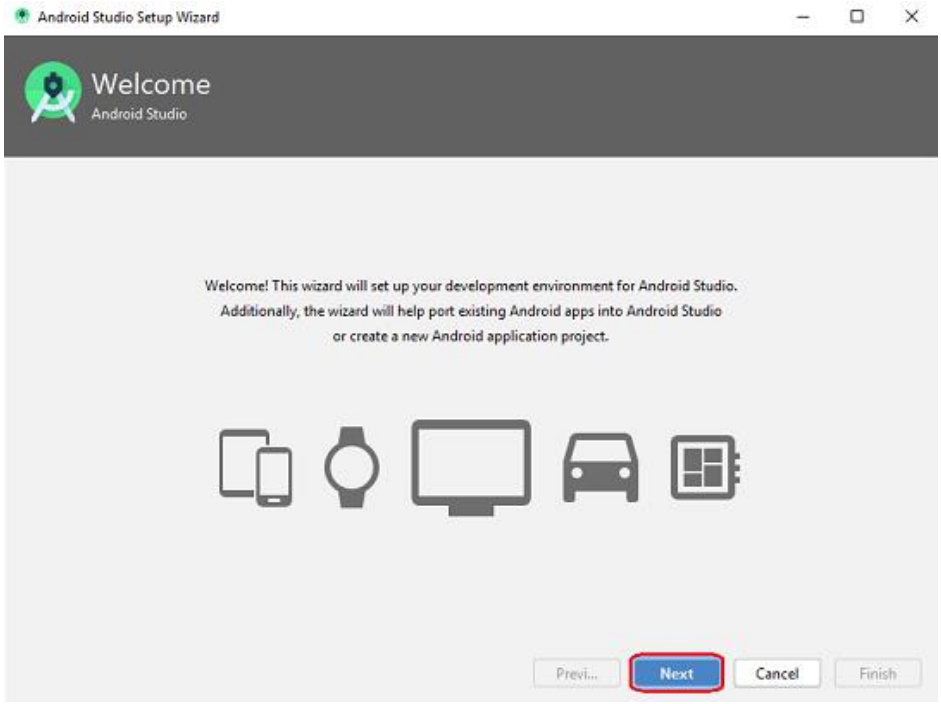
Android Studioni o‘rnatish muvaffaqiyatli tugagandan so‘ng, "Keyingi" tugmasini bosing.



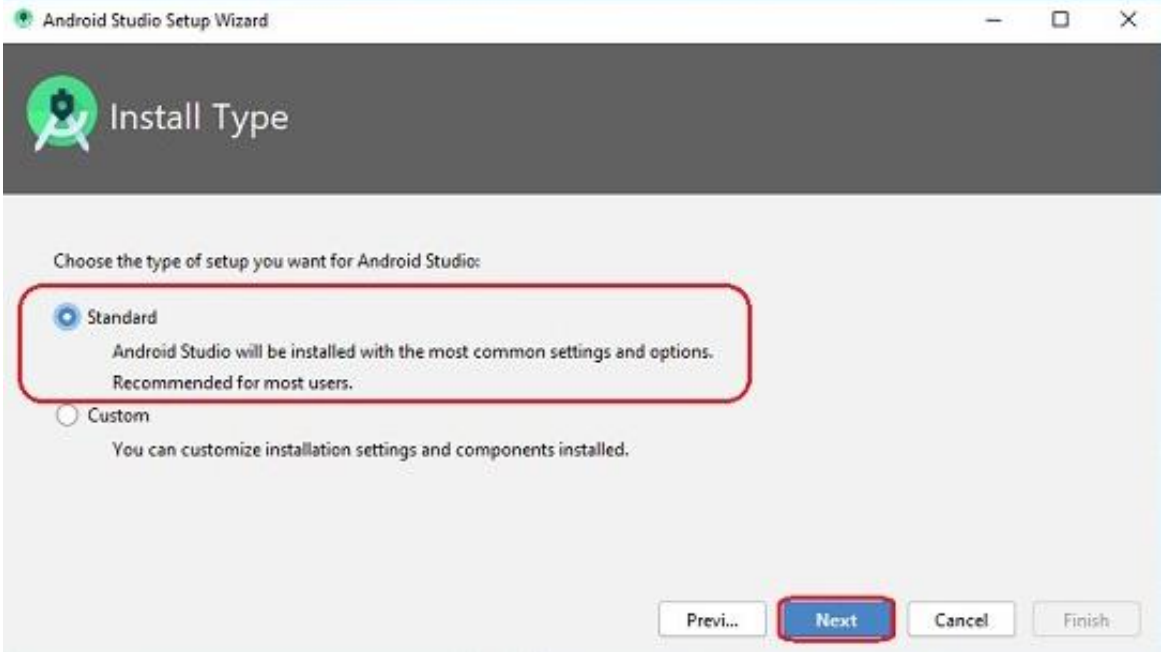
Davom etish uchun "Finish" tugmasini bosish orqali Android studioni ishga tushuramiz.



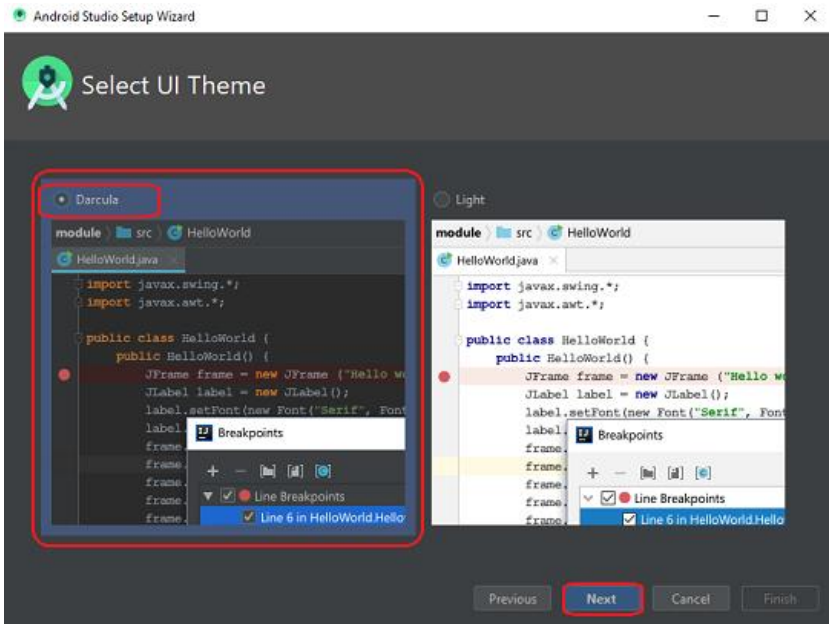
Android Studio o‘rnatish konfiguratsiyasi "Keyingi" tugmasini bosamiz.



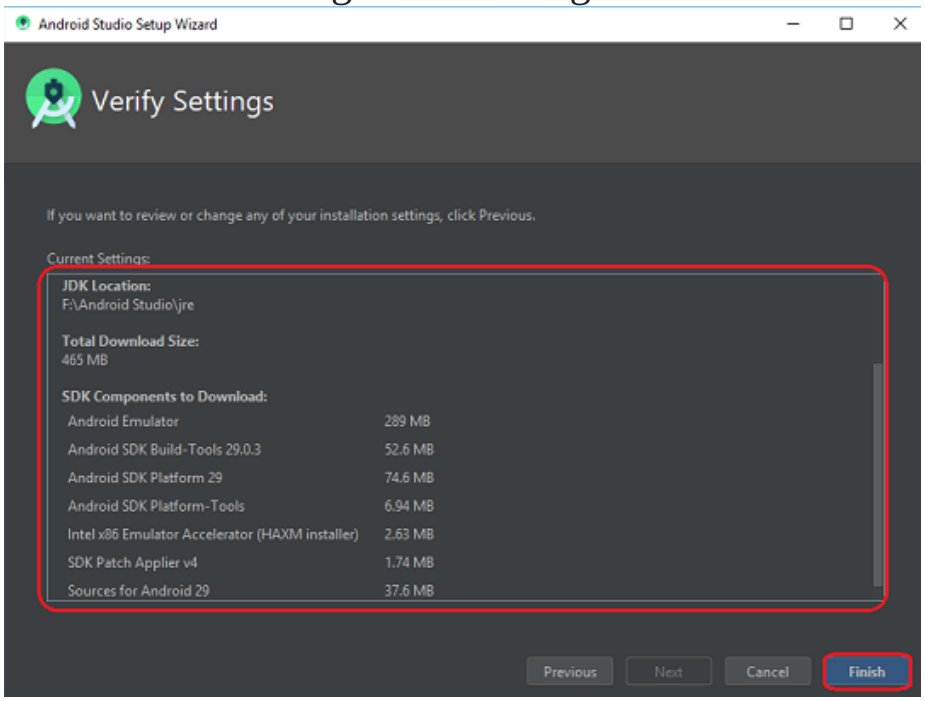
Agar siz yangi boshlovchi bo‘lsangiz va Android Studio haqida hech qanday tasavvurga ega bo‘lmasangiz, "Standart" variantini tanlang (tekshiring). U siz uchun eng keng tarqalgan sozlamalar va variantlarni o‘rnatadi. Davom etish uchun “Keyingi” tugmasini bosing.



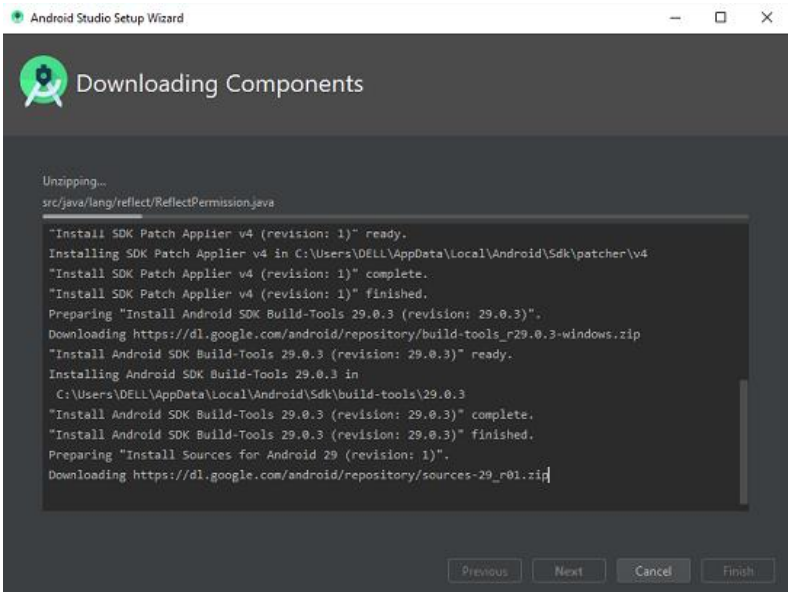
Endi siz foydalanuvchi interfeysi ko‘rinishini tanlashingiz. Keyin, "Keyingi" tugmasini bosing.



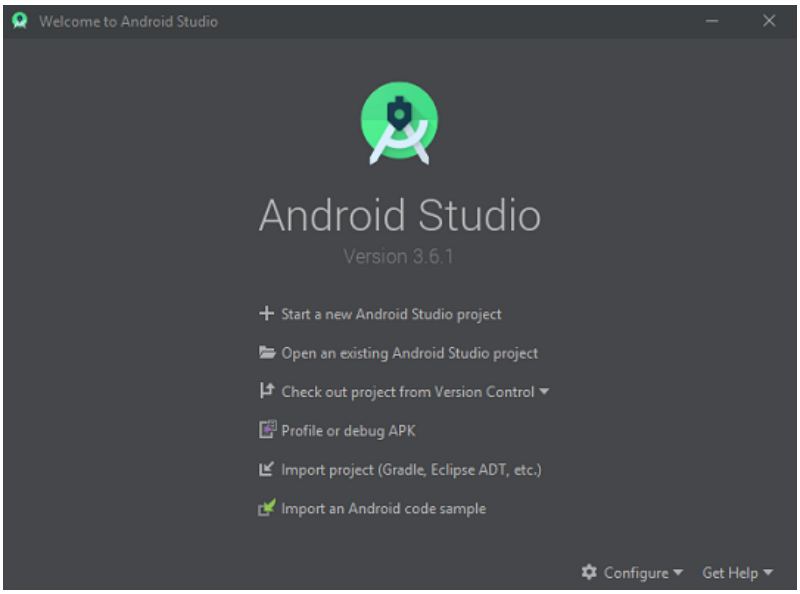
Endi, barcha SDK(software development kit) komponentlarini yuklab olish uchun "Finish" tugmasini bosing.



SDK dasturiy ta'minotni ishlab chiqish to‘plamini anglatadi SDK - bu ma'lum bir platforma uchun dasturiy ta'minot yaratish uchun vositalar to‘plami. Ushbu vositalar, shuningdek, dastur ishlab chiquvchisiga boshqa dastur, ya'ni Adjust kabi mobil o‘lchov hamkori (MMP) bilan integratsiyalasha oladigan ilova yaratish imkonini beradi. Mazkur komponentlarni yuklab olish va o‘rnatish jarayoni boshlanadi.



Barcha kerakli komponentlarni yuklab olgandan so‘ng, "Finish" tugmasini bosing. Android Studio tizimingizga muvaffaqiyatli o‘rnatiladi va siz yangi Android Studio loyihasini boshlashingiz mumkin.



Endigi navbatda Android OS uchun ishlab chiqiladigan dasturlarni virtual tarzda ishlatishimiz uchun Emulatorlardan faol tarzda foydalanamiz.

Android emulyatori ishlab chiquvchilarga ilovalarni sinovdan o‘tkazish uchun o‘z kompyuterlarida Android qurilmalarini simulyatsiya qilish imkonini beradi. Bu jismoniy uskunaga ehtiyoj sezmasdan Android ilovalarini ishga tushirish uchun virtual muhitni taqdim etadi. Ishlab chiquvchilar Android Studio‘da Android Virtual

Device (AVD) menejeri yordamida virtual qurilmalarni yaratishi, sozlashi va boshqarishi mumkin. Keyin ular o‘z ilovalarini to‘g'ridan-to‘g'ri ushbu virtual qurilmalarda joylashtirishlari va sinab ko‘rishlari mumkin.

Android Studiodagi emulyatorlar ilovalarni ishlab chiqish jarayoniga sezilarli yordam beradigan ko‘plab afzalliklarni taklif qiladi:

Qurilmaning xilma-xilligi: emulyatorlar ishlab chiquvchilarga turli xil ekran o‘lchamlari, ruxsati, apparat konfiguratsiyasi va Android versiyalari bilan Android qurilmalarining keng doirasini simulyatsiya qilish imkonini beradi. Bu xilma-xillik har bir jismoniy qurilmaga ega bo‘lmasdan, ilova turli qurilmalarda yaxshi ishlashini ta'minlash uchun to‘liq sinovdan o‘tkazish imkonini beradi.

Iqtisodiy samaradorlik: emulyatorlardan foydalanish qimmat bo‘lishi mumkin bo‘lgan sinov maqsadlarida bir nechta jismoniy qurilmalarni sotib olish zaruratini yo‘q qiladi. Ishlab chiquvchilar o‘zlarining ilovalarini taqlid qilingan qurilmalarda qo‘shimcha xarajatlarsiz sinab ko‘rishlari mumkin, bu esa apparat sotib olish va texnik xizmat ko‘rsatishning moliyaviy yukini kamaytiradi.

Rivojlanish tezligi: emulyatorlar Android Studio ichidan ilovalarni tezkor joylashtirish va sinovdan o‘tkazish orqali tezroq ishlab chiqish sikllarini osonlashtiradi. Ishlab chiquvchilar taqlid qilingan qurilmaga ta'sirini darhol kuzatib, kod o‘zgarishlari orqali tezda takrorlashlari mumkin. Bu nosozliklarni tuzatish va takomillashtirish jarayonini tezlashtiradi, bu esa yanada samarali rivojlanishga olib keladi.

Qulaylik: emulyatorlar yordamida ishlab chiquvchilar virtual Android qurilmalaridan istalgan vaqtda, istalgan joyda, agar ular o‘zlarining ishlab chiqish muhitiga kirish imkoniga ega bo‘lsalar, foydalanishlari mumkin. Ushbu qulaylik ishlab chiquvchilarga o‘z loyihalari ustida jismoniy qurilma mavjudligi yoki joylashuvi bilan cheklanmasdan ishlash imkonini beradi.

Nosozliklarni tuzatish va profil yaratish vositalari: emulyatorlar Android Studio‘ning disk raskadrovka va profillash vositalari to‘plami bilan muammosiz integratsiyalashgan bo‘lib, ilovalarning ishlashi, xotiradan foydalanish va xatti-harakatlari haqida to‘liq ma’lumot beradi. Ishlab chiquvchilar muammolarni samarali aniqlashlari va hal qilishlari, ilovalarning ishlashi va barqarorligini optimallashtirishlari mumkin.

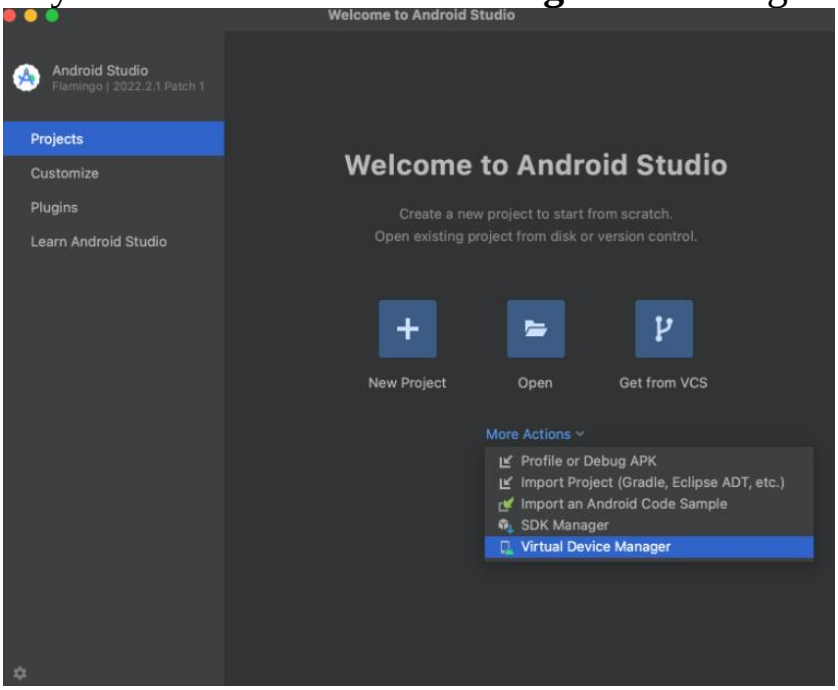
Moslashtiruv opsiyalari: Emulatorlar keng ko‘lamli moslashtirish imkoniyatlarini taklif etadi, bu esa ishlab chiquvchilarga taqlid qilingan qurilma xususiyatlarini real stsenariylarga moslashtirishga imkon beradi. Ishlab chiquvchilar ekran o‘lchami, operativ xotira, saqlash hajmi kabi parametrlarni sozlashi va turli tarmoq sharoitlarini simulyatsiya qilishlari mumkin, bu esa sinovning aniqligini oshiradi.

Moslik testi: Emulatorlar turli xil Android versiyalari va qurilma konfiguratsiyalari bo‘yicha to‘liq moslik sinovini osonlashtiradi. Ishlab chiquvchilar o‘z ilovalarining keng doiradagi qurilmalarda to‘g'ri ishlashini ta'minlashi mumkin, bu esa moslik bilan bog'liq muammolar xavfini kamaytiradi va foydalanuvchi qoniqishini oshiradi.

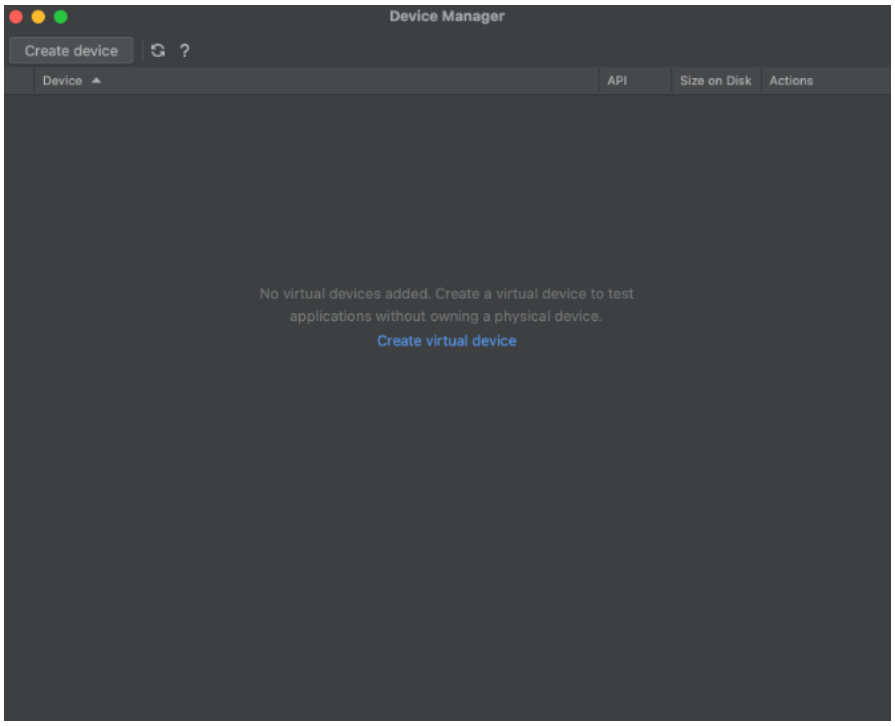
Rivojlanish muhiti bilan integratsiya: emulyatorlar Android Studio bilan muammosiz integratsiyalashib, yaxlit rivojlanish muhitini ta'minlaydi. Ishlab chiquvchilar emulyatorlarni to‘g'ridan-to‘g'ri Android Studio‘dan osongina ishga tushirishi, boshqarishi va boshqarishi mumkin, bu esa ish jarayonini soddalashtiradi.

Quyida Android studioda emulator o‘rnatish ketma-ketligini ko‘rib chiqamiz.

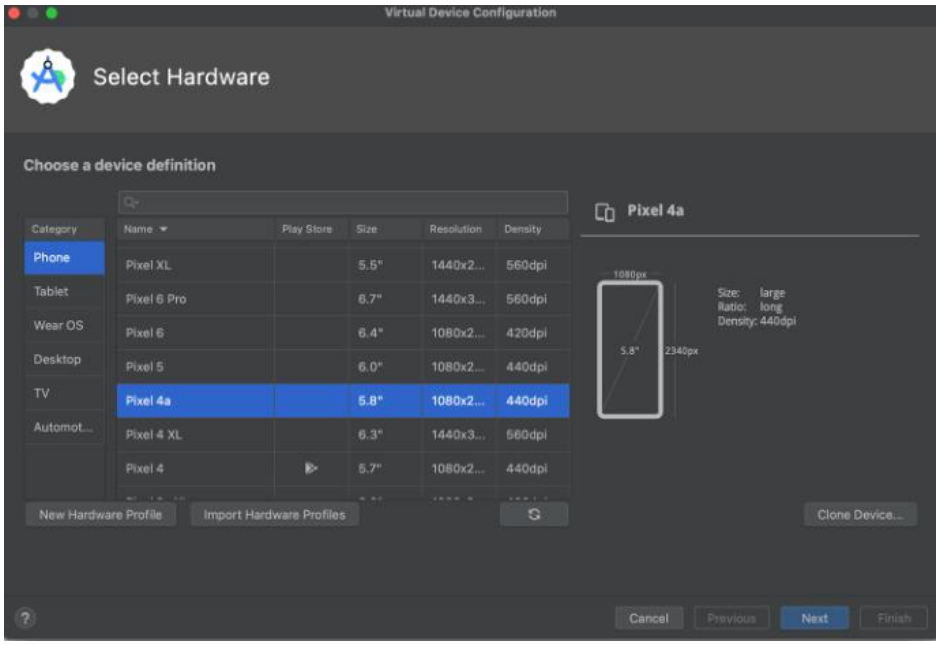
Android Studio asosiy ekranida "**More actions**" ni, so‘ngra ochiladigan menyuda "**Virtual Device Manager**" ni bosing.



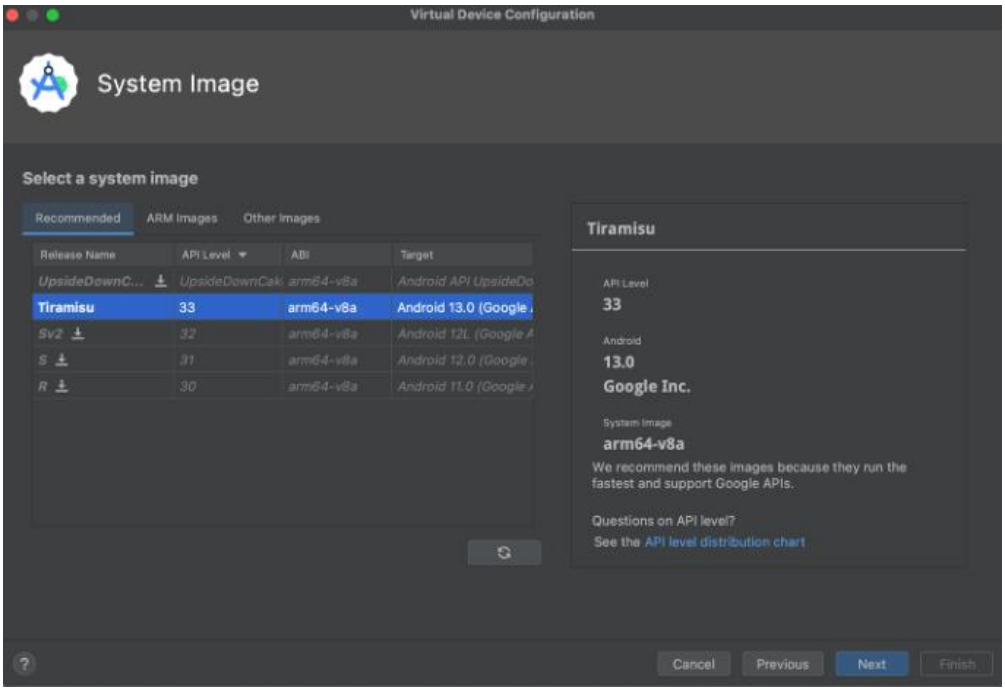
“**Create virtual device**” tugmasi bosilib yangi virtual qurilma yaratish bo‘limiga o‘tiladi.



Uskunani tanlashda, foydalanmoqchi bo‘lgan uskuna turi tanlanadi. Turli xil qurilmalarda sinovdan o‘tishni tavsiya etamiz, lekin qayerdan boshlashni bilmasangiz, Pixel liniyasidagi eng yangi qurilma yaxshi tanlov bo‘lishi mumkin.



Emulyatorga yuklash uchun OS versiyasini tanlang (ehtimol, Tavsiya etilgan yorlig'idagi tizim tasvirlaridan biri) va tasvirni yuklab olinadi.



O‘zingiz xohlagan boshqa sozlamalarni o‘zgartiring va virtual qurilma yaratish uchun Finish tugmasini bosing. Endi siz ushbu qurilmani istalgan vaqtda AVD Manager oynasidagi Play tugmasini bosish orqali ishga tushirishingiz mumkin.

Nazorat savollari.

1. Android studioni o‘rnatish tartibini tavsiflang.
2. Android studioni o‘rnatish uchun belgilangan texnik talablarni ayting.
3. Android emulyatori imkoniyatlarini tavsiflang.
4. Android Studiodagi emulyatorlar ilovalarni ishlab chiqishdagi afzalliklarini ko‘rsating.

3. JAVA DASTURLASH TILINING ASOSIY TUSHUNCHALARI. TILNING ASOSIY TASHKIL ETUVCHILARI. BERILGANLARNING ASOSIY VA PRIMITIVE TURLARI. OPERATORLAR. MASSIVLAR.

Reja.

3.1 Java dasturlash tilining asosiy tushunchalari.
3.2 O‘zgaruvchilar tiplari, ularni e’lon qilish.
3.3 Operatorlar. Massivlar bilan ishlash.

Bugungi kunda Java tili eng keng tarqalgan va ommabop dasturlash tillaridan biri hisoblanadi. Tilning birinchi versiyasi 1995 yilda Sun Microsystems kompaniyasi tomonidan yaratilgan va keyinchalik Oracle tomonidan o‘zlashtirilgan. Java har xil turdagi vazifalar uchun ishlatilishi mumkin bo‘lgan universal dasturlash tili bo‘lishi kerak edi. Va bugungi kunga qadar Java tili uzoq yo‘lni bosib o‘tdi, ko‘plab turli xil versiyalar nashr etildi. Joriy versiya Java 22 bo‘lib, u 2024 yil mart oyida chiqarilgan. Va Java shunchaki universal tildan bir qator vazifalar uchun ishlatiladigan turli texnologiyalarni birlashtirgan butun platforma va ekotizimga aylandi: ish stoli ilovalarini yaratishdan tortib yirik veb-portallar va xizmatlarni yozishgacha. Bundan tashqari, Java tili ko‘plab qurilmalar uchun dasturiy ta'minotni yaratishda faol foydalaniladi: oddiy shaxsiy kompyuterlar, planshetlar, smartfonlar va mobil telefonlar va hatto maishiy texnika. Ko‘pgina dasturlar Java-da yozilgan Android mobil operatsion tizimining mashhurligini eslash kifoya.

Java tilining asosiy xususiyati shundaki, uning kodi birinchi navbatda platformadan mustaqil maxsus bayt-kodga tarjima qilinadi. Va keyin bu bayt-kod JVM (Java Virtual Machine) tomonidan bajariladi. Shu nuqtai nazardan, Java PHP yoki Perl kabi standart talqin qilinadigan tillardan farq qiladi, ularning kodi tarjimon tomonidan darhol bajariladi. Shu bilan birga, Java C yoki C++ kabi sof kompilyatsiya qilingan til emas.

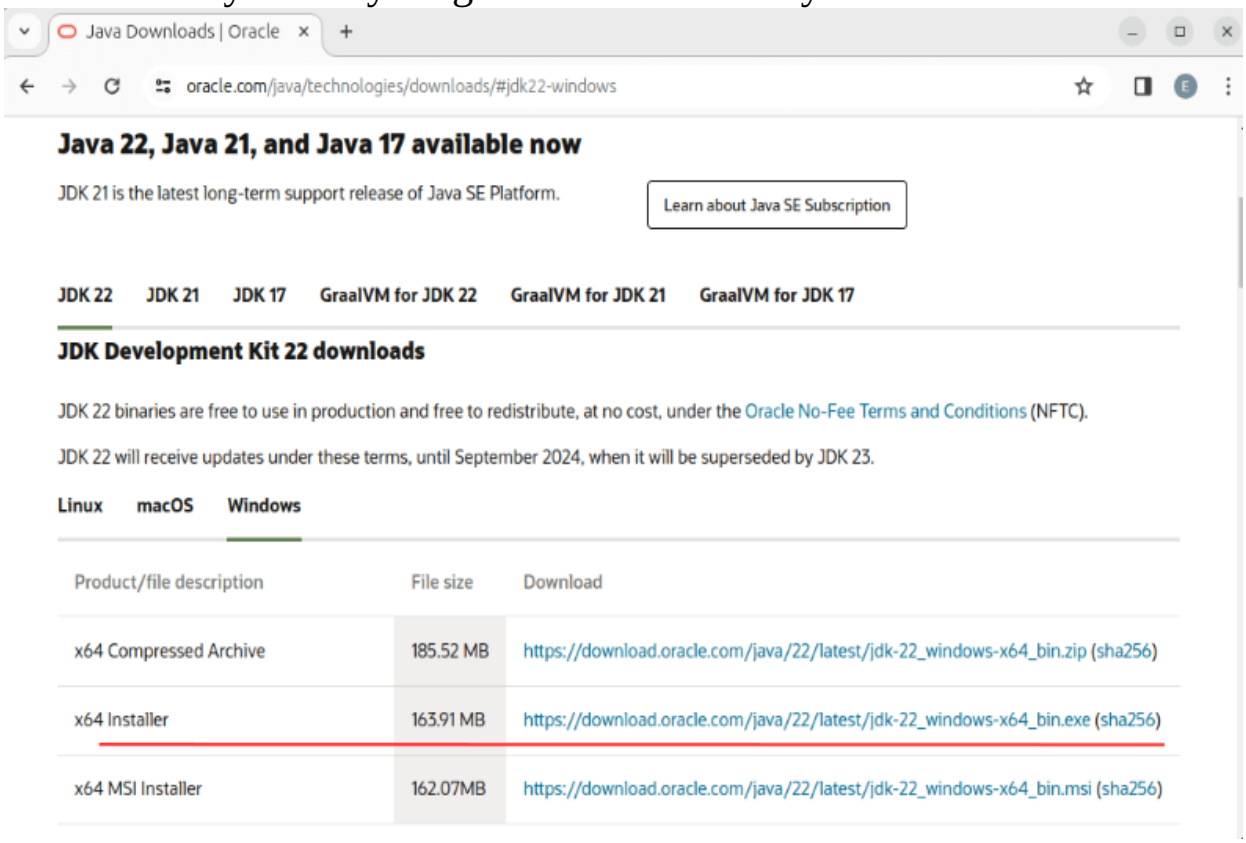
Java C-ga o‘xshash sintaksisga ega til bo‘lib, bu borada C/C++ va C# tillariga yaqin. Shuning uchun, agar siz ushbu tillardan biri bilan tanish bo‘lsangiz, Java-ni o‘zlashtirish osonroq bo‘ladi.

