MOBIL ILOVALARINI ISHLAB CHIQISH

Nishanov A.X.,
Allaberganov O.R.,
Madaminov U.A.

O‘ZBEKISTON RESPUBLIKASI OLIY TA’LIM, FAN VA INNOVATSIYALAR VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI URGANCH FILIALI

NISHANOV A.X., ALLABERGANOV O.R., MADAMINOV U.A.

MOBIL ILOVALARINI ISHLAB CHIQISH

O‘zbekiston Respublikasi Oliy ta’lim, fan va innovatsiyalar vazirligi tomonidan o‘quv qo‘llanma sifatida tavsiya etilgan

5330600 – “Dasturiy injiniring” yo‘nalishi talabalari uchun

TOSHKENT
“METODIST NASHRIYOTI”
2024

UDK: 004.77(075.8)
BBK: 32.973ya7
N 69

Nishanov A.X.
Mobil ilovalarini ishlab chiqish / Allaberganov O.R., Madaminov U.A./. O‘quv qo‘llanma. – Toshkent: “METODIST NASHRIYOTI”, 2024. – 198 b.

Ushbu «Mobil ilovalarini ishlab chiqish» o‘quv qo‘llanma mobil ta’limning o‘quv jarayonida tutgan o‘rni, ularning tamoyillari, modellari, mobil o‘qitishning o‘quv-metodik, dasturiy va texnik ta’minoti to‘g’risida bilimlarga ega bo‘lishga yordam beradi. Qo‘llanma yordamida talabalar muayyan mobil ilovalarning ta’lim platformalari va ularda elektron resurslarni qanday tuzish haqida bilim va ko‘nikmaga ega bo‘ladilar.

Taqrizchilar:
O.K. Xo‘jayev
TATU UF, “Axborot texnologiyalari” kafedrasi mudiri, t.f.n. PhD,
M.S. Sharipov
Urganch Davlat universiteti, “Axborot texnologiyalari” kafedrasi dotsenti.

O‘zbekiston Oliy va o‘rta maxsus ta’lim vazirligining 2022–yil 9–sentabrdagi 302–sonli buyrig‘iga asosan nashr etishga ruxsat berilgan.

ISBN 978-9910-03-215-8

KIRISH

Ushbu «Mobil ilovalarini ishlab chiqish» o'quv qo'llanmasi 5330600 «Dasturiy injiniring» yo'nalishi 3-kurs talabalari uchun mo'ljallangan. Mobil ilovalarni ishlab chiqish fanining asosiy maqsadi talabalarga mobil dasturlash, ta'limda va sohalarda mobil texnologiyalarni yanada ko'proq tadbiq qilish, android muhitida java dasturlash tilini o'rganish, ularni amaliyotda qo'llash, ko'nikma, malakalarini shakillantirish va rivojlantirishdan iboratdir.

Bugungi kunda zamonaviy dasturlash tillari jadallik bilan rivojlanib bormoqda. Java dasturlash tili inson tiliga ancha yaqin va unda tuzilgan dasturni deyarli barcha operatsion tizimlarda ishga tushirish mumkin. Bu esa o'z navbatida har bir tizim uchun alohida dastur ishlab chiqish, o'z navbatida vaqt yo'qotishdan saqlaydi. Bundan tashqari bu tilni o'rgangan dasturchi boshqa zamonaviy dasturlash tillarini o'rganishi oson bo'ladi. Qo'llanmada android muhitida java dasturlash tilini o'rganish uchun nazariy ma'lumotlar, amaliy misollar va topshiriqlar, ko'p uchraydigan savollar bilan berilgan.

Ta'lim tizimida masoaviy ta'limning keng qo'llanilishi bugungi kunda elektron darsliklar va mobil ilovalarga talab oshganini ko'rsatadi. O'quvchi yoshlar va talabalarimiz darslarni online va offline tarzda shaxsiy kompyuter yoki mobil qurilmalarida olib bormoqdalar. Shu ma'noda ta'limda mobil ilovalarni yaratish bo'yicha tayyorlangan o'quv qo'llanma ham android muhitida java dasturlash tili va maxsus komponentalarni tez, oson va samarali o'rganishga yordam beradi.

Ushbu o'quv qo'llanma, android platformasi va undan foydalanish, android uchun maxsus instrumental dasturiy vositalarni o'rnatish va sozlash, android tizimi uchun java dasturlash tili, android ilovalarda hodisalar va jarayonlar va ularni bir-biri bilan bog'lash, androidda foydalanuvchi interfeysi, view va layoutlar, foydalanuvchi interfeysini yaratish, view lardan foydalanish, android ilovalarda rasm va menyularni boshqarish, mobil ilovalarda ma'lumotlar bazasi, SQLite asoslari mavzularidan va foydalangan adabiyotlar ro'yxatidan iborat bo'lib bir qancha mavzularni o'z ichiga oladi.

1-MAVZU: MOBIL ILOVALARNI YARATISH UCHUN DASTURLASH MUHITLARI VA DASTURLASH TILLARI. MOBIL ILOVALARNING HAYOT SIKLI

Android (yunoncha so'z bo'lib, ikki bo'g'ini — «erkak» va «o'xshash») — odam sifat robot ma'nosini ifoda etadi. Android operatsion tizimi logotipida robot tasvirlangani ham bejizmas. Android operatsion tizimining yaratilish tarixi 2002-yillardan boshlangan. Mana shu davrda Google korporatsiyasi yaratuvchilari E. Rubinning dasturiy ishlanmalari to'plami bilan qiziqib qoladilar. Dastlab mobil qurilmalar uchun yangi operatsion tizimni yaratish loyihasi bilan katta maxfiylik ostida TAndroid Inc. Kompaniyasi shug'ullangan, ushbu kompaniyani keyinchalik Google sotib oladi. Android — Linux yadrosiga asoslangan kommunikatorlar, planshetli kompyuterlar, elektron kitoblar, raqamli musiqa uskunalari, qo'l soatlari, netbooklar va smartbooklar uchun portativ (tarmoqli) operatsion tizimdir. Dastlab, Android Inc. kompaniyasi tomonidan yaratila boshlangan, uni keyinchalik Google sotib olgan. Keyinchalik Google Open Handset Alliance (OHA) alyansini tashkil qildi, u hozirda ham platformani qo'llab-quvvatlash va yanada rivojlantirish bilan shug'ullanadi. Android Google tomonidan ishlab chiqilgan kutubxona orqali qurilmani boshqaruvchi Java-ilovasini yaratishga imkon beradi. Android Native Development Kit Si va boshqa tillarda yozilgan ilovalarni yaratadi. 2012-yilning uchinchi choragida sotilgan smartfonlarning 75 foizida Android operatsion tizimi o'rnatilgan. HTC Dream (T-Mobile G1) — Android OT asosida birinchi smartfoni yangiliklar kiritilishi tarixi 2008-yil sentabr oyida birinchi versiyasi chiqarilgandan so'ng tizimga bir necha yangiliklar kiritilishi sodir bo'ldi. Ushbu yangiliklar odatda, aniqlangan xatolarni tuzatish va tizimga yangi funksiyani kiritish bilan bog'liq bo'ldi. Tizimning har bir versiyasi yangilik sifatida o'zining kodli nomi bilan ataldi. Kod nomlari alifbo tartibida berildi. 2012-yil noyabr oyiga kelib, tizimning 14 ta versiyasi yaratildi. Oxirgi versiyasi — 4.1 Jelly Bean («qo'shimchasi bilan chaynash obaki») deb nomlandi. HTC kompaniyasi tomonidan yaratilgan HTC Dream smartfoni (T-Mobile G1 nomi ostida rasman T-Mobile mobil aloqa operatori tomonidan yaratilgan) Android boshqaruvi ostida ishlovchi birinchi qurilma bo'ldi, uning taqdimoti 2008 - yil 23-sentabr kuni bo'lib o'tdi. Ko'p

o‘tmay, smartfonlar boshqa ishlab chiqaruvchilari tomonidan Android asosida qurilma ishlab chiqarish istagi bilan ko‘plab murojaatlar kelib tusha boshladi. Planshetlar uchun mo‘ljallangan Android uchinchi (Honeycomb) versiyasi chiqishi bilan borgan sari ko‘proq ishlab chiqaruvchilar ushbu platformada planshetlar ishlab chiqarishlarini e’lon qila boshladilar. Smartfonlar va planshetlardan tashqari Android operatsion tizimini boshqa qurilmalarga ham o‘rnata boshladilar. Masalan, 2009-yil oxirida Android asosida ishlovchi birinchi fotoramka savdoga chiqarildi. 2011-yil iyun oyida Italiyaning BlueSky kompaniyasi Android operatsion tizimi boshqaruvi ostida ishlovchi I’m Watch intellektual qo‘l soatlarini ishlab chiqarishini ma’lum qildi. 2012-yil avgust oyida Nikon Google platformasida ishlovchi jahonda birinchi foto kamerasini taqdim etdi. Bundan tashqari, tashabbuskor Androidni qator mashhur qurilmalarga ko‘chirib o‘tkazib joriy qildilar, ular orasida misol uchun, Windows Mobile HTC Touch Dual va HTC TyTN II platformasidagi smartfonlar bor, ularda Android emulyatsiya rejimida ishga solingan. Maemoda ishlovchi — Nokia N810 va Nokia N900 (Nitdroid nomli port) internet-planshetlari — Windows Mobile operatsion tizimida, MeeGo, va HTC HD2 platformasida ishlovchi Nokia N9 smartfonlari kabi qurilmalarga to‘laqonli ko‘chirib o‘tkazish ham amalga oshirildi, ularda Android operatsion tizimini micro SD-kartalar sifatida ichki NAND-xotira sifatida ham ishga solish mumkin. Shu bilan birga, o‘rnatilgan tizim to‘la, hech bir cheklanishlarsiz funksionallikga ega. Bulardan tashqari, Android operatsion tizimini Apple qurilmalariga — iPhone, iPod Touch va iPad larga Openiboot nomli maxsus dastur yordamida o‘rnatish muvaffaqiyatli tajribasi ham mavjud, u ushbu qurilmalarda turli operatsion tizimlarni, shu jumladan, Android operatsion tizimini ham ishga solish uchun mo‘ljallangan. Bada operatsion tizimidagi qurilmalarda cheklangan funksiyalari bilan dastlabki proshivkalari paydo bo‘lmoqda.

Koolu kompaniyasi Neo Free Runnerga Androidni o‘rnatish bilan shug‘ullanish bilan birga, qayta o‘rnatilgan Google mobil platformasi bilan ushbu smartfonlarni sotishda o‘z biznesini rivojlantirmoqda. Koolu kompaniyasidan Neo Free Runnerga birinchi rasmiy va umumiy foydalanish uchun Android o‘rnatilishi beta-relizi 2008-yil dekabr oyida bo‘lib o‘tdi. Android x86 arxitekturasiga ham ko‘chirib o‘tkazilgan. Ustunliklari: ba’zi sharhlovchilar Android qator

hollarda web-syorfing, Google Inc. servislari bilan mosligi kabi va boshqa xususiyatlari bilan o‘z raqobatchilaridan biri Apple iOS kompaniyasiga qaraganda o‘zini yaxshi namoyon qilishini aytadilar. Android, iOS ga nisbatan ochiq platforma hisoblanadi, bu holat unda ko‘proq funksiyalarni amalga oshirishga imkon beradi. iOS va Windows Phone 7 dan farqli ravishda, Androidda fayllarni qabul qilish va uzatishga ham imkon beruvchi Bluetooth oqimini to‘la amalga oshirish mavjud. FTP-serverini, tarmoqqa ulanish nuqtasi rejimi (PAN xizmati) va Bluetooth orqali guruhli birinchi darajali tarmoqni (GN xizmati) amalga oshirish mavjud. Android-apparatlarida, odatda, USB va xotira kartalarini olmasdan turib boshqa uzatish usullari tezlik cheklanishlaridan qat’iy nazar kompyuter fayllarini tezlikda telefonga ko‘chirishga imkon beruvchi, MicroSD-kardrider mavjud; bundan tashqari, iOS va Window Phone 7 sinxronlashtirish dasturi (iTunes va Zune), orqali amalga oshirishdan tashqari, biror-bir fayllarni telefonga/telefondan to‘g‘ridan-to‘g‘ri uzatish mumkin emas, Android operatsion tizimidagi telefonlar esa xotira kartasi fayllar tizimini USB mass storage device («fleshka») kabi uzatish imkoniga ega. Avvaldan dasturni "tekshirilmagan manbalardan" (misol uchun, xotira kartasidan) o‘rnatish taqiqlanishiga qaramay, ushbu cheklash apparat sozlashlarida doimiy vositalar yordamida o‘chiriladi, bu holat esa internet-ulanishlarsiz dasturlarni telefonlar va planshetlarga o‘rnatishga imkon beradi (misol uchun, Wi-Fi-ulanish nuqtalariga ega bo‘lmagan va odatda juda qimmat turadigan mobil internetga pul sarflashni istamaydigan foydalanuvchilar uchun), hamda barcha istaganlarga Android uchun ilovalarni bepul yozish va o‘z apparatida test sinovlaridan o‘tkazish imkonini beradi, shu bilan birga, iOS va Windows Phone 7 da hatto o‘z dasturlarini tarqatish istagi bo‘lmaganda ham loyihachining qayd etish ro‘yxatini sotib olishi kerak bo‘lardi. Android ARM, MIPS, x86 kabi turli apparatli platformalarda foydalanish mumkin. Ilovalar boshqa muqobil Google play magazinlari mavjud, misol uchun Amazon’dan Appstore for Android, Opera Store, Yandex.Store.

Hozirgi kunda OHA 34 kompaniyalarni birlashtiradi, ular orasida katta T-Mobile mobil aloqa operatorlari, HTC, Intel, Sprint Nextel, KDDI, NTT DoCoMo, China Mobile kabi mobil qurilmalar ishlab chiqaruvchilari, Broadcom, Marvell, NVIDIA, Qualcomm, SiRF, Texas Instruments, LG, Motorola, Samsung Electronics

mikrosxemalar loyihachilari hamda IT-industriyasi jahon giganti va birlashmaning g'oyaviy rahnamolaridan biri, Google kompaniyasi mavjud.

Dasturlashni endigina boshlovchilar yoki shu sohaga qiziquvchilar qaysi yo'nalishni tanlash kerak va uning yaqin kelajagi qanday degan savol qiziqtiradi. Albatta, tanlangan yo'nalish jamiyatga foydasi tegadigan, kelajagi bor va eng muhimi yuqori daromadga olib keladigan bo'lishi kerak.

Hozirgi kunda dunyo aholisining deyarli barchasi mobil qurilmalarga ega ekanligi hammaga ma'lum. Bu qurilmalarning juda katta qismi Android operatsion tizimida ishlaydi. Statistik ma'lumotlarga qaraganda, aqlli qurilmalar(smart devices)ning 80% dan ortiq qismi Android tizimida ishlaydi. Android tizimi keng imkoniyatli, dasturchilarga qulay, bepul va albatta ochiq kodlidir. 2012-yilning 3-choragida sotilgan smartfonlarning **75%** da Android operatsion tizimi o'rnatilgan edi. Bugungi kunga kelib **50 milliard**dan ortiq android ilovalari ro'yhatga olingan. Bu android tizimida ishlaydigan qurilmalarning ko'pligi va ular uchun ilovalarga talablarning yuqoriligini ko'rsatadi. AQSHning barcha shaharlaridagi android dasturchilari uchun yillik maosh o'rtacha yuqori sonlar oralig'ida qilib belgilangan. O'zbekiston sharoitida Siz boshqalar uchun dasturlarni yozib berish bilan tajribangizga qarab oylik $500 - $2 000 va unda ham ko'proq topa olasiz.

O'zbekiston sharoitida statistikalarga qaraganda dasturchilarning soni aholining 0.3% dan ham kamroq ekani ko'rsatmoqda, ya'ni 3000 atrofida degani. O'zbekiston sharoitidagi android ilovalarga bo'lgan ehtiyojlarni qondirish uchun ancha kam ko'rsatkich. Android tizimida dastur tuzishning qulayligi va imkoniyatlarni cheklanmaganligi bu tizimni ommalshishiga turtki bo'lmoqda.

Dasturlashni o'rganish uchun Siz aynan axborot texnologiyalar sohasida ishlashingiz shart emas. Buni istagan inson o'rganishi mumkin, Siz yurist, iqtisodchi, o'qituvchi yoki istalgan boshqa kasb egasi bo'lishingizdan qat'iy nazar o'rgana olishingiz mumkin. Bu Siz o'ylaganingizdek unchalik qiyin emas ham, unchalik murakkab jarayon ham emas. Faqat huddi hayotdagidek to'g'ri fikrlash, to'g'ri yondashishni bilsangiz bo'lgani. Umuman olganda Sizda mantiq bo'lsa va shu sohaga qiziqishingiz yuqori bo'lsa bo'ldi. Dasturlashni -

fikrlashni o'rgatadi ya'ni bu soxa bilan ishlagan odamning dunyoqarashi boshqalarnikidan ancha farq qiladi.

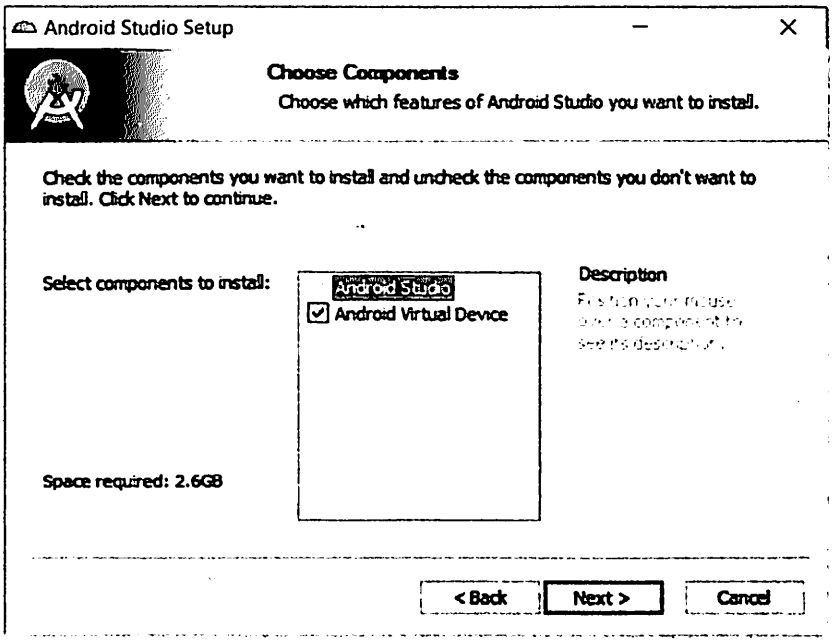
O'zbekistondagi Android dasturlash va uning taraqqiyotini inobatga olib, sizlarga ushbu **"Ta' limda mobil ilovalari"** o'quv qo'llanmasini tavsiya etamiz. Bu qo'llanmada siz ma'lum vaqt oralig'ida tizimli ravishda to'liq hajmda Android dasturlashni ta'lim soxasida qo'llash bo'yicha bilim va ko'nikmalarga ega bo'lishingiz mumkin bo'ladi. Bundan tashqari siz mazkur o'quv qo'llanmada yuqori darajali dasturlash tilida ishlash, ta'lim sohasida va xususiy tashkilotlar uchun mobil ilovalar ishlab chiqish metodlarini o'rganishingiz mumkin.

Android studioni o'rnatish

O'rnatish jarayonini boshlash uchun android-studio-ide-181.5056338-windows.exe-ni ishga tushiramiz. O'rnatuvchi bunga javoban 1.1-rasmda ko'rsatilgan Android Studio Setup dialog oynasini taqdim etdi.

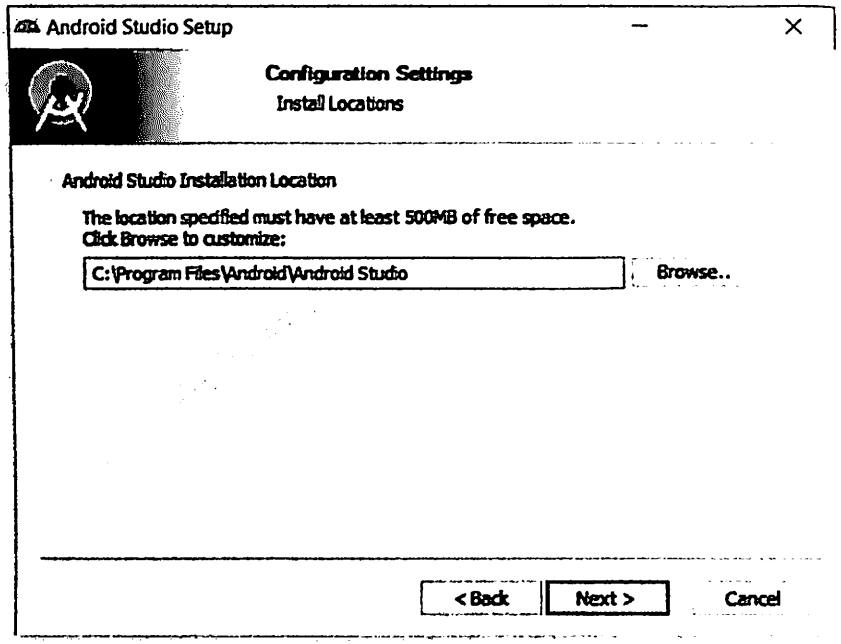
1.1-rasm. Android studioni o'rnatish

Keyingisini bosish orqali quyidagi panelga olib boriladi, bu yerda Android Virtual Device (AVD) ni o'rnatishni rad etish imkoniyati mavjud (1.2-rasm).



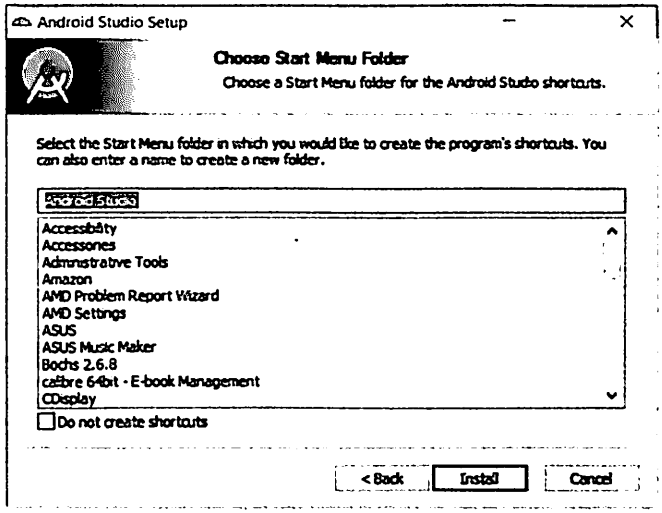
1.2-rasm. O'rnatishlarni belgilash

Standart sozlamalarni saqlashni tanlaymiz. Keyingisini bosgandan so'ng, konfiguratsiya sozlamalari paneliga olib boradi, u yerda Android Studio-ni qayerga o'rnatishni tanlashni so'raydi.



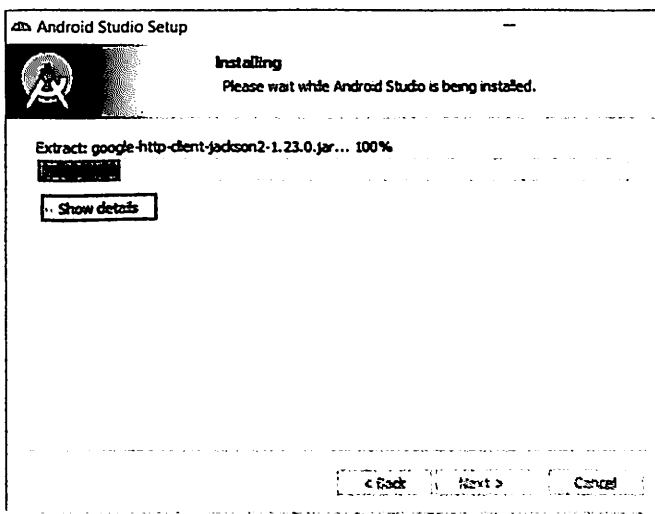
1.3-rasm. Dastur o'rnatish joyini ko'rsatish

Biz standart o'rnatish joyini saqlab qolamiz va "Next" tugmachasini bosamiz.



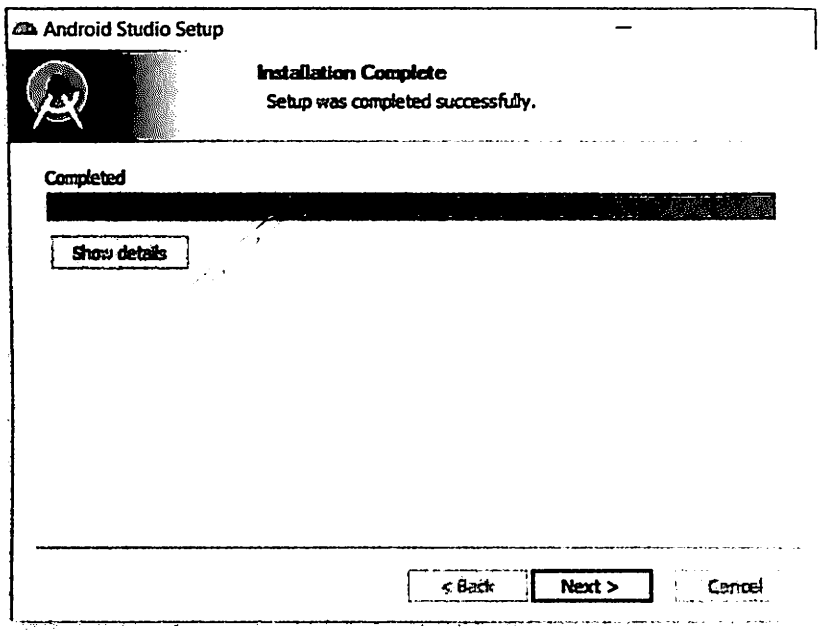
1.4-rasm. Menu papkasini tanlash.

Standart sozlamani saqlab qolamiz va Install tugmachasini bosamiz. Quyidagi o'rnatish paneli paydo bo'ldi:



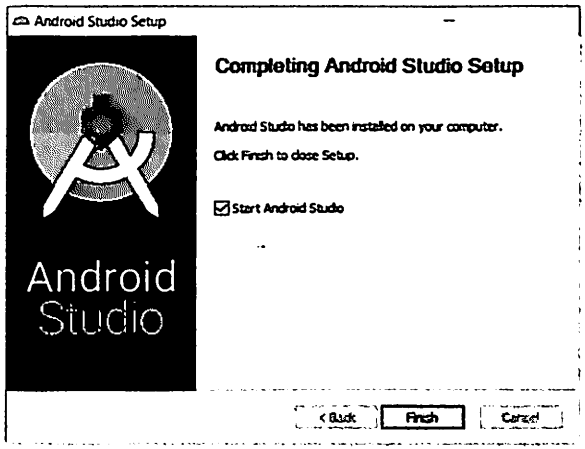
1.5-rasm. Dasturni o'rnatish jarayoni

Tavsilotlarni ko'rsatish tugmachasini bosish, o'rnatilgan fayllar nomlarini va boshqa faoliyatni ko'rsatishga olib keladi. O'rnatish tugagandan so'ng, *o'rnatish tugallandi* paneli paydo bo'ladi.



1.6-rasm. O'rnatishni yakunlash

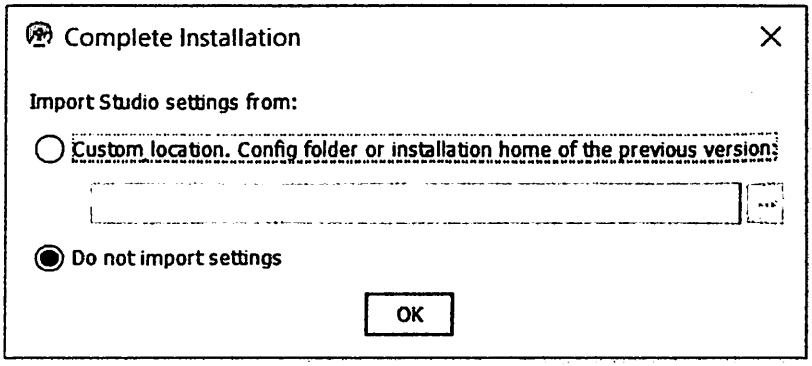
Keyingi tugmachasini bosgandan so'ng, o'rnatuvchi Completing Android Studio Setup panelini taqdim etadi.



1.7-rasm. Dasturni ishga tushirish

Android studioni ishga tushirish

Android Studio birinchi marta ishga tushirilganda, avvalgi o'rnatishdan sozlamalarni import qilish imkoniyatini taqdim etadigan "O'rnatishning to'liqligi" dialog oynasini taqdim etadi.

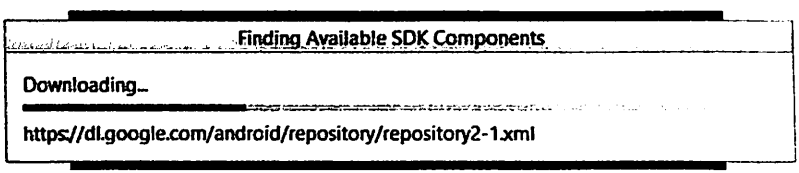


1.8-rasm. Sozlamalarni import qilish

Sozlamalarni import qilmaslikni tanlaymiz (standart tanlov) va OK tugmachasini bosamiz va quyidagi ekran hosil bo'ladi:

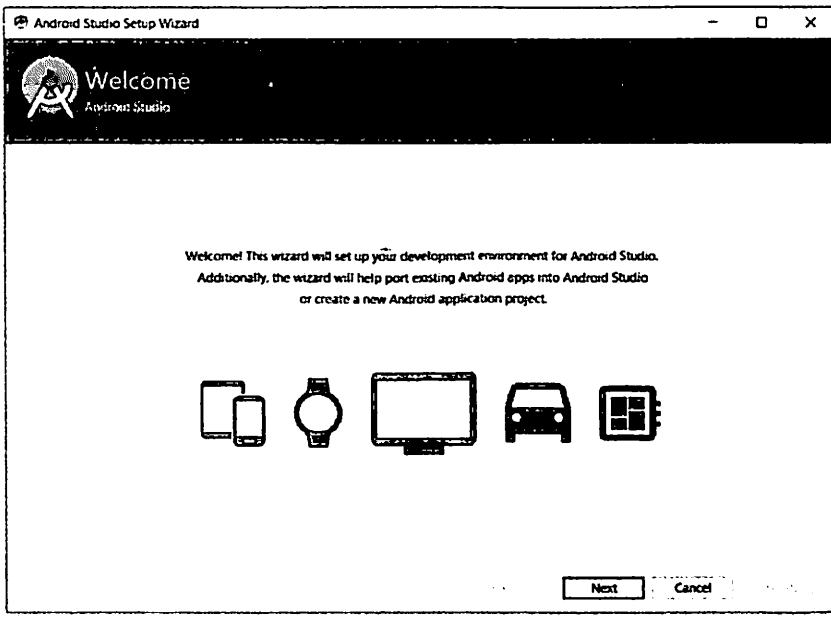
1.9-rasm. Dasturga kirish jarayoni

Shuningdek, biz quyidagi SDK komponentlarini topish xabarlarini ham ko'rishimiz mumkin.



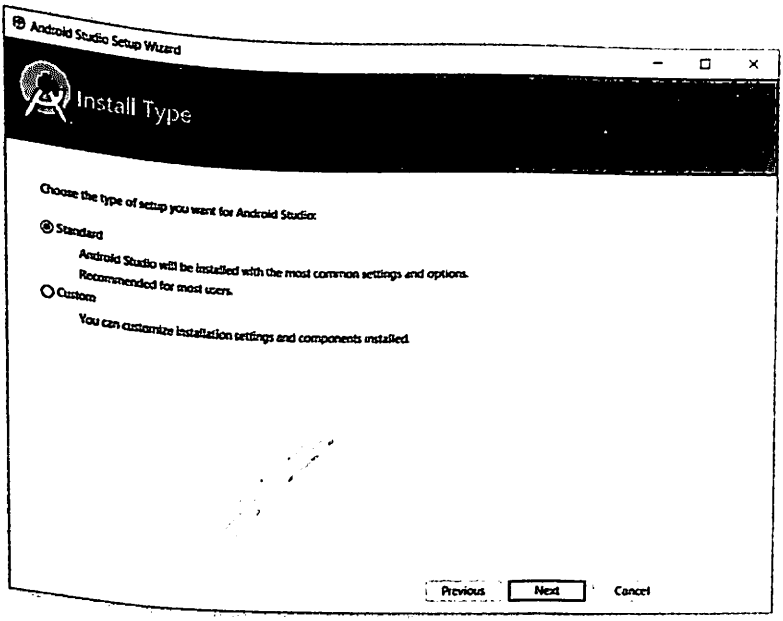
1.10-rasm. SDK komponentalarini yuklab olish

Shu nuqtada Android Studio quyidagi Android Studio Setup Wizard dialog oynasini taqdim etadi:



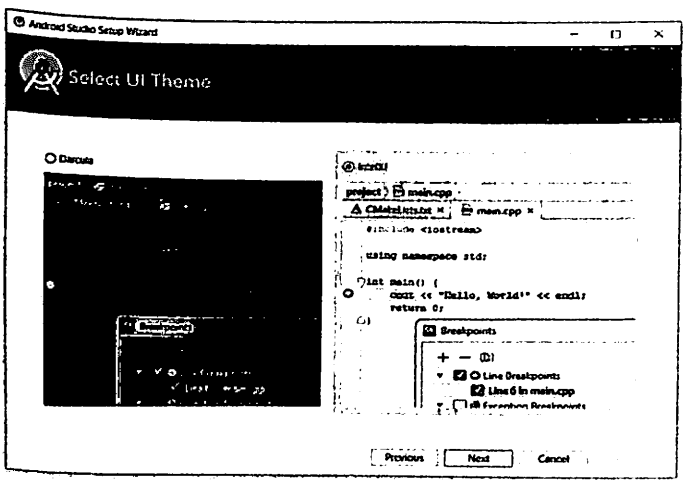
1.11-rasm. Android setup wizard oynasi

Bu joydan keyingisini bosamiz va wizard o'rnatish turini tanlashga taklif qiladi. Standart sozlamani saqlaymiz.



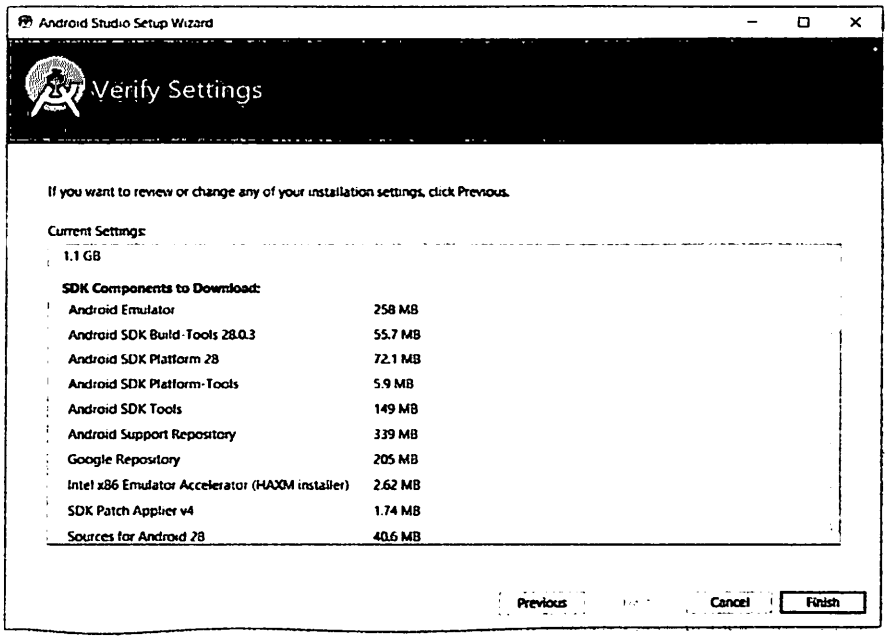
1.12-rasm. Install type (o'rnatish turi)

Keyin bizga foydalanuvchi interfeysi mavzusini tanlash imkoniyatini beriladi.



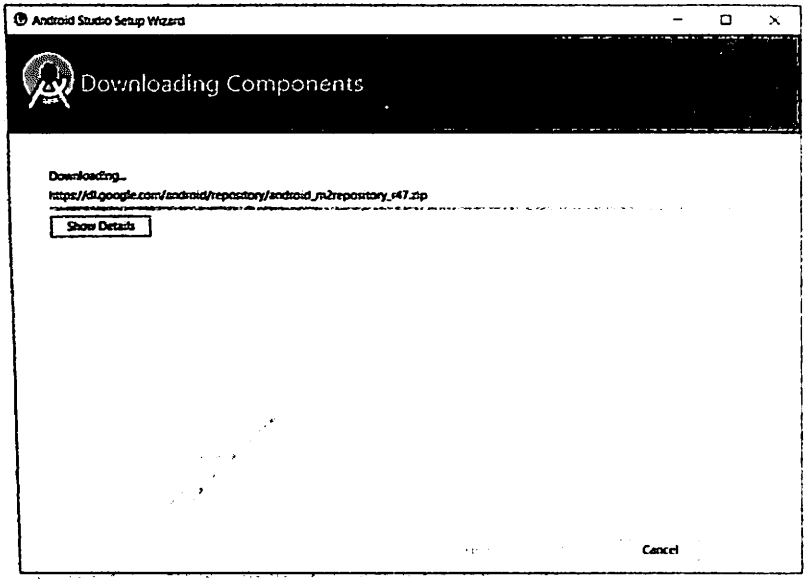
1.13-rasm. Foydalanuvchi interfeysini tanlash

Standart IntelliJ sozlamasini saqlaymiz va Next ni bosamiz. Keyinchalik Android Studio sozlamalarni tekshirish imkoniyatini taqdim etadi.



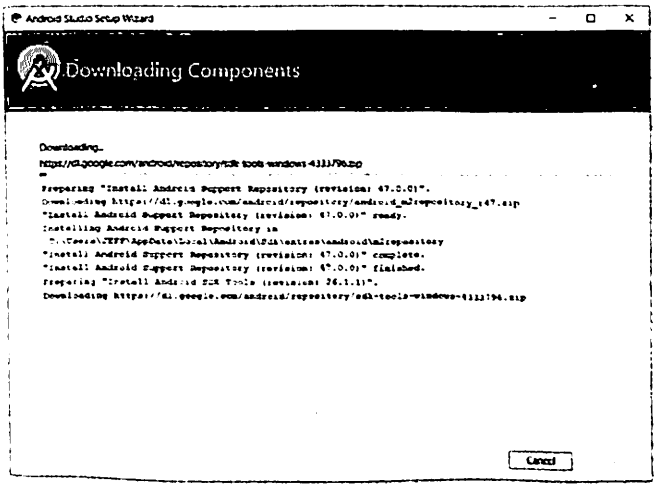
1.14-rasm. Sozlamalarni tekshirish

Finish tugmachasini bosamiz va Android Studio SDK komponentlarini yuklab olish jarayonini boshladi.



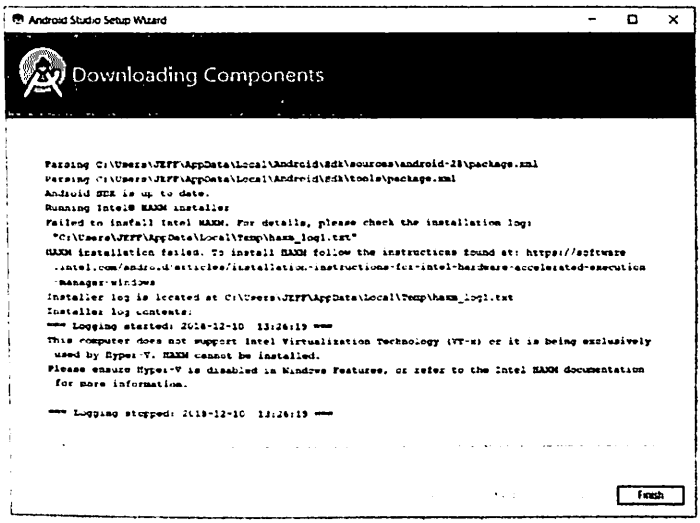
1.15-rasm. Komponentalarni yuklab olish

O'rnatishning ushbu qismini tugatish uchun bir necha daqiqa vaqt ketishi mumkin. "Show details" tugmachasini bosish, yuklab olinayotgan va ochilgan turli xil fayllarni ochib berib, qo'shimcha ma'lumotlarni taqdim qilishi mumkin.



