Jakbarov Odiljon,
Goyipov Umidjon

ANDROID PLATFORMASIDA MOBIL ILOVALARNI ISHLAB CHIQISH

O'ZBEKISTON RESPUBLIKASI
OLIY VA O'RTA MAXSUS TA'LIM
VAZIRLIGI

Jakbarov Odiljon Otamirzaevich,
Goyipov Umidjon Gulomjonovich

ANDROID PLATFORMASIDA MOBIL ILOVALARNI ISHLAB CHIQISH

60610200-Axborot tizimlari va texnologiyalari ta'lim yo'nalishi talabalari uchun

"Chustiy" nashriyoti. -2023 йил.

UYK: 821.512.133-1
KBK: 22.5(O'zb)3
O-2

Jakbarov Odiljon , Goyipov Umidjon **ANDROID PLATFORMASIDA MOBIL ILOVALARNI ISHLAB CHIQISH** : O'quv qo'llanma – N.: "Chust" nashriyoti, 2023 – 148 b.

Ushbu darslik bakalavriatning Axborot tizimlari va texnalogiyalari yo'nalishlari talabalari uchun mo'ljallangan bo'lib, unda «Android platformasida mobil ilovalarni ishlab chiqish» fanidan nazariy va amaliy mashg'ulotlariga oid o'quv materiallari hamda o'qitishning innovasion texnologiyalari asosida darslarni tashkil etish bo'yicha ishlanmalardan na'munalar keltirilgan.

Darslikdan magistrantlar va ilmiy tadqiqotchilar ham foydalanishlari mumkin.

Taqrizchilar:

I.X.Normatov – O'zMU, AMTI fakulteti Axborot xavfsizligi kafedrasi professori, f-m.f.d.

D.X.Baratov – Toshkent Davlat transport universiteti prorektori texnika fanlari doktori, professor.

D.To'xtanazarov – O'zbekiston xalqaro islom akademiyasi dotsenti, PhD.

M.Olimov – NamMQI, Axborot tizimlari va texnologiyalari kafedrasi professori

ISBN 978-9910-9844-8-8

SO'Z BOSHI

Hozirgi vaqtda Android operatsion tizimi mobil qurilmalar uchun eng mashhur platforma hisoblanadi. Ushbu platforma nazorati ostida ishlaydigan turli smartfon va planshetlarning xilma-xilligi va keng tarqalishi mobil ilovalar bozorining o'sishini rag'batlantirib, Androidni ishlab chiqish ko'nikmalarini zamonaviy dunyoda yuqori talabga aylantirmoqda.

Ushbu darslikning maqsadi, Android operatsion tizimi uchun mobil ilovalarni ishlab chiqish bilan tanishtirishdir. Shu bilan birga, ushbu platformaning API tomonidan taqdim etilgan asboblar va vositalarni yoritish bilan birga, asosiy tamoyillarni tushuntirishga va amalda o'ziga xos xususiyatlarni namoyish etishga qaratilgan.

Darslik mobil ilovalarni ishlab chiqish bo'yicha mutaxassis kadrlar va keng o`quvchilar ommasi uchun yozilgan. Uni o'rganish uchun Androidni ishlab chiqish tajribasiga ega bo'lish yoki Android haqida to`liq bilish shart emas. O'quvchi noldan boshlashi ham mumkin. Ayni paytda Android ilovalarini ishlab chiqish uchun ko'plab vositalar va texnologiyalar mavjud. Ko'pgina professional dasturchilar o'zlarining kundalik ishlarida bitta texnologiyadan foydalanadilar va ko'pincha boshqalaridan bexabar. Biz Eclipse vositasi va mobil qurilma emulyatoriga asoslangan eng mashhur texnologiyalardan birini qaraymiz. O`quvchi Java dasturlash tilining elementlarini biladi deb taxmin qilinadi. Biroq, kitobni o'qish uchun ish tajribasi yoki Java tilini chuqur tushunish talab qilinmaydi. Faqat tilning asosiy konstruktsiyalari bilan tanishish kerak bo`ladi.

Birinchi bob platformaga kirishdan iborat. U ishlab chiqish vositalarini, asosiy komponentlarini, loyihani qanday yaratish, kompilyatsiya qilish va joylashtirishni tavsiflaydi. Ikkinchi bobda XML tartib faylidan foydalanish bo`yicha hamda Uchinchi bob Eclipse vizual rivojlanish muhitiga bag'ishlangan. Ushbu bob darslikning dastlabki ikki bobidagi ma'lumotlarni amalda ko'rsatish orqali amalga oshiriladi.

Keyingi boblar mos ravishda Android ilovasi komponentlari, MVC arxitekturasida oddiy loyihalar, platforma vidjet komponentlariga, resurslar bilan ishlashga va ma'lumotlarni saqlash metodlariga bag'ishlangan. Ushbu mavzular har qanday Android ilovasini ishlab chiquvchisi uchun zarur bo'lgan minimal talab hisoblanadi.

Oxirgi ikki bobda Android ilovalarini ishlab chiqishning yanada murakkab masalalari ko'rib chiqiladi. View klass tomonidan boshqariladigan hodisalar, kontent provayderlari, asinxron bajarilish, shunindek, Google tomonidan boshqariladigan Android ilovalari uchun maxsus mexanizm sifatida Android Market saytida ilovani yuklash ishlari amalga oshiriladi.

1-BOB. ANDROID PLATFORMASI HAQIDA QISQACHA MA'LUMOT

Google Android loyihasini 2005 yilda sotib olganida, hamma juda hayratda qolgan va bu korxona muvaffaqiyatiga ishonmagan edi. Muvaffaqiyatli kompaniya istiqbollari juda noaniq bo'lgan, bozorning asosiy bo'lmagan segmentida ishtirok etishi kerak edi. Shunga qaramay, mobil operatsion tizimni ochiq platformada ishlab chiqish va qo'llab-quvvatlash mumkinligini isbotlagan Google strategiyasining to'g'riligini vaqt isbotladi. Hozirda Google Android loyihasiga pul va resurslarni sarmoya qilishda davom etmoqda, bu juda muvaffaqiyatli bo'ldi. 2010 yil iyul oyida har kuni 160000 ta Android mobil qurilmalari faollashtirildi, birinchi Android qurilmalari faqat 2008 yil oktyabr oyida paydo bo'lganini hisobga olsak, bu o`sha davr uchun yomon natija emasdi. Ikki yildan kamroq vaqt ichida Google mobil bozorini tubdan o'zgartira boshlagandi.

Platformaning ochiqligi tufayli ishlab chiquvchilar uchun mustaqil ravishda pul topish hech qachon oson bo'lmagan. Android foydalanuvchilari dasturchi sifatida siz haqingizda hech narsa bilishmasligi mumkin, lekin ular Googleni bilishadi va ishonishadi. Har kim o'z ilovasini Google tomonidan boshqariladigan Android Market saytida nashr qilishi mumkin, bu ilovalar sifatini kafolatlaydi. Tabiiyki, foydalanuvchilar ushbu saytda ilovalarni bajonidil sotib olishadi.

1-§. Android uchun ilovalarni ishlab chiqish haqida

Tayanch so'z va atamalar

Android, Application Programming Interface, platforma, ochiq platforma, Android Market, GPS.

Ilovangizni butun dunyo bo'ylab millionlab yangi foydalanuvchilarga taqdim etishni xohlaysizmi? Ilova tuzatilgan va sinovdan o'tkazilgandan so'ng uni nashr qilishni xohlaysizmi? Intellektual mehnat orqali pul ishlashni xohlaysizmi? Ochiq platformada ishlashni xohlaysizmi? Agar ushbu savollardan kamida bittasiga javob "ha" bo'lsa, Android ilovalarini ishlab chiqish

bilan shug'ullanishingiz mantiqan juda to`g'ri bo'ladi. Agar hali ham bir qarorga kelmagan bo'lsangiz, kitobni o'qishda davom eting, shunda bilim bilan qurollangan holda qaror qabul qilishingiz osonroq bo'ladi.

Android platformasi juda ko'p sonli **API** (Application Programming Interface) kutubxonalari bilan birga keladi. Ularning yordami bilan nisbatan qisqa vaqt ichida to'liq funktsional dasturni ishlab chiqish oson. Android Marketda ro'yxatdan o'tganingizdan so'ng, ilovangizni yuklang va namoyish qiling. Boshqa xarid saytlaridan farqli o'laroq, Android Marketda mahsulotni tasdiqlash bosqichi yo'q. Siz shunchaki ariza yozishingiz va uni yuklashingiz kerak.

Texnik jihatdan, har kim xohlagan narsani yozishi mumkin, ammo Google qoidalarga rioya qilishingizni va ilovalarni qabul qilingan formatda yuklanishini qat'iy tavsiya qiladi. Shuni unutmaslik lozimki, Android foydalanuvchilari dunyoning barcha burchaklarida va har qanday yoshdagi, professional, etnik va boshqa toifalarni qamrab oladi.

Ochiq platforma. Android operatsion tizimi ochiq platforma hisoblanadi. Bu bitta apparat ishlab chiqaruvchisi yoki provayderiga bog'lanmaganligini anglatadi. Platformaning ochiqligi Android-ga bozorni tezda zabt etish imkonini beradi. Barcha apparat ishlab chiqaruvchilari va provayderlari Android qurilmalarini ishlab chiqarishi va sotishi mumkin. Android manba kodi **http://source.android.com** manzilida joylashgan bo'lib, uni hamma ko'rishi va o'zgartirishi mumkin. Sizni qiziqtirgan muammo qanday hal qilinishini tahlil qilish uchun manba kodini o'rganishga hech qanday taqiq qo`yilmaydi. Ochiq dasturlash kodi mobil telefon ishlab chiqaruvchilariga o'z qurilmalari uchun qulay foydalanuvchi interfeyslarini yaratish va ularga har qanday vositalarni joylashtirish imkonini beradi. Bu barcha ishlab chiquvchilarni bir xil raqobatdosh asosga qo'yadi.

Uskunaning mosligi. Android operatsion tizimi turli xil ekran o'lchamlari bilan har xil turdagi qurilmalarda ishlashi mumkin. Albatta, ilovaning ishlashi ekran o`lchamlari va ruxsatiga bog'liq, shuning uchun Android turli ekran

o'lchamlariga moslashuvchi ilovalar yaratishda yordam beradigan vositalar to'plami bilan birga keladi.

Google siyosati yanada cheklangan: u faqat mos keluvchi qurilmalarda ishlaydigan ilovalarni qabul qiladi. Misol uchun, agar ilova old kameraga muhtoj bo'lsa, u faqat o'rnatilgan old kameraga ega bo'lgan qurilmalar uchun Android Marketda ko'rsatiladi. Nostandart uskunalar - bu avtomatik aniqlash tartibi. Ilovalarni Android Marketda chiqarish haqida qo'shimcha ma'lumot olish uchun 10-bobga qarang.

Android qurilmalari muvofiqlik uchun sertifikatlangan ma'lum bir qo'shimcha qurilmalar, jumladan quyidagilar (ro'yxat to'liq emas) mujassamlanadi:

- *kamera;*
- *kompas;*
- *GPS-qabul qiluvchi (Global Positioning System - global joylashishni aniqlash tizimi);*
- *Bluetooth qabul qiluvchi.*

Muayyan qurilma konfiguratsiyasi uchun moslik vositasi tavsifini **http://source.android.com/compatibility/overview.html** sahifasida topish mumkin. Muvofiqlik sertifikati ilovangizni ro'yxatda ko'rinadigan barcha qurilmalarda ishlashini kafolatlaydi.

2-§. Android va dastur infratuzilmasi

Tayanch so'z va atamalar

Android, Linux, Dalvik virtual mashinasi, WebKit, platforma, SQLite, SSL, GPS, Operatsion tizim, Activity manager, View system, Location manager, Android SDK, Silent Mode Toggle, Eclipse, plagin, Telephony manager.

Linux yadrosining tepasida bir qator ilovalarni ta'minlovchi **Android** tizimi mavjud. Ushbu mablag'larning aksariyati ko'plab ochiq manba loyihalaridan

qarzga olingan. Eng muhim Android infratuzilma vositalarini quyida keltirib o`tiladi:

Android ish vaqti. Java tilining asosiy kutubxonalari va **Dalvik** virtual mashinasidan iborat.

Grafik kutubxona. U ikki va uch o'lchovli kompyuter grafikasini yaratish uchun ishlatiladi va platformali ko'p tilli **API (Application Program Interface)** kutubxonasi hisoblanadi.

WebKit. Veb-sahifalar tarkibini ekranda aks ettiruvchi va ularni yuklashni osonlashtiradigan ochiq veb-brauzer mexanizmi.

SQLite. Portativ qurilmalarga o'rnatish uchun mo'ljallangan ochiq relyatsion ma`lumotlar bazasi (DBMS).

Multimedia infratuzilmasi. Ovoz va videoni yozib olish va ijro etish imkonini beruvchi kutubxonalar to'plami.

SSL (Secure Sockets Layer - xavfsiz rozetkalar qatlami). Internet aloqasi xavfsizligini ta'minlovchi kutubxonalar to'plami.

1.1.1-rasmda eng foydali Android kutubxonalari ro'yxatini keltirib o`tiladi.

1.1.1-rasm. Eng foydali Android kutubxonalari ro'yxati.

Dastur infratuzilmasi. Barcha kutubxonalar **Android** operatsion tizimi orqali dastur ishlab chiqaradi. Androidning SQLite ma'lumotlar bazasi yoki media bilan qanday ishlashi haqida tashvishlanish shart emas - ularning barchasi ilovada foydalanishga tayyor. Androidni ishlab chiqish jamoasi eng foydali kutubxonalarni tanlab olgan va ularni API orqali taqdim etgan. Ushbu interfeyslar kutubxonalarni egallab oladi va ulardan Android platformasida foydalanishni osonlashtiradi. Quyidagi ilovada ishlatilishi mumkin bo'lgan eng foydali Android kutubxonalar keltiriladi:

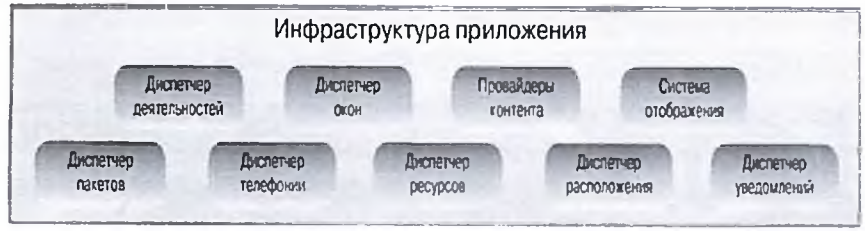
Activity manager. Jarayonning davrlarini boshqaradi.

Telephony manager (telefoniya menejeri). Telefon xizmatlariga va ularning foydalanuvchilari haqidagi ba'zi ma'lumotlarga kirishni ta'minlaydi, masalan, telefon raqamlari.

View system (tizimni ko'rish). Foydalanuvchi interfeysi boshqaruvini boshqarish.

Location manager (joylashuv menejeri). Qurilmaning joriy geografik joylashuvini aniqlash.

Quyidagi 1.3.2-rasmda dastur doirasida foydalaniladigan kutubxonalarni keltirib o`tiladi.

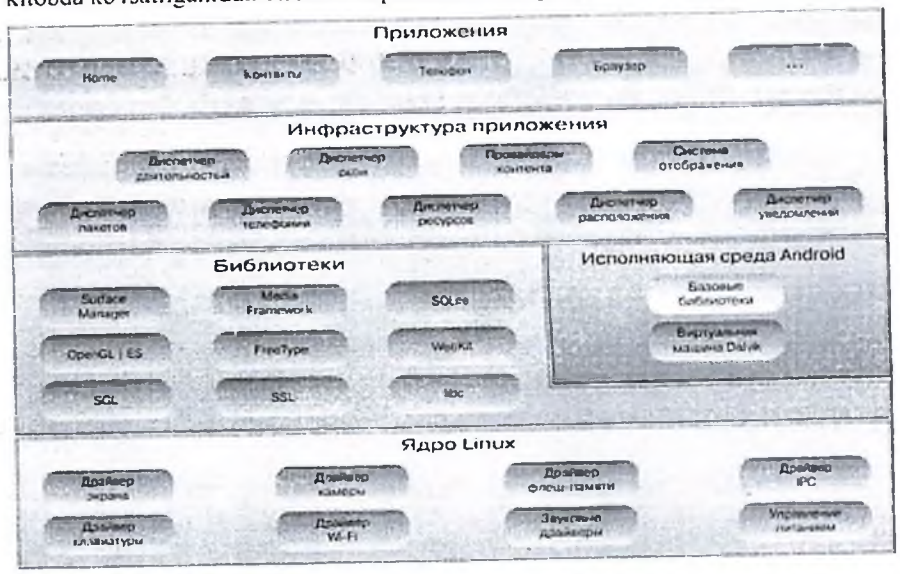


1.3.2-rasm. Android ilovalari kutubxonalari

Android operatsion tizimi, yadrodan APIgacha, kuchli mobil ilovalarni ishga tushirish uchun barqaror, vaqt sinovidan o'tgan ochiq kodli texnologiyalar asosida qurilgan. To'liq Android infratuzilmasi esa quyidagi 1.3.3-rasmda keltirib o`tiladi.

Ba'zan, Android ilovasini ishlab chiqishda, dasturiy interfeysga ega bo'lmagan operatsion tizim resurslariga kirish kerak bo'ladi. Android ilova ishlab chiquvchilari ushbu resurslarga tez-tez kirishlarini taqiqlanmagan. Android manba kodlariga cheksiz kirishingiz mumkin, ularni yuklab olishingiz va o'zgartirishingiz mumkin. Tabiiyki, buning uchun etarli malakaga ega bo'lishingiz kerak va har qanday o'zgarishlar nojo'ya ta'sirlardan qochish uchun turli vaziyatlarda sinchkovlik bilan tekshirilishi kerakligini hisobga olish kerak. Biroq, kodlar nafaqat Android funksiyasini o'zgartirish uchun, balki hujjatlarda tasvirlanmagan kutubxona ishining tafsilotlarini topish uchun ham foydali bo'lishi mumkin.

Tizimni sozlash. Windows, Linux va Mac OS X kabi turli xil operatsion tizimlarda ishlaydigan kompyuterlarda Android ilovalarini ishlab chiqish mumkin. Ushbu darslikda tizim sifatida Windows 10 OS o'rnatilgan deb taxmin qilinadi. Agar kompyuter boshqa operatsion tizimda ishlayotgan bo'lsa, ba'zi oynalar kitobda ko'rsatilganidan biroz boshqacha ko'rinishga ega bo'ladi..



1.3.3-rasm. Android operatsion tizimining tuzilishi.

Operatsion tizim. Android ilovalarini ishlab chiqish muhiti quyidagi platformalar tomonidan qo'llab-quvvatlanadi:

Windows XP (32-bit), Vista (32-bit yoki 64-bit) va Windows 7 (32-bit yoki 64-bit);

Mac OS X 10.5.8 yoki undan keyingi versiyalari (faqat x86 protsessorlarida);

Linux (ishlab chiqish vositalari Linux Ubuntu Hardy Heron tizimida sinovdan o'tgan).

E'tibor berish lozim, 64-bitli ilovalar 32-bitli ilovalarni qo'llab-quvvatlaydi. Ushbu darslik Windows10 opertsion tizimining 64 bitli versiyasi yordamida qo'llaniladi. Agar kompyuteringizda Windowsning boshqa versiyasi ishlayotgan

bo'lsa, ushbu kitobdagi skrinshotlar kompyuter ekranida ko'rganingizdan biroz farq qilishi mumkin. Agar kompyuteringizda **Linux** yoki **Mac** o'rnatilgan bo'lsa, fayl marshrutlari quyidagicha yozilishi kerakligini yodda tutish lozim: *C:\path\to \file. txt, lekin shunday: /path/to/file.txt.*

Rivojlantirish vositalarini o'rnatish va sozlash. Quyida Android ilovalarini ishlab chiqish uchun o'rnatish va sozlash kerak bo'lgan vositalar ro'yxati keltirib o'tiladi:

Java JDK. Rivojlantirish platformasi uchun asos.

Android SDK. Android kutubxonalariga kirishni ta'minlaydi va Android ilovalarini ishlab chiqish imkonini beradi.

Eclipse. Java, Android SDK va Android ADT (Android Development Tools - Android dasturlash vositalari) ni birlashtirgan integratsiyalashgan ishlab chiqish muhiti. Android ilovalarini yaratish vositalarini taqdim etadi.

Android ADT. Android ilovalari uchun zarur bo'lgan fayllar va tuzilmalarni yaratish kabi ishlarni bajaradigan **Eclipse plagini**.

Bu bo'limlar ushbu vositalarning har birini yuklab olish, o'rnatish va sozlash haqida batafsil ma'lumot beradi.

Ochiq kodli dasturiy ta'minotning katta afzalliklaridan biri shundaki, siz istalgan vaqtda kerakli vositalarni yuklab olishingiz va ular bilan bepul ishlashingiz mumkin. Bu qoida Android ishlab chiqish vositalariga to'liq taalluqlidir. Kuchli Android ilovalarini yaratish uchun kerak bo'lgan barcha ilovalar bepul yuklab olinadi.

Bundan tashqari, pullik ilovalar mavjud. Tabiiyki, ular ba'zi qo'shimcha xizmatlarni taqdim etadilar, ammo bepul ilovalar shunchalik yuqori sifatga egaki, ko'pchilik professional ishlab chiquvchilar ulardan foydalanadilar.

3-§. Foydalanuvchi interfeysini yaratish

Tayanch so'z va atamalar

Ovozsiz rejim, Application Name, Project, Silent Mode Toggle, Eclipse, plagin, manager.

Shunday qilib, Android nima ekanligini allaqachon bilamiz va oddiy dastur yaratishga qodirmiz. Shunday qilib, qiziqarli qismga, kundalik hayotda foydalanish yoki Android Marketda yuklash mumkin bo'lgan haqiqiy ilovani yaratishga o'tish mumkin.

Ushbu bobda yaratiladigan ilova bir tugmani bosish bilan telefon qo'ng'irog'i rejimini o'zgartirish imkonini beradi. Bu juda foydali va har bir foydalanuvchi o'z telefonida bo'lishni xohlaydigan dastur hisoblanadi. Shuning uchun, ushbu darslikning yagona o'quvchisi sifatida, ushbu ilovani sotish orqali ko'p pul ishlash mumkin edi. Biroq, hech kim sizga ushbu ilovaning foydali modifikatsiyasini o'ylab topishni va uni o'z nomingiz bilan sotuvga qo'yishni taqiqlay olmaydi, uni takomillashtish va mualliflik qilish imkoniyati mavjud.

Tasavvur qiling, ish joyidasiz va kompaniya ma'muriyati sizni muhim uchrashuvga taklif qildi. Tabiiyki, uchrashuvni boshlashdan oldin telefon qo'ng'iroq'ini jim rejimga (yoki tebranish rejimiga) o'tkazasiz. Axir, uchrashuvning eng keskin vaqtida telefoningiz to'satdan jiringlashini xohlamaysiz va hozir bo'lganlarning hammasining norozi nigohlari sizga qaratiladi. Uchrashuv tugagach, siz telefoningizni qo'ng'iroq qilish rejimiga o'tkazasiz. Biroq, tovushning eng baland ovozda bo'lishini xohlamaysiz va atrofingizdagilarning hammasi hayratda qotib qoladilar. Siz ma'lum bir ovoz balandligini afzal ko'rasiz. Shuning uchun, qo'ng'iroqni har safar yoqganingizda, qo'ng'iroq tovushini qayta sozlashingiz kerak. Har safar qo'ng'iroq rejimini o'zgartirganingizda, menyularni ochishingiz va kerakli pastki menyularni topishingiz kerakligini, shuningdek, yig'ilishlarga tez-tez qo'ng'iroq qilish mumkinligini hisobga olsak, juda mashaqqatli deb taxmin qilishingiz mumkin. Buni biroz osonroq qilish uchun, keling, qo'ng'iroq qilish rejimi bitta tugmani bosgandan keyin o'zgaradigan qilib yarataylik. Bu ilovamiz

orqali amalga oshiriladi. Bundan tashqari, qo'ng'iroqni yoqsangiz, uning ovozini qayta sozlash shart emas. U qo'ng'iroq o'chirilgan paytdagi darajada qoladi.

Ovozsiz rejim loyihasini almashtirishni yaratish. Endi biz **Silent Mode Toggle** ilovasini yaratamiz. Eclipse ish stolida Android loyihasini qanday yaratishni bilamiz.

Yangi loyiha yaratishdan oldin, Eclipse ish stolida ochilgan barcha fayllarni yopiladi. Buni amalga oshirish uchun **Eclipse**ning o'ng panelida har bir tugmani navbat bilan bosiladi. Bundan tashqari, asosiy menyudan **File Close All** tanlanishi ham mumkin.

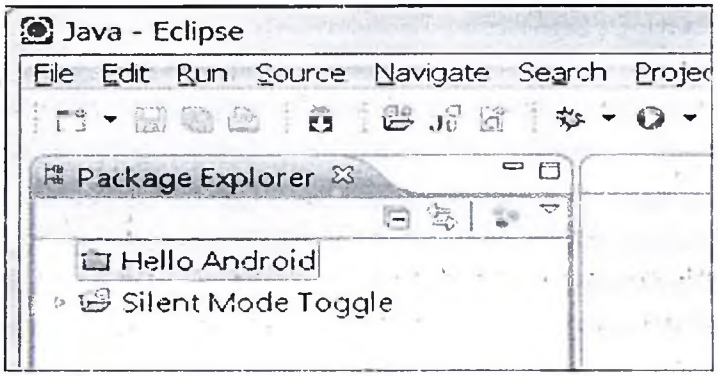
Barcha fayllarni yopgandan so'ng, Hello Android loyihasini ham yopiladi. Buni amalga oshirish uchun **Eclipse** ish stolining **Package Explorer** oynasida Hello Android loyihasini o'ng tugmasini bosib, kontekst menyusidan **Project Close** buyrug'ini tanlanadi. Bu **Eclipse**ga ushbu loyihada bir muncha vaqt ishlamasligingizni bildiradi. Eclipse loyihaning holatini kuzatish uchun foydalaniladigan resurslarni bo'shatiladi va **Eclipse**dagi barcha operatsiyalar tezligi oshadi.

Hamma narsa **Silent Mode Toggle** ilovasini yaratishga tayyor. Avval loyiha yaratiladi, buning uchun **File New Project buyrug'i** tanlanadi. Ro'yxatdan Android loyihasi turini tanlab, "Keyingi" tugmasini bosiladi. Jadvalga muvofiq loyiha xususiyatlarini o'rnatiladi. (*1.3.1-rasm*).

Параметр	Значение
Application Name (ilova nomi)	*Silent Mode Toggle*
Project Name (loyiha nomi)	*Silent Mode Toggle*
Contents (mundarija)	*Standart qiymatlarni qoldiring*
Build Target (maqsad платформаси)	*Android 2.2*
Package Name (paket nomi)	*com.dummies.android.silentmodetoggle*
Create Activity (aktivlik yaratish)	*MainActivity*
Min SDK Version (SDK minimum версия)	*8*

1.3.1-jadval. Ovozsiz rejim loyihasining xususiyatlarini o'zgartirish

Finish tugmasini bosiladi, bunda Silent Mode Toggle ilovasi Package Explorer oynasida ko'rsatilishi kerak (*1.3.4-rasm*).



1.3.4-rasm. Eclipse Workbench-da jimlik rejimni almashtirish ilovasi.

Ba'zan loyihani qurishda **Eclipse** ish stoli quyidagiga o'xshash xato xabarini qaytaradi: "yaratishdagi xatolar hal etilmaguncha loyihani qurish mumkin emas". Bunday holda, loyiha nomini o'ng tugmasini bosiladi va kontekst menyusidan Android Tools Fix Project Properties tanlanadi. Eclipse avtomatik ravishda loyiha va ish maydoni marshrutlarini sinxronlashtiradi.

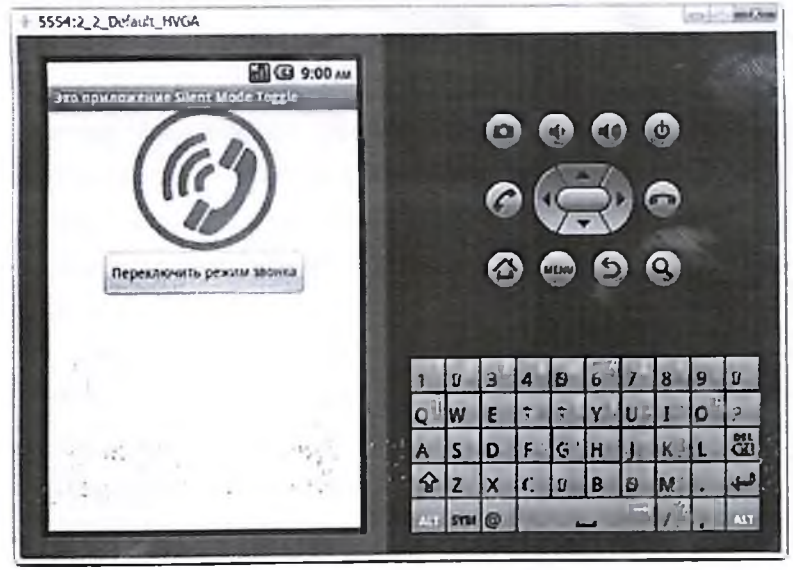
Ilova tartibi. Silent Mode Toggle loyihasi Eclipse ish stolida aniqlangan, endi ilovaning grafik foydalanuvchi interfeysini ishlab chiqishni boshlash mumkin. Foydalanuvchi interfeysi - bu foydalanuvchi qurilma bilan o'zaro aloqada bo'lgan dasturning bir qismi. Aslida, u "ilovaning yuzi". Uning tashqi ko'rinishi ilovaning foydalanuvchi tajribasini, interfeysning joylashuvi esa dasturdan foydalanish qulayligini belgilaydi.

Silent Mode Toggle ilovasida ekranning markazida joylashgan bitta tugma mavjud bo'lib, u jiringlash rejimini jimlik rejimdan baland ovozga yoki aksinchaga o'zgartiradi. Tugmaning tepasida foydalanuvchiga telefon hozirda jim yoki baland ovozda ishlayotganligini vizual ravishda xabardor qiladigan tasvir mavjud bo'ladi. Bir marta ko'rish yuz marta eshitishdan yaxshiroqdir, shuning uchun 1 va 2 rejimlarda ilova qanday ko'rinishini ko'rish uchun *1.3.5-rasm* va *1.3.6-rasm* larni ko'rib chiqamiz.

Ushbu ikkita chizmani yaratilgan ilova qanday ko'rinishini aytadigan o'ziga xos "talab" sifatida tasavvur qilinadi. Shu bilan birga, uni dogma sifatida qabul

qilinadi. Ushbu rejimlarda tasvirning o'lchami va mazmuni, tugma yorlig'ining o'lchami, rangi va shrifti bilan tajriba qilinadi. Estetik jihatdan yanada jozibali interfeys yaratish mumkin. Xususan, rasm va tugma o'rtasida kichik bo'shliqni yaratishga harakat qilamiz.

1.3.5-rasm. Ovozsiz rejimda



1.3.6-rasm. Ovozli rejimda

O'z-o'zini tekshirish uchun savollar va mashqlar

1. Android qurilmalari muvofiqlik uchun sertifikatlangan qaysi qo'shimcha qurilmalar mujassamlanadi?

2. Android operatsion tizimi qanday platforma hisoblanadi?

3. Eng muhim Android infratuzilma vositalarini keltirib o'ting.

4. Ilovada ishlatilishi mumkin bo'lgan eng foydali Android kutubxonalarni keltiring.

5. Android operatsion tizimining tuzilishini keltiring.

6. Android ilovalarini ishlab chiqish muhiti qaysi platformalar tomonidan qo'llab-quvvatlanadi?

2-BOB. XML FAYLIDAN FOYDALANISH

Eclipse ish maydonida barcha ilova fayllari Android loyihasining **res/layouts** papkasida saqlanadi. Bir necha daqiqa oldin bo'sh Silent Mode Toggle loyihasini yaratganingizda, ADT plaginlari siz uchun main.xml faylini yaratdi va uni res/layouts jildiga joylashtirdi. Bu standart foydalanuvchi interfeysi tartib faylidir. ADT plaginlari har bir yangi ilova uchun avtomatik ravishda bittasini yaratadi.

Android ilovalarini ishlab chiqish uchun XML bir qator afzalliklarga ega. XML engil, kodlash oson, va UI bilan bog'liq ma'lumotlarni tavsiflash uchun ishlatilishi mumkin. Ushbu maqola mobil ilovalar uchun XML dan foydalanish haqida qisqacha ma'lumot beradi. Androidni ishlab chiqish loyihangizda XML dan qanday foydalanishni ham o'rganishingiz mumkin. Uning afzalliklari quyida muhokama qilinadi. Agar siz darhol Android ilovalarini ishiab chiqishni bosh-lamoqchi bo'lsangiz, XML - bu eng yaxshi usul.

XML - dastlab ma'lumotlarni tashish va tartibga solish uchun mo'ljallangan belgilash tili. Uning dasturlashdan ko'ra ma'lumotlarga e'tibor qaratilishi uni ommabop kross-platforma standartiga aylantirdi. Garchi bu dasturlash tili bo'lmasa-da, XML ikkilik va matn almashish uchun ishonchli tanlovdir. XML fayllari odatda UTF-8 da kodlanganligini yodda tutish kerak, shuning uchun Android kabi resurs cheklangan platformaga XML qo'ymaganingizga ishonch hosil qiling.

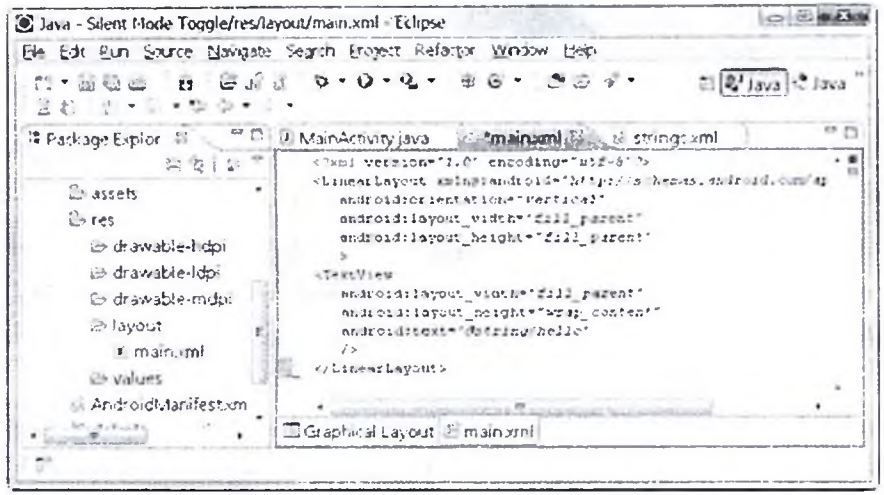
Android ilovalari uchun tartiblar XMLda yozilgan. Ular foydalanuvchi interfeysining tuzilishini aniqlaydi. Layout fayllari view deb nomlangan ildiz elementiga ega bo'lishi kerak. Ko'rinish o'rnatilgan ob'ektni ifodalaydi. View sinfining kichik sinfi ViewGroup deb ataladi. Ushbu ko'rinishlarni o'zlarining XML fayllari bilan birga guruhlash mumkin. ViewGroup barcha maketlar va ko'rish konteynerlari uchun asosiy sinfdir.

1-§. Eclipse muharriri

Tayanch so'z va atamalar

Eclipse, LinearLayout system, XML hujjati, Android SDK, plagin, manager.

Eclipse muharririning chap panelidagi nomini ikki marta bosish orqali main.xml faylini ochiladi (2.1.1-rasm). Faylning mazmuni o'ng panelda ko'rsatiladi.



2.1.1-rasm. main.xml faylining matni

1.1-rasmda ko'rsatilgan oddiy belgilash ekranning o'rtasida matn maydonini aks ettiruvchi teng darajadagi sodda foydalanuvchi interfeysini belgilaydi.