Java - bu ob'ektga yo‘naltirilgan til. U polimorfizm, meros, statik yozishni qo‘llab-quvvatlaydi. Ob'ektga yo‘naltirilgan yondashuv katta

lekin ayni paytda moslashuvchan, kengaytiriladigan ilovalarni qurish muammolarini hal qilish imkonini beradi.

Java-da dasturlarni ishlab chiqish uchun bizga JDK (Java Development Kit) maxsus ishlab chiqish to‘plami kerak bo‘ladi. JDK Java dasturlarini kompilyatsiya qilish va ishga tushirish, shuningdek, bir qator boshqa funktsiyalarni bajarish imkonini beruvchi bir qator dasturlar va yordamchi dasturlarni o‘z ichiga oladi.

Windows-da JDK-ni o‘rnatishning eng keng tarqalgan varianti Oracle rasmiy veb-saytidagi o‘rnatuvchidan foydalanishdir.



O‘zgaruvchilar va doimiylar

O‘zgaruvchilar dasturda ma'lumotlarni saqlash uchun ishlatiladi. O‘zgaruvchi ma'lum bir turdagi qiymatni saqlaydigan nomlangan xotira maydonini ifodalaydi. Har bir o‘zgaruvchining turi, nomi va qiymati bor. Tur o‘zgaruvchi qanday ma'lumotlarni saqlashi mumkinligini yoki haqiqiy qiymatlar oralig'ini belgilaydi.

Masalan, x deb ataladigan o‘zgaruvchini va uning turi bo‘lgan int aniqlaymiz:

```
int x;
```

Bu ifodada biz int tipidagi x o‘zgaruvchini e'lon qilamiz. Ya'ni, x 4 baytdan ko‘p bo‘lmagan ma'lum sonni saqlaydi. O‘zgaruvchi nomi quyidagi talablarga javob beradigan har qanday ixtiyoriy nom bo‘lishi mumkin:

- ism har qanday harf-raqamli belgilarni, shuningdek pastki chiziqni o‘z ichiga olishi mumkin va nomdagi birinchi belgi raqam bo‘lmasligi kerak
- ismda tinish belgilari yoki bo‘sh joy bo‘lmasligi kerak
- nom Java tili kalit so‘zi bo‘lishi mumkin emas

Bunga qo‘shimcha ravishda, uni e'lon qilish va undan keyin foydalanishda, Java katta-kichik harflarga sezgir til ekanligini hisobga olish kerak, shuning uchun quyidagi deklaratsiyalar ikki xil o‘zgaruvchini ifodalaydi int num;.int NUM;

O‘zgaruvchini e'lon qilgandan so‘ng, unga qiymat berishimiz

```
int x;                    //o’zgaruvchini e’lon qilish
x = 10;                   //o’zgaruvchiga qiymat biriktirish
System.out.println(x);    // 10
```

mumkin:

o‘zgaruvchini e'lon qilganingizda unga qiymat ham belgilashingiz mumkin.

Ushbu jarayon ishga tushirish deb ataladi :

```
int x = 10;-- //o’zgaruvchini e’lon qilish va qiymat biriktirish.
System.out.println(x);--// 10
```

Bir vaqtning o‘zida vergul bilan ajratilgan bir xil turdagi bir nechta o‘zgaruvchilarni e'lon qilishingiz mumkin:

```
int x, y;
x = 10;
y = 20;
System.out.println(x); // 10
System.out.println(y); // 20
```

O‘zgaruvchilarning o‘ziga xos xususiyati shundaki, biz dastur ishlayotgan vaqtda ularning qiymatini o‘zgartirishimiz mumkin:

```
        int x = 10;
        System.out.println(x);
// 10

        x = 25;
        System.out.println(y);
// 25
```

Konstantalar

O‘zgaruvchilarga qo‘shimcha ravishda Java-da ma'lumotlarni saqlash uchun konstantalardan foydalanish mumkin. O‘zgaruvchilardan farqli o‘laroq, konstantalarga faqat bir marta qiymat berish mumkin. Konstanta o‘zgaruvchiga o‘xshab e'lon qilinadi, faqat boshida finish kalit so‘z bilan:

```
1.  final int LIMIT = 10;
2.  System.out.println(LIMIT);      // 10
```

Odatda, konstantalar katta harflarga ega.

Konstantalar endi o‘zgarmasligi kerak bo‘lgan o‘zgaruvchilarni o‘rnatishga imkon beradi. Misol uchun, agar bizda pi sonini saqlash uchun o‘zgaruvchi bo‘lsa, uning qiymati doimiy bo‘lgani uchun uni doimiy deb e'lon qilishimiz mumkin.

Java-dagi operatsiyalarning aksariyati boshqa C-ga o‘xshash tillarda qo‘llaniladigan operatsiyalarga o‘xshaydi. Unar amallar (bitta operandda bajariladi), ikkita operandda ikkilik amallar va uchta operandda uchlik operatsiyalar mavjud. Operand - bu operatsiyada ishtirok etadigan o‘zgaruvchi yoki qiymat (masalan, raqam). Keling, barcha turdagi operatsiyalarni ko‘rib chiqaylik.

Arifmetik amallar raqamlarni o‘z ichiga oladi. Java ikkilik arifmetik amallarga (ikki operandda bajariladi) va unar arifmetik amallarga (bitta operandda bajariladi) ega. Ikkilik operatsiyalarga quyidagilar kiradi:

Ikki sonni qo‘shish amali.

```
int a = 10;
int b = 25;
int c = a + b;      // 35
int d = c + 11;    // 46
```

Ikki sonni ayirish amali.

```
int a = 10;
int b = 5;
int c = a - b;      // 5
int d = 11 - c;    // 6
```

```
int a = 5;
int b = 7;
int c = a * b;     // 35
int d = 3 * a;    // 15
```

Ikki sonni ko‘paytirish amali.

Ikki sonni bo‘lish amali.

Bo‘lishda shuni hisobga olish kerakki, agar operatsiya ikkita butun sonni o‘z ichiga olsa, natija float yoki double o‘zgaruvchiga tayinlangan bo‘lsa ham, bo‘linish natijasi butun songa yaxlitlanadi:

```
int a = 28;
int b = 7;
int c = a / b;    // 4
double d = 22.5 / 4.5;    // 5.0
```

```
double k = 10 / 4;    // 2
System.out.println(k);   // 10
```

Ikki sonni bo‘lishda qoldiqni olish.

```
int a = 33;
int b = 7;
int c = a % b;    // 5
```

```
int d = 22 % 4;    // 2  (22 – 4*5 = 2)
```

++ (prefiks ortirish) o‘zgaruvchini bittaga oshirishni o‘z ichiga oladi, masalan z=++y (avval y o‘zgaruvchining qiymati 1 ga oshiriladi, so‘ngra uning qiymati z o‘zgaruvchisiga beriladi)

```
int a = 8;
int b = ++a;
System.out.println(a);   // 9
System.out.println(b);   // 9
```

++ (postfiks ortirish) o‘zgaruvchining bittaga ko‘payishini ifodalaydi, masalan, z=y++ (avval y o‘zgaruvchining qiymati z o‘zgaruvchisiga tayinlanadi, so‘ngra y o‘zgaruvchining qiymati 1 ga oshiriladi)

```
int a = 8;
int b = a++;
System.out.println(a);   // 9
System.out.println(b);   // 8
```

Arifmetik amallarning ustuvorligi

Ba'zi operatsiyalar boshqalarga qaraganda ko‘proq ustuvorlikka ega va shuning uchun birinchi navbatda amalga oshiriladi.
Ustuvorlikni kamaytirish tartibida operatsiyalar:

++ (postfiksning o‘sishi), -- (postfiksning kamayishi)
++ (prefiks ortishi), -- (prefiksning kamayishi)
* (ko‘paytirish), / (bo‘lish), % (bo‘lish qoldig'i)
+ (qo‘shish), - (ayirish)

Arifmetik ifodalar to‘plamini bajarishda operatsiyalarning ustuvorligini hisobga olish kerak:

```
int a = 8;
int b = 7;
int c = a + 5 * ++b;
System.out.println(c);   // 48
```

```
int a = 8;
int b = 7;
int c = (a + 5) * ++b;
System.out.println(c);   // 104
```

Shartli ifodalar

Shartli ifodalar shartni ifodalaydi va mantiqiy qiymatni qaytaradi, ya'ni true (shart rost bo‘lsa) yoki false (agar shart noto‘g'ri bo‘lsa). Shartli ifodalarga taqqoslash operatorlari va mantiqiy operatorlar kiradi.

Taqqoslash operatsiyalari

Taqqoslash operatsiyalari ikkita operandni taqqoslaydi va boolean turdagi true qiymatni qaytaradi agar ifoda to‘g'ri bo‘lsa va ifoda noto‘g'ri bo‘lsa false qiymatni qaytaradi.

== tenglik uchun ikkita operandni taqqoslaydi va true (operandlar teng bo‘lsa) va false (operandlar teng bo‘lmasa) qaytaradi.

```
int a = 10;
int b = 4;
boolean c = a == b;     // false
boolean d = a == 10;     // true
```

!= ikkita operandni solishtiradi va true agar operandlar TENG bo‘lmasa va false agar operandlar teng bo‘lsa.

```
int a = 10;
int b = 4;
boolean c = a != b;     // true
boolean d = a != 10;     // false
```

< belgi true qaytaradi agar birinchi operand ikkinchisidan kichik bo‘lsa, aks holda false qaytaradi.

```
int a = 10;
int b = 4;
boolean c = a < b;     // false
```

> belgi true qaytaradi agar birinchi operand ikkinchisidan katta bo‘lsa, aks holda false qaytaradi.

```
int a = 10;
int b = 4;
boolean c = a > b;     // true
```

Mantiqiy operatsiyalar

Java-da mantiqiy operatorlar ham mavjud bo‘lib, ular ham shartni ifodalaydi va rost yoki yolg'onni qaytaradi va odatda bir nechta taqqoslash operatorlarini birlashtiradi. Mantiqiy operatsiyalarga quyidagilar kiradi:

|| belgi true qaytaradi agar hech bo‘lmaganda ikki operanddan bittasi rost bo‘lsa, false qaytaradi qachonki ikkita operand ham yolg’on bo‘lsa.

```
int a = 10;
int b = 4;
boolean c = a>6 || b<3;     // true
boolean c = a<6 || b<3;     // false
```

&& belgi true qaytaradi agar ikki operand ham rost bo‘lsa, false qaytaradi qachonki ikkita operanddan bittasi yoki ikkitasi ham yolg’on bo‘lsa.

```
int a = 10;
int b = 4;
boolean c = a>6 && b<3;     // false
boolean c = a<6 && b<3;     // false
boolean c = a>6 && b>3;     // true
```

Shart operatorlari

Ko‘pgina dasturlash tillarining asosiy elementlaridan biri bu shart operatoridir. Ushbu dizaynlar dastur ishini muayyan shartlarga qarab yo‘naltirishga imkon beradi. Java tili quyidagi shartlardan foydalanadi: if, else va switch case

If/else konstruktsiyasi

If/else bayonoti ma'lum bir shartning rostligini tekshiradi va test natijalariga qarab, ma'lum kodni bajaradi:

```
int a = 10;
int b = 4;
if(a > b){
System.out.println(“a soni b sonidan katta”);   //
```

```
}
```

If kalit so‘zdan keyin shart mavjud. Va agar bu shart bajarilsa, jingalak qavslardan keyin if blokiga qo‘shimcha joylashtirilgan kod ishga tushiriladi. Shartlar ikki raqamni solishtirish operatsiyasi. Agar shart bajarilmasa, bunday holda biz else: blok qo‘shishimiz mumkin.

```
int a = 10;
int b = 4;
if(a > b){
System.out.println(“a soni b sonidan katta”);   //
} else {
System.out.println(“a soni b sonidan katta emas”);   //
}
```

Ammo raqamlarni solishtirganda, biz uchta holatni sanashimiz mumkin: birinchi raqam ikkinchidan katta, birinchi raqam ikkinchidan kichik va raqamlar teng. Ifodasi yordamida biz else if qo‘shimcha shartlarni bajarishimiz mumkin:

```
int a = 10;
int b = 4;
if(a > b){
    System.out.println(“a soni b sonidan katta”);   //
} else if (a < b){
    System.out.println(“a soni b sonidan katta emas”);   //
} else {
    System.out.println(“a soni b sonlari teng”);   //
}
```

Tanlash shart operatori.

Shart tanlash operatori if/else konstruktsiyasiga o‘xshaydi, chunki u bir vaqtning o‘zida bir nechta shartlarni qayta ishlashga imkon beradi:

```
int num = 8;
switch(num){

    case 1:
        System.out.println("Son 1 ga teng");
        break;
    case 8:
        System.out.println("Son 8 ga teng ");
        num++;
```

```
            break;
        case 9:
            System.out.println("Son 9 ga teng ");
            break;
        default:
            System.out.println("Son 1,8,9 ga teng emas ");

    }
```

Switch kalit so‘zidan keyin taqqoslanadigan ifoda qavs ichida keladi. Ushbu ifodaning qiymati ketma-ketlik bilan case iboralaridan keyin joylashtirilgan qiymatlar bilan taqqoslanadi. Va agar moslik topilsa, tegishli ish bloki bajariladi.

Boshqa bloklarni bajarmaslik uchun ish blokining oxirida break operatori joylashtiriladi.

Takrorlash operatorlari.

Takrorlash operatorlari muayyan shartlarga qarab ma'lum bir amalni bir necha marta bajarishga imkon beradi. Java-da quyidagi turdagi takrorlash operatorlari mavjud:

- for
- while
- do while

For takrorlash operatori uchun standartni ko‘rib chiqamiz:

```
for (int i = 1; i < 9; i++){
        System.out.printf("takrorlash raqami:
", i);
    }
```

For deklaratsiyasining birinchi qismi int i = 1 hisoblagich i ni yaratadi va ishga tushiradi. Bu har qanday boshqa raqamli tur bo‘lishi mumkin, masalan, float. For bajarilishidan oldin hisoblagich qiymati 1 ga teng bo‘ladi. Bu holda, bu o‘zgaruvchini e'lon qilish bilan bir xil.

Ikkinchi qism - siklning bajarilishi sharti. Bunday holda, tsikl i 9 ga yetguncha ishlaydi.

Uchinchi qism esa hisoblagichni bir marta oshirmoqda. Shunga qaramay, biz bittaga ko‘paytirishimiz shart emas. Kamaytirish uchun i—dan foydalanishimiz mumkin:

Natijada, sikl bloki i qiymati 9 ga teng bo‘lguncha 8 marta ishlaydi. Va har safar bu qiymat 1 ga ortadi.

while takrorlash operatori.

while darhol shartning rostligini tekshiradi va agar shart rost bo‘lsa, sikl kodi bajariladi:

```
int j = 6;
while (j > 0) {
    System.out.println(j);
    j--;
}
```

Do while sikli avval sikl kodini bajaradi va keyin while operatoridagi shartni tekshiradi. Va bu shart to‘g'ri bo‘lsa, sikl takrorlanadi.

```
int j = 7;
do {
    System.out.println(j);
    j--;
} while  (j > 0);
```

Bunday holda, sikl kodi j nolga teng bo‘lguncha 7 marta ishlaydi. Shuni ta'kidlash kerakki, do tsikli while ifodasidagi shart to‘g'ri bo‘lmasa ham, harakat kamida bir marta bajarilishini kafolatlaydi.

continue va break bayonotlari

Break operatori sikl o‘z ishini tugatmagan bo‘lsa ham, istalgan vaqtda sikldan chiqishga imkon beradi:

```
for (int i = 0; i < 10; i++){
    if (i == 5)
        break;
    System.out.println(i);
}
```

Hisoblagich 5 ga yetganda, break operatori ishga tushadi va tsikl tugaydi.

Endi ishonch hosil qilaylik, agar raqam 5 bo‘lsa, sikl tugamaydi, shunchaki keyingi iteratsiyaga o‘tadi. Buning uchun continue operatoridan foydalanamiz:

```
for (int i = 0; i < 10; i++){
    if (i == 5)
```

```
            continue;
        System.out.println(i);
    }
```

Bunday holda, 5 raqamiga yetganda, dastur shunchaki bu raqamni o‘tkazib yuboradi va keyingisiga o‘tadi.

Massivlar

Massiv bir xil turdagi qiymatlar to‘plamini ifodalaydi. Massivni e'lon qilish bitta qiymatni saqlaydigan oddiy o‘zgaruvchini e'lon qilishga o‘xshaydi va massivni e'lon qilishning ikki yo‘li mavjud:

```
int nums[];
int[] nums2;
```

Massiv quyidagi konstruksiya yordamida yaratiladi: new malumot tipi[elementlar soni], bu yerda new — qavs ichida ko‘rsatilgan elementlar soni uchun xotirani ajratuvchi kalit so‘z. Masalan, nums = new int[4];- bu ifoda to‘rtta int elementli massivni yaratadi va har bir element 0 raqamining standart qiymatiga ega bo‘ladi.

massivni yaratishda uning elementlari uchun maxsus qiymatlarni ham o‘rnatishingiz mumkin:

```
int [] nums = new int[] { 1, 2, 3, 5 };
int[]nums2 = { 1, 2, 3, 5 };
```

Shuni ta'kidlash kerakki, bu holda kvadrat qavslar massivning o‘lchamini ko‘rsatmaydi, chunki u jingalak qavslardagi elementlarning soni bilan hisoblanadi.

Massivni yaratgandan so‘ng, biz uning istalgan elementiga indeks bo‘yicha kirishimiz mumkin, bu esa massiv o‘zgaruvchisi nomidan keyin kvadrat qavs ichida beriladi:

```
int [] nums = new int[4];
nums[0] = 1;
nums[1] = 2;
nums[2] = 4;
nums[3] = 100;

// massiv 3 elementini ekranga chiqaramiz.
System.out.println(nums[2]);    // 4
```

Massiv elementlarini indekslash 0 dan boshlanadi, shuning uchun bu holda massivdagi to‘rtinchi elementga kirish uchun biz nums[3]

ifodadan foydalanishimiz kerak. Va bizning massivimiz faqat 4 ta element uchun aniqlanganligi sababli, biz, masalan, oltinchi elementga kira olmaymiz: nums[5] = 5;. Agar buni qilishga harakat qilsak, xatoga duch kelamiz.

Massiv uzunligi

Massivlarga ega bo‘lgan eng muhim xususiyat bu uzunlik xususiyati bo‘lib , u massiv uzunligini, ya'ni uning elementlari sonini qaytaradi:

```
int [] nums = new int[4];
nums[0] = 1;
nums[1] = 2;
nums[2] = 4;
nums[3] = 100;
System.out.println(nums.length);     // 4
```

Ko‘p o‘lchovli massivlar

Bir o‘lchovli massivlardan tashqari ko‘p o‘lchovli massivlar ham mavjud. Eng mashhur ko‘p o‘lchovli massiv ikki o‘lchovli massivni ifodalovchi jadvaldir:

```
int [] nums = new int[4] { 0, 1, 2, 3, 4, 5 };
int [][] nums2 = { { 0, 1, 2 }, { 3, 4, 5 } };
```

Vizual ravishda ikkala massivni quyidagicha ko‘rsatish mumkin:

Bir o‘lchovli massiv nums

0	1	2	3	4	5

Ikki o‘lchovli massiv nums 2

0	1	2
3	4	5

nums2 massivi ikki o‘lchovli bo‘lgani uchun u oddiy jadvaldir. U quyidagi tarzda ham yaratilishi mumkin: int[][] nums2 = new int[2][3];. Kvadrat qavslar soni massiv hajmini bildiradi. Qavslar ichidagi raqamlar esa satr va ustunlar sonini bildiradi. Shuningdek, indekslardan

foydalanib, biz dasturda massiv elementlaridan foydalanishimiz mumkin:

Shu bilan birga, for takrorlash operatori ushbu versiyasi ga nisbatan ancha moslashuvchan for (int i : array). Xususan, ushbu versiyada biz elementlarni o‘zgartirishimiz mumkin:

```
int[] array = new int[] { 1, 2, 3, 4, 5 };
for (int i=0; i<array.length;i++){
    array[i] = array[i] * 2;
    System.out.println(array[i]);
}
```

Ko‘p o‘lchovli massivlar bilan takrorlash operatorlari.

```
int[][] nums = new int[][]
{
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
for (int i = 0; i < nums.length; i++){
    for(int j=0; j < nums[i].length; j++){

        System.out.printf("%d ", nums[i][j]);
    }
    System.out.println();
}
```

Birinchidan, qatorlar bo‘ylab takrorlash uchun sikl yaratiladi, so‘ngra birinchi sikl ichida ma'lum bir qator ustunlari ustida takrorlash uchun ichki sikl yaratiladi. Xuddi shunday, siz ko‘p sonli o‘lchamli uch o‘lchovli massivlar va to‘plamlarni takrorlashingiz mumkin.

Nazorat savollari.

1. Java dasturlash tili haqida biroz ma’lumot bering.
2. Java tilining asosiy xususiyati nimadan iborat?
3. O‘zgaruvchilar va doimiylar haqida qisqacha ma’lumot bering.
4. Konstantalar haqida ma’lumot bering.
5. Shartli ifodalardagi opertaorlarni tushuntiring.
6. Shart operatorlani izohlang.
7. Massivlar haqida ma’lumot bering.

4. SINFLAR. METODLAR. SINFNING STATIC A'ZOLARI. KONSTRUKTOR. THIS KALIT SO'ZI.

Reja:

4.1 Java dasturlash tilida OYD (obyektga yo'naltirilgan dasturlash) sinflar, metodlar bilan ishlash.

4.2 Sinfning statik a'zolari. Foydalanuvchi sinfi. Konstruktor. This kalit so'zining qo'llanilishi.

Java ob'ektga yo'naltirilgan tildir, shuning uchun "sinf" va "ob'ekt" kabi tushunchalar unda asosiy rol o'ynaydi. Har qanday Java dasturini bir-biri bilan o'zaro ta'sir qiluvchi ob'ektlar to'plami sifatida ko'rsatish mumkin.

Ob'ektning shabloni yoki tavsifi sinfdir va ob'ekt shu sinfning namunasini ifodalaydi. Quyidagi o'xshashlikni ham chizishingiz mumkin. Har birimiz inson haqida qandaydir tasavvurga egamiz - ikkita qo'l, ikki oyoq, bosh, tana va boshqalar. Muayyan shablon mavjud - bu shablonni sinf deb atash mumkin. Haqiqatan ham mavjud bo'lgan shaxs (aslida berilgan sinfning namunasi) bu sinfning ob'ektidir.

Sinf **class** kalit so'zi yordamida aniqlanadi:

```
class Person{
}
```

Bu holda sinf Shaxs deb ataladi. Sinf nomidan keyin jingalak qavslar mavjud bo'lib, ular orasiga sinf tanasi - ya'ni uning maydonlari va usullari joylashtiriladi.

Har qanday ob'ekt ikkita asosiy xususiyatga ega bo'lishi mumkin: holat - ob'ekt saqlaydigan ba'zi ma'lumotlar va xatti-harakatlar - ob'ekt bajarishi mumkin bo'lgan harakatlar.

Ob'ekt holatini sinfda saqlash uchun maydonlar yoki sinf o'zgaruvchilari ishlatiladi. Usullar sinfdagi ob'ektning harakatini aniqlash uchun ishlatiladi. Masalan, shaxsni ifodalovchi Person sinfi quyidagi ta'rifga ega bo'lishi mumkin:

```
class Person{
    String name;          // nomi
    int age;              // yoshi
    void displayInfo(){
```

```
            System.out.printf("Nomi: %s \tAge: %d\n", name,
age);
        }
    }
```

Person klassi ikkita maydonni belgilaydi: ism shaxsning ismini, yoshi esa odamning yoshini bildiradi. Va displayInfo usuli ham aniqlangan, u hech narsani qaytarmaydi va bu ma'lumotlarni konsolga chiqaradi.

Endi biz ushbu sinfdan foydalanamiz. Buning uchun biz quyidagi dasturni belgilaymiz:

```
    public class Program{

        public static void main(String[] args) {

            Person tom;
        }
    }
    class Person{
        String name;    // nomi
        int age;        // yoshi
        void displayInfo(){
            System.out.printf("Nomi: %s \tAge: %d\n", name,
age);
        }
    }
```

Odatda sinflar turli fayllarda aniqlanadi. Bunday holda, soddalik uchun biz bitta faylda ikkita sinfni aniqlaymiz. Shuni ta'kidlash joizki, bu holda faqat bitta sinf umumiy modifikatorga ega bo‘lishi mumkin (bu holda, Program sinfi) va kod faylining o‘zi ushbu sinf nomi bilan nomlanishi kerak, ya'ni bu holda fayl Dastur.java deb nomlanishi kerak.

Sinf yangi turni ifodalaydi, shuning uchun biz ushbu turni ifodalovchi o‘zgaruvchilarni aniqlashimiz mumkin. Shunday qilib, bu erda asosiy usulda tomPerson sinfini ifodalovchi o‘zgaruvchi aniqlanadi. Ammo hozircha bu o‘zgaruvchi hech qanday ob'ektga ishora qilmaydi va sukut bo‘yicha u null qiymatiga ega. Umuman olganda, biz uni hali ishlata olmaymiz, shuning uchun avval Person sinfining ob'ektini yaratishimiz kerak.

Konstruktorlar

Oddiy usullardan tashqari, sinflar konstruktorlar deb ataladigan maxsus usullarni belgilashi mumkin. Konstruktorlar berilgan sinfning yangi obyekti yaratilganda chaqiriladi. Konstruktorlar ob'ektni ishga tushirishni amalga oshiradilar.

Agar sinfda konstruktor aniqlanmagan bo‘lsa, ushbu sinf uchun avtomatik ravishda parametrsiz konstruktor yaratiladi.

Yuqorida belgilangan Person sinfida konstruktorlar mavjud emas. Shuning uchun u uchun avtomatik ravishda standart konstruktor yaratiladi, biz undan Person obyektini yaratishimiz mumkin. Xususan, bitta ob'ekt yarataylik:

```
  public class Program{

      public static void main(String[] args) {

          Person tom = new Person(); // obyekt yaratish
          tom.displayInfo();

          // ism va yosh beramiz
          tom.name = "Tom";
          tom.age = 34;
          tom.displayInfo();
      }
  }
  class Person{

      String name;    // ism
      int age;        // yoshi
      void displayInfo(){
          System.out.printf("Ismi: %s \tAge: %d\n", name,
age);
      }
  }
```

Bu ifoda Person ob'ektini yaratish uchun ishlatiladi new Person(). Yangi operator Person obyekti uchun xotira ajratadi. Va keyin hech qanday parametrlarni qabul qilmaydigan standart konstruktor chaqiriladi. Natijada, ushbu ifoda bajarilgandan so‘ng, xotirada Person ob'ektining barcha ma'lumotlari saqlanadigan bo‘lim ajratiladi. Va o‘zgaruvchi tomyaratilgan ob'ektga havola oladi.

Agar konstruktor ob'ekt o‘zgaruvchilari qiymatlarini ishga tushirmasa, ularga standart qiymatlar beriladi. Raqamli tipdagi

o‘zgaruvchilar uchun bu 0 raqami, string tipi va sinflar uchun esa bu qiymat null (ya'ni, aslida qiymatning yo‘qligi).

Ob'ekt yaratilgandan so‘ng, biz tom o‘zgaruvchisi orqali Person ob'ektining o‘zgaruvchilariga kirishimiz va ularning qiymatlarini belgilashimiz yoki olishimiz mumkin, masalan tom.name = "Tom".

Agar ob'ektni yaratishda ba'zi bir mantiqni bajarish kerak bo‘lsa, masalan, sinfning maydonlari ma'lum o‘ziga xos qiymatlarni qabul qilish uchun, u holda sinfda o‘z konstruktorlaringizni belgilashingiz mumkin. Masalan:

```
public class Program{

    public static void main(String[] args) {

        Person bob = new Person();      //
        bob.displayInfo();

        Person tom = new Person("Tom"); //
        tom.displayInfo();

        Person sam = new Person("Sam", 25); //
        sam.displayInfo();
    }
}
class Person{

    String name;    // ism
    int age;        // yosh
    Person()
    {
        name = "Undefined";
        age = 18;
    }
    Person(String n)
    {
        name = n;
        age = 18;
    }
    Person(String n, int a)
    {
        name = n;
        age = a;
```

```
    }
    void displayInfo(){
        System.out.printf("Ism: %s \tAge: %d\n", name, age);
    }
}
```

Endi sinfda uchta konstruktor aniqlangan, ularning har biri har xil miqdordagi parametrlarni oladi va sinf maydonlarining qiymatlarini o‘rnatadi.

Dasturning konsol chiqishi:

this kalit so‘zi

this kalit so‘zi sinfning joriy nusxasiga havolani ifodalaydi. Ushbu kalit so‘z orqali biz ob'ektning o‘zgaruvchilari, usullariga kirishimiz, shuningdek uning konstruktorlarini chaqirishimiz mumkin. Masalan:

```
public class Program{

    public static void main(String[] args) {

        Person undef = new Person();
        undef.displayInfo();

        Person tom = new Person("Tom");
        tom.displayInfo();

        Person sam = new Person("Sam", 25);
        sam.displayInfo();
    }
}
class Person{

    String name;    // ism
    int age;        // yosh
    Person()
    {
        this("Undefined", 18);
    }
    Person(String name)
    {
        this(name, 18);
    }
    Person(String name, int age)
    {
```

```
        this.name = name;
        this.age = age;
    }
    void displayInfo(){
        System.out.printf("Ism: %s \tAge: %d\n", name, age);
    }
}
```

Uchinchi konstruktorda parametrlar sinf maydonlari bilan bir xil nomlanadi. Maydonlar va parametrlarni farqlash uchun this kalit so‘zi ishlatiladi:

this.name = name;

Shunday qilib, bu holda biz nom parametrining qiymati nom maydoniga tayinlanganligini ko‘rsatamiz.

Bundan tashqari, bizda bir xil amallarni bajaradigan uchta konstruktor mavjud: ular nom va yosh maydonlarini o‘rnatadilar. Takrorlashning oldini olish uchun siz sinf konstruktorlaridan birini chaqirish va uning parametrlari uchun kerakli qiymatlarni berish uchun foydalanishingiz mumkin:

```
    Person(String name)
    {
        this(name, 18);
    }
```

Natijada, dasturning natijasi avvalgi misoldagi kabi bo‘ladi.

Konstruktorga qo‘shimcha ravishda ob'ektni dastlabki ishga tushirish ob'ektni ishga tushirish vositasi yordamida amalga oshirilishi mumkin. Initsializator har qanday konstruktordan oldin bajariladi. Ya'ni, boshlang'ichda biz barcha konstruktorlar uchun umumiy kodni joylashtirishimiz mumkin:

```
    public class Program{

        public static void main(String[] args) {

            Person undef = new Person();
            undef.displayInfo();

            Person tom = new Person("Tom");
            tom.displayInfo();
```

```
        }
    }
    class Person{

        String name;    // ism
        int age;        // yosh

        /*initsializator bloke boshlanishi*/
        {
            name = "Undefined";
            age = 18;
        }
        /* initsializator bloke tugashi*/
        Person(){

        }
        Person(String name){

            this.name = name;
        }
        Person(String name, int age){

            this.name = name;
            this.age = age;
        }
        void displayInfo(){
            System.out.printf("Ismi: %s \tAge: %d\n", name,
age);
        }
    }
```

Nazorat savollari.

1. Java ob'ektga yo‘naltirilgan til haqida ma’lumot bering.
2. Sinflarni izohlang.
3. Konstruktor usullarini tavsiflang.
4. This kalit so‘zini qo‘llanilishi haqida tushncha bering

5. VORISLIK. ABSTRAKT SINFLAR VA INTERFEYSLAR. POLIMORFIZM

Reja:

5.1 Java dasturlash tilida Polimorfizm.

5.2 Vorislik. Abstrakt sinflar va interfeyslar.

Ob'ektga yo‘naltirilgan dasturlashning asosiy jihatlaridan biri bu merosdir. Merosdan foydalanib, siz yangi funksiyalarni qo‘shish yoki eskilarini o‘zgartirish orqali mavjud sinflarning funksionalligini kengaytirishingiz mumkin. Masalan, individual shaxsni tavsiflovchi quyidagi Person sinfi mavjud:

```
class Person {
    String name;
    public String getName(){ return name; }

    public Person(String name){

        this.name=name;
    }

    public void display(){

        System.out.println("Name: " + name);
    }
}
```

Va, ehtimol, keyinroq biz korxona xodimini tavsiflovchi yana bir sinfni - Xodimlar sinfini qo‘shishni xohlaymiz. Bu sinf Person klassi bilan bir xil funksionallikni amalga oshirganligi sababli, xodim ham shaxs bo‘lganligi sababli, Employee sinfini Person sinfining hosilasi (vorisi, pastki sinfi) qilish oqilona bo‘lar edi, bu esa o‘z navbatida baza deb ataladi. sinf, ota-ona yoki yuqori sinf:

```
class Employee extends Person{
    public Employee(String name){
        super(name);      // agar asosiy sinfda konstruktor
aniqlangan bulsa.
        //  shuning ikkinchi sinf uni chaqiradi
    }
}
```

Bir sinfni boshqasining merosxo‘ri sifatida e'lon qilish uchun siz merosxo‘r sinf nomidan keyin extends kalit so‘zdan va undan keyin

asosiy sinf nomidan foydalanishingiz kerak. Employee klassi Personga asoslangan va shuning uchun Employee klassi Person sinfidagi barcha maydonlar va usullarni meros qilib oladi.