1.16-rasm. Detallar oynasi

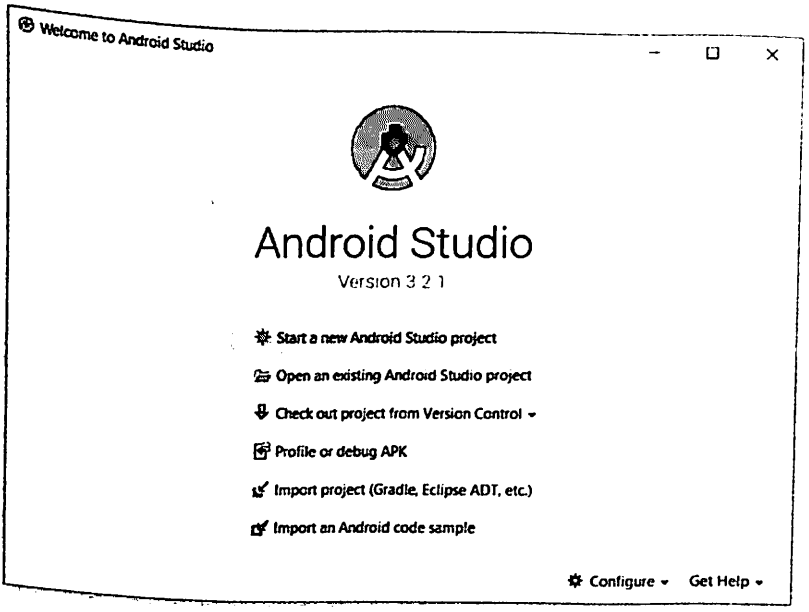
AMD-ga asoslangan kompyuter uchun komponentlar to'liq yuklab olingandan va ochilgandan keyin noxush kutilmagan hodisalar bo'lishi mumkin:



1.17-rasm. Yuklangan fayllar haqida ma'lumotlar oynasi

Nihoyat, wizardni bajarish uchun Finish tugmachasini bosamiz. *Android Studio dasturiga xush kelibsiz* muloqot oynasi paydo bo'ldi.

Ushbu dialog oynasi yangi Android Studio loyihasini ishga tushirish, mavjud loyiha bilan ishlash va boshqalar uchun ishlatiladi. Unga Windows Start menyusidan Android Studio-ni yoki boshqa platformadagi ekvivalentini tanlash orqali kirish mumkin.



1.18-rasm. Dastur bosh sahifasi

Hozirgacha o'rnatganimizdan, siz ham Android Studio-ga xush kelibsiz dialog oynasi bilan ishlaydigan Android Studio-ga ega bo'lishingiz kerak. Bu yerdan yangi Android Studio loyihasini boshlash-ni bosish orqali yangi loyihani ishlashga kirishamiz. Android Studio quyidagi ketma-ketlik bilan ko'rsatilgan yangi loyiha yaratish dialog oynasi bilan javob beradi.

Android Studioni sozlash

Android Studioni sozlash ishlari bir nechta bosqichlardan iborat bo'ladi. Shu bosqichlardan o'tgandan keyingina Android ilovasini yaratish mumkin bo'ladi.

> **Note:** this guide is based on Android Studio 2.2, but the process on other versions is mainly the same.

Loyiha sozlamalarini amalga oshirish

Asosiy sozlamalar (1.19-rasm)

Yangi loyihangizni ikki xil usul bilan boshlashingiz mumkin:

- Xush kelibsiz ekranidan New Android Studio loyihasini boshlash tugmasini bosish orqali.
- Agar sizda loyiha oldindan ochilgan bo'lsa: File → **New** Project.

So'ngra, quyidagilarni kiritish talab qilinadi:

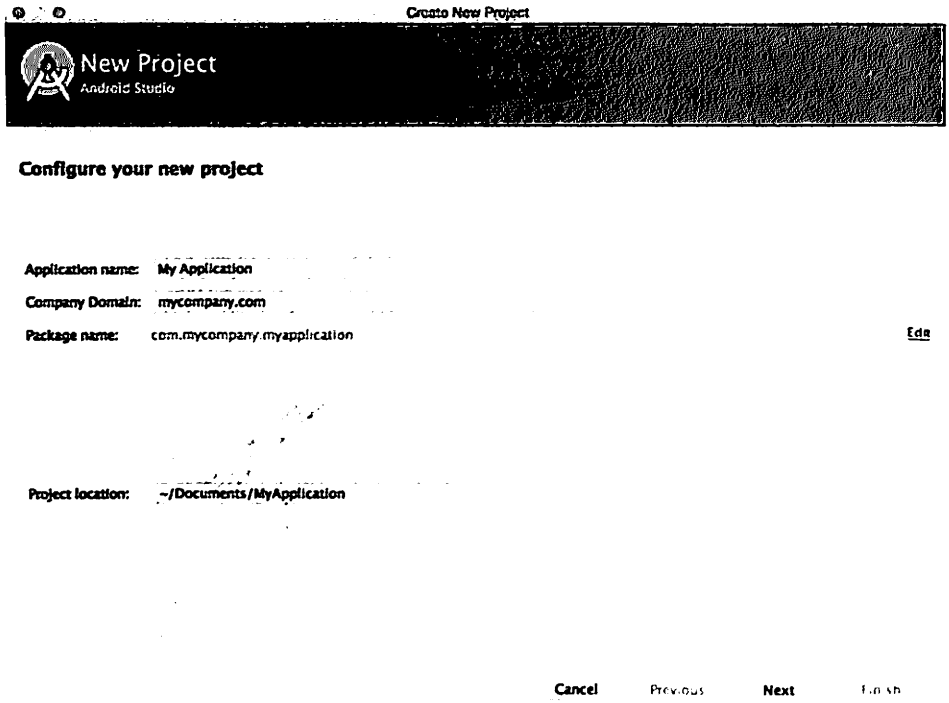
1. **Application Name** - Ushbu nom foydalanuvchiga ko'rsatiladi. Misol: Salom dunyo. Siz uni keyinchalik AndroidManifest.xml da o'zgartirishingiz mumkin.
2. **Company Domain** - Bu sizning loyihangizning to'plami nomi uchundir. Masalan: stackoverflow.com.
3. **Package Name** (applicationId-dastur id si) - Bu to'liq malakali loyiha to'plamining nomi.

   U *teskari domen nomi* yozuviga amal qilishi kerak *(aka Reverse DNS): Top Level Domain . Company Domain. [Company Segment .] Application Name.*

   Masalan: com.stackoverflow.android.helloworld yoki com.stackoverflow.helloworld. Siz har doim o'zingizning applicationId darajasiga qarab bekor qilish orqali o'zgartirishingiz mumkin.

   > Don't use the default prefix "com.example" unless you don't intend to submit your application to the Google Play Store. The package name will be your unique **applicationId** in Google Play.

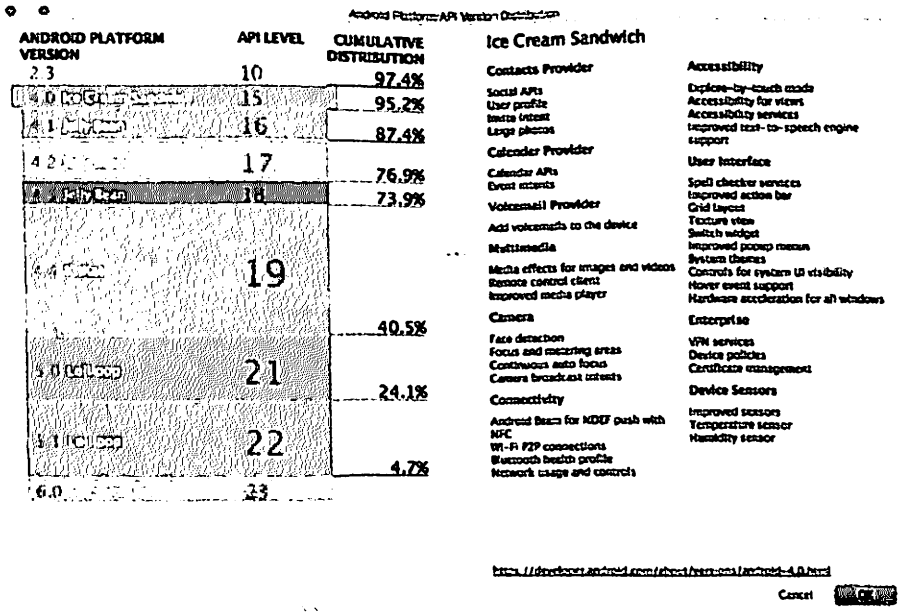
4. **Project Location** - Bu loyiha saqlanadigan katalog.



1.19-rasm. Loyiha sozlamalari

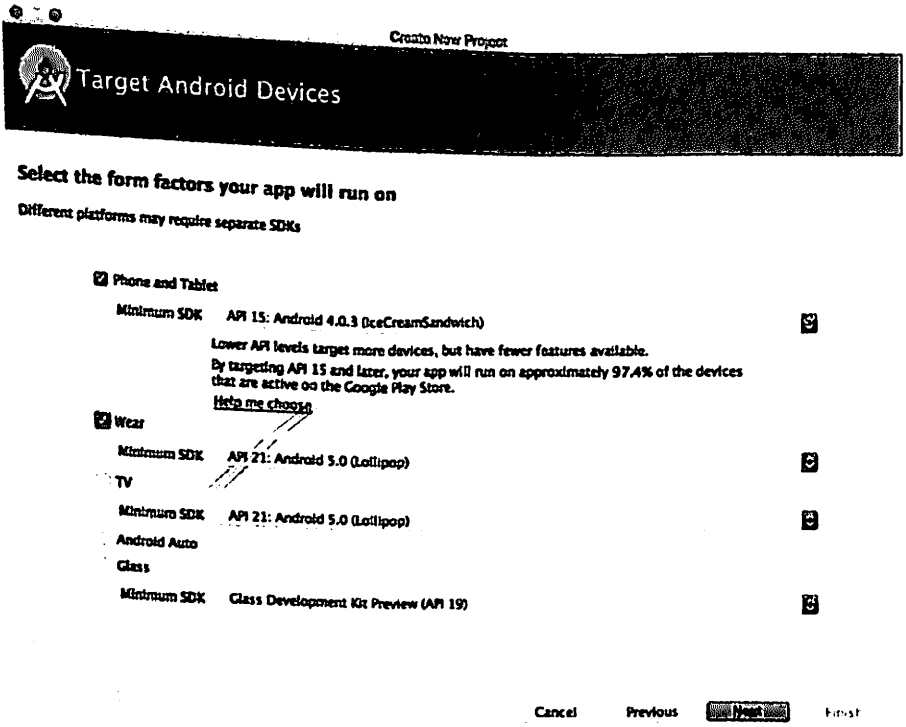
Factors and API Level formasini tanlang

Keyingi oyna (1.20-rasm) sizning ilovangiz tomonidan qo'llab-quvvatlanadigan shakl omillarini tanlashga imkon beradi, masalan phone, tablet, TV, Wear, va Google Glass. Tanlangan shakl omillari loyiha doirasida dastur moduliga aylanadi. Har bir form-faktor uchun siz ushbu dastur uchun API darajasini tanlang. Ko'piroq ma'lumot olish uchun, **Help me choose** ni bosing.



1.20-rasm. API darajalari.
Android-ning joriy tarqatish jadvali, ***Help me for choose*** *bosganingizda ko'rsatiladi.*

Android Platform Distribution oynasi Android-ning har bir versiyasida ishlaydigan mobil qurilmalarning tarqalishini ko'rsatadi (1.21-rasm). Android-ning mos keladigan versiyasida kiritilgan xususiyatlar ro'yxatini ko'rish uchun API darajasini bosing. Bu sizning ilovalaringiz uchun zarur bo'lgan barcha xususiyatlarga ega bo'lgan minimal API darajasini tanlashingizga yordam beradi, shuning uchun iloji boricha ko'proq qurilmalarga kirishingiz mumkin. Keyin **OK** tugmasini bosing. Endi Android SDK-ning qaysi platformalari va versiyasini qo'llab-quvvatlashini tanlang.



1.21-rasm. Mobil qurilmalarni tanlash

Hozircha faqat phone va tablet ni tanlang.

Minimal SDK - bu sizning ilovangizning pastki chegarasi. Bu ilova qaysi qurilmalarga o'rnatilishini aniqlash uchun Google Play Store-dan foydalanadigan signallardan biridir.

ADDITIONAL INFORMATION

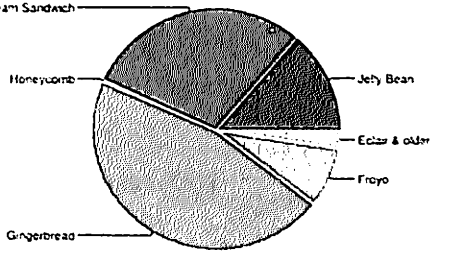
Updated September 28, 2016	**Installs** 100,000 - 500,000	**Current Version** 1.0.89
Requires Android 4.1 and up	**Content Rating** Rated for 12+ Parental Guidance Recommended Learn more	**Interactive Elements** Users Interact
Permissions View details	**Report** Flag as inappropriate	**Offered By** Stack Exchange

Android Studio sizga (taxminan) minimal SDK-ni hisobga olgan holda qurilmalarning qaysi foizida qo'llab-quvvatlanishini aytib beradi.

> Lower API levels target more devices but have fewer features available.

Minimal SDK haqida qaror qabul qilishda siz paneldagi statistikani hisobga olishingiz kerak (1.22-rasm), bu sizga so'nggi bir hafta ichida Google Play do'koniga global tashrif buyurgan qurilmalar haqida ma'lumot beradi.

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	2.2%
2.2	Froyo	8	8.1%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	45.4%
3.1	Honeycomb	12	0.3%
3.2		13	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.0%
4.1	Jelly Bean	16	12.2%
4.2		17	1.4%

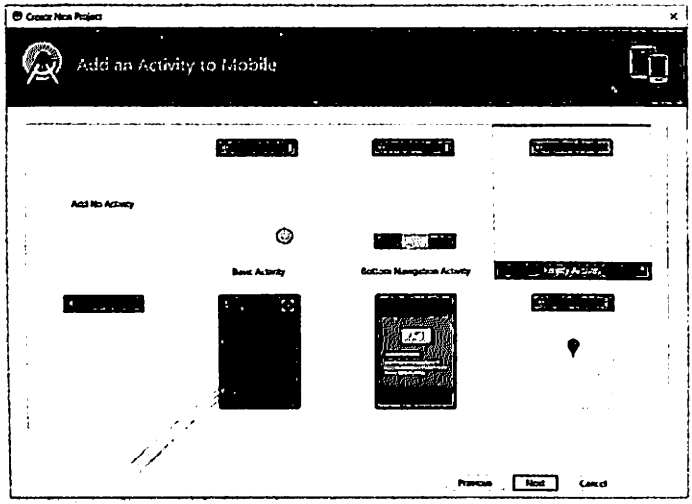


Data collected during a 14-day period ending on February 4, 2013

1.22-rasm. SDK ma'lumotlari

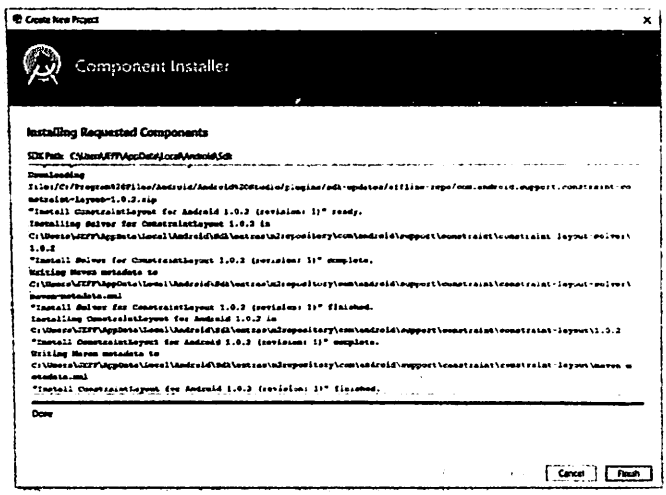
Android Studio biz yaratgan har bir dastur uchun form-faktorlarni yoki maqsadli qurilmalar toifalarini tanlashga imkon beradi. Standart sozlamani saqlaymiz.

Next-ni bosamiz, shunda bizga ilovamizning asosiy faoliyati uchun shablonni tanlash imkoniyati beriladi. Hozircha biz "Empty activity" (1.23-rasm) bilan shug'ullanamiz. Ushbu shablonni tanlaymiz(agar kerak bo'lsa) va Next-ga bosamiz.



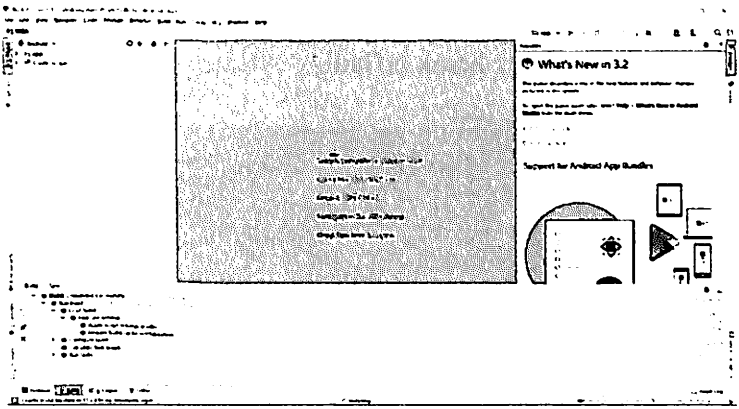
1.23-rasm. Activityni tanlash.

Android Studio-dan birinchi marta foydalanganingizda, uning foydalanuvchi interfeyslarini yaratish uchun foydalaniladigan cheklangan joylashuvi bilan bog'liq ba'zi fayllarni yuklab olish kerakligini bilib olasiz.



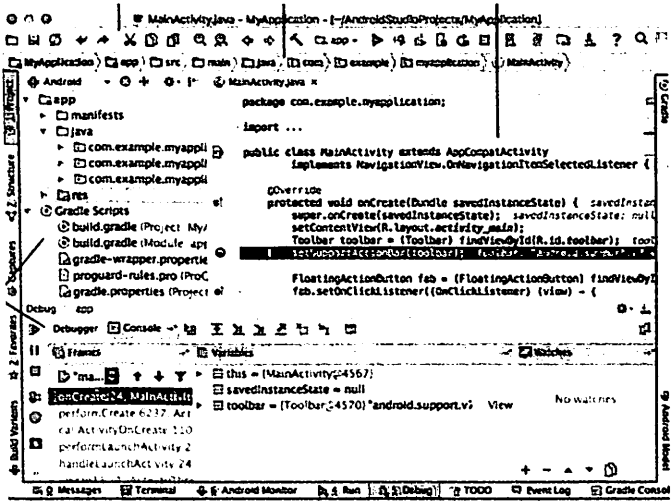
1.24-rasm. Talab qilingan komponentalarni yuklash

Android Studio cheklangan tartib fayllarini yuklab olgandan so'ng Finish-ni faollashadi. Ushbu tugmani bosamiz va Android Studio bizni asosiy oynaga olib boradi.



1.25-rasm. Android Studio asosiy ishchi oynasi

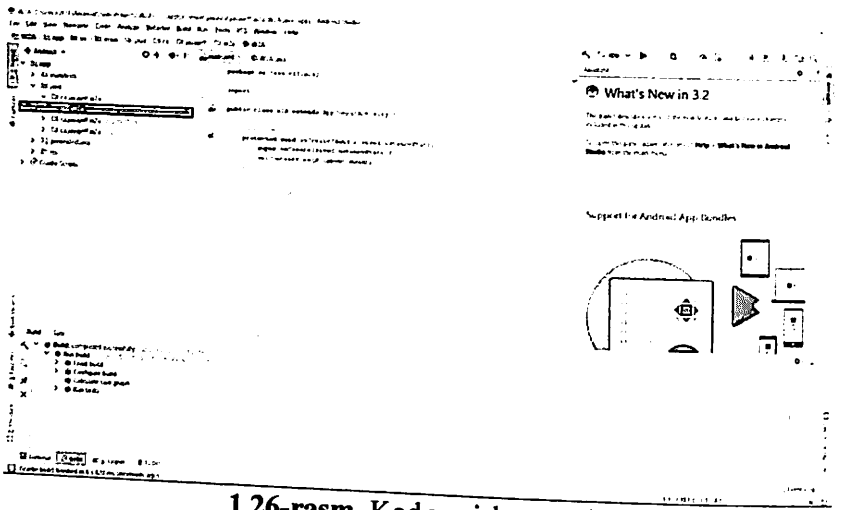
Asosiy yuqoridagi rasmlarda ko'rsatilgan menyu qatoriga va boshqa bir nechta maydonlarga bo'linadi.



1.26-rasm. Dastur interfeysi bilan tanishish

Loyiha va tahrirlash oynalari

Asosiy oynaga kirganingizda (1.25-rasmga qarang), faqatgina ilova va Gradle skriptlarini ko'rsatadigan Project oynasini kuzatasiz. Batafsil ma'lumotni ko'rish uchun loyiha sturkturasining ilova bo'limini kengaytirishingiz kerak bo'ladi.



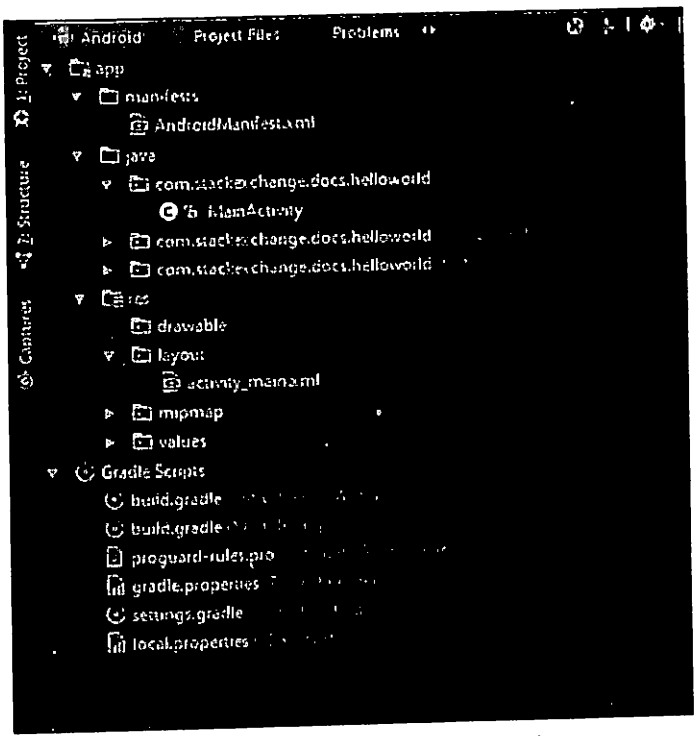
1.26-rasm. Kod yozish oynasi

Activity qo'shish

Endi biz dasturimiz uchun standart faoliyatni tanlaymiz. Android-da, Faoliyat - bu foydalanuvchiga taqdim etiladigan bitta ekran. Ilova bir nechta tadbirlarni o'tkazishi va ular orasida harakatlanishi mumkin. Bu misol uchun, bo'sh Activityni tanlang. Agar xohlasangiz, bu yerda faoliyat nomi va tartibini o'zgartirishingiz mumkin. Yaxshi amaliyot shundan iboratki, Faoliyatni faoliyat nomi uchun qo'shimchalar va faoliyatni maket nomi uchun prefiks sifatida saqlash hisoblanadi. Agar biz bularni shu bo'yicha qoldirsak, Android Studio biz uchun MainActivity deb nomlangan faoliyatni va "main" deb nomlangan maket yaratadi. Keyin **Finish** tugmasini bosamiz. Android Studio bizning loyihamizni yaratadi va tasdiqlaydi, bu tizimga qarab biroz vaqt talab qilishi mumkin.

Loyihani tekshirish

Android-ning qanday ishlashini bilish uchun biz uchun yaratilgan ba'zi narsalarni ko'rib chiqamiz. Android Studio-ning chap oynasida (1.27-rasm), bizning Android application structure ni ko'rishimiz mumkin.



1.27-rasm. Dastur strukturasi

Birinchi AndroidManifest.xml fayliga sichqonchani ikki marta bosamiz. Android manifesti Android ilovasi haqida ba'zi asosiy ma'lumotlarni tavsiflaydi. Unda bizning faoliyatimiz to'g'risidagi deklaratsiya mavjud. Shuningdek, ba'zi bir rivojlangan komponentalar haqida ham bilish mumkin.

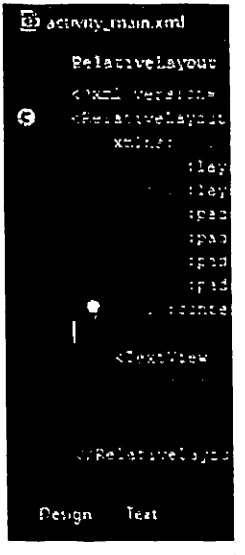
Agar ilova ruxsat bilan himoyalangan xususiyatga kirishni talab qilsa, u manifestda **<uses permission>** elementi bilan ruxsat talab qilinishini e'lon qilishi kerak. So'ngra, dastur qurilmaga o'rnatilgandan

so'ng, o'rnatuvchi talab qilingan ruxsatni berish yoki bermaslikni tekshirish va ba'zi hollarda foydalanuvchidan so'rab belgilaydi. Ilova shuningdek o'z tarkibiy qismlarini (faoliyat, xizmatlar, translyatsiya qabul qiluvchilari va kontent-provayderlar) ruxsat bilan himoya qilishi mumkin. Unda Android tomonidan berilgan (android.Manifest.permission ro'yxatiga kiritilgan) yoki boshqa ilovalar tomonidan e'lon qilingan har qanday ruxsatlardan foydalanish mumkin yoki bu o'z-o'zidan paydo bo'lishi mumkin.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Keyin, app/src/main/res/layout/ da joylashgan activity_main.xml faylini ochamiz. Ushbu sahifada MainActivity-ning komponentlari uchun deklaratsiyalar mavjud va siz bu yerda visual dizaynni ko'rasiz. Bu elementlarni tanlangan maketga sudrab olib tashlashga imkon beradi.

Shuningdek, Android Studio-ning pastki qismidagi "Text" tugmachasini bosish orqali xml layout dizayneriga o'tishingiz mumkin (1.28-rasm):



1.28-rasm. .xml layout dizayneri

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.stackexchange.docs.helloworld.MainActivity">

     <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

Ushbu maket ichida TextView deb nomlangan vidjetni ko'rasiz, android: text xususiyati. Bu foydalanuvchi dasturni ishga tushirganda ko'rsatiladigan matn blokidir.

Keling, MainActivity-ni ko'rib chiqamiz. Bu MainActivity uchun yaratilgan Java kodidir.

```
public class MainActivity extends AppCompatActivity {

    // The onCreate method is called when an Activity starts

    // This is where we will set up our layout
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // setContentView sets the Activity's layout to a specified XML layout
        // In our case we are using the activity_main layout
        setContentView(R.layout.activity_main);
    }
}
```

Android manifestimizdagi kabi, foydalanuvchi HelloWorld dasturini ishga tushirganda MainActivity sukut bo'yicha ishga tushadi.

app / da joylashgan build.gradle nomli faylni ochamiz.

Android studiyasi Android dasturlari va kutubxonalarini kompilyatsiya qilish va yaratish uchun Gradle tizimidan foydalanadi.

```
apply plugin: 'com.android.application'

android {
signingConfigs {
        applicationName {
            keyAlias 'applicationName'
            keyPassword 'password'
            storeFile file('../key/applicationName.jks')
            storePassword 'anotherPassword'
        }
    }
    compileSdkVersion 26
    buildToolsVersion "26.0.0"

    defaultConfig {
        applicationId "com.stackexchange.docs.helloworld"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:26.0.0'
}
```

Ushbu ma'lumotlar tarkibiga va sizning ilova versiyangizga oid ma'lumotlarni o'z ichiga oladi va tashqi kutubxonalarga bog'liqlik qo'shish uchun ham foydalanishingiz mumkin. Hozircha hech qanday o'zgarish qilmaymiz.

Har doim bog'liqliklar uchun mavjud bo'lgan so'nggi versiyani tanlash tavsiya etiladi:
buildToolsVersion: 26.0.0
com.android.support:appcompat-v7: 26.0.0
firebase: 11.0.4 (August 2017)

compileSdkVersion

compileSdkVersion - bu sizning ilovangizni kompilyatsiya qilish uchun Android SDK-ning qaysi versiyasi bilan Gradle-ga aytib berishning usulidir. Keyin yangi Android SDK-dan foydalanish ushbu darajaga qo'shilgan har qanday yangi API-lardan foydalanish talabidir.

Shuni ta'kidlash kerakki, compileSdkVersion-ni o'zgartirish ish vaqti xatti-harakatlarini o'zgartirmaydi. CompileSdkVersion-ni o'zgartirganda yangi kompilyator ogohlantirishlari / xatolari bo'lishi mumkin bo'lsa-da, sizning compileSdkVersion sizning APK-ga kiritilmagan: u faqat kompilyatsiya vaqtida ishlatiladi. Shuning uchun har doim eng so'nggi SDK bilan kompilyatsiya qilish tavsiya etiladi. Mavjud kod bo'yicha yangi kompilyatsiya tekshiruvlarining barcha afzalliklarini olasiz, eskirgan API-lardan saqlanasiz va yangi API-lardan foydalanishingiz mumkin bo'ladi.

minSdkVersion

Agar compileSdkVersion siz uchun mavjud bo'lgan eng yangi API-larni o'rnatadigan bo'lsa, minSdkVersion sizning ilovangiz uchun pastki chegaradir. MinSdkVersion - bu Google Play do'koni foydalanuvchi qaysi qurilmalariga dastur o'rnatilishi mumkinligini aniqlash uchun foydalanadigan signallardan biridir.

Rivojlanish jarayonida u ham muhim rol o'ynaydi: vaqt bo'yicha lint sizning loyihangizga qarshi ishlaydi va minSdkVersion-dan yuqori bo'lgan har qanday API-dan foydalanganda sizni ogohlantiradi va mavjud bo'lmagan API-ga habar qilishga urinish ishi muammosidan qochishga yordam beradi. Tizim versiyasini ishlash

vaqtida tekshirish API-lardan faqat yangi platforma versiyalarida foydalanishda keng tarqalgan usuldir.

targetSdkVersion

targetSdkVersion - bu Android-ning maqsadga muvofiqligini ta'minlashning asosiy usuli, agar targetSdkVersion yangilanmasa, xatti-harakatlarning o'zgarishini qo'llamaydi. Bu xatti-harakatlar o'zgarishi bilan ishlashdan oldin yangi API-lardan foydalanishga imkon beradi. Eng so'nggi SDK-ni yangilash har bir dastur uchun eng muhim vazifa bo'lishi kerak. Bu sizga kiritilgan har bir yangi xususiyatdan foydalanishingiz kerak degani emas va sinovsiz targetSdkVersion-ni testsiz yangilashingiz kerak.

targetSDKVersion - bu mavjud vositalar uchun yuqori chegara bo'lgan Android versiyasi. Agar targetSDKVersion 23 dan kichik bo'lsa, dastur API 23+ da ishlayotgan bo'lsa ham, ilova ish vaqti davomida ruxsat so'rashi shart emas. TargetSDKVersion, tanlangan Android versiyasi ustidagi android versiyalarining dasturni ishlashiga to'sqinlik qilmaydi.

Ilovani ishga tushirish

Bizning HelloWorld ilovamizni ishga tushiramiz. Siz Android virtual qurilmasida oson ishga tushirishingiz mumkin (uni Android Studio-da AVD menejeri yordamida o'rnatishingiz mumkin, quyida keltirilgan misolda aytib o'tilganidek) yoki o'zingizning Android qurilmangizni USB kabeli orqali ulashingiz mumkin.

Android qurilmasini sozlash

Android qurilmangizda Android Studio-dan dasturni ishga tushirish uchun qurilmangiz sozlamalarida USB disk optsiyasi-ni yoqishingiz kerak.

Settings > Developer options > USB debugging

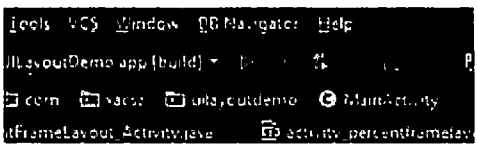
Agar sozlamalarda parametrlari ko'rinmasa, "Telefon haqida" bandiga o'ting va qurilish raqamini yetti marta bosing. Bu sizning sozlamalaringizda Tuzuvchi parametrlarini ko'rsatishni yoqadi.

Settings > About phone > Build number

Qurilmangizda mavjud bo'lgan versiyaga asos solish uchun build.gradle kelishuvini o'zgartirishingiz kerak bo'lishi mumkin.

Android Studiodan ishga tushirish

Android Studio ustki qismida joylashgan asboblar panelidagi "Yashil" tugmasini bosing (1.29-rasm). Ko'rsatilgan oynada dasturni qaysi qurilmada ishga tushirishni xohlayotganingizni tanlang (agar kerak bo'lsa Android Virtual Device-ni ishga tushiring yoki agar o'rnatishingiz kerak bo'lsa AVD (Android Virtual Device) ni sozlash) ga qarang) va OK tugmasini bosing.



1.29-rasm. Run tugmasi

Android 4.4 (KitKat) va undan yuqori versiyalari bilan ishlaydigan qurilmalarda USB disk uchun pop-up ko'rsatiladi. Qabul qilish uchun OK tugmasini bosildi.

Endi dastur Android qurilmangizda yoki emulyatoringizda o'rnatiladi va ishlaydi.

APK fayl joylashuvi

Siz dasturingizni qayta ishlashga tayyorlaganingizda, so'rovning ishonchliligini, tuzilishini va sinov versiyasini sinovdan o'tkazasiz. Murojaat vazifalari sodda bo'lib, ular dasturni optimallashtirishga yordam beradigan asosiy kodlarni tozalash va kodlarni o'zgartirish vazifalarini o'z ichiga oladi. Dastur yaratilish jarayoni disk raskadrovka tuzish jarayoniga o'xshaydi va JDK va Android SDK vositalari yordamida amalga oshirilishi mumkin. Sinov vazifalari haqiqiy tekshiruv bo'lib xizmat qiladi, bu sizning so'rovingiz real sharoitda kutilganidek bajarilishini ta'minlaydi. So'rovingizni chiqarishga tayyorlashni tugatganingizda, tayyor APK-ga ega bo'lasiz,

uni to'g'ridan-to'g'ri foydalanuvchilarga tarqatishingiz yoki Google Play kabi ilovalar bozori orqali tarqatishingiz mumkin.

Android Studio

Yuqoridagi misollarda Gradle ishlatilganligi sababli, yaratilgan APK faylining joylashuvi: **<Sizning** Project Location>/app/build/outputs/apk/app-debug.apk

IntelliJ IDE

Agar siz Studioga o'tishdan oldin IntelliJ foydalanuvchisiz va to'g'ridan-to'g'ri IntelliJ loyihangizni import qilsangiz, unda hech narsa o'zgarmagan. Chiqish joyi quyidagicha bo'ladi:
out/production/...
Habar: bu ba'zida 1;0 atrofida eskiradi

Eclipse

Agar siz to'g'ridan-to'g'ri Android Eclipse loyihasini import qilsangiz, buni qilmang. Loyihangizda (jar fayllar yoki kutubxona loyihalari) bog'liqliklar paydo bo'lishi bilanoq, bu ishlamaydi va sizning loyihangiz to'g'ri sozlanmaydi. Agar sizda bog'liqliklar bo'lmasa, apk sizning Eclipse-da xuddi shu joyda joylashgan bo'ladi:
bin/...

Android Studioni sozlash

Android Studio - bu Android tomonidan ishlab chiqilgan IDE, bu Google tomonidan qo'llab-quvvatlanadi va tavsiya etiladi. Android Studio Android SDK menejeri bilan ta'minlangan, bu dasturlarni ishlab chiqishni boshlash uchun zarur bo'lgan Android SDK komponentlarini yuklab olish vositasi hisoblanadi.

Android Studioni o'rnatish va Android SDK uskunalari bilan tanishish:

1. Android Studio ni yuklab olish va o'rnatish.
2. SDK Tools oxirgi versiyasini yuklab olinadi va SDK Platform-tools Android Studio yordamida ochiladi, keyin Android SDK Tool Updates ko'rsatmalari bajariladi. Siz so'nggi mavjud barqaror paketlarni o'rnatishingiz kerak bo'ladi.

> *If you need to work on old projects that were built using older SDK versions, you may need to download these versions as well*

Android Studio 2.2-dan beri so'nggi OpenJDK-ning nusxasi o'rnatish bilan birga keladi va barcha Android Studio loyihalari uchun recommended JDK JDK (Java Development Kit) hisoblanadi. Bu Oracle-ning JDK paketini o'rnatish talabini bekor qiladi. Birlashtirilgan SDK-dan foydalanish uchun quyidagicha harakat qiling;

1. Android Studio-da o'z loyihangizni oching va menyu satrida **File>Project** tarkibini tanlang.
2. **SDK** joylashuvi sahifasida va **JDK** joylashuvi ostida Ichki JDKdan foydalanish katagiga belgi qo'ying.
3. **OK** ni bosamiz.

Tema qo'shish va o' zgartirish

preference.File->Settings->Editor->Colors & Fonts-> larni o'z xohishingiz bo'yicha o'zgartirishingiz va theme ni tanlashinigz mumkin. Yangi themes ni http://color-themes.com/ ilovasidan ham yuklab olishingiz mumkin. .jar.zip faylini yuklaganingizdan keyin, File -> **Import** ga o'tib Settings va yuklanganlarni tanlashingiz kerak.

Ilovalarni kompilyatsiya qilish

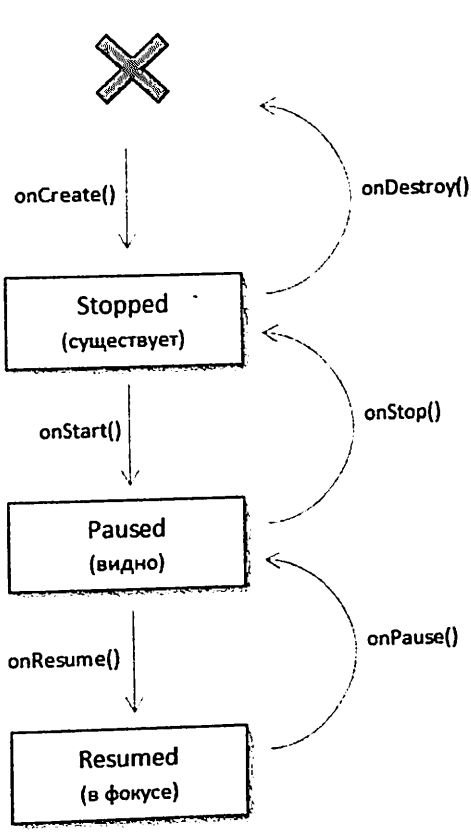
Android Studio-da yangi loyiha yarating yoki mavjud loyihani oching va uni ishga tushirish uchun yuqori asboblar panelidagi yashil Play tugmasini bosing. Agar kulrang bo'lsa, siz Android Studio-ga ba'zi bir qismlarni to'g'ri indeksatsiyalashga ruxsat berish uchun bir soniya kutishingiz kerak, bu jarayonni pastki holat satrida ko'rish mumkin.