Quyida main.xml fayl kodi keltirib o'tamiz:

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

<LinearLayout xmlns:android=

"http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout_width="fill_parent"

android:layout_height="fill_parent"

>

<TextView

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:text="@string/hello"

/>

</LinearLayout>

XML faylining mazmuni foydalanuvchi interfeysining ko'rinishini o'ziga xos tarzda belgilaydi. Ushbu fayl butun rivojlanish kar'era uchun juda muhim, shuning uchun, belgilashning har bir elementini batafsil ko'rib chiqamiz.

XML hujjatini e'lon qilish. Birinchi element, XML hujjat deklaratsiyasi, Eclipse matn muharriri va Android operatsion tizimiga bu XML fayli ekanligini bildiradi va faylning XML versiyasi va kodlanishini belgilaydi.

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

Tartib turi. Keyingi element (**LinearLayout**) foydalanuvchi interfeysining chiziqli tartib turini belgilaydi. **LinearLayout** elementi quyida batafsilroq muhokama qilinadi, ammo hozircha u ekranda ko'rsatiladigan va "ko'rinishlar" deb ataladigan grafik interfeysning vizual elementlari uchun konteyner bo'lib xizmat qiladi. **LinearLayout** elementining atributlari ko'rinishlar qanday va qayerda joylashtirilishini aniqlaydi. Quyida **LinearLayoutni** ochish uchun kod keltirilgan.

Bu erda yopuvchi **</LinearLayout>** tegi ko'rsatilmagan, chunki LinearLayout elementi konteyner va yopish tegi konteynerga qo'shilgan har qanday ko'rinish elementlaridan keyin bo'ladi.

<LinearLayout xmlns:android=

"http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout_width="fill_parent"

android:layout_height="fill_parent"

>

2-§. Vakillik

Tayanch so'z va atamalar

TextView, LinearLayout, XML hujjati, RelativeLayout, FrameLayout, Android SDK manager.

Android platformasida ko'rinish foydalanuvchi interfeysining asosiy elementi hisoblanadi. Ko'rinishlar grafik foydalanuvchi interfeysini tashkil etuvchi qurilish bloklari hamdir. Har bir ko'rinish, bir tomondan, qandaydir sinfning namunasi (masalan, **TextView** klassi), ikkinchi tomondan, ekranda ko'rinadigan interfeys elementi hisoblanadi. **XML** belgilashda ko'rinish deskriptori ko'rinadigan foydalanuvchi interfeysi elementining vizual xususiyatlarini belgilaydi. Quyida **@string/hello** resursida saqlangan matn qatorini ko'rsatadigan **TextView** uchun belgi ko'rsatilgan:

```
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
```

Ko'rinish ekranda to'rtburchaklar maydonini egallaydi va interfeys elementining hodisalarini chizish va boshqarish uchun javobgar hisoblanadi. Mobil qurilma ekranida ko'rinadigan barcha vizual elementlar ko'rinish deyiladi. **View** klassi barcha standart Android vizuallarining asosiy (bosh) sinfi, shuning uchun ular **View** sinfidan maydonlar va metodlarni meros qillb oladi.

Tartib faylining oxirida boshqa elementlar yo'qligini ko'rsatadigan quyidagi yopuvchi **LinearLayout** element dastagi joylashgan.

```
</LinearLayout>
```

Joylashtirish turlari. Foydalanuvchi interfeysini yaratganingizda, uning vizual elementlarini qandaydir tarzda ekranga joylashtirishingiz kerak. Ularni joylashtirish metodi layout deb ataladi. Chiziqli tartib - bu ularni birin-ketin (vertikal yoki gorizontal) joylashtirishdir. Ba'zan elementlarni jadval kataklariga

joylashtirishingiz kerak, ba'zan esa har bir elementning koordinatalarini o'rnatishingiz kerak (bu metod aniq joylashishni aniqlash deb ataladi).

Yaxshiyamki, Android ishlab chiquvchilari biz uchun turli xil tartib turlaridan foydalanishimiz uchun ko'plab vositalarni yaratdilar. Har bir tartib turi ma'lum bir Java sinfiga mos keladi, bu esa o'z navbatida main.xml faylidagi ma'lum XML deskriptoriga mos keladi. 2.2.1-jadvalda Android SDK-da mavjud bo'lgan eng mashhur tartib turlari keltirilgan.

Sinf nomi	Tavsif
LinearLayout	*Ichki konteyner elementlari bir qatorda joylashtirilgan*
RelativeLayout	*Ichki elementlari bir-biriga yoki asosiy elementga nisbatan joylashtirilgan*
FrameLayout	*Ushbu konteyner bitta elementni ko'rsatish uchun ekranning bir qismini bloklaydi. Siz FrameLayout elementiga ko'plab yordamchi elementlarni qo'shishingiz mumkin, ammo ularning barchasi konteynerning yuqori chap burchagiga o'rnatiladi. To'liq elementlar belgilanishda paydo bo'lish tartibida chiziladi, shuning uchun oldingi elementlar keyingilari bilan qoplanadi.*
TableLayout	*Ichki elementlari jadvalning satr va ustunlariga joylashtiriladi*

2.2.1-jadval. Android SDK da mavjud tartib turlari

Yorliqlarni yaratuvchi **TabHost** yoki foydalanuvchi uni surib qo'yganda ko'rinishni yashiradigan yoki ko'rsatadigan **SlidingDrawer** (stol ustidagi maydalagichni tozalash uchun ishlatadigan harakatiga o'xshash) kabi boshqa tartib sinflari mavjud. Ko'pgina tartib sinflari faqat maxsus holatlarda qo'llaniladi. Haqiqiy bayotda ko'pincha 2.2-jadvalda ko'rsatilgan maketlar qo'llaniladi.

O'z-o'zini tekshirish uchun savollar va mashqlar

1. *Eclipse ish maydonida barcha ilova fayllari Android loyihasining qaysi papkasida saqlanadi?*
2. *Android ilovalarini ishlab chiqish uchun XML qanday afzalliklarga ega?*
3. *XML hujjatini e'lon qilishni tushuntiring.*
4. *grafik foydalanuvchi interfeysini tashkil etuvchi qurilish bloklarini keltiring.*
5. *Joylashtirish turlarini tushuntiring.*
6. *FrameLayout vazifasini tushuntiring.*

3-BOB. VIZUAL RIVOJLANISH MUHITI

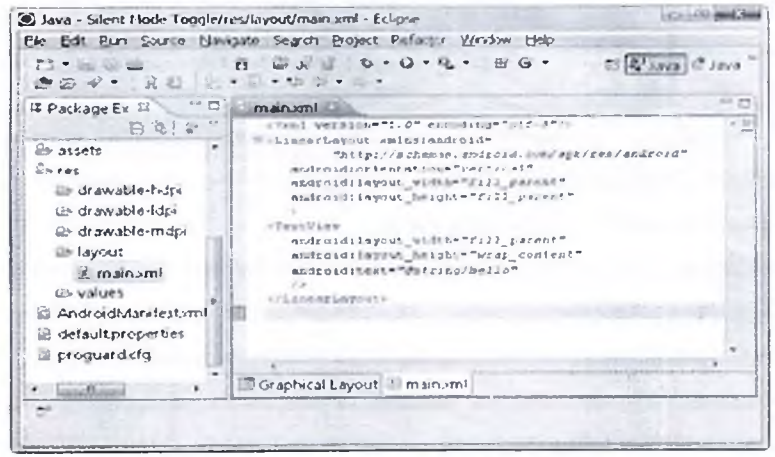
Sizlarga ikkita yangiligim bor: yaxshi va yomon. Yaxshi xabar shundaki, Eclipse vizual rivojlanish muhitiga ega (dizayner oynasi). Yomon xabar shundaki, vizual ishlab chiqish vositalari juda cheklangan, shuning uchun siz sichqonchani bosish bilan kuchli interfeys yarata olmaysiz. Vaqti-vaqti bilan kodlarni yozishingiz kerak bo'ladi. Biroq, keyinchalik bu unchalik yomon emasligini tushunasiz, chunki faqat XML belgilarida siz qaerga joylashtirilganligini aniq ko'rishingiz mumkin.

1-§. Dizayner oynasini ochish

Tayanch so'z va atamalar

Graphical Layout, TextView, dizayn, Properties, LinearLayout, RelativeLayout manager.

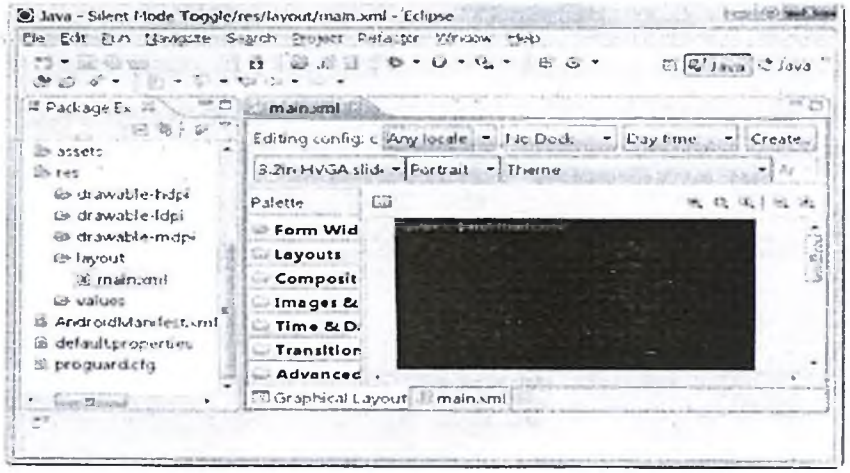
Dastur muharriri oynasida main.xml faylining kodi 3.1.1-rasmda ko'rsatilgan. Dizayner oynasida bir xil faylni ochish uchun ekranning pastki qismidagi Grafik tartib yorlig'ini bosiladi.



3.1.1-rasm. main.xml fayli kod muharriri rejimida

Graphical Layout yorlig'ini bosgandan so'ng, xuddi shu fayl konstruktor oynasida ko'rsatiladi (3.1.2-rasm).

Dizayn ko'rinishlar va konteynerlarni chapdagi asboblar panelidan o'ngdagi dastur oynasiga sudrab o'tish mumkin. Ilova oynasida hozirda bitta element, Hello, main.xml fayli iborasini aks ettiruvchi TextView mavjud. Buni muharrir oynasida ham ko'rish mumkin.



3.1.2-rasm. main.xml fayli dizayn rejimida

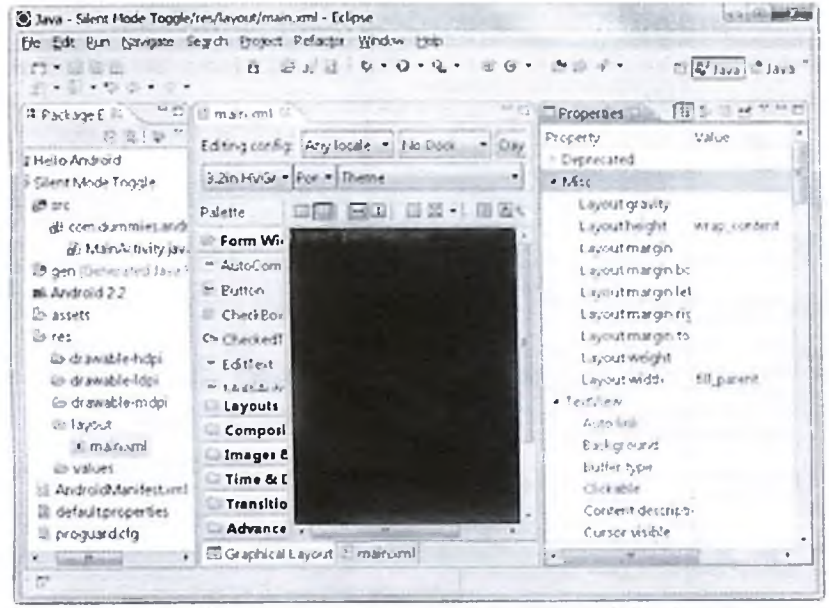
Ko'rinish xususiyatlarini tahrirlash. Dizayn ko'rinishida istalgan ko'rinishning xususiyatlarini ko'rish va tahrirlash mumkin. Buni amalga oshirish, uni tanlash uchun ustida tugma bosiladi. Tanlangan ko'rinishning xususiyatlari **Properties** (Xususiyatlar) panelida ko'rsatiladi. **Properties** paneli mavjud bo'lmasa, uni ochish uchun quyidagi amallarni bajariladi.

1. Ko'rinishni tanlash uchun ustiga bosiladi.

2. Ko'rinishni sichqonchaning o'ng tugmasi bilan bosib, yorliq menyusidan **Properties** ichida ko'rsatishni tanlanadi.

Ekranda xususiyatlar oynasi ko'rsatiladi (3.1.3-rasm). Chap ustunda xususiyat nomlari, o'ng ustunda esa ularning qiymatlari ko'rsatilgan. Qiymat maydonini bosganilganda, quyidagilardan biri sodir bo'ladi: kiritish kursori maydonga joylashtiriladi (keyin klaviaturadan kerakli qiymatni kiritishingiz mumkin) yoki ochiladigan ro'yxatda kerakli qiymatni tanlash uchun maydonning o'ng oxirida strelka paydo bo'ladi.

Xususiyatlar oynasi ko'pincha ko'rish xususiyatlari uchun mos yozuvlar turi sifatida foydalidir. Hech bir ishlab chiquvchi xususiyatlarning to'liq ro'yxatini yoddan eslay olmaydi. Xususiyatlar oynasida kerakli xususiyatni topish Android hujjatlariga qaraganda osonroq, chunki **Properties** oynasi har doim yonida ko'rinib turadi. **Properties** oynasini Windows - Show View – Other – General - Properties buyrug'ini tanlash orqali ochish mumkin.



3.1.3-rasm. Xususiyatlar oynasi (o'ngda) tanlangan ko'rinishning xususiyatlarini ko'rsatadi

Mavjud ko'rish xususiyatlari ro'yxati asosiy konteynerning joylashuv turiga qarab farq qilishi mumkin. Masalan, **LinearLayout** konteynerida **TextView** ko'rinishi xossalari ro'yxati **RelativeLayout** konteyneridagi kabi emas.

Dizayn ko'rinishi statik vazifalar uchun juda foydali bo'lib, unda konteynerning vizual tarkibi bir joyda mahkamlanadi va ko'rinishlar o'lchami o'zgartirilmaydi. Biroq, hisob-kitob yoki foydalanuvchi kiritish natijalariga ko'ra ko'rinishlarni boshqacha ko'rsatish kerak bo'lsa, Dizayn ko'rinishi unchalik foydali bo`lmaydi. Albatta, dizayner oynasida dinamik ko'rinishlarning boshlang'ich

pozitsiyalarini belgilash mumkin, ammo keyingi barcha o'zgarishlar va harakatlar kodda ko'rsatilishi kerak. Xuddi shu narsa ichki mazmunga ham tegishli bo`ladi. Masalan, **TextView** ekranga Dizayn ko'rinishida joylashtirilishi mumkin. Hatto main.xml belgisida boshlang'ich matnni belgilash mumkin, lekin faqat Java kodida ko'rsatilgan matnni o'zgartirish mumkin bo'ladi.

2-§. Foydalanuvchi interfeysini ishlab chiqish

Tayanch so'z va atamalar

Android, Version, Layout deskriptor, TextView, dizayn, Properties, LinearLayout, RelativeLayout manager.

Ekranda XML belgilarini o'z ichiga olgan **main.xml** fayli kodini ko'rsatamiz. Buning uchun 4.7-rasmda ko'rsatilganidek, Grafik tartibning o'ng tomonidagi ekranning pastki qismida joylashgan **main.xml** yorlig'ini bosiladi. Belgilashdan **TextView** elementini olib tashlanadi. Belgilash quyidagicha ko'rinishi kerak^

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

Layout deskriptor atributlari. XML fayli mos keladigan **LinearLayout** ko'rinishi va sinfini bilvosita belgilaydigan LinearLayout deskriptoridan foydalanadi. Ko'rish xususiyatlari XML deskriptorining atributlari bilan belgilanadi. Ushbu misolda foydalanilgan atributlar 3.2.1-jadvalda tasvirlangan.

Atribut	tavsifi
xmlns:android="..."	*XML nom maydoni ta'rifi Android SDK komponentlariga murojaat qilish uchun ishlatiladi*
android:orientation="vertical"	*Ushbu atribut konteynerga ko'rinishlar birin-ketin vertikal ravishda joylashtirilishi kerakligi haqida xabar beradi.*
android:layout_width="fill_parent"	*Bu atribut gorizontal holatda bu konteyner barcha mavjud bo'sh joyni to'ldirishi kerakligini bildiradi.*
android:layout_height="fill_parent"	*Bu atribut vertikal ravishda bu konteyner barcha mavjud bo'sh joylarni to'ldirishi kerakligini bildiradi.*

3.2.1-jadval. Layout atributlari

Shunday qilib, **LinearLayout** konteynerining joylashuv turi va parametrlari aniqlanadi. Konteyner butun ekranni gorizontal va vertikal ravishda to'ldiradi, faqat kerakli vakolatlarini joylashtirish qoladi.

3-§. Ko'rinishlarni konteynerga joylashtirish

Tayanch so'z va atamalar

GUI, layout_width, layout_height, LayoutParams, fill_parent, wrap_contents, piksel, Autosize.

Yuqorida aytib o'tilganidek, ko'rinishlar grafik foydalanuvchi interfeysini tashkil etuvchi qurilish bloklari hisoblanadi. **GUI** vizualini yaratganda, ko'rinishni yaratiladi. Java-da ko'rinishlar bilan ishlashda ular tegishli turlarga o'tkazilishi kerak.

layout_width va layout_height parametrlarini sozlash. Ekranda ko'rinish paydo bo'lishidan oldin uning xususiyatlarini Android operatsion tizimi uni qanday chizishni bilishi uchun sozlash kerak. Ko'rinishning xususiyatlari ko'rinish

deskriptorida berilgan atributlarning qiymatlari bilan belgilanadi. O'z navbatida **layout_width** (ширина компоновки) va **layout_height** (высота компоновки) atributlari talab qilinadi. Android SDKda ular **LayoutParams** sinfiga tegishli.

layout_width atributi ko'rinishning kengligini, **layout_height** atributi esa balandligini belgilaydi.

fill_parent va wrap_contents qiymatlari. Layout_width va **layout_height** atributlari piksel qiymatlarini yoki oldindan belgilangan qator qiymatlarini olishi mumkin. Eng keng tarqalgan ikkita oldindan belgilangan satr qiymatlari **fill_parent** va **wrap_content** hisoblanadi.

fill_parent qiymati Android operatsion tizimiga ketma-ketlik tartibida mavjud bo'lgan barcha bo'sh joyni to'ldirishi kerakligini bildiradi. **wrap_content**ga o'rnatilganda, Android o'z tarkibini ko'rsatish uchun yetarli joy ajratishi kerak.

Biroq, kontent miqdori oshganda (masalan, **TextView** elementi tomonidan ko'rsatilgan chiziq uzaytirilsa), ekrandagi egallagan joy ham kengayadi. **wrap_content** qiymati **Windows Formsdagi Autosize** xususiyatiga o'xshaydi.

Agar statik bog'lanish ishlatilsa, bu ikki atribut XML faylida o'rnatilishi mumkin. Agar tartib dinamik bo'lsa (ya'ni foydalanuvchi qurilma bilan ishlayotganda o'zgarishi kerak), tartib parametrlari Java kodida o'rnatilishi kerak, bu yerda ular atribut qiymatlari sifatida emas, balki sinf maydonlari va nomlari sifatida ifodalanadi. Sababi, maydonlar va atributlar mos kelmasligi mumkin. Har qanday holatda ham, havola parametrlariga qiymatlarni belgilashga ishonch hosil qilish kerak. Bu darsligimizda dinamik ko'rinishlar yoritilmagan, dinamik ko'rinishlarga misollarni Android SDK bilan ta'minlangan kod namunalarida topish mumkin.

Agar **layout_width** va **layout_height** atributlarini o'rnatishni unutib qo'ysangiz, dastur ko'rinishni ko'rsatishga urinayotganda ishdan chiqadi, lekin dasturni sinab ko'rishda buni aniqlash oson.

. Android yangi versiyalarida **fill_parent** nomi **match_parent** qilib o'zgartirilgan. Biroq, muvofiqligi uchun **fill_parent** hali ham qo'llab-quvvatlanadi, shuning uchun uni ushbu darslik davomida ishlatamiz. Ammo agar yuqori versiyalar uchun ilovalarni ishlab chiqishni rejalashtirmoqchi bo'linsa, **match_parent** qiymatidan foydalangan ma'qul.

4-§. Android ilovalarini ishlab chiqish asoslari

Tayanch so'z va atamalar

Android SDK, API, IntelliJ IDEA, NetBeans, Eclipseda Java Development Kit, System Image, Intel x86 Atom System Image.

Android SDK. Android SDK - bu Android platformasi uchun ilovalarni ishlab chiqishga mo'ljallangan asboblar to'plami. U dasturchini Android platformasini **API** bilan ta'minlovchi kutubxonalarni, ilovalar yaratish, virtual qurilma tasvirlarini boshqarish va boshqa amallarni bajarish uchun yordamchi dasturlarni o'z ichiga oladi. Android SDK barcha asosiy platformalar uchun bepul va uni Google saytidan yuklab olish mumkin: *http://developer.android.com/sdk/index.html.*

Android SDKda Java tili kompilyatori mavjud emas, shuning uchun Android ilovalarini kompilyatsiya qilish uchun Java Development Kit (JDK) ham talab qilinadi. Oracle JDK ning so'nggi versiyasini Oracle veb-saytidan yuklab olish mumkin

http://www.oracle.org.com/technetwork/java/javase/downloads/index.html.

Jarayonni soddalashtirish uchun integratsiyalashgan muhitlardan biri (Integrated Development Environment, IDE) ishlatilishi mumkin. Androidni ishlab chiqish barcha asosiy Java dasturlash muhitlarida, jumladan **IntelliJ IDEA, NetBeans** va **Eclipse**da qo'llab-quvvatlanadi. Shuni ta'kidlash kerakki, IDEning mavjudligi majburiy emas, chunki dasturni yaratish va joylashtirish uchun zarur bo'lgan barcha operatsiyalar buyruq qatori yordamida amalga oshirilishi mumkin.

Android SDK paket menejeri. Android SDKni qo'llab-quvvatlash turli kutubxonalar, shu jumladan ko'plab uchinchi tomon kutubxonalari yordamida Android platformasining barcha rasmiy versiyalari uchun ishlab chiqilgan. Android yordamchi dasturi ushbu kutubxonalar yordamida dastur yaratishni ta'minlaydigan paketlarni boshqarish uchun ishlatiladi. Ularni SDK katalogi ostidagi kichik kataiogda topish mumkin.

Android yordam dasturi buyruq qatori interfeysini hamda grafik foydalanuvchi interfeysini qo'llab-quvvatlaydi. Kutubxonalarni boshqarish uchun grafik interfeysga kirishdan foydalanish tavsiya etiladi.

Android yordam dasturini parametrlarsiz ishga tushirish orqali kirish mumkin. Ishga tushgandan so'ng, ekranda tizimda o'rnatilgan paketlarni ham, o'rnatish uchun mavjud paketlarni ham o'z ichiga olgan paketlar ro'yxati ko'rsatiladi. Kerakli paketlarni o'rnatish uchun ularni umumiy ro'yxatda tanlanadi va "O'rnatish" tugmasini bosiladi. Litsenziyani qabul qilgandan so'ng, tanlangan paketlar Internetdan yuklab olinadi va o'rnatiladi.

Ro'yxatdagi paketlar Android platformasi versiyasi bo'yicha guruhlangan. Platformaning har bir versiyasida ikkita raqamlash mavjud: "tijorat" va "ichki". Birinchi turdagi raqamlash platformaning imkoniyatlarini bozordagi qurilmalar tomonidan qo'llab-quvvatlanishini ko'rsatish uchun ishlatiladi. Ilovalarni ishlab chiqish jarayonida odatda ikkinchi raqamlash turidan foydalaniladi.

Ro'yxatdagi guruhlarning har biri quyidagi elementlarni o'z ichiga oladi: ushbu platforma uchun ilovalarni asosiy API to'plami (SDK platformasi), namunaviy ilovalar (SDK uchun namunalar), API hujjatlari (Android SDK uchun hujjatlar), platforma kutubxonalari uchun manba kodi (Android SDK uchun manbalar), turli arxitekturalar uchun virtual qurilma tasvirlari (ARM EABI v7a System Image, Intel x86 Atom System Image va boshqalar). Bundan tashqari, ro'yxatda uchinchi tomon kutubxonalari bo'lishi mumkin, shu jumladan ma'lum bir sinf qurilmalari uchun mo'ljallangan (masalan, Google TV Addon).

Takomillashtirishni boshlash uchun ma'lum bir platforma uchun kutubxonalarning asosiy to'plamini (SDK platformasi), virtual qurilma tasvirlarini, shuningdek API versiyasiga (Android SDK Tools) bog'lanmagan umumiy vositalar to'plamini o'rnatish kerak. Android SDK platformasi - guruhidagi instrumentlar). Qolgan paketlar ixtiyoriy.

5-§. Loyiha yaratish.

Tayanch so'z va atamalar

Target, name, path, activity, package, uses-feature. android.hardware.location, access carse location, application.

Avvalgi bo'limda tasvirlangan android yordam dasturi

yordamida yangi loyiha uchun shablon yaratish mumkin. Buning uchun buyruq qatori interfeysidan foydalaniladi, masalan:

*[1]android create project --target android-15 --name HelloWorld \ --path HelloWorld --activity MainActivity *

--package ru.ac.uniyar.helloworld

Berilgan buyruq tugmalarining maqsadi:

• **target** —yaratilayotgan platforma maqsad identifikatori. loyiha (barcha mavjud platformalar ro'yxati to'plam tomonidan belgilanadi);

• **name** — yaratilgan ilovaning nomi;

• **path** — yaratilayotgan loyiha katalogiga yo'l;

•**activity**– yaratilayotgan loyihaning asosiy faoliyati nomi (ilovaning asosiy ekraniga to'g'ri keladi);

•**package**– yaratilayotgan ilovaning o`zak paketining nomi. (yaratilgan ilovaning barcha sinflari ushbu paket ichiga joylashtiriladi).

Loylha tuzilishi. Oldingi paragrafdagi buyruqni bajarish natijasida joriy katalogda quyidagi fayllar va kataloglarni o'z ichiga olgan HelloWorld nomli loyiha katalogi yaratiladi:

• *AndroidManifest.xml* - Manifest fayli;

• *build.xml* - qurish fayli;

•*ant.properties, project.properties, local.properties*-konfiguratsiya fayllari;

• *src* — ilovaning dastlabki kodini o'z ichiga olgan katalog;

• *res* - resurs fayllarini o'z ichiga olgan katalog.

Manifest fayli. AndroidManifest.xml fayli Android ilovasi uchun asosiy konfiguratsiya faylidir. U o'zida ilovani tashkil etuvchi barcha komponentlarning ta'riflari, ular o'rtasidagi o'zaro ta'sir metodlarini, loyihani qurish shartlarini, turli resurslarga kirish huquqlarini va boshqalarni tavsiflaydi.

Eng oddiy manifest fayli, uning asosiy komponentlarini tavsiflaydi.

[1] PATH muhit o'zgaruvchisiga SDK ni o'z ichiga olgan katalogning asboblar va platforma-asboblar kichik kataloglariga yo'llarni kiritish tavsiya etiladi. Bu android yordam dasturini chaqiradi va boshqa buyruq qatori vositalari to'liq yo'lni ko'rsatmasdan - berilgan misolda xuddi shunday bajarilgan.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="ru.ac.uniyar.helloworld"
android:versionCode="1" android:versionName="1.0">
<uses-sdk android:minSdkVersion="15" />
<uses-feature android:name="android.hardware.location" />
<uses-permission
android:name="android.permission.ACCESS COARSE LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>
<application android:label="@string/app name"
android:icon="@drawable/ic launcher">
<activity android:name="MainActivity"
android:label="@string/app name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

Manifest faylining ildiz elementi *<manifest>* elementi bo`lib, u quyidagi atributlarga ega: **page** (ilova ildiz paketi nomi), **android:versionName** (foydalanuvchi ko'rishi mumkin bo'lgan ilova versiva raqami), **android: versionCode** (ilovaning ichki versiya raqami bolib, qurilmaga o'rnatilgan ilovalarni yangilashda foydalaniladi).

<uses-sdk> elementi SDK versiyalarini belgilaydi va loyihani qurish uchun foydalaniladi. Yuqoridagi misolda **android:minSdkVersion** atributi loyihani Android 4.0.3 ga mos keladigan SDK API 15 yoki undan yuqori versiyalari

yordamida qurish mumkinligini bildiradi. Agar SDKning noto'g'ri versiyasi bilan loyiha yaratishga harakat qilsangiz, xato xabarini olasiz.

<uses-feature> elementlari ilovaning to'g'ri ishlashi uchun maqsadli qurilma tomonidan qo'llab-quvvatlanishi kerak bo'lgan xususiyatlarni e'lon qilish uchun ishlatiladi. Masalan, **android.hardware.location** ilovaga geolocation imkoniyatlariga ega qurilma kerakligini bildiradi. **<uses-feature>** elementlarning mazmuni **Google Play**da ma'lum bir qurilma uchun mavjud bo'lgan ilovalar ro'yxatini filtrlash uchun ishlatiladi.

<uses-permission> elementlari Ilovaning to'g'ri ishlashi uchun qurilma xususiyatlariga kirish ruxsatlarini e'lon qilish uchun ishlatiladi. Masalan, **android. permission. Access Carse Location** ruxsati qurilma joylashuv funksiyalariga va *android.permission*ga kirish uchun so'raladi. *Internet* - Internetdan tarkibni yuklab olish uchun. Ilova manifest faylida ko'rsatilmagan ruxsatlarni talab qiluvchi xususiyatlarga kirishga harakat qilganda istisno qilinadi. Foydalanuvchi *Google Play*dan ilovani yuklab olganda, ro'yxatini ko'rib chiqishi va ilova so'ralgan funksiyalarga kirish huquqiga ega ekanligini tasdiqlashi kerak.

<application> elementi ilovani bir butun sifatida tavsiflaydi. Atributlar ushbu elementning **android:label** va **android:icon** qurilmadagi ilovalar ro'yxatida ko'rsatilgan nomini va uning belgisini aniqlaydi.

Ushbu atributlarning qiymatlari matn satrlari va dastur resurslariga havolalar bo'lishi mumkin.

<application> elementi ichida ilova komponentlarining ta'riflari mavjud. (Manifest faylida faoliyatni ro'yxatdan o'tkazish haqida qo'shimcha ma'lumot olish uchun avvalgi bo'limga qarang).

Loyiha yig'ilishi. Android loyihasi turli metodlar bilan tuzilishi mumkin. Eng oddiy holatda, JDKga kiritilgan va Java loyihalarini yaratish uchun standart vosita bo'lgan **ant** yordam dasturidan foydalaniladi.

"**ant**" qurish fayli odatda **build.xml** deb nomlanadi va loyihaning asosiy katalogida joylashgan. Agar avvalgi banddagi buyruq bilan loyiha yaratilsa, bu fayl ham yaratiladi.

Collection fayli maqsadlarni belgilaydi, ularning har biri qandaydir operatsiyalarga mos keladi (masalan, loyihani kompilyatsiya qilish, qurish, joylashtirish, sinovdan o'tkazish). Jarayon buyruqlardan iborat bo'lib, ularning bajarilishi belgilangan maqsadga erishishga olib keladi.

Maqsadlar o'rtasida bog'liqliklar o'rnatilishi mumkin, shu bilan birga ba'zi maqsadga erishish uchun zarur bo'lgan buyruqlar faqat bajarilgandan keyingina amalga oshadi.

Shu bilan birga, ba'zi buyruqlar ilgari bajarilgan bo'lsa va ularni takroran bajarish bir xil natijani beradigan bo'lsa, ularni moslashuvchan mexanizmi qo'llab-quvvatlanadi (masalan, agar ushbu modul avvalroq tuzilgan bo'lsa, modul kompilyatsiyasi qayta boshlanmaydi) va uning asl matni o'zgartirilmaydi.

Quyidagilar android yordam dasturi tomonidan avtomatik ravishda yaratilgan **ant** qurish faylining maqsadlari hisoblanadi:

• **help** — avtomatik ravishda yaratilgan yordamni ko'rsatadi;

• **debug** — dasturni sinovdan o'tkazish va diskni raskadirovka qilish uchun mo'ljallangan versiyada loyihani yig'ish;

• **release** — loyihani **Google Play**da chiqarish uchun mo'ljallangan loyiha yaratish (bu holda ilova montajdan keyin ishlab chiquvchi kaliti bilan imzolanadi):

• **install** - o'rnatilgan ilovalar paketini ishlaydigan emulyatorga yoki qurilmaga o'rnatadi. Bu faqat qurish maqsadlaridan biri bilan birgalikda ishlatilishi mumkin (disk raskadirovkasi yoki chiqarish) yoki o'rnatiladigan versiyani belgilanadi (o'rnatilgan yoki o'rnatuvchi);

• **clean** - loyihani tozalash, barcha fayllarni o'chirish, ilovaning tuzilgan bayt-kodi, shuningdek, oraliq qurish natijalari.

Shuni esda tutish kerakki, ko'rsatilgan maqsadlarning har qandayiga mos keladigan jarayon muayyan loyihaga xos bo'lgan harakatlarning bajarilishini

ta'minlash uchun o'zgartirilishi mumkin. Shuningdek, yangi maqsadlar qo'shish mumkin.

Loyihani yaratish va paket yaratish uchun quyidagi buyruqni bajariladi:

ant debug

agar qo'shimcha ravishda qurilgan paketni ishlaydigan emulyatorga yoki o'rnatishingiz kerak bo'lsa Android qurilmasi, keyin buyruq ishlatiladi

ant debug install

O'z-o'zini tekshirish uchun savollar va mashqlar

1. Android SDK nima? U qanday komponentlarni o'z ichiga oladi? Android platformasida ilovalarni ishlab chiqish uchun qanday vositalardan foydalanish mumkin?

2. Android paketlar menejeri nima? U qanday vazifalarni hal qiladi?

3. Avtomatik ravishda yaratilgan Android ilova loyihasining tuzilishi qanday? Qanday komponentlar yaratilgan va ular qanday kataloglarga joylashtirilgan?

4. Manifest fayl nima? Uning tuzilishi qanday? Manifest faylida qanday asosiy elementlarni topish mumkin va ular nima uchun?

*5. **Ant** nima? Ilovalarni yaratishda qanday foydalaniladi? Avtomatik ravishda yaratilgan montaj faylida qanday maqsadlar mavjud?*

4-BOB. AKTIVLIK VA REJA (NIYAT)LAR

Jarayonning aktivligi va imkoniyatlariga kirishdan oldin. Android ilovalarining dizayni haqida bir necha so'z aytish kerak, chunki ular ish stoli va boshqa mobil platformalar uchun ilovalar dizaynidan ancha farq qiladi.

Ushbu komponentlar bir-biridan mutlaqo mustaqil va boshqa komponentlar bilan o'zaro aloqa qilish uchun aniq belgilangan interfeyslarga ega. Android ilovalarining qiziqarli xususiyati shundaki, turli ilovalarning komponentlari bir-biri bilan o'zaro ta'sir qilishi mumkin.

1-§. Android ilovasi komponentlari.

Tayanch so'z va atamalar

Android, Asinxron, intent, service, activity, System Image.

Oddiy Android ilovasi komponentlardan iborat quyidagi turda bo'lishi mumkin:

- **jarayon** (activity) - foydalanuvchi bilan o'zaro aloqada bo'lgan komponent;
- **xizmat** (service) — fon jarayoni;
- **kontent provayderi** — ba'zi xotirada saqlangan ma'lumotlarga kirishni ta'minlovchi komponent;
- **translyatsiya xabarlari uchun foydalanuvchi** (broadcast receiver) — operatsion tizimdagi ba'zi global hodisalar uchun ishlov beruvchi (masalan, ekran o'chirilgan, batareya quvvati kam va hokazo).

Misol uchun, ilova elektron pochta xabarini yuborishi kerak bo'lganda, u elektron pochta mijozining funksionalligi bilan standart faoliyatni chaqirishi mumkin va bu jarayon tugagandan so'ng, foydalanuvchi asl ilova bilan ishlaydi. Aksincha, dasturchi o'zining pochta mijozini ishlab chiqishi va dasturni o'rnatishda uni tizimda ro'yxatdan o'tkazishi va shu bilan boshqa ilovalarga undan foydalanish imkonini berishi mumkin.