Agar konstruktorlar tayanch sinfda aniqlangan bo‘lsa, u holda olingan sinf konstruktorida super kalit so‘z yordamida asosiy sinf konstruktorlaridan birini chaqirishingiz kerak. Masalan, Person sinfida bitta parametrni qabul qiluvchi konstruktor mavjud. Shuning uchun, Employee sinfida, konstruktorda siz Person sinfining konstruktorini chaqirishingiz kerak. Ya'ni, murojaat super(name)Person sinfining konstruktoriga murojaatni ifodalaydi.

Konstruktorni chaqirganda, super qavs ichidagi so‘zdan keyin o‘tkazilgan argumentlar ro‘yxati mavjud. Bunday holda, asosiy sinf konstruktoriga murojaat hosil qilingan sinf konstruktorida eng boshida sodir bo‘lishi kerak. Shunday qilib, xodim nomini belgilash asosiy sinf konstruktoriga topshiriladi.

Bundan tashqari, meros olgan sinf konstruktorda boshqa ishni bajarmasa ham, yuqoridagi misoldagidek, baribir asosiy sinf konstruktorini chaqirish kerak.

```
public class Program{

    public static void main(String[] args) {

        Person tom = new Person("Tom");
        tom.display();
        Employee sam = new Employee("Sam");
        sam.display();
    }
}
class Person {

    String name;
    public String getName(){ return name; }

    public Person(String name){

        this.name=name;
    }

    public void display(){

        System.out.println("Name: " + name);
    }
}
```

```
 class Employee extends Person{
     public  Employee(String name){
         super(name);    // agar asosiy sinfda konstruktor
aniqlangan bulsa.
                         //  shuning ikkinchi sinf uni
chaqiradi
     }
 }
```

Olingan sinf asosiy sinfning barcha usullari va maydonlariga kirish huquqiga ega (agar asosiy sinf boshqa paketda bo‘lsa ham), xususiy modifikator bilan belgilanganlardan tashqari . Bunday holda, olingan sinf o‘z maydonlari va usullarini ham qo‘shishi mumkin:

```
  public class Program{

      public static void main(String[] args) {

          Employee sam = new Employee("Sam", "Microsoft");
          sam.display();  // Sam
          sam.work();     // Sam works in Microsoft
      }
  }
  class Person {

      String name;
      public String getName(){ return name; }

      public Person(String name){

          this.name=name;
      }

      public void display(){

          System.out.println("Name: " + name);
      }
  }
  class Employee extends Person{

      String company;

      public Employee(String name, String company) {

          super(name);
          this.company=company;
      }
      public void work(){
```

```
            System.out.printf("%s works in %s \n", getName(),
company);
        }
    }
```

Bunday holda, Xodimlar toifasi xodimning ish joyini, shuningdek, ish uslubini saqlaydigan kompaniya maydonini qo‘shadi.

Usulni bekor qilish

Olingan sinf o‘z usullarini belgilashi va asosiy sinfdan meros bo‘lib qolgan usullarni bekor qilishi mumkin. Masalan, Employee sinfidagi ko‘rsatish usulini bekor qilaylik:

```
public class Program{

    public static void main(String[] args) {

        Employee sam = new Employee("Sam", "Microsoft");
        sam.display();  // Sam
                        // Works in Microsoft
    }
}
class Person {

    String name;
    public String getName(){ return name; }

    public Person(String name){

        this.name=name;
    }

    public void display(){

        System.out.println("Name: " + name);
    }
}
class Employee extends Person{

    String company;

    public Employee(String name, String company) {

        super(name);
        this.company=company;
    }
    @Override
```

```
        public void display(){

            System.out.printf("Name: %s \n", getName());
            System.out.printf("Works in %s \n", company);
        }
    }
```

Bekor qilinadigan usuldan oldin @Override izohi mavjud. Ushbu izoh printsipial jihatdan ixtiyoriydir.

Usulni bekor qilishda u asosiy sinfdagi kirish darajasidan kam bo‘lmagan kirish darajasiga ega bo‘lishi kerak. Misol uchun, agar asosiy sinfdagi metod umumiy o‘zgartiruvchiga ega bo‘lsa, hosila sinfidagi usul ham umumiy o‘zgartiruvchiga ega bo‘lishi kerak.

Biroq, bu holda, biz Employee-dagi ko‘rsatish usulining bir qismi asosiy sinfning ko‘rsatish usulidagi amallarni takrorlashini ko‘ramiz. Shunday qilib, biz Xodimlar sinfini qisqartirishimiz mumkin:

```
    class Employee extends Person{

        String company;

        public Employee(String name, String company) {

            super(name);
            this.company=company;
        }
        @Override
        public void display(){

            super.display();
            System.out.printf("Works in %s \n", company);
        }
    }
```

Super kalit so‘zi yordamida biz asosiy sinf usullarini amalga oshirishga ham kirishimiz mumkin.

Merosni taqiqlash

Meros juda qiziqarli va samarali mexanizm bo‘lsa-da, ba'zi hollarda undan foydalanish istalmagan bo‘lishi mumkin. Va bu holda, siz final kalit so‘zidan foydalanib, merosni o‘chirib qo‘yishingiz mumkin. Masalan:

```
public final class Person {
}
```

Agar Person sinfi shu tarzda aniqlangan bo‘lsa, unda quyidagi kod xato bo‘lar edi va ishlamaydi, chunki biz merosni o‘chirib qo‘yamiz:

```
class Employee extends Person{ {
}
```

Merosni taqiqlashdan tashqari, siz individual usullarni bekor qilishni ham taqiqlashingiz mumkin. Masalan, yuqoridagi misolda usul bekor qilingan display(), biz uni bekor qilishni taqiqlaymiz:

```
public class Person {

    //.....................

    public final void display(){

        System.out.println("Имя: " + name);
    }
}
```

Bunday holda, Xodimlar sinfi ko‘rsatish usulini bekor qila olmaydi.

Abstrakt sinflar

Oddiy sinflarga qo‘shimcha ravishda Java-da mavhum sinflar mavjud. Abstrakt sinf oddiy sinfga o‘xshaydi. Mavhum sinfda siz maydonlar va usullarni ham belgilashingiz mumkin, lekin ayni paytda ob'ektni yoki mavhum sinfning namunasini yarata olmaysiz. Abstrakt sinflar avlod sinflari uchun asosiy funksionallikni ta'minlash uchun mo‘ljallangan. Va olingan sinflar allaqachon ushbu funktsiyani amalga oshiradi.

Mavhum sinflarni belgilashda abstract kalit so‘z ishlatiladi:

```
public abstract class Human{

    private String name;

    public String getName() { return name; }
}
```

Ammo asosiy farq shundaki, biz uning ob'ektini yaratish uchun mavhum sinfning konstruktoridan foydalana olmaymiz. Masalan, quyidagicha:

Human h = new Human();

Oddiy usullardan tashqari, mavhum sinf mavhum usullarni o‘z ichiga olishi mumkin. Bunday usullar abstract kalit so‘z yordamida aniqlanadi va amalga oshirilmaydi:

public abstract void display();

Olingan sinf asosiy abstrakt sinfda mavjud bo‘lgan barcha mavhum usullarni bekor qilishi va amalga oshirishi kerak. Shuni ham hisobga olish kerakki, agar sinfda kamida bitta abstrakt metod mavjud bo‘lsa, u holda bu sinf mavhum sifatida belgilanishi kerak.

Abstrakt sinflar nima uchun kerak? Aytaylik, biz bank operatsiyalariga xizmat ko‘rsatish dasturini tuzamiz va unda uchta sinfni aniqlaymiz: shaxsni tavsiflovchi Person, bank xodimini tavsiflovchi Employee va bank mijozini ifodalovchi Client sinfi. Shubhasiz, Employee va Client sinflari Person sinfidan kelib chiqadi, chunki ikkala sinfda ham umumiy maydonlar va usullar mavjud. Va barcha ob'ektlar bankning xodimi yoki mijozini ifodalashi sababli, biz to‘g'ridan-to‘g'ri Person sinfidan ob'ektlar yaratmaymiz.

```
  public class Program{

      public static void main(String[] args) {

          Employee sam = new Employee("Sam", "Leman
Brothers");
          sam.display();
          Client bob = new Client("Bob", "Leman Brothers");
          bob.display();
      }
  }
  abstract class Person {

      private String name;

      public String getName() { return name; }

      public Person(String name){

          this.name=name;
      }

      public abstract void display();
  }

  class Employee extends Person{

      private String bank;

      public Employee(String name, String company) {

          super(name);
          this.bank = company;
      }

      public void display(){

          System.out.printf("Employee Name: %s \t Bank: %s
\n", super.getName(), bank);
      }
  }

  class Client extends Person
  {
      private String bank;

      public Client(String name, String company) {

          super(name);
          this.bank = company;
```

```
        }

        public void display(){

            System.out.printf("Client Name: %s \t Bank: %s \n",
super.getName(), bank);
        }
   }
```

Meros mexanizmi juda qulay, ammo uning cheklovlari bor. Xususan, biz faqat bir sinfdan meros olishimiz mumkin, masalan, C++ tilidan farqli o‘laroq, bu erda bir nechta meros mavjud.

Java tilida interfeyslar bu muammoni qisman hal qiladi. Interfeyslar ma'lum bir amalga oshirishga ega bo‘lmagan ba'zi funksiyalarni belgilaydi, keyin esa ushbu interfeyslardan foydalanadigan sinflar tomonidan amalga oshiriladi. Va bitta sinf ko‘plab interfeyslarni qo‘llashi mumkin.

Interfeysni aniqlash uchun interfeys kalit so‘zi ishlatiladi. Masalan:

```
interface Printable{
    void print();
}
```

Ushbu interfeys Printable deb ataladi. Interfeys konstantalar va usullarni belgilashi mumkin, ular amalga oshirilishi mumkin yoki bo‘lmasligi mumkin. Amalga oshirilmagan usullar mavhum sinflarning mavhum usullariga o‘xshaydi. Shunday qilib, bu holda, amalga oshirilmaydigan bir usul e'lon qilinadi.

Barcha interfeys usullarida kirish modifikatorlari mavjud emas, lekin aslida standart kirish ommaviydir, chunki interfeysning maqsadi sinfni amalga oshirish uchun funksionallikni aniqlashdir. Shuning uchun, barcha funktsiyalar amalga oshirish uchun ochiq bo‘lishi kerak.

Sinfni interfeysni amalga oshirish uchun siz implements kalit so‘zidan foydalanishingiz kerak:

```
public class Program{

    public static void main(String[] args) {

        Book b1 = new Book("Java. Complete Referense.", "H.
Shildt");
        b1.print();
    }
}
interface Printable{

    void print();
}
class Book implements Printable{
```

```
    String name;
    String author;

    Book(String name, String author){

        this.name = name;
        this.author = author;
    }

    public void print() {

        System.out.printf("%s (%s) \n", name, author);
    }
}
```

Bu holda Book klassi Printable interfeysini amalga oshiradi. Shuni hisobga olish kerakki, agar sinf interfeysdan foydalansa, u holda yuqoridagi holatda bo‘lgani kabi print interfeysning barcha usullarini amalga oshirishi kerak. Keyin asosiy metodda Book sinfining ob'ektini yaratishimiz va uni chop etish usulini chaqirishimiz mumkin. Agar sinf hech qanday interfeys usullarini amalga oshirmasa, u holda bu sinf mavhum sifatida belgilanishi kerak va uning mavhum bo‘lmagan avlod sinflari bu usullarni amalga oshirishi kerak.

Shu bilan birga, biz to‘g'ridan-to‘g'ri interfeys ob'ektlarini yarata olmaymiz, shuning uchun quyidagi kod ishlamaydi:

Printable pr = new Printable();

pr.print();

Interfeyslardan foydalanishning afzalliklaridan biri shundaki, ular ilovangizga moslashuvchanlikni qo‘shish imkonini beradi. Masalan, Book sinfiga qo‘shimcha ravishda biz Printable interfeysini amalga oshiradigan boshqa sinfni aniqlaymiz:

```
class Journal implements Printable {

    private String name;

    String getName(){
        return name;
    }

    Journal(String name){

        this.name = name;
    }
    public void print() {
        System.out.println(name);
```

```
        }
    }
```

Book klassi va Journal sinfi Chop etish interfeysini amalga oshirishi bilan bog'liq. Shunday qilib, biz dasturda ikkala sinfning misollari sifatida dinamik ravishda Chop etish ob'ektlarini yaratishimiz mumkin:

```
    public class Program{

        public static void main(String[] args) {

            Printable printable = new Book("Java. Complete
Reference", "H. Shildt");
            printable.print();      //  Java. Complete
Reference (H. Shildt)
            printable = new Journal("Foreign Policy");
            printable.print();      // Foreign Policy
        }
    }
    interface Printable{

        void print();
    }
    class Book implements Printable{

        String name;
        String author;

        Book(String name, String author){

            this.name = name;
            this.author = author;
        }

        public void print() {

            System.out.printf("%s (%s) \n", name, author);
        }
    }
    class Journal implements Printable {

        private String name;

        String getName(){
            return name;
        }
```

```
    Journal(String name){

        this.name = name;
    }
    public void print() {
        System.out.println(name);
    }
}
```

Turlarni o‘zgartirishdagi interfeyslar

Turni o‘zgartirish haqida aytilganlarning barchasi interfeyslar uchun ham amal qiladi. Masalan, Journal klassi Printable interfeysini amalga oshirganligi sababli Printable tipidagi o‘zgaruvchi Journal tipidagi ob'ektga havolani saqlashi mumkin:

```
Printable p =new Journal("Foreign Affairs");
p.print();
// Interfeysda getName metodi mavjud emas
String name = ((Journal)p).getName();
System.out.println(name);
```

Va agar biz Journal sinfining Printable interfeysida emas, balki Journal sinfining o‘zida aniqlangan usullariga kirishni istasak, unda biz aniq turdagi konvertatsiya qilishimiz kerak:((Journal)p).getName();

Interfeyslardagi konstantalar

Usullarga qo‘shimcha ravishda, statik konstantalar interfeyslarda aniqlanishi mumkin:

```
interface Stateable{
    int OPEN = 1;
    int CLOSED = 0;
    void printState(int n);
}
```

Bunday konstantalarda modifikatorlar bo‘lmasa ham, ular sukut bo‘yicha kirish modifikatoriga ega public static final va shuning uchun ularning qiymatiga dasturning istalgan joyidan kirish mumkin.

Konstantalardan foydalanish:

```
public class Program{

    public static void main(String[] args) {
```

```
        WaterPipe pipe = new WaterPipe();
        pipe.printState(1);
    }
}
class WaterPipe implements Stateable{

    public void printState(int n){
        if(n==OPEN)
            System.out.println("Water is opened");
        else if(n==CLOSED)
            System.out.println("Water is closed");
        else
            System.out.println("State is invalid");
    }
}
interface Stateable{

    int OPEN = 1;
    int CLOSED = 0;

    void printState(int n);
}
```

Interfeyslarni bir nechta amalga oshirish

Agar biz sinfda bir nechta interfeyslarni qo‘llashimiz kerak bo‘lsa, ularning barchasi amalga oshiriladigan so‘zdan keyin vergul bilan ajratiladi:

```
interface Printable {
    // методы интерфейса
}

interface Searchable {
    // методы интерфейса
}

class Book implements Printable, Searchable{
    // реализация класса
}
```

Ichki interfeyslar

Sinflar singari, interfeyslarni ham o‘rnatish mumkin, ya'ni ular sinflar yoki boshqa interfeyslar ichida aniqlanishi mumkin. Masalan:

```
class Printer{
    interface Printable {
        void print();
    }
}
```

Bunday interfeysdan foydalanganda biz uning to‘liq nomini sinf nomi bilan birga ko‘rsatishimiz kerak:

```
public class Journal implements Printer.Printable {

    String name;

    Journal(String name){

        this.name = name;
    }
    public void print() {
        System.out.println(name);
    }
}
```

Interfeysdan foydalanish avvalgi holatlarga o‘xshash bo‘ladi:

```
Printer.Printable p =new Journal("Foreign Affairs");
p.print();
```

Interfeyslar usullarning parametrlari va natijalari sifatida

Xuddi sinflarda bo‘lgani kabi, interfeyslar usul parametrlari turi yoki qaytarish turi sifatida ishlatilishi mumkin:

```
public class Program{

    public static void main(String[] args) {

        Printable printable = createPrintable("Foreign Affairs",false);
        printable.print();

        read(new Book("Java for impatients", "Cay Horstmann"));
        read(new Journal("Java Dayly News"));
    }

    static void read(Printable p){

        p.print();
    }
```

```
        static Printable createPrintable(String name,
boolean option){

            if(option)
                return new Book(name, "Undefined");
            else
                return new Journal(name);
        }
    }
    interface Printable{

        void print();
    }
    class Book implements Printable{

        String name;
        String author;

        Book(String name, String author){

            this.name = name;
            this.author = author;
        }

        public void print() {

            System.out.printf("%s (%s) \n", name, author);
        }
    }
    class Journal implements Printable {

        private String name;

        String getName(){
            return name;
        }

        Journal(String name){

            this.name = name;
        }
        public void print() {
            System.out.println(name);
        }
    }
```

Usul read() parametr sifatida chop etish mumkin bo‘lgan interfeys ob'ektini oladi, shuning uchun biz bu usulga Book ob'ektini ham, Journal ob'ektini ham o‘tkazishimiz mumkin.

Usul createPrintable() chop etish mumkin bo‘lgan ob'ektni qaytaradi, shuning uchun biz kitob va jurnal ob'ektini ham qaytarishimiz mumkin.

Nazorat savollari.

1. Ob'ektga yo‘naltirilgan dasturlashda merosdan foydalanishni tushuntiring.
2. Ob'ektga yo‘naltirilgan dasturlashda usulni bekor qilish amalini izohlang.
3. Ob'ektga yo‘naltirilgan dasturlashda Abstrakt sinflarni tavsiflang.
4. Java tilida interfeyslar deganda nimani tushunasiz.

6. ISTISNO HOLATLAR. HODISALARNI QAYTA ISHLASH. EXCEPTION, THROWABLE, HANDLING. TRY…CATCH YORDAMIDA USHLASH.

Reja:

6.1 Java dasturlash tilida istisnolar va hodisalar bilan ishlash.

6.2 Exception, Throwable, Handling. Try…catch.

Throws operatori.

Ba'zida istisno keltirishi mumkin bo‘lgan usulning o‘zi istisno bilan ishlamaydi. Bunday holda, usul deklaratsiyasi bu usulni chaqirganda qayta ishlanishi kerak bo‘lgan throws bayonotidan foydalanadi. Misol uchun, bizda faktorialni hisoblash usuli bor va agar usulga 1 dan kichik raqam o‘tkazilsa, vaziyatni hal qilishimiz kerak:

```
     public static int getFactorial(int num) throws
Exception{

          if(num<1) throw new Exception("The number is less
than 1");
          int result=1;
          for(int i=1; i<=num;i++){

               result*=i;
          }
          return result;
     }
```

Throw shartli operator istisno qiladi. Shu bilan birga, metodning o‘zi try..catch yordamida bu istisnoni hal qilmaydi, shuning uchun ifoda usul throws Exception ta'rifida ishlatiladi

Endi, ushbu usulni chaqirganda, tashlangan istisnoni hal qilishimiz kerak:

```
     public static void main(String[] args){

          try{
               int result = getFactorial(-6);

               System.out.println(result);
          }
          catch(Exception ex){

               System.out.println(ex.getMessage());
          }
     }
```

Istisnoni hisobga olmasak, biz kompilyatsiya xatosiga duch kelamiz va dasturni kompilyatsiya qila olmaymiz.

Shu bilan bir qatorda, biz throws bayonotini o‘tkazib yuborishimiz va buning o‘rniga istisnoni to‘g'ridan-to‘g'ri usulda hal qilishimiz mumkin:

```
    public static int getFactorial(int num){

        int result=1;
        try{
            if(num<1) throw new Exception("The number is
less than 1");

            for(int i=1; i<=num;i++){

                result*=i;
            }
        }
        catch(Exception ex){

            System.out.println(ex.getMessage());
            result=num;
        }
        return result;
    }
```

Istisno sinflar

Barcha istisnolar uchun asosiy sinf Throwable sinfidir. Ikkita sinf allaqachon undan meros bo‘lib qolgan: Error va Exception. Boshqa barcha sinflar bu ikki sinfdan olingan.

Error klassi Java ish vaqtidagi ichki xatolarni tavsiflaydi. Dasturchining bunday xatolarni boshqarish qobiliyati juda cheklangan.

Istisnolarning o‘zi Exception sinfidan meros qilib olingan. Ushbu istisnolar orasida RuntimeException sinfini ajratib ko‘rsatish kerak. RuntimeException tekshirilmagan istisnolar guruhi deb ataladigan asosiy sinfdir - kompilyator bunday istisnolar qayta ishlanganligini tekshirmaydi va ularni usul deklaratsiyasida throws bayonoti bilan birga olib tashlash mumkin. Bunday istisnolar ishlab chiquvchi xatolarining natijasidir, masalan, noto‘g'ri turdagi konvertatsiya yoki massiv chegarasidan tashqariga chiqish.

Tekshirilmagan istisnolar sinflarining ba'zilari:

- A**rithmeticException**: nolga bo‘linganda chiqarilgan istisno
- **IndexOutOfBoundException**: indeks massiv chegarasidan tashqarida
- **IllegalArgumentException**: Usul chaqirilganda yaroqsiz argument ishlatilgan
- **NullPointerException**: Null havola ishlatilgan
- **NumberFormatException**: satrni raqamga aylantirishda xato

Exception sinfidan olingan barcha boshqa sinflar tekshirilgan istisnolar deb ataladi.

Tekshirilgan istisnolarning ba'zi sinflari quyidagilardir:

- **CloneNotSupportedException**: Ob'ekti klonlanayotgan sinf Cloneable interfeysini amalga oshirmaydi
- **InterruptedException**: Mavzu boshqa ip bilan uzildi
- **ClassNotFoundException**: sinf topilmadi

Bunday istisnolar try..catch konstruksiyasi yordamida hal qilinadi. Yoki throws iborasidan keyin istisnolarni ko‘rsatish orqali ishlov berishni ushbu usulni chaqiradigan usulga o‘tkazishingiz mumkin:

```
public Person clone() throws CloneNotSupportedException{

    Person p = (Person) super.clone();
    return p;

}
```

Barcha istisnolar sinflari Exception sinfidan meros bo‘lganligi sababli, ularning barchasi istisnoning tabiati haqida ma'lumot olish imkonini beruvchi uning bir qator usullarini meros qilib oladi. Ushbu usullar orasida biz eng muhimlarini ta'kidlaymiz:

- **getMessage()** usuli istisno xabarini qaytaradi
- **getStackTrace()** usuli istisnoning stek izini o‘z ichiga olgan massivni qaytaradi.
- **printStackTrace()** usuli stek izini ko‘rsatadi

```
try{
    int x = 6/0;
}
catch(Exception ex){
```

```
        ex.printStackTrace();
    }
```

O‘zingizning istisno sinflaringizni yaratish

Standart Java sinf kutubxonasida taqdim etilgan istisno sinflari dasturni bajarish jarayonida yuzaga kelishi mumkin bo‘lgan ko‘pgina istisno holatlarini tavsiflagan bo‘lsa-da, siz o‘zingizning mantiqingiz bilan o‘zingizning istisno sinflaringizni yaratishingiz kerak bo‘lgan vaqtlar mavjud.

O‘zingizning istisno sinfingizni yaratish uchun uni Exception sinfidan meros qilib olishingiz kerak. Misol uchun, bizda faktorialni hisoblaydigan sinf mavjud va agar usulga o‘tgan raqam 1 dan kichik bo‘lsa, biz alohida istisno qilishimiz kerak:

```
    class Factorial{

        public static int getFactorial(int num) throws
FactorialException{

            int result=1;
            if(num<1) throw new FactorialException("The
number is less than 1", num);

            for(int i=1; i<=num;i++){

                result*=i;
            }
            return result;
        }
    }

    class FactorialException extends Exception{

        private int number;
        public int getNumber(){return number;}
        public FactorialException(String message, int num){

            super(message);
            number=num;
        }
    }
```

Bu erda faktorialni hisoblash bilan bog'liq xatolikni aniqlash uchun FactorialExceptionIstisnodan meros bo‘lgan va hisoblash haqidagi barcha ma'lumotlarni o‘z ichiga olgan sinf aniqlanadi.

FactorialException konstruktorida xato xabari Exception asosiy sinf konstruktoriga uzatiladi: super(message). Bundan tashqari, faktorial hisoblanayotgan raqamni saqlash uchun alohida maydon mo‘ljallangan.

Istisnoni tashlash uchun throw: ifodasi yordamida faktoriy hisoblash usulida istisno tashlanadi throw new FactorialException ("Число не может быть меньше 1", num). Bundan tashqari, bu istisno try..catch tomonidan ishlanmaganligi sababli, biz throws iborasi yordamida ishlov berishni chaqiruv usuliga o‘tkazamiz: public static int getFactorial(int num) throws FactorialException

Endi biz asosiy usulda sinfdan foydalanamiz:

```
public static void main(String[] args){

    try{
        int result = Factorial.getFactorial(6);
        System.out.println(result);
    }
    catch(FactorialException ex){

        System.out.println(ex.getMessage());
        System.out.println(ex.getNumber());
    }
}
```

Nazorat savollari.

1. Istesno holatlarda Throws operatorining vazifasini izohlang.
2. Istesno holatda sinflarni tushuntiring.
3. Tekshirilmagan istisnolar sinflarining ba’zilarini sanab o‘ting.
4. Tekshirilgan istisnolar sinflariga misollar keltiring.
5. Try..catch konstruksiyasini izohlang.
6. Istisno holatlarini yaratish haqida ma’lumot bering.

7. KOLLEKSIYALAR. COLLECTION VA ITERATOR INTERFEYSLARI. RO‘YXATLAR. RO‘YXATLAR MASSIVI. TO‘PLAMLAR. DARAXT VA XESH TO‘PLAMLAR. IZLASH VA SARALASH.

Reja

7.1 Java dasturlash tilida Kolleksiyalar.

7.2 Collection va Iterator interfeyslari. Ro‘yxatlar. Ro‘yxatlar massivi.

7.3 To‘plamlar. Daraxt va Xesh to‘plamlar. Izlash va saralash.

To‘plam turlari. To‘plam interfeysi

Massivlar Java-da ma'lumotlar to‘plamini saqlash uchun ishlatiladi. Biroq, ulardan foydalanish har doim ham qulay emas. To‘plamlar bu muammoni Java-da hal qiladi. To‘plam sinflari java.util paketida joylashgan, shuning uchun to‘plamlardan foydalanishdan oldin ushbu paketni kiritishingiz kerak.

Java-da ko‘plab to‘plamlar mavjud bo‘lsa-da, ularning barchasi izchil va mantiqiy tizimni tashkil qiladi. Birinchidan, barcha to‘plamlar asosiy funksionallikni belgilaydigan u yoki bu interfeysdan foydalanishga asoslangan. Ushbu interfeyslar orasida quyidagilar mavjud:

- **Collection**: barcha to‘plamlar va boshqa to‘plam interfeyslari uchun asosiy interfeys
- **Queue**: To‘plam interfeysini meros qilib oladi va navbat ko‘rinishidagi ma'lumotlar tuzilmalari uchun funksionallikni ta'minlaydi
- **Deque**: Queue interfeysini meros qilib oladi va ikki tomonlama navbatlar uchun funksionallikni ta'minlaydi
- **List**: To‘plam interfeysini meros qilib oladi va oddiy ro‘yxatlar funksiyasini taqdim etadi
- **Set**: shuningdek, Collection interfeysini kengaytiradi va noyob ob'ektlar to‘plamini saqlash uchun ishlatiladi
- **SortedSet**: tartiblangan to‘plamlarni yaratish uchun Set interfeysini kengaytiradi
- **NavigableSet**: SortedSet interfeysini mos ravishda qidirish mumkin bo‘lgan to‘plamlarni yaratish uchun kengaytiradi

- **Map**: Har bir element o‘ziga xos kalit va qiymatga ega bo‘lgan lug'at uslubidagi ma'lumotlar tuzilmalarini yaratish uchun mo‘ljallangan. Boshqa to‘plam interfeyslaridan farqli o‘laroq, u Collection interfeysidan meros bo‘lmaydi.

Ushbu quyidagi interfeyslar qisman mavhum sinflar tomonidan amalga oshiriladi:

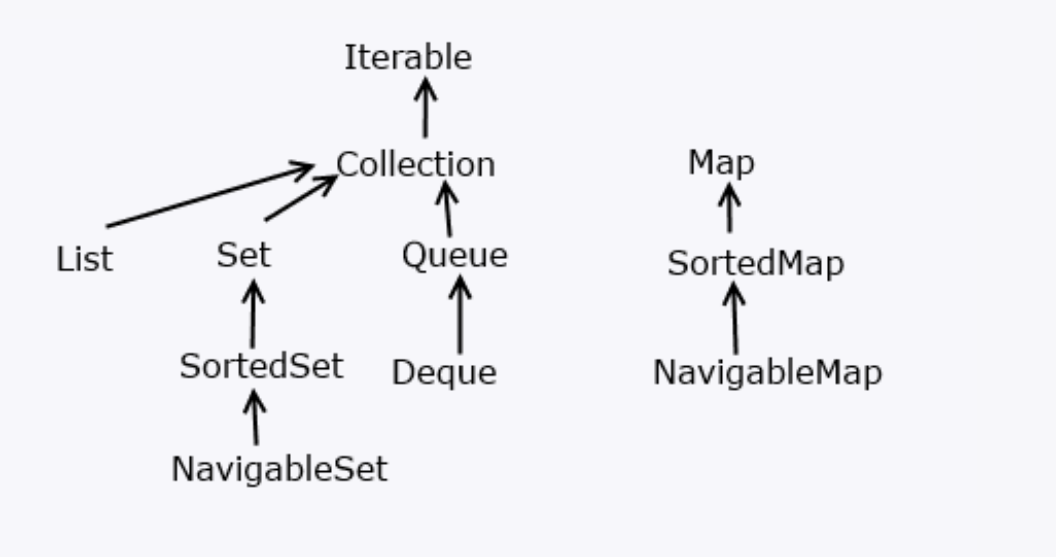
- **AbstractCollection**: Collection interfeysidan foydalanadigan boshqa to‘plamlar uchun asosiy mavhum sinf
- **AbstractList**: AbstractCollection sinfini kengaytiradi va ro‘yxatlar ko‘rinishida to‘plamlar yaratish uchun mo‘ljallangan List interfeysidan foydalanadi.
- **AbstractSet**: AbstractCollection sinfini kengaytiradi va to‘plamlarni to‘plam sifatida yaratish uchun Set interfeysidan foydalanadi
- **AbstractQueue**: AbstractCollection sinfini kengaytiradi va navbatlar va steklar ko‘rinishida to‘plamlar yaratish uchun mo‘ljallangan Queue interfeysidan foydalanadi.
- **AbstractSequentialList**: Shuningdek, AbstractList sinfini kengaytiradi va List interfeysini amalga oshiradi. Bog'langan ro‘yxatlarni yaratish uchun foydalaniladi
- **AbstractMap**: kalit-qiymat juftligi sifatida ob'ektlarning lug'atga o‘xshash to‘plamlarini yaratish uchun mo‘ljallangan Map interfeysidan foydalanadi

Yuqorida tavsiflangan interfeyslar va abstrakt sinflardan foydalanib, Java keng ko‘lamli to‘plam sinflarini - ro‘yxatlar, to‘plamlar, navbatlar, displeylar va boshqalarni amalga oshiradi, ular orasida quyidagilarni ajratib ko‘rsatish mumkin:

- **ArrayList**: ob'ektlarning oddiy ro‘yxati
- **LinkedList**: Bog'langan ro‘yxatni ifodalaydi
- **ArrayDeque**: to‘plamning boshida yoki oxirida biz qo‘shishimiz va o‘chirishimiz mumkin bo‘lgan ikki tomonlama navbat klassi
- **HashSet**: ob'ektlar to‘plami yoki xesh to‘plami, bu erda har bir element kalitga ega - noyob xesh kodi
- **TreeSet**: daraxt ko‘rinishidagi tartiblangan ob'ektlar to‘plami
- **LinkedHashSet**: bog'langan xesh to‘plami
- **PriorityQueue**: ustuvor navbat

- **HashMap**: har bir ob'ekt noyob kalit va ba'zi qiymatga ega bo‘lgan lug'at ma'lumotlar tuzilmasi
- **TreeMap**: daraxtga o‘xshash ma'lumotlar tuzilmasi bo‘lib, unda har bir element noyob kalit va ba'zi bir qiymatga ega

Sxematik tarzda, to‘plamlarning butun tizimini qisqacha quyidagicha ifodalash mumkin:



To‘plam interfeysi

To‘plam interfeysi barcha to‘plamlar uchun asos bo‘lib, asosiy funksiyalarni belgilaydi:

```
public interface Collection<E> extends Iterable<E>{

    // metodlarni aniqlash
}
```

To‘plam interfeysi umumiy bo‘lib, Iterable interfeysini kengaytiradi, shunda barcha to‘plam ob'ektlari tsiklda takrorlanishi mumkin.