Agar siz qobiqdan loyiha yaratmoqchi bo'lsangiz, Android Studio tomonidan avtomatik ravishda yaratiladigan local.properties fayli mavjudligiga ishonch hosil qiling. Agar sizga loyihani Android Studio siz yaratishingiz kerak bo'lsa, sizga sdk.dir = bilan boshlanadigan satr kerak bo'ladi, so'ngra SDK o'rnatishingiz uchun yo'l kerak.

Qobiqni oching va loyihaning katalogiga o'ting. ./Gradlew aR kiriting va enter tugmasini bosing. aR - assembleRelease uchun yorliq, bu siz uchun barcha bog'liqliklarni yuklab oladi va dasturni yaratadi. Asosiy APK dasturi ProjectName / ModuleName / build / outputs / apk da bo'ladi va ModuleName-release.apk deb nomlanadi.

Mobil ilovalarda hayot sikli:

Faoliyat bir holatdan ikkinchisiga o'tganda, tizim o'zining turli xil usullarini chaqiradi, ularni biz o'z kodimiz bilan to'ldirishimiz mumkin. Uni quyidagicha sxematik tasvirlash mumkin:



1.30-rasm. Hayot siklining ishlash jarayoni.

Shunday qilib, biz tizim chaqiradigan quyidagi faoliyat usullariga egamiz:

onCreate() - faoliyat birinchi marta yaratilganda chaqiriladi

onStart() - faoliyat foydalanuvchiga ko'rinmasidan oldin chaqiriladi

onResume() - foydalanuvchi faoliyati (o'zaro ta'sir) uchun mavjud bo'lguncha chaqiriladi

onPause() - boshqa faoliyat ko'rsatilishidan oldin chaqiriladi

onStop() - faoliyat foydalanuvchi uchun ko'rinmaydigan bo'lib qolganda chaqiriladi

onDestroy() - Faoliyat yo'q qilinishidan oldin chaqirilgan

Ushbu usullar holat o'zgarishiga olib kelmaydi. Aksincha, faoliyat holatining o'zgarishi ushbu usullarni chaqiruvchisidir. Shunday qilib, o'zgarish haqida bizga xabar beriladi va biz shunga muvofiq

munosabat qilishimiz mumkin. Keling, ushbu usullarning qachon va qanday tartibda chaqirilishini amalda ko'rib chiqamiz.

Amaliyot

Ushbu qo'llanmada biz ekran yo'nalishini o'zgartirish hodisasini taqlid qilishimiz kerak bo'ladi. Ammo Android 2.3 operatsion tizimiga ega emulyator buni egri bajaradi, shuning uchun biz loyihada 2.2 versiyasidan foydalanamiz. Buni amalga oshirish uchun 2.2 versiyasiga muvofiq yangi AVD yaratishingiz kerak, loyiha yarating (biz Android 2.2 dan foydalanamiz):

Project name: P0231_OneActivityState
Build Target: Android 2.2
Application name: OneActivityState
Package name: uz.mobililova.develop.acvitystate
Create Activity: MainActivity

Biz Layoutni o'zgartirmaymiz, bu hozir biz uchun muhim emas. MainActivity.java-ni oching, odatdagidek standart kod mavjud:

```
package uz.mobililova.develop.acvitystate;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Biz sxema bo'yicha bizga tanish bo'lgan onCreate usuli amalga oshirilganligini ko'ramiz. Shunga qaramay, ushbu usul Faoliyat yaratmasligini tushunish muhimdir.

Yaratish - bu tizim masalasidir. O'sha tizim o'zi faoliyat yaratadi va bizga ozgina ishtirok etish va onCreate () usulida kodimizni bajarish imkoniyatini beradi. Biz ushbu imkoniyatdan foydalanamiz va tizimga R.layout.main-dan faoliyat ekranni ko'rsatishi kerakligini aytamiz.

Boshqa barcha usullarni sxemadan qo'shamiz va har biriga jurnal yozuvini log ga chiqaramiz:

```
package uz.mobililova.develop.acvitystate;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends Activity {
 final String TAG = "States";
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Log.d(TAG, "MainActivity: onCreate()");
  }
 @Override
 protected void onStart() {
  super.onStart();
  Log.d(TAG, "MainActivity: onStart()");
 }
 @Override
 protected void onResume() {
  super.onResume();
  Log.d(TAG, "MainActivity: onResume()");
 }
 @Override
 protected void onPause() {
  super.onPause();
  Log.d(TAG, "MainActivity: onPause()");
 }
 @Override
 protected void onStop() {
  super.onStop();
  Log.d(TAG, "MainActivity: onStop()");
 }
 @Override
 protected void onDestroy() {
  super.onDestroy();
  Log.d(TAG, "MainActivity: onDestroy()");
```

Ushbu usullarni amalga oshirayotganda, tegishli superklass usullarini chaqirganingizga ishonch hosil qiling. Yuqoridagi kodga qarang. Har bir usul super-klass uslubiga chaqiruvni o'z ichiga oladi va o'z kodi ushbu chaqiruvlardan keyin joylashgan.

Savol va topshiriqlar:

1. Android tarixi nechanchi yillardan boshlangan?
2. Android boshqaruvi ostida ishlovchi birinchi qurilma qachon ishga tushdi?
3. Android qaysi korporatsiya mahsuloti hisoblanadi?
4. Android Studio dasturini o'rnatish uchun zarur bo'lgan qismlar?
5. Ta'lim soxasida bugungi kunda mashxur bo'lgan mobil ilovalar haqida misollar keltiring.
6. Mobil ilova yaratish uchun qaysi dasturlash tilidan foydalaniladi?
7. Dasturda loyiha struturasini tushuntirib bering.
8. SDK components haqida nimalarni bilasiz?
9. AVD-Android Virtual Device o'rnatish ketma-ketligini ko'rsating.
10. Dastur uchun qo'shimcha kutubxonalarni yuklash ketma-ketligini ko'rsating.

2-MAVZU. MOBIL OPERATSION TIZIMLAR PLATFORMASI VA ARXITEKTURASI

Android ilovalari Java tilida yozilgan. Android SDK vositalari kodni, ma'lumotlarni va resurslarni APK-da to'playdi (Android to'plami). Odatda bitta APK faylida ilovaning barcha mazmuni mavjud bo'ladi.

Har bir dastur o'z virtual mashinasida (VM) ishlaydi, shunda ilova boshqa dasturlardan ajratilgan holda ishlaydi. Android tizimi eng kam imtiyoz printsipi bilan ishlaydi. Har bir ilova faqat o'z ishini bajarishni talab qiladigan tarkibiy qismlarga kirish huquqiga ega, bundan tashqari. Shu bilan birga, dasturning boshqa ilovalar bilan ma'lumot almashish usullari mavjud, masalan, Linux foydalanuvchi identifikatorini dastur o'rtasida bo'lishish yoki ilovalar SD-karta, kontaktlar va boshqalar kabi qurilmalar ma'lumotlariga kirish uchun ruxsat so'rashlari mumkin.

Android SDK (dasturiy ta'minotni ishlab chiqarish to'plami) - bu Android platformasi uchun dasturlarni ishlab chiqish uchun ishlatiladigan vositalar to'plami bo'lib, u smartfonlar maydonida Apple-ning eng katta raqibiga aylandi. Android SDK quyidagilarni o'z ichiga oladi:

- Kerakli kutubxonalar.
- Xatolarni tuzatuvchi.
- Emulyator.
- Android dastur dasturi interfeyslari (API) uchun tegishli hujjatlar.
- Namuna manba kodi.
- Android OS uchun qo'llanmalar.

Google har safar Android-ning yangi versiyasini chiqarganida, tegishli SDK ham chiqariladi. Eng so'nggi xususiyatlarga ega dasturlarni yozish uchun ishlab chiquvchilar har bir versiyaning SDK-ni ma'lum bir telefon uchun yuklab olishlari va o'rnatishlari kerak. SDK, asosan, Android-ning operatsion tizimlarining ma'lum bir versiyasi va texnologiyasi uchun yetkazib beriladigan vositalarini taqdim etadi.

SDK bilan mos keladigan ishlab chiqish platformalariga Windows (XP yoki undan keyingi versiyalar), Linux (har qanday so'nggi Linux tarqatish) va Mac OS X (10.4.9 yoki undan keyingi

versiyalar) kabi operatsion tizimlar kiradi. Android SDK-ning tarkibiy qismlarini alohida yuklab olish mumkin. Uchinchi tomon qo'shimchalarini yuklab olish ham mumkin.

Ushbu IDElarning ko'pchiligi ishlab chiquvchilarga rivojlanish vazifalarini tezroq bajarishga imkon beradigan grafik interfeysni taqdim etadi. Android dasturlari Java kodida yozilganligi sababli, foydalanuvchi Java Development Kit (JDK) ni o'rnatishi kerak. Bundan tashqari, mutaxassislar ba'zida Android uchun SDK-lardan foydalanishni optimallashtirish haqida gapirishadi, masalan, qo'shimcha bog'langan API chaqiruvlariga ehtiyojni bartaraf etish uchun maqsadli versiyani ishlatish yoki SDK maqsadlarini "Android manifestiga qo'shilgan qiymat" deb atashlari mumkin. OS dasturlari bilan, ko'pchilik manifestda maqsadli versiya va minimal versiyani o'rnatishni tavsiya qiladi.

IDE va Android SDK

IDE dasturini tushunishning yana bir usuli va nima uchun uni oddiy SDK dan afzal ko'rish mumkinligi ID identifikatsiyalash usullari va rivojlanish jarayonini yengillashtirishi haqida o'ylashdir. IDE-larning yuqorida aytib o'tilganidek bajaradigan ishlaridan biri bu turli xil vositalarni bitta muhitda birlashtirishdir. Masalan, agar IDE matn muharriri, kompilyator va disk raskadrovka uchun funksiyalarni to'liq ta'minlasa (bu yerda har bir narsa SDK bilan ishlashda alohida ko'rib chiqilishi kerak bo'lsa), bu ishlab chiquvchi uchun aniq foyda keltiradi. Shuningdek, mutaxassislar IDElar, masalan, Netbeans, foydalanuvchi uchun grafik interfeysni yoki SDK-larda qo'lda yoki asosan amalga oshiriladigan deb hisoblanadigan rivojlanish muhitining yanada qulayroq versiyalarini qanday yaratishini tushuntiradi. Amalga oshirish uchun HTML muhitida GUI taqdim etgan ko'plab platformalarning paydo bo'lishidan farqli o'laroq (masalan, Dreamweaver, shu jumladan) web-sahifalarni kodlash uchun HTML-dan foydalanish yaxshi o'xshashlik bo'lishi mumkin.

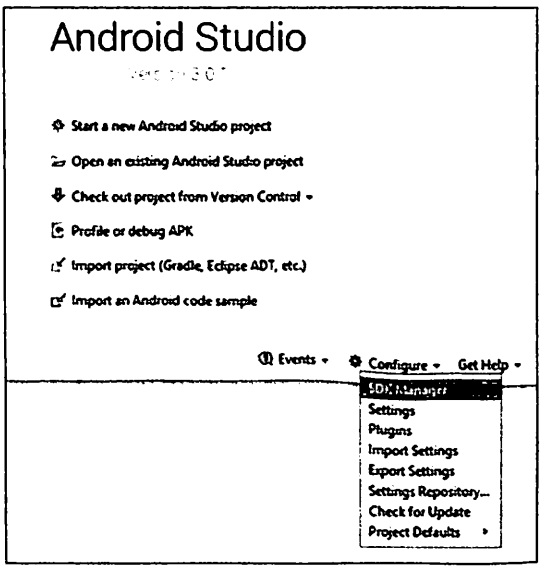
GUI asosiy vazifalarni shu tarzda qisqartirish orqali foydalanuvchilarning keng doirasi uchun vazifalarni yanada qulayroq qiladi va vaqt, kuch tejashga yordam beradi. IDE buni Android ishlab chiqaruvchisi uchun qiladi, ammo ba'zilari juda ko'p yorliqlar ishlab

chiquvchi turli xil muhitlarda ishonchli bo'lishi kerak bo'lgan asosiy bilimlarni buzadi deb aytiladi.

SDKni o'rnatish

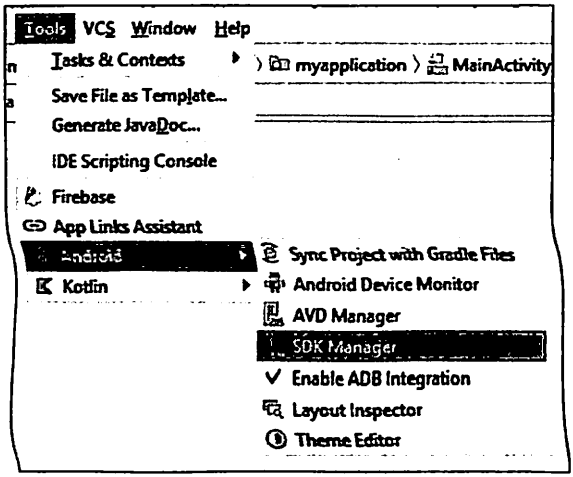
Simulink modellarini Android qurilmangizda ishlatish uchun kompyuteringizga Android Software Development Kit (SDK) platforma paketlari va vositalarini o'rnatishingiz kerak. Android SDK platformasi paketlarini va vositalarini Android Studio-dan o'rnatishingiz mumkin.

1. Android studioni ishga tushiramiz.
2. SDK Manager ni ochib, quyidagilarni bajaramiz.
3. Android Studio ochilish sahifasida **Configure>SDK manager**-ni tanlang.
4.



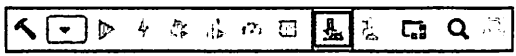
2.1-rasm. SDK sozlamalari

Android Studio dasturlar panelidan **Tools>Android> SDK manager**-ni tanlang.



2.2-rasm. SDK Manager joylashuvi

Android Studio dasturlar panelidan SDK manager-ni bosing.



2.3-rasm. Panel orqali kirish

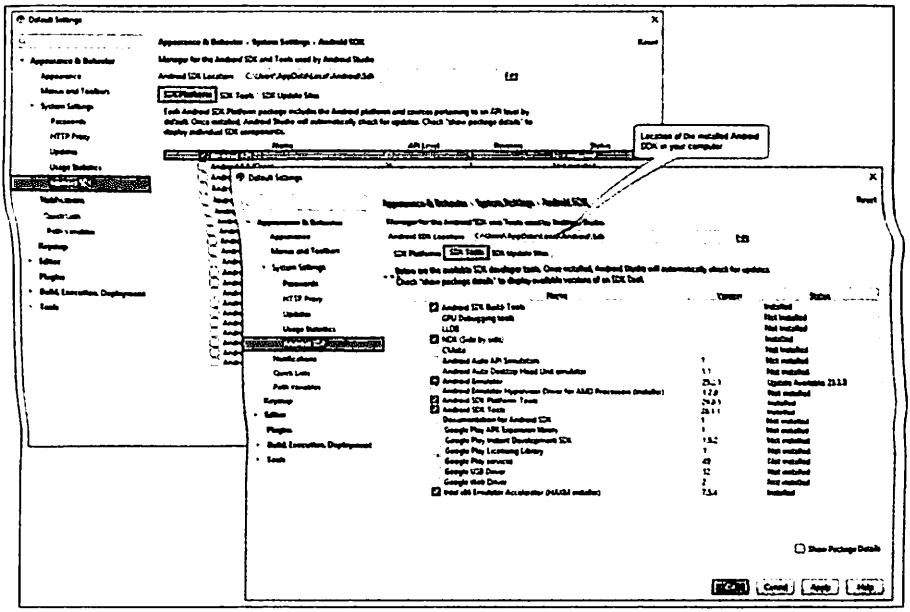
Standart sozlamalar dialog oynasida Android SDK platformasi paketlarini va ishlab chiquvchi vositalarini o'rnatish uchun ushbu yorliqlarni bajaramiz:

a. **SDK Platforms.** Eng so'nggi Android SDK paketini tanlaymiz.

b. **SDK Tools.** Android SDK toolsni tanlaymiz.

- Android SDK Build-Tools
- NDK (yonma-yon)
- Android SDK platformasi vositalari
- Android SDK vositalari

Eslatma! Android SDK joylashuvi parametrida ko'rsatilgan Android SDK-ning joylashuvi bo'sh joylarni o'z ichiga olmaydi. Bo'shliqlar Android NDK vositalari bilan ishlashda muammo tug'dirishi mumkin. SDK manzilini Tahrirlash tugmachasini bosish orqali o'zgartirishingiz mumkin.



2.4-rasm. Android SDK Locationni aniqlash

Ilovaga bosing. Android Studio tanlangan paketlar va vositalarni kompyuteringizga o'rnatishni boshlaydi. O'rnatish tugagandan so'ng, o'rnatilgan paketlar va vositalarning holati o'rnatilmagandan o'rnatilganga o'zgaradi.

Ilova komponentalari

Ilova komponentlari Android dasturining qurilish bloklari hisoblanadi. Har bir komponent Android dasturida aniq ro'l o'ynaydi, bu aniq maqsadga xizmat qiladi va aniq hayot tsikllariga ega (komponent qanday va qachon yaratilgani va yo'q qilinganligi xususiyati). Bu yerda dastur tarkibiy qismlarining to'rt turi mavjud:

1. **Activities:** Faoliyat foydalanuvchi interfeysi (UI) bilan bitta ekranni aks ettiradi. Android ilovasida bir nechta harakatlar bo'lishi mumkin. (masalan, elektron pochta ilovasida barcha elektron pochta xabarlarini ro'yxatlash uchun bir harakat bo'lishi mumkin, ikkinchisida har bir elektron pochta tarkibini ko'rsatish uchun,

boshqasida yangi elektron pochta xabarlarini yaratish uchun.) Ilovadagi barcha tadbirlar birgalikda foydalanuvchi eXperience (UX) yaratish uchun ishlaydi.

2. **Services:** Uzoq muddatli operatsiyalarni bajarish yoki masofaviy jarayonlar ishni bajarish uchun xizmat fonda ishlaydi. Xizmat hech qanday foydalanuvchi interfeysini taqdim etmaydi, faqat foydalanuvchi kiritgan fonda ishlaydi. (masalan. foydalanuvchi turli xil ilovalarda bo'lganida, musiqa fonda musiqa ijro etishi yoki Internetdan ma'lumotlarni yuklab olib, Android qurilmasi bilan o'zaro ta'sirini bloklamasligi mumkin.)

3. **Content Providers:** Kontent-provayder umumiy dastur ma'lumotlarini boshqaradi. Ilovada ma'lumotlarni saqlashning to'rtta usuli mavjud: u faylga yozilishi va tizimda saqlanishi, SQLite ma'lumotlar bazasiga qo'shilishi yoki yangilanishi, Internetga joylashtirilishi yoki boshqa har qanday doimiy saqlash joyida saqlanishi mumkin. Kontent-provayderlar orqali boshqa dasturlar ma'lumotlarni so'rashi yoki o'zgartirishi mumkin. (masalan, Android tizimi foydalanuvchi bilan aloqa ma'lumotlarini boshqaradigan kontent-provayderni taqdim etadi, shunda ruxsati bo'lgan har qanday dastur kontaktlarni so'rashi mumkin.) Kontent-provayderlar, shuningdek, ma'lumotlarning yaxlitligi uchun dastur uchun shaxsiy ma'lumotlarni saqlash uchun ishlatilishi mumkin.

4. **Broadcast receivers:** Teleradio eshittirish qabul qiluvchisi tizimdagi e'lonlarni eshitishga javob beradi (masalan 4. ekran o turned ga burilganligi, batareyasi past bo'lganligi va h.k.) yoki Ilovalardan (masalan, ba'zi ilovalar qurilmaga ba'zi ma'lumotlar yuklab olinganligini va ulardan foydalanish uchun boshqa ilovalarga xabar berish uchun). Teleradio eshittirish qabul qiluvchilarida foydalanuvchi interfeysi mavjud emas, lekin ular foydalanuvchi to'g'risida ogohlantirish uchun holat satrida bildirishnoma ko'rsatishi mumkin. Odatda translyatsiya qabul qilgichlari asosan faoliyat va xizmatlardan iborat dasturning boshqa tarkibiy qismlariga kirish eshigi sifatida ishlatiladi.

Android tizimining o'ziga xos jihati shundaki, har qanday dastur boshqa dasturning tarkibiy qismini ishga tushirishi mumkin (masalan, siz qo'ng'iroq qilishni, SMS yuborishni, web-sahifani ochishni yoki fotosuratni ko'rishni xohlasangiz, buni amalga oshiradigan dastur mavjud va sizning ilovangiz xuddi shu vazifa uchun yangi faoliyatni

rivojlantirish o'rniga, undan foydalanadi). Tizim komponentni ishga tushirganda, ushbu dastur uchun jarayonni boshlaydi (agar u hali ishlamayotgan bo'lsa, ya'ni Android tizimida istalgan vaqtda har bir dastur uchun faqat bitta oldingi jarayon ishlashi mumkin) va ushbu komponent uchun zarur bo'lgan darslarni tayyorlaydi. Ushbu komponent o'zi tegishli bo'lgan Ilova jarayonida ishlaydi. Shuning uchun, boshqa tizimlardagi dasturlardan farqli o'laroq, Android dasturlarida bitta kirish nuqtasi yo'q (main () usuli yo'q). Tizim har bir dasturni alohida jarayonda boshqarganligi sababli, bitta ilova boshqa dasturning tarkibiy qismlarini to'g'ridan-to'g'ri faollashtira olmaydi, ammo Android tizimi buni amalga oshirishi mumkin. Bu boshqa dasturning tarkibiy qismini ishga tushirish uchun bitta dastur tizimga ushbu komponentni boshlash niyatida bo'lgan xabarni yuborishi kerak, shunda tizim ushbu komponentni ishga tushiradi.

Context

Android.content.Context sinfining misollari dasturni bajaradigan Android tizimiga ulanishni ta'minlaydi. Kontekstning loyiha manbalari va ilova muhiti to'g'risidagi global ma'lumotlarga kirish uchun talab qilinadi. Keling, oson bo'lgan misolni keltiramiz: Mehmonxonada ekanligingizni va biror narsa yeyishni xohlayotganingizni o'ylab ko'ring. Siz xona xizmatiga qo'ng'iroq qilasiz va sizga narsalarni olib kelishini yoki siz uchun narsalarni tozalashlarini so'raysiz. Endi bu mehmonxonani Android ilovasi, o'zingizni faoliyat deb biling va xona xizmatidagi odam sizning kontekstingiz bo'lib, sizga xona xizmati, oziq-ovqat mahsulotlari va boshqalar kabi mehmonxona resurslaridan foydalanish imkoniyatini beradi. Yana bir misol. Siz restoranda stolda o'tiribsiz, har bir stolda xizmat ko'rsatuvchi bor, qachonki siz xizmat ko'rsatuvchidan so'ragan ovqatga buyurtma berishni xohlasangiz shuni aytasiz. Keyin xizmatchi sizning buyurtmangizni beradi va sizning ovqatlanadigan stolingizda xizmat qiladi. Ushbu misolda yana restoran Android ilovasi, jadvallar yoki mijozlar App komponentlari, oziq-ovqat mahsulotlari sizning App resurslaringiz va xizmat ko'rsatuvchi sizning kontekstingizdir, shuning uchun sizga oziq-ovqat mahsulotlari kabi manbalarga kirishga imkon beradi.

Yuqoridagi tarkibiy qismlardan birini faollashtirish uchun kontekstning misoli kerak. Nafaqat yuqoridagilar, balki deyarli barcha tizim resurslari: Ko'rishlar yordamida foydalanuvchi interfeysini yaratish (keyinroq muhokama qilinadi), tizim xizmatlarining namunalarini yaratish, yangi faoliyat yoki xizmatlarni boshlash uchun barchasi kontekstni talab qiladi.

AVD (Android Virtual Device) ni sozlash

Android Developer Documentation

> an ***Android Virtual Device (AVD) definition*** lets you define the characteristics of an Android Phone, Tablet, Android Wear, or Android TV device that you want to simulate in the Android Emulator. The AVD Manager helps you easily create and manage AVDs.

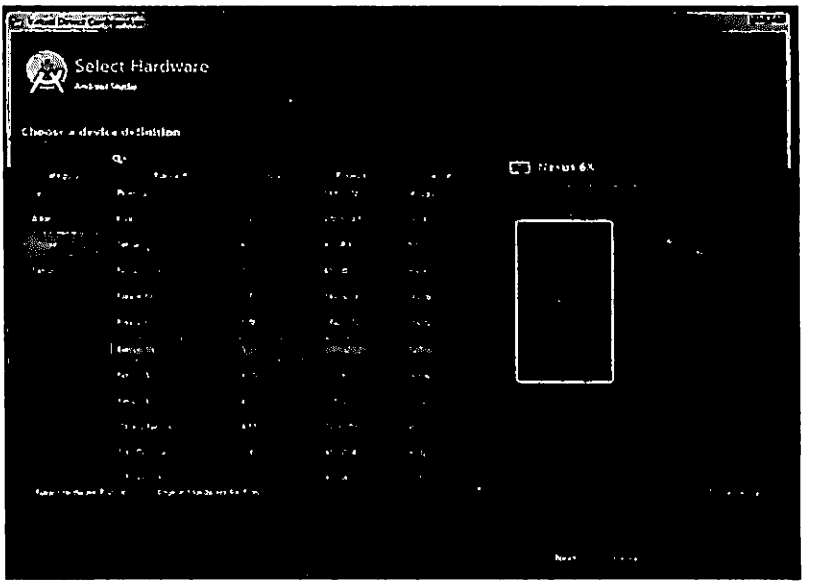
Quyidagi ketma-ketliklar orqali AVD ni sozlaymiz:

1. AVD menejerini ochish uchun ushbu tugmani bosing:

3. Siz shunday dialog oynasini ko'rishingiz kerak (2.1-rasm):

2.1-rasm. AVD oynasi

3. Endi Create Virtual device... tugmasini bosing. Bu Virtual qurilmalarni boshqarish bo'yicha dialogni keltirib chiqaradi (2.2-rasm):

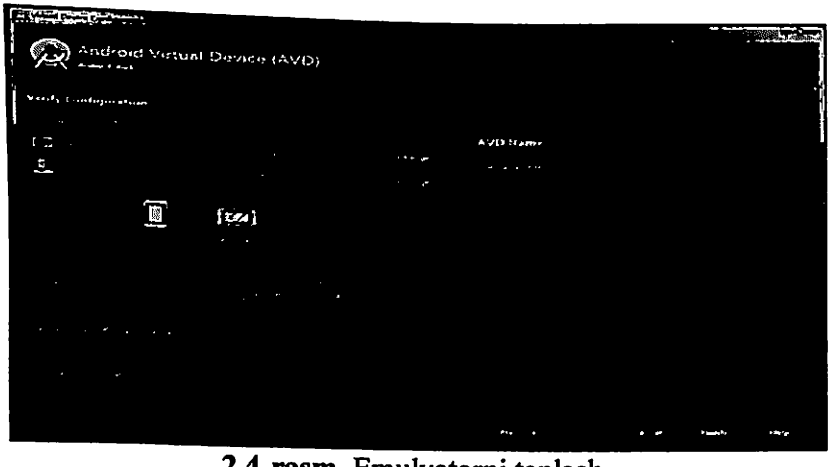


2.2-rasm. Virtual qurilmalarni boshqarish

4. O'zingiz xohlagan qurilmangizni tanlang va Next tugmasini bosing (2.3-rasm):

2.3-rasm. Ixtiyoriy qurilmani tanlash

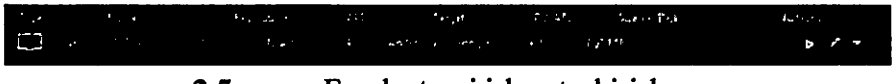
5. Bu yerda siz emulyatoringiz uchun Android versiyasini tanlashingiz kerak. Yuklash tugmachasini bosish orqali uni birinchi marta yuklab olishingiz kerak bo'lishi mumkin. Versiyasini tanlagandan so'ng, Next ni bosing (2.4-rasm).



2.4-rasm. Emulyatorni tanlash

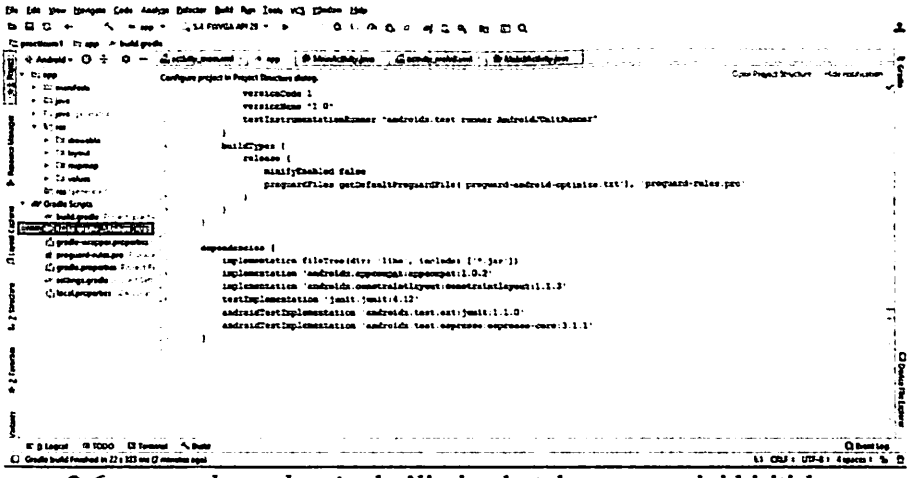
6. Bu yerda emulyatoringiz uchun nom, boshlang'ich yo'nalish va uning atrofida ramka ko'rsatishni xohlaysangiz, kiriting. Bularning barchasini tanlaganingizdan so'ng, Finish tugmachasini bosing.

7. Endi siz o'zingizning ilovalaringizni ishga tushirishga tayyor bo'lgan yangi AVD-ni hosil qildingiz va bemalol android ilovalaringizni ishlatib ko'rishingiz mumkin (2.5-rasm).



2.5-rasm. Emulyatorni ishga tushirish

Buil.gradle faylida dastur uchun qo'shimcha kutubxona fayllarini yuklab olish mumkin. Bu esa o'z navbatida dastur effektivligini va zamonaviyligini ta'minlaydi.



2.6-rasm. *dependencies* bo'limiga kutubxona nomini kiritish

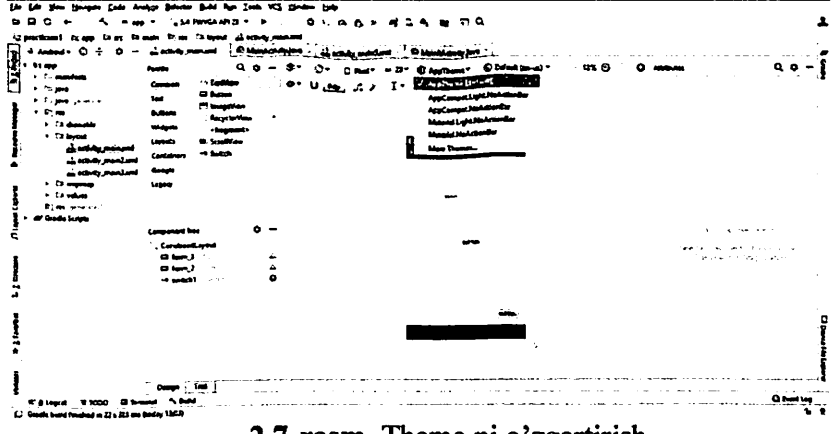
Masalan: kutubxona nomini quyidagi tartibda kiritishimiz mumkin

implementation fileTree(**dir: ' libs' , include: ' *.jar'**])

implementation ' **androidx.appcompat:appcompat:1.0.2'**

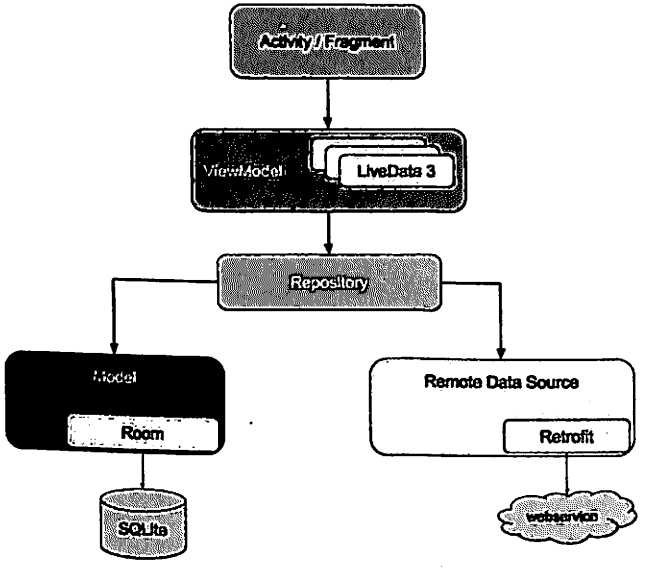
implementation **androidx.constraintlayout:constraintlayout:1.1.3'**

Dasturga qo'shimcha o'rnatishlar bilan tanishishni davom qilamiz. Dastur ko'rinishini o'zgartirish ya'ni theme ni o'zgartirishni ko'rib chiqamiz.



2.7-rasm. Theme ni o'zgartirish

Boshlash uchun dasturni ishlab chiqqandan so'ng barcha modullarning bir-biri bilan o'zaro aloqasini ko'rsatadigan quyidagi diagrammani ko'rib chiqing(2.8-rasm.). E'tibor bering, har bir komponent faqat uning ostidagi bir darajadagi tarkibiy qismga bog'liq. Masalan, faoliyat va parchalar faqat ko'rinish modeliga bog'liq. Ombor boshqa bir nechta sinflarga bog'liq bo'lgan yagona sinfdir; ushbu misolda ombor doimiy ma'lumotlar modeliga va masofaviy orqa ma'lumot manbalariga bog'liq. Ushbu dizayn doimiy va yoqimli foydalanuvchi tajribasini yaratadi. Foydalanuvchi dasturni oxirgi marta yopganidan bir necha daqiqa o'tgach yoki bir necha kundan keyin qaytib keladimi-yo'qligidan qat'i nazar, ular foydalanuvchi dasturning mahalliy sifatida saqlanib qolishi haqidagi ma'lumotlarini darhol ko'rishadi. Agar ushbu ma'lumotlar eskirgan bo'lsa, dasturning ombor moduli fonda ma'lumotlarni yangilashni boshlaydi.



2.8-rasm. Dastur umumiy arxitekturasi.

Savol va topshiriqlar:

1. Android uchun maxsus instrumental dasturiy vositalarga nimalar kiradi?
2. Ilova komponentalari haqida gapirib bering.
3. Activitylar va ularni yaratish ketma-ketligini ko'rsating.
4. Intent va uning metodlari haqida nimalarni bilasiz?
5. Content providers deb nimaga aytiladi?
6. Contex ga misollar keltiring.
7. AVD (Android virtual device): sozlash ketma-ketligini ko'rsating.
8. Android kutubxonalari haqida nimalarni bilasiz?
9. Maxsus komponentalarni yuklab olish manzillariga misollar keltiring.
10. JDK va JRE nima uchun kerak?

3-MAVZU: JAVA DASTURLASH TILINING ASOSIY KONSTRUKSIYALARI

Java 1995 yilda yaratilgan mashhur dasturlash tili hisoblanadi. U quyidagi maqsadlar uchun qo'llaniladi:

- Mobil ilovalar (maxsus Android ilovalar)
- Desktop ilovalalar
- Web ilovalar
- Web serverlar va server ilovalari
- O'yinlar
- Database bog'lanishlar
- Va ko'plab boshqa yo'nalishlar uchun!

Java nima uchun qo'llaniladi?

- Java turli xil platformalarda ishlay oladi (Windows, Mac, Linux, Raspberry Pi, etc.)
- Bu dunyodagi eng mashhur dasturlash tillaridan biri
- O'rganish va ulardan foydalanish oson
- Bu ochiq manbali va bepul
- Bu xavfsiz, tezkor va kuchli
- U jamoatchilikning katta qo'llab-quvvatlashiga ega (o'n millionlab ishlab chiquvchilar)
- Java - bu dasturlarga aniq tuzilmani beradigan va kodni qayta ishlatishga imkon beradigan va ishlab chiqarish xarajatlarini pasaytiradigan obyektga yo'naltirilgan til
- Java C ++ va C# ga yaqin bo'lgani uchun dasturchilarning Java-ga o'tishini yoki aksincha bo'lishini osonlashtiradi

Javani o'rnatish

Ba'zi kompyuterlarda Java allaqachon o'rnatilgan bo'lishi mumkin.

Windows-ning shaxsiy kompyuterida Java o'rnatilganligini tekshirish uchun boshlang'ich satrida Java-ni qidiring yoki buyruq satri (cmd.exe) ga quyidagilarni kiriting:

Agar java o'rnatilgan bo'lsa, shunga o'xshash narsani ko'rasiz (versiyasiga qarab):

Agar kompyuteringizda Java o'rnatilmagan bo'lsa, uni oracle.com dan bepul yuklab olishingiz mumkin.

Eslatma: Ushbu qo'llanmada Java kodini matn muharririga yozamiz. Shu bilan birga, Java-ni IntelliJ IDEA, Netbeans yoki Eclipse kabi Integrated Development Environment-da yozish mumkin, bu ayniqsa Java fayllarining katta to'plamlarini boshqarishda foydalidir.

Java Quickstart

Java-da har bir dastur sinf nomidan boshlanadi va bu sinf fayl nomiga mos kelishi kerak.

Keling, har qanday matn muharriri (Notepad kabi) da bajarilishi mumkin bo'lgan Main.java deb nomlangan birinchi Java faylimizni yarataylik.

Faylda quyidagi kod bilan yozilgan "Hello world" xabari bo'lishi kerak:

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Agar yuqoridagi kodni tushunmasangiz, xavotir olmang - biz keyingi boblarda uni batafsil muhokama qilamiz. Hozircha yuqoridagi kodni qanday ishlatishga e'tibor bering.

Kodni bloknotda "Main.java" sifatida saqlang. Cmd ni oching (cmd.exe), faylni saqlagan katalogga o'ting va " Main.java" deb yozing:

Java sintaksisi

Oldingi bobda biz Main.java nomli Java faylini yaratdik va "Hello World" ni ekranga chiqarish uchun quyidagi koddan foydalandik:

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Java-da ishlaydigan har bir kod satri class ichida bo'lishi kerak. Bizning misolimizda biz sinfni **Main** deb nomladik. Class har doim bosh harf bilan boshlanishi kerak.

Eslatma: Java katta-kichik harflarni farqlaydi: "MyClass" va "myclass" har xil ma'noga ega.

Main () metodi bo'lishi kerak va uni har bir Java dasturida ko'rasiz:

public static void **main**(String[] args)

Main() usuli ichidagi har qanday kod bajariladi. Asosiy so'zdan oldin va keyin kalit so'zlarni tushunishingiz shart emas. Ushbu qo'llanmani o'qiyotganda siz ularni asta-sekin bilib olasiz.

Hozircha har bir Java dasturida fayl nomiga mos keladigan sinf nomi borligini va har bir dasturda main() usuli bo'lishi kerakligini unutmang.

Java o'zgaruvchilari

O'zgaruvchilar ma'lumotlar qiymatlarini saqlash uchun konteynerlardir.

Java-da, masalan, har xil turdagi o'zgaruvchilar mavjud:

- String - "Hello" kabi matnlarni saqlaydi. String qiymatlari ikkita qo'shtirnoq bilan o'ralgan
- int - 123 yoki -123 kabi o'nliksiz butun sonlarni (butun sonlarni) saqlaydi
- float - 19.99 yoki -19.99 kabi o'nlik bilan suzuvchi nuqta raqamlarini saqlaydi
- char - bitta belgini saqlaydi, masalan 'a' yoki 'B'. Char qiymatlari bitta qo'shtirnoq bilan o'ralgan

- boolean - qiymatlarni ikkita holat bilan saqlaydi: haqiqiy yoki noto'g'ri
- int myNum = 5;
- float myFloatNum = 5.99f;
- char myLetter = 'D';
- boolean myBool = true;
- String myText = "Hello";

Java operatorlari

Java operatorlari quyidagi guruhlarga ajratiladi:

- Arifmetik operatorlar
- Topshiriq operatorlar
- Taqqoslash operatorlari
- Mantiqiy operatorlar
- Bit operatorlar

Arifmetik operatorlar

Arifmetik operatorlar umumiy matematik amallarni bajarish uchun ishlatiladi.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Java ko'rsatgich operatorlari

Operator	Misol	Bir xil
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Java taqqolash operatorlari

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Java mantiqiy operatorlar

Operator	Nomi	Tavsif	Misol
&&	Mantiqiy and	Ikkala so'z ham to'g'ri bo'lsa, haqiqiy qiymatni qaytaradi	x < 5 && x < 10
||	Mantiqiy or	Agar . iboralardan biri to'g'ri bo'lsa, haqiqiyni qaytaradi	x < 5 || x < 4
!	Mantiqiy not	Natijani teskari tomonga qaytaradi, agar natija rost bo'lsa, false qiymatini qaytaradi	!(x < 5 && x < 10)

Shunday qilib, har qanday Java dasturi obyektlar yaratilgan sinflardan iborat. Umuman olganda, obyekt bu o'zgaruvchilar va usullar to'plamidir. Metod - bu obyekt o'zgaruvchilarini qayta ishlash va boshqa harakatlarni bajarish uchun mo'ljallangan kodning nomlangan qismi. Dastur deyarli har doim asosiy sinfni va main () asosiy usulini o'z ichiga oladi, ular dastur boshlanganda bajariladi. Hatto kichik o'quv loyihasi ham bir necha o'nlab sinflardan iborat bo'lishi mumkin. Dasturchilar jamoasi tomonidan ishlab chiqilgan jiddiy tijorat loyihalari minglab sinflardan iborat. Bunday vaziyatda ism ziddiyatining haqiqiy muammosi paydo bo'ladi. Bir tomondan, sizning kodingizni tushunishni, disk raskadrovka qilishni va hujjatni osonlashtiradigan tavsiflovchi nomlardan foydalanish maqsadga muvofiqdir. Boshqa tomondan, agar yuzlab va minglab sinflar mavjud bo'lsa, unda sinf nomlari bir-biriga mos kelishi muqarrar.

Nazorat ko'rsatmalari

Boshqarish ko'rsatmalari ma'lum bir shartga qarab dasturni bajarish tartibini o'rnatadi. Nazorat ko'rsatmalarini qo'llash orqali siz filial punktlarini yaratishingiz mumkin, tanaffuslar, bayonotlar bloklarini yoki butun dasturni qayta-qayta bajaring.
Java-dagi shartli if ifodasi quyidagicha ko'rinadi:

```
if (shart) {
// Agar shart to'g'ri bo'lsa, buyruqlar bloki
}
```

```
else {
// Agar shart noto'g'ri bo'lsa, buyruq bloki
}
// Dasturni davom eting
```

Operator bajarilganda qavs ichidagi shartning haqiqati tekshiriladi. Agar shartli ifoda rost bo'lsa, if kalit so'zidan keyin sistema qavsdagi buyruqlarning birinchi bloki va else kalit so'zidan keyingi blok bajariladi. Agar shartli ifoda noto'g'ri qiymatini qaytarsa, unda birinchi buyruqlar bloki e'tiborsiz qoldiriladi va else kalit so'zidan keyingi buyruqlar bloki bajariladi.

Shart operatoriga misol:

```
if (a+b> 100) {
a = 0;
b = 0;
}
else {
a = a +5;
b = b +2;
}
```

Boshqa blok mavjud bo'lmagan operatorning soddalashtirilgan yozuviga ruxsat beriladi. Bunday holda, agar shart noto'g'ri bo'lsa, unda maxsus choralar ko'rilmaydi. Misol soddalashtirilgan yozuv:

```
if (a + b> 100) {
a = 0;
b = 0;
} // Shartli bayonning oxiri
// Dasturning keyingi buyruqlari
```

Ichki shartli operatorlar:

Shartli gap operator shartli gapning ichida joylashgan bo'lishi mumkin. Bunga ichki joylashtirilgan shartli operatorlar deyiladi. Ichki joylashtirilgan raqam operatorlar rasmiy ravishda cheklanmagan, ammo ravshanlik uchun shablonda biz faqat ikkita ichki shartli operatorni ko'rsatamiz

```
if (shart 1) {
// Buyruq qatori 1
}
```

```
else if (shart 2) {
// Buyruq qatori 2
}
else if (shart 3) {
// Buyruq qatori 3
}
else {
// Buyruq qatori 1
 }
```

Agar 1-shart to'g'ri bo'lsa, unda 1-buyruq bloki bajariladi, boshqa bloklar e'tiborga olinmaydi va buyruqlar bajarilishi qurilishdan keyin ham davom etadi. Agar 1-shart yolg'on bo'lsa, unda shart 2 tekshiriladi.2 shart to'g'ri bo'lsa, buyruq bloki 2 bajariladi va hokazo. Buyruqlarning oxirgi bloki faqat avvalgi holatdagina bajariladi.

Tanlash operatori-switch

Switch iborasi ortidagi mantiq biz ko'rib chiqqan if ifodasiga o'xshaydi. Asosiy farq shundaki, sinovdan o'tgan ifoda mantiqiy qiymatlarni haqiqiy yoki noto'g'ri emas, balki faqat butun yoki belgi qiymatini qaytarishi mumkin. Umuman olganda, switch iborasi shabloni quyidagicha:

```
switch (shart) {
case qiymat_1:
// Buyruq qatori 1
break;
case qiymat _2:
// Блок Buyruq qatori 2
break;
case qiymat_3:
// Buyruq qatori 3
break;
// boshqa case—bloklari
case qiymat_n:
// Buyruq qatori n
break;
default:
// Buyruq qatori
```

Switch operatori bajarilganda, qavs ichidagi ifoda qiymati baholanadi. Keyin ushbu qiymat birma-bir yuqoridan pastga qarab har bir ish blokining boshida ko'rsatilgan qiymatlar bilan taqqoslanadi. O'xshashlik topilgandan so'ng, tegishli blokning buyruqlar to'plami bajariladi.

Switch iborasining hiyla-nayrangliligi shundaki, moslik topilganda, barcha buyruqlar bayonot oxirigacha, shu qatorda quyida joylashgan blok bloklaridagi buyruqlar bajariladi. Agar bitta blokning buyruqlari bajarilishi zarur bo'lsa, u break operatori bilan tugatilishi kerak.

Bayonot ixtiyoriy standart blokirovka bilan tugaydi. Ushbu blokdagi buyruqlar, agar boshqaruv qiymatlari bilan mos kelmasa, bajariladi. Standart blok bajarilishni yakunlaganligi sababli, u break operatoridan foydalanmaydi.

Tanlash operatoridan foydalanish misoli

```
// Swing kutubxonasidan JOptionPane sinfini qo'llaymiz
import javax.swing.JOptionPane;
public class Listing4_1 {
public static void main (String [] args) {
int userData;
String userInput;
// Joriiy vaqt oynasini chiqaramiz
userInput = JOptionPane.showInputDialog («Введите число от 1 до
3»);
userData = Integer.parseInt (userInput);
switch (userData) {
case 1:
JOptionPane.showMessageDialog (null, «Вы ввели число 1»);
break;
case 2:
JOptionPane.showMessageDialog (null, «Вы ввели число 2»);
break;
case 3:
JOptionPane.showMessageDialog (null, «Вы ввели число 3»);
break;
default:
```

JOptionPane.showMessageDialog (null, «Вы ввели недопустимое число!»);
}
}
}

Ushbu dasturda biz allaqachon tanish bo'lgan dialog oynalaridan foydalanuvchidan 1 dan 3 gacha raqam kiritishni va javob xabarini ko'rsatishni so'rash uchun foydalanamiz. Foydalanuvchi belgilangan oraliqdagi raqamni kiritadi, keyin tasdiq ko'rsatiladi. Agar kiritilgan raqam boshqaruv qiymatlarining hech biriga mos kelmasa, u holda standart blok ishga tushiriladi va xato xabari ko'rsatiladi.

Ushbu dastur switch operatorining ishlashini aniq namoyish etadi, lekin dastur kodi nuqtai nazaridan maqbul emas.

Mantiqiy operatorlar va shartli if operatorlari haqidagi bilimlardan foydalanib dasturni qayta yozamiz. Qaramasdan dasturni o'zingiz qayta tuzishga harakat qiling.

Mantiqiy va shartli operator yordamida namunaviy dastur

```
// Swing kutubxonasidan JOptionPane sunfini import qilamiz
import javax.swing.JOptionPane;
public class Listing4_2 {
public static void main (String [] args) {
int userData;
String userInput;
userInput = JOptionPane.showInputDialog («Введите число от 1 до 3»);
userData = Integer.parseInt (userInput);
if ((userData> =1) & (userData <=3)) {
JOptionPane.showMessageDialog (null, «Вы ввели число " + userData);
}
else {
JOptionPane.showMessageDialog (null, «Вы ввели недопустимое число!»);
}
}
}
```

Dasturning tahrirlangan qismi qalin harflar bilan ko'rsatilgan. Ko'rib turganingizdek, natija yanada ixcham va ko'p qirrali dizayndir. **If** ifodasida qo'shma shart ishlatiladi.

(userData> =1) & (userData <=3)

Demak, agar userData o'zgaruvchisining qiymati bitta va uchdan kichik yoki unga teng yoki kattaroq bo'lsa, u holda haqiqiy bo'ladi. Bunday holda, dialog oynasi paydo bo'ladi.

Sikl operatorlari

Loop (takrorlanish) operatorlari ko'rsatmalar bloklarini bir necha marta bajarishga mo'ljallangan. Java tili while, do ... while va iboralar uchun foydalanadi.

while (условие) {
// Buyruq qatori
}

Loop operatori bajarilganda avval shart tekshiriladi. Agar shart to'g'ri bo'lsa, u holda sikl tanasidagi buyruqlar bloki bajariladi. Keyin shart yana tekshiriladi. Agar u qolsa to'g'ri, buyruq bloki yana bajariladi. Agar shart noto'g'ri bo'lib qolsa, u holda sikl operatorining ishlashi tugatiladi va boshqaruv sikldan keyingi buyruqlarga o'tkaziladi. While loopining misoli:

```
int a = 0;
while (a <10) {
System.out.println (a);
a++;
}
System.out.println («Sikl bajarilishining yakuni»);
```

Ushbu misolda, a o'zgaruvchisi qiymati 10 dan kam bo'lib qolganda, sikl bajariladi. Siz allaqachon o'zgarib turadigan (++) avtoinkrement operatori bilan tanishsiz. O'zgaruvchining qiymati. Agar siz sikl tanasida o'zgaruvchining qiymatini o'zgartirmasangiz, unda sikl abadiy ishlaydi, chunki shart har doim to'g'ri bo'ladi. Ba'zan bunday "abadiy sikllar» zarur. Ammo aksariyat hollarda, bu dasturning aylanishiga olib keladigan mantiqiy xato hisoblanadi.

Muayyan holatlarda, agar sikl sharti dastlab yolg'on bo'lsa, while sikli ichidagi buyruqlar bloki hech qachon bajarilmasligi mumkin. Masalan, agar siklni bajarishdan oldin a qiymati o'zgaruvchiga 10 qiymati berilsa, u holda misol ko'chadan bir marta ham ishlamaydi.

do while sikl operatori

do ... while iborasi while bayonotiga o'xshaydi, lekin u boshqa tuzilishga ega va buyruqlar bloki kamida bir marta bajariladi, chunki shartning haqiqati keyin tekshiriladi.

```
do {
// Buyruq qatori
} while (shart);
```

do... while qo'llab avvalgi misolni qayta yozamiz

```
int a = 0;
do {
System.out.println (a);
a++;
} while (a <10);
```

Ushbu misoldan olingan dastur 0 dan 9 gacha bo'lgan raqamlarni terminal oynasiga chiqaradi, ammo agar a o'zgaruvchisi 10 yoki undan ortiq qiymat bilan boshlanadigan bo'lsa, u holda sikl bir marta ishlaydi va o'zgaruvchining boshlang'ich qiymatini chop etadi.

for sikl operatori

for loop operatori eng murakkab tuzilishga ega bo'lib, unda barcha tarkibiy qismlar mavjud - ishga tushirish, holat, o'zgarish

```
for (boshlangich qiymat; shart; ++/--) {
// Buyruq qatori
}
```

Loop o'zgaruvchisi sikl operatoriga murojaat qilishda faqat bir marta ishga tushiriladi. Keyin shartning haqiqati tekshiriladi. Agar u to'g'ri bo'lsa, unda buyruq bloki bajariladi. Keyinchalik, sikl o'zgaruvchisining yangi qiymati hisoblab chiqiladi va shartning haqiqati yana tekshiriladi. Agar u to'g'ri bo'lsa, yana buyruq bloki bajariladi. Sikl shart to'g'ri bo'lguncha takrorlanadi.

for tarorlanishga misol:

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

Agar siklning tanasi bitta buyruqdan iborat bo'lsa, siz sistemani qavslarsiz bajarishingiz mumkin:
for (int i=0; i <=10; i++) System.out.println (i);

Murakab sikl

Loop bayonoti boshqa siklning tanasida joylashishi mumkin. Bunday holda, tashqi siklning har bir o'tishida ichki o'rnatilgan sikl ishga tushiriladi va to'liq bajariladi. Ichki ko'chadan, odatda, ikki o'lchovli yoki ko'p o'lchovli tuzilmalar elementlari (matritsalar, massivlar, jadvallar) ustida ketma-ket iteratsiya qilish va shu elementlar bo'yicha amallarni bajarish talab qilinadi.

Tashqi sikl haftaning birinchi kunlari, birinchi kunidan yettinchi kunigacha takrorlanadi. Siklning har bir o'tishida haftaning kun soni, keyin ichki o'rnatilgan sikl boshlanadi. O'rnatilgan sikl tugagandan so'ng, ketma-ketlik \ n qochish ketma-ketligi yordamida amalga oshiriladi va tashqi siklning navbatdagi takrorlanishi boshlanadi.

O'rnatilgan joriy kun davomida soat davomida takrorlanadi, soat 1 dan 24 gacha. Soat hisoblagichining qiymatlari ketma-ket vergul bilan ajratilgan bitta satrda ko'rsatiladi, bo'sh joy bilan.

Ichki sikldan foydalanishga misol

```
public class Listing4_3 {
public static void main (String [] args) {
for (int weekDay=1; weekDay <=7; weekDay++) {
      System.out.print («День недели: "+weekDay+" Часы:»);
for (int dayHour=1; dayHour <=24; dayHour++) {
      System.out.print (dayHour+«»);
}
      System.out.print (»\n»);
}
}
}
```

Mustaqil ish sifatida shunday qilingki, tashqi aylanada haftaning kun soni o'rniga uning nomi ko'rsatilsin. Buni hal qilish uchun sizning bilimingiz allaqachon yetarli, ilgari o'rganilgan operatorlardan birini

ishlatishda muammo. Ammo yechim dasturlash nuqtai nazaridan hali maqbul emas.

Tezda chiqish operatorlari

Ba'zida ma'lum bir vaziyat yuzaga kelganda siklning bajarilishini muddatidan oldin to'xtatish kerak bo'ladi. Buning uchun allaqachon tanish bo'lgan **break** operatoridan, shuningdek davom ettirish va qaytarish operatorlaridan foydalaning.

Break operatori joriy siklning bajarilishini butunlay to'xtatadi. Boshqarish sikldan keyingi buyruqlarga o'tkaziladi. Keling, quruq tavsiflardan o'rganib chiqamiz va birgalikda break operatoridan foydalanadigan dastur yozamiz. Ushbu dastur 1 dan 10 gacha bo'lgan tasodifiy sonni hosil qiladi va taklif qiladi.

Avvalo, tasodifiy son hosil qilamiz. Buni amalga oshirish uchun siz o'zingizni biroz oldinga surishingiz va ba'zi obyektga yo'naltirilgan dasturlash usullaridan foydalanishingiz kerak bo'ladi. Tasodifiy sonlar generatorini tasodifiy ravishda import qilamiz.

import java. util. Random;

Bu yerda biz kichik ekskursiya qilishimiz kerak. Tasodifiy sonlar generatori - bu oddiy kompyuter dasturi, bu baxtsiz hodisalar uchun joy bo'lmagan qat'iy algoritm. shuning uchun aslida, psevdo-tasodifiy sonlar hosil bo'ladi. Ehtimollar taqsimotining avlod oralig'ida bir xilligi generatorning sifatiga bog'liq.

Dastur har safar ishga tushirilganda generator bir xil sonli ketma-ketlikni ishlab chiqarishiga yo'l qo'ymaslik uchun uni dasturga nisbatan tasodifiy bo'lgan va ishga tushirishda takrorlanmaydigan ma'lum bir boshlang'ich qiymat bilan boshlash kerak. Amalda, generatorni ishga tushirish uchun kompyuterning milli soniyadagi tizim vaqti ko'pincha ishlatiladi. Dasturning boshlanish vaqti oldindan belgilanmagan va tizim soati bilan hech qanday aloqasi yo'q. Shuning uchun dasturning boshlanish vaqtini millisekundlik aniqlikda takrorlash ehtimoli yo'qolib ketadigan darajada kichik.

Shunday qilib, tasodifiy sinfning yangi obyektini yaratamiz va uni kompyuterning tizim vaqtidan millisekundlarda foydalanib ishga tushiramiz. Rnd nomli yangi obyekt bo'lsin:

Random rnd = new Random(System.currentTimeMillis ());

Butun sonni yaratish uchun nextInt (limit) usulidan foydalaning. Ushbu usul noldan chegaraga qadar, lekin qo'shilmasdan, yolg'on tasodifiy butun sonni hosil qiladi. Masalan, nextInt (10) usuli 0 dan 9 gacha, shu jumladan butun sonni qaytaradi.

Rnd obyektining nextInt () usuli yordamida 1 dan 10 gacha bo'lgan yolg'on tasodifiy raqam sirini yarating:

int secret = 1 + rnd.nextInt (10);

Soxta tasodifiy butun sonni yaratish uchun oxirgi kod qismi quyidagicha:

Random rnd = new Random(System.currentTimeMillis ());
int secret = 1 + rnd.nextInt (10);

Endi sizda "maxfiy" tasodifiy raqam mavjud bo'lib, unga o'zgaruvchi siri havola qilinadi. Yashirin qiymatni foydalanuvchi kiritgan qiymat bilan taqqoslashni amalga oshirish qoladi. Qiymatni kiritish haqidagi so'rov foydalanuvchiga qadar takrorlanishi kerak.

```
import javax.swing.JOptionPane;
import java. util. Random;
public class Listing4_4 {
public static void main (String [] args) {
Random rnd = new Random(System.currentTimeMillis ());
int secret = 1 + rnd.nextInt (10);
int userData;
String userInput;
while (true) {
// Выводим окно запроса
userInput = JOptionPane.showInputDialog («Угадайте число от 1 до 10»);
// Преобразуем строку в число в явном виде
userData = Integer.parseInt (userInput);
if (userData == secret) {
JOptionPane.showMessageDialog (null, «Вы угадали число!»);
break;
```

```
}
}
}
}
```

Savol necha marta berilishi kerakligi oldindan ma'lum bo'lmaganligi sababli, biz ataylab shartning o'rniga xizmat qiymati haqiqiy bo'lgan "abadiy" va halqani ishga tushiramiz. Har birida sikl orqali foydalanuvchi tomonidan kiritilgan raqamni dasturda ko'zda tutilgan qiymat bilan taqqoslaymiz. Agar mos keladigan bo'lsa, biz xabarni namoyish qilamiz va siklni majburan to'xtatamiz. Harakatlar soni o'ndan ortiq bo'lishi mumkin emas, shuning uchun dasturdan chiqishning boshqa usullari yo'q.

- Dastur tanasiga sinash hisoblagichini qo'shing. Hisoblagichning qiymati mosligini ko'rsatadigan oynada ko'rsatilsin: «Siz raqamni taxmin qildingiz! Harakatlar soni: ". Satrlarni birlashtirish va qatorlarni uzish uchun "\ n" xizmat ketma-ketligidan foydalaning.

Continue operatori

Davom etish jarayoni sikl tanasining bajarilishini to'xtatadi va siklning keyingi iteratsiyasiga erta o'tishni keltirib chiqaradi, masalan:

```
for (int i=1; i <=10; i++) {
if (i== (i/2) *2) {
      continue;
      }
System.out.println («i=" + i);
}
```

i == (i / 2) * 2 sharti faqat i qiymati juft bo'lsa, bajariladi, chunki i o'zgaruvchining turi int deb e'lon qilinadi. Toq sonni 2 ga bo'lishda kasr qismi bo'ladi va bekor qilinadi va 2 ga ko'paytirilgandan so'ng asl qiymati qaytmaydi. Agar ifoda to'g'ri bo'lsa, davom etuvchi operator yonadi va chiqishni chetlab o'tib, keyingi siklni takrorlashni chaqiradi. Shuning uchun konsol oynasida faqat toq raqamlar ko'rsatiladi.

Return operatori

Qaytish operatori odatda pastki dasturlardan chiqish uchun ishlatiladi va odatda tashqaridan foydalanilmaydi. Ammo, shuningdek, buyruqlar blokining bajarilishini muddatidan oldin to'xtatishi mumkin, shuning uchun biz uni ushbu bo'limda ko'rib chiqamiz.

Qaytish iborasi pastki dasturdan kalit so'zdan keyin ko'rsatilgan parametrni qaytarishi mumkin, masalan:

```
if (a <5) return a*20;
else return a*10;
```

Agar parametr ko'rsatilmagan bo'lsa, chaqiruvchi dasturga hech qanday qiymat o'tkazmasdan chiqadi.

Massiv va qatorlar.

Massiv - bu umumiy nom bilan birlashtirilgan, bir xil turdagi ma'lumotlarning tartiblangan to'plamidir. Aytaylik, biz bir nechta foydalanuvchilarning yoshini saqlamoqchi edik. UserAge1, userAge2, userAge3 va boshqalar nomli bir nechta o'zgaruvchini yaratishimiz mumkin. Ammo, bu holda, barcha qiymatlarni takrorlash kerak bo'lsa, o'zgaruvchiga kirish muammosi mavjud. Bundan tashqari, dasturni ishlab chiqishda, qancha foydalanuvchimiz bo'lishini aniq bilishimiz va oldindan ularning har biri uchun o'zgaruvchini e'lon qilishimiz kerak.

Siz yanada oqilona ish qilishingiz va userAge nomli ma'lumotlar qatorini e'lon qilishingiz mumkin. O'rnatilgan elementga murojaat qilish uchun elementning tartib raqami (indeks) ishlatiladi: userAge [i]. Java-da elementlarni indekslash noldan boshlanadi.

Massiv elementi boshqa massiv bo'lishi mumkin, u o'z navbatida massivlardan ham iborat bo'lishi mumkin. Noyob ko'rsatkich uchun ko'rsatiladigan indekslar soni elementni identifikatsiyalash massivning o'lchami deb ataladi. Massivning o'lchami o'zboshimchalik bilan bo'lishi mumkin, ammo amalda ko'pincha bir o'lchovli va ikki o'lchovli massivlardan foydalaniladi. Uch o'lchovli massivlar juda kam qo'llaniladi.

Bir o'lchovli massivlar

Massivni yaratishda massiv emas, balki massivga havolani o'z ichiga olgan o'zgaruvchi e'lon qilinadi. Massivni o'zi yaratish uchun (xotira katakchalarini ajratish) foydalanamiz.

```
int [] userAge;
userAge = new int [10];
```

Birinchi qatorda userAge o'zgaruvchisi e'lon qilinadi, bu butun sonli qator. Int kalit so'zidan keyin to'rtburchak qavslarga e'tibor bering. Ikkinchi satr bilan bog'langan o'nta butun massiv elementlarini saqlash uchun xotira ajratiladi, userAge deb nomlangan.

```
int [] userAge = new int [10];
```

Massivdagi elementlar soni massivning kattaligi deyiladi. Bir o'lchovli massivning kattaligi ko'pincha uzunlik deb nomlanadi. Massivdagi oxirgi elementning ko'rsatkichi uzunlikdan bittaga kam. Massivni xotirada saqlash uchun qachon e'lon qilingan bo'lsa, shuncha bo'sh joy ajratiladi.
Massivning hajmini aniqlash uchun uning uzunlik xususiyatiga murojaat qiling:

```
int a = userAge.length;
```

Bir o'lchovli massivga qiymat berish

Massivni yaratishda, agar asosiy tip elementlari haqida gapiradigan bo'lsak, uning katakchalari avtomatik ravishda nol bilan to'ldiriladi yoki agar massiv boshqalarga havolalardan iborat bo'lsa, nol qiymatlar bilan to'ldiriladi.

Obyektlar. Uni dasturda ishlatishdan oldin massivni boshlash kerak - massiv indekslariga mos qiymatlarni berish.

Siz qatorni to'g'ridan-to'g'ri deklaratsiya paytida ishga tushirishingiz mumkin:

```
int [] userAge = {28,32,19,44,52};
```

Ekvivalent, ammo murakkabroq sintaktik qurilish ruxsat berildi:
int [] userAge = new int [] {28,32,19,44,52};
Xuddi shunday, siz qator qiymatlari qatorini yaratishingiz va boshlashingiz mumkin:
String [] userName = {«Иван», «Петр», «Ольга», «Егор»};
Massiv elementlarini yakka tartibda boshlash mumkin:
userAge [0] = 28;
userAge [1] = 32;

Agar massivda ma'lum bir qonunga binoan shakllangan ba'zi bir ketma-ket ma'lumotlar bo'lishi kerak bo'lsa, unda massiv elementlarini ketma-ket takrorlanadigan massivni ishga tushirish uchun sikldan foydalanish qulay.

for operatorining maxsus formasi

For ifodasining maxsus shakli massiv elementlari ustida indekslarni ishlatmasdan to'g'ridan-to'g'ri takrorlashga imkon beradi. Ushbu holatda for bayonining qurilishi quyidagicha:

```
for (o'zgaruvchi turi: массив) {
// Buyruq qatori
}
```

Masalan, userAge qatori qiymatlari bo'yicha takrorlash davri quyidagicha ko'rinadi:

```
for (int age: userAge) {
System.out.println (age);
}
```

Ushbu misolda ish o'zgaruvchisi yoshi o'z navbatida userAge massividagi barcha elementlarning qiymatlarini oladi. Loop tanasida yangi o'zgaruvchisining joriy qiymati chop etiladi. Shunday qilib, biz userAge qatorining tarkibini chop etamiz.

For bayonotining maxsus shakli bilan biz faqat massiv elementlarining joriy qiymatlarini o'qiy olamiz. Massiv elementlarini

ishga tushirish yoki o'zgartirish uchun sizga kerak qator indekslari aniq takrorlanadigan muntazam sikldan foydalaning.

Quyidagi misolda birinchi sikl hatto [] qator elementlariga 2 dan 20 gacha bo'lgan juft qiymatlarni beradi. Keyin qiymatlarni ko'rsatish uchun for loopining maxsus shakli ishlatiladi. Chop etish uchun massivning barcha elementlari e'tiborga olinadi.

```
public class Listing5_1 {
public static void main (String [] args) {
int [] even = new int [10];
// Qiymat berish
for (int i=0;i <10;i++) {
even [i] = i*2+2;
}
// Massiv elementlarini ekranga chiqarish
for (int data: even) {
System.out.println (data);
}
}
}
```

Array o'zgaruvchilari mos yozuvlar turi o'zgaruvchilari deb nomlanadi. Bu shuni anglatadiki, massiv o'zgaruvchisi massiv saqlanadigan xotira maydoniga havolani saqlaydi. Shuning uchun ushbu o'zgaruvchiga boshqa massivga havola berilishi mumkin. Massivlar bir xil tipdagi va o'lchamdagi bo'lishi kerak, lekin o'lcham bir xil bo'lishi shart emas, chunki massiv o'zgaruvchisiga yangi ma'lumotlar emas, balki ularga yangi havola beriladi. Massivlarni tayinlash jarayoni oddiy, ammo bu aniq bo'lmagan oqibatlarga olib kelishi mumkin. Keling, oddiy topshiriq misolini ko'rib chiqaylik:

```
int [] first = {10,20,30,40};
int [] second = new int [6];
second = first;
first [2] = 50;
```

Birinchi qatorda biz to'rtta elementdan iborat massiv yaratamiz. Ikkinchi satrda oltita elementdan iborat massivni e'lon qilamiz.

Uchinchi satrda ikkinchi qator o'zgaruvchisiga birinchi qatorga havola beriladi. Buyruqni bajargandan so'ng, ikkala o'zgaruvchi bir xil qatorga murojaat qiladi. Sizningcha, birinchi [2] = 50 dan keyin ikkinchi [2] nimaga ega bo'ladi? To'g'ri, shuningdek, 50. Axir, bu bir xil xotira joylashuvi, uni turli xil o'zgaruvchilar deyishadi.

Ikki o'lchovli massivlar

Ikki o'lchovli qatorni satrlar va ustunlar jadvali sifatida tasavvur qilish eng osondir. Ikki o'lchovli massivning har bir elementi ikkita indeks - raqam bilan noyob tarzda aniqlanadi. Element joylashgan chorrahada qator va ustun raqami bir xil bo'lgan paytda element indeksi aniqlanadi. Afsuski, ushbu rasm tavsiflovchi bo'lsa-da, bu to'liq to'g'ri emas. Gap shundaki, ushbu "jadval" dagi qatorlar bir xil uzunlikda bo'lishi shart emas. Aniqrog'i, ikki o'lchovli massivni bir o'lchovli "tashqi" massiv deb hisoblash mumkin, uning elementlari bir o'lchovli "ichki" massivlarga havola. Birinchi indeks ichki qatorga havolani belgilaydi. Ikkinchi indeks ichki o'rnatilgan qator elementini aniqlaydi. Bunday holda, ichki massivlar har xil o'lchamlarda bo'lishi mumkinligi aniqroq. To'liq aytganda, bir xil o'lchamdagi ichki massivlarga ega bo'lgan ko'p o'lchovli massivlar - bu obyektlar to'plamining umumiy turidagi maxsus holat. Java tili ichki obyektlar turli o'lchamlarga ega bo'lgan to'plamlar bilan ishlashga imkon beradi. Ammo to'plamlarni chuqur o'rganish boshlang'ich kitobiga kirmaydi. Hozircha ichki o'rnatilgan massivlar turli uzunliklarga ega bo'lishi va ishda bo'lishini bilish kifoya bunday massivlarga ega bo'lgan asosiy xususiyatlar mavjud emas.

Ikki o'lchovli massiv bir o'lchovli kabi aniqlanadi:

```
int [] [] coord = new int [10] [15];
int [] [] coord;
coord = new int [10] [15];
```

Har bir indeks uchun raqamlash noldan boshlanadi. Massivning kattaligi massivning tashqi yoki ichki joylashganligiga bog'liq. Birinchi indeksdagi o'lcham ichki qatorlar sonini (qatorlar sonini) anglatadi:

int x = coord. length; // x = 10

Ikkinchi indeksdagi o'lchov ichki joylashtirilgan qator elementlari sonini bildiradi (ustunlar soni):

int y = coord [0].length; // y = 15

Klassik ikki o'lchovli massivni (to'plam emas) yaratishda barcha ichki massivlar bir xil darajada bo'lganligi sababli, birinchi indeksning ahamiyati yo'q. Muhim faqat u ichki qatorlar soniga ko'ra o'lchovlar oralig'ida bo'lishi uchun.

Ikki o'lchovli massivni ishga tushirish

Ikki o'lchovli qatorni qiymatlarni sistema qavslarda aniq ro'yxatlash orqali yaratishda boshlash mumkin.

int [] [] nums = {{4,9,12,0}, {2,7,3,5}};

Ichki sikllar belgilangan ma'lumotlar bilan massivni ishga tushirish uchun ishlatiladi. Endi 2D siklni boshlash dasturini yozishga quyidagi ketma-ketlikda ko'rsatilgan.

```
public class Listing5_2 {
public static void main (String [] args) {
// massivni e'lon qilish10x15
int [] [] coord = new int [10] [15];
// Tashqi massiv elementlari orqali takrorlash
for (int i=0;i <coord. length; i++) {
// Ichki massiv elementlari orqali takrorlash
for (int j=0;j <coord [0].length; j++) {
// Generatsiya qilish uchun ifodaga misol
coord [i] [j] = (i+j) *j;
}
}
// Yaratilgan qiymatlarni chop etish uchun chiqarish
for (int [] tmp1:coord) {
for (int tmp2:tmp1) {
```

```
System.out.print (tmp2+"\t»);
}
System.out.print (»\n»);
}
}
}
```

Keling, ushbu misolni batafsil ko'rib chiqaylik. 10x15 o'lchamdagi ikki o'lchovli massivni e'lon qilgandan so'ng, biz massivning katakchalarini ba'zi bir avtomatik ravishda yaratilgan ma'lumotlar bilan to'ldirish uchun ichki ko'chadan tashkil qilamiz. Loopning chegara parametri sifatida, masalan, massivning uzunligi uchun so'rovdan foydalanamiz.

```
for (int i=0;i <coord. length; i++) {
```

Esingizda bo'lsa, indekslash noldan boshlanadi va maksimal indeks massiv kattaligidan bittaga kam. Shuning uchun shartdan kam emas, kamroqdan foydalanadi yoki tengdir. Bu holda tashqi massivning kattaligi 10 ga teng, indekslar 0 dan 9 gacha bo'lgan qiymatlarni oladi. Xuddi shunday massiv elementlari ikkinchi indeks bilan sanab chiqiladi.

Ixtiyoriy ifoda qiymatlarni yaratish uchun ishlatiladi:

```
coord [i] [j] = (i+j) *j;
```

Buning o'rniga har qanday boshqa ibora yoki ma'lumotlar manbasini almashtirish mumkin. Faqatgina ifoda bilan qaytarilgan ma'lumotlar turi massivning ma'lumotlar turiga mos kelishi muhimdir.

Ma'lumotni shakllantirgandan so'ng, biz uni tekshirish uchun chop etamiz. Odatdagidek takrorlash uchun for bayonining qisqa shaklidan foydalanamiz. Ikki o'lchovli qator bo'lsa, ba'zi nozikliklar mavjud. Tashqi va ichki sikl deklaratsiyalaridagi sikl o'zgaruvchilar turlariga e'tibor bering:

```
for (int [] tmp1:coord) {
for (int tmp2:tmp1) {
```

Tmp1 o'zgaruvchisi to'rtburchak qavsli int [] turi deb e'lon qilinadi, chunki tashqi massivning elementlari o'zlari massivdir (ya'ni tashqi siklda biz takrorlaymiz). Tmp2 o'zgaruvchisi to'rtburchak qavssiz int deb e'lon qilinadi, chunki ichki massivning elementlari butun sonlardir.

Bosilgan qiymatlar "\t" belgisi ketma-ketligi yordamida yorliqlar bilan ajratiladi:

System.out.println (tmp2+"\t»);

Java tilidagi massivlar bilan ishlash uchun java.util.Arrays sinfida tavsiflangan standart usullar mavjud. Usullarni chaqirishdan oldin siz sinfni import qilishingiz kerak:

import java.util.Arrays;

Eslatib o'tamiz, import buyruqlari paketni nomlash bayonotidan so'ng darhol paydo bo'lishi kerak, lekin asosiy sinf e'lonidan oldin. Agar standart paket ishlatilsa, u holda dastur to'g'ridan-to'g'ri sinflarni import qilish bilan boshlanadi. Keyinchalik, biz kundalik amaliyotda ishlatiladigan massivlar bilan ishlashning asosiy usullarini batafsil ko'rib chiqamiz. Usullarning to'liq ro'yxatini bu erda topishingiz mumkin:

https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html

equals () - massivlarni taqqoslash uchun usul qo'llaniladi. Ma'lumki, massiv o'zgaruvchisi massivga havolani saqlaydi. Agar ikkita massiv bir xil bo'lsa-da, lekin turli joylarda saqlanadigan bo'lsa, u holda ular turli xil havolalarga ega bo'ladi va o'zgaruvchilarni oddiy taqqoslash salbiy natijani noto'g'ri qaytaradi. Shuning uchun massivlarni taqqoslash uchun kataklar sonini, katakchalar tarkibini va ularning tartibini taqqoslaydigan maxsus usul qo'llaniladi. Agar parametrlardan kamida bittasi mos kelmasa, taqqoslash natijasi salbiy bo'ladi. Masalan, uchta massivni olaylik:

```
int [] arr1 = {5,3,4,6,8,10};
int [] arr2 = {5,3,4,6,8,10};
int [] arr3 = {10,8,6,4,3,5};
```

boolean result1 = Arrays. equals (arr1, arr2);
boolean result2 = Arrays. equals (arr1, arr3);

Arr1 va arr2 massivlarini taqqoslash haqiqiy bo'ladi, chunki massivlar har jihatdan mos keladi. Arr1 va arr3 massivlarini taqqoslash false qiymatini beradi, chunki ular bir xil qiymatlar tartibiga ega emas.

copyOfRange () - asl massivning bir qismini boshqa massivga nusxalash. Usul uchun uchta argument kerak: manba, boshlang'ich indeks, yakuniy indeks. Aytaylik, bizda e'lon qilingan qator bor:

int [] source = {-2, -1,0,1,2,3,4,5,6};
Buyruq bajarilgandan keyin
int [] dest = Arrays.copyOfRange (source,2,5);

{0,1,2} qiymatlari yangi qatorga ko'chiriladi. E'tibor bering, ikkinchi ko'rsatkichga qadar, lekin indeksga ega bo'lmagan elementlar ko'chiriladi. Shuning uchun indeksli element 5 nusxa ko'chirilmaydi.

toString () - massiv tarkibini mag'lubiyatga aylantirish. Bu massiv tarkibini bosib chiqarishning oson usuli, masalan:

int [] arr = {3,8,10,1,6};
System.out.println(Arrays.toString (arr));
Ekranga *[3, 8, 10, 1, 6]* chiqadi.

sort () - massiv elementlarini ortish tartibida saralash. sort () usuli yangi qatorni qaytarmaydi. Bu shunchaki mavjud bo'lgan narsani o'zgartiradi. Aytaylik, biz qator e'lon qildik:

int [] arr = {10,3, -1,6,0};
Arrays.sort (arr);
System.out.println(Arrays.toString (arr));
Ekranga *[-1, 0, 3, 6, 10]* chiqadi.

binarySearch() - berilgan qator indeksini saralangan qatordan qidirish. Agar siz massiv elementlari qiymatlari o'sish tartibida ekanligiga amin bo'lmasangiz, unda oldin binarySearch () usuli

yordamida qatorni sort () usuli yordamida saralashingiz kerak. Aytaylik, bizda tartiblangan qator bor:

int [] arr = {2,7,15,42,56,78};
int myIndex = Arrays.binarySearch (arr, 56);

MyIndex o'zgaruvchisiga 4 qiymat beriladi. Bu 56 qiymatiga ega bo'lgan massiv elementining indeksidir. Agar qatorda bo'lmagan qiymatni argument sifatida keltirsangiz nima bo'ladi? Biz g'alati natijaga erishamiz, bu alohida tushuntirishni talab qiladi. Masalan, harakat qilib ko'raylik:

18 qiymatining indeksini toping:

int myIndex = Arrays.binarySearch (arr, 18);

myIndex o'zgaruvchisiga -4 qiymati beriladi. Minus shuni anglatadiki, bunday qiymat topilmadi. 3 raqami bu qiymat qanday indeksga ega bo'lishini anglatadi bir qatorda. Ammo ba'zi bir sabablarga ko'ra ushbu ko'rsatkichga bittasi qo'shildi! Boshqacha qilib aytganda, agar massiv massivi 18 ga teng bo'lsa, u holda u 4-1 = 3 ko'rsatkichiga ega bo'lar edi.

Avvalgi mavzularda aytib o'tganimizdek, massiv uzunligi (elementlar soni) uzunlik xususiyati orqali aniqlanadi, shuning uchun massiv uzunligini aniqlash usuli qo'llanilmaydi.

Qatorlar

To'liq aytganda, Java-dagi mag'lubiyat ma'lumotlar turi emas, balki o'rnatilgan String sinfining namunasidir. String sinfida satrlar bilan ishlashning o'ziga xos usullari mavjud. Torli sinf, Arrays sinfidan farqli o'laroq, u importni talab qilmaydi. Biz satrlar va massivlar haqida gaplashayotganimiz bejiz emas. Matn qatorini har bir belgi o'ziga xos tartib raqamiga (indeksiga) ega bo'lgan tartiblangan qiymatlar to'plami deb hisoblash mumkin. String bilan bog'langan o'zgaruvchi xotira maydoniga havolani saqlaydi. Bunda satrlar massivga o'xshaydi.

Qatorlarni turli yo'llar bilan yaratishingiz mumkin. Eng aniq narsa, String sinfining namunasini aniq yaratishdir:

String str = «Hello, World!»;

Siz bo'sh String obyektini yaratishingiz mumkin:

String str = new String ();
Siz qatorlar qatoridan qator yaratishingiz mumkin:
char [] chars = {«J», 'a', 'v', 'a'};
String str = new String (chars);

Satrlar qator elementlari bo'lishi mumkin:

String [] userNames = {«Uktam», «Madaminov», «Ataxanovich»};

Satrlarni + operatori yordamida birlashtirish mumkin. Ushbu operatsiyaga mag'lubiyat birikmasi deyiladi:

String str1 = «Java»;
String str2 = «Language»;
String str3 = str1 + str2;
+ = Tayinlash operatorining qisqartirilgan shakliga ruxsat beriladi:
String str3 += «Language»;

Ip - bu o'zgarmas obyekt. Agar mag'lubiyatni manipulyatsiya qilish natijasida uning matni o'zgarsa, unda aslida xotirada yangi satr hosil bo'ladi va satr o'zgaruvchisiga yangi mos yozuvlar tayinlanadi. Agar eski chiziq boshqa joyda ishlatilmasa, avtomatik uni olib tashlaydi va xotirani bo'shatadi.

Qatorlar bilan ishlovchi metodlar

Java tili qatorlar bilan ishlash uchun ko'plab foydali usullarni taklif etadi. Ushbu qo'llanmada biz faqat eng zarur narsalarni sanab o'tamiz.

charAt () - satr boshidan boshlab belgilangan ofsetdagi belgini qaytaradi. Ortga hisoblash noldan boshlanadi. Salbiy va mavjud bo'lmagan indeks qiymatlaridan foydalanmang. Usulqator elementiga indeks bo'yicha kirishga o'xshaydi:

String lang = «Java»;
char myChr = lang.charAt (2); // myChr = «v»
contains () – moslikni topish:
String str = «Codemagic»;
boolean tmp = str.contains («mag»); // true qaytaradi

endsWith () - satr berilgan belgilar qatori bilan tugaganligini tekshiradi:

String str = «Codemagic»;

boolean tmp = str. endsWith («magic»); // true qaytaradi

***BeginWith** () usuli xuddi shu tarzda satr berilgan belgilar qatoridan boshlanganligini tekshiradi.*

***equals** () - satrlarni taqqoslaydi va agar belgilar soni, ularning tartibi va ishi bir xil bo'lsa, mantiqiy haqiqatni qaytaradi:*

String str1 = «Java program»;

String str2 = «Java Program»;

boolean cmp1 = str1.equals (str2); // false – moslik to'g'ri kelmaydi

boolean cmp2 = str1.equals («Java program»); // true – to'g'ri keladi

***equalsIgnoreCase** () - qatorlarni befarq holda taqqoslaydi.*

***length** () - bo'shliqlarni o'z ichiga olgan qatordagi belgilar sonini qaytaradi.*

***split** () - qatorni belgilangan ajratuvchiga qarab qismlarga ajratadi va qator fragmentlarini qaytaradi:*

String names = «Mobil ilovalarni ishlab chiqish»;

String [] splitNames = names. split (»,»);

Ushbu misolda split () usuli qatorni qaytaradi {"Vasiliy", "Piter", "Olga", "Igor"}.

***substring** () - berilgan qatorni qaytaradi. Argument sifatida boshlang'ich belgi indeksini va oxiridan keyin belgi indeksini ko'rsating:*

String str1 = «Hello, Java»;

String str2 = str1.substring (0,4); // str2 = «Hell»

String str3 = str1.substring (7); // str3 = «Java»

Agar usul uchun argument sifatida faqat bitta indeks ko'rsatilgan bo'lsa, u holda fragment belgilangan indeksdan mag'lubiyatning oxirigacha olinadi.

toUpperCase () / **toLowerCase** () - satrdagi barcha belgilar ishini katta / kichik harfga aylantirish:

String str1 = «Hello, Java»;

String str2 = str1.toUpperCase (); // str2 = «HELLO, JAVA»;

trim () - satr boshida va oxirida bo'sh joylarni va xizmat belgilarini olib tashlaydi.

4-MAVZU: CLASS VA OBYEKTLAR

Obyektga yo'naltirilgan dasturlash (OOP-object orientation programming) asoslarini yaxshi bilishda, class larning va ularning obyetlarini yaratish, dastur jarayonida qo'llash bilimlarini egallash juda muhim hisoblanadi. Har qanday holatda, OOP tushunchasini tushunmasdan, siz javada dasturlashingiz mumkin emas.

Oldinga qarab, obyektga yo'naltirilgan yondashuv barcha muammolar uchun davolovchi vosita emas va barcha holatlar uchun vosita emasligini ta'kidlaymiz.

Java 8, lambda ifodalarini qo'shdi, bu bilan kechiktirilgan kodni bajarish va dasturlash hodisalari bilan ishlashni amalga oshirish ancha qulaydir. Bu bo'ladi. "Nima uchun bizga OOP kerak va u qanday ishlaydi?" Degan savolga javob. ko'plab maqolalar va kitoblarga bag'ishlangan. Agar siz dasturlashni jiddiy boshlashga qaror qilsangiz, siz ma'lumotlarning qatorini chuqur o'rganmasdan qilolmaysiz. Ammo bu keyinroq bo'ladi. Bu birinchi marta, ayniqsa, sevimli mashg'ulot darajasida dasturlash uchun yetarli bo'ladi.

Class va obyektlarni yaratish

Sinf tavsifi sinf kalit so'zidan boshlanadi, so'ngra sinf nomi va sistemali qavslar blokiga joylashtirilgan:

```
class nomi {
// class tasnifi
}
```

Faqatgina maydonlardan iborat bo'lgan va metodlarni o'z ichiga olmagan sinf tavsifining namunasini ko'rib chiqamiz.

```
class MyFields {
int data;
char letter;
}
// Dasturning asosiy usuli bilan sinf tavsifi
// Tavsif shablonini NetBeans avtomatik ravishda yaratadi
sinf ro'yxati6_1 {
```

```
// Asosiy usul
public static void main {
// MyFields sinfining obyektini yarating
MyFields demo = yangi MyFields ();
// Maydonlarga qiymatlarni belgilash
demo. ma'lumotlar = 1234;
demo. xat = "B";
// Bosib chiqarish uchun maydon qiymatlarini chop eting
System.out.println ("Raqam:" + demo. Ma'lumotlar);
System.out.println ("Letter:" + demo. Letter);
}
}
```

Ushbu misol MyFields odatiy sinfini tavsiflaydi, u faqat ikkita maydondan iborat - butun son va belgi. Bu shunchaki tavsif bo'lsa-da, biz dalalarga murojaat qila olmaymiz. Sinf tavsifiga asoslanib, demo nomli obyekt (sinf misoli) yaratiladi. Endi biz obyekt maydonlariga kirishimiz, ularga qiymatlarni belgilashimiz va o'qishimiz mumkin. Boshqacha qilib aytganda, sinf - bu tavsif, va sinf obyekti - bu manipulyatsiya qilinishi mumkin bo'lgan moddiy shaxs. Dasturda bir sinfning bir nechta obyektlarini yaratishimiz va ularga har xil nomlarni berishimiz mumkin. Obyekt maydoniga murojaat qilish uchun avval obyekt nomini, va nuqta o'tgandan so'ng maydon nomini ko'rsating. Endi faqat metodlarni o'z ichiga olgan sinfni tavsiflaylik quyidagi misolda ko'rsatilgan. Usulni tavsiflashda, bajariladigan buyruqlar blokidan tashqari, qaytish natijasining turini, ismini ko'rsatishingiz kerak

Usul va argumentlar ro'yxati. Agar usul natija bermagan bo'lsa, unda tip identifikatori void kalit so'zidir.

Usul local o'zgaruvchilardan foydalanishi mumkin. Ular obyekt maydonlaridan tubdan farq qiladi, chunki ularga faqat metod tanasi ichida kirish mumkin va ular mavjud usul ishlaydi. Usul tugagandan so'ng, local o'zgaruvchilar xotiradan o'chiriladi.

```
// Maxsus sinf tavsifi
MyClass sinfi {
// Qo'shishni amalga oshiradigan usulning tavsifi
int summ (int a, int b) {
int summa = a + b;
qaytish summasi;
```

```
}
// Ko'paytirishni amalga oshiradigan usulning tavsifi
int proiz (int a, int b) {
int proizvedenie = a * b;
qaytish proizvedenie;
}
}
umumiy sinf Listing6_2 {
public static void main (String [] arglar) {
// MyClass sinfining obyektini yarating
MyClass testi = yangi MyClass ();
// Qo'shimchani bajaradigan usulni chaqiring
System.out.println ("Raqamlar yig'indisi 4 + 5 =" + test.summ
(4,5));
// Ko'paytirishni amalga oshiradigan usulni chaqiring
System.out.println ("5 * 6 raqamlari mahsuloti =" + test.proiz
(5,6));
}
}
```

Bu misolda biz ikkita usulni o'z ichiga olgan sinfni tavsifladik: ikkita butun sonni qo'shish va ikkita butun sonni ko'paytirish. Usullarda faqat metod buyruqlar blokini bajarish paytida mavjud bo'lgan local a va b o'zgaruvchilar ishlatiladi. Dasturning asosiy usulida biz sinf obyekti yaratamiz va unga o'zgaruvchan obyekt testiga havola beramiz. Usulni chaqirish va unga dalillarni yetkazish uchun biz foydalanamiz. Usul (dalillar). Biz bir xil sinfdagi obyektlarni istaganimizcha yaratishimiz mumkin, shuning uchun usulni chaqirishda avval qaysi birini belgilash kerak. Bu biz nazarda tutgan obyekt va keyin nuqta orqali usul nomini ko'rsatadi. Dasturni soddalashtirish uchun usul chaqiruvi to'g'ridan-to'g'ri qatorni bosib chiqarish buyrug'iga joylashtiriladi.

Endi siz oddiy sinflarni tavsiflashingiz va ular asosida obyektlar yaratishingiz mumkin. Trening uchun dasturingizni yozing. Unda maydonlarni o'z ichiga olgan sinfni tavsiflang. Dasturga foydalanuvchiga ikkita butun sonni kiritish uchun modal dialog oynalaridan foydalanishga ruxsat bering. Keyin ushbu raqamlarni qo'shish va ko'paytirish natijalari dialog oynasida ko'rsatilishi kerak.

```
import javax.swing.JOptionPane;
MyClass sinfi {
// Sinf maydonlari
int fieldOne;
int fieldTwo;
// Maydonlarga qiymatlarni berish usuli
bekor to'plami (int a, int.b) {
fieldOne = a;
fieldTwo = b;
}
// Maydon qiymatlarini ko'paytirish usuli
int ko'paytmasi () {
return fieldOne * fieldTwo;
}
// Maydon qiymatlarini yig'ish usuli
int summ () {
return fieldOne + fieldTwo;
}
}
jamoat klassi Listing6_3 {
public static void main (String [] arglar) {
// Asosiy sinf o'zgaruvchilarini e'lon qiling
int input1, input2;
String inputString;
// Bizning sinfimiz obyektini yarating
MyClass obj = yangi MyClass (); // Birinchi qiymatni kiritish
uchun oyna
inputString = JOptionPane.showInputDialog ("Birinchi
qiymatni kiriting");
input1 = Integer.parseInt (inputString);
// Ikkinchi qiymatni kiritish uchun oyna
inputString = JOptionPane.showInputDialog ("Ikkinchi qiymatni
kiriting");
input2 = Integer.parseInt (inputString);
// Obyekt maydonlariga qiymatlarni berish usulini chaqiring
obj.set (input1, input2);
// dialog oynasiga qo'shilish natijasini chiqaring
```

JOptionPane.showMessageDialog (null, "Qo'shish natijasi:" + obj.summ ());

// Ko'paytirish natijasini dialog oynasida ko'rsating

JOptionPane.showMessageDialog (null, "Ko'paytirish natijasi:" + obj.multiply ());

}

}

Savol va topshiriqlar:

1. Java dasturlash tili qachon yaratilgan?
2. Java dasturlash tili qanday maqsadlarda qo'llaniladi?
3. Javani o'rantish ketma-ketligini ko'rsating.
4. Java uchun maxsus IDE lar va ularning imkoniyatlariga misollar keltiring.
5. O'zgaruvchilar va ularning tiplari haqida ma'lumot bering.
6. Java tili operatorlari va ularning ishlash prinspi qanday?
7. Qanday string metodlarni bilasiz?
8. Fayllar bilan ishlash qanday amalga oshiriladi.
9. Java tilining Andoidda qo'llanilishi.
10. Java Androida misollar keltiring.

5-MAVZU: JAVA DASTURLASH TILIDA MOBIL ILOVALAR YARATISH

Android ilovalarida hodisalar (actions) va jarayonlar(process) dastur ishlash jarayonida aniq bir ishni qilish uchun ishlatiladi. Har bir dasturning aniq bir ishni bajarish uchun hodisalar bilan ishlash qismi mavjud. Keling bunga bir oddiy misol keltiramiz. Biz bilamizki kalkulyator dasturida sonlar va amallarni bajarish uchun tugmalarni bosish orqali amalga oshiramiz. Dasturda har bir tugmaning o'z vazifasi bor ya'ni tugmani bosish orqali vazifasi bajariladi. Dasturning asosiy qismi uning orqa tomonida (backend) tugma bosilish (onClick) hodisasiga dastur kodlari yozilgan bo'ladi. Bu tushunishingiz uchun bir oddiy misol. Qolgan dasturlarda ham tugma yoki boshqa komponentalar hodisalari mavjud bo'ladi va shu orqali ishlaydi. Quyidagi misolda tugma bosilgandagi hodisasiga misol ko'ramiz.

```
package com.example.practicum1;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
  Button btn_3;
  @Override
  protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    btn_3 = findViewById(R.id.form_3);
    btn_3.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Intent in = new Intent(MainActivity.this,
Main3Activity.class);
        startActivity(in);
        finish();
      }
    });
```

Bu yerda **btn_3** tugma nomi va **setOnClickListener** xizmatchi so'zi yordamida hodisa yaratayabmiz. Quyidagi misolda boshqa komponenta hodisasini ko'ramiz.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="example.javatpoint.com.seekbar.MainActivity">

      <SeekBar
      android:id="@+id/seekBar"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginEnd="8dp"
      android:layout_marginStart="8dp"
      android:layout_marginTop="372dp"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>

ackage example.javatpoint.com.seekbar;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
   SeekBar seekBar;
   @Override
```

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

    seekBar=(SeekBar)findViewById(R.id.seekBar);
    seekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
      @Override
      public void onProgressChanged(SeekBar seekBar, int
progress,
                                    boolean fromUser) {
        Toast.makeText(getApplicationContext(),"seekbar progress:
"+progress, Toast.LENGTH_SHORT).show();
      }

      @Override
      public void onStartTrackingTouch(SeekBar seekBar) {
        Toast.makeText(getApplicationContext(),"seekbar touch
started!", Toast.LENGTH_SHORT).show();
      }
      @Override
      public void onStopTrackingTouch(SeekBar seekBar) {
        Toast.makeText(getApplicationContext(),"seekbar touch
stopped!", Toast.LENGTH_SHORT).show();
      }
    });
  }
}
```

Hodisalar

public static interface View.OnClickListener
android.view.View.OnClickListener

Public methods

abstract onClick(View v)
void Qachonki ko'rinish bosilganda chaqiriladi.

onClick

public abstract void onClick (View v)
Qachonki view bosilganda chaqiriladi.

Parameters

void Ko'rinish: view bosilganda.

View.OnContextClickListener

public static interface View.OnContextClickListener
android.view.View.OnContextClickListener
view context bosilganda.

Public metodlari

abstract onContextClick(View v)
boolean view context bosilganda.

View.OnFocusChangeListener

public static interface View.OnFocusChangeListener
android.view.View.OnFocusChangeListener

Public metodlari

abstract onFocusChange(View v, boolean hasFocus)

void Ko'rinishning focus holati o'zgarganda chaqiriladi.

onFocusChange

public abstract void onFocusChange (View v, boolean hasFocus)

View.OnScrollChangeListener

public static interface View.OnScrollChangeListener
android.view.View.OnScrollChangeListener

Public metodlar

abstract void	onScrollChange(View v, int scrollX, int scrollY, int oldScrollX, int oldScrollY) Scroll position holati o'zgarganda chaqiriladi.

onScrollChange

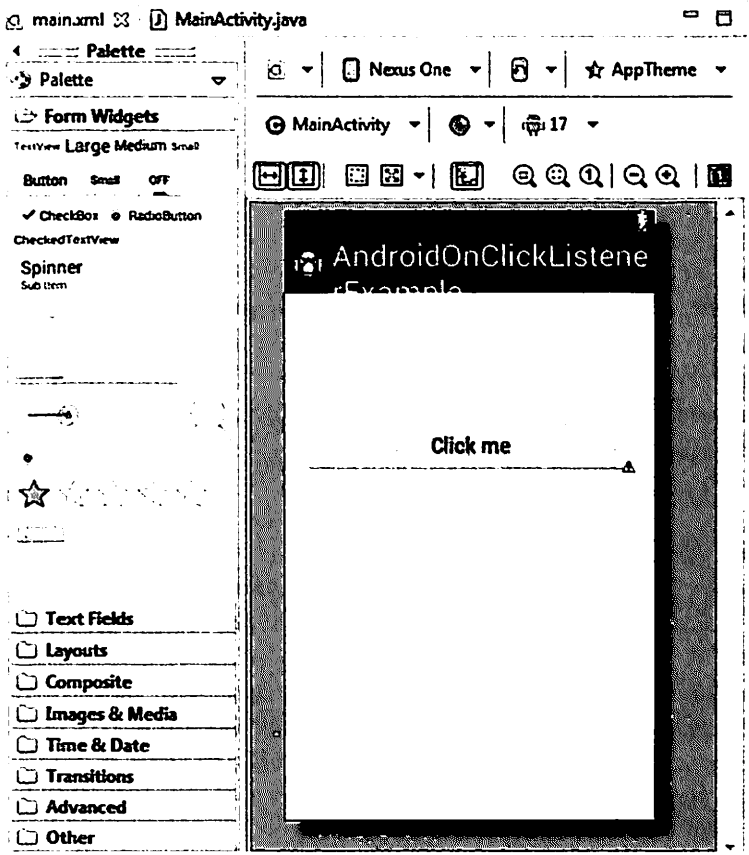
public abstract void onScrollChange (View v, int scrollX, int scrollY, int oldScrollX, int oldScrollY)

scroll position ko'rinish holati o'zgarganda chaqiriladi.

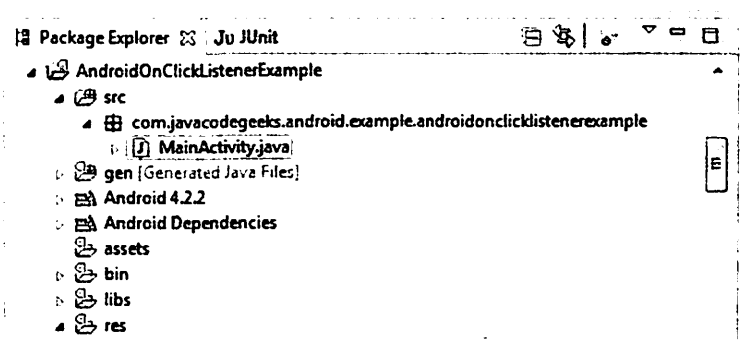
Parametrlar

V	View: scroll position o'zgarganda.
scrollX	int: Joriy gorizontal aylantirishning kelib chiqishi.
scrollY	int: Joriy vertikal aylantirishning kelib chiqishi.
oldScrollX	int: Oldingi gorizontal scroll joylashuvi.
oldScrollY	int: Oldingiz vertical scroll joylashuvi.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity" >
   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentTop="true"
      android:layout_marginTop="92dp"
      android:text="Click me" />
</RelativeLayout>
```



4.1-rasm. Ilova bosh oynasi



4.2-rasm. Loyiha fayllari

```
package
com.javacodegeks.android.example.androidonclicklistenerexample;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.view.View.OnClickListener;
public class MainActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is
present.
getMenuInflater().inflate(R.menu.main, menu);
Button button = (Button) findViewById(R.id.button1);
button.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View view) {
Toast.makeText(MainActivity.this, "Button
Clicked",Toast.LENGTH_SHORT).show();
} });
return true;
}
}
```

onClick metodi

```
public void clickFunc(View view){
Toast.makeText(MainActivity.this, "Button
Clicked",Toast.LENGTH_SHORT).show();
}
```

Savol va topshiriqlar:

1. Hodisalar haqida nimalarni bilasiz?
2. Hodisa turlarini aytib bering.
3. Hodisalar uchun maxsus metodlarga misollar keltiring.
4. Jarayonlar va ularni boshqarish haqida nimalarni bilasiz?
5. JOptionPane komponenta hodisalari haqida ma'lumot bering.
6. Java Androidda hodisalar haqida nimalarni bilasiz?
7. Qanday maxsus kutubxonalarni bilasiz?
8. Android Studio komponentalari hodisalariga misollar keltiring.
9. onScrollChange hodisasi haqida aytib bering.
10. Hodisa parametrlari va uning turlariga misol keltiring.

6-MAVZU: ANDROIDDA LAYOUTLARDAN FOYDALANISH. INTENT. VIEW

View ba Layout - faoliyat yoki vidjet kabi foydalanuvchi interfeysi uchun vizual tuzilmani tartibga soladi. XML-da maket e'lon qilinadi, shu jumladan unda paydo bo'ladigan ekran elementlari. Dasturga ekran obyektlarining holatini, shu jumladan XML da e'lon qilingan holatini o'zgartirish uchun kod qo'shilishi mumkin.

Har bir ViewGroup (e.g. LinearLayout, RelativeLayout, CoordinatorLayout, etc.) ga xususiyatlarga oid ma'lumotlar kerak bo'ladi. Ushbu ma'lumotlar ViewGroup.LayoutParams paketidagi sinf obyektlarida saqlanadi.

Muayyan tartib turiga xos parametrlarni kiritish uchun ViewGroups ViewGroup.LayoutParams sinfining pastki sinflaridan foydalanadi..

- LinearLayout bu LinearLayout.LayoutParams
- RelativeLayout bu RelativeLayout.LayoutParams
- CoordinatorLayout bu CoordinatorLayout.LayoutParams

ViewGrouplarning aksariyati o'z farzandlari uchun chekka joylarni belgilash qobiliyatini qayta tiklaydilar, shuning uchun ular to'g'ridan-to'g'ri ViewGroup.LayoutParams subklassini emas, balki ular o'rniga ViewGroup. MarginLayoutParams subklassini (o'zi ViewGroup.LayoutParams subklassi).

Xml da LayoutParams

LayoutParams moslamalari xml formatidagi kengaytirilgan fayl asosida yaratiladi.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">

    <TextView
       android:layout_width="wrap_content"
       android:layout_height="50dp"
       android:layout_gravity="right"
       android:gravity="bottom"
       android:text="Example text"
       android:textColor="@android:color/holo_green_dark"/>
```

```
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@android:color/holo_green_dark"
        android:scaleType="centerInside"
        android:src="@drawable/example" />

</LinearLayout>
```

Layout_ bilan boshlanadigan barcha parametrlar, qanday qilib tartibni ishlashini belgilaydi. Tartibni o'rnatganda, ushbu parametrlar tegishli LayoutParams obyektiga qo'shiladi, keyinchalik u Layout tomonidan ViewGroup ichidagi ma'lum bir ko'rinishni to'g'ri joylashtirish uchun ishlatiladi. View ning boshqa atributlari to'g'ridan-to'g'ri view bilan bog'liq va Viewning o'zi tomonidan qayta ishlanadi.

TextView uchun:

- layout_width, layout_height va layout_gravity LinearLayout.LayoutParams obyektida saqlanadi va LinearLayout tomonidan ishlatiladi
- gravity, text va textColor dan TextView o'zi foydalanadi

ImageView uchn:

- layout_width, layout_height va layout_weight LinearLayout.LayoutParams obyektida saqlanadi va LinearLayout tomonidan ishlatiladi
- background, scaleType va src ImageView o'zi tomonidan ishlatiladi

LayoutParams obyektini olish

getLayoutParams - bu hozirgi LayoutParams obyektini olish imkonini beradigan View usulidir.

LayoutParams obyekti to'g'ridan-to'g'ri yopiladigan ViewGroup bilan bog'liq bo'lganligi sababli, bu usul faqat View ViewGroup-ga biriktirilganida null bo'lmagan qiymatni qaytaradi. Shuni yodda tutishingiz kerakki, ushbu obyekt doimo mavjud bo'lmasligi mumkin. Ayniqsa, siz View konstruktorida bo'lishiga bog'liq bo'lmasligingiz kerak.

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setupView(context);
    }

    private void setupView(Context context) {
        if (getLayoutParams().height == 50){  // DO NOT DO THIS!
                                              // This might produce NullPointerException
            doSomething();
        }
    }

    //...
}
```

Agar LayoutParams obyektiga bog'liq bo'lishni istasangiz, buning o'rniga onAttachedToWindow usulidan foydalanishingiz kerak.

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
    }



    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        if (getLayoutParams().height == 50) { // getLayoutParams() will NOT return null here
            doSomething();
        }
    }

    //...
}
```

LayoutParams obyektini chaqirish

Muayyan ViewGroup-ga xos xususiyatlardan foydalanishingiz kerak bo'lishi mumkin (masalan, RelativeLayout qoidalarini dasturiy ravishda o'zgartirishingiz mumkin). Buning uchun ViewGroup.LayoutParams obyektini qanday qilib to'g'ri tashlab qo'yishni bilishingiz kerak bo'ladi.

Bu aslida boshqa ViewGroup bo'lgan ko'rinishi uchun LayoutParams obyektini olishda biroz chalkash bo'lishi mumkin.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/outer_layout"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">

    <FrameLayout
       android:id="@+id/inner_layout"
       android:layout_width="match_parent"
       android:layout_height="50dp"
       android:layout_gravity="right"/>

</LinearLayout>
```

IMPORTANT: LayoutParams obyekti turi to'g'ridan-to'g'ri **ENGLOSING** ViewGroup turiga bog'liq.

Noto'g'ri translatsiya:

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
                                    // INCORRECT! This will produce ClassCastException
```

Correct casting:

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
                                    // CORRECT! the enclosing layout is a LinearLayout
```

CoordinatorLayout va Behaviors

CoordinatorLayout - bu juda kuchli FrameLayout va ushbu ViewGroup-ning maqsadi uning ichidagi qarashlarni muvofiqlashtirishdir.

Koordinator Layout-ning asosiy ko'rinishi - bu XML formatidagi animatsiyalar va ko'rinishlarning o'tishini muvofiqlashtirish imkoniyatiga ega.

CoordinatorLayout ikkita asosiy foydalanish uchun mo'ljallangan:

: Yuqori darajadagi dastur dekorasi yoki chrome tartibi sifatida

: Bir yoki bir nechta aniq ta'sir o'tkazish uchun

Oddiy Behavior yaratish

Behavior yaratish uchun CoordinatorLayout.Behavior class ni kengaytiring.

CoordinatorLayout.Behavior-ni kengaytirish

Misol:

```
public class MyBehavior<V extends View> extends CoordinatorLayout.Behavior<V> {

    /**
     * Default constructor.
     */
    public MyBehavior() {
    }

    /**
     * Default constructor for inflating a MyBehavior from layout.
     *
     * @param context The {@link Context}.
     * @param attrs The {@link AttributeSet}.
     */
    public MyBehavior(Context context, AttributeSet attrs) {
      super(context, attrs);
    }
}
```

Ushbu xatti-harakatni chaqirish uchun koordinatorning joylashuvi ko'rinishidagi bolaga biriktirish kerak.

Xatti-harakatni dasturiy ravishda biriktirish

```
MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) view.getLayoutParams();
params.setBehavior(myBehavior);
```

XML

XML-da xatti-harakatni biriktirish uchun layout_behavior atributidan foydalanishingiz mumkin:

<View

```
android:layout_height="...."
android:layout_width="...."
app:layout_behavior=".MyBehavior" />
```

Behavior ni avtomatik ravishda biriktirish: Agar siz odatiy ko'rinish bilan ishlayotgan bo'lsangiz, @ CoordinatorLayout.DefaultBehavior izohi yordamida xatti-harakatlarni biriktirishingiz mumkin.:

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {

}
```

ViewPager siz TabLayout-ni qo'llash

Ko'pincha TabLayout-dan ViewPager-dan foydalaniladi. TabLayout-dan foydalanib, ViewPager-ni ishlatmasdan TabLayout-dan foydalanish mumkin. OnTabSelectedListener. Birinchidan, o'zingizning XML faylingizga TabLayout qo'shing:

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout" />
```

Faoliyat davomida navigatsiya qilish uchun tanlangan yorliq asosida foydalanuvchi interfeysini qo'lda to'ldiring

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        int position = tab.getPosition();
        switch (tab.getPosition()) {
            case 1:
                getSupportFragmentManager().beginTransaction()
                    .replace(R.id.fragment_container, new ChildFragment()).commit();
                break;
            // Continue for each tab in TabLayout
    }

    @Override
    public void onTabUnselected(TabLayout.Tab tab) {

    }

    @Override
    public void onTabReselected(TabLayout.Tab tab) {

    }
});
```

Maxsus ko'rinishlarni yaratish

Agar sizga to'liq moslashtirilgan ko'rinish kerak bo'lsa, siz "View" (barcha Android versiyalarining super klassi) subklassini va o'z o'lchamlarini (onMeasure (...)) va chizish (onDraw (...)) usullarini taqdim etishingiz kerak:

1. O'zingizning maxsus qolibingizni yarating: bu asosan har bir maxsus ko'rinish uchun bir xil. Bu yerda biz SmileyView deb nomlangan tabassumni chizish mumkin bo'lgan maxsus ko'rinish uchun qolip yaratamiz:

```
public class SmileyView extends View {
    private Paint mCirclePaint;
    private Paint mEyeAndMouthPaint;

    private float mCenterX;
    private float mCenterY;
    private float mRadius;
    private RectF mArcBounds = new RectF();

    public SmileyView(Context context) {
        this(context, null, 0);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        initPaints();
    }

    private void initPaints() {/* ... */}

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {/* ... */}

    @Override
    protected void onDraw(Canvas canvas) {/* ... */}
}
```

Initialize paints: Bo'yoq obyektlari - bu sizning geometrik obyektlaringiz qanday ko'rinishini aniqlaydigan virtual chizma cho'tkalari (masalan, rang, to'ldirish va chegara stili va boshqalar). Bu erda biz ikkita bo'yoq, aylana uchun bitta sariq rangli bo'yoq va ko'zlar va og'iz uchun bitta qora turdagi bo'yoqlari yaratamiz.:

```
private void initPaints() {
    mCirclePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mCirclePaint.setStyle(Paint.Style.FILL);
    mCirclePaint.setColor(Color.YELLOW);
    mEyeAndMouthPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mEyeAndMouthPaint.setStyle(Paint.Style.STROKE);
    mEyeAndMouthPaint.setStrokeWidth(16 * getResources().getDisplayMetrics().density);
    mEyeAndMouthPaint.setStrokeCap(Paint.Cap.ROUND);
    mEyeAndMouthPaint.setColor(Color.BLACK);
}
```

2. O'zingizning onMeasure(...) metodingizni amalga oshiring: Odatid viewda (e.g. FrameLayout) bo'lishini talab qiladi. Bu qo'llanilish balandligi va kengligini aniqlash uchun foydalanishingiz mumkin bo'lgan o'lchovlar to'plamini taqdim etadi. Bu yerda balandligi va kengligi bir xil bo'lishiga ishonch hosil qilib kvadrat hosil qilamiz:

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int w = MeasureSpec.getSize(widthMeasureSpec);
    int h = MeasureSpec.getSize(heightMeasureSpec);

    int size = Math.min(w, h);
    setMeasuredDimension(size, size);
}
```

OnMeasure (...) setMeasuredDimension (..) uchun kamida bitta chaqiruvni o'z ichiga olishi kerak, aks holda sizning odatiy ko'rinishingiz IllegalStateException bilan ishdan chiqadi.

3. O'zingizning onSizeChanged (...) usulingizni amalga oshiring: bu sizning ko'rsatuv kodingizni to'g'ri sozlash uchun odatiy ko'rinishingizning hozirgi balandligi va kengligini ushlab turishga imkon beradi. Bu yerda biz faqat markaz va radiusni hisoblaymiz:

```
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    mCenterX = w / 2f;
    mCenterY = h / 2f;
    mRadius = Math.min(w, h) / 2f;
}
```

4. O'zingizning onDraw (...) usulingizni amalga oshiring: bu yerda siz o'zingizning haqiqiy ko'rinishingizni amalga oshirasiz. Siz chizishingiz mumkin bo'lgan Canvas obyektini taqdim etadi (mavjud barcha rasm usullari uchun maxsus Canvas hujjatlarini ko'ring).

```
@Override
protected void onDraw(Canvas canvas) {
    // draw face
    canvas.drawCircle(mCenterX, mCenterY, mRadius, mCirclePaint);
    // draw eyes
    float eyeRadius = mRadius / 5f;
    float eyeOffsetX = mRadius / 3f;
    float eyeOffsetY = mRadius / 3f;
    canvas.drawCircle(mCenterX - eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
mEyeAndMouthPaint);
    canvas.drawCircle(mCenterX + eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
mEyeAndMouthPaint);
    // draw mouth
    float mouthInset = mRadius /3f;
    mArcBounds.set(mouthInset, mouthInset, mRadius * 2 - mouthInset, mRadius * 2 -
mouthInset);
    canvas.drawArc(mArcBounds, 45f, 90f, false, mEyeAndMouthPaint);
}
```

5. **Maxsus viewni layoutga qo'shish:** maxsus ko'rinish endi sizdagi har qanday tartib fayllariga qo'shilishi mumkin. Bu yerda biz uni faqat FrameLayout da ko'ramiz:

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">



    <com.example.app.SmileyView
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

Loyihangizni ko'rish kodi tugagandan so'ng qurish tavsiya etilishini unutmang. Uni qurmasdan siz Android Studio-da oldindan ko'rish ekranida ko'rinishini ko'rmaysiz.

View larga atribut qo'shish

Maxsus ko'rinishlar shuningdek Android layout resurslarida ishlatilishi mumkin bo'lgan maxsus atributlarni qabul qilishi mumkin. Viewga atributlarni qo'shish uchun quyidagilarni bajarish kerak:

1. Atributlaringizning nomi va turini aniqlang: bu res / values / attrs.xml ichida amalga oshiriladi (agar kerak bo'lsa, uni yarating). Bizning tabassum yuzimiz uchun rang atributi va tabassumning ifodasi uchun enum atributi:

```
<resources>
    <declare-styleable name="SmileyView">
        <attr name="smileyColor" format="color" />
        <attr name="smileyExpression" format="enum">
            <enum name="happy" value="0"/>
            <enum name="sad" value="1"/>
        </attr>
    </declare-styleable>
    <!-- attributes for other views -->
</resources>
```

2. **Atributlaringizni layout ichida qo'llang:** bu sizning odatiy ko'rinishingizdan foydalanadigan har qanday maket ichida amalga oshirilishi mumkin. Quyidagi tartibda sariq rangli tabassum bilan ekran paydo bo'ladi:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <com.example.app.SmileyView
        android:layout_height="56dp"
        android:layout_width="56dp"
        app:smileyColor="#ffff00"
        app:smileyExpression="happy" />
</FrameLayout>
```

Maxsus atributlar asboblar bilan ishlamaydi: Android Studio 2.1 va undan yuqori versiyalarida oldindan (va ehtimol kelajakdagi versiyalarida). Ushbu misolda app: smileyColor-ni vositalar bilan almashtirish: smileyColor smileyColor-ga na ish vaqtida va na dizayn vaqtida o'rnatilishiga olib keladi.

3. **Atributlarni o'qish**: bu sizning odatiy view source kodingiz ichida amalga oshiriladi. SmileyView-ning quyidagi parchalari atributlarni qanday chiqarish mumkinligini namoyish etadi:

```
public class SmileyView extends View {
    // ...

    public SmileyView(Context context) {
        this(context, null);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);

        TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
defStyleAttr, 0);
        mFaceColor = a.getColor(R.styleable.SmileyView_smileyColor, Color.TRANSPARENT);
        mFaceExpression = a.getInteger(R.styleable.SmileyView_smileyExpression,
Expression.HAPPY);
        // Important: always recycle the TypedArray
        a.recycle();

        // initPaints(); ...
    }
}
```

4. **(Optional) Odatiy stilni qo'shish:** Bu standart qiymatlar bilan uslubni qo'shish va uni sizning odatiy view ga yuklash orqali amalga oshiriladi. Quyidagi standart tabassum uslubi baxtiyor sariq rangni anglatadi:

```
<!-- styles.xml -->
<style name="DefaultSmileyStyle">
    <item name="smileyColor">#ffff00</item>
    <item name="smileyExpression">happy</item>
</style>
```

Qanday qilib SmileyView-da qo'llaniladigan, uni getStyledAttributes uchun qo'ng'iroqning so'nggi parametri sifatida qo'shish (kodni 3-qadamda ko'ring):

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView, defStyleAttr,
R.style.DefaultSmileyViewStyle);
```

O'rnatilgan tartibda har qanday atribut qiymatlari (2-bosqichdagi kodga qarang) standart uslubning mos qiymatlarini bekor qiladi.

5. **(Ixtiyoriy) Mavzular ichida uslublarni taqdim etish**: Bu sizning mavzularingizda ishlatilishi mumkin bo'lgan yangi uslub mos yozuvlar atributini qo'shish va ushbu xususiyat uchun uslubni taqdim etish orqali amalga oshiriladi. Bu yerda biz oddiygina smileyStyle atributimizga nom beramiz:

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

Keyin biz dastur mavzusida uslubni taqdim etamiz (bu yerda biz standart uslubni faqat 4-bosqichdan foydalanamiz):

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
</style>
```

Murakkab view yaratish

Murakkab ko'rinish - Bu atrofdagi dastur kodlari tomonidan bitta ko'rinish sifatida qaraladigan maxsus ViewGroup hisoblanadi. Bunday ViewGroup DDD-ga o'xshash dizaynda haqiqatan ham foydali bo'lishi mumkin, chunki u yig'indiga mos kelishi mumkin, bu misolda esa - contact. Contact ko'rsatiladigan hamma joyda uni qayta ishlatish mumkin. Bu kodni qayta ishlatishni osonlashtiradi va SOLID priciples ga ko'ra yaxshiroq dizaynni yaratadi.

layout XML: Odatda bu siz boshlagan joy. Sizda mavjud bo'lgan bir oz XML bor, siz <include /> sifatida qayta ishlatishingiz ham mumkin bo'ladi. Uni alohida XML faylga chiqaring va asosiy tegni **<merge>** elementiga bilan bog'lang:

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

        <ImageView
            android:id="@+id/photo"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_alignParentRight="true" />

        <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/photo" />

        <TextView
            android:id="@+id/phone_number"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/name"
            android:layout_toLeftOf="@id/photo" />
</merge>
```

Ushbu XML-fayl Layout Editor Android Studioda mukammal ishlaydi. Siz uni boshqa har qanday tartib kabi ko'rib chiqishingiz mumkin.