Bu xususiyat dastur tushunchasini biroz xiralashtiradi va butun platformani ochiq tizim sifatida ko'rib chiqishga majbur qiladi, uning tarkibiy qismlari hamkorlikda harakat qilishga qodir.

Komponentlar **asinxron** xabarlarni yuborish orqali o'zaro ta'sir qiladi. Ilovada ushbu xabarlarning har biri niyat (intent) deb ataladigan ob'ekt bilan bog'langan.

Интент. Android ilovalarida интент ba'zi amallarni bajarish uchun so'rovni qamrab oluvchi ob'ektdir. Интент quyidagi tarkibiy qismlarni o'z ichiga olishi mumkin:

- bajariladigan harakat (majburiy komponent);
- guruhlash imkonini beruvchi toifalar to'plami;
- ma'lumotlarni aniqlaydigan URI;
- amalni bajarish uchun zarur bo'lgan qo'shimcha parametrlar (extras).

Shuni ta'kidlash kerakki, интент, qoida tariqasida, yuborilayotgan xabarning manzilini aniq ko'rsatmaydi. Buning o'rniga u amalga oshiriladigan harakatni belgilaydi. Tizimning har bir komponenti o'zi bajarishga qodir bo'lgan maqsadlar to'plamini ro'yxatdan o'tkazadi. Tegishli интент faollashtirilganda, tizim ushbu harakatni bajarishga qodir komponentni topadi va unga boshqaruvni o'tkazadi. Agar bir nechta mos komponentlar mavjud bo'lsa, tanlov foydalanuvchiga qoldiriladi.

2-§. Manifest faylida jarayon deklaratsiyasi.

Tayanch so'z va atamalar

AndroidManifest, android, activity, intent, callback, OnCreate, onDestroy, OnPause, running, resumed, paused, stopped.

Har bir jarayon manifest faylida e'lon qilinadi. Bunday ta'rifni o'z ichiga olgan **AndroidManifest.xml** faylining bir qismi quyida ko'rsatib o'tiladi:

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="ru.ac.uniyar.helloworld"

```
android:versionCode="1" android:versionName="1.0">
...
<activity android:name="MainActivity"
android:label="@string/app name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.VIEW"></action>
<data android:scheme="http"></data>
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
...
</manifest>
```

Deklaratsiyaning asosiy elementi *<activity>* deb ataladi. Ushbu elementning *android:name* parametri jarayon nomini o'z ichiga oladi va (ilovaning ildiz paketi nomi bilan birga) *activity* kodini o'z ichiga olgan sinfni aniqlash uchun ishlatiladi.

Masalan, yuqoridagi misolda bunday sinf chaqiriladi *ru.ac.uniyar. helloworld.MainActivity* va *activity* boshlanganda yuklanadigan sinf *Android:label* parametri foydalanuvchiga ko'rsatiladigan sarlavhani o'z ichiga oladi.

<intent-filter> elementlari ushbu jarayon bajara oladigan maqsadlarni belgilaydi. Yuqoridagi misolda ikkita shunday niyet aniqlangan. Ulardan birinchisi ilovani barcha Android ilovalari ro'yxatida ko'rsatish va ushbu ro'yxatdagi faoliyatni mustaqil ravishda ishga tushirish imkoniyatini beradi. Bu *android.intent.action.MAIN* harakat turini va *android.intent.category.LAUNCHER* toifasini belgilash orqali amalga oshiriladi. Ikkinchi **<intent-filter>** elementi **http** ma'lumotlar turi uchun *android.intent.action.VIEW* amalini bajarish orqali standart veb-brauzer sifatida ishlashi mumkinligini bildiradi.

Yuqoridagi misoldan har qanday niyet komponentlari bajaradigan amallarni aniqlash uchun ishlatilishini ko'rish mumkin. Ushbu ta'rifni faollashtiradigan **manifest** faylining o'ziga xos elementlari haqida qo'shimcha ma'lumot olish uchun platforma **API** hujjatlariga qarash lozim.

Activity (faoliyat) doimiyligi. Android ilovasi arxitekturasida jarayon qisqa muddatli komponentlardir.

Ular turli vaziyatlarda, masalan, ekran yo'nalishi o'zgarganda yoki boshqa ilovalar uchun xotira yetarli bo'lmaganda avtomatik ravishda yaratilishi va yo'q qilinishi mumkin. Bundan tashqari, jarayon fonga yoki oldingi planga o'tishi, diqqatni jalb qilishi va yo'qotishi mumkin. Ushbu vaziyatlarning barchasini to'g'ri hal qilish uchun *faoliyatning doimiyligi* tushunchasi kiritiladi va ushbu tsiklning turli bosqichlari o'rtasida o'tish paytida muayyan harakatlarni amalga oshirishga imkon beradigan vositalar taqdim etiladi.

Jarayonning doimiyligi quyidagi asosiy holatlarni o'z ichiga oladi:

• **running/resumed** (yugurish/davom etish) - jarayon birinchi o'rinda va diqqat markazda. Bu holatda foydalanuvchi grafik interfeys orqali ilova bilan bevosita muloqot qilishi mumkin;

• **paused** (to'xtatilgan) - jarayon birinchi rejada, lekin diqqat markazida emas, ya'ni qalqib chiquvchi oyna yoki dialog oynasi bilan qoplangan. Foydalanuvchi bunday jarayon bilan bevosita muloqot qila olmaydi;

• **stopped** (to'xtatildi) — jarayon fonda va ekranda ko'rsatilmaydi, yani foydalanuvchi bunday jarayon bilan muloqot qila olmaydi.

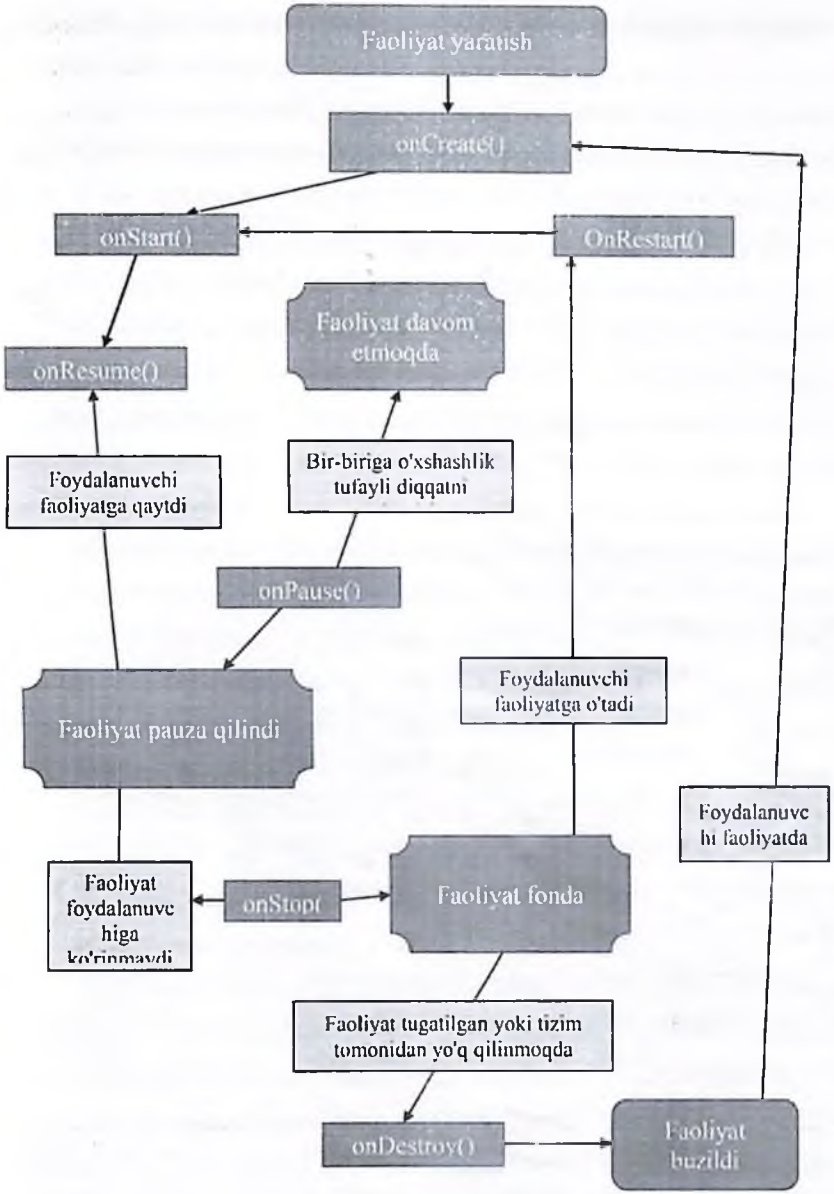
Ro'yxatga olingan holatlar o'rtasidagi o'tishlar foydalanuvchi tomonidan boshlangan hodisalar (masalan, boshqa faoliyatga o'tish) yoki tizim hodisalari (masalan, xotira etishmasligi tufayli) orqali amalga oshiriladi. Har bir o'tish **Activity** sinfidagi ma'lum metodlarga qo'ng'iroq bilan birga keladi, ulardan har qanday foydalanuvchi faoliyati meros bo'lishi kerak. O'z navbatida, avlod sinflarida bu metodlar sikl holatlari orasidagi o'tishga to'g'ri javob berish uchun bekor qilinishi mumkin. Tegishli qayta qo'ng'iroq qilish **callback** metodlari bilan

bir qatorda o'tishlar 2.1-rasmda ko'rsatilgan. Keling, aytib o'tilgan metodlarning eng muhimlarini batafsil ko'rib chiqaylik.

OnCreate() metodi yaratilgandan so'ng darhol chaqiriladi. Bu yerda foydalanuvchi interfeysi ishga tushiriladi, ma'lumotlar interfeys elementlariga bog'lanadi, oqimlar yaratiladi va hokazo. Bu metod **onDestroy()** metodi bilan bog'langan bo'lib, unda **onCreate()** ni chaqirish orqali olingan resurslarni chiqarish kerak bo'ladi.

OnPause() metodi qalqib chiquvchi oyna, dialog yoki boshqa jarayon bilan qoplanganligi sababli fokusni yo'qotganda chaqiriladi. Ushbu metod qaytgandan so'ng, jarayon to'xtatilgan holatga o'tadi. E'tibor bering, agar boshqa ustuvorroq bo'lgan ilovalar uchun operativ xotira yetarli bo'lmasa, ushbu holatdagi jarayon istalgan vaqtda operatsion tizim tomonidan yo'q qilinishi mumkin. Shu sababli, **onPause()** metodi dastur qayta ishga tushirilganda uni qayta tiklash uchun jarayon holatini saqlashi kerak. Holatni tiklash **onResume()** qayta qo'ng'iroq metodi callback orqali amalga oshiriladi.

Jarayonning doimiyligini muhokama qilish yakunida Android platformasining yana bir xususiyatiga e'tibor qaratamiz. Masalan, 17 ta ekran oynalarini o'z ichiga olgan dasturning tizim konfiguratsiyasini o'zgartirish yoki joylashuvni o'zgartirish aktivlikni yo'q qilishga va keyinchalik uni qayta yaratishga olib keladi. Bu barcha **callback**-metodlarini, jumladan, yo'q qilinayotgan jarayon uchun **onDestroy()** va yangi yaratilgan jarayon uchun **onStart()** ni bajaradi. Ushbu holatni ilova foydalanuvchi tomonidan yopilgan va keyin qayta ochilgan holatdan farqlash uchun **onSaveInstanceState()** va **onRestoreInstanceState()** juft metodlari taqdim etiladi. **Activity** sinfida ushbu metodlarni qo'llash barcha foydalanuvchi interfeysi elementlarining holatini avtomatik ravishda saqlaydi va tiklaydi. Biroq, ba'zi murakkab holatlarda, konfiguratsiya o'zgarganda dasturning to'g'ri ishlashini ta'minlash uchun ushbu metodlarni bekor qilinishi mumkin.



4.2.1-rasm. Android faoliyatining aktiv aylanishi

Activity metodini интент orqali chaqirish. Yuqorida aytib o'tilganidek, *activity*ga chaqirish интент orqali amalga oshiriladi. Kodda bu Intent sinfining namunasini yaratishni va keyin ushbu misolni *Context—Activity* суперкласс sinfida belgilangan *startActivity()* metodiga o'tkazishni talab qiladi.

Masalan:

Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://developer.android.com"));
startActivity(intent);
startActivity(intent);

Ikkinchi holda, manifest faylida aktivlikni e'lon qilishda *<intent-filter>* elementi ko'rsatilmaydi:

<activity android:name="OtherActivity"
android:label="@string/other activity name" />

3-§. Vazifalar va aktivlik to'plami.

Tayanch so'z va atamalar

manifest, activity, startActivityForResult, startActivity, extra, setResult, finish , Background activity.

Androiddagi vazifa - bu bir-biridan chaqiriladigan va bitta foydalanuvchi ehtiyojini qondirishga qaratilgan harakatlar to'plami. Qurilmada ishlaydigan barcha vazifalar ro'yxati foydalanuvchi " Home " tugmasini bosib ushlab turganda ko'rsatiladi.

Foydalanuvchi ilovani ishga tushirgandan so'ng, yangi vazifa yaratiladi va ishlayotgan ilovaning birinchi ochilgan faoliyati vazifaning faollik to'plamiga suriladi. Vazifaga nisbatan bu jarayon ichki jarayon deb ataladi. Vazifa ichki jarayon tugamaguncha tugatilmaydi.

Ichki jarayon ikkinchi faoliyatni chaqirishi mumkin, u joriy vazifa stekiga asosiy faoliyatning tepasiga suriladi. O'z navbatida, ikkinchi jarayon uchinchisini chaqirishi mumkin va hokazo. Bu harakatlarning barchasi stekga suriladi. Stekning yuqori qismidagi jarayon yopilganda (Android qurilmasidagi "Back" tugmasini

bosish yoki mavjud metodi orqali) u stekdan o'chiriladi va boshqaruv stekdagi faoliyatga o'tkaziladi.

"Home" tugmachasini bosganingizda, joriy jarayon fonga o'tadi, ammo tegishli vazifaning barcha to'plami saqlanadi. Agar siz hozir Bosh sahifa tugmasini bosib ushlab tursangiz, foydalanuvchi faol vazifalar ro'yxatini ko'radi. Bittasi tanlansa, stekning yuqori qismidagi jarayon aktiv bo'ladi va qolgan barcha harakatlar ularning ustidagi amallar yopilmaguncha stekda qoladi.

Shuni ta'kidlash kerakki, ma'lum bir vazifa stekiga kiritilgan harakatlar turli xil ilovalarga kiritilishi mumkin. Bu Androidning muhim tushunchalaridan birini ifodalaydi, foydalanuvchi ehtiyojlarini qondirish uchun bir nechta turli xil ilovalarning hamkorligini va ushbu hamkorlikni amalga oshirish uchun интент mexanizmini ta'minlaydi.

Intentdan ma'lumotlarni olish. Jarayonni boshlaydigan intent, shuningdek, ishga tushiruvchi va ishga tushirilayotgan jarayon o'rtasida vositachi hisoblanadi. Uni ichki sinfida getIntent() metodini chaqirish orqali olish mumkin. Olingan intentdan siz qo'ng'iroq qiluvchi tomonidan kiritilgan amal, toifalar, URI va qo'shimcha parametrlarni ajratib olishingiz mumkin.

Buning uchun quyidagi metodlar qo'llaniladi:

public String getAction();
public Set<String> getCategories();
public Uri getData();
public Bundle getExtras();

Ushbu xususiyatdan ushbu aktivlik tomonidan qayta ishlanishi kerak bo'lgan ma'lumotlarni olish uchun ham, bir jarayon bir nechta turli harakatlarni bajarishga qodir bo'lgan hollarda harakatlarni yuborish uchun ham foydalanish mumkin.

Aktivlik natijasini qaytarish. Ko'pincha ma'lumotni nafaqat ishga tushirish boshlanganiga, balki teskari yo'nalishda ham o'tkazish zarurati tug'iladi. Odatiy misol, asosiy jarayon uchun zarur bo'lgan ma'lumotlarni kiritish uchun fondagi faoliyatdan foydalaniladi. Bunday holda, orqa fon faoliyatini boshlash uchun

startActivity() metodidan emas, balki *startActivityForResult()* metodidan foydalanish kerak:

```
Intent intent = new Intent(this, EnterAddressActivity.class);
startActivityForResult(intent, 1);
```

Bu *EnterAddressActivity* deb nomlangan faoliyatni ishga tushirishga misol edi. Aytaylik, bu jarayon foydalanuvchidan manzilni so'raydi, tugma bosilganda chiqadi va kiritilgan qiymatni asosiy faoliyatga qaytaradi. Ushbu qaytarish qiymatini *EnterAddressActivity* sinfidagi tugma ishlovchisida amalga oshirish uchun quyidagi kod kerak bo'ladi:

```
String address = ...; // retrieve address from the text field
Intent intent = new Intent();
intent.putExtra("address", address);
setResult(RESULT_OK, intent);
finish();
```

Shunday qilib, "**address**" kaliti bilan qo'shimcha parametrda (extra) foydalanuvchidan olingan manzilni o'z ichiga olgan harakat va sinfni ko'rsatmasdan maxsus INTENT yaratiladi. Keyinchalik, *setResult()* ga qo'ng'iroq, faoliyatga qo'ng'iroq muvaffaqiyatli bo'lganligini bildiradi, shundan so'ng *finish()* metodini chaqirish orqali orqa fondagi faoliyati yopiladi.

Back() - aktiv faoliyatning bekor qilingan metodida amalga oshiriladi.

protected void onActivityResult(int requestCode, int resultCode, Intent data);

Background activity tomonidan o'rnatilgan qiymatlar natija kodi va ma'lumotlar parametrlari sifatida o'tkaziladi.

requestCode sifatida, *startActivityForResult()* metodi chaqiruvidagi identifikatsiya parametri (yuqoridagi 1-misolda) turli aktiv qo'ng'iroqlardan qaytishni farqlash uchun ishlatiladi.

O'z-o'zini tekshirish uchun savollar va mashqlar

1. *Android ilovalari qanday komponentlardan iborat bo'lishi mumkin?*
2. *Komponentlarning har birining maqsadi va xarakterli xususiyatlari nimada?*
3. *Intent nima? Android platformasidagi komponentlarning o'zaro ta'sirida intentlar qanday rol o'ynaydi?*
4. *Manifest faylida jarayon qanday e'lon qilinadi?*
5. *Manifest faylida jarayon qanday e'lon qilinadi? Ushbu e'londan maqsad nima?*
6. *Jarayon davomiyligi nima? Platformaning qanday xususiyatlari hayot sikli konseptsiyasini joriy etish zaruriyatini keltirib chiqaradi?*
7. *Ishlab chiquvchi faoliyatning qanday aktiv siklini qayta chaqirish metodlarini bekor qilishi mumkin?*
8. *Aktiv siklning qaysi bosqichida ular chaqiriladi? Ushbu qayta qo'ng'iroq qilish metodlarining har birining odatiy maqsadi nima?*
9. *Jarayonni intent orqali chaqirishning ikkita metodini ayting. Ularning orasidagi farq nima? Ushbu metodlarning har biri qachon qo'llaniladi?*
10. *Android atamasida "topshiriq" nima? Vazifalar jarayon bilan qanday bog'liq? Vazifalar foydalanuvchi nuqtai nazaridan qanday ko'rinadi?*
11. *Bir faoliyatdan ikkinchisiga ma'lumotlarni qanday olish mumkin? Activity qo'ng'irog'i natijasini qanday olish mumkin?*

5-BOB. MODEL VIEW CONTROLLER (MVC) ARXITEKTURASIDA ODDIY LOYIHALAR

Android ilovalari odatda model-view-controller (MVC) arxitekturasiga muvofiq ishlab chiqiladi. Ushbu arxitektura dasturning individual mas'uliyatini, kodni saqlashni va o'zgartirishni osonlashtiradigan tarzda ajratish imkonini beradi.

Arxitektura pattern larni qo'llash orqali android ilovalarni yaratish doimo dasturchilar tomonidan afzal ko'rilgan ishdir. Arxitektura pattern proyekt fayllariga modularlikni beradi va barcha kodlarga Unit Testing yozilishini ta'minlaydi. Bu dasturchilarga dasturiy ta'minotni saqlab qolish va kelajakda dastur xususiyatlarini kengaytirish vazifasini osonlashtiradi. Dasturchilar orasida juda mashhur bo'lgan ba'zi arxitekturalar mavjud va ulardan biri **Model — View — Controller (MVC) Pattern. MVC pattern** kodni 3 komponentga bo'lishni taklif qiladi. Dasturchi ilova klassini/faylini yaratishda uni quyidagi uchta qatlamdan biriga ajratishi kerak:

Model — bu komponenta dastur ma'lumotlarini o'zida saqlaydi. U interfeys haqida hech qanday ma'lumotga ega emas. Model domen logikasi(haqiqiy biznes qoidalari), ma'lumotlar bazasi va tarmoq qatlamlari bilan aloqa qilish uchun javobgardir.

View — bu ekranda ko'rinadigan barcha elementlarni o'zida saqlaydigan UI (foydalanuvchi interfeys) qatlami. Bundan tashqari, u Modelda saqlangan ma'lumotlarning vizualizatsiyasini ta'minlaydi va foydalanuvchiga o'zaro aloqani taklif qiladi.

Controller — bu komponenta **View** va **Model** o'rtasidagi munosabatni o'rnatadi. U dasturning asosiy logikasini o'z ichiga oladi va foydalanuvchining xatti harakati to'g'risida ma'lumot oladi va ehtiyojga qarab modelni yangilaydi.

1-§. Loyiha yaratish.

Tayanch so'z va atamalar

resetButton, Recovery, Button, android:layout, padding, LinearLayout, activity, finish.

Ushbu bo'limdan boshlab, MVC arxitekturasi asosida oddiy Android ilovasini yaratish jarayoni bosqichma-bosqich tasvirlanadi. Bu misol, bir tomondan, ishlab chiqish uchun ishlatiladigan standart vositalar haqida qisqacha ma'lumot beradi. Boshqa tomondan, Android ilovalari (masalan, resurslar va jarayon sikli, hodisalar, ishlov beruvchilari) Android platformasida MVC dizayn namunasini namoyish etadi.

Ishlab chiqilayotgan dastur juda oddiy. U "Hisoblagich" deb ataladi va hisoblagich qiymatini mos ravishda bittaga oshirish va uni nolga qaytarish uchun ikkita tugmani o'z ichiga olgan bitta ekrandan iborat bo'ladi.

5.1.1-rasm. Ilovaning ko'rinishi.

Loyiha 5.1.1-rasmda tavsiflanganidek, buyruq qatori yoki har qanday integratsiyalashgan muhitidan foydalangan holda yaratilishi mumkin.

Aniqlik uchun loyihaning ildiz paketi *ru.uniyar.ac.counter* deb nomlanadi (bunda barcha dastur sinflari joylashtiriladi) va asosiy *activity* sinfi *MainActivity* deb nomlanadi. Malumot uchun, quyida yaratilgan loyihaning manifest faylini keltirib o'tamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="ru.ac.uniyar.counter"
android:versionCode="1"
android:versionName="1.0">
<application android:label="@string/app name"
android:icon="@drawable/ic launcher">
<activity android:name="MainActivity"
android:label="@string/app name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

Foydalanuvchi interfeysini yaratish. Android platformasida ilovaning foydalanuvchi interfeysi ikki metodda yaratilishi mumkin: deklarativ ravishda XML resurs faylida va majburiy ravishda kodda UI yaratish va joylashtirish orqali. Amalda, birinchi metod deyarli har doim qo'llaniladi, chunki u qulayroq va vizual interfeys yaratish vositalari bilan qo'llab-quvvatlanadi. Bizning misolimizda interfeys tavsif fayli bo'yicha main.xml nomi ostida yaratilgan va loyiha katalogining **res/layout** kichik katalogida joylashgan. Hisoblagich qiymatini ko'rsatish uchun yorliq va hisoblagich qiymatini bittaga oshirish va uni qayta o'rnatish uchun ikkita tugmani qo'shish orqali uni tahrirlash kerak. Tahrirlashdan so'ng ushbu fayl quyidagicha bo`ladi:

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



```
android:orientation="vertical"
android:layout width="fill parent"
android:layout height="fill parent"
android:padding="5dp">
<TextView
android:layout width="wrap content"
android:layout height="wrap content"
android:text="0"
android:id="@+id/counterText" android:layout gravity="center"
android:textSize="48dp"/>
<LinearLayout
android:orientation="horizontal"
android:layout width="fill parent"
android:layout height="fill parent" android:padding "5dp">
<Button
android:layout width="0dp"
android:layout weight="1"
android:layout height="wrap content"
android:text="+1"
android:id="@+id/increaseButton"/>
<Button
android:layout width="0dp"
android:layout weight="1"
android:layout height="wrap content"
android:text="Reset"
android:id="@+id/resetButton"/>
</LinearLayout>
</LinearLayout>
```

Yuqoridagi misolda XML hujjatining elementi <LinearLayout> elementidir. Bu

konteyner bo'lib,` ichki elementlarni qator yoki ustunga joylashtirish uchun mo'ljallangan. Bunday holda, *android:orientation* xususiyati vertikal holatga o'rnatilganda, ichki o'rnatilgan *<TextView>* va *<LinearLayout>* elementlari 5 dp[2] to'ldirish bilan ustunga joylashtiriladi (to'ldirish miqdori android qiymati bilan belgilanadi: *padding* atributi).

Ichki *<TextView>* elementi joriy hisoblagich qiymatini aks ettiruvchi matn maydonidir. Elementning xususiyatlariga ko'ra, markazlashtirilgan matnning o'lchami 48 dp va boshlang'ich qiymati "0" dir. Bundan tashqari, matn maydonida kontratext identifikatori mavjud bo'lib, bu maydonga dastur kodidan kirishni ta'minlash uchun zarur.

O'rnatilgan *<LinearLayout>* elementi o'z elementlarini gorizontal ravishda bir xil 5 dp chekinish bilan joylashtiradi. "+1" va "**Recovery**" tugmalari mos ravishda oshirish **Button** va **resetButton** identifikatorlari konteyner elementlari sifatida ishlaydi. Tugmalar konteynerning barcha bo'sh gorizontal maydonini 1:1 nisbatda taqsimlaydi, bu ularning **android:layout** vazn atributining qiymati bilan belgilanadi. Ushbu atributdan foydalanilganda, **android:layout width** atributida aniq o'rnatilgan tugmalar kengligi ishlatilmaydi.

Yuqoridagi misol XML resurs elementlarning joylashuvi qanday tasvirlanganligi haqida fikr beradi. Ushbu mavzu bo'yicha qo'shimcha ma'lumotni boshqa yordamchi ma`lumotlarda ham toppish mumkin. Bunday resurs odatda vizual foydalanuvchi interfeysi yaratuvchisi tomomdan yaratilgan bo'lsa-da, xatolarni muvaffaqiyatli topish va murakkab interfeyslarni samarali yaratish uchun ularning tuzilishini XML belgilash darajasida tushunish foydalidir.

[2] dp (density-independent pixel) - maqsadli qurilma ekranidagi nuqta zichligiga bog'liq bo'lmagan o'lchov birligi. Bunday birliklar turli o'lchamdagi ekranlardan foydalanganda interfeys elementlari orasidagi nisbat buzilmasligi uchun ishlatiladi.

2-§. XML faylidan foydalanuvchi interfeysini yuklash va uning komponentlariga kirish.

Tayanch so'z va atamalar

onCreate, layout, Resurs, identifikator, findViewById,MainActivity, onIncreaseButtonClick, deklarativ, public.

Foydalanuvchi interfeysmi tavsiflovchi XML fayli yaratilgandan so'ng, u dastur kodidan yuklanishi kerak. Bu odatda onCreate() sinfida amalga oshiriladi:

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

Ushbu metodning birinchi qatori superklassning **onCreate()** sinfini chaqiradi, bu esa barcha siklni qayta murojaat qilish metodlari uchun zarur bo'lgan amaldir. Ikkinchi qatorda faoliyatning UI tavsifi **layout/main.xml** resurs faylidan yuklanishi kerakligini bildiradi. Resurs faylining joylashuvi va nomi maxsus identifikator (bu holda R.layout.main) yordamida belgilanadi, u avtomatik ravishda hosil qilingan R sinfidan konstanta sifatida olinadi. Bu klass har gal qaytadan yaratiladi. Bundan tashqari, onCreate() metodida, qoida tariqasida, interfeys elementlarining Java ob'ektlariga havolalar so'raladi, keyinchalik ularga dastur kodidan kirish kerak bo'ladi. Ushbu havolalar **findViewById()** sinfini chaqirish orqali olinadi va odatda aktiv sinf maydonlarida saqlanadi.

Bularning barchasi bilan **MainActivity.java** fayli quyidagicha ko'rinishi mumkin:

```
public class MainActivity extends Activity {
private TextView counterText;
private Button increaseButton, resetButton;
@Override
```

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
counterText = (TextView) findViewById(R.id.counterText);
increaseButton = (Button) findViewById(R.id.increaseButton);
resetButton = (Button) findViewById(R.id.resetButton);
}
}
```

XML faylining tegishli elementlarining **android:id** xususiyatining qiymatlari sifatida identifikatorlar qaysi interfeys elementlarini olish kerakligini aniqlash uchun foydalanililadi.

Foydalanuvchi interfeysi elementi hodisalarini boshqarish. Ushbu misolda "+1" va "Reinstall" ikkita foydalanuvchi interfeysi tugmalaridan hodisalarni boshqarish va ushbu tugmalarni bosishga javoban harakatlarni bajarish kerak. Ushbu muammoni hal qilishning ikki yo'li mavjud:

Birinchi metod to'g'ridan-to'g'ri koddan tugma ishlov beruvchisini qo'shishdir. Buning uchun quyidagi amallarni bajarish kerak:

1) **Button.OnClickListener** foydalanuvchi interfeysini amalga oshiruvchi sinfni aniqlanadi;

2) ushbu interfeys tomonidan aniqlangan **onClick()** metodini amalga oshiriladi;

3) foydalanuvchilar sinfi ob'ektini yaratib, tugmani bosish moslamasi sifatida o'rnatiladi.

Odatda, ichki o'rnatilgan anonim sinf ishlov beruvchi vazifasini bajaradi. Uning ishlatilishi koddagi barcha uchta amalni eng ixcham tarzda amalga oshirish imkonini beradi. Hodisani boshqarish uchun anonim sinfdan foydalanishni ko'ramiz - "+1" tugmasini bosamiz:

```
@Override
public void onCreate(Bundle savedInstanceState) {
```



```
// ...
increaseButton.setOnClickListener(new Button.OnClickListener() {
@Override
public void onClick(View v) {
onIncreaseButtonClick(v);
}
});
}
private void onIncreaseButtonClick(View v) {
// handler code here
}
```

O'rnatilgan sinflarning sintaksisi ancha og'ir bo'lgani uchun, haqiqiy voqealarni qayta ishlash kodi **MainActivity** sinfining alohida **onIncreaseButtonClick()** metodiga o'tkaziladi. Ushbu yondashuv kodni o'qilishi mumkin bo`lgani uchun tavsiya etiladi.

Hodisani boshqarishning ikkinchi metodi **deklarativdir**. Bu **MainActivity** sinfidagi hodisalarni qayta ishlash metodining nomi oddiygina main.xml faylidagi **<Button>** elementining tegishli atributida ko'rsatiladi:

```
<Button
android:layout width="0dp" android:layout weight="1"
android:layout height="wrap content" android:text="+1"
android:id="@+id/increaseButton"
android:onClick="onIncreaseButtonClick" />
```

Shuni ta'kidlash kerakki, bu holda *MainActivity* sinfidagi ishlov beruvchi klass **public** (ommaviy) deb e'lon qilinishi va **Button.Listener** sinfidagi mos ishlov beruvchi klass bilan bir xil argumentlarni olishi kerak:

```
public void onIncreaseButtonClick(View v) {
// handler code here
}
```

Ushbu qayta ishlash sinfi oddiyroq, u **onCreate()** sinfi kodiga o'zgartirish kiritishni talab qilmaydi, ammo uning kichik kamchiliklari bor: ishlov berish sinfini aniqlashda xatolik yuz berganda (nomdagi xato, noto'g'ri argument turlari va boshqalar), bu xato faqat dasturni bajarish paytida aniqlanadi. Birinchi metodda bunday kamchilik yo'q va uni ishlatishda bunday turdagi barcha xatolar kompilyatsiya bosqichida allaqachon aniqlanadi.

3-§. Hisoblagich modeli.

Tayanch so'z va atamalar

updateCounterView, Recovery, MainActivity, faol model, onModification, kontroller.

Muammoni hal qilish uchun MVC arxitekturasiga muvofiq, ma'lumotlarni saqlaydigan va dasturning biznes mantig'ini o'z ichiga olgan model darajasidagi sinfni aniqlash kerak, bu holda hisoblagichni ko'paytirish va tiklash operatsiyalarini bajarishga to'g'ri keladi. Eng oddiy holatda, model quyidagicha ko'rinishi mumkin:

```
package ru.ac.uniyar.counter;
public class Counter {
private int value = 0;
public int getValue() { return value; }
public void increase() { value++; }
public void reset() { value = 0; }
}
```

Modelni kontrollerga joylashtirish. Oldingi paragrafda keltirilgan model passiv bo'lgani uchun (ya'ni, u ko'rinishga va hoshqaruvchiga o'zgarishlari haqida gapira olmaydi), MVCning ushbu versiyasida model o'zgarganda ko'rinishni yangilash uchun boshqaruvchi javobgardir. Ushbu misolda *MainActivity* nazoratchi vazifasini bajaradi va kontratext matn maydoni ko'rinish sifatida ishlaydi.

Birinchidan, hisoblagich maydonini MainActivity sinfiga qo'shamiz va uni ishga tushiramiz:

```
public class MainActivity extends Activity {
// ...
private Counter counter = new Counter();
// ...
}
```

Keyinchalik, main.xml faylida ikkita tugmani bosish bilan ishlov berish metodlarining nomlarini belgilaymiz:

```
<Button android:id="@+id/increaseButton"
android:layout width="0dp" android:layout weight="1"
android:layout height="wrap content" android:text="+1"
android:onClick="onIncreaseButtonClick" />
<Button android:id="@+id/resetButton"
android:layout width="0dp" android:layout weight="1"
android:layout height="wrap content" android:text="Reset"
android:onClick="onResetButtonClick" />
```

Nihoyat, ushbu metodlarning **MainActivity** sinfda qo'llanilishini ko'ramiz:

```
public void onIncreaseButtonClick(View v) {
counter.increase();
counterText.setText(String.valueOf(counter.getValue()));
}
public void onResetButtonClick(View v) {
counter.reset();
counterText.setText(String.valueOf(counter.getValue()));
}
```

Yuqoridagi ma'lumotlarni kerakli tarzda o'zgartirish (hisoblagich qiymatini oshirish yoki uni qayta tiklash), shuningdek joriy qiymatni olish uchun modelni chaqiradi. Oxirgi qadam ko'rinishni yangilash hisoblanadi.

Ushbu qayta ishlash sinfi oddiyroq, u **onCreate()** sinfi kodiga o'zgartirish kiritishni talab qilmaydi, ammo uning kichik kamchiliklari bor: ishlov berish sinfini aniqlashda xatolik yuz berganda (nomdagi xato, noto'g'ri argument turlari va boshqalar), bu xato faqat dasturni bajarish paytida aniqlanadi. Birinchi metodda bunday kamchilik yo'q va uni ishlatishda bunday turdagi barcha xatolar kompilyatsiya bosqichida allaqachon aniqlanadi.

3-§. Hisoblagich modeli.

Tayanch so'z va atamalar

updateCounterView, Recovery, MainActivity, faol model, onModification, kontroller.

Muammoni hal qilish uchun MVC arxitekturasiga muvofiq, ma'lumotlarni saqlaydigan va dasturning biznes mantig'ini o'z ichiga olgan model darajasidagi sinfni aniqlash kerak, bu holda hisoblagichni ko'paytirish va tiklash operatsiyalarini bajarishga to'g'ri keladi. Eng oddiy holatda, model quyidagicha ko'rinishi mumkin:

```
package ru.ac.uniyar.counter;
public class Counter {
private int value = 0;
public int getValue() { return value; }
public void increase() { value++; }
public void reset() { value = 0; }
}
```

Modelni kontrollerga joylashtirish. Oldingi paragrafda keltirilgan model passiv bo'lgani uchun (ya'ni, u ko'rinishga va boshqaruvchiga o'zgarishlari haqida gapira olmaydi), MVCning ushbu versiyasida model o'zgarganda ko'rinishni yangilash uchun boshqaruvchi javobgardir. Ushbu misolda *MainActivity* nazoratchi vazifasini bajaradi va kontratext matn maydoni ko'rinish sifatida ishlaydi.

Birinchidan, hisoblagich maydonini MainActivity sinfiga qo'shamiz va uni ishga tushiramiz:

```
public class MainActivity extends Activity {
// ...
private Counter counter = new Counter();
// ...
}
```

Keyinchalik, main.xml faylida ikkita tugmani bosish bilan ishlov berish metodlarining nomlarini belgilaymiz:

```
<Button android:id="@+id/increaseButton"
android:layout width="0dp" android:layout weight="1"
android:layout height="wrap content" android:text="+1"
android:onClick="onIncreaseButtonClick" />
<Button android:id="@+id/resetButton"
android:layout width="0dp" android:layout weight="1"
android:layout height="wrap content" android:text="Reset"
android:onClick="onResetButtonClick" />
```

Nihoyat, ushbu metodlarning **MainActivity** sinfda qo'llanilishini ko'ramiz:

```
public void onIncreaseButtonClick(View v) {
counter.increase();
counterText.setText(String.valueOf(counter.getValue()));
}
public void onResetButtonClick(View v) {
counter.reset();
counterText.setText(String.valueOf(counter.getValue()));
}
```

Yuqoridagi ma'lumotlarni kerakli tarzda o'zgartirish (hisoblagich qiymatini oshirish yoki uni qayta tiklash), shuningdek joriy qiymatni olish uchun modelni chaqiradi. Oxirgi qadam ko'rinishni yangilash hisoblanadi.

Hisoblagich ilovasi tayyor. Kompilyatsiya qilish, qurish va joylashtirishdan so'ng, emulyator yoki Android qurilmasi 1.1-rasmda ko'rsatilgandek ekranni ko'rsatishi kerak.

Aktiv model. Oldingi bo'limlarda tasvirlangan hisoblagich ilovasini nazoratchidagi barcha o'zgarishlarini kuzatib borish va ko'rinishni yangilash kerak bo'lgan kamchiliklarga ega. Murakkab ilovalarda bu jiddiy muammoga aylanishi mumkin, natijada boshqaruvchi kodini sezilarli darajada ortiqcha yuklaydi. Uni hal qilish uchun passiv o'rniga faol modeldan foydalanish mumkin, ya'ni o'zgarishlari haqida foydalanuvchilarni xabardor qilishi lozim bo`ladi.

Faol modeldagi bildirishlarni amalga oshirish Android UI komponentlari jarayonga ishlov beruvchilarini amalga oshirishga o'xshaydi. Faol model kodini ko'rib chiqamiz:

```
package ru.ac.uniyar.counter;
public class Counter {
public interface OnModificationListener {
void onModification(Counter sender);
}
private int value = 0;
private OnModificationListener listener = null;
public void setOnModificationListener(OnModificationListener listener) {
this.listener = listener;
}
public int getValue() {
return value;
}
public void increase() {
value++;
if (listener != null) { listener.onModification(this); }
}
```



```
public void reset() {
value = 0;
if (listener != null) { listener.onModification(this); }
}
}
```

Model sinfi ichida maxsus **Counter.OnModificationListener** belgilangan. Ushbu interfeys foydalanuvchi ob'ekti tomonidan amalga oshirilishi kerak (bunday ob'ekt MVC arxitekturasida ko'rinish yoki kontroller bilan bog'liq bo'lgan sinf bo'lishi mumkin). Ushbu ob'ekt foydalanuvchi sifatida ro'yxatdan o'tkazish uchun modelning **setOnModificationListener()** metodini ham chaqirishi kerak. Ro'yxatdan o'tish vaqtida model o'tkazilgan havolani **onModificationListener** maydonida saqlaydi.

Har safar maydonning qiymati o'zgarganda, model ob'ektida **onModification()** interfeys metodini chaqiradi va foydalanuvchi javob berishi mumkin bo'lgan xabar yuboradi, masalan, ekrandagi tasvirni qayta chizish.

Faol modeldan foydalanish uchun sinfini o'zgartirish. Faol modeldan foydalanganda, aktivlik sinfi kodiga uchta o'zgartirish kiritish kerak:

- modelni ishlovchisini ro'yxatdan o'tkazish
- kodni amalga oshirish
- model o'zgarganda ko'rinish yangilanishini olib tashlanadi.

Ichki sinf sifatida model sinfining mos keladigan metodiga o'tkazamiz:

```
@Override
public void onCreate(Bundle savedInstanceState)
{
// ...
counter.setOnModificationListener(new Counter.OnModificationListener() {
@Override
public void onModification(Counter sender) {
updateCounterView();
```

```
}
});
}
```

Ko'rinib turibdiki, qo'llaniladigan yondashuv avvalgi bo'limdagi voqealarni qayta ishlashning birinchi metodi bilan to'liq mos keladi, hodisani qayta ishlash **MainActivity** sinfining **updateCounterView()** metodiga o'tadi. Ushbu metodni amalga oshirish quyidagicha ko'rinadi:

```
public void updateCounterView() {
counterText.setText(String.valueOf(counter.getValue()));
}
```

"+1" va "**Recovery**" tugmachalarini foydalanuvchilardan olib tashlash qoladi, chunki yangilash endi model o'zgarganda avtomatik ravishda chaqiriladigan **updateCounterView()** metodi bilan amalga oshiriladi.

Keraksiz kodni olib tashlaganinganidan so'ng, yuqoridagi ishlov beruvchilar quyidagicha ko'rinadi:

```
public void onIncreaseButtonClick(View v) {
counter.increase();
}
public void onResetButtonClick(View v) {
counter.reset();
}
```

Bu yerda passiv modelni faol modelga almashtirishni yakunlandi. Ushbu paragrafda o'rnatilgan dastur passiv modelli dastur bilan bir xil ishlashiga ishonch hosil qilish mumkin.

4-§. Faol va passiv modellarning afzalliklari va kamchiliklari.

Tayanch so'z va atamalar

lokalizatsiya, arxitektura, model, onCreate, Passiv model, faol model, MVC kontroller.

Biz bir xil modelning ikki xil amalga oshirilishini solishtiramiz va uning asosida ushbu modellardan foydalanishning afzalliklari va kamchiliklarini shakllantirishga harakat qilamiz, shuningdek, qaysi hollarda ulardan biriga ustunlik berish kerakligini aniqlaymiz.

Passiv (3.6-sek.) va faol (3.8-sek.) modellarning sinflarini hisobga olsak, faol model passiv modelga qaraganda sezilarli darajada murakkabroq ekanligini ko'rish oson. qo'shimcha ravishda foydalanuvchiga havolani saqlash uchun maydon, ushbu maydonni o'rnatish metodi va model o'zgarishining barcha holatlarida kod foydalanuvchisini ham ko'rib otiladi. Bundan tashqari, faol modeldan foydalanganda faollik sinfining **onCreate()** metodida passiv modelda mavjud bo'lmagan foydalanuvchini ro'yxatga olish kodi kerak bo'ladi.

Biroq, yuqori murakkabligiga qaramay, faol model ham muhim afzalliklarga ega: undan foydalanish tufayli model o'zgarganda ko'rinishni yangilashning hojati yo'q. Model o'zgarishlari kodda lokalizatsiya qilinmaganda, bu xususiyat qimmatli bo'ladi. Misol sifatida, foydalanuvchiga tarmoq orqali qandaydir o'yin o'ynash imkonini beruvchi dasturni ko'rib chiqish mumkin. Modeldagi o'zgarishlar (o'yin maydoni) grafik foydalanuvchi interfeysi orqali yoki tarmoq o'zaro ta'siri moduli orqali sodir bo'lishi mumkin. Agar ushbu vaziyatda faol model ishlatilmasa, tarmoq aloqasi xaritalashni amalga oshiradigan kodga bog'liq bo'lishi mumkin va bu MVC arxitektura talablarini buzishi mumkin. Faol modeldan foydalanganda modelni o'zgartirish kodi ko'rinishni yangilash kodidan ajratiladi va o'rnatish kontrollerda amalga oshiriladi. Aniq ajratilishi tufayli dasturni kontseptual jihatdan yanada tushunarli qiladi. Yuqoridagilardan kelib chiqib, model turini tanlash bo'yicha quyidagi fikrlarni shakllantirisli mumkin.

Agar model o'zgarishlari kodda lokalizatsiya qilingan bo'lsa va katta hajmni tashkil qilmasa, unda passiv model etarli bo'lishi mumkin. Aks holda, faol modeldan foydalanish kerak va foydalanuvchilar bilan ishlash uchun kod yozish qo'shimcha kodning oshkoraligi va yuqori ishonchliligi bilan qoplanadi.

5-§. Ekran yo'nalishini o'zgartirish bilan ishlash.

Tayanch so'z va atamalar

Activity, protected, onSaveInstanceState, Bundle, Counter, MainActivity, konstruktor arxitektura, model.

Ishlab chiqilgan ilovada dastur ishlayotgan qurilmani aylantirish yoki Ctrl+F11 yoki Ctrl+F12 tugmalar birikmalaridan foydalangan holda emulyatorda aylanishni osongina aniqlash mumkin bo'lgan kamchilik mavjud bo`lib, hisoblagich qiymati nolga qaytarilganligini ko'rish oson. Buning sababi, aylantirilganda, jarayon yo'q qilinadi va qayta yaratiladi va jarayon sinfining yangi namunasi hisoblagich qiymati nolga teng bo'lgan yangi modelni yaratadi.

Ushbu kamchilikni tuzatish uchun faoliyatni yo'q qilishdan oldin model holatini saqlash va yangi namunani yaratgandan so'ng uni tiklash kerak. **Activity** sinfidagi holatni saqlash uchun metod taqdim etiladi.

protected void onSaveInstanceState(Bundle outState);

Ushbu metod jarayon namunasi yo'q qilinganda chaqiriladi, lekin qayta yaratiladi va dastur odatdagi tarzda tugashi bilan chaqirilmaydi (masalan, agar foydalanuvchi **"Back"** tugmasini bossa). Argument sifatida **onSaveInstanceState ()** metodiga o'tkazilgan **Bundle** sinfining ob'ekti qayta yaratilgandan keyin holatni tiklashi kerak bo'lgan ma'lumotlarni saqlash uchun mo'ljallangan. Qayta tiklashning o'zi ham metodda, ham **onCreate()** metodida amalga oshirilishi mumkin.

protected void onRestoreInstanceState (Bundle savedInstanceState);

Ikkala metod ham **onSaveInstanceState()** bilan saqlangan ma'lumotlar bilan **Bundle** sinfining ob'ektiga o'tkaziladi.

Agar ma'lumotlarni qayta tiklash talab etilmasa (ya'ni, konfiguratsiya o'zgarishi tufayli uni qayta yaratish emas, balki sinfining dastlabki yaratilishi sodir bo'lgan), **onRestoreInstanceState()** chaqirilmaydi va null **onCreate**ga o'tkaziladi. **savedInstanceState()** parametrining qiymati sifatida avvalgi bo'limdagi **Counter** sinfining amalga oshirilishi qiymatlarni faqat noldan hisoblashni nazarda tutganligi sababli, modelni ixtiyoriy qiymat bilan ishga tushirish uchun konstruktorni yaratish kerak.

Standart konstruktor aniq belgilanishi kerak. Natijada, **Counter** sinfiga quyidagi o'zgarishlar kiritish kerak bo`ladi:

```
public class Counter {
private int value;
public Counter() { value = 0; }
public Counter(int initialValue) { value = initialValue; }
// ...
}
```

Endi model holatini saqlaydigan **MainActivity** sinfiga **onSaveInstanceState()** metodini qo'shamiz:

```
@Override
protected void onSaveInstanceState(Bundle bundle) {
super.onSaveInstanceState(bundle);
bundle.putInt("counterValue", counter.getValue());
}
```

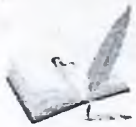
OnCreate() isbga tushirish metodiga o'zgartirishlar kiritish qoladi. To'liqlik uchun bu metodning to'liq kodini ko`ramiz:

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
if(savedInstanceState != null) {
```

Agar model o'zgarishlari kodda lokalizatsiya qilingan bo'lsa va katta hajmni tashkil qilmasa, unda passiv model etarli bo'lishi mumkin. Aks holda, faol modeldan foydalanish kerak va foydalanuvchilar bilan ishlash uchun kod yozish qo'shimcha kodning oshkoraligi va yuqori ishonchliligi bilan qoplanadi.

5-§. Ekran yo'nalishini o'zgartirish bilan ishlash.

Tayanch so'z va atamalar



Activity, protected, onSaveInstanceState, Bundle, Counter, MainActivity, konstruktor arxitektura, model.

Ishlab chiqilgan ilovada dastur ishlayotgan qurilmani aylantirish yoki Ctrl+F11 yoki Ctrl+F12 tugmalar birikmalaridan foydalangan holda emulyatorda aylanishni osongina aniqlash mumkin bo'lgan kamchilik mavjud bo`lib, hisob-lagich qiymati nolga qaytarilganligini ko'rish oson. Buning sababi, aylantirilganda, jarayon yo'q qilinadi va qayta yaratiladi va jarayon sinfining yangi namunasi hisoblagich qiymati nolga teng bo'lgan yangi modelni yaratadi.

Ushbu kamchilikni tuzatish uchun faoliyatni yo'q qilishdan oldin model holatini saqlash va yangi namunani yaratgandan so'ng uni tiklash kerak. **Activity** sinfidagi holatni saqlash uchun metod taqdim etiladi.

protected void onSaveInstanceState(Bundle outState);

Ushbu metod jarayon namunasi yo'q qilinganda chaqiriladi, lekin qayta yaratiladi va dastur odatdagi tarzda tugashi bilan chaqirilmaydi (masalan, agar foydalanuvchi **"Back"** tugmasini bossa). Argument sifatida **onSaveInstanceState ()** metodiga o'tkazilgan **Bundle** sinfining ob'ekti qayta yaratilgandan keyin holatni tiklashi kerak bo'lgan ma'lumotlarni saqlash uchun mo'ljallangan. Qayta tiklashning o'zi ham metodda, ham **onCreate()** metodida amalga oshirilishi mumkin.

protected void onRestoreInstanceState (Bundle savedInstanceState);

Ikkala metod ham **onSaveInstanceState()** bilan saqlangan ma'lumotlar bilan **Bundle** sinfining ob'ektiga o'tkaziladi.

Agar ma'lumotlarni qayta tiklash talab etilmasa (ya'ni, konfiguratsiya o'zgarishi tufayli uni qayta yaratish emas, balki sinfining dastlabki yaratilishi sodir bo'lgan), **onRestoreInstanceState()** chaqirilmaydi va null **onCreate**ga o'tkaziladi. **savedInstanceState()** parametrining qiymati sifatida avvalgi bo'limdagi **Counter** sinfining amalga oshirilishi qiymatlarni faqat noldan hisoblashni nazarda tutganligi sababli, modelni ixtiyoriy qiymat bilan ishga tushirish uchun konstruktorni yaratish kerak.

Standart konstruktor aniq belgilanishi kerak. Natijada, **Counter** sinfiga quyidagi o'zgarishlar kiritish kerak bo`ladi:

```
public class Counter {
private int value;
public Counter() { value = 0; }
public Counter(int initialValue) { value = initialValue; }
// ...
}
```

Endi model holatini saqlaydigan **MainActivity** sinfiga **onSaveInstanceState()** metodini qo'shamiz:

```
@Override
protected void onSaveInstanceState(Bundle bundle) {
super.onSaveInstanceState(bundle);
bundle.putInt("counterValue", counter.getValue());
}
```

OnCreate() ishga tushirish metodiga o'zgartirishlar kiritish qoladi. To'liqlik uchun bu metodning to'liq kodini ko`ramiz:

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
if(savedInstanceState != null) {
```

```
counter = new Counter(savedInstanceState.getInt("counterValue"));
}
counterText = (TextView) findViewById(R.id.counterText);
updateCounterView();
counter.setOnModificationListener(new Counter.OnModificationListener() {
@Override
public void onModification(Counter sender) {
updateCounterView()
}
});
}
```

Kodda ikkita asosiy o'zgarish mavjud. Ulardan birinchisi, parametrli konstruktor yordamida model ob'ektini yaratishdir (bu **savedInstanceState** parametrini null bilan solishtirish orqali aniqlanadi). Aks holda, standart konstruktor tomonidan yaratilgan ob'ekt hisoblagich maydonida qoladi. Ikkinchi o'zgarish **updateCounterView()**ni chaqirishdir. Bu zarur, chunki boshlang'ichda hisoblagichning qiymati nolga teng bo'lmasligi mumkin, bu qiymat ko'rsatiladigan matn maydoni uchun standart qiymatga mos kelmaydi. Yuqoridagi qo'ng'iroq matn maydonini model holati bilan sinxronlashtirishga majbur qilish orqali bu muammoni hal qiladi.

O'z-o'zini tekshirish uchun savollar va mashqlar

1. MVC arxitektura shabloni nima? Ushbu shablonga asoslangan tizimlar qanday tarkibiy qismlardan iborat?

2. Komponentlarning MVC ichida muloqot qilish va o'zaro ta'sir qilishning maqbul metodlarini tavsiflang. Nima uchun o'zaro munosabatlarga ruxsat berilgan va boshqalarga ruxsat berilmaganligini asoslang.

3. Android ilovalarida qo'llaniladigan foydalanuvchi interfeysini qurish metodini aytib bering.

4. Foydalanuvchi interfeysi tavsifini koddan qanday yuklash va individual vidjetlarga qanday kirish mumkinligini tasvirlab bering.

5. Androidda hodisalarni boshqarishning ikkita metodini tasvirlab bering. Har birining afzalliklari va kamchiliklarini aytib bering.

6. MVC arxitektura shablonlari nuqtai nazaridan faol va passiv modellarni aniqlang. Har bir turdagi modelning afzalliklari va kamchiliklarini keltiring.

*7. Foydalanuvchining ekranni aylanish hodisasini qanday to'g'ri boshqarishni tasvirlab bering. **Aktivity**ning sikl aylanishida nima sodir bo'ladi?*

8. Ushbu bobda tasvirlangan dastur loyihasini yarating. Ilovani kompilyatsiya qiling va uni emulyatorda yoki haqiqiy qurilmada ishga tushiring.

```
counter = new Counter(savedInstanceState.getInt("counterValue"));
}
counterText = (TextView) findViewById(R.id.counterText);
updateCounterView();
counter.setOnModificationListener(new Counter.OnModificationListener() {
@Override
public void onModification(Counter sender) {
updateCounterView()
}
});
}
```

Kodda ikkita asosiy o'zgarish mavjud. Ulardan birinchisi, parametrli konstruktor yordamida model ob'ektini yaratishdir (bu **savedInstanceState** parametrini null bilan solishtirish orqali aniqlanadi). Aks holda, standart konstruktor tomonidan yaratilgan ob'ekt hisoblagich maydonida qoladi. Ikkinchi o'zgarish **updateCounterView()**ni chaqirishdir. Bu zarur, chunki boshlang'ichda hisoblagichning qiymati nolga teng bo'lmasligi mumkin, bu qiymat ko'rsatiladigan matn maydoni uchun standart qiymatga mos kelmaydi. Yuqoridagi qo'ng'iroq matn maydonini model holati bilan sinxronlashtirishga majbur qilish orqali bu muammoni hal qiladi.

O'z-o'zini tekshirish uchun savollar va mashqlar

1. MVC arxitektura shabloni nima? Ushbu shablonga asoslangan tizimlar qanday tarkibiy qismlardan iborat?

2. Komponentlarning MVC ichida muloqot qilish va o'zaro ta'sir qilishning maqbul metodlarini tavsiflang. Nima uchun o'zaro munosabatlarga ruxsat berilgan va boshqalarga ruxsat berilmaganligini asoslang.

3. Android ilovalarida qo'llaniladigan foydalanuvchi interfeysini qurish metodini aytib bering.

4. Foydalanuvchi interfeysi tavsifini koddan qanday yuklash va individual vidjetlarga qanday kirish mumkinligini tasvirlab bering.

5. Androidda hodisalarni boshqarishning ikkita metodini tasvirlab bering. Har birining afzalliklari va kamchiliklarini aytib bering.

6. MVC arxitektura shablonlari nuqtai nazaridan faol va passiv modellarni aniqlang. Har bir turdagi modelning afzalliklari va kamchiliklarini keltiring.

*7. Foydalanuvchining ekranni aylanish hodisasini qanday to'g'ri boshqarishni tasvirlab bering. **Aktivity**ning sikl aylanishida nima sodir bo'ladi?*

8. Ushbu bobda tasvirlangan dastur loyihasini yarating. Ilovani kompilyatsiya qiling va uni emulyatorda yoki haqiqiy qurilmada ishga tushiring.

```
counter = new Counter(savedInstanceState.getInt("counterValue"));
}
counterText = (TextView) findViewById(R.id.counterText);
updateCounterView();
counter.setOnModificationListener(new Counter.OnModificationListener() {
@Override
public void onModification(Counter sender) {
updateCounterView()
}
});
}
```

Kodda ikkita asosiy o'zgarish mavjud. Ulardan birinchisi, parametrli konstruktor yordamida model ob'ektini yaratishdir (bu **savedInstanceState** parametrini null bilan solishtirish orqali aniqlanadi). Aks holda, standart konstruktor tomonidan yaratilgan ob'ekt hisoblagich maydonida qoladi. Ikkinchi o'zgarish **updateCounterView()**ni chaqirishdir. Bu zarur, chunki boshlang'ichda hisoblagichning qiymati nolga teng bo'lmasligi mumkin, bu qiymat ko'rsatiladigan matn maydoni uchun standart qiymatga mos kelmaydi. Yuqoridagi qo'ng'iroq matn maydonini model holati bilan sinxronlashtirishga majbur qilish orqali bu muammoni hal qiladi.

O'z-o'zini tekshirish uchun savollar va mashqlar

1. MVC arxitektura shabloni nima? Ushbu shablonga asoslangan tizimlar qanday tarkibiy qismlardan iborat?

2. Komponentlarning MVC ichida muloqot qilish va o'zaro ta'sir qilishning maqbul metodlarini tavsiflang. Nima uchun o'zaro munosabatlarga ruxsat berilgan va boshqalarga ruxsat berilmaganligini asoslang.

3. Android ilovalarida qo'llaniladigan foydalanuvchi interfeysini qurish metodini aytib bering.

4. Foydalanuvchi interfeysi tavsifini koddan qanday yuklash va individual vidjetlarga qanday kirish mumkinligini tasvirlab bering.

5. Androidda hodisalarni boshqarishning ikkita metodini tasvirlab bering. Har birining afzalliklari va kamchiliklarini aytib bering.

6. MVC arxitektura shablonlari nuqtai nazaridan faol va passiv modellarni aniqlang. Har bir turdagi modelning afzalliklari va kamchiliklarini keltiring.

*7. Foydalanuvchining ekranni aylanish hodisasini qanday to'g'ri boshqarishni tasvirlab bering. **Aktivity**ning sikl aylanishida nima sodir bo'ladi?*

8. Ushbu bobda tasvirlangan dastur loyihasini yarating. Ilovani kompilyatsiya qiling va uni emulyatorda yoki haqiqiy qurilmada ishga tushiring.

6-BOB. VIEW SINFI VA UNING IMKONIYATLARI.

View klass Androiddagi barcha vidjet sinflarining yuqori sinfidir, jumladan, **TextView, ImageView, Button** va boshqalar. **View** sinfining har bir nusxasi ekranda qandaydir to'rtburchaklar maydonini chizish, shuningdek, bu bilan bog'liq hodisalarni boshqarish uchun javob beradi. Android uchun ilovalarni ishlab chiqishda, **View** sinfining tayyor kutubxona kichik sinflaridan foydalaniladi. Biroq, ba'zi hollarda, o'ziga xos ko'rinishga ega bo'lgan komponentlar kerak bo'ladi. Bunday komponentlar **View** sinfidan o'z sinfini olish va renderlash yoki hodisalarni boshqarish uchun mas'ul bo'lgan metodlarni bekor qilish orqali amalga oshirilishi mumkin.

View klass tomonidan boshqariladigan barcha hodisalardan eng muhimi foydalanuvchi **View** mas'ul bo'lgan hodisalar, fokuslarni uzatish hodisalari, tugmalarni bosish va chizish hodisalaridir. Keling, ularni batafsil ko'rib chiqaylik.

1-§. Ekranga teginish hodisalari.

Tayanch so'z va atamalar

View, OnTouchListener, MotionEvent, action, pressure, size , OnTouchEvent, Menu, EditText, TextView.

Foydalanuvchi **View** sinfining ma'lum bir nusxasi egallagan ekrandagi maydonga tegsa, ishlov beriladigan voqea sodir bo'ladi.

public boolean onTouchEvent(MotionEvent event);

Ushbu metodni bekor qilishdan tashqari, **View.OnTouchListener** tipidagi foydalanuvchilar sinfi yordamida sensorli hodisani boshqarish imkoniyati ham mavjud. Ushbu hodisa uchun ishlov berish metodi quyidagicha:

public boolean onTouch(View v, MotionEvent event);

Ikkala metod ham parametr sifatida **MotionEvent** sinfi ob'ektini oladi, bu teginish tafsilotlarini tavsiflaydi. Ushbu sinfning[3] asosiy xususiyatlari:

[3] Eslatib o'tamiz, qiymatlarni olish uchun getX(), getY() va boshqalar kabi mos keladigan yordamchi usullarni chaqirish kerak.

• **x, y** — vidjetning o'z koordinata tizimidagi teginish koordinatalari;

• **action** — hodisa turi:

• **pressure** — ekrandagi bosim kuchi (haqiqiy raqam 0,0 dan 1,0 gacha; bosim kuchi ta'rifini qo'llab-quvvatlamaydigan ekranlar uchun u har doim 1,0 ni tashkil qiladi);

• **size** - teginish maydoni o'lchami.

Foydalanuvchi ekranga tegsa, **MotionEvent.ACTION UP** tipidagi hodisa hosil bo'ladi va foydalanuvchi barmog'ini ekrandan ko'targanda **MotionEvent. ACTION DOWN** tipidagi hodisa hosil bo'ladi. Agar foydalanuvchi barmog'ini ekran bo'ylab harakatlantirsa, qo'shimcha ravishda **MotionEvent.ACTION MOVE** tipidagi bir qator hodisalar hosil bo'ladi, ularning har biri foydalanuvchi barmog'i harakatlanayotgan traektoriya bo'lagining tavsifini o'z ichiga oladi.

Traektoriyaga kiritilgan nuqtalarni olish uchun **MotionEvent** sinfining metodlari qo'llaniladi. E'tibor bering, nuqtalarni yo'l bo'laklariga guruhlash har bir nuqta uchun alohida hodisalar ishlashini yaxshilash uchun amalga oshiriladi.

```
public final int getHistorySize();
public final float getHistoricalX(int index);
public final float getHistoricalY(int index);
public final float getHistoricalPressure(int index);
public final float getHistoricalSize(int pos);
```

Ekranni teginish hodisalarini boshqarishga misol sifatida, jurnaldagi barcha sodir bo'lgan voqealar tavsifini aks ettiruvchi **OnTouchEvent()** metodining kodini keltiramiz:

```
@Override
public boolean onTouch(View v, MotionEvent event) {
float x = event.getX(), y = event.getY();
switch (event.getAction()) {
case MotionEvent.ACTION_DOWN:
Log.d(getClass().getSimpleName(), "down: " + x + ", " + y);
```

```
break;
case MotionEvent.ACTION_UP:
Log.d(getClass().getSimpleName(), "up: " + x + ", " + y);
break;
case MotionEvent.ACTION_MOVE:
Log.d(getClass().getSimpleName(), "move: ");
for (int i = 0; i < event.getHistorySize(); i++) {
Log.d(getClass().getSimpleName(), "----- " +
event.getHistoricalX(i) + ", " + event.getHistoricalY(i));
}
Log.d(getClass().getSimpleName(), "----- " + x + ", " + y);
break;
}
return true;
}
```

Ilova chiqish misoli:

```
18:32:30.270: DEBUG/MainActivity(1478): down: 241.0, 427.0
18:32:30.412: DEBUG/MainActivity(1478): up: 241.0, 427.0
18:32:31.740: DEBUG/MainActivity(1478): down: 137.0, 165.0
18:32:31.760: DEBUG/MainActivity(1478): move:
18:32:31.770: DEBUG/MainActivity(1478): ----- 137.0, 154.0
18:32:31.770: DEBUG/MainActivity(1478): move:
18:32:31.770: DEBUG/MainActivity(1478): ----- 139.0, 135.0
18:32:31.770: DEBUG/MainActivity(1478): ----- 141.0, 133.0
18:32:31.800: DEBUG/MainActivity(1478): move:
18:32:31.810: DEBUG/MainActivity(1478): ----- 147.0, 141.0
18:32:31.990: DEBUG/MainActivity(1478): up: 147.0, 141.0
```

Klaviatura hodisalari. Klaviatura hodisalarini boshqarish juda kam uchraydi, chunki ko'pgina Android qurilmalarida apparat klaviaturasi yo'q. Biroq,

bu qobiliyat **"Menu"** va **"Back"** kabi standart tugmalarni qayta ishlash uchun talab qilinishi mumkin.

Klaviatura hodisalarini boshqarish uchun metodlarni bekor qilish quyidagicha:

public boolean onKeyDown(int keyCode, KeyEvent event);

public boolean onKeyUp(int keyCode, KeyEvent event);

yoki **View.OnKeyEventListener** tipidagi foydalanuvchini ro'yxatdan o'tkazish va metodni belgilash kerak.

boolean onKey(View v, int keyCode, KeyEvent event);

Ekranni teginish hodisalarida bo'lgani kabi, **KeyEvent** obyektining harakat xususiyati hodisa turini aniqlash imkonini beradi:

- *KeyEvent.ACTION_DOWN - tugmani bosish;*
- *KeyEvent.ACTION_UP - uni chiqarish;*
- *KeyEvent.ACTION_MULTIPLE - avtomatik takrorlash.*

KeyCode parametri bosilgan tugma kodini o'z ichiga oladi. Masalan, **KeyEvent.KEYCODE_DPAD_LEFT** qiymati telefonning joystik chap tugmasiga, **KeyEvent.KEYCODE_MENU** qiymati esa **Menyu** tugmasiga mos keladi. Boshqa kod qiymatlarini hujjatlarda topish mumkin. E'tibor berilsa, klaviatura hodisalari faqat hozirda kiritish fokusiga ega bo'lgan vidjetga uzatiladi. Bunday vidjetlar, masalan, **TextView** komponentini emas, balki **EditText** komponentini o'z ichiga oladi. Maxsus komponent kirish fokusini olishi uchun uning fokuslanadigan xususiyati rost qiymatiga o'rnatilishi kerak.

2-§. Vidjet ierarxiyasi bo'yicha jarayonni boshqarish qoidalari.

Tayanch so'z va atamalar

drawArrow, DrawHands, DrawNumbers, antialias, parameter, kanvas, Vidjet, kontekst.

Har qanday vidjet o'zi mas'ul bo'lgan joydagi boshqa vidjetlarni o'z ichiga olishi mumkin. Himoya ierarxiyasi to'g'ridan-to'g'ri kodda yoki foydalanuvchi

```
break;
case MotionEvent.ACTION_UP:
Log.d(getClass().getSimpleName(), "up: " + x + ", " + y);
break;
case MotionEvent.ACTION_MOVE:
Log.d(getClass().getSimpleName(), "move: ");
for (int i = 0; i < event.getHistorySize(); i++) {
Log.d(getClass().getSimpleName(), "------ " +
event.getHistoricalX(i) + ", " + event.getHistoricalY(i));
}
Log.d(getClass().getSimpleName(), "------ " + x + ", " + y);
break;
}
return true;
}
```

Ilova chiqish misoli:

```
18:32:30.270: DEBUG/MainActivity(1478): down: 241.0, 427.0
18:32:30.412: DEBUG/MainActivity(1478): up: 241.0, 427.0
18:32:31.740: DEBUG/MainActivity(1478): down: 137.0, 165.0
18:32:31.760: DEBUG/MainActivity(1478): move:
18:32:31.770: DEBUG/MainActivity(1478): ---- 137.0, 154.0
18:32:31.770: DEBUG/MainActivity(1478): move:
18:32:31.770: DEBUG/MainActivity(1478): ---- 139.0, 135.0
18:32:31.770: DEBUG/MainActivity(1478): ---- 141.0, 133.0
18:32:31.800: DEBUG/MainActivity(1478): move:
18:32:31.810: DEBUG/MainActivity(1478): ---- 147.0, 141.0
18:32:31.990: DEBUG/MainActivity(1478): up: 147.0, 141.0
```

Klaviatura hodisalari. Klaviatura hodisalarini boshqarish juda kam uchraydi, chunki ko'pgina Android qurilmalarida apparat klaviaturasi yo'q. Biroq,

bu qobiliyat **"Menu"** va **"Back"** kabi standart tugmalarni qayta ishlash uchun talab qilinishi mumkin.

Klaviatura hodisalarini boshqarish uchun metodlarni bekor qilish quyidagicha:

public boolean onKeyDown(int keyCode, KeyEvent event);

public boolean onKeyUp(int keyCode, KeyEvent event);

yoki **View.OnKeyEventListener** tipidagi foydalanuvchini ro'yxatdan o'tkazish va metodni belgilash kerak.

boolean onKey(View v, int keyCode, KeyEvent event);

Ekranni teginish hodisalarida bo'lgani kabi, **KeyEvent** obyektining harakat xususiyati hodisa turini aniqlash imkonini beradi:

- *KeyEvent.ACTION DOWN - tugmani bosish;*
- *KeyEvent.ACTION UP - uni chiqarish;*
- *KeyEvent.ACTION MULTIPLE - avtomatik takrorlash.*

KeyCode parametri bosilgan tugma kodini o'z ichiga oladi. Masalan, **KeyEvent.KEYCODE DPAD LEFT** qiymati telefonning joystik chap tugmasiga, **KeyEvent.KEYCODE MENU** qiymati esa **Menyu** tugmasiga mos keladi. Boshqa kod qiymatlarini hujjatlarda topish mumkin. E'tibor berilsa, klaviatura hodisalari faqat hozirda kiritish fokusiga ega bo'lgan vidjetga uzatiladi. Bunday vidjetlar, masalan, **TextView** komponentini emas, balki **EditText** komponentini o'z ichiga oladi. Maxsus komponent kirish fokusini olishi uchun uning fokuslanadigan xususiyati rost qiymatiga o'rnatilishi kerak.

2-§. Vidjet ierarxiyasi bo'yicha jarayonni boshqarish qoidalari.

Tayanch so'z va atamalar

drawArrow, DrawHands, DrawNumbers, antialias, parameter, kanvas, Vidjet, kontekst.

Har qanday vidjet o'zi mas'ul bo'lgan joydagi boshqa vidjetlarni o'z ichiga olishi mumkin. Himoya ierarxiyasi to'g'ridan-to'g'ri kodda yoki foydalanuvchi

interfeysini tavsiflovchi XML faylida elementlarni joylashtirish orqali tuzilishi mumkin. Qoida tariqasida, ikkinchi metod amalda qo'llaniladi.

Vidjetlar ierarxiyasi hodisalarni boshqarish uchun zarurdir. Ierarxiya bo'ylab jarayonlarni boshqarishni tartibga soluvchi maxsus qoidalar mavjud:

• avvalo, hodisa sodir bo'lgan hudud uchun mas'ul bo'lgan eng ko'p joylashtirilgan (daraxtdagi barg) vidjetiga o'tkaziladi;

• agar vidjet sodir bo'lgan voqea uchun o'zining ishlov beruvchisini aniqlamasa, u ishlov berish uchun asosiy vidjetga uzatiladi, aks holda, ichki vidjeti chaqiriladi;

• agar hodisa **true** bo'lsa, u holda vidjet voqeani boshqargan deb hisoblanadi va boshqa qayta ishlash talab etilmaydi;

• agar ishlov beruvchi **false** qiymatini qaytarsa, vidjet voqeani boshqargan, lekin asosiy vidjet ham bu hodisani qayta ishlashni davom ettirishi kerak.

Xulosa sifatida, ushbu qoidalar juda moslashuvchan bo'lib, ular ichki vidjetlarda jarayonlarni qayta ishlash metodlarini bekor qilish va ierarxiyaning turli vidjetlarini zanjirga bog'lash imkonini beradi.

Vidjetda chizish. Ko'pgina hollarda, o'z vidjetingizni yaratishning sababi kontentni ko'rsatish metodini aniqlashdir. Barcha turdagi o'yin ilovalari, diagrammalar, tasvirlarni qayta ishlash va hokazolar uchun maydonlarni o'z ichiga olgan vidjetlar bunga misoldir.

Vidjet mazmunini o'ziga xos metodini aniqlash uchun metodni bekor qilish lozim.

protected void onDraw(Canvas canvas);

Ushbu metodga o'tkazilgan kanvas parametri grafik kontekstni qamrab oluvchi va grafik chizish va koordinatalar tizimini boshqarish metodlarini ta'minlaydigan ob'ektdir. Bunda chizma parametrlari **Paint** sinfi ob'ektlari tomonidan o'rnatiladi. Android platformasining grafik xususiyatlari juda keng va ulardan foydalanish tafsilotlarini boshqa adabiyotlarda ham topish mumkin, shuning uchun biz yuqorida aytib o'tilgan ba'zi misollar bilan cheklanamiz.

Ushbu misol soat yuzini ko'rsatadigan vidjetdir. Kerakli izohlarni berib, tegishli metodlarini ketma-ket ko'rib chiqamiz.