To‘plam interfeysining usullari orasida quyidagilar mavjud:

- boolean add (E item): To‘plamga element ob'ektini qo‘shadi. Muvaffaqiyatli qo‘shilgan bo‘lsa, rost, muvaffaqiyatsiz qo‘shilsa, noto‘g'ri qaytariladi
- boolean addAll (Collection<? extends E> col): To‘plamdagi barcha elementlarni to‘plamga qo‘shadi. Muvaffaqiyatli qo‘shilgan bo‘lsa, rost, muvaffaqiyatsiz qo‘shilsa, noto‘g'ri qaytariladi
- void clear (): to‘plamdan barcha elementlarni olib tashlaydi

- boolean contains (Object item): agar element ob'ekti to‘plamda bo‘lsa, true qiymatini qaytaradi, aks holda noto‘g'ri qaytaradi
- boolean isEmpty (): agar to‘plam bo‘sh bo‘lsa, true qiymatini qaytaradi, aks holda noto‘g'ri qaytaradi
- Iterator<E> iterator (): To‘plam elementlarini takrorlash uchun Iterator ob'ektini qaytaradi
- boolean remove (Object item): agar element ob'ekti to‘plamdan muvaffaqiyatli olib tashlangan bo‘lsa, true qiymatini qaytaradi, aks holda noto‘g'ri qaytaradi
- boolean removeAll (Collection<?> col): Joriy to‘plamdan barcha to‘plam ob'ektlarini olib tashlaydi. Agar joriy to‘plam o‘zgargan bo‘lsa, rost, aks holda noto‘g'ri qaytariladi
- boolean retainAll (Collection<?> col): Joriy to‘plamdagi barcha ob'ektlarni o‘chiradi, to‘plamdagilardan tashqari. Agar joriy to‘plam o‘chirilgandan so‘ng o‘zgargan bo‘lsa, true qiymatini qaytaradi, aks holda noto‘g'ri qaytaradi
- int size (): to‘plamdagi elementlar sonini qaytaradi
- Object[] toArray (): to‘plamning barcha elementlarini o‘z ichiga olgan massivni qaytaradi

To‘plam interfeysida mavjud bo‘lgan barcha bu va boshqa usullar barcha to‘plamlar tomonidan amalga oshiriladi, shuning uchun umuman to‘plamlar bilan ishlashning umumiy tamoyillari bir xil bo‘ladi. Izchil interfeys turli turdagi to‘plamlarni tushunish va ular bilan ishlashni osonlashtiradi. Shunday qilib, elementni qo‘shish addqo‘shilgan elementni parametr sifatida qabul qiladigan usul yordamida amalga oshiriladi. Yo‘q qilish uchun usul chaqiriladi remove(). clear usuli to‘plamni tozalaydi va size usuli to‘plamdagi elementlar sonini qaytaradi.

ArrayList klassi va List interfeysi

Oddiy ro‘yxatlarni yaratish uchun To‘plam interfeysi funksiyasini kengaytiruvchi List interfeysidan foydalanamiz. Ro‘yxat interfeysining eng ko‘p ishlatiladigan usullaridan ba'zilarini keltirib o‘tamiz:

- void add(int index, E obj) : obj ob'ektini index indeksidagi ro‘yxatga qo‘shadi
- boolean addAll(int index, Collection<? extensions E> col) : col kolleksiyasining barcha elementlarini index indeksidagi ro‘yxatga

qo‘shadi. Agar ro‘yxat qo‘shish natijasida o‘zgartirilgan bo‘lsa, u holda rost qaytariladi, aks holda false qaytariladi

- E get(int index) : ob'ektni index indeksidagi ro‘yxatdan qaytaradi
- int indexOf(Object obj) : Ro‘yxatdagi obj ob’yektining birinchi marta paydo bo‘lgan indeksini qaytaradi. Agar ob'ekt topilmasa, u holda -1 qaytariladi.
- int lastIndexOf(Object obj) : Ro‘yxatdagi ob'ektning oxirgi marta paydo bo‘lgan indeksini qaytaradi. Agar ob'ekt topilmasa, u holda -1 qaytariladi.
- ListIterator<E> listIterator() : Ro‘yxat elementlarini takrorlash uchun ListIterator obyektini qaytaradi
- statik <E> List<E> of(elementlar) : Elementlar to‘plamidan List obyektini yaratadi
- E remove(int index) : O‘chirilgan ob'ektni qaytarish paytida ob'ektni indeksdagi ro‘yxatdan o‘chiradi
- E set(int index, E obj) : ob'ekt qiymatini indeksda topilgan elementga belgilaydi
- void sort(Comparator<? super E> comp) : ro‘yxatni solishtiruvchi kompilyatsiya yordamida tartiblaydi
- List<E> subList(int start, int end) : boshlanish va tugash indekslari orasidagi ro‘yxatdagi elementlar to‘plamini oladi.

ArrayList klassi AbstractList sinfidan o‘z funksionalligini meros qilib olgan va List interfeysidan foydalanadigan umumiy to‘plamni ifodalaydi. Oddiy so‘z bilan aytganda, ArrayList massivga o‘xshash oddiy ro‘yxat bo‘lib, undagi elementlar soni aniqlanmagan.

ArrayList quyidagi konstruktorlarga ega:

- ArrayList(): bo‘sh ro‘yxat yaratadi
- ArrayList(Collection <? extends E> col): To‘plamning barcha elementlari qo‘shiladigan ro‘yxatni yaratadi.
- ArrayList (int capacity): boshlang'ich sig'imga ega bo‘lgan ro‘yxatni yaratadi

ArrayList-dagi sig'im ob'ektlarni saqlash uchun ishlatiladigan massiv hajmini ifodalaydi. Elementlarni qo‘shganda xotira aslida qayta taqsimlanadi - yangi massiv yaratish va unga eski massivdan elementlarni nusxalash. ArrayList sig'imini o‘rnatish dastlab bunday xotirani qayta taqsimlashni kamaytirishga imkon beradi va shu bilan ishlashni yaxshilaydi.

Dasturda ArrayList sinfi va uning ba'zi usullaridan foydalanamiz:

```
 import java.util.ArrayList;

 public class Program{

     public static void main(String[] args) {

         ArrayList<String> people = new ArrayList<String>();
         // добавим в список ряд элементов
         people.add("Tom");
         people.add("Alice");
         people.add("Kate");
         people.add("Sam");
         people.add(1, "Bob"); // добавляем элемент по
индексу 1

         System.out.println(people.get(1));// получаем 2-й
объект
         people.set(1, "Robert"); // установка нового
значения для 2-го объекта

         System.out.printf("ArrayList has %d elements \n",
people.size());
         for(String person : people){

             System.out.println(person);
         }
         // проверяем наличие элемента
         if(people.contains("Tom")){

             System.out.println("ArrayList contains Tom");
         }

         // удалим несколько объектов
         // удаление конкретного элемента
         people.remove("Robert");
         // удаление по индексу
         people.remove(0);

         Object[] peopleArray = people.toArray();
         for(Object person : peopleArray){

             System.out.println(person);
         }
     }
 }
```

Bu yerda ArrayList obyekti String sinfi bilan yoziladi, shuning uchun ro‘yxat faqat satrlarni saqlaydi. ArrayList klassi Collection<E>

interfeysidan foydalanganligi sababli, biz ushbu interfeysning usullaridan ro‘yxatdagi ob'ektlarni boshqarish uchun foydalanishimiz mumkin.

Qo‘shish uchun usul deyiladi add. Uning yordamida biz ro‘yxat oxiriga ob'ekt qo‘shishimiz mumkin: people.add("Tom"). Shuningdek, biz ob'ektni ro‘yxatning ma'lum bir joyiga qo‘shishimiz mumkin, masalan, biz ikkinchi o‘ringa ob'ektni qo‘shamiz (ya'ni 1-indeksda, chunki raqamlash noldan boshlanadi):people.add(1, "Bob")

Usul size()sizga to‘plamdagi ob'ektlar sonini aniqlash imkonini beradi.

To‘plamda elementning mavjudligi contains yordamida tekshiriladi. Va remove yordamida o‘chirishimiz mumkin. Va xuddi qo‘shish bilan bo‘lgani kabi, biz ham ma'lum bir elementni people.remove("Tom"); yoki elementni indeks orqali olib tashlashimiz mumkin people.remove(0);- birinchi elementni olib tashlash.

Biz get(): usuli yordamida ma'lum bir elementni indeks bo‘yicha olishimiz String person = people.get(1);va set(): usuli yordamida elementni indeks bo‘yicha o‘rnatishimiz mumkin people.set(1, "Robert");

toArray() usuldan foydalanib, biz ro‘yxatni ob'ektlar massiviga aylantirishimiz mumkin.

ArrayList klassi Iterable interfeysini qo‘llaganligi sababli, biz ro‘yxatni har biri uchun siklda takrorlashimiz mumkin: for(String person : people).

ArrayList ob'ektiga qo‘shimcha ob'ektlarni erkin qo‘shishimiz mumkin bo‘lsa-da, massivdan farqli o‘laroq, ArrayList ob'ektlarni saqlash uchun yana massivdan foydalanadi. Odatiy bo‘lib, bu massiv 10 ta ob'ekt uchun mo‘ljallangan. Agar dastur davomida ko‘proq qo‘shilsa, butun miqdorni sig'dira oladigan yangi massiv yaratiladi. Bunday xotirani qayta taqsimlash unumdorlikni pasaytiradi. Shuning uchun, agar biz ro‘yxatimizda ma'lum miqdordagi elementlar, masalan, 25 dan ortiq bo‘lmasligini aniq bilsak, biz darhol ushbu raqamni konstruktorda yoki ArrayList<String> people = new ArrayList<String>(25);usuldan foydalanib aniq belgilashimiz mumkin ensureCapacity:people.ensureCapacity(25);

SortedSet

SortedSet interfeysi elementlarni tartiblangan shaklda saqlaydigan to‘plamlarni yaratish uchun mo‘ljallangan (o‘sish tartibida tartiblash). SortedSet Set interfeysini kengaytiradi, shuning uchun bu to‘plam yana faqat noyob qiymatlarni saqlaydi. SortedSet quyidagi usullarni taqdim etadi:

- E first() : to‘plamning birinchi elementini qaytaradi
- E last() : to‘plamning oxirgi elementini qaytaradi
- SortedSet<E> headSet(E end) : oxirgi elementgacha bo‘lgan asosiy o‘rnatishning barcha elementlarini o‘z ichiga olgan SortedSet obyektini qaytaradi
- SortedSet<E> subset(E start, E end): boshlanish va tugatish elementlari orasidagi asosiy to‘plamning barcha elementlarini o‘z ichiga olgan SortedSet obyektini qaytaradi.
- SortedSet<E> tailSet(E start): Boshlang‘ich elementdan boshlab asosiy to‘plamning barcha elementlarini o‘z ichiga olgan SortedSet obyektini qaytaradi.

NavigableSet

NavigableSet interfeysi SortedSet interfeysini kengaytiradi va ularning qiymatlari asosida elementlarni olish imkonini beradi. NavigableSet quyidagi usullarni belgilaydi:

- E shift(E obj): to‘plamdagi obj dan katta bo‘lgan eng kichik e elementni topadi (e >=obj). Agar shunday element topilsa, u natija sifatida qaytariladi. Aks holda null qaytariladi.
- E qavat(E obj): obj (e <=obj) dan kichik bo‘lgan to‘plamdagi eng katta elementni topadi. Agar shunday element topilsa, u natija sifatida qaytariladi. Aks holda null qaytariladi.
- E oliy(E obj): to‘plamdagi obj (e >obj) dan katta bo‘lgan eng kichik e elementni topadi. Agar shunday element topilsa, u natija sifatida qaytariladi. Aks holda null qaytariladi.
- E low(E obj): obj (e < obj) dan kichik bo‘lgan to‘plamdagi eng katta elementni topadi. Agar shunday element topilsa, u natija sifatida qaytariladi. Aks holda null qaytariladi.
- E pollFirst(): Birinchi elementni qaytaradi va uni to‘plamdan olib tashlaydi
- E pollLast(): oxirgi elementni qaytaradi va uni to‘plamdan olib tashlaydi

- NavigableSet<E> descendingSet(): Asosiy NavigableSetning barcha elementlarini teskari tartibda o‘z ichiga olgan NavigableSet obyektini qaytaradi.
- NavigableSet<E> headset(E upperBound, boolean incl): NavigableSet ob'ektini qaytaradi, unda asosiy NavigableSet ning barcha elementlarini o‘z ichiga oladi. Incl parametri, rost qiymatga o‘rnatilganda, chiqish to‘plamiga upperBound elementini kiritish imkonini beradi
- NavigableSet<E> tailSet(E lowBound, boolean incl): LowerBounddan boshlab asosiy NavigableSetning barcha elementlarini o‘z ichiga olgan NavigableSet obyektini qaytaradi. Incl parametri, rost qiymatga o‘rnatilganda, chiqish to‘plamiga lowBound elementini kiritish imkonini beradi
- NavigableSet<E> subset(E lowBound, mantiqiy lowIncl, E upperBound, mantiqiy highIncl): NavigableSet ob'ektini qaytaradi, unda birlamchi NavigableSet ning barcha elementlarini o‘z ichiga oladi lowBound dan upperBound.

Tree Set

TreeSet<E> umumiy sinfi daraxt ko‘rinishidagi ma'lumotlar strukturasini ifodalaydi, unda barcha ob'ektlar o‘sish tartibida tartiblangan holda saqlanadi.

TreeSet klassi quyidagi konstruktorlarni belgilaydi:

- TreeSet(): bo‘sh daraxt yaratadi
- TreeSet(Collection<? extends E> col): kol to‘plamining barcha elementlarini qo‘shadigan daraxt yaratadi
- TreeSet(SortedSet <E> set): saralangan to‘plamning barcha elementlarini qo‘shadigan daraxt yaratadi
- TreeSet(Comparator<? super E> comparator): Bo‘sh daraxt yaratadi, unda barcha qo‘shilgan elementlar keyinchalik taqqoslash tomonidan tartiblanadi.

TreeSet elementlarni kiritish va olib tashlashning barcha standart usullarini qo‘llab-quvvatlaydi:

```
import java.util.TreeSet;

public class Program{

    public static void main(String[] args) {
```

```
        TreeSet<String> states = new TreeSet<String>();

        // добавим в список ряд элементов
        states.add("Germany");
        states.add("France");
        states.add("Italy");
        states.add("Great Britain");

        System.out.printf("TreeSet contains %d elements \n",
states.size());

        // удаление элемента
        states.remove("Germany");
        for(String state : states){

            System.out.println(state);
        }
    }
 }
```

TreeSet NavigableSet interfeysini va u orqali SortedSetni amalga oshirganligi sababli, daraxt tuzilishiga turli usullarni qo‘llashimiz mumkin:

```
import java.util.*;

public class Program{

    public static void main(String[] args) {

        TreeSet<String> states = new TreeSet<String>();

        // добавим в список ряд элементов
        states.add("Germany");
        states.add("France");
        states.add("Italy");
        states.add("Spain");
        states.add("Great Britain");

        System.out.println(states.first()); // получим первый
- самый меньший элемент
        System.out.println(states.last()); // получим
последний - самый больший элемент
        // получим поднабор от одного элемента до другого
        SortedSet<String> set = states.subSet("Germany",
"Italy");
        System.out.println(set);
        // элемент из набора, который больше текущего
        String greater = states.higher("Germany");
```

```
        // элемент из набора, который меньше текущего
        String lower = states.lower("Germany");
        // возвращаем набор в обратном порядке
        NavigableSet<String> navSet = states.descendingSet();
        // возвращаем набор в котором все элементы меньше
текущего
        SortedSet<String> setLower=states.headSet("Germany");
        // возвращаем набор в котором все элементы больше
текущего
        SortedSet<String>
setGreater=states.tailSet("Germany");
        System.out.println(navSet);
        System.out.println(setLower);
        System.out.println(setGreater);
    }
}
```

Iteratorlar

Collection interfeysining asosiy usullaridan biri bu iterator() metodidir. U ya'ni Iterator interfeysini amalga oshiradigan ob'ektni qaytaradi.

Iterator interfeysi quyidagi ta'rifga ega:

```
public interface Iterator <E>{

    E next();
    boolean hasNext();
    void remove();
}
```

Interfeysning amalga oshirilishi keyingi elementni next() usulni chaqirish orqali olish mumkinligini nazarda tutadi. hasNext() usuldan foydalanib, siz keyingi element bor yoki yo‘qligini va to‘plamning oxiriga yetganligini bilib olishingiz mumkin. Va agar hali ham elementlar mavjud bo‘lsa, u true qiymatni qaytaradi. hasNext() usuli next() usuldan oldin chaqirilishi kerak, chunki to‘plam oxiriga yetganda next() usuli NoSuchElementException istisno qiladi Va remove() usuli oxirgi next() murojaat tomonidan qabul qilingan joriy elementni olib tashlaydi .

ArrayList to‘plamini takrorlash uchun iteratordan foydalanamiz:

```
import java.util.*;

public class Program {

    public static void main(String[] args) {
```

```
            ArrayList<String> states = new
ArrayList<String>();
            states.add("Germany");
            states.add("France");
            states.add("Italy");
            states.add("Spain");

            Iterator<String> iter = states.iterator();
            while(iter.hasNext()){

                System.out.println(iter.next());
            }
        }
     }
```

Iterator interfeysi cheklangan funksionallikni ta'minlaydi. Boshqa iterator, ListIterator interfeysi ancha kengroq usullar to‘plamini taqdim etadi. Ushbu iterator List interfeysni amalga oshiradigan sinflar tomonidan ishlatiladi, ya'ni LinkedList, ArrayList va hokazo.

ListIterator interfeysi Iterator interfeysini kengaytiradi va bir qator qo‘shimcha usullarni belgilaydi:

- void add(E obj) : keying next() murojaatda qaytariladigan elementdan oldin obj ob'ektini kiritadi
- boolean hasNext(): agar to‘plamda keyingi element bo‘lsa, true qaytaradi, aks holda false qaytaradi
- boolean hasPrevious(): agar to‘plamda oldingi element bo‘lsa, true qaytaradi, aks holda false qaytaradi
- E next(): joriy elementni qaytaradi va keyingi elementga o‘tadi, agar mavjud bo‘lmasa, NoSuchElementException istisno chiqariladi
- E oldingi(): joriy elementni qaytaradi va oldingisiga o‘tadi, agar mavjud bo‘lmasa, NoSuchElementException istisno chiqariladi
- int nextIndex(): Keyingi elementning indeksini qaytaradi. Agar bunday narsa bo‘lmasa, ro‘yxat hajmi qaytariladi
- int previousIndex(): Oldingi elementning indeksini qaytaradi. Agar bunday narsa bo‘lmasa, unda -1 qiymatini qaytariladi.
- void remove(): Joriy elementni ro‘yxatdan o‘chiradi. Shunday qilib, bu next() usuli yoki previous()usullardan keyin chaqirilishi kerak, aks holda IlligalStateException istisno chiqariladi.

- void set (E obj): ob'ekt ob'ektiga havola next()yoki previous() usullarni chaqirish orqali tanlangan joriy elementni tayinlaydi.

ListIterator-dan foydalanish:

```
import java.util.*;

public class Program {

    public static void main(String[] args) {

        ArrayList<String> states = new
ArrayList<String>();
        states.add("Germany");
        states.add("France");
        states.add("Italy");
        states.add("Spain");

        ListIterator<String> listIter =
states.listIterator();

        while(listIter.hasNext()){

            System.out.println(listIter.next());
        }
        // сейчас текущий элемент - Испания
        // изменим значение этого элемента
        listIter.set("Португалия");
        // пройдемся по элементам в обратном порядке
        while(listIter.hasPrevious()){

            System.out.println(listIter.previous());
        }
    }
}
```

Nazorat savollari.

1. To‘plam, to‘plam turlari va interfeysi haqida ma’lumot bering.
2. Mavhum sinflar tomonidan amalga oshiriladigan interfeyslarga misollar keltiring.
3. Javadagi keng ko‘lamli to‘plam sinflarini sanab o‘ting
4. To‘plam interfeyslariga misollar keltiring.
5. ArrayList klassi va List interfeysi haqida ma’lumot bering.

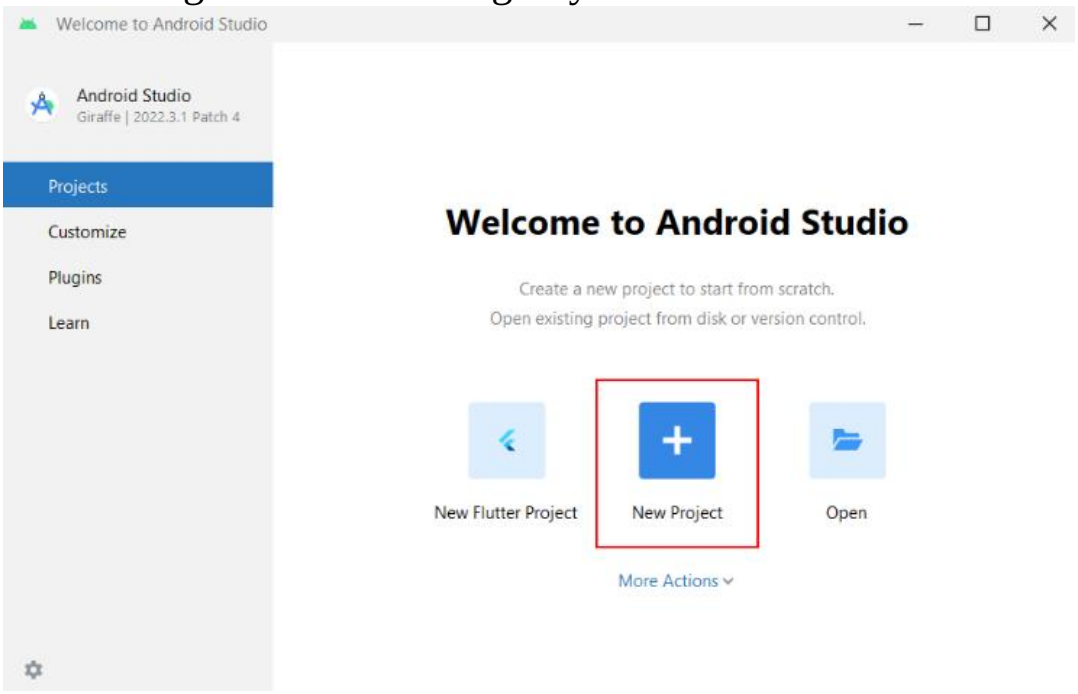
8. ANDROID ILOVADA FOYDALANUVCHINING GRAFIK INTERFEYSI. ELEMENTLARNI BOSHQARISH KOMPONOVKALARI. KOMPONOVKALAR TURLARI: LINEARLAYOUT, CONSTRAINTLAYOUT, RELATICELAYOUT, FRAMELAYOUT, TABLELAYOUT.

Reja:

8.1 Android ilovada foydalanuvchining grafik interfeysi. Elementlarni boshqarish komponovkalari.

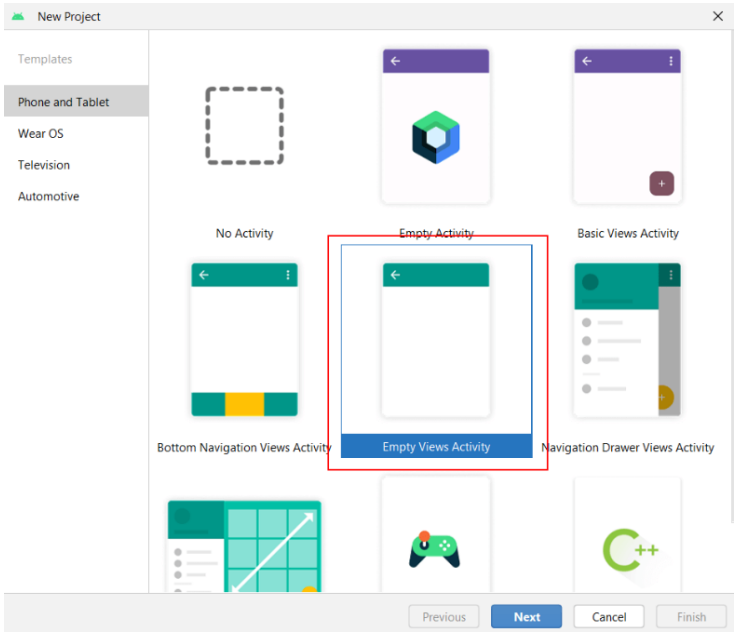
8.2 Komponovkalar turlari: LinearLayout, RelativeLayout, TableLayout, FrameLayout.

Android Studioni ishga tushurganimizda va loyiha yaratishimiz uchun boshlang'ich ekranda Yangi loyihani tanlashimiz kerak:

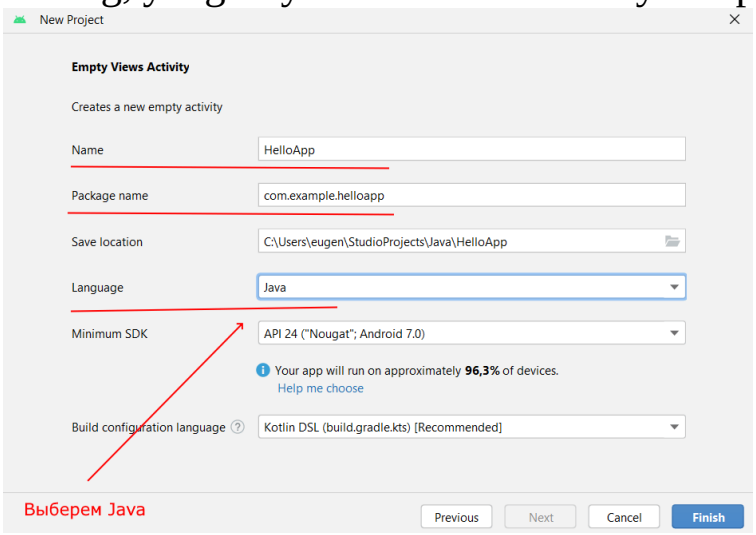


Loyihani yaratishda Android Studio birinchi navbatda loyiha shablonlarini tanlashni taklif qiladi:

Java-da dastur yaratish uchun ushbu ro‘yxatdan Bo‘sh ko‘rishlar faoliyati shablonini tanlang, u ishga tushirish uchun zarur bo‘lgan eng oddiy funksiyalarni ta'minlaydi.



Shundan so‘ng, yangi loyiha uchun sozlash oynasi paydo bo‘ladi:



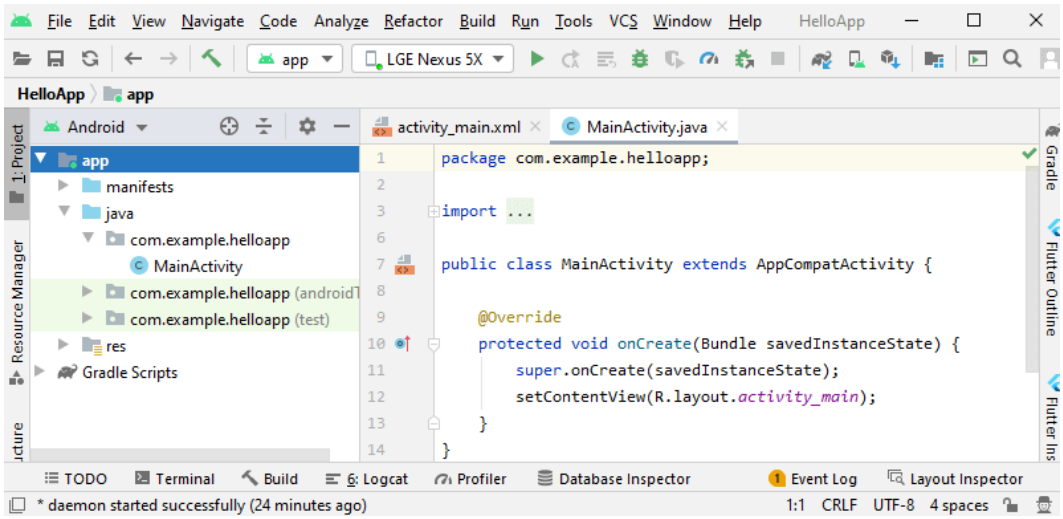
Yangi loyiha yaratish oynasida biz uning dastlabki sozlamalarini o‘rnatishimiz mumkin:

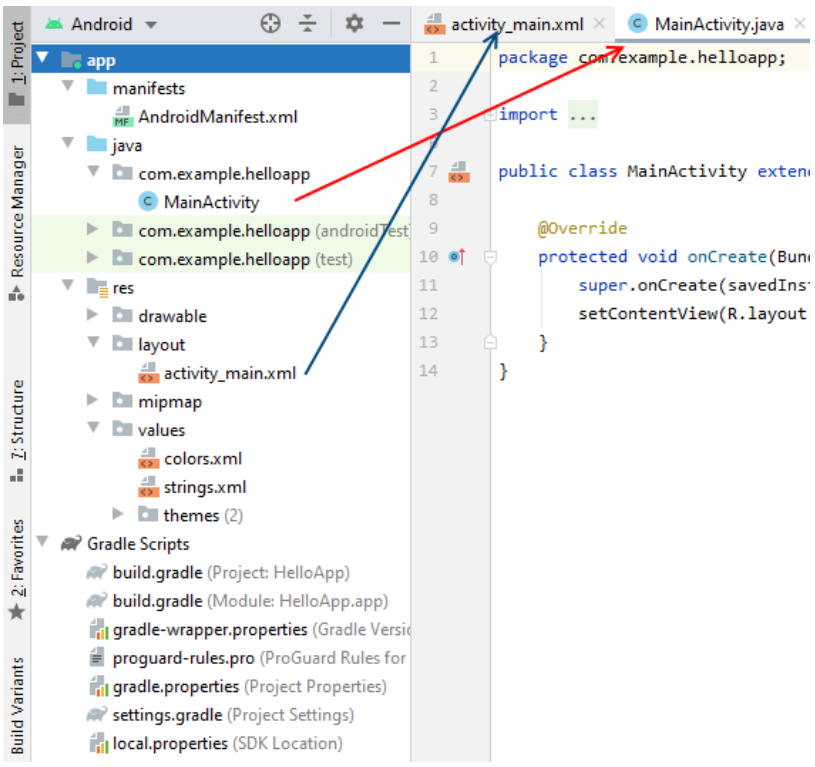
- Name maydoniga ilova nomini kiriting. Nom sifatida HelloApp nomini belgilaymiz
- "Packege name" maydonida asosiy dastur sinfi joylashgan paket nomi ko‘rsatilgan. Bunday holda, test loyihalari uchun bu qiymat juda muhim emas, shuning uchun biz com.example.helloapp ni o‘rnatamiz.
- Save location maydoni qattiq diskdagi loyiha fayllari joylashuvini belgilaydi. Siz standart qiymatni qoldirishingiz mumkin.
- Language maydonida biz Java-ni dasturlash tili sifatida ko‘rsatamiz.

- Minimal SDK maydoni minimal qo‘llab-quvvatlanadigan SDK versiyasini ko‘rsatadi. Keling, standart qiymatni qoldiramiz. Minimal versiya bizning ilovamizni ushbu versiyadan boshlab ishga tushirish mumkinligini anglatadi. Eski qurilmalarda ishlash mumkin bo‘lmaydi.

Shuni hisobga olish kerakki, SDK versiyasi qanchalik baland bo‘lsa, qo‘llab-quvvatlanadigan qurilmalar oralig'i shunchalik kichik bo‘ladi.

Keyin, Finish tugmasini bosganingizdan keyin Android Studio yangi loyiha yaratadi:





ConstraintLayout

ConstraintLayout sizga moslashuvchan va kengaytiriladigan vizual interfeyslarni yaratishga imkon beruvchi konteynerni taqdim etadi.

Elementni ConstraintLayout ichiga joylashtirish uchun siz cheklovlarni belgilashingiz kerak. Bir necha turdagi cheklovlar mavjud. Xususan, ma'lum bir elementga nisbatan pozitsiyani o‘rnatish 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 joylashtirilgan
- 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 chiziq boshqa elementning asosiy chizig'iga nisbatan joylashtirilgan
- layout_constraintStart_toEndOf: element boshqa element tugagan joyda boshlanadi
- layout_constraintStart_toStartOf: element boshqa element boshlangan joydan boshlanadi
- layout_constraintEnd_toStartOf: element boshqa element boshlangan joyda tugaydi
- layout_constraintEnd_toEndOf: element boshqa element tugagan joyda tugaydi

Elementning boshlanishi yoki oxiri nimani anglatishini so‘nggi to‘rtta xususiyat haqida chalkashliklar bo‘lishi mumkin. Ba'zi tillar

(masalan, arab yoki fors) o‘ng yo‘nalishiga ega, ya'ni belgilar Evropa tillarida bo‘lgani kabi chapdan o‘ngga emas, o‘ngdan chapga qarab harakatlanadi. Va joriy yo‘nalishga qarab - chap yoki o‘ng - elementning qaerdan boshlanishi va qaerda tugashi o‘zgaradi. Misol uchun, chap yo‘nalishda boshlanish chapda va oxiri o‘ngda bo‘ladi, shuning uchun atribut layout_constraintStart_toEndOf amalda bir xil bo‘ladi layout_constraintLeft_toRightOf. Va o‘ng tomon yo‘nalishi bilan - atributga layout_constraintRight_toLeftOf, chunki boshi o‘ngda va oxiri chapda.