ViewGroup

XML faylini olganingizdan so'ng, maxsus viewgroup yarating.

```
import android.annotation.TargetApi;
import android.content.Context;
import android.os.Build;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.ImageView;
import android.widget.TextView;

import myapp.R;

/**
 * A compound view to show contacts.
 *
 * This class can be put into an XML layout or instantiated programmatically, it
 * will work correctly either way.
 */
public class ContactView extends RelativeLayout {

    // This class extends RelativeLayout because that comes with an automatic
    // (MATCH_PARENT, MATCH_PARENT) layout for its child item. You can extend
    // the raw android.view.ViewGroup class if you want more control. See the
    // note in the layout XML why you wouldn't want to extend a complex view
    // such as RelativeLayout.

    // 1. Implement superclass constructors.
    public ContactView(Context context) {
        super(context);
        init(context, null);
    }

    // two extra constructors left out to keep the example shorter

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public ContactView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init(context, attrs);
    }

    // 2. Initialize the view by inflating an XML using 'this' as parent
    private TextView mName;
    private TextView mPhoneNumber;
    private ImageView mPhoto;

    private void init(Context context, AttributeSet attrs) {
        LayoutInflater.from(context).inflate(R.layout.contact_view, this, true);
        mName = (TextView) findViewById(R.id.name);
        mPhoneNumber = (TextView) findViewById(R.id.phone_number);
        mPhoto = (ImageView) findViewById(R.id.photo);
    }

    // 3. Define a setter that's expressed in your domain model. This is what the example is
    //    all about. All controller code can just invoke this setter instead of fiddling with
    //    lots of strings, visibility options, colors, animations, etc. If you don't use a
```

```
    //    custom view, this code will usually end up in a static helper method (bad) or copies
    //    of this code will be copy-pasted all over the place (worse).
    public void setContact(Contact contact) {
        mName.setText(contact.getName());
        mPhoneNumber.setText(contact.getPhoneNumber());
          if (contact.hasPhoto()) {
              mPhoto.setVisibility(View.VISIBLE);
              mPhoto.setImageBitmap(contact.getPhoto());
          } else {
              mPhoto.setVisibility(View.GONE);
          }
    }
}
```

Init (Context, AttributeSet) usuli - har qanday maxsus XML atributlarini ko'rish va qo'shish uchun qo'llaniladi. Ushbu qismlar mavjud bo'lganda siz uni o'zingizning ilovangizda ishlatishingiz mumkin.

XML da foydalanish

Bu yerda fragment_contact_info.xml misolida bitta ContactView-ni qanday qilib xabarlar ro'yxatiga qo'yish aks ettirilgan:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- The compound view becomes like any other view XML element -->
    <myapp.ContactView
        android:id="@+id/contact"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/message_list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>

</LinearLayout>
```

Code-da foydalanish

Kontaktlar ro'yxatini ko'rsatadigan RecyclerView.Adapter-ga misol. Ushbu misol, View manipulyatsiyasidan butunlay xalos bo'lganda, kontroller kodining qanchalik ahamiyatini ko'rsatadi.

```
package myapp;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.ViewGroup;

public class ContactsAdapter extends RecyclerView.Adapter<ContactsViewHolder> {

    private final Context context;

    public ContactsAdapter(final Context context) {
        this.context = context;


    }

    @Override
    public ContactsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        ContactView v = new ContactView(context); // <--- this
        return new ContactsViewHolder(v);
    }

    @Override
    public void onBindViewHolder(ContactsViewHolder holder, int position) {
        Contact contact = this.getItem(position);
        holder.setContact(contact);  // <--- this
    }

    static class ContactsViewHolder extends RecyclerView.ViewHolder {

        public ContactsViewHolder(ContactView itemView) {
            super(itemView);
        }

        public void setContact(Contact contact) {
            ((ContactView) itemView).setContact(contact); // <--- this
        }
    }
}
```

Savol va topshiriqlar:

1. Dastur interfeysida maxsus ko'rinishlar qanday amalga oshiriladi?
2. onMeasure metodi nima uchun xizmat qiladi?
3. Ob'yekt o'zgaruvchanligi deganda nimani tushunasiz?
4. View va layoutlarga misollar keltiring.
5. Activity yaratish ketma-ketligi qanday amalga oshiriladi?
6. View ob'yekti atributlariga nimalar kiradi?
7. Ob'yektlarga stillar qanday qo'shiladi?
8. ViewGroup qanday holatlarda qo'llaniladi?
9. Forma yaratishda Java fayl o'rni qanday?
10. Modal ko'rinish qanday amalga oshiriladi?
11. View haqida nimalarni bilasiz?

12. Foydalanuvchi interfeysi uchun zarur bo'lgan resurslarga misollar.
13. Layout va ularni yaratish ketma-ketligini ko'rsating.
14. res papkasi strukturasini tushuntiring.
15. Rasm va galereya joylashtirishni ko'rsating.
16. Layoutning qanday turlari mavjud?
17. ViewGroup deganda nimani tushunasiz?
18. Ob'yekt olish usullari qanday?
19. Contex nima?
20. View larni guruhlash qanday amalga oshiriladi?
21. Android studioda rasmlar joylashuvi haqida aytib bering.
22. Dastur bazasidagi .xml faylli iconlarni qanday yaratsa bo'ladi?
23. Image komponentasi atributlarifa nimalar kiradi?
24. Rasmlarni ekranda to'liq yoki qisman joylashtirish qanday amalga oshiriladi?
25. Image bilan ishlovchi kutubxonalar qanday qo'llaniladi?
26. Menular va ularning turlari haqida nimalarni bilasiz?
27. Menular yaratilish ketma-ketligini ko'rsating?
28. Menularga icon qo'yish qanday amalga oshiriladi?
29. Menu stillarini o'zgartirish qanday amalga oshiriladi?
30. Kontekst menularni yaratish haqida nimalarni bilasiz?

7-MAVZU: MA'LUMOTLAR BAZASI BILAN ISHLASH

SQLite - bu C tilida yozilgan relyatsion ma'lumotlar bazasini boshqarish tizimidir. Android doirasida SQLite ma'lumotlar bazalari bilan ishlashni boshlash uchun SQLiteOpenHelper dasturini kengaytiradigan sinfni aniqlash va kerak bo'lganda sozlash lozim bo'ladi. Sqlite local ma'lumotlar bazasidan foydalanish uchun avval uning kutubxonasini loyihamizga qo'shishimiz lozim.

```
dependencies {
   def sqlite_version = "2.1.0"
   // Java language implementation
   implementation "androidx.sqlite:sqlite:$sqlite_version"
}
```

Version 2.2.0

androidx.sqlite:sqlite:2.2.0-alpha01, androidx.sqlite:sqlite-framework:2.2.0-alpha01, va androidx.sqlite:sqlite-ktx:2.2.0-alpha01
SQLiteOpenHelper klassi SQLite ma'lumotlar bazasidan foydalanish imkoniyatini beradi.
SQLiteOpenHelper sinfi Android.database.sqlite.SQLiteOpenHelper sinfi ma'lumotlar bazasini yaratish va versiyalarni boshqarish uchun ishlatiladi. Ma'lumotlar bazasining har qanday operatsiyasini bajarish uchun SQLiteOpenHelper sinfining onCreate () va onUpgrade () usullarini amalga oshirishni ta'minlashingiz kerak.
SQLiteOpenHelper sinfining constructorlari:

Constructor	Tavsif
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	ma'lumotlar bazasini yaratish, ochish va boshqarish uchun obyekt yaratadi.
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	ma'lumotlar bazasini yaratish, ochish va boshqarish uchun obyekt yaratadi. Bu xatolarni ko'rib chiquvchini belgilaydi.

SQLiteOpenHelper sinfining metodlari

Method	Tavsif
public abstract void onCreate(SQLiteDatabase db)	ma'lumotlar bazasi birinchi marta yaratilganda faqat bir marta chaqiriladi.
public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	ma'lumotlar bazasini yangilash zarur bo'lganda chaqiriladi.
public synchronized void close ()	ma'lumotlar bazasi obyektini yopadi.
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	ma'lumotlar bazasini pasaytirish kerak bo'lganda chaqiriladi.

SQLiteDatabase sinfi

U sqlite ma'lumotlar bazasida yaratish, yangilash, o'chirish, tanlash va hk kabi usullarni o'z ichiga oladi. Sinf metodlari bilan tanishib chiqsak.

Method	Tavsif
void execSQL(String sql)	sql so'rovini tanlamagan holda bajaradi.
long insert(String table, String nullColumnHack, ContentValues values)	ma'lumotlar bazasiga yozuv qo'shadi. Jadvalda jadval nomi ko'rsatilgan, nullColumnHack to'liq bo'sh qiymatlarga yo'l qo'ymaydi. Agar ikkinchi argument null bo'lsa, agar bo'sh bo'lsa, android null qiymatlarni saqlaydi. Uchinchi argument saqlanadigan qiymatlarni belgilaydi.
int update(String table, ContentValues values, String whereClause, String[] whereArgs)	qatorni yangilaydi.
Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)	natijalar to'plami ustiga kursorni qaytaradi.

Android SQLite ma'lumotlar bazasi misoli

```
package example.javatpoint.com.sqlitetutorial;
public class Contact {
      int _id;
   String _name;
   String _phone_number;
   public Contact(){   }
   public Contact(int id, String name, String _phone_number){
      this._id = id;
      this._name = name;
      this._phone_number = _phone_number;
   }
   public Contact(String name, String _phone_number){
      this._name = name;
      this._phone_number = _phone_number;
   }
   public int getID(){
```

```
    return this._id;
  }
  public void setID(int id){
    this._id = id;
  }
  public String getName(){
    return this._name;
  }
  public void setName(String name){
    this._name = name;
  }
  public String getPhoneNumber(){
    return this._phone_number;
  }
  public void setPhoneNumber(String phone_number){
    this._phone_number = phone_number;
  }
}
```

onUpgrade() metodi

SQLiteOpenHelper - ma'lumotlar bazasini yaratish va versiyalarni boshqarishni boshqarish uchun yordamchi sinf hisoblanadi.

Bu sinfda, onUpgrade() metodi bazaga o'zgartirish kiritilganda ma'lumotlar bazasini yangilash uchun javobgardir. U ma'lumotlar bazasi mavjud bo'lganda chaqiriladi, lekin uning versiyasi dasturning amaldagi versiyasida ko'rsatilganidan pastroq. Ma'lumotlar bazasining har bir versiyasi uchun siz kiritgan aniq o'zgarishlar qo'llanilishi kerak.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Loop through each version when an upgrade occurs.
    for (int version = oldVersion + 1; version <= newVersion; version++) {
        switch (version) {

            case 2:
                // Apply changes made in version 2
                db.execSQL(
                    "ALTER TABLE " +
                    TABLE_PRODUCTS +
                    " ADD COLUMN " +
                    COLUMN_DESCRIPTION +
                    " TEXT;"
                );
                break;

            case 3:
                // Apply changes made in version 3
                db.execSQL(CREATE_TABLE_TRANSACTION);
                break;
        }
    }
}
```

Savol va topshiriqlar

1. Mobil ilovalarda ma'lumotlar bazasi tushunchasi.
2. SQLite database haqida nimalarni bilasiz?
3. Ma'lumotlar bazasi bilan ishlovchi maxsus funksiyalar?
4. ContentValues class ning vazifasi?
5. Bazadan ma'lumotlarni o'chirish qanday amalga oshiriladi?
6. Bazadan ma'lumotlarni o'zgartirish qanday amalga oshiriladi?
7. Bazani yaratish qanday amalga oshiriladi?
8. Drop table ning vazifasi?
9. SQLite buyruqlari?
10. SQLiteDatabase classning vazifasi?

8-MAVZU: SO'ROVLAR YARATISH

Cursor dan ma'lumotlarni o'qish

SQLiteOpenHelper subclass ichida joylashgan metodga misol. Bu natijalarni o'zgartirish uchun searchTerm String dan foydalanadi, cursor tarkibida takrorlanadi va ushbu tarkibni product obyektlari ro'yxatiga qaytaradi.

Birinchidan, ma'lumotlar bazasidan olingan har bir qator uchun konteyner bo'lgan Product POJO sinfini ishlab chiqing:

```
public class Product {
  long mId;
  String mName;
  String mDescription;
  float mValue;
  public Product(long id, String name, String description, float value) {
    mId = id;
    mName = name;
    mDescription = description;
    mValue = value;
  }
}
```

So'ngra ma'lumotlar bazasidan so'rov o'tkazadigan metodni toping va product obyektlari ro'yxatini qaytaring:

```
Contact getContact(int id) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.query(TABLE_CONTACTS, new String[] {
KEY_ID,
            KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
        new String[] { String.valueOf(id) }, null, null, null, null);
    if (cursor != null)
      cursor.moveToFirst();
    Contact contact = new Contact(Integer.parseInt(cursor.getString(
0)),
        cursor.getString(1), cursor.getString(2));
    // return contact
    return contact;
  }
```

```
public List<Product> searchForProducts(String searchTerm) {

    // When reading data one should always just get a readable database.
    final SQLiteDatabase database = this.getReadableDatabase();

    final Cursor cursor = database.query(
            // Name of the table to read from
            TABLE_NAME,

            // String array of the columns which are supposed to be read
            new String[](COLUMN_NAME, COLUMN_DESCRIPTION, COLUMN_VALUE),

            // The selection argument which specifies which row is read.
            // ? symbols are parameters.
            COLUMN_NAME + " LIKE ?",

            // The actual parameters values for the selection as a String array.
            // ? above take the value from here
            new String[]("%" + searchTerm + "%"),

            // GroupBy clause. Specify a column name to group similar values
            // in that column together.
            null,

            // Having clause. When using the GroupBy clause this allows you to
            // specify which groups to include.
            null,

            // OrderBy clause. Specify a column name here to order the results
            // according to that column. Optionally append ASC or DESC to specify
            // an ascending or descending order.
            null
    );

    // To increase performance first get the index of each column in the cursor
    final int idIndex = cursor.getColumnIndex(COLUMN_ID);
    final int nameIndex = cursor.getColumnIndex(COLUMN_NAME);
    final int descriptionIndex = cursor.getColumnIndex(COLUMN_DESCRIPTION);
    final int valueIndex = cursor.getColumnIndex(COLUMN_VALUE);

    try {

        // If moveToFirst() returns false then cursor is empty
        if (!cursor.moveToFirst()) {
            return new ArrayList<>();
        }

        final List<Product> products = new ArrayList<>();

        do {

            // Read the values of a row in the table using the indexes acquired above
            final long id = cursor.getLong(idIndex);
            final String name = cursor.getString(nameIndex);
            final String description = cursor.getString(descriptionIndex);
            final float value = cursor.getFloat(valueIndex);

            products.add(new Product(id, name, description, value));

        } while (cursor.moveToNext());
```

```
        return products;

    } finally {
        // Don't forget to close the Cursor once you are done to avoid memory leaks.
        // Using a try/finally like in this example is usually the best way to handle this
        cursor.close();

        // close the database
        database.close();
    }
}
```

SQLiteOpenHelper class-ni qo'llash

```
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Example.db";
    private static final int DATABASE_VERSION = 1;

    // For all Primary Keys _id should be used as column name
    public static final String COLUMN_ID = "_id";

    // Definition of table and column names of Products table
    public static final String TABLE_PRODUCTS = "Products";
    public static final String COLUMN_NAME = "Name";
    public static final String COLUMN_DESCRIPTION = "Description";
    public static final String COLUMN_VALUE = "Value";

    // Definition of table and column names of Transactions table
    public static final String TABLE_TRANSACTIONS = "Transactions";
    public static final String COLUMN_PRODUCT_ID = "ProductId";
    public static final String COLUMN_AMOUNT = "Amount";

    // Create Statement for Products Table
    private static final String CREATE_TABLE_PRODUCT = "CREATE TABLE " + TABLE_PRODUCTS + "  (" +
            COLUMN_ID + " INTEGER PRIMARY KEY, " +
            COLUMN_DESCRIPTION + " TEXT, " +
            COLUMN_NAME + " TEXT, " +
            COLUMN_VALUE + " REAL" +
            ");";
```

```
    // Create Statement for Transactions Table
    private static final String CREATE_TABLE_TRANSACTION = "CREATE TABLE " + TABLE_TRANSACTIONS + "
(" +
            COLUMN_ID + " INTEGER PRIMARY KEY," +
            COLUMN_PRODUCT_ID + " INTEGER," +
            COLUMN_AMOUNT + " INTEGER," +
            " FOREIGN KEY (" + COLUMN_PRODUCT_ID + ") REFERENCES " + TABLE_PRODUCTS + "(" +
COLUMN_ID + ")" +
            ");";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // onCreate should always create your most up to date database
        // This method is called when the app is newly installed
        db.execSQL(CREATE_TABLE_PRODUCT);
        db.execSQL(CREATE_TABLE_TRANSACTION);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // onUpgrade is responsible for upgrading the database when you make
        // changes to the schema. For each version the specific changes you made
        // in that version have to be applied.
        for (int version = oldVersion + 1; version <= newVersion; version++) {
            switch (version) {

                case 2:
                    db.execSQL("ALTER TABLE " + TABLE_PRODUCTS + " ADD COLUMN " +
COLUMN_DESCRIPTION + " TEXT;");
                    break;

                case 3:
                    db.execSQL(CREATE_TABLE_TRANSACTION);
                    break;
            }
        }
    }
}
```

Bazaga ma'lumot joylash

```
// You need a writable database to insert data
final SQLiteDatabase database = openHelper.getWritableDatabase();

// Create a ContentValues instance which contains the data for each column
// You do not need to specify a value for the PRIMARY KEY column.
// Unique values for these are automatically generated.
final ContentValues values = new ContentValues();
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());
```

```
// This call performs the update
// The return value is the rowId or primary key value for the new row!
// If this method returns -1 then the insert has failed.
final int id = database.insert(
        TABLE_NAME, // The table name in which the data will be inserted
        null,       // String: optional; may be null. If your provided values is empty,
                    // no column names are known and an empty row can't be inserted.
                    // If not set to null, this parameter provides the name
                    // of nullable column name to explicitly insert a NULL

        values      // The ContentValues instance which contains the data
);
```

Ommaviy qo'shimchalar

Ma'lumotlarning katta qismlarini bir vaqtning o'zida kiritishning bir misoli. Siz kiritmoqchi bo'lgan barcha ma'lumotlar ContentValues qatoriga to'plangan.

```
@Override
public int bulkInsert(Uri uri, ContentValues[] values) {
    int count = 0;
    String table = null;

    int uriType = IChatContract.MessageColumns.uriMatcher.match(uri);
    switch (uriType) {
        case IChatContract.MessageColumns.MESSAGES:

            table = IChatContract.MessageColumns.TABLE_NAME;
            break;
    }
    mDatabase.beginTransaction();
    try {
        for (ContentValues cv : values) {
            long rowID = mDatabase.insert(table, " ", cv);
            if (rowID <= 0) {
                throw new SQLException("Failed to insert row into " + uri);
            }
        }
        mDatabase.setTransactionSuccessful();
        getContext().getContentResolver().notifyChange(uri, null);
        count = values.length;
    } finally {
        mDatabase.endTransaction();
    }
    return count;
}
```

Bu yerda qanday foydalanish haqida bir misol ko'rib o'tamiz:

```
ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//add whatever you like to the valueList
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);
```

Androidda SQLite uchun Helper va Provider, Contract yaratish

```
//Define the tables and columns of your local database
public final class DBContract {
/*Content Authority its a name for the content provider, is convenient to use the package app name to
be unique on the device */

    public static final String CONTENT_AUTHORITY = "com.yourdomain.yourapp";

    //Use CONTENT_AUTHORITY to create all the database URI's that the app will use to link the
content provider.
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);

    /*the name of the uri that can be the same as the name of your table.
    this will translate to content://com.yourdomain.yourapp/user/ as a valid URI
    */
    public static final String PATH_USER = "User";

    // To prevent someone from accidentally instantiating the contract class,
    // give it an empty constructor.
    public DBContract () {}

    //Intern class that defines the user table
    public static final class UserEntry implements BaseColumns {
        public static final URI CONTENT_URI =
BASE_CONTENT_URI.buildUpon().appendPath(PATH_USER).build();

        public static final String CONTENT_TYPE =
ContentResolver.CURSOR_DIR_BASE_TYPE+"/"+CONTENT_AUTHORITY+"/"+PATH_USER;


        //Name of the table
        public static final String TABLE_NAME="User";

        //Columns of the user table
        public static final String COLUMN_Name="Name";
        public static final String COLUMN_Password="Password";

        public static Uri buildUri(long id){
            return ContentUris.withAppendedId(CONTENT_URI,id);
        }
}
```

DBHelper.java

```
public class DBHelper extends SQLiteOpenHelper{

//if you change the schema of the database, you must increment this number
private static final int DATABASE_VERSION=1;
static final String DATABASE_NAME="mydatabase.db";
private static DBHelper mInstance=null;
public static DBHelper getInstance(Context ctx){
    if(mInstance==null){
    mInstance= new DBHelper(ctx.getApplicationContext());
    }
    return mInstance;
}

public DBHelper(Context context){
    super(context,DATABASE_NAME,null,DATABASE_VERSION);
}

public int GetDatabase_Version() {
    return DATABASE_VERSION;
}

@Override
public void onCreate(SQLiteDatabase sqLiteDatabase){
//Create the table users
final String SQL_CREATE_TABLE_USERS="CREATE TABLE "+UserEntry.TABLE_NAME+ " ("+
UserEntry._ID+" INTEGER PRIMARY KEY, "+
UserEntry.COLUMN_Name+" TEXT , "+
UserEntry.COLUMN_Password+" TEXT "+
" ); ";

sqLiteDatabase.execSQL(SQL_CREATE_TABLE_USERS);

}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + UserEntry.TABLE_NAME);
}

}
```

DBProvider.java

```
public class DBProvider extends ContentProvider {

    private static final UriMatcher sUriMatcher = buildUriMatcher();
    private DBHelper mDBHelper;
    private Context mContext;

    static final int USER = 100;

    static UriMatcher buildUriMatcher() {

        final UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
        final String authority = DBContract.CONTENT_AUTHORITY;

        matcher.addURI(authority, DBContract.PATH_USER, USER);

        return matcher;
    }

    @Override
    public boolean onCreate() {
        mDBHelper = new DBHelper(getContext());
        return false;
    }

    public PeaberryProvider(Context context) {
        mDBHelper = DBHelper.getInstance(context);
        mContext = context;
    }

    @Override
    public String getType(Uri uri) {
        // determine what type of Uri is
        final int match = sUriMatcher.match(uri);
```

```
    switch (match) {
        case USER:
            return DBContract.UserEntry.CONTENT_TYPE;

        default:
            throw new UnsupportedOperationException("Uri unknown: " + uri);
    }
}

@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
                    String sortOrder) {
    Cursor retCursor;
    try {
        switch (sUriMatcher.match(uri)) {
            case USER: {
                retCursor = mDBHelper.getReadableDatabase().query(
                        DBContract.UserEntry.TABLE_NAME,
                        projection,
                        selection,
                        selectionArgs,
                        null,
                        null,
                        sortOrder
                );
                break;
            }
            default:
                throw new UnsupportedOperationException("Uri unknown: " + uri);
        }
    } catch (Exception ex) {
        Log.e("Cursor", ex.toString());
```

```
        } finally {
            mDBHelper.close();
        }
        return null;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        Uri returnUri;
        try {

            switch (match) {
                case USER: {
                    long _id = db.insert(DBContract.UserEntry.TABLE_NAME, null, values);
                    if (_id > 0)
                        returnUri = DBContract.UserEntry.buildUri(_id);
                    else
                        throw new android.database.SQLException("Error at inserting row in " +
uri);
                    break;
                }
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            mContext.getContentResolver().notifyChange(uri, null);
            return returnUri;
        } catch (Exception ex) {
            Log.e("Insert", ex.toString());
            db.close();
        } finally {
            db.close();
        }
        return null;
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        final SQLiteDatabase db = DBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int deletedRows;
        if (null == selection) selection = "1";
        try {
            switch (match) {
                case USER:
                    deletedRows = db.delete(
                            DBContract.UserEntry.TABLE_NAME, selection, selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            if (deletedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return deletedRows;
        } catch (Exception ex) {
            Log.e("Insert", ex.toString());
        } finally {
            db.close();
        }
        return 0;
```

```
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int updatedRows;
        try {
            switch (match) {
                case USER:
                    updatedRows = db.update(DBContract.UserEntry.TABLE_NAME, values, selection,
selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            if (updatedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return updatedRows;
        } catch (Exception ex) {
            Log.e("Update", ex.toString());
        } finally {
            db.close();
        }
        return -1;
    }

}
```

Bu qanday qo'llaniladi:

```
public void InsertUser() {
    try {
        ContentValues userValues = getUserData("Jhon","XXXXX");
        DBProvider dbProvider = new DBProvider(mContext);
        dbProvider.insert(UserEntry.CONTENT_URI, userValues);

    } catch (Exception ex) {
        Log.e("Insert", ex.toString());
    }
}

public ContentValues getUserData(String name, String pass) {
    ContentValues userValues = new ContentValues();
    userValues.put(UserEntry.COLUMN_Name, name);
    userValues.put(UserEntry.COLUMN_Password, pass);
    return userValues;
}
```

Jadvaldan qator(lar)ni o'chirish

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

db.delete(TABLE_NAME, null, null);
db.close();
```

Jadvaldagi barcha qatorlarni o'chirish va o'chirilgan qatorning sonini qaytarish qiymatiga olish uchun

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

Qatorlarni o'chirish uchun **WHERE** shartini qo'llash

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

String whereClause = KEY_NAME + " = ?";
String[] whereArgs = new String[]{String.valueOf(KEY_VALUE)};

//for multiple condition, join them with AND
//String whereClause = KEY_NAME1 + " = ? AND " + KEY_NAME2 + " = ?";
//String[] whereArgs = new String[]{String.valueOf(KEY_VALUE1), String.valueOf(KEY_VALUE2)};

int numRowsDeleted = db.delete(TABLE_NAME, whereClause, whereArgs);
db.close();
```

Jadvaldagi qatorni o'chirish

```
// You need a writable database to update a row
final SQLiteDatabase database = openHelper.getWritableDatabase();

// Create a ContentValues instance which contains the up to date data for each column
// Unlike when inserting data you need to specify the value for the PRIMARY KEY column as well
final ContentValues values = new ContentValues();
values.put(COLUMN_ID, model.getId());
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// This call performs the update
// The return value tells you how many rows have been updated.
final int count = database.update(
        TABLE_NAME,          // The table name in which the data will be updated
        values,              // The ContentValues instance with the new data
        COLUMN_ID + " = ?", // The selection which specifies which row is updated. ? symbols are
parameters.
        new String[] {       // The actual parameters for the selection as a String[].
                String.valueOf(model.getId())
        }
);
```

Transaction ni ijro etish

Tranzaksiyalar yordamida ma'lumotlar bazasiga avtomatik ravishda bir nechta o'zgartirishlar kiritish mumkin. Har qanday normal transaction quyidagilarga amal:

```
// You need a writable database to perform transactions
final SQLiteDatabase database = openHelper.getWritableDatabase();

// This call starts a transaction
database.beginTransaction();
```

```
// Using try/finally is essential to reliably end transactions even
// if exceptions or other problems occur.
try {

    // Here you can make modifications to the database
    database.insert(TABLE_CARS, null, productValues);
    database.update(TABLE_BUILDINGS, buildingValues, COLUMN_ID + " = ?", new String[] {
String.valueOf(buildingId) });

    // This call marks a transaction as successful.
    // This causes the changes to be written to the database once the transaction ends.
    database.setTransactionSuccessful();
} finally {
    // This call ends a transaction.
    // If setTransactionSuccessful() has not been called then all changes
    // will be rolled back and the database will not be modified.
    database.endTransaction();
}
```

Faol operatsiyalar ichidagi beginTransaction () chaqiruvlari natijasi yo'q.

Assets papkasidan ma' lumotlar bazasini yarating

Dbname.sqlite yoki dbname.db faylingizni loyihangizning assets papkasiga qo'ying.

```
public class DatabaseHelper extends SQLiteOpenHelper {
        public static final String TAG = DatabaseHelper.class.getSimpleName();
        public static int flag;
        // Exact Name of you db file that you put in assets folder with extension.
        static String DB_NAME = "dbname.sqlite";
        private final Context myContext;
        String outFileName = "";
        private String DB_PATH;
        private SQLiteDatabase db;

        public DatabaseHelper(Context context) {
            super(context, DB_NAME, null, 1);
            this.myContext = context;
            ContextWrapper cw = new ContextWrapper(context);
            DB_PATH = cw.getFilesDir().getAbsolutePath() + "/databases/";
            Log.e(TAG, "DatabaseHelper: DB_PATH " + DB_PATH);
            outFileName = DB_PATH + DB_NAME;
            File file = new File(DB_PATH);
            Log.e(TAG, "DatabaseHelper: " + file.exists());
            if (!file.exists()) {
                file.mkdir();
            }
        }

        /**
         * Creates a empty database on the system and rewrites it with your own database.
         */
        public void createDataBase() throws IOException {
            boolean dbExist = checkDataBase();
            if (dbExist) {
                //do nothing - database already exist
            } else {
                //By calling this method and empty database will be created into the default system
path
                //of your application so we are gonna be able to overwrite that database with our
database.
```

```
            this.getReadableDatabase();
            try {
                copyDataBase();
            } catch (IOException e) {
                throw new Error("Error copying database");
            }
        }
    }

    /**
     * Check if the database already exist to avoid re-copying the file each time you open the
application.
     *
     * @return true if it exists, false if it doesn't
     */
    private boolean checkDataBase() {
        SQLiteDatabase checkDB = null;
        try {
            checkDB = SQLiteDatabase.openDatabase(outFileName, null,
SQLiteDatabase.OPEN_READWRITE);
        } catch (SQLiteException e) {
            try {
                copyDataBase();
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        }

        if (checkDB != null) {
            checkDB.close();
        }
        return checkDB != null ? true : false;
    }

    /**
     * Copies your database from your local assets-folder to the just created empty database in
the
     * system folder, from where it can be accessed and handled.
     * This is done by transfering bytestream.
     */

    private void copyDataBase() throws IOException {

        Log.i("Database",
                "New database is being copied to device!");
        byte[] buffer = new byte[1024];
        OutputStream myOutput = null;
        int length;
        // Open your local db as the input stream
        InputStream myInput = null;
        try {
            myInput = myContext.getAssets().open(DB_NAME);
            // transfer bytes from the inputfile to the
            // outputfile
            myOutput = new FileOutputStream(DB_PATH + DB_NAME);
            while ((length = myInput.read(buffer)) > 0) {
                myOutput.write(buffer, 0, length);
            }
            myOutput.close();
            myOutput.flush();
            myInput.close();
            Log.i("Database",
```

```
                "New database has been copied to device!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void openDataBase() throws SQLException {
        //Open the database
        String myPath = DB_PATH + DB_NAME;
        db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE);
        Log.e(TAG, "openDataBase: Open " + db.isOpen());
    }

    @Override
    public synchronized void close() {
        if (db != null)
            db.close();
        super.close();
    }

    public void onCreate(SQLiteDatabase arg0) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {

    }
}
```

Bu yerda qanday qilib o'zingizning activity dan baza ob'yektiga bog'lanishni ko'rishingiz mumkin.

```
// Create Databasehelper class object in your activity.
   private Databasehelper db;
```

Keyin onCreate Method-da uni ishga tushiring va CreateDatabase () usulini quyida ko'rsatilgandek chaqiring.

```
db = new Databasehelper(MainActivity.this);
        try {
            db.createDataBase();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Barcha qo'shimchalarni bajaring, yangilang, o'chiring va quyida ko'rsatilgandek operatsiyani tanlang.

```
String query = "select Max(Id) as Id from " + TABLE_NAME;
        db.openDataBase();
        int count = db.getId(query);
        db.close();
```

Rasmni SQLite-da saqlash

```
public class DatabaseHelper extends SQLiteOpenHelper {
    // Database Version
    private static final int DATABASE_VERSION = 1;


    // Database Name
    private static final String DATABASE_NAME = "database_name";

    // Table Names
    private static final String DB_TABLE = "table_image";

    // column names
    private static final String KEY_NAME = "image_name";
    private static final String KEY_IMAGE = "image_data";

    // Table create statement
    private static final String CREATE_TABLE_IMAGE = "CREATE TABLE " + DB_TABLE + "("+
                        KEY_NAME + " TEXT," +
                        KEY_IMAGE + " BLOB);";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        // creating table
        db.execSQL(CREATE_TABLE_IMAGE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // on upgrade drop older tables
        db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);

        // create new table
        onCreate(db);
    }
}
```

Database ga ma'lumot joylash:

```
public void addEntry( String name, byte[] image) throws SQLiteException{
    SQLiteDatabase database = this.getWritableDatabase();
    ContentValues cv = new  ContentValues();
    cv.put(KEY_NAME,     name);
    cv.put(KEY_IMAGE,     image);
    database.insert( DB_TABLE, null, cv );
}
```

Ma'lumotni qabul qilish:

```
byte[] image = cursor.getBlob(1);
```

Eslatma:

Ma'lumotlar bazasiga qo'shishdan oldin Bitmap rasmingizni bayt ko'rinishga aylantirishingiz kerak, keyin ma'lumotlar bazasi so'rovi yordamida amalga oshirasiz. ·

Ma'lumotlar bazasidan olayotganda, albatta, sizda rasmning baytlar majmuasi mavjud, baytlar qatorini asl rasmga qaytarish uchun nima qilish kerak? Shu savolga javob bersak. Shunday qilib, kodni hal qilish uchun BitmapFactory-dan foydalanishingiz kerak.

Quyidagi misolni ko'rib chiqsak:

```
public class DbBitmapUtility {

    // convert from bitmap to byte array
    public static byte[] getBytes(Bitmap bitmap) {
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        bitmap.compress(CompressFormat.PNG, 0, stream);
        return stream.toByteArray();
    }

    // convert from byte array to bitmap
    public static Bitmap getImage(byte[] image) {
        return BitmapFactory.decodeByteArray(image, 0, image.length);
    }
}
```

SQLite ma'lumotlar bazasiga qatorlarni kiritish va yangilash

Birinchidan, siz SQLite ma'lumotlar bazasini ochishingiz kerak, bu quyidagicha bajarilishi mumkin:

```
SQLiteDatabase myDataBase;
String mPath = dbhelper.DATABASE_PATH + dbhelper.DATABASE_NAME;
myDataBase = SQLiteDatabase.openDatabase(mPath, null, SQLiteDatabase.OPEN_READWRITE);
```

Ma'lumotlar bazasini ochgandan so'ng, ContentValues sinfidan foydalanib, qatorlarni osongina qo'shishingiz yoki yangilashingiz mumkin. Quyidagi misollarda ism str_edtfname va familiya str_edtlname tomonidan berilgan deb taxmin qilinadi. Shuningdek, table_name-ni o'zgartirmoqchi bo'lgan jadvalingiz nomi bilan almashtirishingiz kerak.

Ma'lumot qo'shish

```
ContentValues values = new ContentValues();
values.put("First_Name", str_edtfname);
values.put("Last_Name", str_edtlname);
myDataBase.insert("tablo_name", null, values);
```

Ma'lumotni o'zgaritirish

```
ContentValues values = new ContentValues();
values.put("First_Name", str_edtfname);
values.put("Last_Name", str_edtlname);
myDataBase.update("table_name", values, "id" + " = ?", new String[] {id});
```

Savol va topshiriqlar:

1. Ma'lumot bazasi haqida bilganlaringizni aytib bering.
2. Local database haqida nimlarni bilasiz?
3. Firebase sistemasi nima uchun kerak?
4. SQLite ishlash prinspi qanday?
5. Ma'lumotlarni qo'shish, o'zgartirish, o'chirish qanday amalga oshiriladi?
6. Kutubxonalarni qanday yuklash mumkin?
7. Foydalanuvchi id nomi qanday aniqlanadi (methods)?
8. Cursor bilan ishlash prinspini tushuntirib bering.
9. Ommaviy qo'shimchalar haqida nimalarni bilasiz?
10. SQLite uchun Helper va Provider yaratish ketma-ketligini qanday?

9-MAVZU: FOYDALANUVCHINING JOYLASHGAN O'RNINI ANIQLASH

Android qurilmalar bizga hozirgi joylashuvimizga asoslangan ma'lumotlarni taqdim etishi mumkin. Bu albatta juda qulay va masalan xaritadan foydalanish, mintaqangizga tegishli ma'lumotlarni olish (ob-havo prognozi), har qanday ro'yxatdan o'tish va hk. Bularning barchasini amalga oshirish juda oddiy. Biz tinglovchini provayderga joylab qo'yamiz va ma'lumotlarni olamiz. Ayni paytda ikkita provayder mavjud: GPS va Tarmoq.

GPS - bu erda hamma narsa aniq, bu GPS sun'iy yo'ldoshlaridan olingan ma'lumotlar.

Network - bu uyali yoki WiFi orqali olinadigan koordinatalar. Ushbu provayderga internet kerak.

Foydalanuvchi joylashuvini aniqlash bo'yicha amaliy misolda ko'rib o'tsak:

strings.xml faylini loyihamizga kiritamiz.

```
<string name="provider_gps">GPS</string>
<string name="provider_network">Network</string>
<string name="location_settings">Location settings</string>
```

main.xml faylida loyihamizning front end qismini yaratamiz.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="5dp">
   <TextView
      android:id="@+id/tvTitleGPS"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/provider_gps"
      android:textSize="30sp">
   </TextView>
   <TextView
      android:id="@+id/tvEnabledGPS"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp">
</TextView>
<TextView
    android:id="@+id/tvStatusGPS"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp">
</TextView>
<TextView
    android:id="@+id/tvLocationGPS"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp">
</TextView>
<TextView
    android:id="@+id/tvTitleNet"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:text="@string/provider_network"
    android:textSize="30sp">
</TextView>
<TextView
    android:id="@+id/tvEnabledNet"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp">
</TextView>
<TextView
    android:id="@+id/tvStatusNet"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp">
</TextView>
<TextView
    android:id="@+id/tvLocationNet"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp">
  </TextView>
  <Button
    android:id="@+id/btnLocationSettings"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:onClick="onClickLocationSettings"
    android:text="@string/location_settings">
  </Button>
</LinearLayout>
```

Bizga ma'lumotlar chiqaradigan bir nechta TextView va joylashuv sozlamalarini ochish tugmasi.