```
public class ClockView extends View {
public ClockView(Context context, AttributeSet attrs) {
super(context, attrs);
}
```

Soat yuzi vidjeti **View** sinfidan meros bo'lib, superklass konstruktorini belgilaydi. Bu XML faylidan tavsifiga ko'ra sinfni to'g'ri yaratish uchun kerak.

```
@Override
protected void onDraw(Canvas canvas) {
int size = Math.min(getWidth(), getHeight()) / 2;
canvas.translate(getWidth() / 2, getHeight() / 2);
Paint paint = new Paint();
paint.setAntiAlias(true);
drawClockFace(canvas, size, paint);
drawScale(canvas, size, paint);
drawNumbers(canvas, size, paint);
drawHands(canvas, size, paint);
}
```

onDraw() metodi kelgusi o'lchamlarni aniqlaydi va uning kelib chiqishini markazga o'tkazadi, shundan so'ng, **Paint** sinfining ob'ekti yaratiladi va ekranda ko'rsatilgan grafiklarni ***antialias***ini ta'minlaydigan parametr o'rnatiladi. Keyingi renderlash bosqichlari alohida metodlar bilan amalga oshiriladi.

```
private void drawClockFace(Canvas canvas, int size, Paint paint) {
paint.setStyle(Paint.Style.FILL);
paint.setColor(Color.DKGRAY);
paint.setStrokeWidth(4);
canvas.drawCircle(0, 0, size, paint);
}
```

drawClockFace() metodi **drawCircle()** metodi yordamida bo'sh to'q-kulrang yuzani chizadi.

```
private void drawScale(Canvas canvas, int size, Paint paint) {
paint.setStyle(Paint.Style.STROKE);
paint.setColor(Color.YELLOW);
paint.setStrokeWidth(3);
for (int i = 0; i < 12; i++) {
canvas.drawLine(size - 20, 0, size, 0, paint);
canvas.rotate(30);
}
}
```

DrawScale() metodi qalinligi 3 bo'lgan sariq chiziqlar bilan 12 ta terish shkalasi bo'linmasini chizadi. Kodni soddalashtirish uchun har bir bo'linish chizilgandan so'ng, koordinatalar tizimi 30 gradusga aylantiriladi. Bu bir xil koordinatalar yordamida turli koordinata tizimlarida barcha belgilarni chizish imkonini beradi. Chizish oxirida koordinatalar tizimi avtomatik ravishda dastlabki holatiga qaytadi.

```
private void drawNumbers(Canvas canvas, int size, Paint paint) {
paint.setTextSize(28);
paint.setTextAlign(Paint.Align.CENTER);
canvas.drawText("12", 0, -size + 60, paint);
canvas.drawText("6", 0, size - 60 + textSize("6", paint).y, paint);
paint.setTextAlign(Paint.Align.RIGHT);
canvas.drawText("3", size - 40, textSize("3", paint).y / 2, paint);
paint.setTextAlign(Paint.Align.LEFT);
canvas.drawText("9", -size + 40, textSize("9", paint).y / 2, paint);
}
```

DrawNumbers() metodi vaqt jadvalida raqamlar belgilarini chizadi. Oddiylik uchun uchga ko'payadigan teglar bilan cheklaymiz. **Paint** sinfining

setTextAlign() metodi chizilgan matnning **drawText()** parametri bilan belgilangan nuqtaga gorizontal tekislanishini ta'minlash uchun ishlatiladi. **Android API**da vertikal tekislash uchun shunga o'xshash vosita yo'q va to'g'ridan-to'g'ri koordinatalarni hisoblash orqali bu muammoni hal qilish kerak. Buning uchun **textSize()** yordamchi metodi qo'llaniladi.

```
private static Point textSize(String text, Paint paint) {
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
return new Point(bounds.width(), bounds.height());
}
```

Haqiqiy matn hajmini aniqlaydigan **getTextBounds()** metodi qiymat qaytarmasligi, lekin chiqish parametriga ega bo'lgani uchun yordamchi metod talab qilinadi.

```
private void drawHands(Canvas canvas, int size, Paint paint) {
canvas.rotate(-90);
canvas.save();
canvas.rotate(1.9f * 360 / 12);
drawArrow(canvas, size / 3, paint);
canvas.restore();
canvas.rotate(1.9f * 360);
drawArrow(canvas, size * 5 / 8, paint);
}
```

DrawHands() metodi soat strelkalarini chizadi. O'qlarni kerakli joylarga chizish uchun belgilarni chizishdagi kabi yondashuv qo'llaniladi: koordinatalar tizimi yangi koordinatalar tizimidagi strelka gorizontal bo'lishi uchun aylantiriladi. Masalan, oddiylik uchun har doim belgilangan vaqtni ko'rsatadi - 1:54 (1,9 soat).

```
private void drawArrow(Canvas canvas, int length, Paint paint) {
Path path = new Path();
path.moveTo(0, 0);
```

```
path.rLineTo(length, 0);
path.rLineTo(0, 10);
path.rLineTo(20, -10);
path.rLineTo(-20, -10);
path.rLineTo(0, 10);
paint.setStyle(Paint.Style.FILL_AND_STROKE);
paint.setStrokeWidth(2);
paint.setColor(Color.RED);
canvas.drawPath(path, paint);
}
} // class ClockView
```

Oxirgi **drawArrow()** metodi soat strelkasini chizib, quyidagini ko'rsatadi: birinchi navbatda segmentlar ketma-ketligidan iborat yo'l ob'ekti (**Path** klassi) yaratiladi, so'ngra u chiziladi va rang bilan to'ldiriladi (Paint. **Style.FILL** uslubi **AND STROKE**) **drawPath()** metodini chaqirish orqali ishlatiladi.

Bo'limni yakunlash uchun bu erda **CloskView** tipidagi vidjet bilan asosiy foydalanuvchi interfeysini tavsiflovchi **res/layout/main.xml** faylining mazmuni keltirib o`tamiz:

```
<?xml version="1.0" encoding="utf-8"?>
<view android:layout width="wrap content"
android:layout height="wrap content"
class="ru.ac.uniyar.clockviewdemo.ClockView"
xmlns:android="http://schemas.android.com/apk/res/android" />
```


O'z-o'zini tekshirish uchun savollar va mashqlar

1. View sinfining maqsadi nima? Ushbu sinfning kichik sinflarini qachon yaratish kerak?

2. Foydalanuvchi tomonidan qurilma teginish hodisalarini boshqarish uchun nima qilish kerakligini tasvirlab bering.

3. Android qurilmasi klaviaturasidan hodisalarni boshqarish uchun nima qilish kerakligini tasvirlab bering.

4. Vidjetlar ierarxiyasi bo'yicha hodisalarni boshqarish qoidasini shakllantirish. Ushbu qoidaning maqsadi nima?

5. Ixtiyoriy vidjetlarda chizish imkonini beruvchi asosiy sinflarni ayting. Ushbu sinflarning imkoniyatlarini ta'kidlang. Ushbu imkoniyatlarni misollar bilan ko'rsating.

6. Bo'limda tasvirlangan dastur loyihasini yarating. Ilovani kompilyatsiya qiling va uni emulyatorda yoki haqiqiy qurilmada ishga tushiring.

7-BOB. RESURSLAR BILAN ISHLASH

Androiddagi manbalar - bu dasturning bir qismi bo'lgan statik ma'lumotlar (masalan, matn, rasmlar, foydalanuvchi interfeysi tavsifi). Resurslar loyiha ichiga fayllar ko'rinishida joylashtiriladi va yaratish jarayonida avtomatik ravishda dasturning **apk** paketiga o'tkaziladi.

Resurslardan foydalanish ikkita asosiy maqsadga ega.

1. Ma'lumotlarni koddan ajratish. Resurslardan foydalanganda ma'lumotlar koddan alohida deklarativ shaklda saqlanadi, shuning uchun uni o'zgartirish oson va o'zgarishlarni dasturchi bo'lmaganlar, masalan, foydalanuvchi interfeysi dizaynerlari yoki tarjimonlar bajarishlari mumkin. Resurslarni o'zgartirish ilovani qayta kompilyatsiya qilishni talab qilmaydi, faqat uni qayta tiklashni talab qiladi.

2.Konfiguratsiyaga qarab foydalaniladigan resurslarning o'zgaruvchanligini ta'minlash. Resurslar sizga har xil turdagi ekran yo'nalishini va turli xil foydalanuvchi interfeysi tillarini qo'llab-quvvatlashni dasturchiga qo'shimcha xarajatlarsiz amalga oshirish imkonini beradi, chunki joriy tizim konfiguratsiyasiga mos keladigan resurslar avtomatik ravishda yuklanadi.

1-§. Resurslar tasnifi.

Tayanch so'z va atamalar

Drawable, Values, loyiha, resurs, layout, identifikator, vidjet, Resurs.

Android resurslarning quyidagi asosiy turlarini ajratib turadi.

• **Layout** - foydalanuvchi interfeysi elementlarining tartibini tavsiflovchi XML fayllari.

• **Menyu** - menyu yoki harakatlar paneli elementlarining tartibini tavsiflovchi XML fayllari.

• **Drawable** - ilova tomonidan ishlatiladigan rasm fayllari. Bunga foydalanuvchi interfeysida ishlatiladigan grafik fayllar ham kiradi.

• **Values** - XML matn formatida taqdim etilgan ilova ma'lumotlari. Avvalo, ushbu turdagi resurslar an'anaviy tarzda **strings.xml** faylida joylashtirilgan ilovaning barcha matn satrlarini o'z ichiga oladi. Bunday faylning namunasi quyida keltirib o'tiladi:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app name">Alarm clock</string>
<string name="hours label">Hours</string>
<string name="minutes label">Minutes</string>
</resources>
```

• **Raw** - ixtiyoriy ma'lumotlar, odatda **binary** shaklda. Har bir turdagi manbalar loyihaning **res** katalogiga nomlari resurs turlari bilan bir xil bo'lgan, lekin kichik harflar bilan yozilgan pastki kataloglarda joylashtiriladi.

Ilova ichidagi resurslardan foydalanish. Loyihani yaratish jarayonida (va ishchi stoldan foydalanganda, shuningdek, dastur resurslarida har qanday o'zgarishlar yuz berganda) dastur resurslaridan foydalanish uchun barcha loyiha resurslarining identifikatorlarini o'z ichiga olgan maxsus ***R.java*** fayli yaratiladi. Ularning barchasi **R** kichik sinflaridagi statik konstantalar sifatida aniqlanadi, ularning har biri boshqa resurs turiga mos keladi:

• **R.layout** – foydalanuvchi interfeysi maketini tavsiflovchi **res/layout** katalogi manbalari;

• **R.menu** - menyu yoki harakatlar paneli tarkibini tavsiflovchi **res/menu** katalogidan resurslar;

• **R.id** — resurs fayllarida tasvirlangan foydalanuvchi interfeysi komponentlari (res/layout katalogining fayllari); elementlar android:id atributi tomonidan berilgan identifikatorlar bilan aniqlanadi;

• **R.drawable — res/drawables** katalogidan tasvir resurslari;

• **R.string, R.integer, R.boolean, R.color, R.array** va boshqalar — ma'lumotlar turi bo'yicha guruhlangan ma'lumotlar fayllari (res/values katalogidagi fayllar) resurslari;

• **R.raw - res/raw** katalogidagi boshqa resurslar.

Keling, resurslardan foydalanishga misollar keltiraylik. Ilovaning foydalanuvchi interfeysini tavsiflovchi resursni **res/layout/main.xml** faylidan yuklash quyidagi chaqiruv orqali amalga oshiriladi:

setContentView(R.layout.main);

Res/strings.xml faylida aniqlangan matn resursini o'qish,

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

<resources>

<string name="error message">Error!</string>

</resources>

Shuningdek, ushbu resursdagi ma'lumotlarni xabar ko'rinishida ko'rsatish quyidagi tarzda amalga oshirilishi mumkin:

Toast.makeText(this, R.string.error message, Toast.LENGTH LONG). show();

Yuqoridagi misollarda faqat turli vidjetlarga uzatilgan resurs identifikatorlaridan foydalanilgan, bu esa o'z navbatida kerakli resurslarni yuklab olgan. Ammo Resurslar sinfining metodlaridan foydalangan holda to'g'ridan-to'g'ri yuklash imkoniyati mavjud. Ushbu sinf ob'ektini Kontekst sinfining **getResources()** ni chaqirish natijasida olish mumkin:

Resurs resurslari = getResources();

Keyin matn manbasini va rasm manbasini quyidagi kabi olish mumkin:

String text = resources.getText(R.string.app name);

Drawable icon = resources.getDrawable(R.drawable.ic launcher);

Shuningdek, boshqa resurslar tavsifidan resurslarga havola qilish mumkin. U foydalanuvchi interfeysi fayllarida element identifikatorlarini aniqlash uchun ishlatiladi:

<TextView android:id="@+ id/counterTextView" />

res/values fayllaridagi satrlarni vidjet belgilari sifatida almashtirish:

<TextView android:text="@string/counterText" />

va manifest faylida:

<application android:label="@string/application name"

android:icon="@drawable/ic launcher">

Ikkinchi holda, ilova nomini o'z ichiga olgan matn resursi va uning belgisi bo'lgan rasm resursi ko'rsatiladi.

2-§. Konfiguratsiyaga bog'liq manbalar.

Tayanch so'z va atamalar

masshtab, portret, landshaft, interfeys, layout, drawable, identifikator, vidjet, Resurs.

Yuqorida aytib o'tilganidek, resurslardan foydalanish maqsadlaridan biri tizim konfiguratsiyasiga qarab ularning o'zgaruvchanligini qo'llab-quvvatlashdir. Ushbu o'zgaruvchanlikni talab qiladigan eng muhim konfiguratsiya elementlari:

•**ekran ravshanligi** (masshtab o'lchashda sifatning yomonlashuviga yo'l qo'ymaslik uchun ilova foydalanuvchi ekrani ruxsatiga mos keladigan tasvirlardan foydalanishi kerak);

• **ekran orientatsiyasi** (portret va landshaft odatda foydalanuvchi interfeysi elementlarining boshqa joylashuvini talab qiladi);

• **joriy til** (ilovaning foydalanuvchi interfeysi tizimda o'rnatilgan tilda bo'lishi kerak).

O'zgaruvchanlik muammosini hal qilish uchun turli xil konfiguratsiyalarga mos keladigan resurs fayllari turli kataloglarga joylashtiriladi. Shu bilan birga, qo'shimchalar katalogdagi resurslar qaysi konfiguratsiya uchun mo'ljallanganligini ko'rsatadi:

• **layout** - portretni yo'naltirish uchun interfeys tavsifi katalogi; layout-land - landshaft uchun;

• **R.string, R.integer, R.boolean, R.color, R.array** va boshqalar — ma'lumotlar turi ho'yicha guruhlangan ma'lumotlar fayllari (res/values katalogidagi fayllar) resurslari;

• **R.raw - res/raw** katalogidagi boshqa resurslar.

Keling, resurslardan foydalanishga misollar keltirayllk. Ilovaning foydalanuvchi interfeysini tavsiflovchi resursni **res/layout/main.xml** faylidan yuklash quyidagi chaqiruv orqali amalga oshiriladi:

setContentView(R.layout.main);

Res/strings.xml faylida aniqlangan matn resursini o'qisb,

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

<resources>

<string name="error message">Error!</string>

</resources>

Shuningdek, ushbu resursdagi ma'lumotlarni xabar ko'rinisbida ko'rsatish quyidagi tarzda amalga oshirilishi mumkin:

Toast.makeText(this, R.string.error message, Toast.LENGTH LONG). show();

Yuqoridagi misollarda faqat turli vidjetlarga uzatilgan resurs identifikatorlaridan foydalanilgan, bu esa o'z navbatida kerakli resurslarni yuklab olgan. Ammo Resurslar sinfining metodlaridan foydalangan holda to'g'ridan-to'g'ri yuklash imkoniyati mavjud. Ushbu sinf ob'ektini Kontekst sinfining **getResources()** ni chaqirisb natijasida olish mumkin:

Resurs resurslari = getResources();

Keyin matn manbasini va rasm manbasini quyidagi kabi olish mumkin:

String text = resources.getText(R.string.app name);

Drawable icon = resources.getDrawable(R.drawable.ic launcher);

Shuningdek, boshqa resurslar tavsifidan resurslarga havola qilish mumkin. U foydalanuvchi interfeysi fayllarida element identifikatorlarini aniqlash uchun ishlatiladi:

<TextView android:id="@+id/counterTextView" />

res/values fayllaridagi satrlarni vidjet belgilari sifatida allnashtirish:

<TextView android:text="@string/counterText" />

va manifest faylida:

<application android:label="@string/application name"

android:icon="@drawable/ic launcher">

Ikkinchi holda, ilova nomini o'z ichiga olgan matn resursi va uning belgisi bo'lgan rasm resursi ko'rsatiladi.

2-§. Konfiguratsiyaga bog'liq manbalar.

Tayanch so'z va atamalar

masshtab, portret, landshaft, interfeys, layont, drawable, identifikator, vidjet, Resurs.

Yuqorida aytib o'tilganidek, resurslardan foydalanish maqsadlaridan biri tizim konfiguratsiyasiga qarab ularning o'zgaruvchanligini qo'llab-quvvatlashdir. Ushbu o'zgaruvchanlikni talab qiladigan eng muhim konfiguratsiya elementlari:

•**ekran ravshanligi** (masshtab o'lchashda sifatning yomonlashuviga yo'l qo'ymaslik uchun ilova foydalanuvchi ekrani ruxsatiga mos keladigan tasvirlardan foydalanishi kerak);

• **ekran orientatsiyasi** (portret va landshaft odatda foydalanuvchi interfeysi elementlarining boshqa joylashuvini talab qiladi);

• **joriy til** (ilovaning foydalanuvchi interfeysi tizimda o'rnatilgan tilda bo'lishi kerak).

O'zgaruvchanlik muammosini hal qilish uchun turli xil konfiguratsiyalarga mos keladigan resurs fayllari turli kataloglarga joylashtiriladi. Shu bilan birga, qo'shimchalar katalogdagi resurslar qaysi konfiguratsiya uchun mo'ljallanganligini ko'rsatadi:

• **layout** - portretni yo'naltirish uchun interfeys tavsifi katalogi; layout-land - landshaft uchun;

• **drawable** - sukut bo'yicha tasvirlar to'plami; *drawable ldpi, drawable-mdpi, drawable-hdpi* - mos ravishda past, o'rta va yuqori aniqlikdagi qurilmalar uchun tasvirlar to'plami;

• **values** - qiymatlar standart til (ingliz) uchun dastur satrlari, *values-ru* ruscha lokalizatsiyadagi dastur satrlari, *values-de* nemis lokalizatsiyasidagi dastur satrlari va hokazo.

Ilovani xalqarolashtirish uchun ushbu mexanizmdan foydalanishga misol ko'raylik. Avvalgi bo'limda keltirilgan resurs fayli **strings.xml** deb nomlansin va loyihaning **res/values** katalogida joylashgan bo'lsin. Keyin, rus tili uchun xalqarolashtirishni ta'minlash uchun quyidagi faylni yaratish kerak:

```
<?xml version="1.0" encoding="utf-8"?>
<resurslar>
<string name="app name">Budilnik</string>
<string name="hours label">Soat</string>
<string name="minutes label">Daqiqalar</string>
</ resurslar>
```

va uni *res/values-ru* katalogiga **strings.xml** nomi ostida joylashtiriladi.

3-§. Menyular va harakatlar panelini shakllantirish uchun resurslardan foydalanish.

Tayanch so'z va atamalar

Drawable, Values, loyiha, resurs, layout, identifikator, vidjet, Resurs.

Keling, asosiy menyuning shakllanishini ko'rsatadigan resurslardan foydalanishning yana bir misolini ko'rib chiqaylik. Androidning oldingi versiyalarida dasturning asosiy menyusi "Menu" apparat tugmachasini bosish orqali chaqiriladi. Android 3 dan boshlab, boshqacha yondashuv qo'llaniladi: menyu elementlari ilovaning sarlavha satrining o'ng burchagida joylashgan bo'lib, bu harakatlar paneli

deb ataladi. Amalga oshirish printsipi ikkala mexanizm uchun ham bir xil. Keling, uning tavsifini beraylik.

Birinchidan, menyu resurs faylida tasvirlangan bo'lishi kerak. Bunday holda, fayl nomga ega deb taxmin qilanadi:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/menu save"
android:icon="@drawable/ic menu save"
android:title="@string/menu save"
android:showAsAction="ifRoom|withText" />
<item android:id="@+id/menu delete"
android:icon="@drawable/ic menu delete"
android:title="@string/menu delete"
android:showAsAction="ifRoom|withText" />
<item android:id="@+id/menu options"
android:title="@string/menu options"
android:showAsAction="never" />
</menu>
```

Menyu uchta elementni o'z ichiga oladi. Birinchi va ikkinchisi harakatlar paneli elementlari sifatida yaratilgan (**android:showAsAction** atributi mavjud va "never" ga teng emas), har birida belgi va matn aniqlangan (mos ravishda *android:icon* va *android:title* atributlari). "**IfRoom|withText**" qiymati harakat belgisi ham, uning matnli tavsifi ham harakatlar panelida ko'rsatilishi kerakligini anglatadi (ikkinchisi faqat panelda yetarli bo'sh joy bo'lsa). Boshqa qiymatlarga "always" kiradi, ya'ni element har doim ko'rsatilishi kerak va "never", ya'ni element harakatlar panelida ko'rsatilmaydi.

Bu yuqoridagi misoldagi uchinchi elementning tavsifida ishlatiladi. Foydalanuvchi ushbu elementga faqat "**Menu**" tugmasini bosish orqali kira oladi. Portret va landshaft yo'nalishidagi ochiq menyuga ega ilovaning skrinshotlari 7.3.1-rasmda ko'rsatilgan.

• **drawable** - sukut bo'yicha tasvirlar to'plami; *drawable ldpi, drawable-mdpi, drawable-hdpi* - mos ravishda past, o'rta va yuqori aniqlikdagi qurilmalar uchun tasvirlar to'plami;

• **values** - qiymatlar standart til (ingliz) uchun dastur satrlari, *values-ru* ruscha lokalizatsiyadagi dastur satrlari, *values-de* nemis lokalizatsiyasidagi dastur satrlari va hokazo.

Ilovani xalqarolashtirish uchun ushbu mexanizmdan foydalanishga misol ko`raylik. Avvalgi bo'limda keltirilgan resurs fayli **strings.xml** deb nomlansin va loyihaning **res/values** katalogida joylashgan bo'lsin. Keyin, rus tili uchun xalqarolashtirishni ta'minlash uchun quyidagi faylni yaratish kerak:

```
<?xml version="1.0" encoding="utf-8"?>
<resurslar>
<string name="app name">Budilnik</string>
<string name="hours label">Soat</string>
<string name="minutes label">Daqiqalar</string>
</ resurslar>
```

va uni *res/values-ru* katalogiga **strings.xml** nomi ostida joylashtiriladi.

3-§. Menyular va harakatlar panelini shakllantirish uchun resurslardan foydalanish.

Tayanch so'z va atamalar

Drawable, Values, loyiha, resurs, layout, identifikator, vidjet, Resurs.

Keling, asosiy menyuning shakllanishini ko'rsatadigan resurslardan foydalanishning yana bir misolini ko'rib chiqaylik. Androidning oldingi versiyalarida dasturning asosiy menyusi "Menu" apparat tugmachasini bosish orqali chaqiriladi. Android 3 dan boshlab, boshqacha yondashuv qo'llaniladi: menyu elementlari ilovaning sarlavha satrining o'ng burchagida joylashgan bo'lib, bu harakatlar paneli

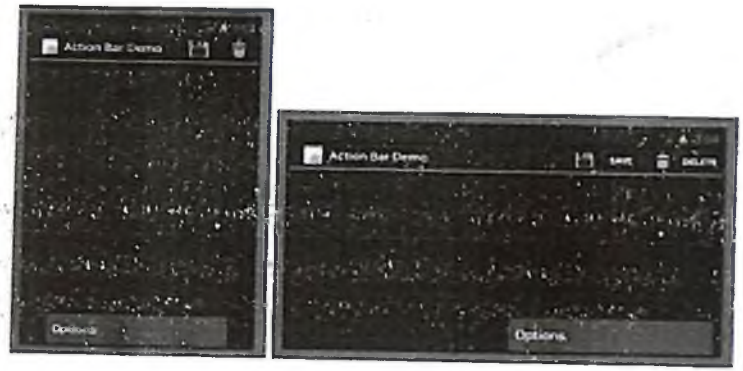
deb ataladi. Amalga oshirish printsipi ikkala mexanizm uchun ham bir xil. Keling, uning tavsifini beraylik.

Birinchidan, menyu resurs faylida tasvirlangan bo'lishi kerak. Bunday holda, fayl nomga ega deb taxmin qilanadi:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/menu save"
android:icon="@drawable/ic menu save"
android:title="@string/menu save"
android:showAsAction="ifRoom|withText" />
<item android:id="@+id/menu delete"
android:icon="@drawable/ic menu delete"
android:title="@string/menu delete"
android:showAsAction="ifRoom|withText" />
<item android:id="@+id/menu options"
android:title="@string/menu options"
android:showAsAction="never" />
</menu>
```

Menyu uchta elementni o'z ichiga oladi. Birinchi va ikkinchisi harakatlar paneli elementlari sifatida yaratilgan (**android:showAsAction** atributi mavjud va "never" ga teng emas), har birida belgi va matn aniqlangan (mos ravishda *android:icon* va *android:title* atributlari). "**IfRoom|withText**" qiymati harakat belgisi ham, uning matnli tavsifi ham harakatlar panelida ko'rsatilishi kerakligini anglatadi (ikkinchisi faqat panelda yetarli bo'sh joy bo'lsa). Boshqa qiymatlarga "always" kiradi, ya'ni element har doim ko'rsatilishi kerak va "never", ya'ni element harakatlar panelida ko'rsatilmaydi.

Bu yuqoridagi misoldagi uchinchi elementning tavsifida ishlatiladi. Foydalanuvchi ushbu elementga faqat "**Menu**" tugmasini bosish orqali kira oladi. Portret va landshaft yo'nalishidagi ochiq menyuga ega ilovaning skrinshotlari 7.3.1-rasmda ko'rsatilgan.



7.3.1-rasm. Portret va landshaft yo'nalishidagi harakatlar paneli.

Yuqoridagi tavsifni kodga yuklash uchun sinfning **onCreateOptionsMenu()** metodini bekor qilish kerak bo`ladi:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.main, menu);
return true;
}
```

Bu metod menyuni shakllantirish uchun ishlatilishi kerak bo'lgan "menu" parametridan o'tadi. Yuqoridagi kodda bu operatsiya XML fayllaridan menyularni yuklash imkoniyatiga ega **MenuInflater** class yordamida amalga oshiriladi. **Inflate()** metodi yuklanishi kerak bo'lgan resurs identifikatori, shuningdek shakllantiriladigan menyu obyekti.

Menyu va vazifalar panelidagi amallarni boshqarish. Menyu va vazifalar panelidagi amallarni bajarish uchun sinfning onOptionsItemSelected() metodini bekor qilish kerak. Ushbu metod dispetcherdir, shuning uchun uni amalga oshirishning odatiy metodi tanlangan harakatni aniqlash va keyin ushbu harakatni qayta ishlashdir:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
```



```
switch (item.getItemId()) {
case R.id.menu_save:
// Handle the "save" operation
break;
case R.id.menu_delete:
// Handle the "delete" operation
break;
// ...
}
return true;
}
```

O'z-o'zini tekshirish uchun savollar va mashqlar

1. Resurslar nima? Ushbu mexanizm Androidda qanday vazifalar uchun ishlab chiqilgan? Resurs mexanizmidan foydalanishning qanday afzalliklari bor?

2. Resurslarning qanday turlari mavjud? Resurs fayllari loyihada qanday joylashgan?

3. Ilovangizdagi resurslardan, shuningdek, boshqa resurslardan qanday foydalanish mumkin?

4. Konfiguratsiyaga bog'liq resurslar nima? Ushbu mexanizmning maqsadi nima va undan qanday foydalanish mumkin?

5. Resurs mexanizmidan foydalanib, Android ilovasining asosiy menyusi yoki paneli qanday yaratiladi?

6. Android platformasining dastlabki va keyingi versiyalari uchun menyuni amalga oshirishda qanday farqlar bor?

7. Asosiy menyu yoki paneldan amallarni tanlash qanday amalga oshiriladi?

8-BOB. MA'LUMOTLARNI SAQLASH.

Ko'pgina ilovalar ma'lumotlarni doimiy xotirada saqlashi va keyingi ishga tushirilganda ularni qayta tiklashi kerak. Android platformasi ushbu vazifa uchun uchta variantni taqdim etadi: afzalliklar, fayl tizimi va ma'lumotlar bazalari. Keling, ushbu imkoniyatlarni batafsil ko'rib chiqaylik.

1-§. Sozlamalar mexanizmi.

SQLite, DBMS, MainActivity, onPause, onResume, SharedPreferences, Kontekst.

Nomidan ko'rinib turibdiki, ushbu mexanizmning asosiy maqsadi dastur sozlamalarini saqlashdir. Android API sozlamalarni kalit-qiymat juftligi sifatida saqlash imkonini beradi va ushbu sozlamalar saqlanadigan fayllarni yaratish va boshqarish bo'yicha barcha vazifalarni avtomatik tarzda bajaradi.

Afzalliklari bilan ishlashni boshlash uchun umumiy **SharedPreferences** **getSharedPreferences** (String nomi, int rejimi) orqali **SharedPreferences** sinfining ob'ektini olish kerak;

Kontekst klass. Sozlamalar identifikatori birinchi argument sifatida uzatiladi. Ikkinchi argument kirish rejimini belgilaydi.

Aksariyat hollarda standart rejimdan foydalanish (**Context.MODE PRIVATE**) kifoya qiladi. Ushbu parametr uchun boshqa qiymatlar bir nechta ilovalar tomonidan ulashiladigan sozlamalarni yaratishga imkon beradi.

SharedPreferences sinfining ob'ekti olingandan so'ng, metodlar yordamida oldindan yozib olingan sozlamalar olinishi mumkin.

public String getString(String key, String defaultValue);

public int getInt(String key, int defaultValue);

...

Qabul qilingan parametrlarning har biri kalit (asosiy parametr) bilan aniqlanadi. Agar so'ralgan qiymat faylda topilmasa, standart qiymat qaytariladi - u *get* metodi chaqiruviga ikkinchi argument sifatida o'tkaziladi.

Sozlamalarni saqlash biroz qiyinroq. Birinchidan, metodni chaqirish orqali **SharedPreferences.Editor** interfeysini amalga oshiradigan sinf ob'ektini olish kerak.

public SharedPreferences.Editor edit();

Olingan ob'ektda mos keladigan turdagi sozlamalarni saqlashga imkon beruvchi metodlar mavjud.

public SharedPreferences.Editor putString(String key, String value);

public SharedPreferences.Editor putInt (String key, int value);

...

Agar yozishni amalga oshiradigan kalitga mos keladigan parametrning qiymati sozlamalar faylida allaqachon mavjud bo'lsa, bu qiymat qayta yoziladi.

public boolean commit();

Ma'lumotlarni yozib olish tugallangandan so'ng, sozlamalar faylidagi o'zgarishlarni avtomik tarzda saqlaydigan metodni chaqirish kerak.

Keling, quyidagi misol bilan sozlash mexanizmini qo'llashni ko'ramiz. Avvalgi boblarda keltirilgan Hisoblagich ilovasmi ko'rib chiqamiz va dasturni ishga tushirish oralig'ida hisoblagich holatini saqlash imkoniyatini qo'shamiz. Umuman olganda, dastur sozlamalariga qaramasdan, ushbu mexanizmdan foydalanish muammoning eng oddiy yechimi hisoblanadi.

Biz holatni **onPause()** metodida saqlaymiz va **onResume()** metodida tiklaymiz. Holat tiklanganligini ta'minlash uchun biz **Counter** sinfida sozlash metodini aniqlaymiz:

```
public void setValue(int value) {
this.value = value;
if (listener != null) { listener.onModification(this); }
}
```

8-BOB. MA'LUMOTLARNI SAQLASH.

Ko'pgina ilovalar ma'lumotlarni doimiy xotirada saqlashi va keyingi ishga tushirilganda ularni qayta tiklashi kerak. Android platformasi ushbu vazifa uchun uchta variantni taqdim etadi: afzalliklar, fayl tizimi va ma'lumotlar bazalari. Keling, ushbu imkoniyatlarni batafsil ko'rib chiqayllk.

1-§. Sozlamalar mexanizmi.

SQLite, DBMS, MainActivity, onPause, onResume, SharedPreferences, Kontekst.

Nomidan ko'rinib turibdiki, ushbu mexanizmming asosiy maqsadi dastur sozlamalarini saqlashdir. Android API sozlamalarni kalit-qiymat juftligi sifatida saqlash imkonini beradi va ushbu sozlamalar saqlanadigan fayllarni yaratish va boshqarish bo'yicha barcha vazifalarni avtomatik tarzda bajaradi.

Afzalliklari bilan ishlashni boshlash uchun umumiy **SharedPreferences** **getSharedPreferences** (String nomi, int rejimi) orqali **SharedPreferences** sinfming ob'ektini olish kerak;

Kontekst klass. Sozlamalar Identifikatori birinchi argument sifatida uzatiladi. Ikkinchi argument kirish rejimini belgilaydi.

Aksariyat hollarda standart rejimdan foydalanish (**Context.MODE PRIVATE**) kifoya qlladi. Ushbu parametr uchun boshqa qiymatlar hir nechta ilovalar tomonidan ulashiladigan sozlamalarni yaratishga imkon beradi.

SharedPreferences sinfining ob'ekti olingandan so'ng, metodlar yordamida oldindan yozib olingan sozlamalar olinishi mumkin.

public String getString(String key, String defaultValue);

public int getInt(String key, int defaultValue);

...

Qabul qilingan parametrlarning har biri kallt (asosiy parametr) bilan aniqlanadi. Agar so'ralgan qiymat faylda topilmasa, standart qiymat qaytariladi - u *get* metodi chaqiruviga ikkinchi argument sifatida o'tkaziladi.

Sozlamalarni saqlash biroz qiyinroq. Birinchidan, metodni chaqirish orqali **SharedPreferences.Editor** interfeysini amalga oshiradigan sinf ob'ektini olish kerak.

public SharedPreferences.Editor edit();

Olingan ob'ektda mos keladigan turdagi sozlamalarni saqlashga imkon beruvchi metodlar mavjud.

public SharedPreferences.Editor putString(String key, String value);

public SharedPreferences.Editor putInt (String key, int value);

...