Har bir cheklov elementning gorizontal yoki vertikal joylashishini belgilaydi. ConstraintLayout-da elementning o‘rnini aniqlash uchun kamida bitta gorizontal cheklov va bitta vertikal cheklovni belgilashingiz kerak.

Joylashtirish ContentLayout konteynerining chegaralariga nisbatan (bu holda cheklov parent qiymatiga ega) yoki ConstraintLayout ichidagi istalgan boshqa elementga nisbatan amalga oshirilishi mumkin, keyin bu elementning identifikatori cheklov qiymati sifatida belgilanadi.

```
     <?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>
```

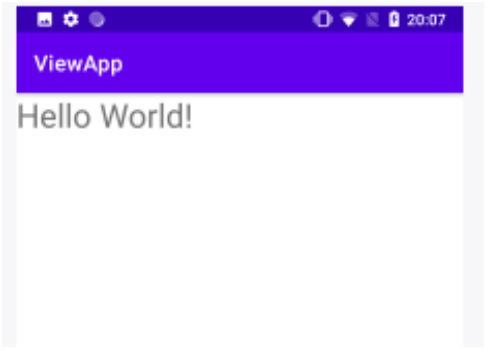
Yuqoridagi holatda, TextView elementi ikkita cheklovga ega: biri gorizontal chiziq bo‘ylab (app:layout_constraintLeft_toLeftOf="parent"), ikkinchisi vertikal chiziq bo‘ylab (app:layout_constraintTop_toTopOf="parent"). Ikkala

cheklovlar ConstraintLayout konteyneriga nisbatan o‘rnatiladi, shuning uchun ular ConstraintLayout bo‘lgan parent-ning qiymatini oladi.

Cheklov app:layout_constraintLeft_toLeftOf="parent"TextView chap chegarasini konteynerning chap chegarasiga o‘rnatadi.

Cheklov app:layout_constraintTop_toTopOf="parent"TextView ning yuqori chegarasini konteynerning yuqori chegarasiga o‘rnatadi.

Natijada, TextView konteynerning yuqori chap burchagida joylashgan bo‘ladi.

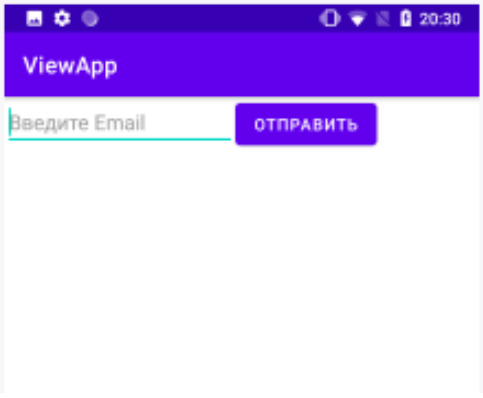


Agar siz boshqa elementga nisbatan cheklov o‘rnatishingiz kerak bo‘lsa, ushbu elementning identifikatorini ko‘rsatishingiz kerak:

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

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="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/editT
ext"

app:layout_constraintTop_toTopOf="parent" />

    </androidx.constraintlayout.widget.Constra
intLayout>
```

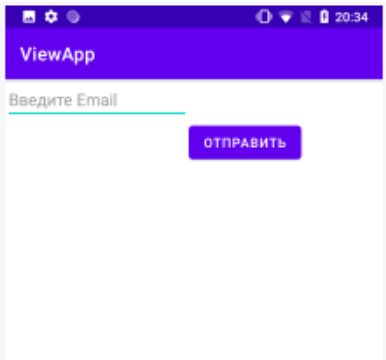
Yuqorida EditText kiritish matn maydoni o‘zining asosiy ConstraintLayout konteyneriga nisbatan ikkita cheklovga ega bo‘ladi, shuning uchun cheklovlar ota-onaga o‘rnatiladi va EditTextning o‘zi konteynerning chap va yuqori chetiga tekislanadi. Tugmaning yuqori chegarasi ham idishning yuqori chegarasiga to‘g'ri keladi. Lekin tugmachaning chap chegarasi EditTextning o‘ng chegarasi bilan tekislanadi. Buning uchun id EditText atribut qiymati sifatida uzatiladi:



biz kerakli joylashishni aniqlash uchun atributlarning turli kombinatsiyalarini yaratishimiz mumkin. Masalan, tugma kodini o‘zgartiramiz:

```
          <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_toBottomOf="@id/editText"/>
```

Bunday holda, tugmaning yuqori chegarasi EditTextning pastki chegarasi bilan tekislanadi



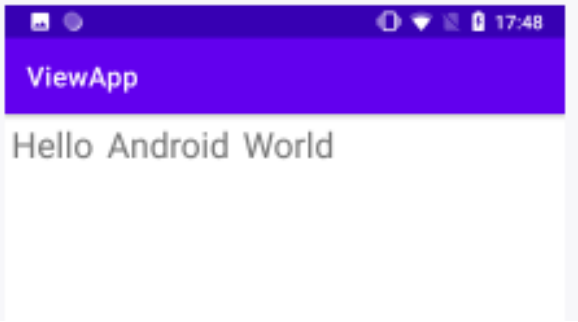
LinearLayout

LinearLayout konteyneri ViewGroup barcha asosiy elementlarni bir yo‘nalishda joylashtirgan eng oddiy konteyner ob'ektini ifodalaydi: gorizontal yoki vertikal. Barcha elementlar birin-ketin joylashgan. Tartibning yo‘nalishi android:orientation atributi yordamida aniqlanadi.

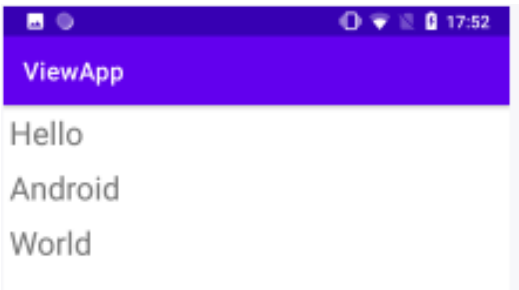
Agar, masalan, tartib yo‘nalishi vertikal bo‘lsa (android:orientation="vertical"), u holda barcha elementlar ustun shaklda joylashtiriladi - har bir satrda bitta element bo‘ladi. Agar yo‘nalish gorizontal bo‘lsa (android:orientation="horizontal"), u holda elementlar bir qatorda joylashadi.

```
     <?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>
```



Agar LinearLayout uchun android:orientation="vertical" atributini belgilagan bo‘lsak , elementlar vertikal ravishda joylashtiriladi:



Elementning og'irligi

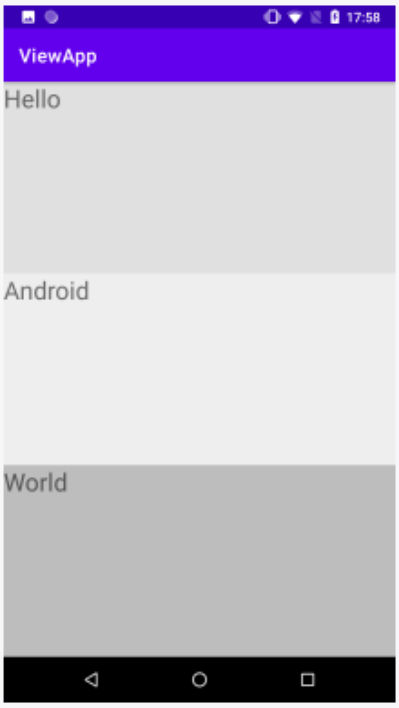
LinearLayout android:layout_weight atributi orqali uzatiladigan element og'irligi deb nomlangan xususiyatni qo‘llab-quvvatlaydi . Bu xususiyat boshqa ob'ektlarga nisbatan konteynerning qolgan bo‘sh joyining qancha qismini egallashini ko‘rsatadigan qiymatni oladi. Misol uchun, agar bir elementning xossa android:layout_weightqiymati 2, boshqasi esa 1 ga teng bo‘lsa, u holda ular 3 ga teng bo‘ladi, shuning uchun birinchi element qolgan joyning 2/3 qismini, ikkinchisi esa 1/ 3.

Agar barcha elementlar qiymatga ega bo‘lsa android:layout_weight="1", unda bu elementlarning barchasi idishning butun maydoniga teng taqsimlanadi:

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

Bunday holda, LinearLayout vertikal yo‘nalishga ega, shuning uchun barcha elementlar yuqoridan pastgacha joylashadi. Har uch elementning qiymati ga ega android:layout_weight="1", shuning uchun

barcha elementlarning og'irliklari yig'indisi 3 ga teng bo‘ladi va har bir element LinearLayoutdagi bo‘sh joyning uchdan bir qismini oladi:



Bundan tashqari, bizda vertikal stek borligi sababli, biz xususiyatni 0dplayout_height ga o‘rnatishimiz kerak. Agar LinearLayout gorizontal yo‘nalishga ega bo‘lsa, u holda xususiyat 0dp layout_width ga o‘rnatilishi kerak bo‘ladi.

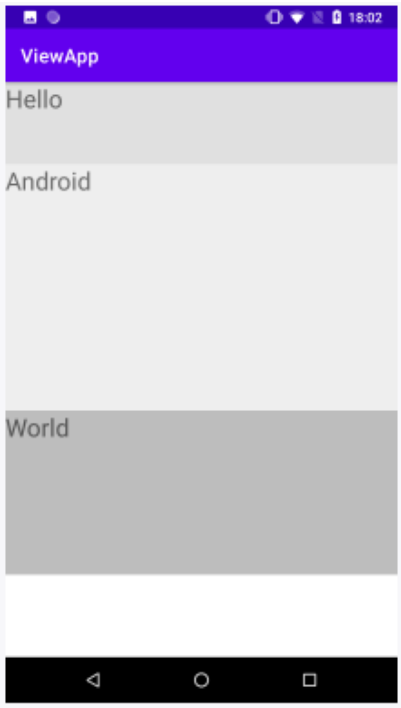
Yana bir atribut android:weightSum barcha elementlarning og'irliklari yig'indisini belgilash imkonini beradi. Masalan:

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

```
            android:background="#eeeeee"<font></font>
            android:text="Android"<font></font>
            android:layout_weight="3"<font></font>
            android:textSize="26sp" /><font></font>
        <TextView<font></font>
            android:layout_width="match_parent"<font></font>
            android:layout_height="0dp"<font></font>
            android:text="World"<font></font>
            android:background="#bdbdbd"<font></font>
            android:layout_weight="2"<font></font>
            android:textSize="26sp" /><font></font>
    </LinearLayout><font></font>
```

LinearLayout bu erda 7 ga teng og'irliklar yig'indisini o‘rnatadi. Ya'ni butun vertikal bo‘shliq shartli ravishda etti teng qismga bo‘linadi.

Birinchi TextView 1 og'irligiga ega, ya'ni u ushbu etti qismdan faqat bittasini oladi. Ikkinchi TextView ning og'irligi 3 ga teng, ya'ni u etti qismdan uchtasini oladi. Uchinchisining vazni 2 ga teng. Jami 6. Lekin LinearLayout vaznni 7 ga o‘rnatganligi sababli, bir qism barcha elementlardan xoli bo‘ladi.



RelativeLayout

RelativeLayout ViewGroup boshqa ichki tartib elementlarining joylashuviga yoki RelativeLayout tartibining o‘ziga nisbatan kichik elementlarni joylashtiradigan ob'ektni ifodalaydi. Nisbatan joylashishni aniqlashdan foydalanib, biz elementni o‘ngga, markazga yoki

konteyner taqdim etgan boshqa narsalarni joylashtirishimiz mumkin. Xml faylida elementni o‘rnatish uchun biz quyidagi atributlarni qo‘llashimiz mumkin:

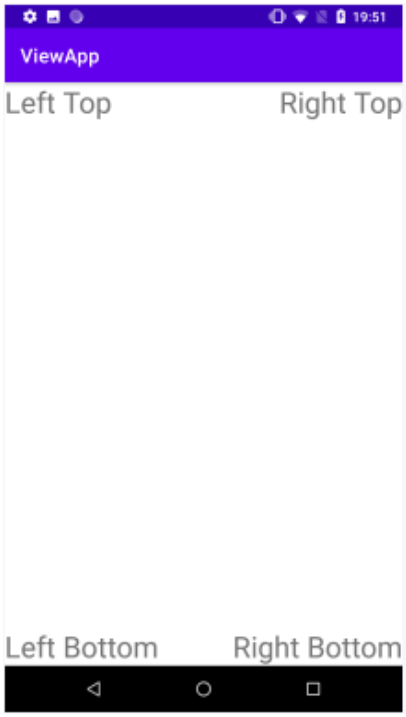
- android:layout_above: Elementni belgilangan identifikatorga ega element ustida joylashtiradi
- android:layout_below: Elementni belgilangan identifikatorli element ostida joylashtiradi
- android:layout_toLeftOf: belgilangan identifikatorga ega elementning chap tomonida joylashgan
- android:layout_toRightOf: belgilangan identifikatorga ega elementning o‘ng tomonida joylashgan
- android:layout_alignBottom: Elementni belgilangan Id bilan boshqa elementning pastki chegarasiga tenglashtiradi
- android:layout_alignLeft: Elementni belgilangan identifikator bilan boshqa elementning chap chetiga tekislaydi
- android:layout_alignRight: Elementni belgilangan identifikator bilan boshqa elementning o‘ng chetiga tekislaydi
- android:layout_alignStart: Elementni belgilangan identifikatorga ega boshqa element boshlanadigan chiziq bo‘ylab tekislaydi
- android:layout_alignEnd: Elementni belgilangan identifikatorga ega boshqa element tugaydigan qatorga tekislaydi
- android:layout_alignTop: Elementni belgilangan identifikator bilan boshqa elementning yuqori chegarasiga tekislaydi
- android:layout_alignBaseline: Elementning asosiy chizig‘ini belgilangan identifikatorli boshqa elementning asosiy chizig‘i bilan tekislaydi
- android:layout_alignParentBottom: agar atribut rost bo‘lsa, element konteynerning pastki chegarasiga to‘g'ri keladi
- android:layout_alignParentRight: agar atribut rost bo‘lsa, element konteynerning o‘ng chetiga tekislanadi
- android:layout_alignParentLeft: agar atribut rost bo‘lsa, element konteynerning chap chetiga tekislanadi.
- android:layout_alignParentStart: agar atribut rost bo‘lsa, element konteynerning boshlang'ich chetiga tekislanadi (agar matn chapga, chap chetiga yo‘naltirilgan bo‘lsa)

- android:layout_alignParentEnd: agar atribut rost bo‘lsa, element konteynerning oxirgi chetiga hizalanadi (matnni chap tomonga yo‘naltirish uchun, o‘ng chetiga)
- android:layout_alignParentTop: agar atribut rost bo‘lsa, element konteynerning yuqori chegarasiga to‘g'ri keladi
- android:layout_centerInParent: agar atribut to‘g'ri bo‘lsa, element asosiy konteynerda markazlashtirilgan
- android:layout_centerHorizontal: agar rost bo‘lsa, elementni gorizontal markazlashtirilgan holda tekislaydi
- android:layout_centerVertical: rost bo‘lsa, elementni vertikal markazlashtirilgan holda 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"
```

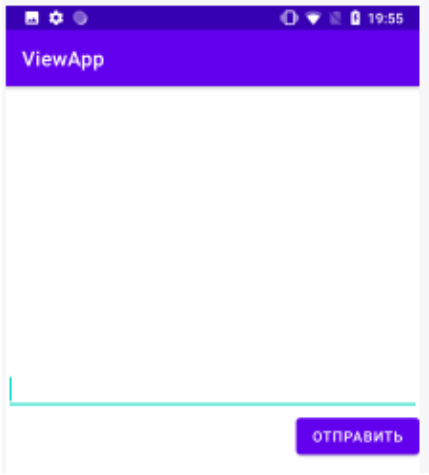
```
            android:layout_alignParentBottom="true" />
    </RelativeLayout>
```



Boshqa elementga nisbatan joylashtirish uchun biz ushbu elementning identifikatorini ko‘rsatishimiz kerak. Shunday qilib, RelativeLayout-ga matn maydoni va tugmani joylashtiramiz:

```
    <?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 maydoni RelativeLayout markazida joylashgan va tugma EditText ostida joylashtiriladi va uning o‘ng chegarasiga tenglashtiriladi:

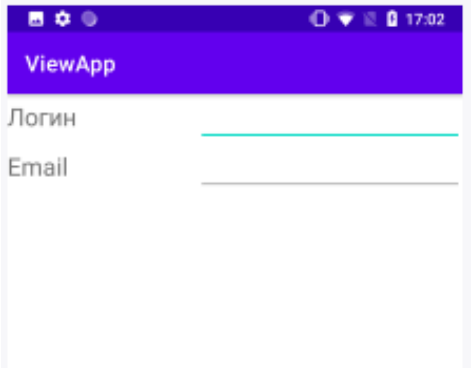


TableLayout

TableLayout konteyner tuzilmalari ustunlar va satrlar bo‘yicha jadvalga o‘xshash tarzda boshqaradi. Activity_main.xml faylida ikkita satr va ikkita ustundan iborat TableLayout elementini aniqlaymiz :

```
     <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="Логин"
                 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="Email"
                 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>
```



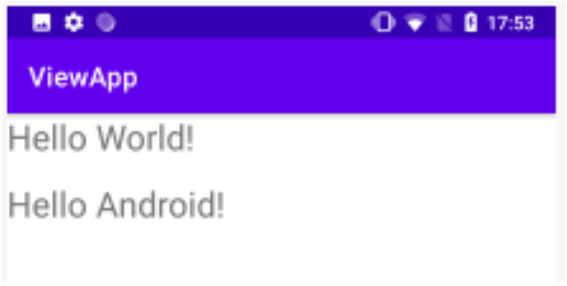
FrameLayout

FrameLayout konteyneri ekranda unda joylashtirilgan bitta vizual elementni ko‘rsatish uchun mo‘ljallangan. Agar biz bir nechta elementlarni joylashtirsak, ular bir-birining ustiga chiqadi. Biroq, FrameLayout-da bir nechta elementlarga ega bo‘lish ham mumkin.

Aytaylik, biz ikkita TextView elementini FrameLayout-ga joylashtiramiz:

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

xmlns:android="http://schemas.android.com/apk/res/android"<fo
nt></font>
        android:layout_width="match_parent"<font></font>
        android:layout_height="match_parent"><font></font>
    <font></font>
        <TextView<font></font>
            android:layout_width="wrap_content"<font></font>

android:layout_height="wrap_content"<font></font>
            android:text="Hello World!"<font></font>
            android:textSize="26sp"/><font></font>
        <TextView<font></font>
            android:layout_width="wrap_content"<font></font>

android:layout_height="wrap_content"<font></font>
            android:text="Hello Android!"<font></font>
            android:textSize="26sp"<font></font>
            android:layout_marginTop="50dp"/><font></font>
    <font></font>
    </FrameLayout><font></font>
```

Bu erda ikkala element bir joyda - FrameLayout konteynerining yuqori chap burchagida joylashgan va bir-birining ustiga tushmaslik uchun, bu holda ikkinchi TextView 50 birlik yuqori chetiga ega qilib o‘rnatiladi.

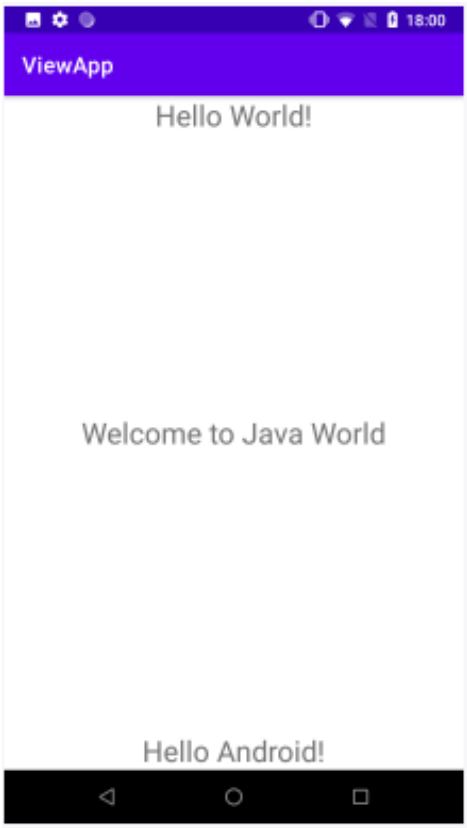


Ko‘pincha FrameLayout aylantirishni ta'minlaydigan ScrollView kabi olingan konteynerlarni yaratish uchun ishlatiladi.

FrameLayout-ga joylashtirilgan boshqaruv elementlari android:layout_gravity atributi yordamida o‘z joylashuvini o‘rnatishi mumkin :

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

xmlns:android="http://schemas.android.com/apk/res/android"<fo
nt></font>
         android:layout_width="match_parent"<font></font>
         android:layout_height="match_parent"><font></font>
     <font></font>
         <TextView<font></font>
             android:layout_width="wrap_content"<font></font>

android:layout_height="wrap_content"<font></font>
             android:text="Hello World!"<font></font>
             android:textSize="26sp"<font></font>
             android:layout_gravity="center_horizontal"
/><font></font>
         <TextView<font></font>
             android:layout_width="wrap_content"<font></font>

android:layout_height="wrap_content"<font></font>
             android:text="Welcome to Java
World"<font></font>
             android:textSize="26sp"<font></font>
             android:layout_gravity="center"/><font></font>
         <TextView<font></font>
             android:layout_width="wrap_content"<font></font>
```

```
android:layout_height="wrap_content"<font></font>
             android:text="Hello Android!"<font></font>
             android:textSize="26sp"<font></font>

android:layout_gravity="bottom|center_horizontal"/><font></fo
nt>
     <font></font>
     </FrameLayout><font></font>
```



Nazorat savollari.

1. Android Studioning grafik interfeysi haqida ma’lumot bering.
2. ConstraintLayout nima va uning fuksiyalari?
3. Ma'lum bir elementga nisbatan pozitsiyani o‘rnatish uchun qanday cheklovlar qo‘llaniladi?
4. LinearLayout konteyneri haqida ma’lumot bering.
5. Element og'irligi nima va u qanday vazifani bajaradi?
6. RelativeLayout tartibi nimani ifodalaydi?
7. TableLayout elementining vazifasi nimadan iborat?
8. FrameLayout elementining vazifasi nimadan iborat?

9. BOSHQARUV ELEMENTLARIDAN FOYDALANISH. BOSHQARUVNING MATNLI ELEMENTLARI. MATNLI MAYDON KOMPONENTALARI BILAN ISHLASH.

Reja:

9.1 Boshqaruv elementlaridan foydalanish.

9.2 Matnli maydon komponentalari bilan ishlash.

TextView

TextView elementi matnni ekranda oddiy ko‘rsatish uchun mo‘ljallangan. U shunchaki matnni tahrirlash imkoniyatisiz ko‘rsatadi. Uning asosiy atributlaridan ba'zilarini keltirib o‘tamiz:

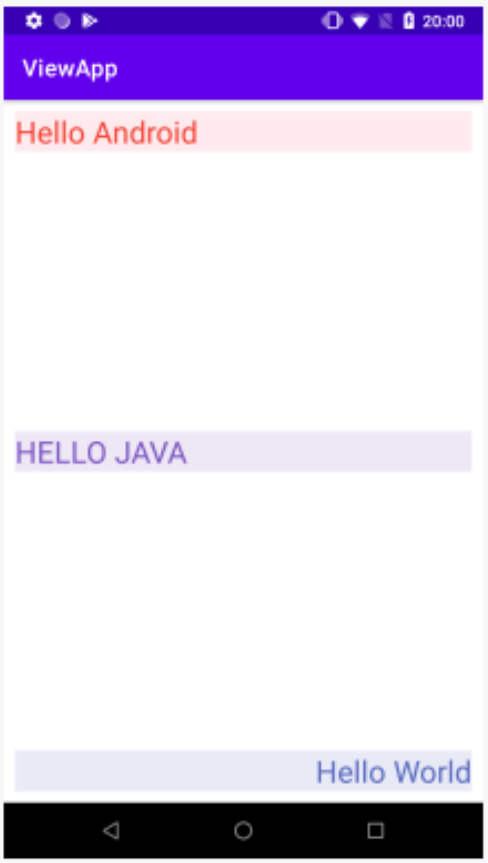
- ✓ android:text: element matnini o‘rnatadi
- ✓ android:textSize: matn o‘lchamini belgilaydi, o‘lchamni belgilash uchun ishlatiladigan birliklar sp, dp, px.
- ✓ android:background: Elementning fon rangini o‘n oltilik sanoq tizimida rang yoki rang manbai sifatida o‘rnatadi
- ✓ android:textColor: matn rangini o‘rnatadi
- ✓ android:textAllCaps: true qiymatiga o‘rnatilganda, matndagi barcha belgilarni bosh harf bilan yozadi
- ✓ android:textDirection: Matn yo‘nalishini o‘rnatadi. Standart yo‘nalish chapdan o‘ngga, lekin rtl qiymati yo‘nalishni o‘ngdan chapga o‘rnatish uchun ishlatilishi mumkin
- ✓ android:textAlignment: Matnni tekislashni o‘rnatadi. Quyidagi qiymatlarni qabul qilishi mumkin:
- center: markazga moslashtirish
- textStart: chap
- textEnd: o‘ng chetida
- viewStart: matn chapdan o‘ngga yo‘naltirilganda tekislash chapga, o‘ngdan chapga yo‘naltirilganda esa o‘ngga tekislanadi.
- viewEnd: matn chapdan o‘ngga yo‘naltirilganda tekislash o‘ngga, o‘ngdan chapga yo‘naltirilganda esa chapga tekislanadi.
- ✓ android:fontFamily: Shrift turini o‘rnatadi. Quyidagi qiymatlarni qabul qilishi mumkin:
- monospace
- serif

- serif-monfazo
- sans serif
- sans-serif-kondensatsiyalangan
- sans-serif-smallcaps
- sans-serif-nur
- casual
- 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>
```



EditText

EditText elementi TextView ning pastki sinfidir. Shuningdek, u matn maydonini taqdim etadi, ammo endi matnni kiritish va tahrirlash imkoniyati mavjud. Shunday qilib, EditText-da biz TextView-dagi kabi barcha funktsiyalardan foydalanishimiz mumkin.

TextView mavzusida muhokama qilinmagan atributlardan android: hint atributini ta'kidlash kerak. U EditText elementi bo‘sh bo‘lsa, maslahat sifatida ko‘rsatiladigan matnni belgilash imkonini

beradi. Bundan tashqari, biz kiritish uchun klaviaturani belgilash imkonini beruvchi android:inputType atributidan foydalanishimiz mumkin . Xususan, uning ma'nolari orasida quyidagilarni ajratib ko‘rsatish mumkin:

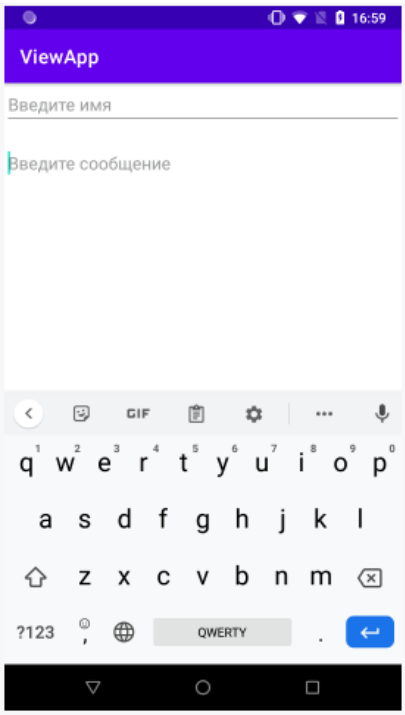
- text: bitta qatorli matn kiritish uchun oddiy klaviatura
- textMultiLine: ko‘p qatorli matn maydoni
- textEmailAddress: @ belgisi bo‘lgan oddiy klaviatura elektron pochta xabarlarini kiritishga qaratilgan
- textUri: / belgisi bo‘lgan oddiy klaviatura Internet manzillarini kiritish uchun mo‘ljallangan
- textPassword: Parolni kiritish uchun
- textCapWords: Yozayotganda, kiritilgan birinchi so‘z belgisi bosh harfni ifodalaydi, qolganlari kichik harfdir.
- number: raqamli klaviatura
- phone: Telefon uslubidagi klaviatura
- date: Sana klaviaturasi
- time: Vaqtni kiritish uchun klaviatura
- datetime: Sana va vaqtni kiritish uchun klaviatura

EditText-ni quyida 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">

        <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 yerdagi birinchi maydon oddiy bir qatorli maydon, ikkinchisi esa ko‘p qatorli maydondir. Ikkinchi maydondagi matn tepaga tekislanganligini ta'minlash uchun atribut android:gravity="top" qo‘shimcha ravishda o‘rnatiladi.



Button

Ko‘p ishlatiladigan elementlardan biri bu tugmalar bo‘lib, ular android.widget. Button klassi bilan ifodalanadi . Tugmalarning asosiy xususiyati bosish orqali foydalanuvchi bilan muloqot qilish qobiliyatidir.

Tugmalar uchun o‘rnatilishi mumkin bo‘lgan ba'zi asosiy atributlar:

- text: Tugmadagi matnni o‘rnatadi
- textColor: Tugmadagi matn rangini o‘rnatadi
- background: Tugmaning fon rangini o‘rnatadi
- textAllCaps: Qachon true bo‘lsa, matnni bosh harfga o‘rnatadi. Standart qiymat true

- onClick: Tugmani bosish hodisasini 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="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="Ввод"
            android:onClick="sendMessage"

app:layout_constraintTop_toBottomOf="@+id/editText"
            app:layout_constraintLeft_toLeftOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
```

android:onClick Atributdan foydalanib siz java kodida tugmani bosish bilan sodir bo‘ladigan hodisa funksiyasini o‘rnatishingiz mumkin. Shunday qilib, yuqoridagi misolda bu sendMessage. Endi MainActivity kodiga o‘tamiz va unga 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);
        }
        // Обработка нажатия кнопки
        public void sendMessage(View view) {
            TextView textView = findViewById(R.id.textView);
            EditText editText = findViewById(R.id.editText);
            textView.setText("Добро пожаловать, " +
editText.getText());
        }
    }
```

OnClick bilan ishlash usulini yaratishda quyidagi fikrlarni hisobga oling:

- Usul public modifikator bilan e'lon qilinishi kerak
- Void Qiymatni qaytarish kerak
- View Ob'ektni parametr sifatida oling . Ushbu View obyekti bosilgan tugmani ifodalaydi

Tugmani dasturiy jihatdan yaratishda biz uning View.OnClickListener uchun bosish hodisasini belgilashimiz va uning onClick usulini bosishni qayta ishlash uchun ham foydalanishimiz mumkin:

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

Kalkulyator ilovasi

Ba'zi tartib asoslarini va TextView, EditText va Button kabi elementlarni bilgan holda, siz oddiy dastur yaratishingiz mumkin. Bunday holda, biz oddiy kalkulyator qilamiz.