MainActivity.java:

```
package ru.startandroid.develop.p1381location;

import java.util.Date;
import android.app.Activity;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends Activity {

  TextView tvEnabledGPS;
  TextView tvStatusGPS;
  TextView tvLocationGPS;
  TextView tvEnabledNet;
  TextView tvStatusNet;
  TextView tvLocationNet;
```

```
 private LocationManager locationManager;
 StringBuilder sbGPS = new StringBuilder();
 StringBuilder sbNet = new StringBuilder();
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  tvEnabledGPS = (TextView) findViewById(R.id.tvEnabledGPS);
  tvStatusGPS = (TextView) findViewById(R.id.tvStatusGPS);
  tvLocationGPS = (TextView) findViewById(R.id.tvLocationGPS);
  tvEnabledNet = (TextView) findViewById(R.id.tvEnabledNet);
  tvStatusNet = (TextView) findViewById(R.id.tvStatusNet);
  tvLocationNet = (TextView) findViewById(R.id.tvLocationNet);
  locationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);
 }
 @Override
 protected void onResume() {
  super.onResume();
  locationManager.requestLocationUpdates(LocationManager.GPS_
PROVIDER,
      1000 * 10, 10, locationListener);
  locationManager.requestLocationUpdates(
     LocationManager.NETWORK_PROVIDER, 1000 * 10, 10,
     locationListener);
  checkEnabled();
 }
 @Override
 protected void onPause() {
  super.onPause();
  locationManager.removeUpdates(locationListener);
 }
 private LocationListener locationListener = new LocationListener()
{
  @Override
  public void onLocationChanged(Location location) {
   showLocation(location);
  }
```

```
  @Override
  public void onProviderDisabled(String provider) {
    checkEnabled();
  }
  @Override
  public void onProviderEnabled(String provider) {
    checkEnabled();
    showLocation(locationManager.getLastKnownLocation(provider)
);
  }
  @Override
  public void onStatusChanged(String provider, int status, Bundle
extras) {
    if (provider.equals(LocationManager.GPS_PROVIDER)) {
      tvStatusGPS.setText("Status: " + String.valueOf(status));
    } else if
(provider.equals(LocationManager.NETWORK_PROVIDER)) {
      tvStatusNet.setText("Status: " + String.valueOf(status));
    }
  }
 };
 private void showLocation(Location location) {
  if (location == null)
    return;
  if
(location.getProvider().equals(LocationManager.GPS_PROVIDER))
{
    tvLocationGPS.setText(formatLocation(location));
  } else if (location.getProvider().equals(
      LocationManager.NETWORK_PROVIDER)) {
    tvLocationNet.setText(formatLocation(location));
  }
 }
 private String formatLocation(Location location) {
  if (location == null)
    return "";
  return String.format(
```

```
        "Coordinates: lat = %1$.4f, lon = %2$.4f, time = %3$tF
%3$tT",
        location.getLatitude(), location.getLongitude(), new Date(
            location.getTime()));
    }
    private void checkEnabled() {
      tvEnabledGPS.setText("Enabled: "
          + locationManager
              .isProviderEnabled(LocationManager.GPS_PROVIDER));
      tvEnabledNet.setText("Enabled: "
          + locationManager
              .isProviderEnabled(LocationManager.NETWORK_PROVIDE
R));
    }
    public void onClickLocationSettings(View view) {
      startActivity(new Intent(
          android.provider.Settings.ACTION_LOCATION_SOURCE_SET
TINGS));
    };
}
```

OnCreate-da biz TextView komponentlarini aniqlaymiz va LocationManager-ni olamiz, u orqali ishlaymiz.
RequestLocationUpdates usuli yordamida tinglovchini **onResume**-ga qo'ying. Biz kirish joyiga jo'natamiz:
- provayder turi: GPS_PROVIDER yoki NETWORK_PROVIDER
- ma'lumotlarni qabul qilish o'rtasidagi minimal vaqt (millisekundlarda). Biz bu yerda 10 soniyani ko'rsatamiz, bu biz uchun yetarli.
- agar siz koordinatalarni kechiktirmasdan olishni istasangiz - 0 ni yuboring, ammo bu faqat minimal vaqt ekanligini unutmang. Haqiqiy kutish uzoqroq bo'lishi mumkin.
- minimal masofa (metrda). O'sha. agar sizning manzilingiz ko'rsatilgan metrga qarab o'zgargan bo'lsa, unda siz yangi koordinatalarni olasiz.
- tinglovchi, locationListener obyekti, bu quyida muhokama qilinadi.
Bundan tashqari, biz bu yerda provayderlarni ekranda kiritish to'g'risidagi ma'lumotlarni yangilaymiz.

OnPause-da, RemoveUpdates usuli yordamida tinglovchini o'chirib qo'ying.
locationListener - tinglovchi, LocationListener interfeysini quyidagi usullar bilan amalga oshiradi:
onLocationChanged - yangi joylashuv ma'lumotlari, joylashuv obyekti. Bu erda biz joylashuv ma'lumotlarini ekranda aks ettiradigan showLocation usulimizni chaqiramiz.
onProviderDisabled - ko'rsatilgan provayder foydalanuvchi tomonidan o'chirib qo'yilgan. Ushbu usulda biz provayderlarning ekrandagi hozirgi holatini yangilaydigan checkEnabled usulimizni chaqiramiz.
onProviderEnabled - ko'rsatilgan provayder foydalanuvchi tomonidan yoqilgan. Biz bu erda checkEnabled-ni ham chaqiramiz.
Keyin **getLastKnownLocation** usulidan foydalanib (u null qiymatini qaytarishi mumkin), biz mavjud provayderdan mavjud bo'lgan so'nggi manzilni so'raymiz va uni namoyish qilamiz. Agar ilgari joylashuvga asoslangan biron bir ilovadan foydalangan bo'lsangiz, bu juda dolzarb bo'lishi mumkin.
onStatusChanged - ko'rsatilgan provayderning holati o'zgardi. Holat maydonida OUT_OF_SERVICE (ma'lumotlar uzoq vaqt davomida mavjud bo'lmaydi), TEMPORARILY_UNAVAILABLE (ma'lumotlar vaqtincha mavjud emas), AVAILABLE (barchasi yaxshi, ma'lumotlar mavjud) qiymatlari bo'lishi mumkin.
Ushbu usulda biz shunchaki yangi holatni ekranda namoyish etamiz. Provayderlar tizim sozlamalarida yoqilgan va o'chirilgan. Shunday qilib, provayder undan koordinatalarni qabul qilishi mumkinligi aniqlanadi. Siz foydalanuvchini ushbu sozlamalarga qanday yuborishingiz mumkinligini birozdan keyin ko'rib chiqamiz. Provayderlarni standart usullar bilan yoqish / o'chirish uchun dasturiy ta'minot mavjud emas.
Tayyor metodlar bilan tanishib chiqsak:
showLocation manzilni kirish sifatida qabul qiladi, getProvider usuli yordamida provayderini aniqlaydi va koordinatalarni tegishli matn maydonida aks ettiradi.
formatLocation joylashuvni kirish sifatida qabul qiladi, undan ma'lumotlarni o'qiydi va undan satr formatlaydi. Buning uchun qanday ma'lumotlar kerak: getLatitude - kenglik, getLongitude - uzunlik, getTime - aniqlash vaqti.

checkEnabled provayderlarning isProviderEnabled usuli yordamida yoqilgan yoki o'chirilganligini aniqlaydi va ushbu ma'lumotlarni ekranda aks ettiradi.

OnClickLocationSettings usuli Joylashuv sozlamalari tugmachasini bosish orqali ishga tushiriladi va foydalanuvchi provayderni yoqishi yoki o'chirib qo'yishi uchun sozlamalarni ochadi. Buning uchun = ACTION_LOCATION_SOURCE_SETTINGS harakati bilan Niyatdan foydalaning.

Koordinatalarni aniqlash uchun ruxsatni ro'yxatdan o'tkazish manifestda qoladi - ACCESS_FINE_LOCATION, bu bizga ham Tarmoq, ham GPS-dan foydalanishga imkon beradi. ACCESS_COARSE_LOCATION uchun ruxsat ham mavjud, ammo u faqat tarmoq provayderiga kirish huquqini beradi.

Savol va topshiriqlar

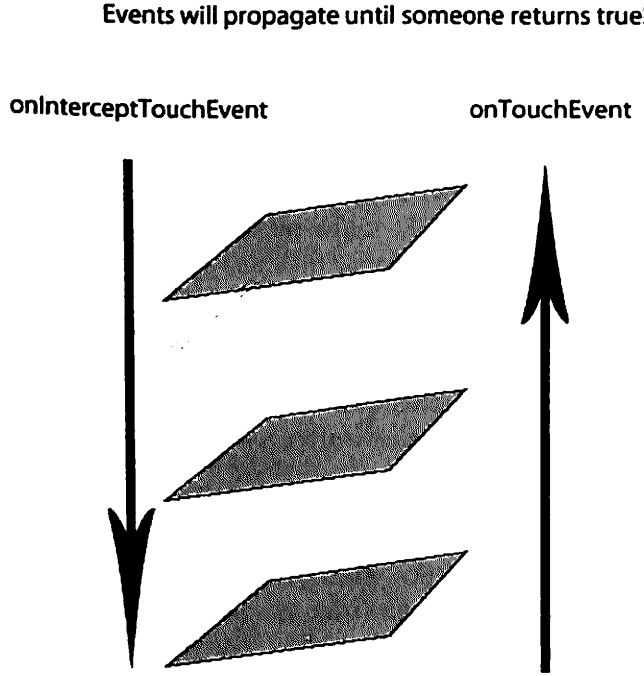
1. Foydalanuvchi joylashuvi deganda nimani topshirasiz?
2. Joriy locationni aniqlash uchun qo'llaniladigan funksiyalar?
3. formatLocation metodining vazifasi?
4. checkEnabled funksiyasining vazifasi?
5. Joylashuvni ko'rsatish qanday amalga oshiriladi?
6. Providerlar bilan ishlash haqida aytib bering.
7. GPS ni faollashtirish haqida.
8. onCreate va onResume metodlari ichida qo'llaniladigan funksiyalar?
9. Wifi orqali koordinatalarni aniqlang.
10. Joylashuv o'zgarishini hisoblagich yordamida ishlash qanday amalga oshiriladi?

10-MAVZU: ANDROID SENSOR IMKONIYATLARI

Ichki ko'rish guruhlari uchun onTouchEvents () mantiqiy onInterceptTouchEvent tomonidan boshqarilishi mumkin. Parentning onTouchEvent dasturi childdan oldin qabul qilinadi. Agar OnInterceptTouchEvent false qiymatini qaytarsa, u harakatlanish hodisasi zanjir bo'ylab childning OnTouchEvent ishlov beruvchisiga yuboradi. Agar u to'g'ri bo'lsa, parent bilan birgalika ishlaydi. Ammo ba'zi bir child elementlari OnTouchEvents-ni boshqarishini, ba'zilari esa shunday bo'lishini istagan holatlar bo'lishi mumkin.
Buni bir necha usul bilan boshqarish mumkin.

Child elementini parentning OnInterceptTouchEvent-dan himoya qilishning bir usuli bu 2-ni amalga oshirishdir. requestDisallowInterceptTouchEvent.
public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)

Agar elementda voqea bo'lsa, bu har qanday parent ko'rinishini ushbu element uchun OnTouchEvent-ni boshqarishiga yo'l qo'ymaydi. Agar OnInterceptTouchEvent noto'g'ri bo'lsa, asosiy element OnTouchEvent baholanadi. Agar sizda o'z metodingiz bo'lsa, har xil teginish hodisalarini boshqaradigan child elementlari ichida, yaroqsiz bo'lgan har qanday tegishli touch ishlovchilari qaytadi
OnTouchEvent.
Sensorli hodisalarni targ'ib qilish jarayoni qanday o'tishi haqida ingl.
parent -> child|parent -> child|parent -> child views.



10.1-rasm. Sensor hodisasi.

Yana bir usul - parent uchun OnInterceptTouchEvent-dan turli xil qiymatlarni qaytarish. Ushbu misol ViewGroup-da Touch hodisalarini boshqarish bo'yicha olingan va child qanday tutish kerakligini ko'rsatib beradi OnTouchEvent.

```
@Override
public boolean onInterceptTouchEvent(MotionEvent ev) {
    /*
     * This method JUST determines whether we want to intercept the motion.
     * If we return true, onTouchEvent will be called and we do the actual
     * scrolling there.
     */


    final int action = MotionEventCompat.getActionMasked(ev);

    // Always handle the case of the touch gesture being complete.
    if (action == MotionEvent.ACTION_CANCEL || action == MotionEvent.ACTION_UP) {
        // Release the scroll.
        mIsScrolling = false;
        return false; // Do not intercept touch event, let the child handle it
```

```
    }

    switch (action) {
        case MotionEvent.ACTION_MOVE: {
            if (mIsScrolling) {
                // We're currently scrolling, so yes, intercept the
                // touch event!
                return true;
            }

            // If the user has dragged her finger horizontally more than
            // the touch slop, start the scroll

            // left as an exercise for the reader
            final int xDiff = calculateDistanceX(ev);

            // Touch slop should be calculated using ViewConfiguration
            // constants.
            if (xDiff > mTouchSlop) {
                // Start scrolling!
                mIsScrolling = true;
                return true;
            }
            break;
        }
        ...
    }

    // In general, we don't want to intercept touch events. They should be
    // handled by the child view.
    return false;
}
```

Surface uchun voqea ishlovchilarini tegizish (masalan, SurfaceView, GLSurfaceView va boshqalar):

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.SurfaceView;
import android.view.View;

public class ExampleClass extends Activity implements View.OnTouchListener{
    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        CustomSurfaceView csv = new CustomSurfaceView(this);
        csv.setOnTouchListener(this);
        setContentView(csv);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        // Add a switch (see buttons example) if you handle multiple views
        // here you can see (using MotionEvent event) to see what touch event
        // is being taken. Is the pointer touching or lifted? Is it moving?
        return false;
    }
}
```

Yoki muqobil ravishda (suface):

```
public class CustomSurfaceView extends SurfaceView {
    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        super.onTouchEvent(ev);
        // Handle touch events here. When doing this, you do not need to call a listener.
        // Please note that this listener only applies to the surface it is placed in
        // (in this case, CustomSurfaceView), which means that anything else which is
        // pressed outside the SurfaceView is handled by the parts of your app that
        // have a listener in that area.
        return true;
    }
}
```

surface iccida Handling multitouchni qo'llash

```
public class CustomSurfaceView extends SurfaceView {
    @Override
    public boolean onTouchEvent(MotionEvent e) {
        super.onTouchEvent(e);
        if(e.getPointerCount() > 2){
            return false; // If we want to limit the amount of pointers, we return false
                // which disallows the pointer. It will not be reacted on either, for
                // any future touch events until it has been lifted and repressed.
        }

        // What can you do here? Check if the amount of pointers are [x] and take action,
        // if a pointer leaves, a new enters, or the [x] pointers are moved.
        // Some examples as to handling etc. touch/motion events.

        switch (MotionEventCompat.getActionMasked(e)) {
            case MotionEvent.ACTION_DOWN:
            case MotionEvent.ACTION_POINTER_DOWN:
                // One or more pointers touch the screen.
                break;
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_POINTER_UP:
                // One or more pointers stop touching the screen.
                break;
            case MotionEvent.ACTION_MOVE:
                // One or more pointers move.
                if(e.getPointerCount() == 2){
                    move();
                }else if(e.getPointerCount() == 1){
                    paint();
                }else{
                    zoom();
                }
                break;
        }
        return true; // Allow repeated action.
    }
}
```

Savol va topshiriqlar

1. Mobil qurilmalarda qanday sensor imkoniyatlari mavjud?
2. Sensor orientation qanday aniqlanadi?
3. Sensor yorug'lik darajasi anday amalga oshiriladi?
4. Sensorda canvas metodini qo'llash?
5. Vibratorni qo'llash qanday amalga oshiriladi?
6. Gripsensor nima?
7. Touch ning vazifasi?
8. Sensorda rang rejimlarni qo'llash?
9. MultiTouchni qo'llash qanday amalga oshiriladi?
10. MotionEvent ning vazifasi?

11-MAVZU: MOBIL DATCHIK TURLARI VA ULAR BILAN ISHLASH

Bugungi kunda mobil datchiklardan keng qo'llaniladi. Datchiklar nafaqat mobility soxasida balki ko'pgina elektron qurilmalarda keng qo'llanilmoqda. Bulardan eng ko'p qo'llaniladiganlaridan biri yuz bilan, barmoq bilan ishlovchi datchiklar. Quyidagi misollarimizda shu va shunga o'xshash datchiklar bilan qanday ishlash haqida bilib olamiz.

Ushbu misol yordamchi sinf print nger bosib chiqarish menejeri bilan o'zaro aloqada bo'lib, shifrlash va parolini echishni amalga oshiradi. Iltimos, ushbu misolda shifrlash uchun ishlatiladigan usul AES ekanligini unutmang. Shifrlashning yagona usuli bu emas va boshqa misollar mavjud. Ushbu misolda ma'lumotlar quyidagi tarzda shifrlangan va shifrdan chiqarilgan:

1. Foydalanuvchi yordamchiga kerakli shifrlanmagan parolni beradi.
2. Foydalanuvchidan nprprint taqdim etilishi talab qilinadi.
3. Autentifikatsiya qilingandan so'ng, yordamchi KeyStore-dan kalitni oladi va shifr yordamida parolni shifrlaydi.
4. Parol va IV tuzi (IV har bir shifrlash uchun yaratiladi va qayta ishlatilmaydi) keyinchalik parolni hal qilishda foydalaniladigan umumiy parametrlarga saqlanadi.

Parolni hal qilish:

1. Parolni parolini hal qilish uchun foydalanuvchi so'rovlari.
2. Foydalanuvchidan nprprint taqdim etilishi talab qilinadi.
3. Yordamchi IV yordamida shifr yaratadi va foydalanuvchi autentifikatsiya qilingandan so'ng, KeyStore KeyStore-dan kalitni oladi va parolni ochadi.

```
public class FingerPrintAuthHelper {

    private static final String FINGER_PRINT_HELPER = "FingerPrintAuthHelper";
    private static final String ENCRYPTED_PASS_SHARED_PREF_KEY = "ENCRYPTED_PASS_SHARED_PREF_KEY"
    private static final String LAST_USED_IV_SHARED_PREF_KEY = "LAST_USED_IV_SHARED_PREF_KEY";
    private static final String MY_APP_ALIAS = "MY_APP_ALIAS";

    private KeyguardManager keyguardManager;
    private FingerprintManager fingerprintManager;

    private final Context context;
    private KeyStore keyStore;
    private KeyGenerator keyGenerator;

    private String lastError;


    public interface Callback {
        void onSuccess(String savedPass);

        void onFailure(String message);

        void onHelp(int helpCode, String helpString);
    }

    public FingerPrintAuthHelper(Context context) {
        this.context = context;
    }

    public String getLastError() {
        return lastError;
    }



    @TargetApi(Build.VERSION_CODES.M)
    public boolean init() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
            setError("This Android version does not support fingerprint authentication");
            return false;
        }

        keyguardManager = (KeyguardManager) context.getSystemService(KEYGUARD_SERVICE);
        fingerprintManager = (FingerprintManager) context.getSystemService(FINGERPRINT_SERVICE);

        if (!keyguardManager.isKeyguardSecure()) {
            setError("User hasn't enabled Lock Screen");
            return false;
        }

        if (!hasPermission()) {
            setError("User hasn't granted permission to use Fingerprint");
            return false;
        }

        if (!fingerprintManager.hasEnrolledFingerprints()) {
            setError("User hasn't registered any fingerprints");
            return false;
        }

        if (!initKeyStore()) {
            return false;
        }
        return false;
    }
```

```
    @Nullable
    @RequiresApi(api = Build.VERSION_CODES.M)
    private Cipher createCipher(int mode) throws NoSuchPaddingException, NoSuchAlgorithmException,
UnrecoverableKeyException, KeyStoreException, InvalidKeyException,
InvalidAlgorithmParameterException {
        Cipher cipher = Cipher.getInstance(KeyProperties.KEY_ALGORITHM_AES + "/" +
                KeyProperties.BLOCK_MODE_CBC + "/" +
                KeyProperties.ENCRYPTION_PADDING_PKCS7);

        Key key = keyStore.getKey(MY_APP_ALIAS, null);
        if (key == null) {
            return null;
        }
        if(mode == Cipher.ENCRYPT_MODE) {
            cipher.init(mode, key);
            byte[] iv = cipher.getIV();
            saveIv(iv);
        } else {
            byte[] lastIv = getLastIv();
            cipher.init(mode, key, new IvParameterSpec(lastIv));
        }
        return cipher;
    }

    @NonNull
    @RequiresApi(api = Build.VERSION_CODES.M)
    private KeyGenParameterSpec createKeyGenParameterSpec() {
        return new KeyGenParameterSpec.Builder(MY_APP_ALIAS, KeyProperties.PURPOSE_ENCRYPT |
KeyProperties.PURPOSE_DECRYPT)
                        .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
                        .setUserAuthenticationRequired(true)
```

Android-da barmoq izlari skanerini qo'shish dastur

Android Android 6.0 (Marshmallow) SDK 23-dan ngerprint api-ni qo'llab-quvvatlaydi

Ushbu xususiyatdan ilovangizda foydalanish uchun birinchi bo'lib manifestingizga USE_FINGERPRINT ruxsatini qo'shing.

Manifiest.xml ga quyidagini kiritamiz.

<uses-permission
android:name="android.permission.USE_FINGERPRINT" />

Bu erda amal qilish tartibi

Dastlab Android Key Store-da KeyGenerator yordamida nosimmetrik kalit yaratishingiz kerak, u faqat foydalanuvchi fi ngerprint bilan autentifikatsiya qilingandan va KeyGenParameterSpec-dan o'tgandan keyin foydalanish mumkin.

```
KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");
keyPairGenerator.initialize(
        new KeyGenParameterSpec.Builder(KEY_NAME,
                KeyProperties.PURPOSE_SIGN)
                .setDigests(KeyProperties.DIGEST_SHA256)
                .setAlgorithmParameterSpec(new ECGenParameterSpec("secp256r1"))
                .setUserAuthenticationRequired(true)
                .build());
keyPairGenerator.generateKeyPair();
```

KeyGenParameterSpec.Builder.setUserAuthenticationRequired-ni rost deb belgilash orqali siz foydalanuvchiga uni autentifikatsiya qilgandan keyingina foydalanishga ruxsat berishingiz mumkin, shu jumladan foydalanuvchi fingerprint bilan tasdiqlanganda ham.

```
KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PublicKey publicKey =
        keyStore.getCertificate(MainActivity.KEY_NAME).getPublicKey();

KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PrivateKey key = (PrivateKey) keyStore.getKey(KEY_NAME, null);
```

FingerprintManager.authenticate-ga chaqiruvni amalga oshirib, barmoq izlari sensori ustida fingerprint bilan ishlashni boshlang. Yoki muqobil ravishda autentifikator sifatida server tomonidan tasdiqlangan parolga qaytishingiz mumkin.
FingerprintManger-ni fingerprintManger.class-dan yarating va ishga tushiring.
getContext().getSystemService(FingerprintManager.class)
authenticate uchun FingerprintManger api va yaratilgan ichki classni qo'llang.
FingerprintManager.AuthenticationCallback va override metodlar:

onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
Boshlash uchun fingerPrint hodisasini tinglash kripto yordamida autentifikatsiya qilish usulini chaqiradi
fingerprintManager
        .authenticate(cryptoObject, mCancellationSignal, 0 , this, null);

Bekor qilish:
android.os.CancellationSignal;
Barmoq izi (yoki parol) tekshirilgandan so'ng,
FingerprintManager.AuthenticationCallback
onAuthenticationSucceeded () qayta chaqiruv chaqiriladi.

```
@Override

public void onAuthenticationSucceeded(AuthenticationResult result) {

        }
```

Barmoq izi bilan dasturga kirish bo'yicha yana bir misol ko'rib o'tsak. Buning uchun avval biometric bilan ishlovchi kutubxonani dastur loyihasidagi **dependencies** bo'limiga qo'shish kerak bo'ladi.

implementation 'androidx.biometric:biometric:1.0.1'
implementation 'org.jetbrains:annotations:15.0'

Zarur kutubxonalar:

```
package com.example.fingerprint;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.biometric.BiometricManager;
import androidx.biometric.BiometricPrompt;
import androidx.core.content.ContextCompat;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;
import java.util.concurrent.Executor;
```

Loyihamizga kerakli kutubxonalarni **import** qilish shart emas. Dastur tuzish jarayonida avtomat import qilinadi ya'ni foydalangan class va metodlarga qarab.
Quyidagi dastur kodida barmoq izidan foydalanish, xatoligi, to'g'riligi, telefonda maxsus qurilma yo'qligi va boshqa holatlarni tekshirib beruvchi ko'rsatilgan.

```
public class MainActivity extends AppCompatActivity {
   TextView tv;
```

```
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     tv = findViewById(R.id.tv);
     BiometricManager biometricManager =
BiometricManager.from(this);
     switch (biometricManager.canAuthenticate()) {
        case
BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE:
           Toast.makeText(this, "unavialable",
Toast.LENGTH_SHORT).show();
           break;
           case
BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE:
              Toast.makeText(this, "Error",
Toast.LENGTH_SHORT).show();
              break;
              case
BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED:
                 Toast.makeText(this, "Enrolled",
Toast.LENGTH_SHORT).show();
                 break;
                 case BiometricManager.BIOMETRIC_SUCCESS:
                    Toast.makeText(this, "Success",
Toast.LENGTH_SHORT).show();
                    break;
     }
     Executor executor = ContextCompat.getMainExecutor(this);
     BiometricPrompt biometricPrompt = new
BiometricPrompt(MainActivity.this, executor, new
BiometricPrompt.AuthenticationCallback() {
        @Override
        public void onAuthenticationError(int errorCode, @NonNull
@org.jetbrains.annotations.NotNull CharSequence errString) {
           super.onAuthenticationError(errorCode, errString);
           tv.setText("Error");
        }
```

```
        @Override
        public void onAuthenticationSucceeded(@NonNull
@org.jetbrains.annotations.NotNull
BiometricPrompt.AuthenticationResult result) {
            super.onAuthenticationSucceeded(result);
            tv.setText("Success");
        }
        @Override
        public void onAuthenticationFailed() {
            super.onAuthenticationFailed();
            tv.setText("Failed");
        }
    });
    BiometricPrompt.PromptInfo promptInfo = new
BiometricPrompt.PromptInfo.Builder()
            .setTitle("Title")
            .setNegativeButtonText("NBText")
            .setSubtitle("SubTitle")
```

```
            .setDescription("Description")
        .build();
    biometricPrompt.authenticate(promptInfo);
  }
}
```

Savol va topshiriqlar

1. Mobil datchiklar deganda nimani tushinasiz?
2. Qanday mobil datchik turlari mavjud?
3. Mobil datchiklar bilan ishlovchi qanday funksiyalarni bilasiz?
4. BiometricPrompt class ning vazifasi?
5. Biometric ma'lumotlardan foydalanish uchun qanday kutubxonalardan foydalaniladi?
6. Barmoq izining ishlash algoritmi?
7. Import qilingan class lar va ularning vazifasi?
8. Yuzni aniqlash algoritmini tushintirib bering.
9. Ekranda barmoq izini aniqlashda qanday global class lardan foydalaniladi?
10. Datchiklar bilan ishlovchi maxsus mobil ilovalarga misol keltiring va ularning ishlash prinspini tushintiring.

12-MAVZU: GOOGLE SERVISLARIDAN FOYDALANISH (FIREBASE MISOLIDA)

Firebase - bu YC11 startapi sifatida boshlangan va Google Cloud Platform-da yangi avlod dasturlarini ishlab chiqish platformasi bo'lib yaratilgan BaaS(Backend-as-a-Service).

Firebase dasturchilar uchun katta web, mobil va desktop dasturlarning back end tomonidan juda katta yordam beradi. Siz Firebase bilan serverlarni tekshirishngiz kerak emas, API yozishning hojati yo'q, bu ishlarni Firebase ning o'zi sizga yozib beradi. Firebase - bu sizning serveringiz, sizning API va ma'lumotlar omboringiz. Firebase juda ham ko'p qulayliklarga ega BaaS platforma hisoblanadi. Undan juda ko'p dasturchilar foydalanishadi. Shaxsan o'zim ham!

- **Firebase** web sayti: https://firebase.google.com
- **Firebase** consuli: https://console.firebase.google.com

Firebasening asosiy imkoniyatlari

1. Real-time ma'lumotlar bazasi

Ko'pgina ma'lumotlar bazalari sizning ma'lumotlaringizni olish va sinxronlashtirish uchun HTTP so'rovlarini amalga oshirishni talab qiladi. Ko'pgina ma'lumotlar bazalari faqat siz so'raganda ma'lumot beradi. Firebase esa bunaqa emas! Firebase siz so'rasangiz ham, so'ramasangiz ham ma'lumotlar bazasida nima bo'lib turganini sizga yetkazib turadi. Agar siz Firebase ma'lumotlar bazasi bilan ishlasangiz, qolgan ma'lumotlar bazalaridan ko'ra Firebase tezroq ishlaydi.

2. Autentifikatsiya va Identifikatsiya

Firebase orqali siz o'z platformangiz, dasturingiz, mobil ilovangiz va hattoki o'yinlarda ham Autentifikatsiya va Identifikatsiya larni amalga oshirishingiz mumkin. Siz Firebase da email,telefon raqam, Google/Facebook/Twitter, Google play va boshqa ko'plab tizimlar orqali kirish va ro'yhatdan o'tishni yaratishingiz mumkin.

3. Hosting xizmati

Firebase-da barcha statik fayllaringiz uchun foydalanishga qulay xosting xizmati mavjud. U ularga HTTP / 2 bilan global CDN-dan xizmat qiladi.

Va sizning rivojlanishingizni ayniqsa og'riqsiz qilish uchun Firebase xostingi sizning barcha sinovlaringiz uchun mahalliy ravishda ishlaydigan Superstatic -dan foydalanadi .

4. Firebase Cross platforma

Firebase jamoasi bir qator yangi va mavjud Google mahsulotlarini Firebase bilan birlashtirdi. Ushbu funktsiyalar to'plami iOS va Android uchun amal qiladi, ammo web-saytlarga tegishli emas.

- Masofadan sozlash
- Sinov laboratoriyasi
- Bildirishnomalar
- Dinamik havolalar
- AdMob

Firebase-ning ijobiy va salbiy tomonlari

Ijobiy tamonlari

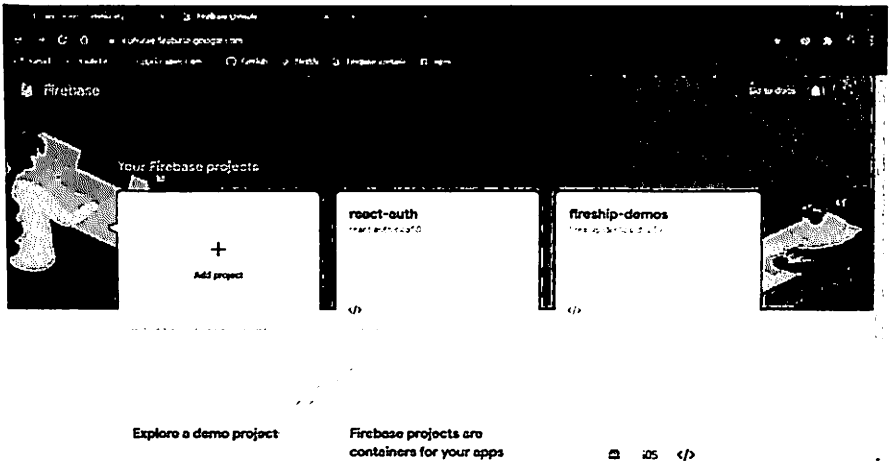
- Elektron pochta va parol, Google, Facebook va Github autentifikatsiyasi
- Real-time ma'lumot almashinish
- Tayyor API
- Kuchli xavfsizlik
- Google Cloud Storage tomonidan qo'llab-quvvatlanadigan fayllarni saqlash
- Statik fayllarni joylashtirish

Kamchiliklari va salbiy tomonlari

- Firebase-ning ma'lumotlar oqimi modeli tufayli cheklangan so'rov yuborish qobiliyatlari
- Ma'lumotlarning an'anaviy relyatsion modellari NoSQL uchun qo'llanilmaydi
- Mahalliy o'rnatish yo'q

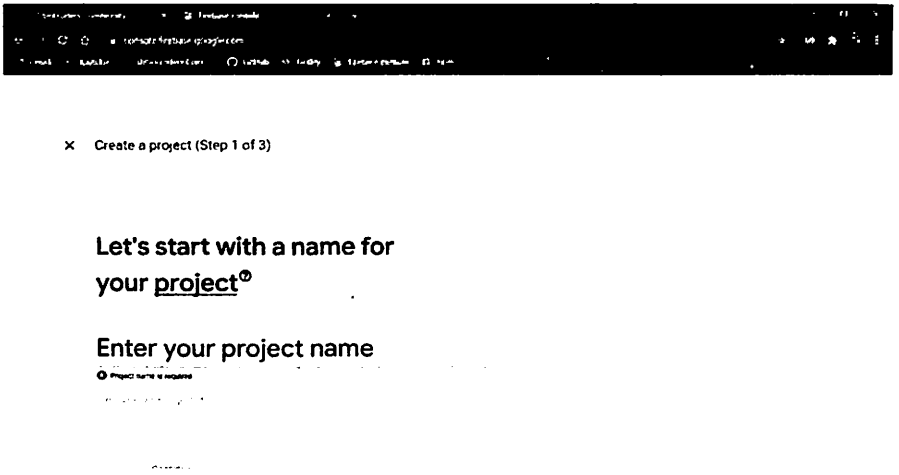
2. **Firebase dan qanday foydalaniladi?**

Avvalo siz Firebase ning consuli https://console.firebase.google.com saytiga o'tishingiz zarur:



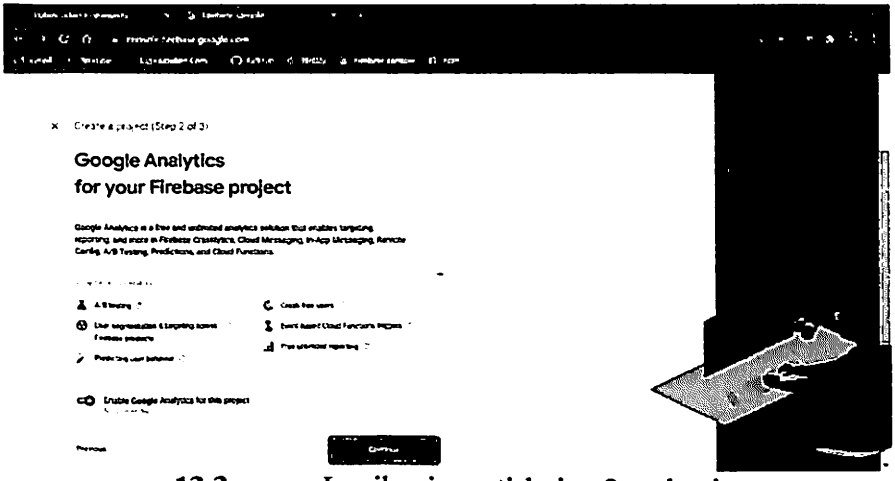
13.1-rasm. Firebase bosh oynasi.

Saytga o'tganingizdan keyin sizda "react-auth", "fireship-demos" kabi loyihalar bo'lmaydi.
Keyin "+Add project" tugmasini bosing:



13.2-rasm. Loyiha qo'shish.

Va sizda quyidagicha oyna hosil bo'ladi shu yerga siz yaratmoqchi bo'lgan yangi loyihangizni nomini kiritasiz.



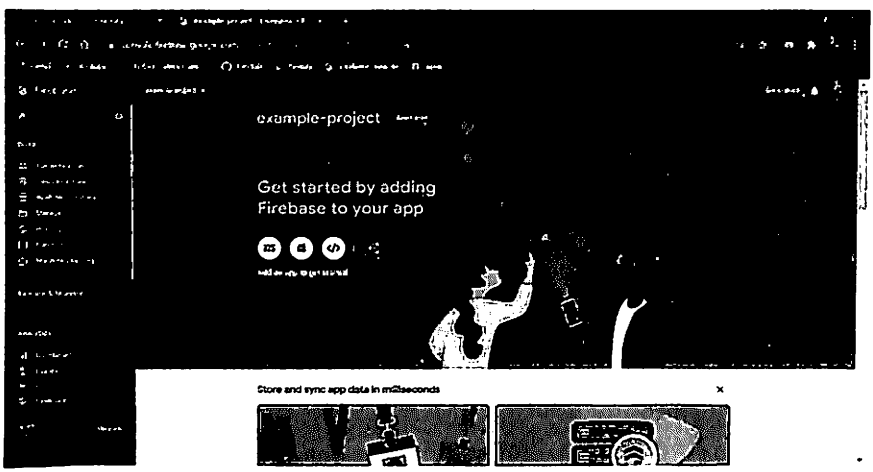
13.3-rasm. Loyihani yaratishning 2-qadami.

Keyin nomni yozib bo'lib **"Continue"** tugmasini bosganingizdan keyin sizdan loyihangiz *"Google Analytics da analiz qilaymi va ko'proq imkoniyatlarni ochib beraymi"* degandek so'raydi va siz bunga shu yerdagi hech narsaga tegmasdan **"Continue"** tugmasini bosishingiz kerak.

Keyingi hosil bo'lgan oynada siz Firebase hisobingiz tanlaysiz. Eslatma: Agar siz Chrome brauzeridan Google hisobingiz bilan kirgan bo'lsangiz, Firebase shu hisobingizni avtomatik olib ketadi. Boshqa brauzerlarda Firebase ga yangi hisob yaratishingiz zarur.

13.4-rasm. Loyiha yaratilish jarayoni.

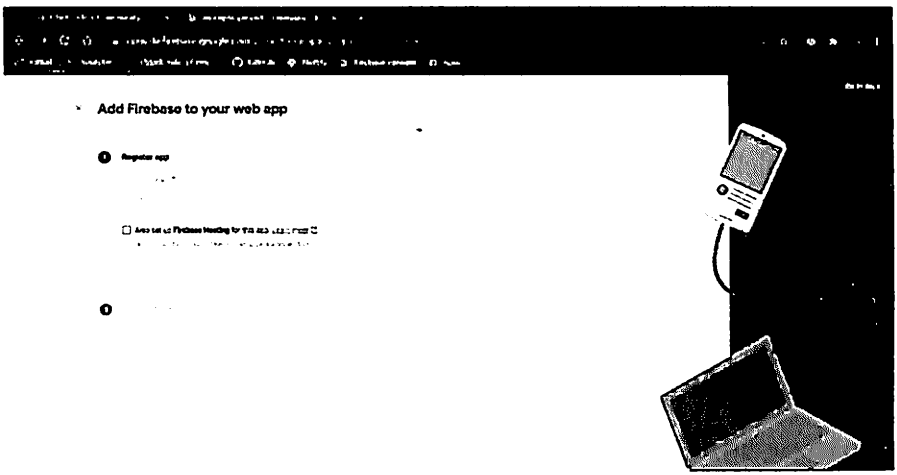
Va nihoyat sizning loyihangiz yaratilishni boshlaydi. Keyin sizda "Continue" tugmasi chiqadi, shu tugmani bosib quyidagi dashboard ga o'tasiz:



13.5-rasm. Loyiha bosh oynasi.

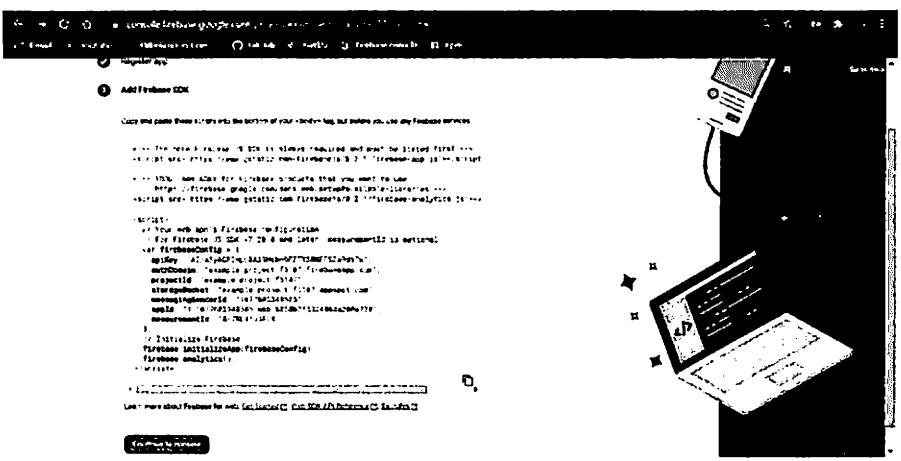
Shu yerdan siz o'zingizga kerak bo'lgan web-sayt,ilova,o'yin uchun bitta **web-ilova** yaratasiz.
Keling hozir biz bitta web-sayt uchun **web-ilova** yaratsak:

Buning uchun tepadagi rasmdagi **IOS,Android** ikonkalaridan keyingi </> ikonkasini bosamiz va bizda quyidagi oyna ochiladi:



13.6-rasm. Firebase loyihani web yoki mobil loyihaga qo'shish oynasi.

Bunda ham nomni kiritib "Register app" tugmasini bosamiz:



13.7-rasm. Loyihani bog'lash dastur kodi.

Sizga Firebase loyihangiz uchun **SDK** larni beradi, siz shu **SDK** lar orqali o'z loyihangizni ma'umotlar bazasiga ulaysiz. Huddi phpMyAdmin ni "localhost,username,password" qilib ulagandek.

Bizga asosiy kerak bo'ladigan kodlar manabu:

```
var firebaseConfig = {
  apiKey: "AIzaSyAGBIHpL3A39H6bn9PZTY3WNFFSZa9d67w",
  authDomain: "example-project-f5107.firebaseapp.com",
  projectId: "example-project-f5107",
  storageBucket: "example-project-f5107.appspot.com",
  messagingSenderId: "1077603340585",
  appId: "1:1077603340585:web:6850b2f532486da200e738",
  measurementId: "G-7ML41J3FER"
 };
```

Keyin **"Continue to Console"** tugmasini bosamiz :
Shu tariqa firebase tizimi bilan ishashni davom qilamiz.

Firebase foydalanuvchilarining elektron pochtasini yangilash

```
public class ChangeEmailActivity extends BaseAppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {

    @BindView(R.id.et_change_email)
    EditText mEditText;
    private FirebaseUser mFirebaseUser;

    @OnClick(R.id.btn_change_email)
    void onChangeEmailClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter valid email");
            return;
        }

        changeEmail(mEditText.getText().toString());
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mFirebaseAuth.getCurrentUser();
    }
```

```
    private void changeEmail(String email) {
        DialogUtils.showProgressDialog(this, "Changing Email", "Please wait...", false);
        mFirebaseUser.updateEmail(email)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            showToast("Email updated successfully.");
                            return;
                        }

                        if (task.getException() instanceof
FirebaseAuthRecentLoginRequiredException) {
                            FragmentManager fm = getSupportFragmentManager();
                            ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
ReAuthenticateDialogFragment();
                            reAuthenticateDialogFragment.show(fm,
reAuthenticateDialogFragment.getClass().getSimpleName());
                        }
                    }
                });
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_email;
    }

    @Override
    public void onReauthenticateSuccess() {
        changeEmail(mEditText.getText().toString());
    }
}
```

Savol va topshiriqlar

1. Googleda qanday mobile servislar mavjud?
2. Firbase sistemasi haqida aytib bering.
3. Realtime database ning vazifasi?
4. Firebase bilan ishlovchi maxsus funksiyalar?
5. Loyihani firebase ga ulash algoritmi?
6. Firebase tizimida json xizmatlari?
7. Bazaga media fayllarni yuklash algoritmi?
8. Bazadan ma'lumotlarni o'qib oluvchi funksiyalar?
9. Bazadagi ma'lumotlarni ekranga chiqarishda qo'llaniladigan komponentalar.
10. Loyihani Play Market ga joylashtirish algoritmi?

13-MAVZU: CROSS PLATFORMALAR

Umuman olganda, mobil ilovani ishlab chiqish murakkab va qiyin vazifadir. Mobil ilovani ishlab chiqish uchun ko'plab ramkalar mavjud. Android Java tiliga asoslangan mahalliy tizimni va iOS obyektiv-C / Shift tiliga asoslangan mahalliy ramkani taqdim etadi. Biroq, ikkala OSni ham qo'llab-quvvatlaydigan dasturni ishlab chiqish uchun biz ikki xil ramkalardan foydalangan holda ikki xil tilda kodlashimiz kerak. Ushbu murakkablikni yengishga yordam berish uchun ikkala operatsion tizimni qo'llab-quvvatlovchi mobil ramkalar mavjud. Ushbu ramkalar oddiydan farq qiladi. HTMLga asoslangan gibrid mobil dastur doirasi (foydalanuvchi interfeysi uchun HTML va dastur mantig'i uchun JavaScript-ni ishlatadigan) murakkab tilga xos tizimga (kodni mahalliy kodga aylantirishni og'irlashtiradigan). Oddiyligi yoki murakkabligidan qat'i nazar, ushbu ramkalar har doim juda ko'p kamchiliklarga ega, ularning asosiy kamchiliklaridan biri bu ularning sust ishlashi.

Ushbu senariyda Flutter - Dart tiliga asoslangan sodda va yuqori ishlash doirasi, interfeysni mahalliy ramka orqali emas, balki to'g'ridan-to'g'ri operatsion tizimning tuvalasida ko'rsatish orqali yuqori ishlashni ta'minlaydi.

Flutter shuningdek, zamonaviy dastur yaratish uchun ko'plab vidjetlarni (UI) ishlatishga tayyor. Ushbu vidjetlar mobil muhit uchun optimallashtirilgan va vidjetlar yordamida dasturni loyihalashtirish HTMLni yaratish kabi oddiy.

Xususan, Flutter dasturining o'zi vidjetdir. Flutter vidjetlari animatsiya va imo-ishoralarni ham qo'llab-quvvatlaydi. Ilova mantig'i reaktiv dasturlashga asoslangan. Vidjet ixtiyoriy ravishda holatga ega bo'lishi mumkin. Vidjet holatini o'zgartirib, Flutter vidjet holatini (eski va yangi) taqqoslaydi va vidjetni to'liq qayta ishlash o'rniga faqat kerakli o'zgarishlar bilan ishlaydi (reaktiv dasturlash).

Flutter ramkasi ishlab chiquvchilarga quyidagi xususiyatlarni taqdim etadi:

- zamonaviy va reaktiv asos.
- Dart dasturlash tilidan foydalanadi va uni o'rganish juda oson.
- Tez rivojlanish.
- Chiroyli va suyuq foydalanuvchi interfeyslari.
- ulkan vidjet katalogi.

- Bir nechta platformalar uchun bir xil interfeys ishlaydi.
- Yuqori mahsuldorlikka ega dastur.

Flutter yuqori ishlashi va ajoyib mobil ilovasi uchun chiroyli va moslashtirilgan vidjetlar bilan ta'minlangan. Bu barcha odatiy ehtiyojlar va talablarni qondiradi. Bundan tashqari, Flutter quyida aytib o'tilganidek ko'proq afzalliklarga ega:

- Dart-da dasturiy ta'minot paketlarining katta ombori mavjud, bu sizga ilova imkoniyatlari kengaytirishga imkon beradi.
- Ishlab chiquvchilar ikkala dastur uchun (Android va iOS platformalarida) faqat bitta kod bazasini yozishlari kerak. Flutter kelajakda boshqa platformaga ham tarqalishi mumkin.
- Flutter kamroq sinovga muhtoj. Yagona kod bazasi tufayli ikkala platforma uchun ham bir marta avtomatlashtirilgan testlar yozsak kifoya.
- Flutterning soddaligi uni tez rivojlanish uchun yaxshi nomzodga aylantiradi. Uning sozlash qobiliyati va kengaytirilishi uni yanada kuchliroq qiladi.
- Flutter yordamida ishlab chiquvchilar vidjetlar va uning joylashuvi ustidan to'liq nazoratga ega.
- Flutter ajoyib ishlab chiquvchi vositalarni taklif etadi, bu ajoyib qayta yuklash bilan ajralib turadi.

Ko'plab afzalliklarga qaramay, quyidagi kamchiliklari ham mavjud:

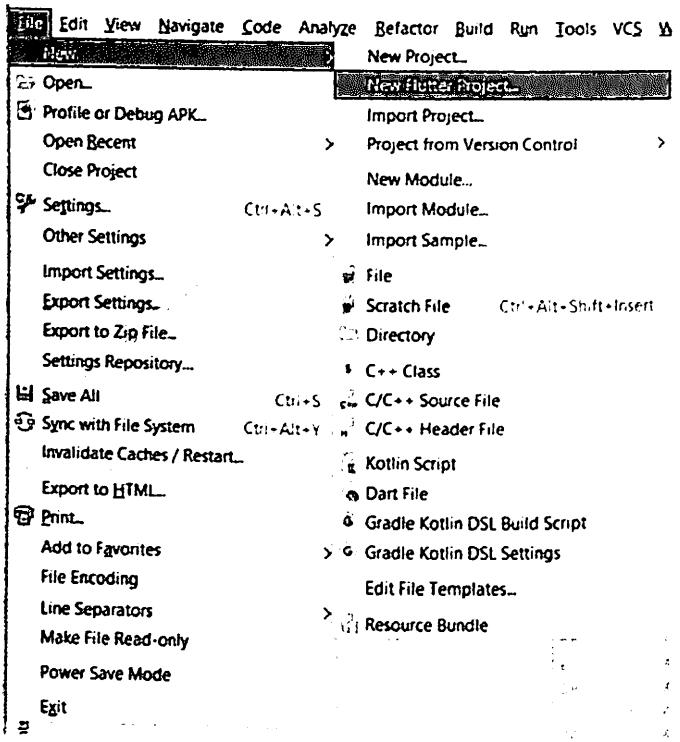
- Dart tilida kodlanganligi sababli, dasturchi yangi tilni o'rganishi kerak (garchi uni o'rganish oson bo'lsa ham).
- Zamonaviy ramka mantiq va interfeyslarni iloji boricha ajratishga harakat qilmoqda, ammo Flutter-da foydalanuvchi interfeysi va mantiq aralashgan. Biz buni aqlli kodlash va foydalanuvchi interfeysi va mantiqni ajratish uchun yuqori darajadagi modul yordamida engishimiz mumkin.
- Flutter - bu mobil dastur yaratishning yana bir asosidir. Dasturchilar juda ko'p sonli segmentda to'g'ri rivojlanish vositalarini tanlashda qiynalmoqda.

Flutter - android studioda oddiy dastur yaratish

Ushbu darsimizda Android Studio-da flutter dasturini yaratish asoslarini tushunish uchun oddiy Flutter dasturini yaratishni ko'rib chiqamiz.

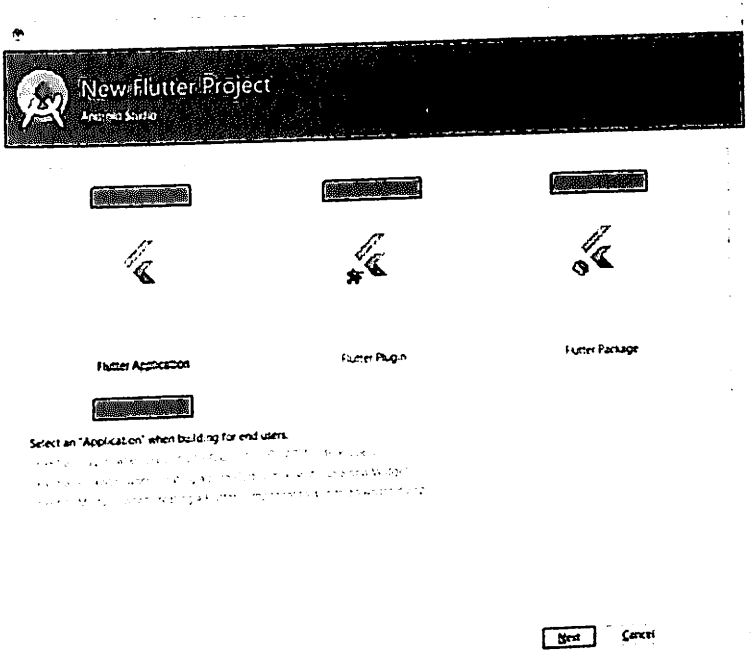
1-qadam: Android Studio dasturini oching

2-qadam: Flutter loyihasini yarating. Buning uchun k File -> New -> New Flutter Project ni bosing.



13.1-rasm. Yangi flutter loyihasini yaratish.

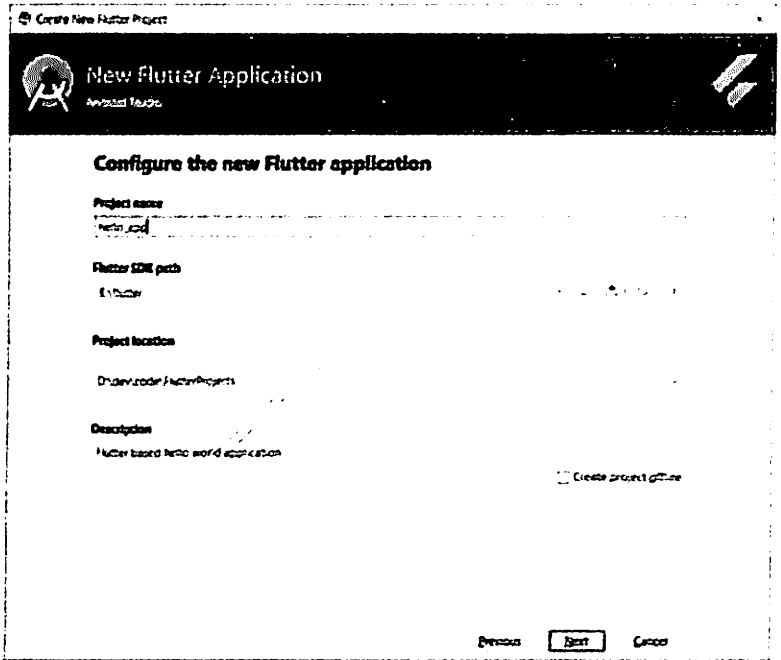
3-qadam: Flutter Application-ni tanlang. Buning uchun Flutter Application-ni tanlang va Keyingiga bosing.



13.2-rasm. Layoutni tanlash.

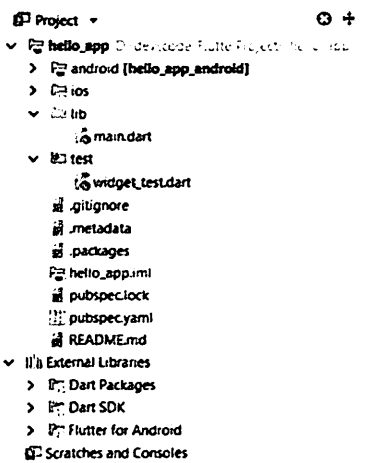
4-qadam: Ilovani quyidagi kabi sozlang va Keyingiga bosing.
- Loyiha nomi: salom_app
- Flutter SDK yo'li: <path_to_flutter_sdk>
- Loyiha joylashuvi: <yo'l_to_project_folder>
Tavsif: Flutter asosidagi salom dunyosi dasturi (13.2-rasm)

5-qadam: Loyihani sozlash.
Kompaniya domenini flutterapp.tutorialspoint.com qilib o'rnating va Finish tugmachasini bosing.
6-qadam: Kompaniya domenini kiriting.
Android Studio minimal funktsiyaga ega to'liq ishlaydigan flutter dasturini yaratadi. Keling, dastur tuzilishini tekshirib chiqamiz, keyin bizning vazifamizni bajarish uchun kodni o'zgartiring.



13.3-rasm. Loyiha nomini kiritish.

Ilova tarkibi va uning maqsadi quyidagicha:

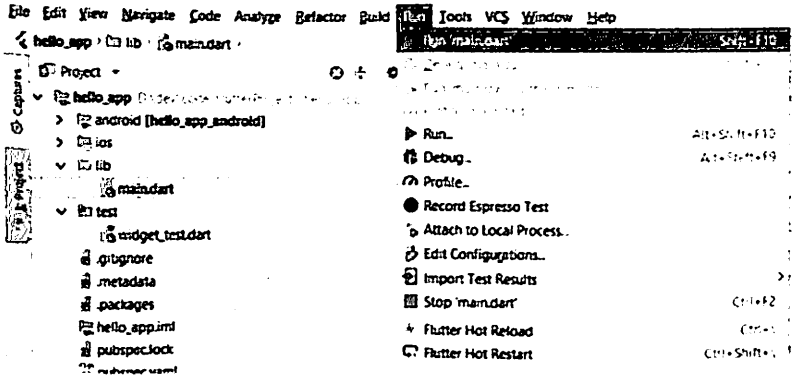


13.4-rasm. Loyiha strukturasi.

7-qadam: lib / main.dart faylidagi dart kodini quyidagi kod bilan almashtiring:

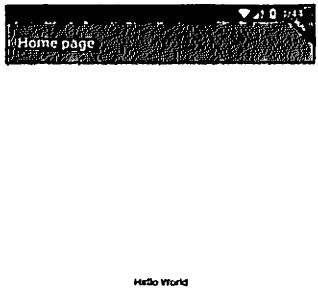
```
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Hello World Demo Application',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Home page'),
    );
  }
}
class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(this.title),
      ),
      body: Center(
        child:
            Text(
              'Hello World',
            )
      ),
    );
  }
}
```

8-qadam: Endi quyidagicha dasturni ishga tushiring, Run -> Run main.dart



13.5-rasm. Loyihani ishga tushirish.

9-qadam: Nihoyat, dasturning natijasi quyidagicha:



13.6-rasm. Loyiha natijasi

Savol va topshiriqlar

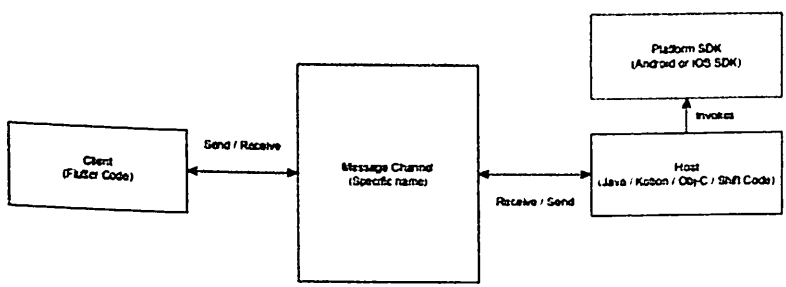
1. Cross platformasining vazifasi?
2. Flutter tilining imkoniyatlari?
3. Flutter tilining sintaksisi?
4. O‘zgaruvchilar va ma’lumotlar tiplari?
5. Asosiy operatorlar?
6. Flutter tilining json bilan ishlash imkoniyatlari?
7. Sinflar va constructorlar yaratish ketma-ketligini aytib bering.
8. Metodlar va maxsus funksiyalardan foydalanish.
9. Ma’lumotlar bazasi bilan ishlash imkoniyatlari?
10. Flutterda loyiha rusurslaridan foydalanish.

14-MAVZU: CROSS PLATFORMALARIDAN FOYDALANIB IOS UCHUN ILOVALAR YOZISH

Flutter platformaning o'ziga xos xususiyatlariga kirish uchun umumiy asosni taqdim etadi. Bu ishlab chiquvchiga Flutter ramkasining funksiyasini platformaga xos kod yordamida kengaytirish imkoniyatini beradi. Kamera, batareya quvvati darajasi, brauzer va boshqalar kabi platformaning o'ziga xos funksiyalariga ramka orqali osongina kirish mumkin.

Platformaning o'ziga xos kodiga kirishning umumiy g'oyasi oddiy xabar almashish protokoli orqali amalga oshiriladi. Flutter kodi, mijoz va platforma kodi va Xost umumiy xabarga ulanadi Kanal. Mijoz Xabar kanaliga Xost orqali xabar yuboradi. Xost Xabarlar kanalini tinglaydi, xabarni qabul qiladi va kerakli funktsiyalarni bajaradi va nihoyat, Xabar kanali orqali mijozga natijani qaytaradi.

Platformaga xos kod arxitekturasi quyida keltirilgan blok diagrammada ko'rsatilgan:



14.1-rasm. Blok diogramma.

Xabarlar protokoli raqamlar, satrlar, mantiqiy va boshqalar kabi JSON-ga o'xshash qiymatlarni ikkilik ketma-ketlashtirishni qo'llab-quvvatlaydigan standart xabar kodekidan (StandardMessageCodec class) foydalanadi, ketma-ketlashtirish, mijoz va mijoz o'rtasida shaffof ishlaydi.

Android SDK-dan foydalangan holda brauzerni ochish va flutter dasturidan SDK-ni qanday chaqirishni tushuntirish uchun oddiy dastur yozamiz.

- Android studiyasida flutter_browser_app yangi Flutter ilovasini yarating

- Main.dart kodini quyidagi kod bilan almashtiring:

```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(this.title),
      ),
      body: Center(
        child: RaisedButton(
          child: Text('Open Browser'),
          onPressed: null,
        ),
      ),
    );
  }
}
```

- Bu erda biz brauzerni ochish va uning onPressed usulini null qilib o'rnatish uchun yangi tugma yaratdik.
- Endi quyidagi paketlarni import qiling:

```
 import 'dart:async';
import 'package:flutter/services.dart';
```

- Bu yerda services.dart platformaga xos kodni chaqirish funktsiyasini o'z ichiga oladi.
- MyHomePage vidjetida yangi xabar kanalini yarating.

```
 static const platform = const
MethodChannel('flutterapp.tutorialspoint.com/browser');
```

Metodni yozing, platformaga xos usulni chaqirish uchun _openBrowser, xabar kanali orqali openBrowser usulini qo'llang.

```
Future<void> _openBrowser() async {
    try {
      final int result = await platform.invokeMethod('openBrowser',
<String, String>{
        'url': "https://flutter.dev"
      });
    } on PlatformException catch (e) {
      // Unable to open the browser
      print(e);
    }
  }
```

OpenBrowser-ni chaqirish uchun platform.invokeMethod-dan foydalandik (keyingi bosqichlarda tushuntiriladi). openBrowser-ning argumenti bor, ma'lum bir urlni ochish uchun url.
- RaisedButton-ning onPressed xususiyatining qiymatini null-dan _openBrowser-ga o'zgartiring.

```
onPressed: _openBrowser,
```

MainActivity.java-ni oching (android papkasida) va kerakli kutubxonani import qiling:

```
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;

import io.flutter.app.FlutterActivity;
import io.flutter.plugin.common.MethodCall;
import io.flutter.plugin.common.MethodChannel;
import io.flutter.plugin.common.MethodChannel.MethodCallHandler;
import io.flutter.plugin.common.MethodChannel.Result;
import io.flutter.plugins.GeneratedPluginRegistrant;
```

Brauzerni ochish uchun openBrowser usulini yozing

```
private void openBrowser(MethodCall call, Result result, String url) {
    Activity activity = this;
    if (activity == null) {
      result.error("ACTIVITY_NOT_AVAILABLE", "Browser cannot be opened
without foreground activity", null);
      return;
    }

    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse(url));

    activity.startActivity(intent);
    result.success((Object) true);
}
```

Endi MainActivity sinfida kanal nomini o'rnating:

```
private static final String CHANNEL =
"flutterapp.tutorialspoint.com/browser";
```

OnCreate usulida xabarlar bilan ishlashni o'rnatish uchun androidga xos kodni yozing.

```
new MethodChannel(getFlutterView(), CHANNEL).setMethodCallHandler(
    new MethodCallHandler() {
      @Override
      public void onMethodCall(MethodCall call, Result result) {

        String url = call.argument("url");

        if (call.method.equals("openBrowser")) {
```

```
          openBrowser(call, result, url);
        } else {
          result.notImplemented();
        }

      }
});
```

Bu yerda biz MethodChannel sinfidan foydalanib xabar kanalini yaratdik va xabarni boshqarish uchun MethodCallHandler sinfidan foydalandik.
onMethodCall - bu xabarni tekshirish orqali to'g'ri platforma kodini chaqirish uchun mas'ul bo'lgan haqiqiy usul.
onMethodCall usuli urlni xabardan ajratib oladi va keyin faqat chaqiruv penBrowser bo'lganida openBrowser-ni chaqiradi. Aks holda, u notImplemented usulini qaytaradi.
Ilovaning to'liq manba kodi quyidagicha:
main.dart
MainActivity.java

```
package com.tutorialspoint.flutterapp.flutter_browser_app;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import io.flutter.app.FlutterActivity;
import io.flutter.plugin.common.MethodCall;
import io.flutter.plugin.common.MethodChannel.Result;
import io.flutter.plugins.GeneratedPluginRegistrant;

public class MainActivity extends FlutterActivity {
  private static final String CHANNEL =
"flutterapp.tutorialspoint.com/browser";

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    GeneratedPluginRegistrant.registerWith(this);
```

```
  new                                MethodChannel(getFlutterView(),
CHANNEL).setMethodCallHandler(
        new MethodCallHandler() {
          @Override
          public void onMethodCall(MethodCall call, Result result) {

            String url = call.argument("url");

            if (call.method.equals("openBrowser")) {
              openBrowser(call, result, url);
            } else {
              result.notImplemented();
            }

          }
      });
  }

  private void openBrowser(MethodCall call, Result result, String url)
{
    Activity activity = this;
    if (activity == null) {
      result.error("ACTIVITY_NOT_AVAILABLE", "Browser cannot be
opened
without foreground activity", null);
      return;
    }

    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse(url));

    activity.startActivity(intent);
    result.success((Object) true);
  }
}
main.dart
import 'package:flutter/material.dart';
```

```
import 'dart:async';
import 'package:flutter/services.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

  static const platform = const
MethodChannel('flutterapp.tutorialspoint.com/browser');
  Future<void> _openBrowser() async {
    try {
      final int result = await platform.invokeMethod('openBrowser',
<String, String>{
        'url': "https://flutter.dev"
      });
} on PlatformException catch (e) {
      // Unable to open the browser
      print(e);
    }
  }
```

```
@override
Widget build(BuildContext context) {
 return Scaffold(
   appBar: AppBar(
    title: Text(this.title),
   ),
   body: Center(
    child: RaisedButton(
     child: Text('Open Browser'),
     onPressed: _openBrowser,
    ),
   ),
  );
 }
}
```

Ilovani ishga tushiring va Brauzerni ochish tugmachasini bosing va brauzer ishga tushirilganligini ko'rishingiz mumkin. Brauzer dasturi - Uy sahifasi bu erda skrinshotda ko'rsatilgandek:

Open Browser

14.2-rasm. Natija oynasi.

Savol va topshiriqlar

1. IOS uchun ilovalar ishlab chiqishda Cross platformadan foydalanish?
2. IOS uchun maxsus platformalar haqida aytib bering.
3. Swift tilida ios uchun ilova yaratish algoritmi?
4. IOS uchun qo'llaniladigan kutubxonalar?
5. Maxsus metod va class lardan foydalanish?
6. Ilovalarning ekranga moslashuvchanligini ta'minlash.
7. Ilovada maxsus komponentalardan foydalanish.
8. Bugungi kunda ios uchun maxsus yangi texnologiyalarga misollar?
9. Cross platformalarning ios uchun ilova ishlab chiqishdagi o'rni?
10. Flutterda ios uchun ilova yozish.

15-MAVZU: CROSS PLATFORMALARIDAN FOYDALANIB IOS UCHUN ILOVALAR YOZISH

IOS-ning o'ziga xos kodlariga kirish Android platformasidagi kodga o'xshaydi, faqat iOS-da o'ziga xos tillarni ishlatadi - Objective-C yoki Swift va iOS SDK. Aks holda, kontseptsiya Android platformasi bilan bir xil. Keling, iOS platformasi uchun dastur yozish algoritmini ko'rib chiqamiz:

- Android Studio (macOS) da flutter_browser_ios_app yangi dasturini yaratamiz
- Oldingi bobdagi kabi 2 - 6 bosqichlarni bajaring.
- XCode ni ishga tushiring va File -> Open ga bosing
- Bizning flutter loyihamizning iOS katalogidagi xcode loyihasini tanlang.
- Runner -> Runner path ostida AppDelegate.m-ni oching. Unda quyidagi kod mavjud:

```
#include "AppDelegate.h"
#include "GeneratedPluginRegistrant.h"
@implementation AppDelegate
- (BOOL)application:(UIApplication *)application
   didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
//
  [GeneratedPluginRegistrant registerWithRegistry:self];
  // Override point for customization after application launch.
  return [super application:application
didFinishLaunchingWithOptions:launchOptions];
}
@end
```

- Belgilangan url bilan brauzerni ochish uchun openBrowser usulini qo'shdik. U bitta argumentni, urlni qabul qiladi.

```
- (void)openBrowser:(NSString *)urlString {
   NSURL *url = [NSURL URLWithString:urlString];
   UIApplication *application = [UIApplication sharedApplication];
   [application openURL:url];
}
```

- DidFinishLaunchingWithOptions usulida tekshirgichni toping va uni kontroller o'zgaruvchisiga o'rnating.
FlutterViewController* controller =
(FlutterViewController*)self.window.rootViewController;
- DidFinishLaunchingWithOptions usulida brauzer kanalini flutterapp.tutorialspoint.com/browse qilib o'rnating:
FlutterMethodChannel* browserChannel = [FlutterMethodChannel

methodChannelWithName:@"flutterapp.tutorialspoint.com/browser"
binaryMessenger:controller];
- ZaifSelf o'zgaruvchisini yarating va joriy sinfni o'rnating:
__weak typeof(self) weakSelf = self;
- Endi setMethodCallHandler dasturini amalga oshiring. Call.method-ga mos ravishda openBrowser-ga qo'ng'iroq qiling. Call.arguments-ni chaqirish orqali url-ni oling va openBrowser-ga qo'ng'iroq paytida uni o'tkazing.

```
[browserChannel setMethodCallHandler:^(FlutterMethodCall* call,
FlutterResult result) {
   if ([@"openBrowser" isEqualToString:call.method]) {
      NSString *url = call.arguments[@"url"];
      [weakSelf openBrowser:url];
   } else {
      result(FlutterMethodNotImplemented);
   }
}];
```

To'liq kod quyidagicha:

```
#include "AppDelegate.h"
#include "GeneratedPluginRegistrant.h"
@implementation AppDelegate
- (BOOL)application:(UIApplication *)application
   didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
   // custom code starts
   FlutterViewController* controller =
(FlutterViewController*)self.window.rootViewController;
   FlutterMethodChannel*          browserChannel          =
[FlutterMethodChannel
```

```
methodChannelWithName:@"flutterapp.tutorialspoint.com/browser"
                                binaryMessenger:controller];
   __weak typeof(self) weakSelf = self;
   [browserChannel      setMethodCallHandler:^(FlutterMethodCall*
call,
FlutterResult result) {
      if ([@"openBrowser" isEqualToString:call.method]) {
         NSString *url = call.arguments[@"url"];
[weakSelf openBrowser:url];
      } else {
         result(FlutterMethodNotImplemented);
      }
   }];
   // custom code ends
  [GeneratedPluginRegistrant registerWithRegistry:self];
  // Override point for customization after application launch.
  return [super application:application
didFinishLaunchingWithOptions:launchOptions];
}
- (void)openBrowser:(NSString *)urlString {
    NSURL *url = [NSURL URLWithString:urlString];
    UIApplication *application = [UIApplication sharedApplication];
    [application openURL:url];
}
@end
```

- Loyihani sozlashni oching.
- Imkoniyatlarga o'ting va Fon rejimlarini yoqing
- * Fonni olish va masofadan xabarnoma qo'shish **
- Endi, dasturni ishga tushiring. U Android versiyasiga o'xshash ishlaydi, ammo Chrome o'rniga Safari brauzeri ochiladi.

Rest API

Flutter HTTP manbalarini iste'mol qilish uchun http to'plamini taqdim etadi. http - kelajakka asoslangan kutubxona va kutish va mos kelmaslik xususiyatlaridan foydalanadi. Bu ko'plab yuqori darajadagi

usullarni taqdim etadi va REST asosidagi mobil ilovalarni ishlab chiqishni soddalashtiradi.

http to'plami yuqori darajadagi sinfni va web-so'rovlarni bajarish uchun http ni taqdim etadi.

- http class http so'rovlarining barcha turlarini bajarish uchun funksionallikni ta'minlaydi.
- http usullari urlni va Dart Map orqali qo'shimcha ma'lumotlarni qabul qiladi (ma'lumotlar, qo'shimcha sarlavhalar va boshqalar). U serverdan so'raydi va javobni qaytarib to'playdi async / kutish namunasi. Masalan, quyida keltirilgan kod belgilangan urldan ma'lumotlarni o'qiydi va ularni konsolda chop etadi.

print(await http.read('https://flutter.dev/'));

Ba'zi asosiy usullar quyidagicha:

- o'qish - GET usuli orqali ko'rsatilgan urlni so'rang va javobni Future <String> sifatida qaytaring
- get - Belgilangan urlni GET usuli orqali so'rang va javobni Future <Response> sifatida qaytaring. Javob - bu javob ma'lumotlarini ushlab turadigan sinf.
- post - Belgilangan URL manzilini POST usuli orqali yuborilgan ma'lumotlarni joylashtiring va javobni Future <Response> sifatida qaytaring
- put - Belgilangan urlni PUT usuli orqali so'rang va javobni Future <Response> deb qaytaring
- head - Belgilangan urlni HEAD usuli orqali so'rang va javobni Future <Response> deb qaytaring
- o'chirish - Belgilangan URL manzilini DELETE usuli orqali so'rang va javobni Future <Response> deb qaytaring

http shuningdek ko'proq standart HTTP mijoz sinfini, mijozni taqdim etadi. mijoz doimiy aloqani qo'llab-quvvatlaydi. Muayyan serverga juda ko'p so'rovlar yuborilganda foydali bo'ladi. Yopish usuli yordamida uni to'g'ri yopish kerak. Aks holda, u http sinfiga o'xshaydi. Namuna kodi quyidagicha:

```
var client = new http.Client();
try {
 print(await client.get('https://flutter.dev/'));
```

```
} finally {
  client.close();
}
```

Product service API-ga kirish

Web-serverdan mahsulot ma'lumotlarini olish uchun oddiy dastur yaratib, keyin ListView-dan foydalanib mahsulotlarni namoyish qilaylik.

- Android studiyasida yangi Flutter dasturini yarating, product_rest_app
- Standart boshlang'ich kodini (main.dart) bizning product_nav_app kodimiz bilan almashtiring.
- Aktivlar papkasini product_nav_app-dan product_rest_app-ga nusxalash va pubspec.yaml fayliga aktivlar qo'shish.

```
flutter:
  assets:
    - assets/appimages/floppy.png
    - assets/appimages/iphone.png
    - assets/appimages/laptop.png
    - assets/appimages/pendrive.png
    - assets/appimages/pixel.png
    - assets/appimages/tablet.png
```

- Http paketini pubspec.yaml faylida quyida ko'rsatilgandek sozlang:

```
dependencies:
  http: ^0.12.0+2
```

- Bu yerda biz http to'plamining so'nggi versiyasidan foydalanamiz. Android studiyasi pubspec.yaml yangilanganligi to'g'risida paketli ogohlantirish yuboradi.

- Bog'liqlarni olish parametrini bosing. Android studio to'plamni Internetdan oladi va uni dastur uchun to'g'ri sozlaydi.
- Http. paketini main.dart fayliga import qiling:

```
import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;
```

Mahsulot ma'lumotlari bilan yangi JSON faylini yarating, products.json, quyida ko'rsatilgandek:

```
[
  {
    "name": "iPhone",
    "description": "iPhone is the stylist phone ever",
    "price": 1000,
    "image": "iphone.png"
  },
  {
    "name": "Pixel",
    "description": "Pixel is the most feature phone ever",
    "price": 800,
    "image": "pixel.png"
},
  {
    "name": "Laptop",
    "description": "Laptop is most productive development tool",
    "price": 2000,
    "image": "laptop.png"
  },
  {
    "name": "Tablet",
    "description": "Tablet is the most useful device ever for meeting",
    "price": 1500,
    "image": "tablet.png"
  },
  {
    "name": "Pendrive",
    "description": "Pendrive is useful storage medium",
    "price": 100,
    "image": "pendrive.png"
  },
  {
```

"name": "Floppy Drive",
"description": "Floppy drive is useful rescue storage medium",
"price": 20,
"image": "floppy.png"
}
]

- JSONWebServer yangi jild yarating va JSON faylini joylashtiring products.json
JSONWebServer bilan har qanday web-serverni ildiz katalogi sifatida ishga tushiring va web-yo'lini oling. Masalan, http://192.168.184.1:8000/products.json. Biz apache, nginx va boshqalar kabi har qanday web-serverlardan foydalanishimiz mumkin.
- Eng oson yo'li - tugun asosidagi http-server dasturini o'rnatish. Http-server dasturini o'rnatish va ishga tushirish uchun quyidagi amallarni bajaring.
    - Nodejs dasturini o'rnating (https://nodejs.org/en/)
    - JSONWebServer papkasiga o'ting.
    cd /path/to/JSONWebServer
- Http-server paketini npm yordamida o'rnating
    npm install -g http-server
Endi serverni ishga tushiring.

http-server . -p 8000
Starting up http-server, serving .
Quyidagi manzil bo'yicha kirish mumkin:
http://192.168.99.1:8000
http://192.168.1.2:8000
http://127.0.0.1:8000
Hit CTRL-C to stop the server
- Lib papkasida Product.dart yangi fayl yarating va unga Product sinfini ko'chiring.
- Xaritadagi, ma'lumotlar xaritasini mahsulot obyektiga aylantirish uchun Product.fromMap mahsulotiga Product konstruktorini yozing. Odatda, JSON fayli Dart Map obyektiga, so'ngra tegishli obyektga (Product) aylantiriladi.

factory Product.fromJson(Map<String, dynamic> data) {

```
    return Product(
      data['name'],
      data['description'],
      data['price'],
      data['image'],
    );
  }
```

- Product.dart-ning to'liq kodi quyidagicha:

```
class Product {
 final String name;
 final String description;
 final int price;
 final String image;
 Product(this.name, this.description, this.price, this.image);
 factory Product.fromMap(Map<String, dynamic> json) {
   return Product(
     json['name'],
     json['description'],
     json['price'],
     json['image'],
   );
  }
}
```

Mahsulot ma'lumotlarini web-serverdan Ro'yxat <Product> obyektiga olish va yuklash uchun ikkita usulni yozing: parseProducts va fetchProducts - asosiy sinfda.

```
List<Product> parseProducts(String responseBody) {
 final parsed = json.decode(responseBody).cast<Map<String,
dynamic>>();
 return parsed.map<Product>((json) =>
Product.fromJson(json)).toList();
}

Future<List<Product>> fetchProducts() async {
 final response = await
http.get('http://192.168.1.2:8000/products.json');
```

```
  if (response.statusCode == 200) {
    return parseProducts(response.body);
  } else {
    throw Exception('Unable to fetch products from the REST API');
}
}
```

Savol va topshiriqlar

1. Cross platformada qo‘llaniladigan maxsus belgilar?
2. Flutterda controller yaratish?
3. Rest Api va uning ishlash prinspi?
4. GET metodini tushintirib bering.
5. To‘plamlardan foydalanish?
6. Product service API va uni qo‘llash algoritmi?
7. Http paketi bilan ishlash bo‘yicha misollar keltiring.
8. Flutterda xatoliklar va ularning oldini olish?
9. Json bilan ishlash va undan ma’lumotlarni o‘qib olish?
10. JSONWebServer bilan ishlash prinspi?

FOYDALANGAN ADABIYOTLAR

1. Company(s) nor Stack Overflow. Android™ Notes for Professionals. 1329 pages, 2019.

2. Валерий Станиславович Яценков. Java за неделю. 2018. ISBN 978-5-4490-4684-0.

3. Ян.Ф.Давин. Android зборник рецептов. 2-е издание. Москва. Санкт-Петербург. Киев. 2018.

4. Head First Java, 2nd Edition, Kathy Sierra, Bert Bates, O'Reilly, 2005

5. Reto Meier, "Professional Android Application Development", Pulished by John Wiley & Sons-2013-816 p.

6. Wei-Meng Lee, Android™ Application cookbook –John Wiley & Sons-2013-410p.

7. Java за неделю Вводный курс, Валерий Станиславович Яценков, 2018.

8. Android Programming: The Big Nerd Ranch Guide (Big Nerd Ranch Guides) 4th Edition, Dec 15, 2021.

9. iOS Programming: The Big Nerd Ranch Guide 7th Edition, Dec 15, 2021.

10. Swift Programming: The Big Nerd Ranch Guide (Big Nerd Ranch Guides) 3rd Edition, October 14, 2020.

MUNDARIJA

NISHANOV A.X., ALLABERGANOV O.R.,
MADAMINOV U.A.

MOBIL ILOVALARINI ISHLAB CHIQISH

O‘QUV QO‘LLANMA

Toshkent - “METODIST NASHRIYOTI” – 2024

Muharrir: Bakirov Nurmuhammad

Texnik muharrir: Tashatov Farrux
Musahhih: Muhammadiyeva Sevinch
Dizayner: Ochilova Zarnigor

Bosishga 1.04.2024.da ruxsat etildi.
Bichimi 60x90. “Times New Roman” garniturasi.
Ofset bosma usulida bosildi.
Shartli bosma tabog‘i 13. Nashr bosma tabog‘i 12,25.
Adadi 300 nusxa

“METODIST NASHRIYOTI” MCHJ matbaa bo‘limida chop etildi.
Manzil: Toshkent shahri, Shota Rustaveli 2-vagon tor ko‘chasi, 1-uy.

  ***+99893 552-11-21***