Agar yozishni amalga oshiradigan kalitga mos keladigan parametrning qiymati sozlamalar faylida allaqachon mavjud bo'lsa, bu qiymat qayta yoziladi.

public boolean commit();

Ma'lumotlarni yozib olish tugallangandan so'ng, sozlamalar faylidagi o'zgarishlarni avtomik tarzda saqlaydigan metodni chaqirish kerak.

Keling, quyidagi misol bilan sozlash mexanizmini qo'llashni ko'ramiz. Avvalgi boblarda keltirilgan Hisoblagich ilovasini ko'rib chiqamiz va dasturni ishga tushirish oralig'ida hisoblagich holatini saqlasb imkoniyatini qo'shamiz. Umuman olganda, dastur sozlamalariga qaramasdan, ushbu mexanizmdan foydalanish muammoning eng oddiy yechimi hisohlanadi.

Biz holatni **onPause()** metodida saqlaymiz va **onResume()** metodida tiklaymiz. Holat tiklanganligini ta'minlash uchun biz **Counter** sinfida sozlash metodini aniqlaymiz:

```
public void setValue(int value) {
this.value = value;
if (listener != null) { listener.onModification(this); }
}
```

MainActivity sinfidan yuqridagi bo'limda kiritilgan burish paytida hisoblagich holatini saqlash va tiklash kodini olib tashlash kerak, chunki bu vazifa dasturning yangi versiyasida avtomatik ravishda hal qilinadi. O'zgartirilgan **MainActivity** sinfining to'liq kodi quyidagicha bo'ladi:

```
public class MainActivity extends Activity {
private TextView counterText;
private Counter counter = new Counter();
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
counterText = (TextView) findViewById(R.id.counterText);
counter.setOnModificationListener(
new Counter.OnModificationListener() {
@Override
public void onModification(Counter sender) { updateCounterView(); }
});
}
public void updateCounterView() {
counterText.setText(String.valueOf(counter.getValue()));
}
public void onIncreaseBu  onClick(View v) {
counter.increase();
}
public void onResetBu  onClick(View v) {
counter.reset();
}
@Overrlde
protected void onPause() {
```



```
super.onPause();
SharedPreferences prefs = getSharedPreferences(getLocalClassName(),
Context.MODE PRIVATE);
SharedPreferences.Editor editor = prefs.edit();
editor.putInt("counterValue", counter.getValue());
editor.commit();
}
@Override
protected void onResume() {
super.onResume();
SharedPreferences prefs = getSharedPreferences(getLocalClassName(),
Context.MODE PRIVATE);
counter.setValue(prefs.getInt("counterValue", 0));
}
}
```

SQLite DBMS ning ishlashi uchun asosiy sinflar. SQLite - bu Androiddа qo'llab-quvvatlanadigan o'rnatilgan DBMS. Uning ilovalarda ishlatilishi ikkita Android API klassiga tayanadi: SQLiteDatabase va SQLiteOpenHelper.

Birinchisi, ma'lumotlar bazasiga kirish operatsiyalarini, jumladan, jadvallarga ma'lumotlarni qo'shish, o'zgartirish, o'chirish, ma'lumotlarni olish uchun so'rovlar va ma'lumotlar bazasi strukturasini boshqarishni qamrab oladi. Ikkinchi sinf yordamchi bo'lib, ma'lumotlar bazasini boshqarish uchun mo'ljallangan, shu jumladan ma'lumotlar sxemasini dastlabki yaratish va dasturni yangilashda ushbu sxemani yangilash amalga oshiriladi.

2-§. Ma'lumotlar bazasini boshqarish.

SQLiteOpenHelper, name, version, null, SQLiteDatabase, onUpdate, sinf maydoni.

Android ilovasini ishlab chiqaruvchisi ilova tomonidan talab qilinadigan ma'lumotlar bazasini foydalanishdan oldin yaratilganligi va yangilanganligini ta`minlash uchun mas`uldir. Buning uchun mavhum **SQLiteOpenHelper** sinfidan meros bo'ladigan sinf yaratish, konstruktorni belgilash va quyidagi metodlarni belgilash kerak:

public void onCreate(SQLiteDatabase JDB);

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion);

Meroslangan sinf konstruktori odatda tegishli parametr qiymatlari bilan superklass konstruktorini chaqiradi:

SQLiteOpenHelper (Context context, String name,

SQLiteDatabase.CursorFactory factory, int version);

Kontekst sifatida joriy kontekst uzatiladi (faqat jarayon sinfi ob'ektiga havolani yuborish mumkin), **name** - ma'lumotlar bazasi nomini o'z ichiga oladi, **version** - ma'lumotlar bazasining versiya raqami odatda **null**ga o'rnatiladi.

OnCreate() metodi ilova o'rnatilgandan keyin ma'lumotlar bazasiga birinchi kirish sodir bo'lganda chaqiriladi. Ushbu metodning amalga oshirilishi **SQLiteDatabase** sinfi ob'ektida chaqirish orqali ma'lumotlar sxemasini yaratuvchi **SQL create** buyrug'ini bajarish orqali amalga oshiriladi.

public void execSQL(String sql);

OnUpdate() metodi ilova yangilanganda ma'lumotlar bazasiga o'zgartirishlar kiritish uchun mo'ljallangan. Tizimdagi ma'lumotlar bazasining joriy versiyasi **SQLiteOpenHelper** klassi konstruktoriga versiya berilgan raqamga mos kelmasa parametr sifatida chaqiriladi. Eski va yangi versiya raqamlari argument sifatida **onUpdate()** metodiga uzatiladi. Ularni taqqoslash asosida

dasturchi ma'lumotlar sxemasini eski versiyadan yangisiga o'zgartirish buyruqlarini bajarishi mumkin.

Odatda, **SQLiteOpenHelper**-dan olingan sinf ob'ekti **onCreate()** metodida yaratiladi va sinf maydoniga joylashtiriladi.

public SQLiteDatabase getWritableDatabase();

Keyinchalik, ushbu ob'ektda **SQLiteDatabase** sinfining ob'ektini qaytaradigan metod chaqiriladi, bu orqali ma'lumotlarga kirish mumkin. Oxirida metodni chaqirish orqali ma'lumotlar bazasini yopish lozim:

public void close();

3-§. Ma'lumotlarga kirish.

assotsiativ massiv, ContentValues, put, insert, WhereClause, joker, whereArgs, groupBy, have, orderBy, limit.

SQLiteDatabase klassi ma'lumotlarga kirishning ko'plab metodlarini taqdim etadi. Keling, asosiylarini ko'rib chiqaylik.

public long insert(String table, String nullColumnHack, ContentValues values);

Usul ma'lumotlar bazasi jadvaliga qator kiritish uchun mo'ljallangan. Jadval nomi jadval parametri sifatida, kiritiladigan qiymatlar esa qiymatlar parametri sifatida uzatiladi. Qo'shilgan qatorlar soni yoki qo'shish muvaffaqiyatsiz bo'lsa -1 ni qaytaradi.

Parametr qiymatlarini saqlash uchun ishlatiladigan **ContentValues** klassi assotsiativ massiv bo'lib, unda kalitlar jadval ustunlarining nomlari va qiymatlar mos keladigan kataklarning ma'lumotlari hisoblanadi. Assotsiativ massivga kalit-qiymat juftligini yozish uchun standart **Java** assotsiativ massivlariga o'xshash **put()** metodidan foydalaniladi.

Usul ma'lumotlar bazasidagi yozuvni o'zgartirish uchun mo'ljallangan.

public int update(String table, ContentValues values,

String whereClause, String[] whereArgs);

Jadval va qiymatlar parametrlari **insert()** metodidagi kabi bo`ladi. **WhereClause** parametri yangilanadigan yozuvlarni tanlaydigan **SQL** ifodasini o'z ichiga oladi. Bu ifoda savol belgilari bilan belgilangan **joker** belgilar parametrlarini o'z ichiga olishi mumkin. Ushbu parametrlarning qiymatlari **SQL** bayonotida paydo bo'lgan tartibda **whereArgs** ga massiv sifatida uzatiladi.

public int delete(String jadvali, String whereClause, String[] whereArgs);

Usul ma'lumotlar bazasidan yozuvlarni o'chirish uchun mo'ljallangan.

Bitta ma'lumotlar bazasi jadvalidan ma'lumotlarni olish uchun ushbu metoddan foydalaniladi:

public Cursor query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit);

Usul parametrlari quyidagilarni anglatadi:

• **table** – tanlash amalga oshiriladigan jadvalning nomi;

• **columns** — natijada qaytariladigan ustunlar ro'yxati

so'rov (**null** barcha ustunlarni bildiradi);

• **selection** — SQL so'rovining **where** bandi uchun SQL ifodasi

har bir namuna uchun (almashtirish parametrlarini) o'z ichiga olishi mumkin:

• **selectionArgs** — almashtirish parametrlarining qiymatlari;

• **groupBy**, **have**, **orderBy**, **limit** — guruh uchun SQL ifodalari, ma`lumotlar olish so`rovi bo`yicha, egalik, tartib va chegara konstruksiyalari.

Parametrlarning aksariyati ixtiyoriy va **null** bo'lishi mumkin.

Agar bir nechta jadvallardan ma'lumotlarni olish kerak bo'lsa, umumiyroq metod qo'llaniladi.

public Cursor rawquery(String sql, String[] selectionArgs);

Bu metod SQL so'rovini qabul qiladi, u to'g'ridan-to'g'ri bajarish uchun DBMSga uzatiladi.

4-§. Kursorlar bilan ishlash.

Query, rawquery, Cursor, ContentValues, put, insert, WhereClause, groupBy, have, orderBy, limit.

Query() va **rawquery()** metodlari natijalar to`plami uchun foydalaniladigan **Cursor** interfeysini amalga oshiradigan sinf ob'yektini qaytaradi.

public boolean moveToNext();

Dastlab, **Cursor** ma'lumotlar to'plamining birinchi qatoridan oldingi holatda bo'ladi. Keyingi pozitsiyaga o'tish uchun, agar o'tish muvaffaqiyatli bo'lsa, **true** va ma'lumotlar to'plami tugagan bo'lsa, **false** qaytaradigan metod qo'llaniladi.

Kursor "passed over" qator maydonlarini olish uchun olingan ma'lumotlar turiga qarab quyidagi metodlardan biri qo'llaniladi.

public int getInt(int columnIndex);

public long getLong(int columnIndex);

public String getString(int columnIndex);

...

columnIndex parametri sifatida qiymati olinadigan ustunning tartib raqami yuqoridagi metodlarga uzatiladi. Natijada olingan ma'lumotlar to'plamining ustun raqami, shuningdek uning turi quyidagi metodlardan foydalangan holda nom bilan olinishi mumkin:

public int getColumnIndex(String columnName);

public int getType(int columnIndex);

Ma'lumotlar turlari **Cursor** interfeysining statik konstantalari sifatida aniqlanadi.

Bir misolni ko'rib chiqamiz. Quyidagi funktsiya jadval mazmunini jurnal fayliga chop etadi (bu ilovani tuzatish uchun foydali bo'lishi mumkin). Oddiylik uchun jadval ustunlari faqat butun son yoki satr bo'lishi mumkin deb olinadi.

private void printTable(SQLiteDatabase database, String tableName) {

Cursor cursor = database.query(tableName, null, null, null, null,

```
null, null, null);
while (cursor.moveToNext()) {
Log.d(getLocalClassName(), "Record:");
for (int i = 0; i < cursor.getColumnCount(); i++) {
String columnName = cursor.getColumnName(i) + ": ";
switch (cursor.getType(i)) {
case Cursor.FIELD_TYPE_INTEGER:
Log.d(getLocalClassName(), columnName + cursor.getInt(i));
break;
case Cursor.FIELD_TYPE_STRING:
Log.d(getLocalClassName(), columnName + cursor.getString(i));
break;
}
}
}
}
```

O'z-o'zini tekshirish uchun savollar va mashqlar

1. Android platformasida ma'lumotlarni doimiy saqlash metodlarini sanab o'ting. Ularning har biridan qachon foydalanish maqsadga muvofiqligini tushuntiring.

2. Sozlash mexanizmi nima? Bu nima uchun? Uni qanday qo'llash kerak?

3. SQLite ma'lumotlar bazasi bilan ishlash uchun mo'ljallangan Android sinflarini sanab o'ting. Ulardan qanday foydalanishni misollar bilan tushuntiring.

4. Ma'lumotlar bazasi nimadan iborat? Android platformasidagi qanday vositalar boshqarishga imkon beradi?

5. SQLiteDatabase sinfining ma'lumotlar bilan ishlash metodlarini ayting. Ulardan qanday foydalanish mumkinligini misollar bilan tushuntiring.

*6. SQLiteDatabase sinfining **query()** va **rawquery()** metodlarining farqi nimada? Ularning har biri qanday hollarda qo'llaniladi?*

7. Ma'lumotlarni tanlash kursori nima? Bu nima uchun? Kursorlardan qanday foydalanishni misollar bilan tushuntiring.

9-BOB. MA'LUMOTLARNI SAQLASH UCHUN MA'LUMOTLAR BAZASIDAN FOYDALANADIGAN DASTUR YARATISH.

Ushbu bobda foydalanuvchining vazifalar ro'yxatini saqlash uchun ma'lumotlar bazasidan foydalanadigan ilovalarni ko'rib chiqamiz. Misol sifatida ma'lumotlar bazasidan ilovada qanday foydalanish mumkinligini ko'rsatish uchun mo'ljallangan dasturlar, jumladan, ma'lumotlar bazasini boshqarish, ma'lumotlar bazasida yozuvlarni qo'shish, o'zgartirish va olish, ma'lumotlarni ro'yxatda ko'rsatish va alohida ma'lumotlarni tahrirlash kabi amallarni bajarish ishlarini qarab chiqamiz.

Ilova ikkita holatni o'z ichiga oladi. Asosiysi esa, vazifalar ro'yxatini ko'rsatishdir. Ikkinchisi foydalanuvchi vazifasi atributlarini, jumladan, sarlavha, tavsif va tugash sanasini tahrirlash imkonini beradi. Ilovaning ko'rinishi 9.1-rasmda ko'rsatilgan.

1-§. Ma'lumotlar bazasining hayot aylanishini boshqarish klassi.

Xronologik, kontekst, todos, DBHelper, MainActivity, SimpleCursorAdapter.

Siklni boshqarish uchun **SQLiteOpenHelper** sinfidan olingan ilovada **DBHelper** sinfini aniqlaymiz:

```
public class DBHelper extends SQLiteOpenHelper {
public DBHelper(Context context) {
super(context, "todos", null, 1);
}
@Override
public void onCreate(SQLiteDatabase db) {
```

```
db.execSQL("create table todos (" +
" id integer primary key autoincrement," +
"title text," +
"description text," +
"dueDate text);"
);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
}
```

9.1-rasm. Vazifalar ro'yxati ilovasidagi asosiy jarayon va muharrir faoliyati

Ushbu sinf ma'lumotlar bazasiga kirishga birinchi urinishda ma'lumotlar sxemasini yaratadi. Ilova bitta versiyaga ega bo'lgani uchun **SQLiteOpenHelper** sinf konstruktoridagi versiya raqami 1 ga o'rnatiladi va **onUpgrade()** metodi bo'sh qoladi.

Har bir topshiriqning sarlavhasi (**title**), tavsifi (**description**) va tugash sanasi (**dueDate**) mavjud. **SQLite**-da sanalarni saqlash uchun maxsus ma'lumotlar turi

mavjud emasligi sababli, ISO 8601 formatidagi (masalan, 2022-01-21) qator ko'rinishi ishlatiladi. Asosiy kalit uchun **id** deb nomlangan maydon yaratildi. Bu nom ma'lumotlar bazasi jadvali bilan bog'langan ro'yxat adapterining to'g'ri ishlashi uchun zarur bo`ladi.

Foydalanuvchi interfeysi. Asosiy interfeysning tavsifi **res/layout/main.xml** faylida joylashgan bo`lib, bitta elementni o'z ichiga oladi, va u **todoList** identifikatorli ro'yxat deyiladi:

```
<?xml version="1.0" encoding="utf-8"?>
<ListView android:id="@+id/todoList"
android:layout width="fill parent"
android:layout height="fill parent"
xmlns:android="http://schemas.android.com/apk/res/android">
</ListView>
```

Activity-da boshlash. Asosiy jarayon **MainActivity** sinfida joylashgan. Ilovaga funksiya qo'shganda biz ushbu sinfning mazmunini alohida qismlarga ajratamiz. Aktiv jarayon uchun ishga tushirish kodidan boshlaylik:

```
public class MainActivity extends Activity {
private DBHelper dbHelper;
private Cursor cursor;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
ListView todoListView = (ListView) findViewById(R.id.todoList);
todoListView.setOnItemClickListener(new ListView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
onToDoListItemClick(id);
```

```
db.execSQL("create table todos (" +
" id integer primary key autoincrement," +
"title text," +
"description text," +
"dueDate text);"
);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
}
}
```

9.1-rasm. Vazifalar ro'yxati ilovasidagi asosiy jarayon va muharrir faoliyati

Ushbu sinf ma'lumotlar bazasiga kirishga birinchi urinishda ma'lumotlar sxemasini yaratadi. Ilova bitta versiyaga ega bo'lgani uchun **SQLiteOpenHelper** sinf konstruktoridagi versiya raqami 1 ga o'rnatiladi va **onUpgrade()** metodi bo'sh qoladi.

Har bir topshiriqning sarlavhasi **(title)**, tavsifi **(description)** va tugash sanasi **(dueDate)** mavjud. **SQLite**-da sanalarni saqlash uchun maxsus ma'lumotlar turi

mavjud emasligi sababli, ISO 8601 formatidagi (masalan, 2022-01-21) qator ko'rinishi ishlatiladi. Asosiy kalit uchun **id** deb nomlangan maydon yaratildi. Bu nom ma'lumotlar bazasi jadvali bilan bog'langan ro'yxat adapterining to'g'ri ishlashi uchun zarur bo`ladi.

Foydalanuvchi interfeysi. Asosiy interfeysning tavsifi **res/layout/main.xml** faylida joylashgan bo`lib, bitta elementni o'z ichiga oladi, va u **todoList** identifikatorli ro'yxat deyiladi:

```
<?xml version="1.0" encoding="utf-8"?>
<ListView android:id="@+id/todoList"
android:layout width="fill parent"
android:layout height="fill parent"
xmlns:android="http://schemas.android.com/apk/res/android">
</ListView>
```

Activity-da boshlash. Asosiy jarayon **MainActivity** sinfida joylashgan. Ilovaga funksiya qo'shganda biz ushbu sinfning mazmunini alohida qismlarga ajratamiz. Aktiv jarayon uchun ishga tushirish kodidan boshlaylik:

```
public class MainActivity extends Activity {
private DBHelper dbHelper;
private Cursor cursor;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
ListView todoListView = (ListView) findViewById(R.id.todoList);
todoListView.setOnItemClickListener(new ListView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
onToDoListItemClick(id);
```

```
}
});
dbHelper = new DBHelper(this);
cursor = dbHelper.getWritableDatabase().query("todos", null, null, null,
null, null, "dueDate");
String[] from = new String[] { "title", "description" };
int[] to = new int[] { R.id.titleText, R.id.descriptionText };
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
R.layout.todo_item, cursor, from, to,
CursorAdapter.FLAG_AUTO_REQUERY);
todoListView.setAdapter(adapter);
}
// ...
```

Aktiv jarayon ishga tushirilganda, interfeys tavsifi XML faylidan yuklanadi (3-bo'limga qarang), elementni tanlash hodisasi ishlov beruvchisi ro'yxatga biriktiriladi, **DBHelper** sinfining namunasi yaratiladi va sinf maydoniga joylashtiriladi. Keyinchalik, barcha vazifalar ma'lumotlar bazasining **todos** jadvalidan olinadi va xronologik tartibda sanasi bo'yicha tartiblanadi. Buyruqni bajarish natijasida kursor qaytariladi va u ham **MainActivity** klassi maydonida saqlanadi. Keyinchalik, **SimpleCursorAdapter** sinfining namunasi yaratilatilib, ma'lumotlar to'plamining yozuvlarini ro'yxat elementining matn maydonlariga moslashtiriladi.

Ro'yxat elementining ko'rinishi **res/layout/todo_item.xml** fayli bilan tavsiflanadi. Har bir elementda vazifa nomi uchun maydon va uning tavsifi uchun maydon mavjud:

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



```
android:layout_width="match_parent"
android:layout_height="match_parent">
<TextView android:id="@+id/titleText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="le  |center_vertical"/>
<TextView android:id="@+id/descriptionText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#808000"
android:layout_gravity="le  |center_vertical"/>
</LinearLayout>
```

2-§. Ilova menyusi va yozuvni qo'shishni qayta ishlash.

Xronologik, kontekst, todos, DBHelper, MainActivity, SimpleCursorAdapter.

Ilova menyusi (harakat paneli formatida) **res/menu/main.xml** faylida tasvirlangan va vazifa qo'shish tugmasidan iborat:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/add_todo"
android:showAsAction="always"
android:icon="@android:drawable/ic_menu_add" />
</menu>
```

Menyuni ishga tushirish va yozuvni qo'shish tugmachasi bilan boshqarish **MainActivity** sinfida standart tarzda amalga oshiriladi.

```
// ...
@Override
```

```
public boolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
Intent intent = new Intent(this, ToDoEditorActivity.class);
startActivityForResult(intent, 1);
return true;
}
// ...
```

Tugmani bosish, yozuvni tahrirlash uchun **ToDoEditorActivity** ni chaqirishga olib keladi. Harakatlar panelida faqat bitta tugma mavjud bo'lganligi sababli, **onOptionsItemSelected()** metodi tanlangan menyu elementini tekshirish uchun hech qanday kodni o'z ichiga olmaydi.

Tahrirlovchi interfeysi. Vazifa atributlarini tahrirlash uchun mo'ljallangan interfeysi vazifa nomini va uning tavsifini, matn maydonlarini, vazifani bajarish sanasini kiritish uchun **DatePicker** komponentini, ikkita "**OK**" va "**Cancel**" tugmalarini, shuningdek teglarni (**TextView**) o'z ichiga oladi.

Res/layout/todo editor.xml fayli foydalanuvchi interfeysini tavsiflash uchun ishlatiladi. Bu quyidagicha keltiriladi:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout width="match parent"
android:layout height="match parent">
<TextView android:text="Title"
android:layout width="wrap content"
```



```
android:layout height="wrap content"
android:layout gravity="left|center vertical"/>
<EditText android:id="@+id/titleText"
android:layout width="fill parent"
android:layout height="wrap content"
android:layout gravity="left|center vertical"/>
<TextView android:text="Description"
android:layout width="wrap content"
android:layout height="wrap content"
android:layout gravity="left|center vertical"/>
<EditText android:id="@+id/descriptionText"
android:layout width="fill parent"
android:layout height="wrap content"
android:layout gravity="left|center vertical"/>
<TextView android:id="@+id/textView" android:text="Due date"
android:layout width="wrap content"
android:layout height="wrap content"
android:layout gravity="left|center vertical"/>
<DatePicker android:id="@+id/dueDatePicker"
android:layout width="fill parent"
android:layout height="wrap content" />
<LinearLayout android:orientation="horizontal"
android:layout width="fill parent"
android:layout height="fill parent">
<Button android:id="@+id/cancelButton" android:text="Cancel"
android:layout width="fill parent"
android:layout height="wrap content"
android:layout weight="1"
android:layout gravity="center horizontal|bottom"
```

```
android:onClick="onCancelButtonClick" />
<Button android:id="@+id/okButton" android:text="OK"
android:layout width="fill parent"
android:layout height="wrap content"
android:layout weight="1"
android:layout gravity="center horizontal|bottom"
android:onClick="onOkButtonClick" />
</LinearLayout>
</LinearLayout>
```

3-§. Jarayonning o'zaro ta'siri interfeysi.

String title, description, title, ToDoEditorActivity, MainActivity, SimpleCursorAdapter.

Tahrirlovchi faoliyatini amalga oshirishga o'tishdan oldin biz ushbu jarayon va ilovaning asosiy faoliyati o'rtasidagi interfeysni aniqlashimiz kerak. Tahrirlovchi faoliyatidan ikki xil foydalanish mumkin: yangi yozuv yaratish yoki mavjudni tahrirlash.

Keling, yangi yozuvni yaratishda asosiy faoliyatdan hech qanday ma'lumot o'tkazilmasin, qo'shimcha maydonlar (qo'shimchalari) orqali tahrirlashda quyidagi ma'lumotlar tahrirlashga o'tkaziladi:

- **int id** — ma'lumotlar bazasida tahrir qilinayotgan yozuvning identifikatori;
- **String title** — tahrirlangan vazifaning sarlavhasi;
- **String description** — tahrirlangan vazifa tavsifi;
- **String dueDate** - ISO 8601 formatidagi tahrirlangan topshiriqning tugash sanasi.

Agar foydalanuvchi **OK** tugmasini bosish orqali o'zgarishiarni qabul qilgan bo'lsa, tahrirlovchining faoliyati **RESULT OK** ni va agar foydalanuvchi **Bekor** qilish tugmasi yoki qurilmaning uskunaviy **Orqaga** tugmasi yordamida bekor qilgan bo'lsa, **RESULT CANCELED**ni qaytaradi.

Agar foydalanuvchi o'zgarishlarni qabul qilgan bo'lsa, qo'shimcha maydonlar mexanizmi orqali quyidagi qiymatlar qaytariladi:

- **int id** — ma'lumotlar bazasida tahrir qilinayotgan yozuvning identifikatori (faqat vazifani tahrirlash uchun chaqirilgan bo'lsa; aks holda bu qiymat o'tkazilmaydi);
- **String title** — qo'shilgan/tahrirlangan vazifaning sarlavhasi;
- **String description** — qo'shilgan/tahrirlangan vazifa tavsifi;
- **String dueDate** – ISO 8601 formatidagi qo'shilgan/tahrirlangan topshiriqning tugash sanasi.

Bu yerda, hujjatlashtirish ilovalarni ishlab chiqish uchun juda muhimdir. **Android**da interfeyslarni tavsiflashning standartlashtirilgan mexanizmi mavjud bo'lmasa, yuqoridagi kabi ma'lumotlarni uzatish muammosini muvaffaqiyatli hal qilishi mumkin.

Vazifa muharriri. Tahrirlovchi **ToDoEditorActivity** sinfiga joylashtirilgan:

```
public class ToDoEditorActivity extends Activity {
private EditText titleText, descriptionText;
private DatePicker dueDatePicker;
private Intent resultIntent = new Intent();
private static final SimpleDateFormat dateFormat
= new SimpleDateFormat("yyyy-MM-dd");
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.todo editor);
titleText = (EditText) findViewByld(R.id.titleText);
```

```
descriptionText = (EditText) findViewById(R.id.descriptionText);
dueDatePicker = (DatePicker) findViewById(R.id.dueDatePicker);
if (getIntent().hasExtra("id")) {
resultIntent.putExtra("id", getIntent().getIntExtra("id", 0));
titleText.setText(getIntent().getStringExtra("title"));
descriptionText.setText(getIntent().getStringExtra("description"));
GregorianCalendar calendar = stringToDate(
getIntent().getStringExtra("dueDate"));
dueDatePicker.init(calendar.get(Calendar.YEAR),
calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY OF MONTH), null);
}
}

private static String dateToString(int year, int month, int day) {
GregorianCalendar calendar = new GregorianCalendar(year, month, day);
return dateFormat.format(calendar.getTime());
}

private static GregorianCalendar stringToDate(String dateString) {
try {
Date date = dateFormat.parse(dateString);
GregorianCalendar calendar = new GregorianCalendar();
calendar.setTime(date);
return calendar;
} catch (ParseException e) {
return null;
}
```



```
}

public void onOkBu  onClick(View v) {
resultIntent.putExtra("title", titleText.getText().toString());
resultIntent.putExtra("description", descriptionText.getText().toString());
resultIntent.putExtra("dueDate", dateToString(dueDatePicker.getYear(),
dueDatePicker.getMonth(), dueDatePicker.getDayOfMonth()));
setResult(RESULT_OK, resultIntent);
finish();
}

public void onCancelButtonClick(View v) {
setResult(RESULT_CANCELED);
finish();
}
}
```

OnCreate() metodi muharrir interfeysini ishga tushiradi, sinf sohalarida interfeys komponentlariga havolalarni saqlaydi va agar aktiv rejimida ochilgan bo'lsa, qiymatlarni mos keladigan vidjetlarga o'tkazadi (bu qo'shimcha parametrlarda juftlik identifikatori mavjudligi bilan belgilanadi). **OK** tugmasi ma'lumotlarni vidjetlardan asl holga qaytaradi. Bundan tashqari, **Cancel** tugmasi qaytarilgan qiymatini o'rnatadi va **finish()** chaqiruvi bilan yakunlaydi.

ISO 8601 formatidagi satrlardan alohida sana komponentlarini ajratib olish va alohida komponentlardan bunday qatorlarni yaratish uchun ikkita yordamchi metod belgilangan: **dateToString()** va **stringToDate()**.

Mavjud vazifani o'zgartirish uchun muharrir. Muharrir allaqachon yuqoridagi bo'limda ko`rilgan. Ikkinchi holatda ro'yxatdagi elementlardan biriga

tegsa, tahrirlash amalga oshiriladi. Bu harakat faoliyatning **onToDoListItemClick()** metodida bajariladi:

```
// ...
public void onToDoListItemClick(long id) {
Cursor todoCursor = dbHelper.getReadableDatabase().query("todos", null,
" id = ?", new String[] { String.valueOf(id) }, null, null, null);
todoCursor.moveToNext();
Intent intent = new Intent(this, ToDoEditorActivity.class);
intent.putExtra("id", todoCursor.getInt(
todoCursor.getColumnIndex(" id")));
intent.putExtra("title", todoCursor.getString(
todoCursor.getColumnIndex("title")));
intent.putExtra("description", todoCursor.getString(
todoCursor.getColumnIndex("description")));
intent.putExtra("dueDate", todoCursor.getString(
todoCursor.getColumnIndex("dueDate")));
startActivityForResult(intent, 1);
}
// ...
```

Ushbu identifikator bo'yicha ma'lumotlar bazasidan yozuvning mazmunini oladi va keyin qo'shimcha parametrlarini qabul qilingan yozuv maydonining qiymatlari bilan to'ldiradi, shundan so'ng u muharrirni aktivlashtiradi.

4-§. Aktiv muharrir bilan ishlash.

ContentValues, insert, update, requery, title, ToDoEditorActivity, MainActivity.

Muharrirdan qaytarilgan qiymatni qayta ishlashni ko'rib chiqaylik. Ushbu qayta ishlash asosiy aktiv sinfining **onActivityResult()** metodida amalga oshiriladi.

```
// ...
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
if (resultCode != RESULT_OK) return;
ContentValues cv = new ContentValues();
cv.put("title", data.getStringExtra("title"));
cv.put("description", data.getStringExtra("description"));
cv.put("dueDate", data.getStringExtra("dueDate"));
if (data.hasExtra("id")) {
dbHelper.getWritableDatabase().update("todos", cv, " id = ?",
new String[] { String.valueOf(data.getIntExtra("id", 0)) });
} else {
dbHelper.getWritableDatabase().insert("todos", null, cv);
}
cursor.requery();
}
} // class MainActivity
```

Agar foydalanuvchi kiritilgan o'zgarishlarni qabul qilmasa, tahrir uchun qaytariladi. Aks holda, o'zgartirilgan atributlar **ContentValues** sinfini ob'ektiga o'tkaziladi. Bundan tashqari, qo'shimcha parametrlarda **id** qiymatini bilan harakat aniqlanadi, ya'ni: yangi yozuvni qo'shish yoki mavjudni o'zgartirish. Natijada,

ma'lumotlar bazasi ob'ektida **insert()** metodi yoki **update()** metodi chaqiriladi. Usulning oxirgi qatori **requery()** metodini chaqiradi, bu esa vazifalar ro'yxatini ekranda yangilanishiga olib keladi.

O'z-o'zini tekshirish uchun savollar va mashqlar

1. Ushbu bobda keltirilgan dastur loyihasini yarating. Ilovani kompilyatsiya qiling va uni emulyatorda yoki haqiqiy qurilmada ishga tushiring.

2. Ilovaga yozuvlarni o'chirish imkoniyatini qo'shing.

3. Ro'yxat ko'rinishini shunday o'zgartiringki, u vazifani tavsiflash o'rniga, bajarilgunga qadar kunlar ko'rsatilsin.

***Ko'rsatma.** Kunlar sonini olish uchun **rawquery()** metodi va **SQLite** sana funksiyalaridan foydalaning va uni qulay tarzda taqdim eting (masalan, "1 kundan keyin to'lanadi", "2 kundan keyin to'lanadi", "muddati 4 kundan keyin o'tib ketgan" va hokazo).*

*4. Ma'lumotlar bazasiga topshiriq qo'shilgan sanani o'z ichiga olgan qo'shimcha maydon qo'shing. Birinchi versiya ma'lumotlar bazasini yangilash uchun **onUpgrade()** metodi va SQL o'zgartirish jadvali bayonidan foydalaning.*

10-BOB. ASINXRON BAJARILISH

Asinxron bajarish ba'zi harakatlarni bajarish uchun alohida ip-lardan foydalanishni o'z ichiga oladi. Asinxron bajarish zarurati ko'pincha juda yuqori vaqt xarajatlarini talab qiladigan ba'zi jarayonlar (hisob-kitoblar, tarmoq resurslariga kirish, ma'lumotlar bazasidan o'qish) mavjudligi bilan bog'liq.

Agar ushbu jarayonlarning bajarilishini alohida oqimlarga o'tkazilmasa, bu muqarrar ravishda foydalanuvchi interfeysining sezgirligiga ta'sir qiladi va dasturdan foydalanish qulayligini kamaytiradi.

Asinxron bajarishni tashkil qilish **Java** mexanizmiga (**Thread** class va **Runnable** interfeysi) tayanishi yoki standart ip-lar uchun yuqori darajadagi **Android**-ga xos **API**-lardan foydalanishi mumkin.

Asinxron bajarish bilan bog'liq muhim masala - bu ip-ni sinxronlashtirish. Bunga ehtiyoj **Android API**-ning, har qanday **GUI** kutubxonasi kabi, xavfsiz emasligidan kelib chiqadi. Ikkinchisi, vidjetlarga kirish oldindan aytib bo'lmaydigan oqibatlarga olib kelishi mumkinligini bildiradi.

Sinxronizatsiya muammosini hal qilish uchun Android API turli xil vositalarni, jumladan xabarlar, asinxron vazifalarni tashkil qilish uchun **AsyncTask** sinfini, shuningdek, ma'lumotlar bazasidan ma'lumotlarni asinxron yuklash kabi individual vazifalarni hal qilishga ixtisoslashgan mexanizmlarni taqdim etadi.

1-§. Ishlovchi sinfi va xabarlar navbati.

Android API, asinxron, AsyncTask, Handler, Message, Object, Parametr, Runnable, run, trek.

Handler class ip-lari bilan bog'liq xabarlarni boshqarish uchun mo'ljallangan. Xabarlar navbatga har qanday chizig'dan yuborilishi mumkin, lekin

ular har doim asosiy (foydalanuvchi interfeysi bilan bog'liq) ip-da qayta ishlanadi. Shunday qilib, **Handler** class **ip** sinxronizatsiyasini ta'minlaydi.

Xabar yuborish uchun quyidagi metodlar qo'llaniladi.

public boolean sendMessage(Message msg);

public boolean sendMessageDelayed(Message msg, long delayMillis);

public boolean sendMessageAtTime(Message msg, long uptimeMillis);

Xabar **Message** sinfining ob'ektidir. What, arg1 va arg2 nomli butun son xossalari, shuningdek **Object** tipidagi **obj** xossalari yuboriladigan xabarning tafsilotlarini saqlash uchun ishlatilishi mumkin. **Android API** bu xususiyatlar qanday qo'llanilishini belgilamaydi, shuning uchun dasturchi ulardan o'zi xohlagancha foydalanishi mumkin.

Operativ xotiradan foydalanishni optimallashtirish uchun yangi **Message** sinfi ob'ektlarini yaratilmasa, **Handler** sinfining statik metodlaridan biri tavsiya etiladi:

public Message obtainMessage(int what, int arg1, int arg2, Object obj);

public Message obtainMessage(int what);

public Message obtainMessage(int what, Object obj);

...