Buning uchun yangi loyiha yaratamiz va activity_main.xml faylida quyidagi ko‘rinishlarni belgilaymiz:

```
    <?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">
        <!-- поле результата -->
        <TextView
```

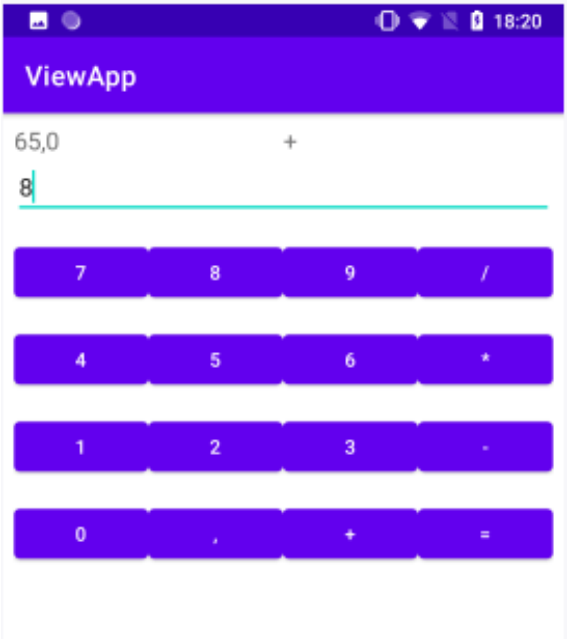
```
            android:id="@+id/resultField"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            app:layout_constraintHorizontal_weight="1"
            android:textSize="18sp"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"

app:layout_constraintRight_toLeftOf="@+id/operationField"/>
        <!-- поле знака операции -->
        <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"
            />
        <!-- поле ввода чисел -->
        <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:id="@+id/n7" />
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
```

```
                android:text="8"
                android:id="@+id/n8"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="9"
                android:id="@+id/n9"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="/"
                android:id="@+id/div"/>
        </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:id="@+id/n4"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="5"
                android:id="@+id/n5"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="6"
                android:id="@+id/n6"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="*"
                android:id="@+id/mul"/>
        </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:id="@+id/n1"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="2"
                android:tag="num"
                android:id="@+id/n2"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="3"
                android:id="@+id/n3"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="-"
                android:id="@+id/sub" />
        </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:id="@+id/n0"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
```

```
                android:layout_height="wrap_content"
                android:text=","
                android:id="@+id/comma"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="+"
                android:id="@+id/add"/>
            <Button
                android:layout_weight="1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="="
                android:id="@+id/eq"/>
        </LinearLayout>

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

Natijada, quyidagi ko‘rinishni hosil qilamiz:



Tartibning ildiz konteyneri ConstraintLayout elementi bilan ifodalanadi. Yuqori qismida u ikkita TextView matn maydonini belgilaydi: biri hisob-kitoblar natijasini ko‘rsatish uchun va ikkinchisi operatsiyaning joriy belgisini ko‘rsatish uchun.

Keyin raqamlarni kiritish uchun mo‘ljallangan EditText elementi keladi.

Va keyin gorizontal qatorli tugmalar bilan to‘rtta LinearLayout elementi mavjud. Barcha tugmalar konteyner ichida teng joy

egallashini ta'minlash uchun ular uchun android:layout_weight="1"va atributlari o‘rnatiladi android:layout_width="0dp".

Har bir tugma identifikatorga ega, shuning uchun kod unga ma'lum bir bosish moslamasini biriktirishi mumkin.

Endi MainActivity sinfini o‘tamiz:

```
     import androidx.annotation.NonNull;
     import androidx.appcompat.app.AppCompatActivity;

     import android.os.Bundle;
     import android.widget.EditText;
     import android.widget.TextView;

     public class MainActivity extends AppCompatActivity {

         TextView resultField; // текстовое поле для вывода
результата
         EditText numberField;   // поле для ввода числа
         TextView operationField;    // текстовое поле для
вывода знака операции
         Double operand = null;  // операнд операции
         String lastOperation = "="; // последняя операция
         @Override
         protected void onCreate(Bundle savedInstanceState) {
             super.onCreate(savedInstanceState);
             setContentView(R.layout.activity_main);
             // получаем все поля по id из activity_main.xml
             resultField = findViewById(R.id.resultField);
             numberField = findViewById(R.id.numberField);
             operationField =
findViewById(R.id.operationField);


findViewById(R.id.add).setOnClickListener((view)-
>onOperationClick("+"));

findViewById(R.id.sub).setOnClickListener((view)-
>onOperationClick("-"));

findViewById(R.id.mul).setOnClickListener((view)-
>onOperationClick("*"));

findViewById(R.id.div).setOnClickListener((view)-
>onOperationClick("/"));
             findViewById(R.id.eq).setOnClickListener((view)-
>onOperationClick("="));

             findViewById(R.id.n0).setOnClickListener((view)-
>onNumberClick("0"));
```

```
            findViewById(R.id.n1).setOnClickListener((view)-
>onNumberClick("1"));
            findViewById(R.id.n2).setOnClickListener((view)-
>onNumberClick("2"));
            findViewById(R.id.n3).setOnClickListener((view)-
>onNumberClick("3"));
            findViewById(R.id.n4).setOnClickListener((view)-
>onNumberClick("4"));
            findViewById(R.id.n5).setOnClickListener((view)-
>onNumberClick("5"));
            findViewById(R.id.n6).setOnClickListener((view)-
>onNumberClick("6"));
            findViewById(R.id.n7).setOnClickListener((view)-
>onNumberClick("7"));
            findViewById(R.id.n8).setOnClickListener((view)-
>onNumberClick("8"));
            findViewById(R.id.n9).setOnClickListener((view)-
>onNumberClick("9"));

findViewById(R.id.comma).setOnClickListener((view)-
>onNumberClick(","));
        }
        // сохранение состояния
        @Override
        protected void onSaveInstanceState(Bundle outState)
{
            outState.putString("OPERATION", lastOperation);
            if(operand!=null)
                outState.putDouble("OPERAND", operand);
            super.onSaveInstanceState(outState);
        }
        // получение ранее сохраненного состояния
        @Override
        protected void onRestoreInstanceState(@NonNull
Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);
            lastOperation =
savedInstanceState.getString("OPERATION");
            operand=
savedInstanceState.getDouble("OPERAND");
            resultField.setText(operand.toString());
            operationField.setText(lastOperation);
        }
        // обработка нажатия на числовую кнопку
        public void onNumberClick(String number){
            numberField.append(number);
            if(lastOperation.equals("=") && operand!=null){
                operand = null;
            }
        }
```

```
        // обработка нажатия на кнопку операции
        public void onOperationClick(String op){

            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){

            // если операнд ранее не был установлен (при
вводе самой первой операции)
            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()usulda biz activity_main.xml dan barcha maydonlarni olamiz:

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

Keyinchalik, har bir tugma uchun biz ma'lum bir bosish moslamasini tayinlaymiz - tugma turiga qarab, bu onNumberClick bosilgan raqam uzatiladigan usul yoki onOperationClick operatsiya belgisi uzatiladi.

Amaliyot natijasi Double turini ifodalovchi operand o‘zgaruvchisiga, operatsiya belgisi esa lastOperation o‘zgaruvchisiga tushadi:

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

Portretdan landshaft yo‘nalishiga yoki aksincha o‘tishda biz kiritilgan barcha ma'lumotlarni yo‘qotishimiz mumkin, chunki ularni yo‘qotmaslik uchun biz uni onSaveInstanceState() usulda saqlaymiz va uni onRestoreInstanceState(). usulda qaytarib olamiz.

Raqamli tugmani bosganingizda, onNumberClick usul chaqiriladi, unda biz kiritilgan raqam yoki vergulni raqam maydonidagi matnga qo‘shamiz:

```
numberField.append(number);

if(lastOperation.equals("=") && operand!=null){
    operand = null;
}
```

Bundan tashqari, agar oxirgi operatsiya natijani olish bo‘lsa ("teng" belgisi), biz operand o‘zgaruvchisini qayta o‘rnatamiz.

onOperationClick Usul operatsiya belgisi bilan tugmani bosishni qayta ishlatadi:

```
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 erda biz avval kiritilgan raqamni va kiritilgan operatsiyani olamiz va ularni performOperation()usulga o‘tamiz . Usulga shunchaki satr emas, balki Double raqam o‘tkazilganligi sababli, satrni raqamga aylantirishimiz kerak. Raqamli bo‘lmagan belgilarni nazariy jihatdan kiritish mumkin bo‘lganligi sababli, try...catch konstruktsiyasi konvertatsiya paytida yuzaga kelishi mumkin bo‘lgan istisnoni ushlash uchun ishlatiladi.

Bundan tashqari, Java-dagi Double-da o‘nlik ajratuvchi nuqta bo‘lgani uchun, vergulni nuqta bilan almashtirishimiz kerak, chunki biz vergulni ajratuvchi sifatida ishlatamiz deb taxmin qilinadi.

performOperation() Ushbu usulda biz haqiqiy operatsiyani bajaramiz. Birinchi operatsiyani kiritishda, operand hali o‘rnatilmagan bo‘lsa, biz shunchaki operandni o‘rnatamiz:

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

Ikkinchi va keyingi operatsiyalarni kiritishda biz oldingi operatsiyani qo‘llaymiz, uning belgisi lastOperation o‘zgaruvchisida operand operandiga va raqamli maydonga kiritilgan ikkinchi raqamga qo‘llaniladi. Amaliyotning natijasi operand o‘zgaruvchisida saqlanadi.

Nazorat savollari.

1. TextView elementi haqida ma’lumot bering.
2. EditText elementini izohlang.
3. Kalkulyator ilovasini yaratish usulini tushuntiring.

10. TOAST BILDIRISHNOMA OYNASI. RADIOBUTTON, RADIOGROUP, CHECKBOX, TOGGLEBUTTON, SNACKBAR.

Reja:

10.1 Toast bildirishnoma oynasi. Tugmalar va flajoklar

10.2 Button, RadioButton, RadioGroup, CheckBox, ToggleButton, Snackbar.

Toast bildirishnoma oynasi.

Android-da oddiy bildirishnomalarni yaratish uchun Toast sinfidan foydalaniladi. Toast aslida bir muncha vaqt ko‘rsatiladigan ba'zi matnlarni chiqaruvchi oynani taqdim etadi.

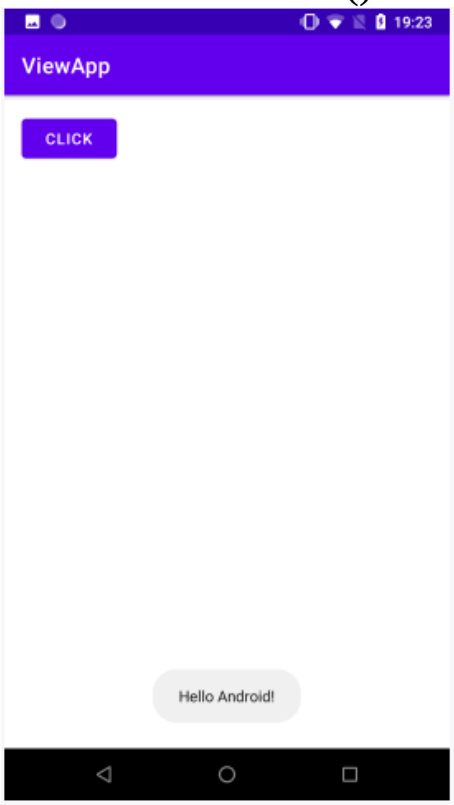
Toast ob'ektini xml belgilash kodida yaratib bo‘lmaydi, masalan, activity_main.xml. Tost faqat java kodida ishlatilishi mumkin.

```
Toast toast = Toast.makeText(this, "Hello Android!", Toast.LENGTH_LONG);
toast.show();
```

Bildirishnoma chiquvchi oyna yaratish uchun Toast.makeText() usulidan foydalaniladi, u uchta parametrdan iborat: joriy kontekst (joriy faoliyat obyekti), ko‘rsatiladigan matn va oynani ko‘rsatish vaqti.

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

Oynaning o‘zini ko‘rsatish uchun show() usuli chaqiriladi :



Odatiy bo‘lib, oyna interfeysning pastki qismida, markazda ko‘rsatiladi. Ammo biz setGravity() va setMargin() usullari yordamida bildirishnoma joylashishini sozlashimiz mumkin.

Snackbar

Snackbar elementi toastga o‘xshaydi: u sizga bildirishnoma chiquvchi xabarlarni ham ko‘rsatishga imkon beradi, ammo xabarlar ekran kengligiga mos ravishda bo‘ladi.

Snackbardan foydalanish uchun activity_main.xml fayliga button yaratamiz va uni bosgandan so‘ng Snackbar paydo bo‘lsin:

```
     <?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"
```

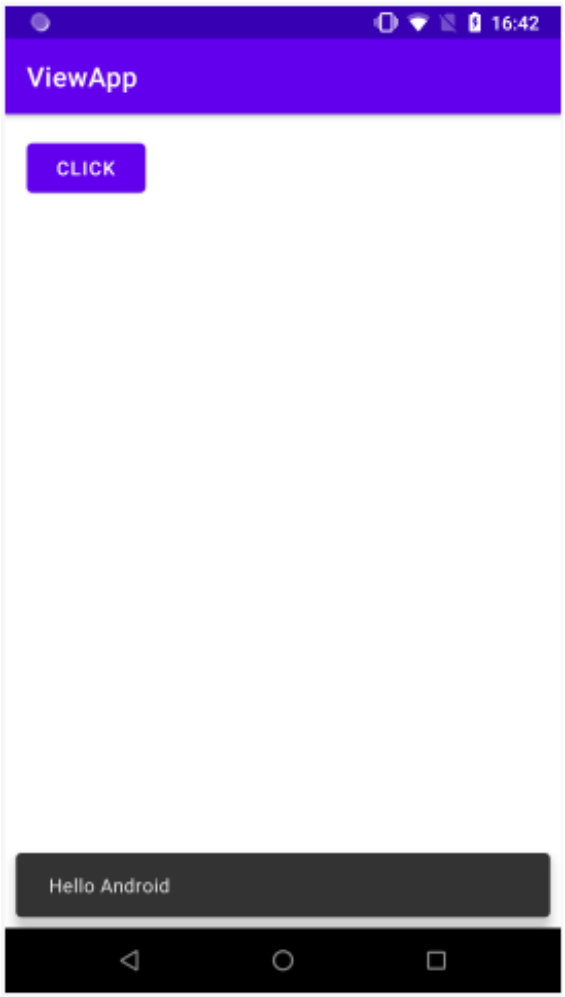
```
            android:onClick="onClick"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

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

Shuningdek, MainActivity sinfiga snackbar hosil qilish bo‘lishini keltirib o‘tamiz.

```
    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 , unga uchta parametr uzatiladi: biriktirilgan View ob'ekti, xabarning o‘zi va xabar qancha vaqt ko‘rsatilishini belgilaydigan parametr. Oxirgi parametr raqamli qiymatni olishi mumkin - millisekundlar soni yoki uchta konstantadan birini: Snackbar.LENGTH_INDEFINITE (cheklanmagan vaqt uchun ko‘rsatish), Snackbar.LENGTH_LONG (uzoq ekran) yoki Snackbar.LENGTH_SHORT (qisqa ekran).

Yaratilgandan so‘ng, Snackbar show usuli yordamida ko‘rsatiladi :



Biroq, Tostdan farqli o‘laroq, biz xabarning holatiga ta'sir qila olmaymiz, u ekranning pastki qismida ko‘rsatiladi va butun pastki qismini egallaydi.

Vizual ko‘rinishni sozlash

Snackbarning bir qator usullari tashqi ko‘rinishni sozlash imkonini beradi:

- setTextColor() : matn rangini o‘rnatadi
- setBackgroundTint() : fon rangini o‘rnatadi
- setActionTextColor() : tushdi xabaridagi tugma matn rangini o‘rnatadi

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

Checkbox

Checkbox elementlari - bu belgilangan yoki belgilanmagan holatda bo‘lishi mumkin bo‘lgan tasdiqlash elementlari. Checkbox katakchalari bir nechta qiymatlardan bir nechta tanlash imkonini beradi. Shunday qilib, activity_main.xml 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"/>
```

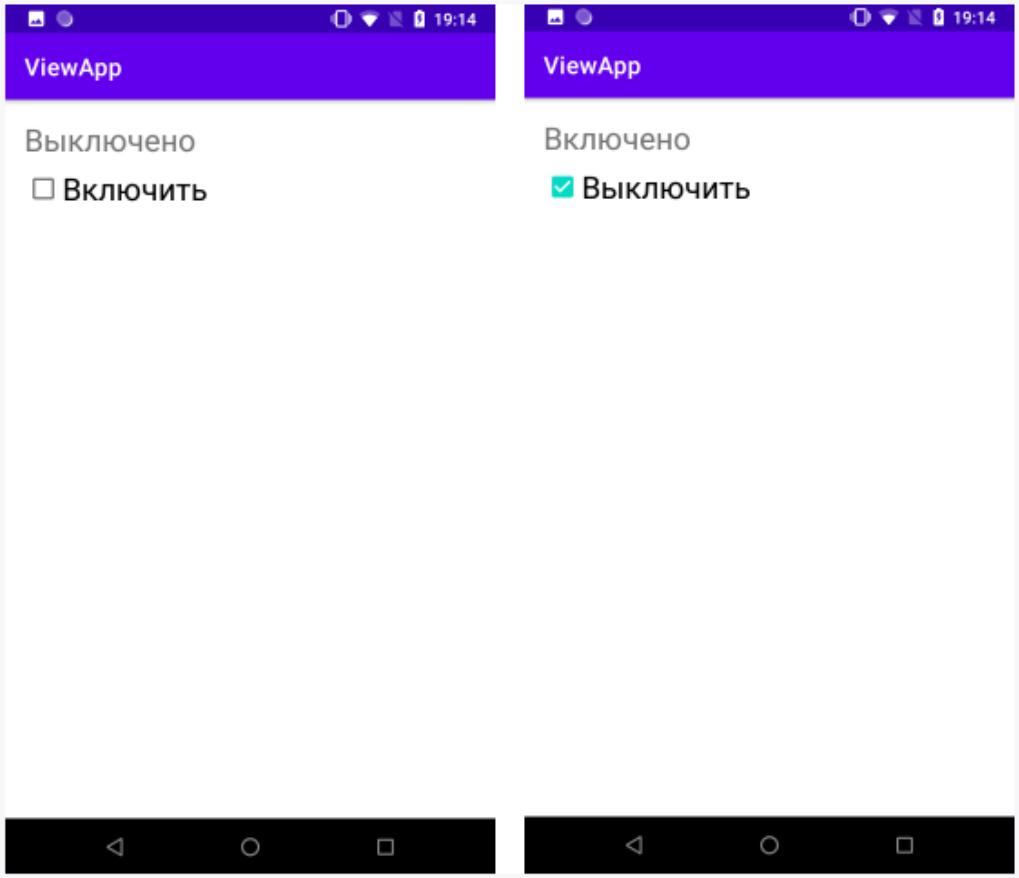
```
        <CheckBox android:id="@+id/enabled"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Включить"
            android:textSize="26sp"

            android:onClick="onCheckboxClicked"

            app:layout_constraintLeft_toLeftOf="parent"

app:layout_constraintTop_toBottomOf="@+id/selection"/>


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

Atribut android:onClick, oddiy tugmalar holatida bo‘lgani kabi, checkbox-ni tanlash uchun checkbox-ni tanlashga imkon beradi. MainActivity kodida tanlash checkbox-ni aniqlaymiz:

```
    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) {
            // Получаем флажок
            CheckBox checkBox = (CheckBox) view;
            TextView selection =
findViewById(R.id.selection);
            // Получаем, отмечен ли данный флажок
            if(checkBox.isChecked()) {
                selection.setText("Включено");
                checkBox.setText("Выключить");
            }
            else {
                selection.setText("Выключено");
                checkBox.setText("Включить");
            }
        }
```

```
    }
```

Bosilgan belgilash katakchasi onCheckboxClicked usulida parametr sifatida o‘tkaziladi. Checkbox har bosilganda onCheckboxClicked usuli ishga tushadi. Ya'ni, biz katakchani belgilaganimizda ham, belgini olib tashlaganimizda ham isChecked() usulidan foydalanib , tanlagich tanlangan yoki yo‘qligini bilib olishimiz mumkin.



Shu tarzda bir nechta belgilash katakchalaridan ham foydalanishimiz 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">
```

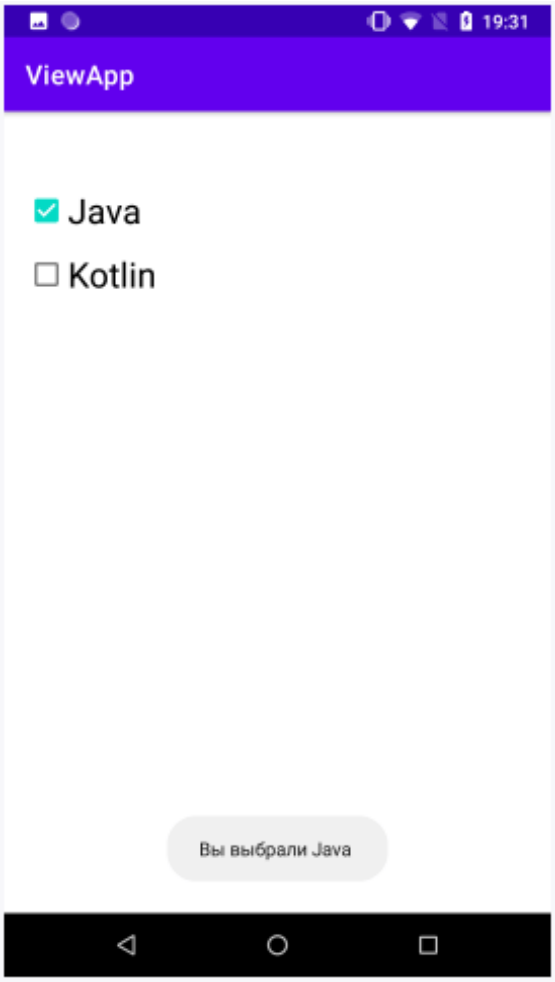
```
        <TextView android:id="@+id/selection"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="26sp"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toTopOf="parent"/>

        <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:text="Kotlin"
            android:textSize="26sp"

            android:onClick="onCheckboxClicked"

            app:layout_constraintLeft_toLeftOf="parent"

app:layout_constraintTop_toBottomOf="@+id/java"/>

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

Har bir checkbox-ni tanlashingiz mumkin. Bunday holda, biz switch...case tanlash shart operatori 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) {
            // Получаем флажок
            CheckBox checkBox = (CheckBox) view;
            // Получаем, отмечен ли данный флажок
            boolean checked = checkBox.isChecked();

            TextView selection =
findViewById(R.id.selection);

             // Смотрим, какой именно из флажков отмечен
            switch(view.getId()) {
                case R.id.java:
                    if (checked)
                        Toast.makeText(this, "Вы выбрали
Java ",Toast.LENGTH_LONG).show();
                    break;
                case R.id.kotlin:
                    if (checked)
                        Toast.makeText(this, "Вы выбрали
Kotlin",Toast.LENGTH_LONG).show();
                    break;
                default:
                    selection.setText("");
            }
        }
    }
```

Switch... case tanlash shart operatoridan foydalanib, siz bosilgan katakchaning identifikatorini olishingiz va tegishli amallarni bajarishingiz mumkin.



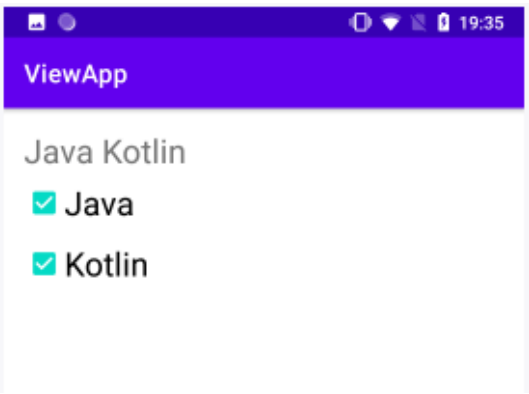
Agar biz faqat tanlangan katakchadan qiymat olishimiz kerak bo‘lsa, bu holda Switch... case tanlash shart operatoridan foydalanish shart emas, chunki biz butun kodni quyidagicha qisqartirishimiz mumkin:

```
public void onCheckboxClicked(View view) {

    CheckBox language = (CheckBox) view;

    TextView selection = findViewById(R.id.selection);
    if(language.isChecked())
        selection.setText(language.getText());
}
```

Biroq, bu holda, muammo saqlanib qolmoqda: matn maydoni faqat bitta tanlangan elementni ko‘rsatadi. Ikkala tanlangan elementni ko‘rsatish uchun MainActivity kodini o‘zgartiramiz:

```
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) {

        CheckBox java = findViewById(R.id.java);
        CheckBox kotlin = findViewById(R.id.kotlin);
        String selectedItems = "";
        if(java.isChecked())
            selectedItems +=java.getText() + " ";
        if(kotlin.isChecked())
            selectedItems +=kotlin.getText();

        TextView selection =
findViewById(R.id.selection);
        selection.setText(selectedItems);
    }
}
```



OnCheckedChangeListener

OnCheckedChangeListener -dan foydalanish checkbox tanlashda o‘zgarishlarni kuzatishning muqobil usulini taqdim etadi. Belgilangan katakchani belgilaganimizda yoki belgini olib tashlaganimizda, bu hodisa ishga tushadi. Masalan, quyidagi katakchani belgilaymiz:

```
<?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="Включить"
            android:textSize="26sp"
            app:layout_constraintLeft_toLeftOf="parent"

app:layout_constraintTop_toBottomOf="@+id/selection"/>

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

MainActivity kodida biz holatni o‘zgartirishni bajaramiz:

```
    package com.example.viewapp;

    import androidx.appcompat.app.AppCompatActivity;
    import android.os.Bundle;
    import android.widget.CheckBox;
    import android.widget.CompoundButton;
    import android.widget.TextView;

    public class MainActivity extends AppCompatActivity {

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

            TextView selection =
findViewById(R.id.selection);
            CheckBox enableBox = findViewById(R.id.enabled);

            enableBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
                public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {

                    if(isChecked) {
                        selection.setText("Включено");
                        buttonView.setText("Выключить");
                    }
```

```
                    else {
                        selection.setText("Выключено");
                        buttonView.setText("Включить");
                    }
                }
            });
        }
    }
```

OnCheckedChangeListener usuli CompoundButton asosiy sinfida aniqlanadi va onCheckedChanged ning bitta usulini belgilaydi, . Ushbu usulning birinchi parametri buttonView o‘zgartirilgan CheckBox-ning o‘zi. Va ikkinchi parametr isChecked tasdiqlash belgilangan yoki yo‘qligini ko‘rsatadi.

RadioButton

Radio tugmalari checkbox-larga o‘xshash funksiyalarni bajaradi va RadioButton klassi bilan ifodalanadi. Biroq, elementni belgilash farqli o‘laroq, biz bir vaqtning o‘zida radio tugmalar guruhida faqat bitta radio tugmachani tanlashimiz mumkin.

Tanlash uchun radio tugmalar ro‘yxatini yaratish uchun avvalo barcha radio tugmachalarini o‘z ichiga olgan RadioGroup ob'ektini yaratishingiz 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"
```

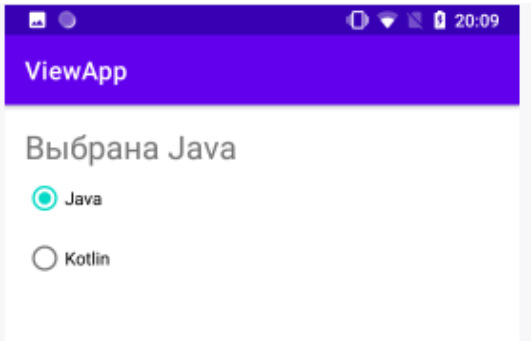
```
app:layout_constraintTop_toBottomOf="@+id/selection"
            >

            <RadioButton android:id="@+id/java"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Java" />
            <RadioButton android:id="@+id/kotlin"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Kotlin" />
        </RadioGroup>
    </androidx.constraintlayout.widget.ConstraintLayout>
```

RadioGroup LinearLayout sinfdan olinganligi sababli , biz ro‘yxatning vertikal yoki gorizontal yo‘nalishini o‘rnatishimiz mumkin, shu bilan birga faqat kalitlarning o‘zini emas, balki boshqa ob'ektlarni, masalan, Button yoki TextViewni ham o‘z ichiga oladi.

MainActivity sinfida biz radio tugmalarni tanlash jarayonini 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 {
        TextView selection;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            selection = findViewById(R.id.selection);
            // устанавливаем обработчики для кнопок

findViewById(R.id.java).setOnClickListener((view)-
>onRadioButtonClicked(view));
            // устанавливаем обработчики для кнопок

findViewById(R.id.kotlin).setOnClickListener((view)-
>onRadioButtonClicked(view));
        }
        public void onRadioButtonClicked(View view) {
            RadioButton radio = (RadioButton) view;
            // если переключатель отмечен
```

```
            boolean checked = radio.isChecked();
            // получаем текст нажатой радиокнопки
            String text = radio.getText().toString();
            // Получаем нажатый переключатель
            switch(text) {
                case "Java":
                    if (checked) selection.setText("Выбрана
Java");
                    break;
                case "Kotlin":
                    if (checked) selection.setText("Выбран
Kotlin");
                    break;
            }
        }
    }
```



Har bir alohida radiobutton bosishlarni qayta ishlashga qo‘shimcha ravishda, biz odatda OnCheckedChangeListener usulini uning radiobutton va undagi tanlashlash bilan butun RadioGroupga biriktirishimiz mumkin. Buning uchun RadioGroup elementini belgilashdan olib tashlab va identifikatorni 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>
```

Keyinchalik, MainActivity kodida biz OnCheckedChangeListener RadioGroup usulini ob'ektga biriktiramiz :

```
    package com.example.viewapp;

    import androidx.appcompat.app.AppCompatActivity;

    import android.os.Bundle;
    import android.widget.RadioButton;
    import android.widget.RadioGroup;
    import android.widget.TextView;

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

            selection = findViewById(R.id.selection);
            // получаем объект RadioGroup
            RadioGroup radios = findViewById(R.id.radios);
            // обработка переключения состояния
переключателя
            radios.setOnCheckedChangeListener((radiogroup,
id)-> {

                // получае выбранную кнопку
                RadioButton radio = findViewById(id);
```

```
                switch (radio.getText().toString()) {
                    case "Java":
                        selection.setText("Выбрана Java");
                        break;
                    case "Kotlin":
                        selection.setText("Выбран Kotlin");
                        break;
                    default:
                        break;
                }
            });
        }
    }
```

RadioGroup.OnCheckedChangeListener onCheckedChanged() usulini belgilaydi , bu RadioGroup obyekti va tanlangan kalitning identifikatori orqali uzatiladi. Keyinchalik, tanlangan radio tugmani identifikator bo‘yicha ham olishimiz va muayyan ishlov berishni amalga oshirishimiz mumkin.

ToggleButton

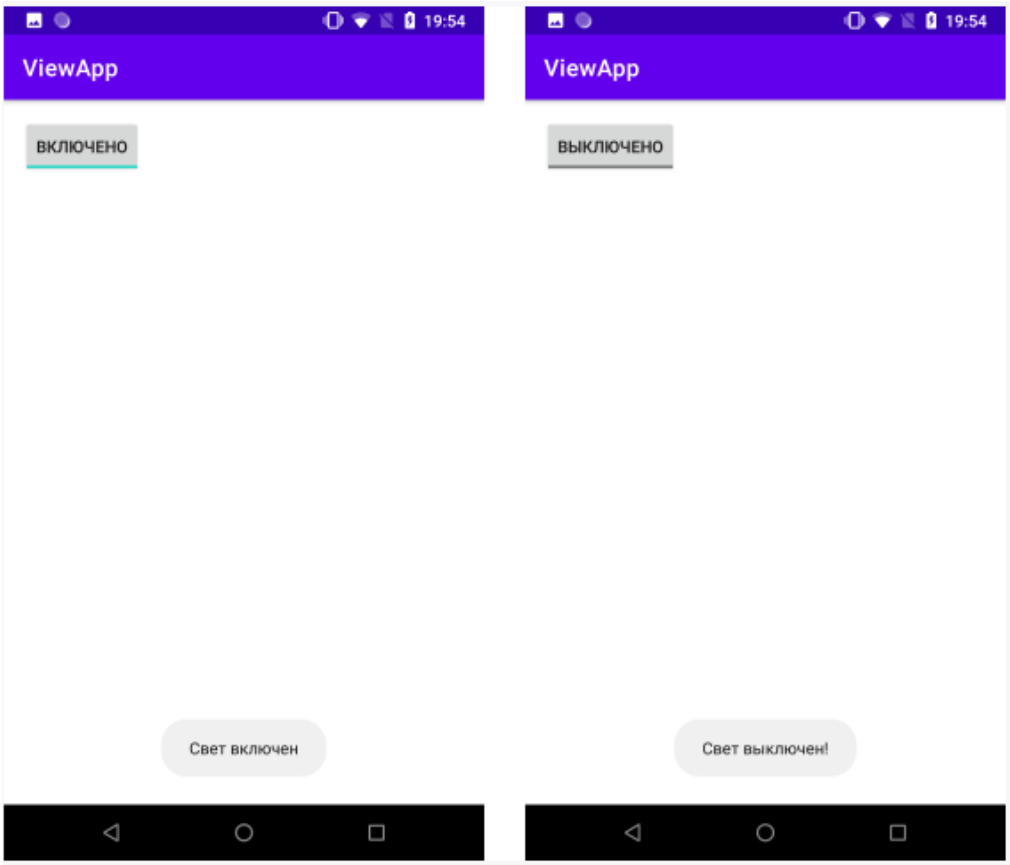
ToggleButton, xuddi CheckBox elementi kabi, ikkita holatda bo‘lishi mumkin: belgilangan va belgilanmagan va har bir holat uchun biz alohida qiymatni o‘rnatishimiz mumkin. Masalan, quyidagi ToggleButton elementini aniqlaymiz:

```
    <?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="Включено"
            android:textOff="Выключено"
            android:onClick="onToggleClicked"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
```

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

android:textOnva android:textOff atributlari tugma qiymatini mos ravishda belgilangan va belgilanmagan holatda belgilaydi. Va xuddi boshqa tugmalar kabi, biz onClick hodisa yordamida elementni bosishni boshqarishimiz mumkin. Bunday holda, biz Activity sinfida voqea ishlov beruvchisini 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) {

        // включена ли кнопка
        boolean on = ((ToggleButton) view).isChecked();
        if (on) {
            // действия если включена
            Toast.makeText(this, "Свет включен",
Toast.LENGTH_LONG).show();
        } else {
            // действия, если выключена
            Toast.makeText(this, "Свет выключен!",
Toast.LENGTH_LONG).show();
        }
    }
}
```



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("Выключено");
            toggleButton.setTextOn("Включено");
            toggleButton.setText("Выключено");
            toggleButton.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    boolean on = ((ToggleButton)
view).isChecked();

                    if (on) {

Toast.makeText(getApplicationContext(), "Свет включен",
Toast.LENGTH_LONG).show();
                    } else {

Toast.makeText(getApplicationContext(), "Свет выключен!",
Toast.LENGTH_LONG).show();
                    }
                }
            });
            layoutParams.leftToLeft =
ConstraintLayout.LayoutParams.PARENT_ID;
            layoutParams.topToTop =
ConstraintLayout.LayoutParams.PARENT_ID;
            layout.addView(toggleButton);
            setContentView(layout);
        }
    }
```

Nazoarat savollari.

1. Toast bildirishnoma oynasining vazifasini ko‘rsating.
2. Snackbar elementining vazifasi nimadan iborat?
3. Checkbox elementlarining vazifasini tavsiflang.
4. RadioButton nima va uning vazifalarini izohlang.
5. ToggleButton elementini tavsiflang.

11. MULOQOT OYNALARI VA ULARNING TURLARI. ALERTDIALOG, DATAPICKERDIALOG, TIMEPICKERDIALOG.

Reja:

11.1 Dialoglar turlari. Dialogli oynalarni hosil qilish.

11.2 AlertDialog, ProgressDialog, DatePickerDialog, TimePickerDialog.

Ba'zi hollarda siz foydalanuvchi tanlashi yoki xato xabarini koʻrsatishi kerak boʻlgan dialog oynasini koʻrsatishingiz mumkin. Siz oʻzingizning oynangizni yaratishingiz, unga kerakli tugmalarni joylashtirishingiz va ularning bosishlarini qayta ishlashingiz mumkin. Ammo Android allaqachon vazifalar uchun moslashuvchan tarzda sozlanishi mumkin boʻlgan oʻzining oʻrnatilgan dialog oynalariga ega. Oddiy vazifalar uchun dialog oynalaridan foydalanish xotira resurslarini tejash, ilovangizdagi Faoliyat sinflari sonini kamaytirish imkonini beradi.

Android-dagi dialog oynalari shaffof "suzuvchi" harakatlar boʻlib, ular chaqirilgan asosiy ekranni qisman qoplaydi. Odatda, ular xiralashtirish yoki qorayish filtrlari yordamida orqa fon harakatlarini yashirishadi. Sarlavhani setTitle() usuli va kontentni setContentView() usuli yordamida oʻrnatishingiz mumkin .

Android quyidagi dialog oynalarini qoʻllab-quvvatlaydi:

- **Dialog** - barcha turdagi dialog oynalari uchun asosiy sinf;
- **AlertDialog** - tugmalar, roʻyxat, tasdiqlash tugmalari yoki radio tugmalari boʻlgan muloqot oynasi;
- **CharacterPickerDialog** - asosiy belgi bilan bog'langan urg'uli belgini tanlash imkonini beruvchi dialog oynasi;
- **DatePickerDialog** - DatePicker elementi bilan sana tanlash oynasi
- **TimePickerDialog** - TimePicker elementi bilan vaqt tanlash dialogi

Agar mavjud dialog oynalarining hech biri sizga mos kelmasa, siz oʻzingizning muloqot oynangizni yaratishingiz mumkin.

Dialog klassi barcha dialog oynalari sinflari uchun asosiy sinfdir. ProgressDialog, TimePickerDialog va DatePickerDialog AlertDialog sinfini kengaytirgani uchun ularda buyruq tugmalari ham bo‘lishi mumkin.

Har bir dialog u ishlatiladigan faoliyat yoki fragment ichida aniqlanishi kerak. Muloqot oynasi bir yoki bir necha marta ochilishi mumkin.

Muloqot oynasini ko‘rsatish uchun showDialog() usulini chaqirish va uni ko‘rsatmoqchi bo‘lgan dialog oynasining identifikatori (dastur kodida e'lon qilinishi kerak bo‘lgan doimiy) parametr sifatida o‘tkazish kerak.

removeDialog() usuli dialog oynasini faoliyat oynasidan olib tashlaydi. showDialog() usulini qayta chaqirganingizda , dialog oynasi yana yaratilishi kerak bo‘ladi.

Dialog sinfi asosida dialog oynasini yaratishning asosiy misolini ko‘rib chiqamiz .

```
      Dialog dialog;
      @Override
      public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        dialog = new Dialog(MainActivity.this);

        // Установите заголовок
        dialog.setTitle("Заголовок диалога");
        // Передайте ссылку на разметку
        dialog.setContentView(R.layout.dialog_view);
        // Найдите элемент TextView внутри вашей разметки
        // и установите ему соответствующий текст
        TextView text = (TextView)
dialog.findViewById(R.id.dialogTextView);
        text.setText("Текст в диалоговом окне. Вы любите котов?");
      }

      public void onClick(View v)
      {
          // Выводим диалоговое окно на экран
        dialog.show();
      }
```

Odatiy bo‘lib, dialog oynasi ko‘rsatilganda asosiy faoliyat orqa fonda aktiv holatda bo‘lmaydi. Orqa fon xiralashish darajasini nazorat qilish imkonini beruvchi konstantalar mavjud:

```
 WindowManager.LayoutParams lp =
dialog.getWindow().getAttributes();
 lp.dimAmount = 0.6f; // уровень затемнения от 1.0 до 0.0
 dialog.getWindow().setAttributes(lp);

 dialog.getWindow().addFlags(WindowManager.LayoutParams.FLAG_DIM_
BEHIND);
 // Установите заголовок
 dialog.setTitle("Заголовок диалога");
 ...
```

Muloqot oynasining eng keng tarqalgan turi AlertDialog hisoblanadi .

AlertDialog dialog oynasi Dialog sinfining kengaytmasi bo‘lib , u dasturchi amaliyotida eng ko‘p ishlatiladigan dialog oynasidir. Ko‘pincha siz Ha va Yo‘q tugmalari bilan dialogni ko‘rsatishingiz kerak. Yaratilgan dialog oynalarida siz quyidagi elementlarni o‘rnatishingiz mumkin:

- sarlavha
- matnli xabar
- tugmalar: birdan uchtagacha
- ro‘yxatlar
- flajoklar

DialogFragment va dialog oynalarini yaratish

O‘z dialog oynalarimizni yaratish uchun DialogFragment sinfi bilan birgalikda AlertDialog komponentidan foydalanamiz. Loyihaga CustomDialogFragment deb nom beradigan yangi fragment sinfini qo‘shamiz:

```
package com.example.dialogsapp;

import android.app.AlertDialog;
import android.app.Dialog;
import android.os.Bundle;
import androidx.fragment.app.DialogFragment;
import androidx.annotation.NonNull;
```

```
     public class CustomDialogFragment extends DialogFragment
{

          @NonNull
          public Dialog onCreateDialog(Bundle
savedInstanceState) {

               AlertDialog.Builder builder=new
AlertDialog.Builder(getActivity());
               return builder.setTitle("Диалоговое
окно").setMessage("Для закрытия окна нажмите ОК").create();
          }
     }
```

Fragment klassi barcha standart funksiyalarini o‘z ichiga oladi, uning hayot tsikli bilan bir qatorda DialogFragment sinfidan meros bo‘lib, bir qator qo‘shimcha funktsiyalarni qo‘shadi va uni yaratish uchun biz ikkita usuldan foydalanishimiz mumkin:

- Dialog ob'ektni qaytaradigan onCreateDialog() usulni bekor qilish
- Standart onCreateView(). usuldan foydalanish

Muloqot oynasini yaratish uchun onCreateDialog() usul AlertDialog.Builder sinfdan foydalanadi Uning usullaridan foydalanib, u dialog oynasining ko‘rinishini sozlash imkonini beradi:

- **setTitle**: oyna sarlavhasini o‘rnatadi
- **setView**: oyna interfeysi tartibini o‘rnatadi
- **setIcon**: oynada ikonka o‘rnatadi
- **setPositiveButton**: Harakatni tasdiqlash tugmasini o‘rnatadi
- **setNeutralButton**: harakati tasdiqlash yoki bekor qilishdan farq qilishi mumkin bo‘lgan "neytral" tugmani o‘rnatadi
- **setNegativeButton**: Bekor qilish tugmasini o‘rnatadi
- **setMessage**: dialog oynasi matnini o‘rnatadi, lekin setView dan foydalanilganda bu usul ixtiyoriy yoki faqat xabarni ko‘rsatishimiz kerak bo‘lsa, muqobil sifatida ko‘rib chiqilishi mumkin.
- **create**: oyna yaratadi.

Bunday holda, dialog tugmasi oddiygina xabarni ko‘rsatadi.

Ushbu dialog oynasini chaqirish uchun biz activity_main.xml faylida tugmani belgilaymiz.

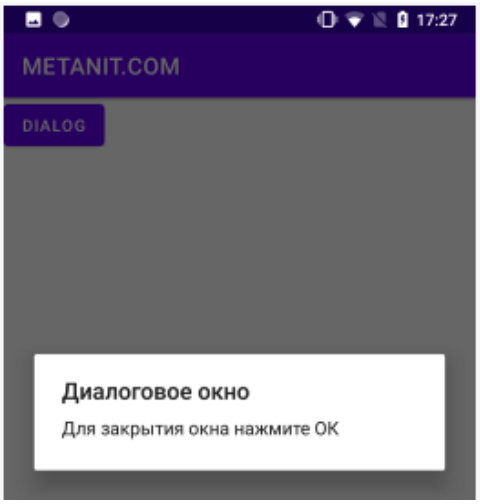
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

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="Dialog"
            android:onClick="showDialog"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"  />

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

MainActivity kodida biz dialog oynasini ishga tushiradigan tugmani bosish hodisasini aniqlaymiz:

```
package com.example.dialogsapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

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

    public void showDialog(View v) {

        CustomDialogFragment dialog = new
CustomDialogFragment();
        dialog.show(getSupportFragmentManager(),
"custom");
    }
}
```

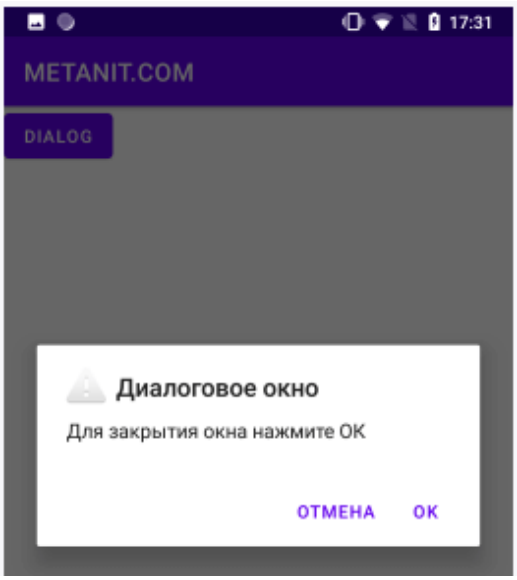
Muloqot oynasini chaqirish uchun CustomDialogFragment ob'ekti yaratiladi, so‘ngra uning show() usuli deb ataladi. Ushbu usul FragmentManager menejeri va string - ixtiyoriy teg orqali uzatiladi.

Va tugmani bosish orqali biz ma'lumotlarni dialog oynasiga berib yuborishimiz mumkin:



Endi dialog oynasini biroz kengaytiramiz:

```
     public Dialog onCreateDialog(Bundle savedInstanceState)
{

         AlertDialog.Builder builder=new
AlertDialog.Builder(getActivity());
         return builder
                 .setTitle("Диалоговое окно")
                 .setIcon(android.R.drawable.ic_dialog_alert)
                 .setMessage("Для закрытия окна нажмите OK")
                 .setPositiveButton("OK", null)
                 .setNegativeButton("Отмена", null)
                 .create();
     }
```



DatePickerDialog va TimePickerDialog

Android allaqachon ikkita dialog oynasiga ega - DatePickerDialog va TimePickerDialog, bu sizga sana va vaqtni tanlash imkonini beradi. Shuningdek tanlangan sanani ilovada qo‘shimcha ishlatish imkonini beradi.

TimePickerDialog bizga OnTimeChangedListener va OnTimeSetListener usullari yordamida vaqt tanlashni boshqarish imkonini beradi.

Ushbu komponentlar bilan ishlashda shuni hisobga olish kerakki, DatePickerDialog-da oylarni hisoblash noldan boshlanadi, ya'ni yanvarda 0, dekabrda esa 11 bo‘ladi. Xuddi shunday TimePickerDialogda ham soniya va daqiqalarni hisoblash. 0 dan 59 gacha, soatlar esa 0 dan 23 gacha bo‘ladi.

Biz ilovada DatePickerDialog va TimePickerDialog dan foydalanamiz. Keling, activity_main.xml da quyidagilarni aniqlaymiz:

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

xmlns:android="http://schemas.android.com/apk/res/android"<fo
nt></font>
         xmlns:app="http://schemas.android.com/apk/res-
auto"<font></font>
         android:layout_width="match_parent"<font></font>
         android:layout_height="match_parent"><font></font>
     <font></font>
         <TextView<font></font>
             android:id="@+id/currentDateTime"<font></font>
             android:layout_width="0dp"<font></font>

android:layout_height="wrap_content"<font></font>
             android:textSize="20sp"<font></font>

app:layout_constraintBottom_toTopOf="@id/timeButton"<font></f
ont>

app:layout_constraintLeft_toLeftOf="parent"<font></font>

app:layout_constraintRight_toRightOf="parent"<font></font>
             app:layout_constraintTop_toTopOf="parent"
/><font></font>
     <font></font>
         <Button<font></font>
             android:id="@+id/timeButton"<font></font>
```

```
            android:layout_width="0dp"<font></font>

android:layout_height="wrap_content"<font></font>
            android:text="Изменить время"<font></font>
            android:onClick="setTime"<font></font>

app:layout_constraintBottom_toTopOf="@id/dateButton"<font></f
ont>

app:layout_constraintLeft_toLeftOf="parent"<font></font>

app:layout_constraintRight_toRightOf="parent"<font></font>

app:layout_constraintTop_toBottomOf="@id/currentDateTime"
/><font></font>
    <font></font>
        <Button<font></font>
            android:id="@+id/dateButton"<font></font>
            android:layout_width="0dp"<font></font>

android:layout_height="wrap_content"<font></font>
            android:text="Изменить дату"<font></font>
            android:onClick="setDate"<font></font>

app:layout_constraintLeft_toLeftOf="parent"<font></font>

app:layout_constraintRight_toRightOf="parent"<font></font>

app:layout_constraintTop_toBottomOf="@id/timeButton"
/><font></font>
    <font></font>
    </androidx.constraintlayout.widget.ConstraintLayout><fon
t></font>
```

Bu sana va vaqtni tanlash uchun ikkita tugmani va tanlangan sana va vaqtni ko‘rsatadigan matn maydonini belgilaydi. Endilikda, MainActivity kodida o‘zgartirishlar kiritamiz:

```
    package com.example.dialogsapp;
    import androidx.appcompat.app.AppCompatActivity;
    import android.app.DatePickerDialog;
    import android.app.TimePickerDialog;
    import android.os.Bundle;
    import android.text.format.DateUtils;
    import android.view.View;
    import android.widget.DatePicker;
    import android.widget.TextView;
    import android.widget.TimePicker;
    import java.util.Calendar;
```

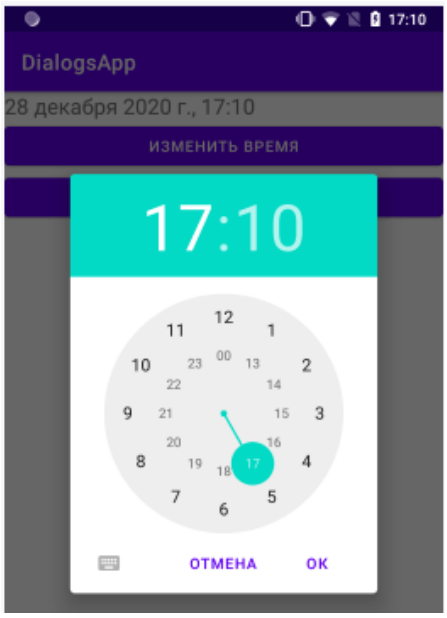
```
    public class MainActivity extends AppCompatActivity {

        TextView currentDateTime;
        Calendar dateAndTime=Calendar.getInstance();
        @Override
        public void onCreate(Bundle savedInstance) {
            super.onCreate(savedInstance);
            setContentView(R.layout.activity_main);
            currentDateTime =
findViewById(R.id.currentDateTime);
            setInitialDateTime();
        }

        // отображаем диалоговое окно для выбора даты
        public void setDate(View v) {
            new DatePickerDialog(MainActivity.this, d,
                    dateAndTime.get(Calendar.YEAR),
                    dateAndTime.get(Calendar.MONTH),
                    dateAndTime.get(Calendar.DAY_OF_MONTH))
                    .show();
        }
        // отображаем диалоговое окно для выбора времени
        public void setTime(View v) {
            new TimePickerDialog(MainActivity.this, t,
                    dateAndTime.get(Calendar.HOUR_OF_DAY),
                    dateAndTime.get(Calendar.MINUTE), true)
                    .show();
        }
        // установка начальных даты и времени
        private void setInitialDateTime() {


currentDateTime.setText(DateUtils.formatDateTime(this,
                    dateAndTime.getTimeInMillis(),
                    DateUtils.FORMAT_SHOW_DATE |
DateUtils.FORMAT_SHOW_YEAR
                            | DateUtils.FORMAT_SHOW_TIME));
        }
        // установка обработчика выбора времени
        TimePickerDialog.OnTimeSetListener t=new
TimePickerDialog.OnTimeSetListener() {
            public void onTimeSet(TimePicker view, int
hourOfDay, int minute) {
                dateAndTime.set(Calendar.HOUR_OF_DAY,
hourOfDay);
                dateAndTime.set(Calendar.MINUTE, minute);
                setInitialDateTime();
            }
        };
        // установка обработчика выбора даты
```

```
        DatePickerDialog.OnDateSetListener d=new
DatePickerDialog.OnDateSetListener() {
            public void onDateSet(DatePicker view, int year,
int monthOfYear, int dayOfMonth) {
                dateAndTime.set(Calendar.YEAR, year);
                dateAndTime.set(Calendar.MONTH,
monthOfYear);
                dateAndTime.set(Calendar.DAY_OF_MONTH,
dayOfMonth);
                setInitialDateTime();
            }
        };
    }
```

Bu erda asosiy sinf java.util.Calendar bo‘lib, u standart Java sinf kutubxonasida saqlanadi. setInitialDateTime() Usulda biz ushbu sinf misolidan millisekundlar sonini olamiz dateAndTime.getTimeInMillis() va uni formatlash yordamida matn maydonida ko‘rsatamiz.

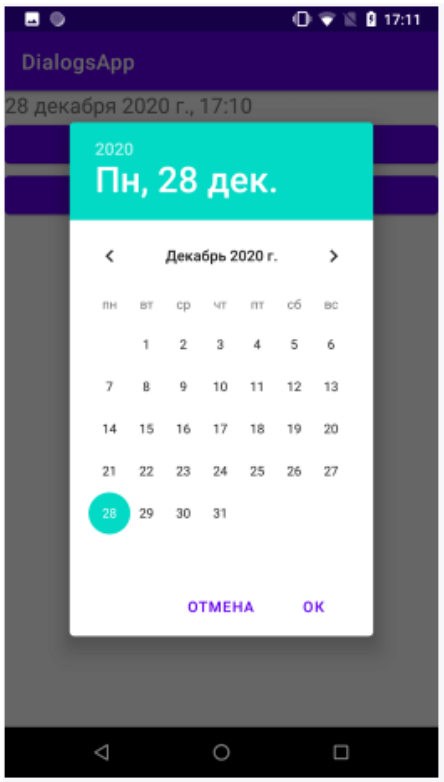
setDate()Tugma bosilganda chaqiriladigan usul sanani tanlash oynasini ko‘rsatadi. Oyna yaratilganda, sana tanlash vositasi uning ob'ektiga uzatiladi DatePickerDialog.OnDateSetListener, bu matn maydonidagi sanani o‘zgartiradi.

Xuddi shunday, setTime() usul vaqtni tanlash uchun oynani ko‘rsatadi. Oyna ob'ekti



TimePickerDialog.OnTimeSetListenermatnmaydonidagi vaqtni o‘zgartiruvchi vaqt tanlagichidan foydalanadi. Va boshlash maydoni,

vaqtni o‘zgartirish tugmasini bosish orqali biz vaqtni belgilashimiz mumkin:



Sana sozlash oynasi ham xuddi shunday ishlaydi:

Nazorat savollari.

1. Android-dagi dialog oynalarni vazifasi nimadan iborat?
2. Android-dagi dialog oynalarni tavsiflang.
3. Muloqot oynasining eng keng tarqalgan dialog turi haqida ma’lumot bering
4. DatePickerDialog va TimePickerDialog lar haqida ma’lumot bering.

12. ADAPTERLAR. MATNLI BERILGANLARNI RO‘YXAT KO‘RINISHIDA AKS ETISH. BERILGANLARNI AKS ETISH UCHUN KOMPONENTALAR. LISTVIEW, GRIDVIEW, RECYCLERVIEW, CARDVIEW.

Reja:

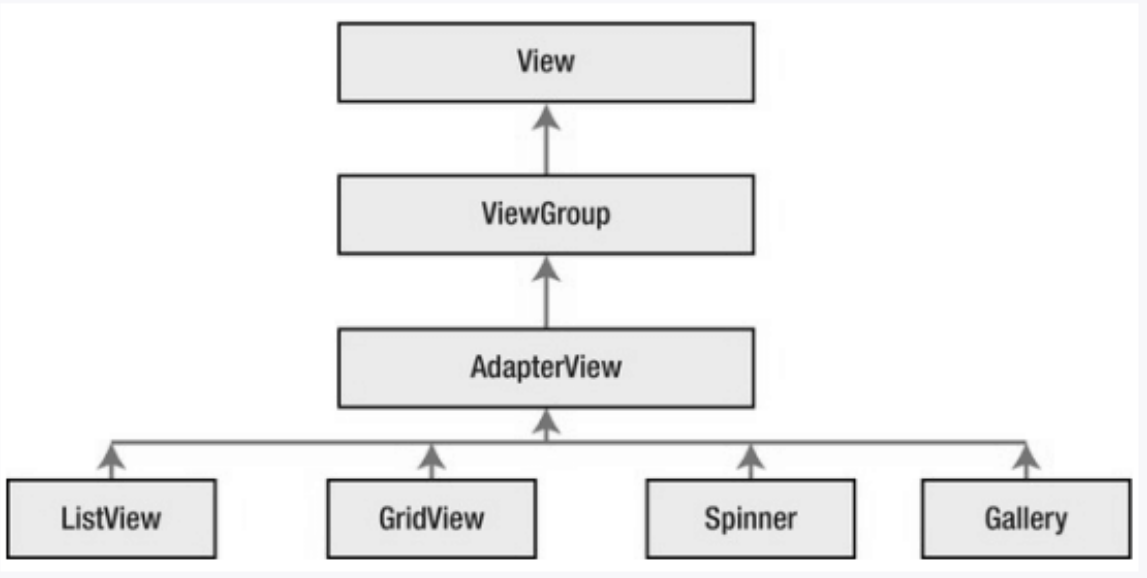
12.1 AlertDialog, ProgressDialog, DatePickerDialog, TimePickerDialog.

12.2 Berilganlarni aks etish uchun komponentalar. ListView, GridView, RecyclerView, CardView.

Adapter ob'ekti AdapterView va ushbu ko‘rinish uchun asosiy ma'lumotlar o‘rtasida ko‘prik vazifasini bajaradi. Adapter ma'lumotlar elementlariga kirishni ta'minlaydi. Adapter shuningdek, ma'lumotlar to‘plamidagi har bir element uchun Ko‘rinishni yaratish uchun javobgardir.

ListView va ArrayAdapter

Android ro‘yxatlarni ifodalovchi elementlarning keng palitrasini taqdim etadi. Ularning barchasi android.widget.AdapterView sinfining avlodlari . Bular ListView, GridView, Spinner kabi vidjetlardir. Ular boshqa boshqaruv elementlari uchun konteyner vazifasini bajarishi mumkin



Ro‘yxatlar bilan ishlashda biz uchta komponent bilan shug'ullanamiz. Birinchidan, bu ekrandagi ro‘yxatni (ListView, GridView) ifodalovchi va ma'lumotlarni aks ettiruvchi vizual element yoki vidjetdir. Ikkinchidan, bu ma'lumotlar manbai - massiv, ArrayList ob'ekti, ma'lumotlar bazasi va boshqalar, unda ko‘rsatilgan ma'lumotlarning o‘zi joylashgan. Uchinchidan, bu adapter - ma'lumotlar manbasini ro‘yxat vidjeti bilan bog'laydigan maxsus komponent.

Eng oddiy va eng keng tarqalgan ro‘yxat elementlaridan biri ListView vidjetidir . Keling, ListView elementi ushbu adapterlardan biri ArrayAdapter klassi yordamida ma'lumotlar manbai bilan qanday bog'lanishini ko‘rib chiqaylik.

ArrayAdapter klassi oddiy adapter bo‘lib, ma'lumotlar massivini TextView elementlar to‘plami bilan bog'laydi masalan, ListView dan iborat bo‘lishi mumkin. Ya'ni, bu holda ma'lumotlar manbai ob'ektlar massividir. ArrayAdapter har bir ob'ektda uni satr shakliga aylantirish uchun toString()usulni chaqiradi va natijada olingan qatorni TextView elementiga o‘rnatadi.

```
     <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"

android:layout_constraintBottom_toBottomOf="parent”
             android:layout_constraintLeft_toLeftOf=" parent "
             android:layout_constraintRight_toRightOf="parent
"
             android:layout_constraintTop_toTopOf="parent">
         </ListView>
     </androidx.constraintlayout.widget.ConstraintLayout>
```

Endi MainActivity-ga 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 {

    // набор данных, которые свяжем со списком
    String[] countries = { "Бразилия", "Аргентина",
"Колумбия", "Чили", "Уругвай"};

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

        // получаем элемент ListView
        ListView countriesList =
findViewById(R.id.countriesList);

        // создаем адаптер
        ArrayAdapter<String> adapter = new
ArrayAdapter(this,
                android.R.layout.simple_list_item_1,
countries);

        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
    }
}
```

Bu erda biz birinchi navbatda ListView elementini id bo‘yicha olamiz va keyin u uchun adapter yaratamiz.

Adapter yaratish uchun quyidagi konstruktor ishlatilgan :

ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, countries)

bu erda:

- this: joriy faoliyat
- android.R.layout.simple_list_item_1: Ramka sukut bo‘yicha taqdim etadigan ro‘yxat tartibi fayli.
- countries: ma'lumotlar to‘plami. Bu yerda massivni ko‘rsatish shart emas, bu ArrayList<T> ro‘yxat bo‘lishi mumkin.

Oxirgi qadamda setAdapter() usuli yordamida ListView adapterini o‘rnatishimiz kerak.

GridView

GridView elementi qatorlar va ustunlar to‘plami bo‘lib jadval ko‘rinishidagi displeyni ifodalaydi.

GridView-ning asosiy atributlari:

- android:columnWidth: ustunlarni belgilangan kenglikka o‘rnatadi
- android:gravitatsiya: har bir katak ichidagi tarkibning joylashuvini o‘rnatadi
- android:horizontalSpacing: ustunlar orasidagi gorizontal masofani o‘rnatadi
- android:numColumns: ustunlar sonini belgilaydi
- android:stretchMode: Ustunlar qanday cho‘zilishi va konteyner maydonini egallashini belgilaydi. Quyidagi qiymatlarni qabul qilishi mumkin:

❖ columnWidth: Har bir ustun butun kengligi bo‘ylab teng ravishda cho‘zilgan. Qiymati 2 ga teng
❖ none: Ustunlar cho‘zilmaydi. Qiymati 0 ga teng
❖ spacingWidth: Ustunlar orasida kenglik yaratiladi. Qiymati 1 ga teng
❖ spacingWidthUniform: Ustunlar orasida teng bo‘shliqlar yaratadi. Qiymati 3 ga teng.

- android:verticalSpacing: satrlar orasidagi vertikal masofani o‘rnatadi

GridView sinfining asosiy usullari:

- int getColumnWidth(): ustunlar kengligini qaytaradi
- int getHorizontalSpacing(): gorizontal oraliq hajmini qaytaradi
- int getNumColumns() : ustunlar sonini qaytaradi
- int getVerticalSpacing(): vertikal oraliq hajmini qaytaradi
- void setAdapter (ListAdapter adapteri): adapterni ma'lumotlar manbasiga ulanish uchun o‘rnatadi
- void setColumnWidth(int columnWidth): ustunlar kengligini o‘rnatadi
- void setHorizontalSpacing(int horizontalSpacing): gorizontal oraliq hajmini o‘rnatadi
- void setNumColumns(int numColumns): ustunlar sonini belgilaydi
- void setVerticalSpacing(int verticalSpacing) : vertikal oraliq hajmini o‘rnatadi

- void setSelection(int pozitsiyasi) : joriy tanlangan elementni o‘rnatadi

Quyida activity_main.xml da GridView 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">
        <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 jadvalning butun kengligi bo‘ylab teng ravishda cho‘zilgan 2 ta ustunga ega bo‘lishini va katakchalar o‘rtasida 16 dp gorizontal va vertikal chegaralar bo‘lishini ko‘rsatamiz.

Endi, ListView misolida bo‘lgani kabi, biz adapter bilan ulanishni 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);

            // получаем элемент GridView
            GridView countriesList =
findViewById(R.id.gridview);
            // создаем адаптер
            ArrayAdapter<String> adapter = new
ArrayAdapter(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);
        }
    }
```

RecyclerView

RecyclerView elementi ro‘yxatlar bilan ishlashni optimallashtirish uchun mo‘ljallangan va standart ListViewga nisbatan ko‘plab imkoniyatlarni taklif etadi.

Ma'lumotlarni taqdim etish uchun keling, loyihaga MainActivity klassi joylashgan papkada yangi Java sinfini qo‘shamiz, biz uni State deb nomlaymiz :

```
    package com.example.listapp;

    public class State {

        private String name; // название
        private String capital;  // столица
        private int flagResource; // ресурс флага
```

```
        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 sinfi mamlakat nomi va poytaxtini saqlash uchun maydonlarni, shuningdek, mamlakat bayrog'i tasvir resursiga havolani o‘z ichiga oladi. Bunday holda, res/drawable papkasida foydalanilgan holda mamlakatlar uchun bayroq rasmlari bo‘ladi.

Biz RecyclerView yordamida State ob'ektlari ro‘yxatini ko‘rsatmoqchimiz. Buning uchun res/layout papkasiga list_item.xml yangi faylini 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="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>
```

ListView-da bo‘lgani kabi, RecyclerView -da murakkab ob'ektlarni ko‘rsatish uchun siz o‘zingizning adapteringizni belgilashingiz kerak. Shuning uchun MainActivity va State sinflari joylashgan papkaga yangi Java sinfini qo‘shamiz, biz uni StateAdapter deb nomlaymiz:

```
    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 = view.findViewById(R.id.flag);
                nameView = view.findViewById(R.id.name);
                capitalView =
view.findViewById(R.id.capital);
            }
        }
    }
```

RecyclerView-da ishlatiladigan adapter RecyclerView.Adapter mavhum sinfidan meros bo‘lishi kerak. Bu sinf uchta usulni belgilaydi:

- onCreateViewHolder: Bitta State ob'ektida ma'lumotlarni saqlaydigan ViewHolder ob'ektini qaytaradi.
- onBindViewHolder: ViewHolder ob'ektini ma'lum bir pozitsiyada State ob'ektiga bog'laydi.
- getItemCount: ro‘yxatdagi ob’yektlar sonini qaytaradi.

Ma'lumotlarni saqlash uchun adapter sinfi list_item.xml da belgilangan boshqaruv elementlaridan foydalanadigan ViewHolder statik sinfini belgilaydi.

activity_main.xml faylida RecyclerView 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">
         <androidx.recyclerview.widget.RecyclerView
             android:id="@+id/list"
             android:layout_width="0dp"
             android:layout_height="0dp"


app:layoutManager="androidx.recyclerview.widget.LinearLayoutM
anager"

             app:layout_constraintLeft_toLeftOf="parent"
             app:layout_constraintRight_toRightOf="parent"
             app:layout_constraintTop_toTopOf="parent"

app:layout_constraintBottom_toBottomOf="parent"/>
     </androidx.constraintlayout.widget.ConstraintLayout>
```

E'tibor bering, RecyclerView androidx.recyclerview.widget to‘plamida joylashgan va Android Jetpack asboblar to‘plamining bir qismidir , shuning uchun vidjetdan foydalanganda uning to‘liq nomi

<androidx.recyclerview.widget.RecyclerView ....

paketni hisobga olgan holda ko‘rsatiladi (asosan, ConstraintLayout bilan bir xil):

RecyclerView dastur boshqaruvchisi turini ko‘rsatish uchun android:layoutManager atributiga ega bo‘lishi kerak. Tartib menejeri LayoutManager sinfi tomonidan taqdim etilgan ob'ektni ifodalaydi. RecyclerView kutubxonasi ushbu menejerning uchta realizatsiyasini taqdim etadi:

- LinearLayoutManager: Elementlarni bitta ustunli ro‘yxat sifatida joylashtiradi
- GridLayoutManager: Elementlarni ustunlar va qatorlar bilan grid ko‘rinishida joylashtiradi. Grid elementlarni gorizontal (gorizontal panjara) yoki vertikal (vertikal panjara) joylashtirishi mumkin.
- StaggeredGridLayoutManager: GridLayoutManagerga o‘xshaydi, lekin har bir elementni bir xil balandlikda (vertikal panjara uchun) va bir xil kenglikda (gorizontal panjara uchun) o‘rnatishni talab qilmaydi.