Ushbu metodlarni tashkil qilish orqali xabar ob'ektlaridan qayta foydalanishni ta'minlaydi. Parametr qiymatlari bo'lgan ob'ektni so'rovida, bu metodlar yangi ob'ekt yaratish o'rniga uni qaytaradi.

Xabarlarni qayta ishlash uchun siz **Handler** sinfidan o'z sinfingizni meros qilib olishingiz, metodni bekor qllishingiz va unda xabarni kodini joylashtirishingiz kerak, bu metodga argument sifatida uzatiladi.

public void handleMessage(Message msg);

Xabarlarni qayta ishlash ilovaning asosiy oqimida amalga oshirlladi, shuning uchun ishlov beruvchidan foydalanuvchi interfeysi elementlariga kirish mumkin.

Navbatga xabarlarni yuborishdan tashqari, **Runnable** interfeysini amalga oshiradigan sinflar ob'ektlarini qo'shish mumkin. Bu metodlar yordamida quyidagicha amalga oshiriladi:

public boolean post(Runnable r);

public boolean postDelayed(Runnable r, long delayMillis);

public boolean postAtTime (Runnable r, long uptimeMillis);

Ushbu yondashuvdan foydalanganda, o'tkazilgan ob'ektning **run()** metodi ishlov beruvchi vazifasini bajaradi.

Handler sinfidan foydalanishga misol. **Handler** sinfidan foydalanishga misol sifatida **Google** veb-xizmati yordamida qurilmaning tashqi **IP**-manzilini aniqlaydigan dasturni ko'rib chiqamiz.

Tarmoq so'rovini bajarish uchun vaqt kerak bo'lganligi sababli, bunday harakatlarni bajarish maqsadga muvofiq emas. Ushbu muammoni hal qilish uchun so'rovni bajaradigan alohida **trek** yaratamiz, so'ngra so'rov natijasida olingan IP-manzilni ilovaning asosiy oqimiga o'tkazamiz.

Ilova interfeysida boshlash tugmasi, bajarilish ko'rsatkichi va natijani ko'rsatish maydoni mavjud. Foydalanuvchi interfeysini tavsiflovchi **res/layout / main.xml** fayli quyidagicha ko'rinadi:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<LinearLayout android:orientation="horizontal"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="left|center_vertical">
<Button android:layout_width="wrap_content"
```

```
android:layout height="wrap content"
android:text="Determine IP address"
android:onClick="onDetermineIPAddressClick"/>
<ProgressBar android:id="@+id/progressBar"
android:layout width="wrap content"
android:layout height="wrap content"
android:visibility="invisible"/>
</LinearLayout>
<TextView android:id="@+id/ipTextView"
android:layout width="wrap content"
android:layout height="wrap content"
android:layout gravity="left|center vertical"/>
</LinearLayout>
```

Ilova to'g'ri ishlashi uchun Internetga kirishni talab qilganligi sababli, manifest fayliga ruxsatni qo'shish kerak:

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

Activity sinfini ishga tushirish standart bo'lib, **onCreate()** metodida foydalanuvchi interfeysi vidjetlariga havolalarni saqlaydigan aktiv sinfning maydonlari to'ldiriladi.

```
public class MainActivity extends Activity {
private Handler handler;
private TextView ipTextView;
private ProgressBar progressBar;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
ipTextView = (TextView) findViewById(R.id.ipTextView);
progressBar = (ProgressBar) findViewById(R.id.progressBar);
```



```
handler = new Handler() {
@Override
public void handleMessage(Message msg) {
handleIPDeterminationMessage(msg);
}
};
}
// ...
```

Ishlovchi maydoni **Handler**-dan meros bo'lgan anonim sinf ob'ektiga o'rnatiladi, bu xabarni qayta ishlashni **handleIPDeterminationMessage()** metodiga topshiradi.

"**Determine IP address**" tugmasi quyidagicha ko'rinadi:

```
// ...
public void onDetermineIPAddressClick(View v) {
ipTextView.setText("");
progressBar.setVisibility(View.VISIBLE);
Thread determinationThread = new Thread() {
@Override
public void run() {
determineIPAddress();
}
};
determinationThread.start();
}
// ...
```

Ushbu ishlov beruvchi natija chiqish maydonini tozalaydi, harakatlanish satrini ko'rinadigan qiladi va aktiv sinfning **determineIPAddress()** metodini alohida bajarilishiga olib keladi. Belgilangan metod veb-xizmatga so'rov yuboradi.

qaytarilgan qiymatni **JSON** formatida qayta ishlaydi va qabul qilingan **IP**-manzilni asosiy oqimga xabar sifatida yuboradi:

```
// ...
private void determineIPAddress() {
try {
URL url = new URL(
"http://ip2country.sourceforge.net/ip2c.php?format=JSON");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.connect();
BufferedReader reader = new BufferedReader(
new InputStreamReader(conn.getInputStream()));
String ip = (String) new JSONObject(reader.readLine()).get("ip");
reader.close();
handler.sendMessage(handler.obtainMessage(0, 0, 0, ip));
} catch (Exception e) {
handler.sendMessage(handler.obtainMessage(0,0,0,e.getMessage()));
}
}
// ...
```

Handler sinfi **handleIPDeterminationMessage()** metodini chaqirish va xabarni argument sifatida yuborish orqali boshqaradi:

```
// ...
private void handleIPDeterminationMessage(Message msg) {
progressBar.setVisibility(View.INVISIBLE);
ipTextView.setText(msg.obj.toString());
}
} // class MainActivity
```

Ushbu metod jarayon indikatorini yashiradi, xabardan ma'lum IP-manzilni chiqaradi va uni matn maydoniga qo'yadi.

2-§. AsyncTask class.

AsyncTask, asinxron, Handler, Message, Object, Parametr, OnPreExecute, OnProgressUpdate, doInBackground, Runnable, run, trek.

Handler sinfidan foydalanishdan tashqari, Android API asinxron ijroni tashkil qilish uchun oddiyroq yondashuvni taqdim etadi. Ushbu metod **AsyncTask** sinfidan foydalanishga asoslangan bo'lib, u kirish va chiqish parametrlariga ega bo'lgan va alohida ish bajarilishini talab qiladigan ba'zi vazifalarni qamrab oladi. Ushbu klassdan foydalanganda aniq ip yaratishning hojati yo'q, shunchaki AsyncTask-dan o'z sinfingizni meros qilib oling va bir nechta metodlarni bekor qiling.

AsyncTask parametrlashtirilgan sinf ekanligini unutmaslik lozim. Uning parametrlari belgilangan va kiritilgan ma'lumotlarning turlarini, bajarilayotgan vazifaning oraliq va yakuniy natijalarini ifodalaydi. Bekor qilingan sinf metodlariga quyidagilar kiradi:

```
protected void onPreExecute();
protected Result doInBackground(Params... params);
protected void onProgressUpdate(Progress... values);
protected void onPostExecute(Result result);
```

Ro'yxatda keltirilgan asosiy metod **doInBackground()**. U alohida vazifaning haqiqiy kodini o'z ichiga oladi. Yuqoridagi metodlarning qolgan qismi asosiy yo'nalishida chaqiriladi va foydalanuvchi interfeysi elementlariga kirishi mumkin. Tarmoqlar orasidagi sinxronizatsiya **AsyncTask** class tomonidan avtomatik ravishda amalga oshiriladi. **OnPreExecute()** va **onPostExecute()** metodlari mos ravishda asinxron vazifani bajarishdan oldin va keyin bajariladi. **OnProgressUpdate()** bajarilishi koddan **doInBackground()** orqali boshlanadi.

```
protected void publishProgress(Progress... values);
```

Bu xususiyat vazifani bajarish paytida foydalanuvchiga oraliq natijalar yoki holatni ko'rsatish uchun ishlatiladi.

AsyncTask sinfidan foydalanishga misol. Avvalgi bo'limdagi misol bilan **Handler** sinfini **AsyncTask** sinfiga almashtirish orqali ko'rsatamiz. Birinchidan, **MainActivity** sinfidan **Handler** maydonini va uni ishga tushirishni olib tashlanadi:

```
public class MainActivity extends Activity {
private TextView ipTextView;
private ProgressBar progressBar;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
ipTextView = (TextView) findViewById(R.id.ipTextView);
progressBar = (ProgressBar) findViewById(R.id.progressBar);
}
// ...
```

AsyncTask IPDeterminationTask-dan meros bo'lib qolgan sinfni chaqiramiz va jarayon sinfiga joylashtiramiz. Bu **Activity** sinfi maydonlari bo'lgan vidjetlar tarkibini yangilashni soddalashtirish uchun zarur. IP-manzilni aniqlash uchun kod **doInBaground()** metodiga joylashtiriladi:

```
// ...
private class IPDeterminationTask extends AsyncTask<Void, Void, String>
{
@Override
protected String doInBackground(Void... params) {
try {
URL url = new URL(
"http://ip2country.sourceforge.net/ip2c.php?format=JSON");
```



```
HttpURLConnection conn =
(HttpURLConnection) url.openConnection();
conn.connect();
BufferedReader reader = new BufferedReader(
new InputStreamReader(conn.getInputStream()));
String ip = (String) new JSONObject(reader.readLine()).get("ip");
reader.close();
return ip;
} catch (Exception e) {
return e.getMessage();
}
}
// ...
```

Qaytarilgan qiymat **String** turiga kiradi va **IP**-manzilni aniqlash natijasini o'z ichiga oladi. **OnPreExecute()** va **onPostExecute()** metodlari matn maydonining qiymatini yangilash va indikatorini ko'rsatish va yashirish uchun ishlatiladi. **DoInBackground()** metodining qaytish qiymati **onPostExecute()** ga parametr sifatida uzatilganligi sababli chiqish soddalashtiriladi:

```
// ...
@Override
protected void onPreExecute() {
ipTextView.setText("");
progressBar.setVisibility(View.VISIBLE);
}
@Override
protected void onPostExecute(String ip) {
progressBar.setVisibility(View.INVISIBLE);
ipTextView.setText(ip);
}
```

```
} // class IPDeterminationTask
// ...
```

Nihoyat, **"Determine IP address"** tugmasi ishlovchisining vazifasi **IPDeterminationTask** sinfining namunasini yaratishdir:

```
// ...
public void onDetermineIPAddressClick(View v) {
new IPDeterminationTask().execute();
}
} // class MainActivity
```

O'z-o'zini tekshirish uchun savollar va masliqlar

1. Asinxron bajarilish nima? Qanday hollarda u qo'llaniladi? U qanday vazifalarni hal qiladi?

2. Android API tomonidan taqdim etilgan asinxron ijro vositalarini sanab o'ting.

*3. Xabar navbati nima? **Handler** sinfining vazifasi nima va undan qanday qilib to'g'ri foydalanish kerak?*

*4. **AsyncTask** sinfining maqsadi nima? Uni qanday ishlatish kerak?*

*5. **IP** sinxronizatsiyasi nima ekanligini tushuntiring. Qanday hollarda kerak? **Android** platformasi vositalari sinxronizatsiya muammosini hal qilishga qanday yordam beradi?*

6. Ilova konfiguratsiyasi o'zgarganda (masalan, ekran yo'nalishi o'zgarganda) asinxron bajarish to'xtatilmasligi uchun ushbu bobdagi misollarni o'zgartiring.

11-BOB. KONTENT PROVAYDERLARI

Kontent provayderlari jarayon va xizmatlar bilan bir qatorda Android ilovalarining asosiy komponentlaridan biri hisoblanadi. Ularning vazifasi ushbu ma'lumotlarni saqlashning ma'lum bir metodi bilan bog'lanmasdan ko'plab ilovalarga ma'lumotlarni taqdim etishdir.

Kontent provayderi tomonidan taqdim etiladigan operatsiyalar odatiy ma'lumotlar bazasi operatsiyalariga o'xshaydi va yozuvlarni qo'shish, yangilash va o'chirish, shuningdek, ma'lumotlarni olish uchun so'rovlarni bajarishni o'z ichiga oladi. Shuni ta'kidlash kerakki, ushbu ma'lumotlar tartibsiz bo'lishi mumkin, chunki ularning tuzilishi va saqlanish metodi butunlay kontent provayderining boshqaruvida bo`ladi.

Kontent provayderi bilan ishlash uchun mijozlar o'zlari ishlayotgan ma'lumotlarning URI-ni quyidagi shaklda belgilaydilar:

content:: //content_provider_name/data_spec.

Kontent provayderining nomi kontent provayderi manifest faylida ro'yxatdan o'tganda ko'rsatilgan qiymatga mos kelishi kerak. Ma'lumotlar spetsifikatsiyasi har qanday formatda bo'lishi mumkin, lekin REST-ga o'xshash spetsifikatsiya qabul qilinadi.

1-§. Kontent provayderlarini tayinlash.

Content, onCreate, DBHelper, query, insert, update, delete, Query.

Standart kontent provayderiga misol. Misol sifatida, mobil qurilma foydalanuvchisining kontaktlariga kirishni ta'minlaydigan standart kontent provayderini ko'rib chiqamiz.

Bu provayder barcha mavjud kontaktlarni olish uchun *content:// com. android.contacts/contacts* URI dan va kontakt raqami 1-ga kirish uchun *content:*

//com.android.contacts/contacts/1 URI dan foydalanadi. Oxirgi URI dan foydalanish nafaqat aloqa ma'lumotlarini olish, balki ularni o'zgartirish yoki o'chirish ham mumkin.

Vazifalar ro'yxati uchun kontent provayderi. Bobning qolgan qismi avvalgi bobdagi o'zgartirilgan ilova yordamida kontent provayderlari mexanizmini muhokamasiga bag'ishlanadi. Bu erda ma'lumotlar bazasiga to'g'ridan-to'g'ri kirishni kontent provayderi orqali o'zaro aloqada o'zgartiramiz. Ushbu provayder uchun kod quyidagicha ko'rinadi:

```
public class ToDoContentProvider extends ContentProvider {
private DBHelper dbHelper;
@Override
public boolean onCreate() {
dbHelper = new DBHelper(getContext());
return false;
}
// ...
```

onCreate() metodi avvalgi bo'limda belgilangan **DBHelper** sinfining namunasini yaratadi. **SQLite** ma'lumotlar bazasi bo'lgan kontent provayderining ma'lumotlar ombori bilan bog'lanish uchun ishlatiladi.

```
// ...
private static final String AUTHORITY =
"ru.ac.uniyar.todoslist.contentprovider";
private static final String BASE PATH = "todos";
private static final int TODOS = 10;
private static final int TODO ID = 20;
private static final UriMatcher matcher =
new UriMatcher(UriMatcher.NO MATCH);
static {
matcher.addURI(AUTHORITY, BASE PATH, TODOS);
```



```
matcher.addURI(AUTHORITY, BASE PATH + "/#", TODO ID);
}
@Override
public Cursor query(Uri uri, String[] projection, String selection,
String[] selectionArgs, String sortOrder) {
SQLitequeryBuilder queryBuilder = new SQLitequeryBuilder();
queryBuilder.setTables("todos");
int uriType = matcher.match(uri);
switch (uriType) {
case TODOS:
break;
case TODO ID:
queryBuilder.appendWhere(" id = " + uri.getLastPathSegment());
break;
default:
throw new IllegalArgumentException("Unknown URI: " + uri);
}
SQLiteDatabase db = dbHelper.getWritableDatabase();
Cursor cursor = queryBuilder.query(db, projection, selection,
selectionArgs, null, null, sortOrder);
cursor.setNotificationUri(getContext().getContentResolver(), uri);
return cursor;
}
@Override
public String getType(Uri uri) {
return null;
}
@Override
public Uri insert(Uri uri, ContentValues values) {
```

```
int uriType = matcher.match(uri);
SQLiteDatabase sqlDB = dbHelper.getWritableDatabase();
long id = 0;
switch (uriType) {
case TODOS:
id = sqlDB.insert("todos", null, values);
break;
default:
throw new IllegalArgumentException("Unknown URI: " + uri);
}
getContext().getContentResolver().notifyChange(uri, null);
return Uri.parse(BASE_PATH + "/" + id);
}
@Override
public int update(Uri uri, ContentValues values, String selection,
String[] selectionArgs) {
int uriType = matcher.match(uri);
SQLiteDatabase sqlDB = dbHelper.getWritableDatabase();
int rowsUpdated;
switch (uriType) {
case TODOS:
rowsUpdated = sqlDB.update("todos", values, selection, selectionArgs);
break;
case TODO_ID:
String id = uri.getLastPathSegment();
if (TextUtils.isEmpty(selection)) {
rowsUpdated = sqlDB.update("todos", values, " id =" + id, null);
} else {
rowsUpdated = sqlDB.update("todos", values, " id =" + id +
```



```
" and " + selection, selectionArgs);
}
break;
default:
throw new IllegalArgumentException("Unknown URI: " + uri);
}
getContext().getContentResolver().notifyChange(uri, null);
return rowsUpdated;
}
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
throw new UnsupportedOperationException();
}
} // class ToDoContentProvider
```

query(), insert(), update() va **delete()** metodlari **ContentProvider** sinfining bekor qilingan metodlari hisoblanadi. Ushbu metodlarning har biri ma'lumotlarni ko'rsatuvchi URI-ni qabul qiladi. **Query()** va **update()** metodlari foydalanuvchi kontaktlariga o'xshash *content://en.ac.uniyar.todoslist. contentprovider/todos* va *content://en.ac.uniyar.todoslist. contentprovider/todos/1* URI-larni qo'llab-quvvatlaydi.

insert() metodi faqat *content://en.ac.uniyar.todoslist. contentprovider/todos* shaklining URI-ni qo'llab-quvvatlaydi, chunki kiritish vaqtida hech qanday kirish identifikatori tayinlanmagan. Ushbu kontent provayderi uchun **delete()** metodi qo'llanilmaydi (istisno qilinadi), chunki bu misolda qo`llashning hojati yo`q.

Yuqoridagi metodlarni amalga oshirishni hisobga olgan holda shuni ta'kidlash mumkinki, kontent provayderining asosiy vazifasi provayderga topshiriq ma'lumotlarini saqlaydigan ma'lumotlar bazasiga aylantirishdir.

2-§. Kontent provayderini manifest faylida ro'yxatdan o'tkazish.

Content, Kontent, asinxron, CursorLoader, SimpleCursorAdapter, initLoader, SwapCursor, update, delete, Query.

Har bir kontent provayderi manifest faylida ro'yxatdan o'tgan bo'lishi kerak. Bizning misolimizda ro'yxatdan o'tish shunday ko'rinadi:

```
<provider
android:name=".ToDoContentProvider"
android:authorities="ru.ac.uniyar.todoslist.contentprovider" >
</provider>
```

Ilova o'rnatilgandan so'ng, kontent provayderiga tegishli URI'lardan foydalangan holda istalgan ilovadan kirish mumkin.

Kontent provayderi tomonidan taqdim etilgan ma'lumotlarni asinxron yuklash. Kontent provayderlaridan foydalanganda **CursorLoader** klass yordamida ma'lumotlarni asinxron tarzda yuklash uchun juda foydali imkoniyat mavjud. Bu sinf kontent provayderiga alohida so'rov yuboradi va yuklab olish tugallangach qayta qo'ng'iroq qilish metodini chaqiradi.

CursorLoader sinfidan foydalanish uchun barcha kerakli qayta qo'ng'iroq metodlarini belgilaydigan **LoaderManager. LoaderCallbacks** interfeysini qo'llaydigan sinf kerak bo'ladi. An'anaga ko'ra, bu sinf odatda aktivlik sinfi hisoblanadi:

```
public class MainActivity extends Activity
implements LoaderManager.LoaderCallbacks<Cursor> {
private SimpleCursorAdapter adapter;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
ListView todoListView = (ListView) findViewById(R.id.todoList);
todoListView.setOnItemClickListener(
```



```
new ListView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
onToDoListItemClick(id);
}
});
getLoaderManager().initLoader(0, null, this);
String[] from = new String[] { "title", "description" };
int[] to = new int[] { R.id.titleText, R.id.descriptionText };
adapter = new SimpleCursorAdapter(this, R.layout.todo_item, null,
from, to, 0);
todoListView.setAdapter(adapter);
}
// ...
```

E'tibor bersak, avvalgi bo'limdagi **onCreate()** metodidan farqli o'laroq, bu misolda ma'lumotlar bazasi so'rovi mavjud emas va **SimpleCursorAdapter** sinfining ob'ekti ma'lum bir kursor bilan bog'lanmagan holda yaratilgan (konstruktorning uchinchi parametri **null**). Buning sababi, ma'lumotlar yuklanganda kursor **CursorLoader** klass tomonidan dinamik ravishda yaratiladi.

initLoader() metodi **LoaderManager.LoaderCallbacks** interfeysining **onCreateLoader()** metodining bajarilishini boshlaydi:

```
// ...
@Override
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
return new CursorLoader(this,
Uri.parse("content://ru.ac.uniyar.todoslist.contentprovider/todos"),
null, null, null, null);
}
```

```
// ...
```

Ma'lumotlar yuklanganda, qabul qilingan ma'lumotlar bilan bog'langan kursorni o'tkazib, **onLoadFinished()** qayta qo'ng'iroq qilish metodi chaqiriladi:

```
// ...
@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
adapter.swapCursor(data);
}
// ...
```

SwapCursor() metodi qabul qilingan ma'lumotlarni asosiy ro'yxatida ko'rsatilishiga olib keladi. Ma'lumotlar mavjud bo'lmaganda, **onLoaderReset** metodi chaqiriladi.

```
// ...
@Override
public void onLoaderReset(Loader<Cursor> loader) {
adapter.swapCursor(null);
}
// ...
```

3-§. Kontent provayderi orqali ma'lumotlarni kiritish va yangilash.

Content, Kontent, SQLiteDatabase, ContentResolver, onOptionsItemSelected, Query, onCreateOptionsMenu, CursorLoader.

Kontent provayderidan foydalanganda ma'lumotlarni kiritish va yangilash ushbu operatsiyalarni bevosita ma'lumotlar bazasida bajarish bilan bir xil. Yagona farq shundaki, ma'lumotlarni olish va o'zgartirish metodlari **SQLiteDatabase** sinfi ob'ektida emas, balki **URI** tomonidan aniqlangan kontent provayderi orqali **ContentResolver** klassi ob'ektida chaqiriladi:


```
// ...
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if (resultCode != RESULT_OK) return;
ContentValues cv = new ContentValues();
cv.put("title", data.getStringExtra("title"));
cv.put("description", data.getStringExtra("description"));
cv.put("dueDate", data.getStringExtra("dueDate"));
if (data.hasExtra("id")) {
getContentResolver().update(
Uri.parse("content://ru.ac.uniyar.todoslist.contentprovider/todos/" +
data.getIntExtra("id", 0)), cv, null, null);
} else {
getContentResolver().insert(
Uri.parse("content://ru.ac.uniyar.todoslist.contentprovider/todos/"), cv);
}
}
public void onToDoListItemClick(long id) {
Cursor todoCursor = getContentResolver().query(
Uri.parse("content://ru.ac.uniyar.todoslist.contentprovider/todos/" + id),
null, null, null, null);
todoCursor.moveToNext();
Intent intent = new Intent(this, ToDoEditorActivity.class);
intent.putExtra("id", todoCursor.getInt(
todoCursor.getColumnIndex("_id")));
intent.putExtra("title", todoCursor.getString(
todoCursor.getColumnIndex("title")));
intent.putExtra("description", todoCursor.getString(
todoCursor.getColumnIndex("description")));
```