Bunday holda, androidx.recyclerview.widget.LinearLayoutManager qiymatidan foydalanib , biz elementlarning oddiy ro‘yxatda joylashishini bildiramiz.

E'tibor bering, LinearLayoutManager sinfi ham RecyclerView kutubxonasida joylashgan va shuning uchun qiymatni belgilashda sinfning to‘liq nomi uning paketi nomi bilan ko‘rsatiladi.

```
  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);
          // начальная инициализация списка
          setInitialData();
          RecyclerView recyclerView =
findViewById(R.id.list);
```

```
            // создаем адаптер
            StateAdapter adapter = new StateAdapter(this,
states);
            // устанавливаем для списка адаптер
            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()usul yordamida dastlabki ma'lumotlar to‘plami o‘rnatiladi. Bunday holda, biz res/drawables papkasida State ob'ektlari uchun bir qator rasm resurslari mavjudligini ko‘ramiz.

ListView orqali ro‘yxatni ko‘rsatishda bo‘lgani kabi, bu erda biz birinchi navbatda RecyclerView elementini olamiz, adapter yaratamiz va RecyclerView uchun adapterni o‘rnatamiz.

Nazorat savollari.

1. ListView va ArrayAdapter elementlari haqida ma’lumot bering.
2. ArrayAdapter ning vazifasi nimadan iborat?
3. ListView elementini yaratish jarayonini tavsiflang.
4. GridView elementi vazifasini nimadan iborat?
5. GridView sinfining asosiy usullarini sanab o‘ting.
6. RecyclerView elementi vazifasini izohlang.

13. MENYU YARATISH. KENGAYTIRILGAN MENYU, KONTEKST MENYU. MENYUDA KOMPONENTALARDAN FOYDALANISH.

Reja:

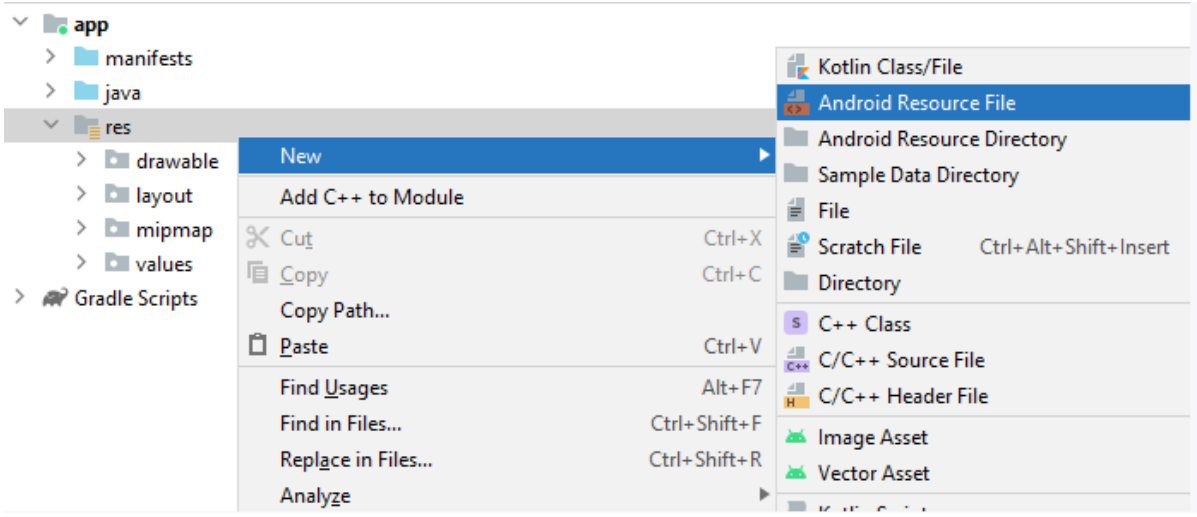
13.1 Menyu yaratish, kengaytirilgan menyu, kontekst menyu.

13.2 Menyuda komponentlardan foydalanish.

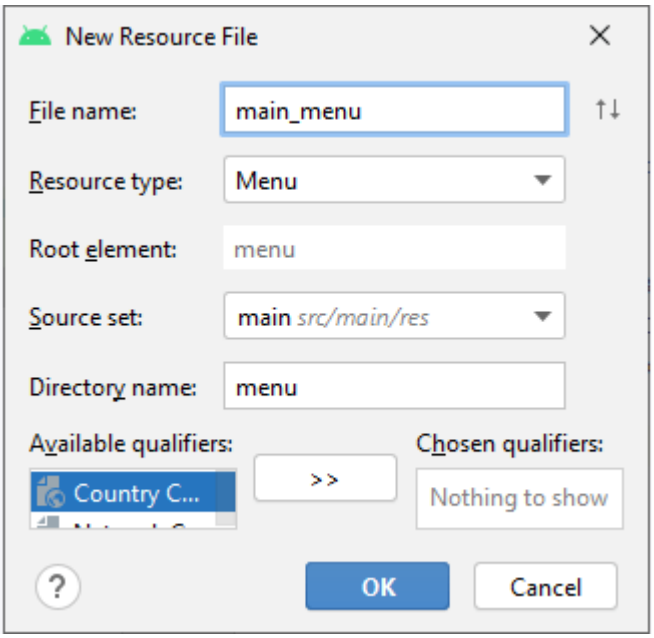
Ilovalardagi menyular android.view.Menu klassi bilan ifodalanadi va har bir faoliyat ushbu turdagi ob'ekt bilan bog'lanadi. android.view.Menu obyekti turli sonli elementlarni o‘z ichiga olishi mumkin, ular o‘z navbatida quyi elementlarni saqlashi mumkin.

Xml da menyuni belgilash

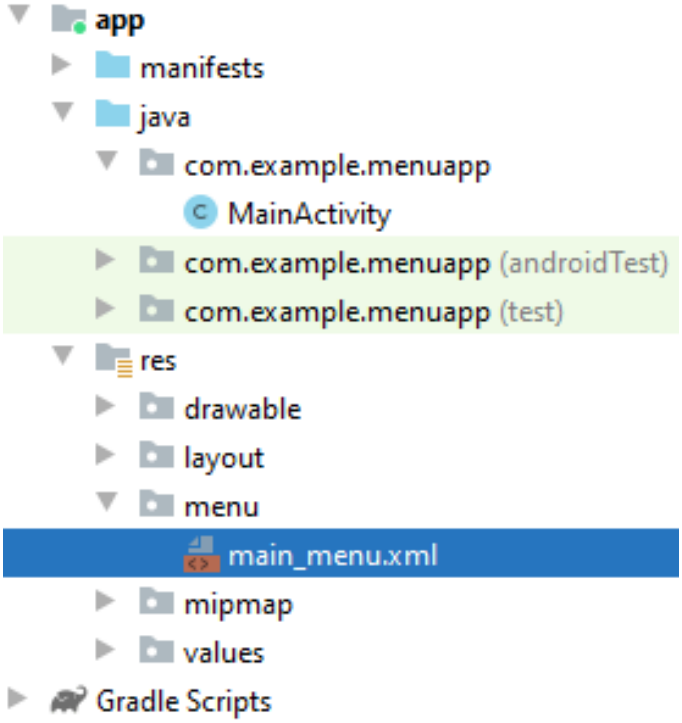
Interfeys yoki tasvir fayllari kabi menyu ham resurs hisoblanadi. Biroq, bo‘sh faoliyat bilan yangi loyihani yaratishda, sukut bo‘yicha menyu resurslari mavjud emas, shuning uchun agar kerak bo‘lsa, ularni qo‘lda qo‘shishingiz kerak. Shunday qilib, loyihada menyu resurslarini aniqlash uchun loyihadagi res katalogiga sichqonchaning o‘ng tugmachasini bosib va ochilgan ro‘yxatda New -> Android Resource File-ni tanlaymiz:



Keyin, paydo bo‘lgan oynada fayl nomi uchun main_menu nomini belgilab va Resource type maydoni uchun Menu ni tanlaymiz :



Shundan so‘ng, res katalogida main_menu.xml faylini o‘z ichiga olgan menyu pastki katalogi yaratiladi.



bu fayl bitta bo‘sh menyu elementini belgilaydi:

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

      </menu>
```

Shundan so‘ng bir nechta ichki menyu optsiyalarini qo‘shib chiqamiz.

```
    <?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> teg faylning ildiz tugunidir va bir yoki bir nechta <item> va <group> elementlardan iborat menyuni belgilaydi

<item> Element menyu bandlaridan biri bo‘lgan MenuItem obyektini ifodalaydi. Ushbu <menu> element pastki menyuni yaratadigan ichki kichik elementni o‘z ichiga olishi mumkin.

<item> element quyidagi atributlarni o‘z ichiga oladi, bu uning tashqi ko‘rinishi va harakatini belgilaydi:

- android:id: menyu elementining identifikatori, bu foydalanuvchi tanlaganida uni tanib olish va id bo‘yicha resurs qidiruvi orqali topish imkonini beradi.
- android:icon android:icon="@drawable/ic_help" : element ( ) uchun tasvirni belgilaydigan chizilgan manbaga havola
- android:title: Element sarlavhasini o‘z ichiga olgan string manbasiga havola.
- android:orderInCategory : menyudagi elementning tartibi

Menyuni elementlar bilan to‘ldirish

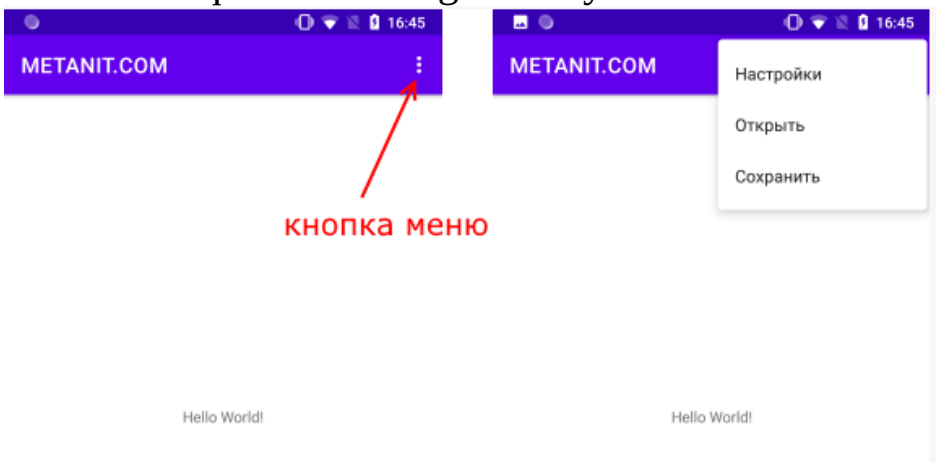
Biz uchta elementdan iborat menyuni belgilab oldik, lekin faqat fayldagi elementlarni belgilash menyuni yaratmaydi. Bu shunchaki deklarativ tavsif. Uni ekranda ko‘rsatish uchun biz uni Activity sinfida ishlatishimiz kerak. Buning uchun onCreateOptionsMenu usulini hosil

qilishimiz kerak. Keling, MainActivity sinfiga o‘tamiz va uni quyidagicha 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 MenuInflater ob'ektni qaytaradi. Bu usul birinchi parametr sifatida bizning xml formatidagi deklarativ menyu tavsifimizni ifodalovchi resursni oladi va u bilan ikkinchi parametr sifatida qabul qilingan menyu ob'ektini to‘ldiradi.

Dasturni ishga tushirib, yuqori o‘ng burchakdagi menyu tugmasini bosish orqali hosil bo‘lgan menyuni ko‘rishimiz mumkin:



Menyu bosishlarini boshqarish

Menyu bandlaridan birini bossak, hech narsa bo‘lmaydi. Harakatlarni menyuga ulash uchun activity sinfida onOptionsItemSelected ni ishga tushirishimiz kerak.

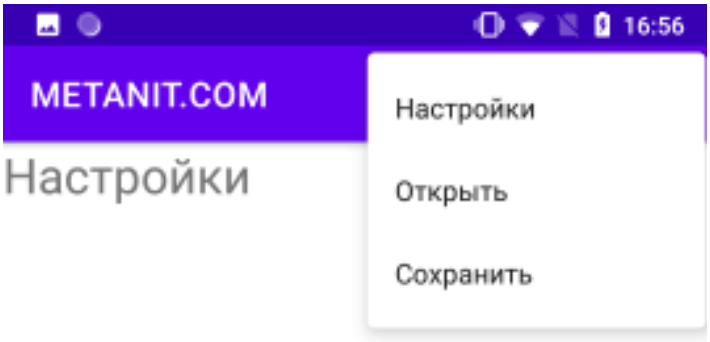
Tanlangan menyu bandini active_main.xml faylida ko‘rsatish uchun id=header bilan matn maydonini belgilaymiz:

```
<?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>
```

```
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 =
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 uning identifikatorini olamiz int id = item.getItemId(). Keyin biz switch..case konstruktsiyasidan foydalanib kerakli variantni tanlaymiz va tanlovga qarab, biz ma'lum harakatlarni bajaramiz - bu holda biz TextView matnini o‘rnatamiz.



Shuni ta'kidlash kerakki, bu holda, agar bizning vazifamiz shunchaki tanlangan menyu elementining matnini ko‘rsatish bo‘lsa, u holda kalit konstruktsiyasi o‘rniga biz shunday yozishimiz mumkin:

```
headerView.setText(item.getTitle());
```

Dasturiy menyu yaratish

Xml da menyu bandlarini belgilashdan tashqari, dasturli ravishda menyu yaratishingiz ham mumkin. Yangi menyu elementlarini qo‘shish uchun Menu sinfining add() usulidan foydalaning.

```
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 =
findViewById(R.id.selectedMenuItem);
        headerView.setText(title);

        return super.onOptionsItemSelected(item);
    }
}
```

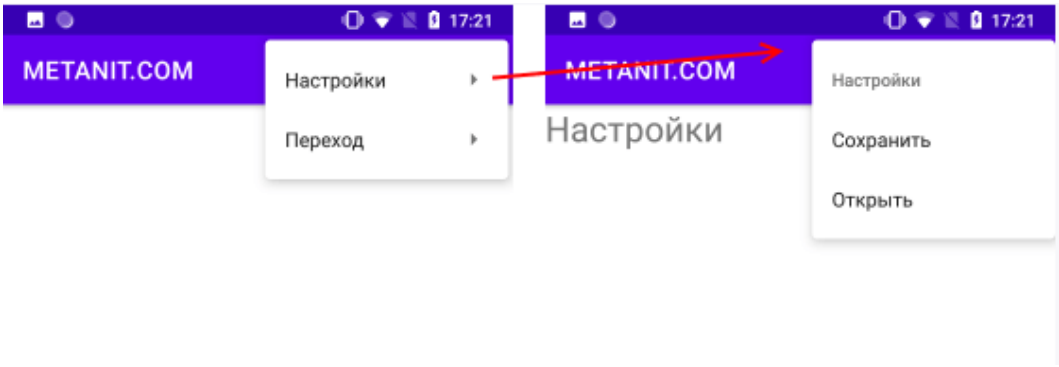
Amaldagi usulning versiyasi add()menyu elementi uchun sarlavha oladi.

Quyi menyu yaratish

Menyu belgilash faylida pastki menyu yaratish uchun biz ichki menu: elementni aniqlaymiz.

```
     <?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>
```

Menyuni bosgandan so‘ng, yuqori darajadagi elementlar ko‘rsatiladi, ularni bosish orqali biz pastki menyuga o‘tishimiz mumkin:



Menyudagi guruhlar

Group elementdan foydalanish menyu elementlarini guruhga 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 qabul qilishi mumkin: single(har bir element uchun radio button yaratiladi), all(har bir element uchun checkbox yaratiladi) va none.

Bunday holda, har bir element uchun radio button yaratiladi. Va birinchi element belgilangan radio button-ga o‘rnatiladi (android:checked="true").

Activity_main.xml interfeys belgilash faylida matn maydonini ham belgilaymiz:

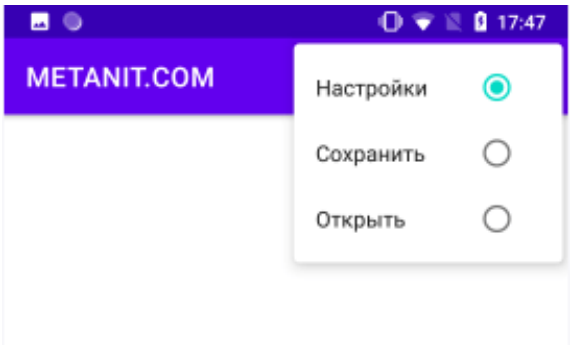
```
      <?xml version="1.0" encoding="utf-8"?>
      <androidx.constraintlayout.widget.ConstraintLayout

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>
```

Shuningdek MainActivity sinfida biz tanlangan menyu elementi uchun radio button tanlashni aniqlaymiz:

```
    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 =
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);
        }
    }
```

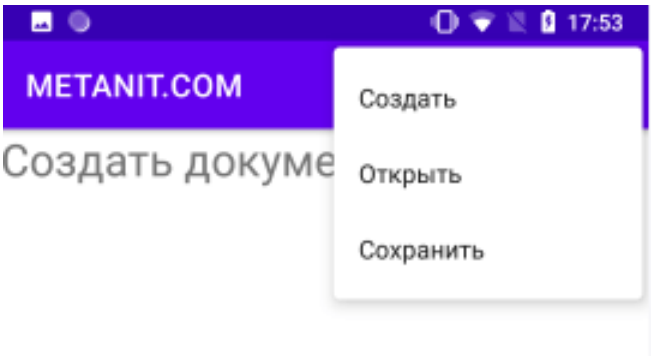


Menyular va quyi menyularda guruhlarni dasturiy yaratish. Dasturiy ravishda guruhlar va pastki menyularni ham yaratishingiz mumkin.

```
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(0        // Группа
                ,1        // id
                ,0        //порядок
                ,"Создать");  // заголовок

        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 =
findViewById(R.id.selectedMenuItem);

            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 qo‘llaniladigan usulning versiyasi add()menyuga element qo‘shib, quyidagi parametrlarni oladi: guruh raqami, identifikator, menyudagi element tartibi va element sarlavhasi.



Nazorat savollari.

1. Ilovalardagi menyu yaratish haqida ma’lumot bering.
2. Menyu bosishlarini boshqarish tartibini tushuntiring.
3. Dasturiy menyu yaratishni izohlang.
4. Quyi menyu yaratishni tavsiflang.

NAZORAT TESTLARI

1. Java dasturlash tilida 10 / 3 qanday natijani chiqaradi?

A) 3
B) 3.33
C) 1
D) 3.0

2. Java dasturlash tilida o‘zgarmas qanday aniqlanadi?

A) const
B) final
C) static
D) def

3. Java dasturlash tilida int ma’lumot turi o‘lchami?

A) 4 bytes
B) 8 bytes
C) 2 bytes
D) 16 bytes

4. Java tilidagi int massividagi elementlarning standart qiymati qanday?

A) 0
B) -1
C) null
D) Undefined

5. Quyidagi kod qanday natijani ekranga chiqaradi?

System.out.println(10 > 9);

A) 0
B) -1
C) null
D) Undefined

6. Java-da massiv yoki to‘plam elementlarini takrorlash uchun qaysi tsikl ishlatiladi?

A) for loop
B) while loop

C) do-while loop
D) Enhanced for loop

7. Javada doimiyni e'lon qilishning to‘g'ri yo‘li qanday?
A) final int CONSTANT_VALUE = 10;
B) const int CONSTANT_VALUE = 10;
C) static int CONSTANT_VALUE = 10;
D) int CONSTANT_VALUE = 10;

8. Java-da satr uzunligini olish uchun qaysi usul ishlatiladi?
A) length()
B) size()
C) count()
D) getSize()

9. Java'da "super" kalit so‘zi nimani anglatadi?
A) Parent class
B) Current class
C) Child class
D) None of the above

10. Quyidagi kod qanday natijani chiqaradi?
System.out.println("Java".charAt(2));
A) v
B) a
C) v (index starts from 0)
D) Compilation Error

11. Quyidagilardan qaysi biri haqiqiy Java identifikatori emas?
A) _variableName
B) 3variableName
C) variableName
D) $variableName

12. Java-da massiv qanday ishga tushiriladi?
A) int[] numbers = {1, 2, 3};
B) int[] numbers = new int[3] {1, 2, 3};
C) int[] numbers = new int[]{1, 2, 3};

D) Both A and C

13. Quyidagi kod qanday natijani chiqaradi?

```
int x = 5;
System.out.println(x++);
```

A) 5
B) 6
C) 4
D) Compilation Error

14. Java-da hech qanday qiymat qaytarmaydigan usulni e'lon qilish uchun qaysi kalit so‘z ishlatiladi?

A) void
B) null
C) none
D) empty

15. Java'da double tipidagi o‘zgaruvchini e'lon qilishning to‘g'ri yo‘li qanday?

A) double x;
B) Double x;
C) float x;
D) None of the above

16. Quyidagilardan qaysi biri Java-da matematika sinfining to‘g'ri usuli emas?

A) min()
B) max()
C) average()
D) sqrt()

17. Java'da ikki o‘lchovli massivni e'lon qilishning to‘g'ri yo‘li qanday?

A) int[][] matrix;
B) int[2][2] matrix;
C) int[2, 2] matrix;
D) int[2] matrix[];

18. Java-da satrni barcha kichik harflarga aylantirish uchun qaysi usul ishlatiladi?

A) toLowerCase()
B) lower()
C) convertToLower()
D) toLower()

19. Java'da "this" kalit so‘zi nimani anglatadi?

A) Current object
B) Current class
C) Previous object
D) Next object

20. Quyidagi kod qanday natijani chiqaradi?

```
String str = "Hello";
str.concat(" World");
System.out.println(str);
```

A) Hello
B) World
C) Hello World
D) Compilation Error

21. Quyidagilardan qaysi biri Java-da primitiv ma'lumotlar turi emas?

A) byte
B) string
C) short
D) double

22. Quyidagi kod qanday natijani chiqaradi?

```
int[][] matrix = {{1, 2}, {3, 4}};
System.out.println(matrix[1][0]);
```

A) 1
B) 2
C) 3
D) 4

23. Java-dagi barcha sinflarning yuqori sinfi nima?

A) Object
B) Class
C) Main
D) None

24. Java-da hech qanday qiymat qaytarmaydigan usulni qanday e'lon qilinadi?

A) void method()
B) null method()
C) empty method()
D) None of the above

25. Java dasturlash tilida quyida qaysi turdagi izohlardan foydalaniladi?

A) // This is a comment
B) /* This is a comment */
C) * This is a comment *
D) # This is a comment

26. Quyidagi kod qanday natijani chiqaradi?

```
int x = 10;
System.out.println(x > 5 ? "Greater" : "Smaller");
```

A) Greater
B) Smaller
C) Compilation Error
D) Runtime Error

27. Quyidagi operatorlardan qaysi biri Java tilida mantiqiy VA amalini bajarish uchun ishlatiladi?

A) &&
B) ||
C) !
D) &

28. Java'da konstruktorni aniqlashning to‘g'ri yo‘li qanday?

A) public MyClass()
B) void MyClass()

C) public void MyClass()
D) MyClass()

29. Quyidagilardan qaysi biri Java tilidagi kirish taqiqlangan ruxsat modifikatori hisoblanadi?
A) private
B) protected
C) package
D) public

30. Quyidagi kod qanday natijani chiqaradi?
System.out.println("Java".substring(1, 3));
A) av
B) Java
C) Jav
D) Compilation Error

31. Qaysi tartib sizga asosiy elementlarni asosiy joylashuvga yoki bir-biriga nisbatan joylashtirish imkonini beradi?
a) RelativeLayout
b) LinearLayout
c) FrameLayout
d) ConstraintLayout

32. Qaysi XML atributi Android maketlarida ko‘rinishning ota-onasiga nisbatan kengligini belgilash uchun ishlatiladi?
a) layout_width
b) match_parent
c) layout_height
d) wrap_content

33. Qaysi komponent Androidda activitilar o‘rtasida navigatsiya uchun ishlatiladi?
a) Intent
b) Fragment
c) BroadcastReceiver
d) Service

34. Androidda onCreateOptionsMenu() usulining vazifasi nima?
a) Menu opsiyasini yaratish
b) Menu opsiyalari bosilish hodisasi
c) List yaratish
d) Alert dialog yaratish

35. Qaysi Layout bitta satr yoki ustunda bolalar ko‘rinishini gorizontal yoki vertikal ravishda tartibga solish uchun ishlatiladi?
a) LinearLayout
b) RelativeLayout
c) GridLayout
d) FrameLayout

36. Android-da parametrlar menyusiga elementlar qo‘shish uchun qaysi usuldan foydalaniladi
a) getMenuInflater().inflate()
b)addMenu()
c) onCreateOptionsMenu()
d) addItem()

37. Androidda Intent ob'ektining maqsadi nima?
a) Komponentlar o‘rtasidagi aloqani yoqish uchun
b) Ilova holatini saqlash uchun
c) Layoutdagi activitini aniqlash
d) Bildirishnomalarni ko‘rsatish uchun

38. Qaysi maket sizga bolalar ko‘rinishlari satr va ustunlarga ajratilgan tartibni yaratishga imkon beradi?
a) TableLayout
b) LinearLayout
c) ConstraintLayout
d) FrameLayout

39. Androidda menyu bandi tanlanganda qaysi usul chaqiriladi?
a) onOptionsItemSelected()
b) onMenuItemClick()
c) onCreateOptionsMenu()
d) onPrepareOptionsMenu()

40. Androidda foydalanuvchiga xabarni ko‘rsatish va javobni so‘rash uchun qaysi komponentdan foydalaniladi?

a) Alert Dialog
b) Toast
c) Snackbar
d) Popup Window

41. Androidda menyuning asosiy maqsadi nima?

a) Faoliyat bandlarini ko‘rsatish uchun
b) Faoliyatlar orasida navigatsiyani ta'minlash
c) Foydalanuvchi imo-ishoralarini boshqarish uchun
d) Reklamalarni ko‘rsatish uchun

42. Moslashuvchan cheklovlarga ega murakkab UI dizaynlari uchun qaysi tartib menejeri tavsiya etiladi?

a) ConstraintLayout
b) RelativeLayout
c) LinearLayout
d) FrameLayout

43. Intent yordamida Androidda yangi faoliyatni boshlash uchun qaysi usul chaqiriladi?

a) startActivity()
b) startIntent()
c) startActivityForResult()
d) startActivityIntent()

44. Androidda elementlar ro‘yxatini ko‘rsatish uchun qaysi komponentdan foydalaniladi?

a) ListView
b) RecyclerView
c) GridView
d) ScrollView

45. Androidda Ogohlantirish dialogi uchun maxsus tartibni qanday yaratasiz?

a) LayoutInflater yordamida tartibni inflate qilish.

b) AlertDialog sinfini bekor qiling
c) Yangi tartib XML faylini yaratish
d) Maxsus ko‘rinishlarni o‘rnatish uchun AlertDialog.Builder dan foydalanish

46. Android XML layout fayllaridagi qaysi atribut ko‘rinish mazmunining og'irligini aniqlash uchun ishlatiladi?
a) gravity
b) layout_gravity
c) orientation
d) layout_alignParent

47. Androidda onOptionsItemSelected() usulining maqsadi nima?
a) Menu bandlarini bosilish hodisasi
b) Menu yaratish hodisasi
c) Menu bandlarini yaratish hodisasi
d) Dialog tugma yaratish

48. Bir xil o‘lchamdagi katakchalar panjarasini yaratish uchun qaysi maket menejeri ishlatiladi?
a) GridLayout
b) LinearLayout
c) RelativeLayout
d) FrameLayout

49. Androidda kontekst menyusiga elementlar qo‘shish uchun qaysi usuldan foydalaniladi?
a) getMenuInflater().inflate()
b) addItem()
c) add()
d) onCreateOptionsMenu()

50. Androidda kontekst menyusining maqsadi nima?
a) Kontekstga asoslangan activity variantlarini ko‘rsatish uchun
b) Activitilar orasida navigatsiyani ta'minlash
c) Android manifestni sozlash
d) Bildirishnomalarni ko‘rsatish

51. Qisqa xabarlarni ekranning pastki qismida aks ettirish uchun qaysi komponentdan foydalaniladi?

a) Toast
b) Dialog
c) Snackbar
d) Alert Dialog

52. Androidda ConstraintLayout ning asosiy maqsadi nima??

a) Moslashuvchan UI dizaynlarini yaratish uchun
b) Satr va ustunlarda bolalar koʻrinishini tashkil qilish
c) Koʻrinishlarni bir-biriga nisbatan tekislash uchun
d) Rasmlar va media kontentini koʻrsatish uchun

53. Androidda koʻrinishda uzoq bosish hodisasi aniqlanganda qaysi usul chaqiriladi?

a) onLongClick()
b) onLongPress()
c) onItemLongClick()
d) onItemLongPress()

54. Bir vaqtning oʻzida bitta koʻrinishni koʻrsatishga imkon beruvchi koʻrinishlar toʻplamini yaratish uchun qaysi tartib menejeri ishlatiladi?

a) FrameLayout
b) LinearLayout
c) RelativeLayout
d) GridLayout

55. Androidda startActivityForResult() usulining maqsadi nima?

a) Yangi activitini boshlash va natija olish uchun
b) Xizmatni asinxron tarzda ishga tushirish uchun
c) Natija kutmasdan yangi activitini boshlash
d) Aniq niyat bilan faoliyatni boshlash

56. Androidda ko‘rinishga biriktirilgan qalqib chiquvchi menyuni ko‘rsatish uchun qaysi komponentdan foydalaniladi?

a) PopupMenu
b) ContextMenu
c) PopupWindow
d) Alert Dialog

57. Androidda Snackbarning maqsadi nima?

a) Foydalanuvchiga fikr bildirish uchun
b) Bildirishnomalarni ko‘rsatish uchun
c) Menyu bandini yaratish uchun
d) Ekranning pastki qismida qisqa xabarlarni ko‘rsatish uchun

58. O‘zgaruvchan satr va ustun o‘lchamlari bilan moslashuvchan ko‘rinishlar to‘plamini yaratish uchun qaysi tartib menejeri ishlatiladi?

a) GridLayout
b) LinearLayout
c) RelativeLayout
d) FrameLayout

59. Android-dagi AlertDialog-da setPositiveButton() usulining maqsadi nima?

a) Ijobiy tugmani va uning hodisasi
b) Muloqot oynasining sarlavhasini o‘rnatish uchun
c) Muloqot oynasi xabarini o‘rnatish uchun
d) Salbiy tugmani va uning hodisasi

60. Foydalanuvchi Androidda ogohlantirish dialogini o‘chirib qo‘yganda qaysi usul chaqiriladi?

a) onDismiss()
b) onCancel()
c) onDialogDismissed()
d) onDialogClosed()

ASOSIY ADABIYOTLAR

1. Дейтел П., Дейтел Х., Дейтел Э. А66 Android для разработчиков. — СПб.: Питер, 2015. — 384— (Серия «Библиотека программиста»).

2. Дон Гриффитс, Дэвид Гриффитс Head First. Программирование для Android.

Марио Цехнер Программирование игр под Android.

3. Сильвен Ретабоуил Android NDK Руководство для начинающих 2-е издание Москва, 2016 С.

4. Хашими С., Коматинени С., Маклин Д. Разработка приложений для Android. – Санкт-Петербург: Питер, 2011. -736 с.

5. Харди Б., Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов. 2-е изд. - Санкт-Петербург: Питер, 2016. - 640 с

6. Меднике З., Дорнин Л., Мик Б., Накамура. -Москва Программирование под Android. 2-е изд. - СПб.: Питер, 2013. - 560 с.

7. Дейтел ГГ, Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. - Санкт-Петербург: Питер, 2016. - 512 с.

8. Start Android - учебник по Android для начинающих и продвинутых [Электронный ресурс] Режим доступа: https://startandroid.ru/ru/

9. Введение в разработку приложений для ОС Android [Электронный ресурс] - Национальный Открытый Университет «ИНТУИТ» 2014г. Режим доступа: https://www.intuit.ru/studies/courses/12643/1191/info

10. Разработка приложений для смартфонов на ОС Android [Электронный ресурс] - Национальный Открытый Университет «ИНТУИТ». Режим доступа: https://www.intuit.ru/studies/courses/12786/1219/info

11. Сайт Александра Климова. Изучаем Android. [Электронный ресурс] Режим доступа: http://developer.alexanderklimov.ru/android/

INTERNET SAYTLARI

https://developer.android.com/ - Google android ilovalarni qo‘llab quvvatlash sayti

https://library.ziyonet.uz/ - Milliy kutubxona

J.J. Atamuradov, S.S. Salimov

MOBIL ILOVALAR YARATISH

O‘quv qo‘llanma

Muharrir: **E.Eshov**
Tex. muharrir: **D.Abduraxmonova**
Musahhih: **M.Shodiyeva**
Badiiy rahbar: **M.Sattorov**

Nashriyot litsenziyasi № 022853. 04.03.2022.
Original maketdan bosishga ruxsat etildi: 29.05.2024. Bichimi 60x84. Kegli 16 shponli. “Times New Roman” garnitura 1/16.
Ofset bosma usulida. Ofset bosma qog‘ozi.
Bosma tabog‘i 11. Adadi 100. Buyurtma № 374

KAMOLOT

“BUXORO DETERMINANTI” MCHJ
bosmaxonasida chop etildi.
Buxoro shahar Namozgoh ko‘chasi 24 uy
Tel:. + 998 91 310 27 22