```
intent.putExtra("dueDate", todoCursor.getString(
todoCursor.getColumnIndex("dueDate")));
startActivityForResult(intent, 1);
}
} // class MainActivity
```

CursorLoader sinfining muhim xususiyati, u qabul qilingan ma'lumotlarga kiritilgan o'zgarishlarni avtomatik ravishda kuzatib boradi. Agar bu ma'lumotlar o'zgartirilsa, dasturchi tomonidan qo'shimcha harakatlarsiz yuklash jarayoni yangidan amalga oshiriladi. Shuning uchun, ma'lumotlarni o'zgartirganda, avvalgi bo'limda ishlatilgan **requery()** metodini chaqirishning hojati yo'q.

Oncreateoptionsmenu() va **onOptionsItemSelected()** metodlari menyu bilan ishlash kodi avvalgi bandda keltirilganga nisbatan o'zgarmaydi.

Ilovani yaratib, ishga tushirgandan so'ng, uning avvalgi bobdagi ilova kabi ishlashini tekshirish oson.

O'z-o'zini tekshirish uchun savollar va mashqlar

1. Kontent provayderi nima? Kontent provayderlari Android infratuzilmasida qanday rol o'ynaydi?

2. Kontent provayderdan foydalanishda URI nima uchun kerak? U qanday qismlardan iborat? URI yaratish uchun qanday qoidalar qo'llaniladi?

*3. Kontent provayderni amalga oshirishda **ContentProvider** sinfining qaysi metodlarini bekor qilish kerak?*

*4. **CursorLoader** sinfining maqsadi nima? To'g'ridan-to'g'ri ilovaning asosiy o'zagida kontent provayderiga so'rovlar qilishda **CursorLoader** sinfidan foydalanishning qanday afzalliklari bor?*

*5. **LoaderManager.LoaderCallbacks** interfeysining qayta qo'ng'iroq qilish metodlari nima uchun? Ulardan foydalanishga misol keltiring.*

6. Ushbu bobda tasvirlangan dastur loyihasini yarating. Ilovani kompilyatsiya qiling va uni emulyatorda yoki haqiqiy qurilmada ishga tushiring.

7. Misoldagi ilovada yozuvlarni o'chirish imkoniyatini qo'shing.

***Ko'rsatma.** Kontent provayderining **delete()** metodini qo'llash orqali muammoni hal qiling.*

12-BOB. ANDROID MARKET SAYTIDA ILOVANI YUKLASH

Android Market - bu Google tomonidan boshqariladigan Android ilovalari uchun maxsus tarqatish mexanizmi. Agar siz ilovangizni Android Market-da nashr qilsangiz, butun dunyo bo'ylab millionlab foydalanuvchilar uni yuklab olishlari, o'rnatishlari va foydalanishlari mumkin. Bundan tashqari, har qanday foydalanuvchi uning reytingi uchun asos bo'lgan ilovangizga baho berishi va ilovaga bag'ishlangan sahifada o'z mulohazalarini qoldirishi mumkin. Bu sizning ilovangiz mashhurligining potentsial tendentsiyalarini aniqlashga va uning zaif tomonlarini aniqlashga yordam beradi.

Android Market sayti ilova haqida foydali statistikani taqdim etadi, uning muvaffaqiyatini kuzatish uchun foydalanish mumkin.

Ushbu bob **Android Market** saytida ilovani nashr qilishni o'z ichiga oladi. Qolaversa, nashr etilgan ilovani tavsiflovchi bir qator sayt suratlarini taqdim etiladi. Ammo dasturni nashr qilish uchun uni maxsus formatda tarqatiladigan faylga to'plash lozim bo`ladi.

1-§. Qayta taqsimlanadigan fayl yaratish.

Android, Manifest, uses, sdk, minSdkVersion, Eclipse, Ant, Maven, APK.

Aytaylik, Android platformasi uchun juda foydali dastur yoki juda qiziqarli o'yin yaratish haqida ajoyib fikr bor. Ushbu ilovani yaratdingiz va uni oxirgi foydalanuvchilarning qo'liga topshirishga tayyor. Buni qanday qilish kerak? Avvalo, dasturni oxirgi foydalanuvchi qurilmasiga joylashtiriladigan tarzda paketlash kerak. Buning uchun **APK** faylini yaratish kerak (**Android Package File** - Android ommaviy ish fayli). Keyingi bo'limda birinchi APK faylingizni yaratish bosqichlarini ko`rib o`tamiz.

Manifest fayli. Qayta taqsimlanadigan APK faylini yaratishdan oldin, dastur iloji boricha ko'proq foydalanuvchilar uchun mavjud ekanligiga ishonch hosil qilish kerak. Buning uchun **AndroidManifest.xml** faylining **uses-sdk** elementini ko'rib chiqiladi. Hozirda **Silent Mode Toggle** ilovasida **uses-sdk** elementi namunaviy ilova yaratilganda o`rnatilgan **minSdkVersion** atributini o`z ichiga oladi.

<uses-sdk android:minSdkVersion="4" />

minSdkVersion xususiyati ushbu ilovani o'rnatish mumkin bo'lgan Android platformasining minimal versiyasini belgilaydi. Bu holda, mimmal 4-versiyaga o'rnatiladi. Biroq, **Eclipse** muhitida **Silent Mode Toggle** ilovasini yaratishda maqsadli 8-versiyaga o'rnatildi. Bu nimani anglatadi? Minimal va maqsadli versiyalar o'rtasidagi farq nima?

Android platformasining versiyalari deyarli har doim (kamdan-kam hollardan tashqari) teskari mos keladi. Misol uchun, deyarli barcha 3-versiya vositalari 4-versiyada. Bu erda "deyarli" so'zi nimani anglatadi. Android platformasi ham boshqa tizimlar kabi mukammal emas. Minglab turli vositalarga ega bo'lgan bunday murakkab tizimda hamma narsani kuzatib borish va bitta xatoga yo'l qo'ymaslik mumkin emas. Ba'zida yangi versiyani ishlab chiquvchilar buning oldini olish uchun katta sa'y-harakatlar qilsalar ham oldingi versiyalarga ta'sir qilishi mumkin bo'lgan kichik o'zgarishlar kiritiladi. Bundan tashqari, ba'zida **Android**-ga oldingi versiyalarda bo'lmagan yangi komponentlar qo'shiladi. Agar dastur bunday komponentdan foydalansa, u holda orqaga qarab muvofiqlik tabiiy ravishda buziladi. Shunday qilib, agar ilovada kamida 4-versiyasi o'rnatilgan bo'lsa, bu Android operatsion tizimining 4 va undan yuqori versiyalari o'rnatilgan qurilmada ishlashini anglatadi.

minSdkVersion atributining qiymatiga asoslanib, **Android Market** ma'lum bir qurilma foydalanuvchisiga qaysi ilovalar ko'rsatilishi kerakligini aniqlaydi, chunki foydalanuvchi o'z qurilmasida o'rnatilgan Android versiya raqamini belgilaydi. Agar ilova **minSdkVersion=4** ga o`rnatilgan bo`lsa va foydalanuvchi

o‘z qurilmasining 3-versiyasi (ya’ni, Android 1.5) yoki undan pastroqda ishlayotganligini aniqlasa, foydalanuvchi ilovani ko‘rmaydi. Android Market sayti 3 va undan past versiyadagi barcha ilovalarni filtrlaydi. Agar foydalanuvchi 4 yoki undan yuqori versiyani ko'rsatsa, u ilovani ko'radi va o'z qurilmasiga o'rnatishi mumkin.

Agar ilovangiz manifestidagi **uses-sdk** elementida **minSdkVersion** atributini o‘tkazib yuborsangiz, Android Market avtomatik ravishda **0** qiymatini almashtiradi. Bu ilova Androidning barcha versiyalari bilan mos kelishini bildiradi. Agar ilova platformaning eski versiyalari uchun mavjud bo‘lmagan funksiyalardan foydalansa (masalan, faqat Android 2.0 dan boshlab mavjud bo‘lgan Bluetooth funksiyalari), foydalanuvchi qurilmasi dasturning ishlashni davom ettira olmasligini bildiruvchi xatolik hosil qiladi. Foydalanuvchi keraksiz dasturni yuklab olish va o'rnatish uchun vaqt va pul sarflaganidan juda norozi bo'ladi. To'g'ri, iqtisodiy jihatdan g'alaba qozonasiz, lekin bunday harakatlar firibgarlik hisoblanadi va buni aniqlash oson.

Eng yaxshi instrumentlar to'plamini tanlash. Android APK faylini quyidagi metodlardan biri bilan yaratish mumkin:

- **Eclipse** dasturiga o'rnatilgan ADT plaginidan foydalanish;
- **Hudson Continuous Integration Server** kabi uzluksiz integratsiya serveriga o'rnatilgan avtomatlashtirilgan qurish jarayonida;
- **Ant** vositasi yordamida buyruq satrida;
- **Maven** qurish tizimidan foydalanish.

Ushbu kitobda **APK** faylini yaratish uchun **Eclipse**-ning **ADT** plaginidan foydalaniladi. U Android ilovasini bitta **APK** fayliga kompilyatsiya qiladigan, imzolaydigan (raqamli imzo) va paketlaydigan bir qator vositalarni taqdim etadi.

APK faylini yaratishning boshqa variantlari murakkabroq va faqat malakali dasturchilar tomonidan **ADT** plaginlari kerakli imkoniyatlarni ta'minlay olmasa ishlatiladi. **Ant** yaratish jarayoni va **APK** faylini yaratish uchun boshqa vositalar haqida ko'proq ma'lumot olish uchun Android hujjatlariga qarash mumkin:

http://d.android.com/guide/publishing/app-signing.html

2-§. Ilovaga raqamli imzo.

Android, kalit, sertifikat, imzo, Keytool, Jarsigner, Sertifikatlash, ADT, SDK, Android, APK.

Android operatsion tizimi har bir o'rnatilgan ilova kalit juftligini (ommaviy va xususiy) aniqlovchi sertifikat asosida raqamli imzolanishini talab qiladi. Shaxsiy kalit faqat ishlab chiquvchiga tegishli bo'ladi. Elektron raqamli imzoni yaratishda foydalaniladigan sertifikat ilovani identifikatsiyalash va ilovalar o'rtasida ishonchli munosabatlarni o'rnatish uchun ishlatiladi.

Android ilovasini to'g'ri imzolash uchun quyidagilarni bilish kerak:

- *barcha android ilovalari imzolangan bo'lishi kerak. operatsion tizim imzosiz dasturni o'rnatmaydi.*
- *elektron raqamli imzoni yaratish uchun o'zingizning sertifikatingizdan foydalanishingiz mumkin. buning uchun rasmiy sertifikatlash organlari kerak emas.*
- *ilova tayyor bo'lganda, albatta shaxsiy kalit bilan imzolanishi kerak.*
- *ilovani ishlab chiqish jarayonida APK faylini imzolagan disk raskadirovka kaliti bilan ilovani imzolab bo'lmaydi.*

Sertifikatning tugash sanasi bor, u faqat dastur o'rnatilgan paytda tekshiriladi. Agar tugash sanasi ilova qurilmaga o'rnatilgandan keyin bo'lsa, ilova normal ishlashda davom etadi.

Agar biron sababga ko'ra sertifikat yaratish uchun ADT vositalaridan foydalanishni xohlamasangiz, APK faylini yaratish va imzolash uchun **Keytool** yoki **Jarsigner** kabi istalgan standart vositadan foydalanish mumkin.

Sertifikatlash tartibi Android hujjatlarida batafsil keltiriladi. Unda turli vositalar va metodlardan foydalangan holda sertifikatlarni qanday yaratish keltiriladi. APK fayllarni imzolash haqida ko'proq ma'lumotni quyidagi manzilda topish mumkin:

http://d.android.com/guide/publishing/app-signing.html

Kalit bazasinini yaratish. Android-dagi kalit bazasi (**Java**-da bo'lgani kabi) shaxsiy sertifikatlarni saqlaydigan konteynerdir. Quyidagi vositalar yordamida kalit bazasi faylini yaratish mumkin:

ADT eksport ustasi. Ushbu vosita ADT sozlamalariga o'rnatilgan va imzolangan APK fayllarini eksport qilish va bosqichma-bosqich jarayonda sertifikatlar va kalit bazalarni yaratish imkonini beradi.

Keytool ilovasi. Buyruqlar satridan foydalanib kalit bazani yaratishga imkon beradi. Ushbu vositani Android SDK asboblar jildida topish mumkin. Bu buyruq satri ko'plab foydali sertifikatlash variantlarini taqdim etadi. Kalit bazani yaratish va APK faylini yaratish uchun ADT eksport ustasidan foydalanadi.

Kalit bazasi xavfsizligi. Kalit bazasi faylida shaxsiy sertifikatingiz mavjud bo'lib, u Android Marketda ilovangizni aniqlash uchun Android platformasi tomonidan foydalaniladi. Kalit bazasini xavfsiz joyda zaxiralash lozim. Agar uni yo'qotib qo'ysangiz, endi ilovalaringizga xuddi shu shaxsiy kalit bilan imzo cheka olmaysiz. Ilovani yangilay olmaysiz, chunki Android Market ilovangiz boshqa kalit bilan imzolanganligini ko'radi va uni yangilashga ruxsat bermaydi. Bunday holda, sayt yangilanish faylini boshqa Android ilovasi sifatida "ko'radi". Ilova paketi nomini o'zgartirsangiz ham xuddi shunday bo'ladi. Bunday holda, Android Market ham ilovani yangilashga ruxsat bermaydi.

3-§. APK faylini yaratish.

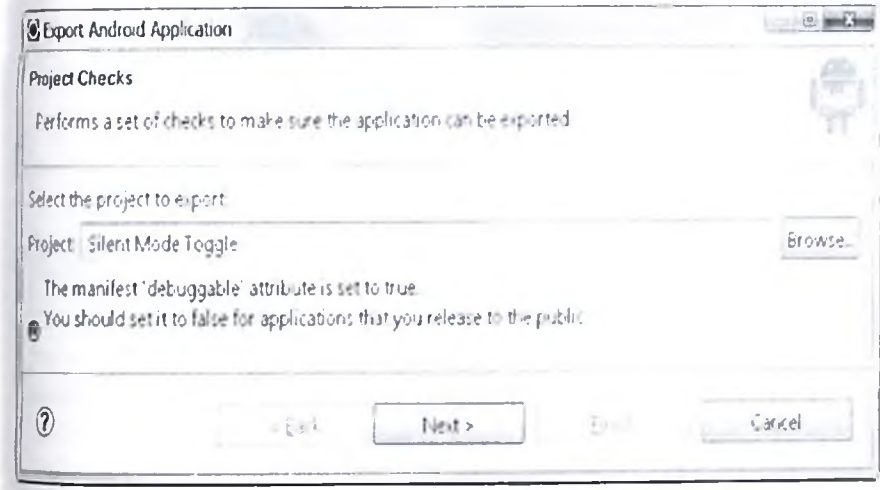
Android, Eclipse, Silent, Mode, Toggle, Keystore, Alias, Password, Confirm, Validity, apk, kalit, sertifikat, imzo.

ADT plagimidan foydalamb APK faylini yaratish uchun quyidagi amallarni bajarish lozim.

1. Eclipse dasturini oching.

2. Silent Mode Toggle ilovasini o'ng tugmasini bosing va kontekst menyusidan ***Android Tools - Export Signed Application Package***-ni tanlang.

Ilovani eksport qilish dialog oynasi joriy loyiha nomi bilan almashtirilgan holda faollashtiriladi (12.3.1-rasm).



12.3.1-rasm. Ilovalarni eksport qilish ustasining birinchi oynasi

3. Keyingi tugmasini bosing.

12.3.2-rasmda ko'rsatilgan **Keystore Selection** dialog oynasi faollashtirilgan.

http://d.android.com/guide/publishing/app-signing.html

Kalit bazasinini yaratish. Android-dagi kalit bazasi (**Java**-da bo'lgani kabi) shaxsiy sertifikatlarni saqlaydigan konteynerdir. Quyidagi vositalar yordamida kalit bazasi faylini yaratish mumkin:

ADT eksport ustasi. Ushbu vosita ADT sozlamalariga o'rnatilgan va imzolangan APK fayllarini eksport qilish va bosqichma-bosqich jarayonda sertifikatlar va kalit bazalarni yaratish imkonini beradi.

Keytool ilovasi. Buyruqlar satridan foydalanib kalit bazani yaratishga imkon beradi. Usbbu vositani Android SDK asboblar jildida topish mumkin. Bu buyruq satri ko'plab foydali sertifikatlash variantlarini taqdim etadi. Kalit bazani yaratish va APK faylini yaratish uchun ADT eksport ustasidan foydalanadi.

Kalit bazasi xavfsizligi. Kalit bazasi faylida shaxsiy sertifikatingiz mavjud bo'lib, u Android Marketda ilovangizni aniqlash uchun Android platformasi tomonidan foydalaniladi. Kalit bazasini xavfsiz joyda zaxiralash lozim. Agar uni yo'qotib qo'ysangiz, endi ilovalaringizga xuddi shu shaxsiy kalit bilan imzo cheka olmaysiz. Ilovani yangilay olmaysiz, chunki Android Market ilovangiz boshqa kalit bilan imzolanganligini ko'radi va uni yangilashga ruxsat bermaydi. Bunday holda, sayt yangilanish faylini boshqa Android ilovasi sifatida "ko'radi". Ilova paketi nomini o'zgartirsangiz ham xuddi shunday bo'ladi. Bunday holda, Android Market ham ilovani yangilashga ruxsat bermaydi.

3§. APK faylini yaratish.

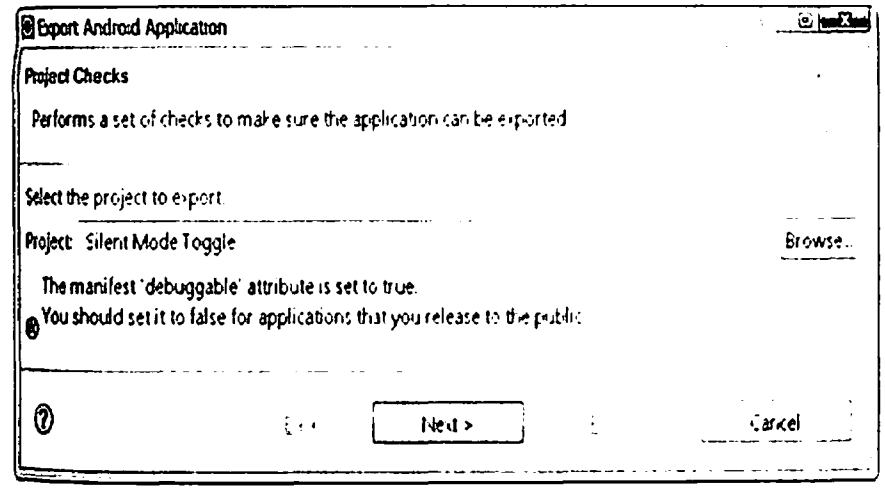
Android, Eclipse, Silent, Mode, Toggle, Keystore, Alias, Password, Confirm, Validity, apk, kalit, sertifikat, imzo.

ADT plaginidan foydalanib APK faylini yaratish uchun quyidagi amallarni bajarish lozim.

1. Eclipse dasturini oching.

2. Silent Mode Toggle ilovasini o'ng tugmasini bosing va kontekst menyusidan ***Android Tools - Export Signed Application Package***-ni tanlang.

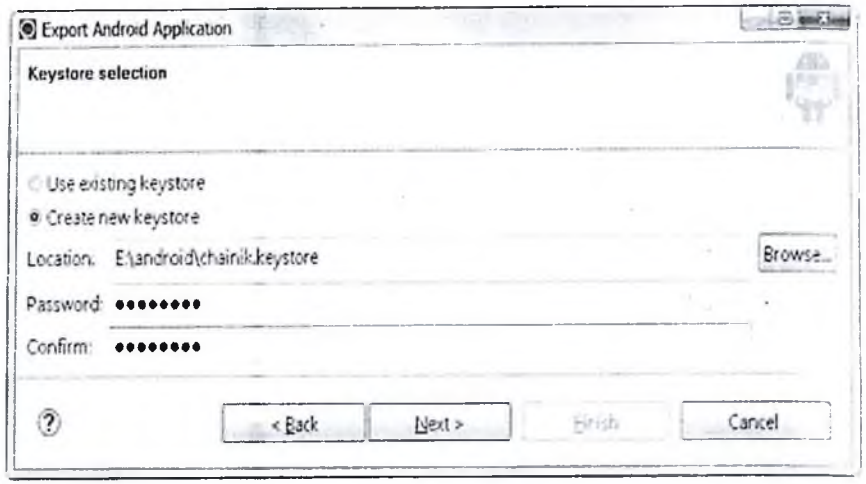
Ilovani eksport qilish dialog oynasi joriy loyiha nomi bilan almashtirilgan holda faoilashtiriladi (12.3.1-rasm).



12.3.1-rasm. Ilovalarni eksport qilish ustasining birinchi oynasi

3. Keyingi tugmasini bosing.

12.3.2-rasmda ko'rsatilgan **Keystore Selection** dialog oynasi faollashtirilgan.



12.3.2-rasm. Kalit bazasini yaratiladi.

4. Hali kalit do'konini yaratilmagan, shuning uchun "Yangi kalit do'konini yaratish" tugmasini tanlanadi.

5. Matn maydoniga saqlash manzilini kiriting yoki tanlang.

Xotirani **E:\android** manzilidagi Android jildiga joylashtirish tavsiya etiladi.

Fayl nomi **.keystore** kengaytmasiga ega bo'lishi kerak. Faylning to'liq manzili quyidagicha ko'rinadi:

E:\android\chainik.keystore

6. Eslab qolish uchun parolni kiriting. Tasdiqlash maydoniga parolni qayta kiriting.

7. Keyingi tugmasini bosing. Kalit yaratish muloqot oynasi faollashtirildi.

8. Quyidagi maydonlarni to'ldiring.

• **Alias** (taxallus). Kalitni vizual aniqlash uchun ishlatiladi.

• **Password** (Parol) va **Confirm** (Tasdiqlash). Kalitdan foydalanganda ushbu parolni kiritasiz.

• **Validity**. Ushbu maydonda ko'rsatilgan qiymat kalitdan qancha vaqt foydalanish mumkinligini aniqlaydi. Ushbu misolda etarlicha katta vaqt oralig'ini o'rnatish lozim.

9. Muloqot oynasining quyidagi maydonlarini to'ldiring (barchasi shart emas, kamida bitta maydonni to'ldirish lozim):

• **First and Last Name** (Ism va familiya);

• **Organization Unit** (Tashkiliy birlik);

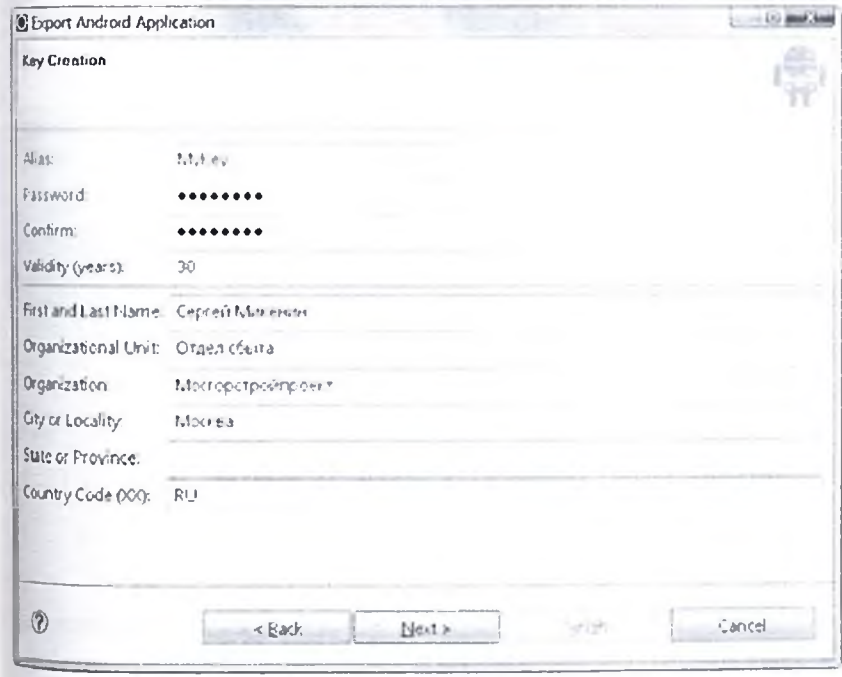
• **Organization** (tashkilot);

• **City or Locality** (shahar yoki tuman);

• **State or Province** (shtat yoki viloyat);

• **Country Code** (Mamlakat kodi).

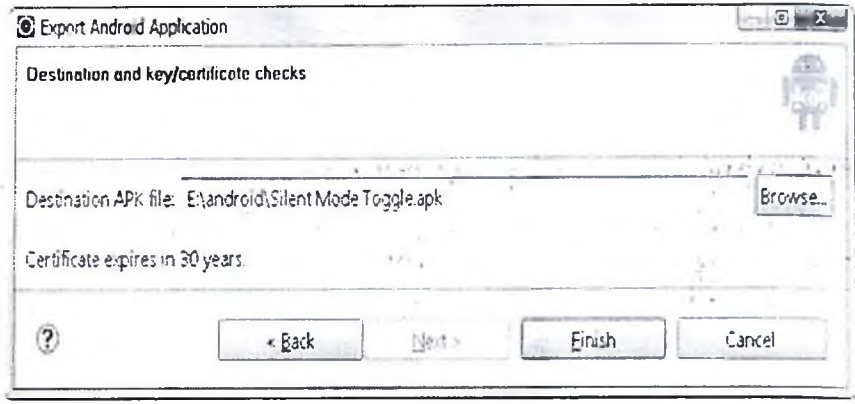
Muloqot oynasi 12.3.3-rasmda ko'rsatilgandek bo'lishi kerak.



12.3.3-rasm. Kalit yaratish

10. Keyingi tugmasini bosing.

Eksport ustasining oxirgi oynasi faollashtirilgan (*12.3.4-rasm*).

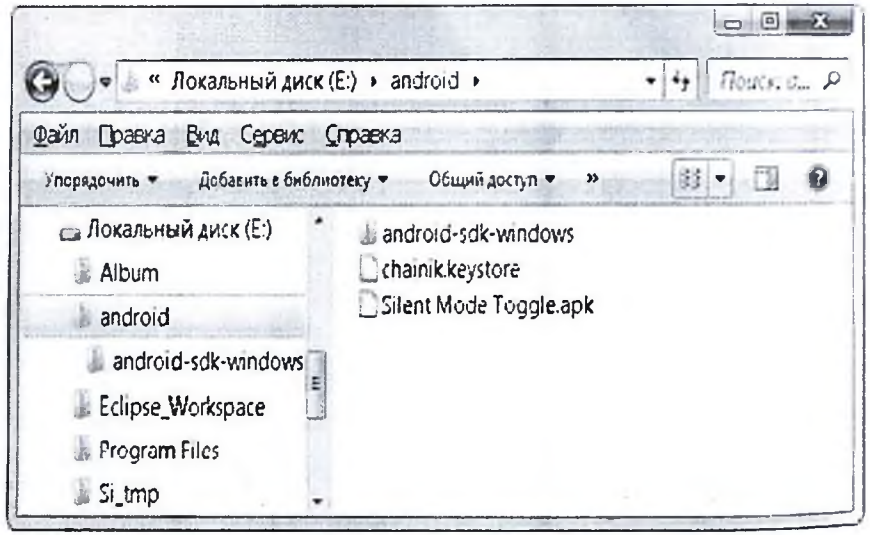


12.3.4-rasm. APK faylining nomi va yo'lini o'rnatish

11. **.apk** fayli uchun nom va yo'lni kiritiladi.

12. Finish tugmasini bosiladi.

.apk va **.keystore** fayllari belgilangan papkada yaratiladi (*12.3.5-rasm*).



12.3.5-rasm. Kompyuterning fayl tizimidagi .apk va .keystore fayllari.

4-§. Android Market hisobini yaratish.

Android, Google, Developer, Name, Email Address, Website URL, Phone Number, Continue, apk.

Endi imzolangan **APK** fayli bor, ilovangizni **Android Market** saytida nashr qilishingiz mumkin. Buning uchun Android Market hisob qaydnomangizni yaratishingiz kerak. Lekin uni yaratish uchun sizda **Google** hisobi bo'lishi kerak. Har qanday Google hisobi buni amalga oshiradi, masalan, **Gmail**. Agar sizda Google hisobi bo'lmasa, uni ***http://www.google.com/accounts*** sahifasida bepul olish mumkin. Keyin Android Market hisobini yaratish uchun quyidagi amallarni bajariladi. To'lov kartasi yordamida sizdan **25 AQSh dollari** miqdoridagi ro'yxatdan o'tish to'lovini to'lashingiz kerak. Agar ro'yxatdan o'tish to'lovini to'lamasangiz, ilovani nashr eta olmaysiz.

1. Veb-brauzerni ishga tushiring va ***http://market.android.com/publish*** saytiga o'ting.

2. Google hisobingiz bilan ro'yxatdan o'ting (*12.4.1-rasm*).

12.4.1-rasm. Ro'yxatdan o'tish sahifasi.

3. Keyingi sahifada quyidagi maydonlarni to'ldiring.

• **Developer Name** (Dasturchi nomi). Ismingiz va familiyangiz (taxallusingizdan foydalanishingiz mumkin), ular saytda e'lon qilingan ilovani ishlab chiquvchisining ismi va familiyasi sifatida ko'rsatiladi. Ushbu maydonga kompaniya nomini kiritishingiz mumkin. Tuzuvchi nomi maydonining mazmuni hisob yaratilgandan keyin o'zgartirilishi mumkin bo'ladi.

• **Email Address** (E-pochta manzili). Foydalanuvchilar sizga elektron pochta xabarlarini yuboradigan manzil. Foydalanuvchilar odatda savol berishadi yoki ilova sifati haqida fikr bildiradilar.

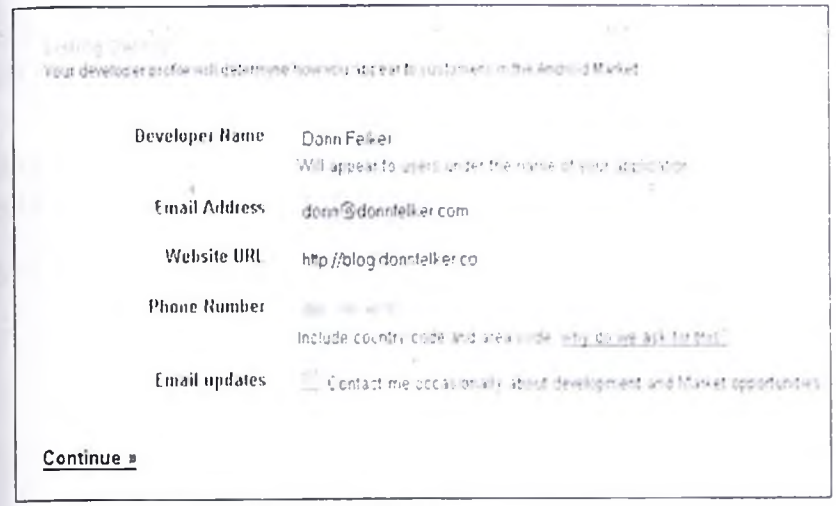
• **Website URL** (Veb-sayt URL manzili). Veb-saytingiz manzili. Agar veb-saytingiz bo'lmasa, bepul blog taqdim etadigan bepul **Blogger** hisobiga ega bo'lishingiz mumkin. Android Market ushbu blogni veb-saytingiz sifatida ko'radi. Bepul Blogger akkauntini www.blogger.com saytidan olish mumkin.

• **Phone Number** (Telefon raqami). Chop etilgan kontent bilan bog'liq muammolar yuzaga kelganda bog'lanishingiz mumkin bo'lgan raqam.

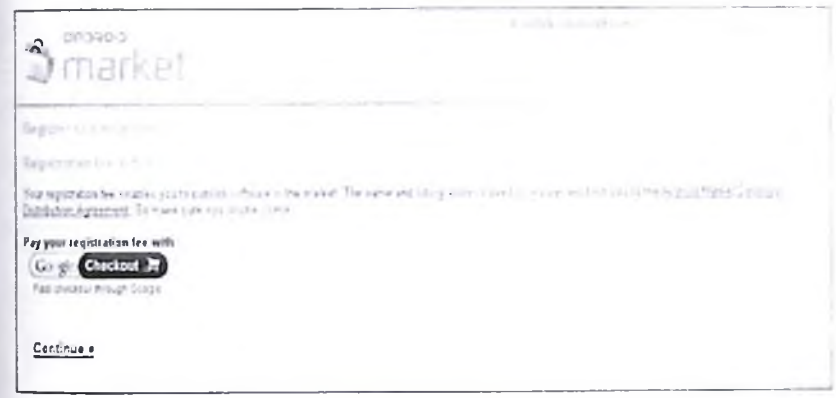
Shaklni to'ldirgandan so'ng, *12.4.2-rasm*dagi kabi ko'rinish bo'lishi kerak.

4. **Continue** tugmasini bosing. Keyingi sahifada $25 ro'yxatdan o'tish to'lovini to'lash taklifi ko'rsatiladi (8.8-rasm).

5. **Google Checkout** hisobingizdan foydalanib, ro'yxatdan o'tish uchun to'lovni amalga oshirish uchun **Continue** tugmasini bosing.



12.4.2-rasm. Ilova ishlab chiquvchisi haqida ma'lumot

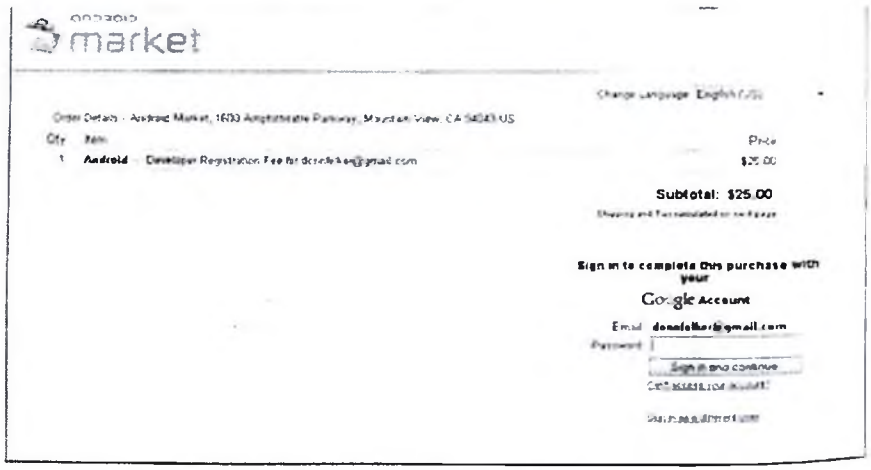


12.4.3-rasm. Ro'yxatdan o'tish to'lovi

6. Quyida keltirilgan *12.4.4-rasmda* ko'rsatilgan sahifada kredit karta ma'lumotlarini va boshqa hisob-kitob ma'lumotlarini kiritiladi. **Agree and Contingue** tugmasini bosiladi. Agar kredit karta allaqachon Google'da ro'yxatdan o'tgan bo'lsa, bu sahifani ko'rmaysiz. Bunday holda, paydo bo'lgan sahifada kartani tanlang va **Continue** tugmasini bosing.

7. Keyingi sahifada (*12.4.5-rasm*) parolni kiriting va **Sign in and continue** tugmasini bosing.

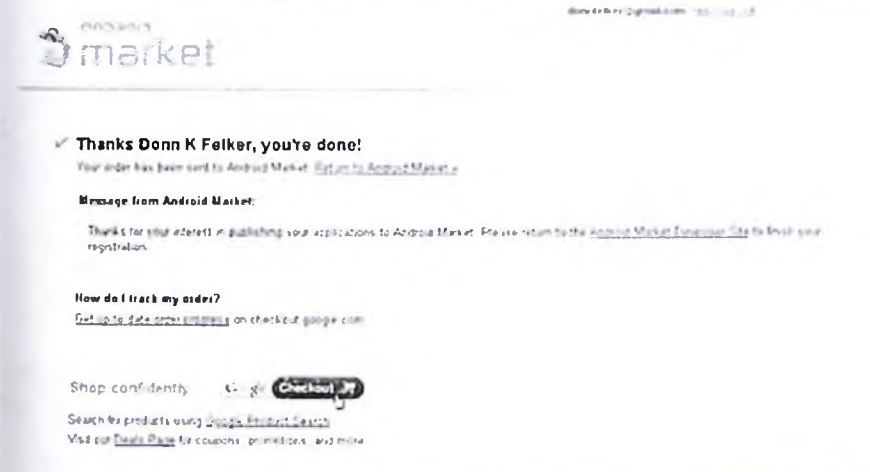
12.4.4-rasm. Kredit karta ma'lumotlari



12.4.5-rasm. Dasturchi sifatida ro'yxatdan o'tish uchun obunani tasdiqlash.

8. Ro'yxatdan o'tish buyurtmasini tasdiqlash sahifasida (*12.4.6-rasm*) buyurtmani berish tugmasini bosiladi. Internetga ulanish tezligiga va sayt qanchalik band ekanligiga qarab, yuklash sahifasini ko'rmasligingiz mumkin. Jarayon tugagach, Android ilovalarini ishlab chiquvchisi sifatida ro'yxatdan o'tganligingizni tasdiqlovchi xabarni ko'rasiz (*12.4.7-rasm*).

12.4.11-rasm. Sizni ishlab chiquvchi sifatida ro'yxatdan o'tkazish talabini tasdiqlang.



12.4.7-rasm. Dasturchi sifatida ro'yxatdan o'tganligingiz haqida xabarnoma.

O'z-o'zini tekshirish uchun savollar va mashqlar

1. *Android Market nima?*
2. *Qayta taqsimlanadigan fayl yaratishni tushuntiring.*
3. *APK faylini qanday yaratiladi?*
4. *Manifest fayli nima?*
5. *Android platformasining minimal versiyasini qanday belgilanadi?*
6. *Android APK faylini qaysi metodlardan biri bilan yaratish mumkin?*
7. *Android ilovasini to'g'ri imzolash uchun nimalarni bilish kerak?*
8. *ADT plaginidan foydalanib APK faylini yaratish uchun ketma-ketligini keltiring*
9. *Android Market hisobini yaratish qanday amalga oshiriladi?*

FOYDALANILGAN ADABIYOTLAR RO'YXATI

1. Разработка мобильных приложений в среде Android Studio / Л. В. Пирская, 2020, -125 стр.
2. Android для программистов. Создаем приложения / П. Дейтел, Х. Дейтел, Э. Дейтел, М. Морганоб. — СПб. : Питер, 2013,
3. Программирование под Android / З. Мединкс, Л. Дорнин, Б. Мик, М. Накамура. — СПб. : Питер, 2012. — 496 с.
4. Коматинени, С. Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов / С. Коматинени, Д. Маклин. — М. : Вильямс, 2012. — 880 с.
5. Android для программистов. Создаем приложения / П. Дейтел, Х. Дейтел, Э. Дейтел, М. Моргано. — СПб. : Питер, 2012. — 560 с.
6. Левин, А. Android на планшетах и смартфонах / А. Левин. — СПб.: Питер, 2013. — 224 с.
7. Парамонов, Н. В. Язык программирования Java и Java-технологии / Н. В. Парамонов. — Ярославль : ЯрГУ, 2006. — 92 с.
8. Develop | Android Developers. — 2013. — URL: http://developer.android.com/develop/index.html (online; accessed: 01.02.2013).
9. Vogel, L. Android Development. Tutorials about development for Android — 2013. — URL: http://www.vogella.com/android.html (online; accessed: 01.02.2013).
10. Nudelman, G. Android Design Patterns: Interaction Design Solutions for Developers / G. Nudelman. — Indianapolis : Wiley, 2013. — 458 p.
11. Friesen, J. Android Recipes: A Problem-solution Approach / J. Friesen, D. Smith. — N.-Y. : Apress, 2011.
12. Darwin, I. Android Cookbook / I. Darwin. — Sebastopol : O'Reilly Media Incorporated, 2012. — 688 p.
13. Haseman, C. Creating Android Applications: Develop and Design / C. Haseman. — Berkeley : Peachpit Press, 2011. — 273 p.
14. Ostrander, J. Android Ui Fundamentals: Develop & Design / J. Ostrander. — Berkeley : Peachpit Press, 2012. — 337 p.

FOYDALANISH UCHUN ELEKTRON ISHORATLAR

1. http://progbook.ru/android/1417-deytel-android-dlya-programmistov-sozdaem-prilozheniya.html
2. http://progbook.ru/android/1227-hashimi-razrabotka-prilozheniy-dlya-android.html
3. http://englishonlineclub.com/pdf/Beginning%20Android%20Programming%20with%20Android%20Studio.html
4. https://www.rulit.me/books/android-dlya-vseh-prakticheskoe-posobie-novichka-download-252878.html
5. https://avidreaders.ru/book/planshety-i-smartfony-na-android-prostoy.html
6. https://programmera.ru/programmirovanie_android/
7. http://www.fandroid.info/android-uchebnik-menyu-s-dinamicheskim-dobavleniem-punktov/
8. https://skillbox.ru/media/code/35_knig_po_mobilnoy_razrabotke_na_android_i_ios/
9. https://play.google.com/store/apps/details?id=com.google.android.apps.books&hl=ru&gl=US
10. https://zavistnik.com/top-5-luchshih-knig-dlya-android-razrabotchikov-na-russkom-yazyke-v-2020-god/

M U N D A R I J A

SO'Z BOSHI ... 3

1-BOB. ANDROID PLATFORMASI HAQIDA QISQACHA MA'LUMOT ... 5

1-§. Android uchun ilovalarni ishlab chiqish haqida ... 5

2-§. Android va dastur infratuzilmasi ... 7

3-§. Foydalanuvchi interfeysini yaratish ... 12

O'z-o'zini tekshirish uchun savollar va mashqlar ... 16

2-BOB. XML FAYLIDAN FOYDALANISH ... 17

1-§. Eclipse muharriri ... 18

2-§. Vakillik ... 20

O'z-o'zini tekshirish uchun savollar va mashqlar ... 22

3-BOB. VIZUAL RIVOJLANISH MUHITI ... 23

1-§. Dizayner oynasini ochish ... 23

2-§. Foydalanuvchi interfeysini ishlab chiqish ... 26

3-§. Ko'rinishlarni konteynerga joylashtirish ... 27

4-§. Android ilovalarini ishlab chiqish asoslari ... 29

5-§. Loyiha yaratish ... 30

O'z-o'zini tekshirish uchun savollar va mashqlar ... 35

4-BOB. AKTIVLIK VA REJA (NIYAT)LAR ... 36

1-§. Android ilovasi komponentlari. ... 36

2-§. Manifest faylida jarayon deklaratsiyasi. ... 37

3-§. Vazifalar va aktivlik to'plami. ... 42

O'z-o'zini tekshirish uchun savollar va mashqlar ... 45

5-BOB. MODEL VIEW CONTROLLER (MVC) ARXITEKTURASIDA ODDIY LOYIHALAR ... 46

1-§. Loyiha yaratish ... 47

2-§. XML faylidan foydalanuvchi interfeysini yuklash va uning komponentlariga kirish. ... 51

3-§. Hisoblagich modeli. ... 54

4-§. Faol va passiv modellarning afzalliklari va kamchiliklari. 59
5-§. Ekran yo'nalishini o'zgartirish bilan ishlash. 60
O'z-o'zini tekshirish uchun savollar va mashqlar 63
6-BOB. VIEW SINFI VA UNING IMKONIYATLARI. 64
1-§. Ekranga teginish hodisalari. 64
2-§. Vidjet ierarxiyasi bo'yicha jarayonni boshqarish qoidalari. 67
O'z-o'zini tekshirish uchun savollar va mashqlar 73
7-BOB. RESURSLAR BILAN ISHLASH 74
1-§. Resurslar tasnifi. 74
2-§. Konfiguratsiyaga bog'liq manbalar. 77
3-§. Menyular va harakatlar panelini shakllantirish uchun resurslardan foydalanish. 78
O'z-o'zini tekshirish uchun savollar va mashqlar 81
8-BOB. MA'LUMOTLARNI SAQLASH. 82
1-§. Sozlamalar mexanizmi. 82
2-§. Ma'lumotlar bazasini boshqarish. 86
3-§. Ma'lumotlarga kirish. 87
4-§. Kursorlar bilan ishlash. 89
O'z-o'zini tekshirish uchun savollar va mashqlar 90
9-BOB. MA'LUMOTLARNI SAQLASH UCHUN MA'LUMOTLAR BAZASIDAN FOYDALANADIGAN DASTUR YARATISH. 91
1-§. Ma'lumotlar bazasining hayot aylanishini boshqarish klassi. 91
2-§. Ilova menyusi va yozuvni qo'shishni qayta ishlash. 95
3-§. Jarayonning o'zaro ta'siri interfeysi 98
4-§. Aktiv muharrir bilan ishlash. 103
O'z-o'zini tekshirish uchun savollar va mashqlar 104
10-BOB. ASINXRON BAJARILISH 105
1-§. Ishlovchi sinfi va xabarlar navbati. 105
2-§. AsyncTask class. 111
O'z-o'zini tekshirish uchun savollar va mashqlar 114
11-BOB. KONTENT PROVAYDERLARI 115
1-§. Kontent provayderlarini tayinlash. 115
2-§. Kontent provayderini manifest faylida ro'yxatdan o'tkazish. 120
3-§. Kontent provayderi orqali ma'lumotlarni kiritish va yangilash 122
O'z-o'zini tekshirish uchun savollar va mashqlar 125
12-BOB. ANDROID MARKET SAYTIDA ILOVANI YUKLASH 126
1-§. Qayta taqsimlanadigan fayl yaratish. 126
2-§. Ilovaga raqamli imzo. 129
3-§. APK faylini yaratish. 131
4-§. Android Market hisobini yaratish. 135
O'z-o'zini tekshirish uchun savollar va mashqlar 140
FOYDALANILGAN ADABIYOTLAR RO'YXATI 141
FOYDALANISH UCHUN ELEKTRON ISHORATLAR 142

O‘QUV ADABIYOTINING NASHR RUXSATNOMASI

O‘zbekiston Respublikasi Oliy ta’lim, fan va innovatsiyalar vazirligining 2023 yil “27” Mart dagi “68”-sonli buyrug‘iga asosan

O.O.JAKBAROV
mualliflning familiyasi, ismi-sharifi

Kompyuter ilmlari va dasturlash texnologiyalari (yo‘nalishlar bo‘yicha)
ta’lim yo‘nalishi (mutaxassisligi)

ning talabalari (o‘quvchilari) uchun tavsiya etilgan

«Android platformasida mobil ilovalarni ishlab chiqish
o‘quv adabiyotining nomi va turi (darslik, o‘quv qo‘llanma)

ga O‘zbekiston Respublikasi Vazirlar Mahkamasi tomonidan litsenziya berilgan nashriyotlarda nashr etishga ruxsat berildi.

Vazir I.Abduraxmonov

№ 68 - 265

Jakbarov Odiljon Otamirzaevich,
Goyipov Umidjon Gulomjonovich

ANDROID PLATFORMASIDA MOBIL ILOVALARNI ISHLAB CHIQISH

Muharrir: Mansur Inomov
Sahifalovch Azam Rahimov
Musahhih: Husanboy *Kamalov*
Tex muharrir: *Ibrohimjon Rustamov*

Terishga berildi 20.07.2023.
Bosmaga ruxsat etildi 12.12.2023
Bichimi 60/84, 1/16, Hajmi 9,25 bosma taboq.
Adadi 50 nusxa 40 raqamli buyurtma.

"Чуст ийи" нашриёти. Наманган вилояти.
Чуст тумани. Хисорак МФЙ. Навбахор кўчаси. 57-уй.
Нашриёт лицензия раками:
№2389-9960-d 1e5-178b-af05-553-2947.
Лицензия 2021 йил 22 июнда берилган
Нашриёт телефон раками: +99891 347 19 43

«Vodiy Poligraf» МЧЖ босмахонасида чоп этилди.
Наманган ш., 5-кичик туман, Ғалаба кўчаси, 19-уй.
Мурожаат учун: +99890 213 1976

Jakbarov Odiljon Otamirzaevich

Namangan muhandislik - qurilish institutining Informatika va AT kafedrasi mudiri, texnika fanlari nomzodi, dotsent. Matematik modellashtirish, Axborot texnologiyalari, Dasturlash hamda Web dasturlash bo`yicha mutaxassis. 1 ta o`quv qo`llanma, 60 dan ortiq ilmiy – uslubiy Ishlar muallifi, dasturiy mahsulotlar yaratish bo`yicha 6 dan ortiq guvohnomalari mavjud. 10 dan ortiq Kasb ta`limi (Informatika va AT) mutaxassislik bo`yicha magistrlik dissertasiyalariga rahbarlik qilgan.

Goyipov Umidjon Gulomjonovich

Namangan muhandislik - qurilish institutining Informatika va AT kafedrasi o'qituvchisi. Axborotlarga ishlov berishni algoritmlash hamda Web dasturlash bo`yicha mutaxassis. 30 dan ortiq ilmiy teziz va maqolalar, 10 dan ortiq uslubiy ishlar muallifi. Dasturiy mahsulotlar yaratish bo`yicha 4 dan ortiq guvohnomalari mavjud.

ISBN 978-9910-9844-8-8