МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ
ТЕХНІЧНИЙ УНІВЕРСИТЕТ

О.О. Степаненко, Є.М. Федорченко

ОСНОВИ ПРОГРАМУВАННЯ ДОДАТКІВ НА БАЗІ РІЗНИХ МОБІЛЬНИХ ОПЕРАЦІЙНИХ СИСТЕМ ТА ПЛАТФОРМ

Навчальний посібник

*Рекомендовано як навчальний посібник для студентів спеціальностей
121 «Інженерія програмного забезпечення» та
122 «Компьютерні науки»*

Запоріжжя, 2019

УДК 004. 451:621.391.2 (075.8)
С 79

Рекомендовано до друку Вченою радою
Запорізького національного технічного університету
(протокол №3 від 30.10.2018 року)

Колектив авторів:
Степаненко О.О. – Вступ, розділ 1–4
Федорченко Є.М. – Розділ 5–7, додатки А–Д

Рецензенти:
С.І Гоменюк – доктор технічних наук, професор, декан математичного факультету Запорізького національного університету
В.З. Грищак – доктор технічних наук, професор, завідувач кафедри прикладної математики та механіки Запорізького національного університету

Степаненко О.О.

С 79 Основи програмування додатків на базі різних мобільних операційних систем та платформ : навч. посіб. / О. О. Степаненко, Є. М. Федорченко. – Запоріжжя : ЗНТУ, 2019. – 124 с.

ISBN 978-617-529-207-5

Останні декілька років у світі спостерігається все більше розповсюдження пристроїв на базі мобільних платформ. З'являються нові мобільні платформи. За своєю потужністю мобільні пристрої вже досягли рівня настільних комп'ютерів.

Розробка мобільних прикладних програм ставить перед програмістами нові задачі. До таких задач, зокрема, відносяться більш уважний підхід керуванням пам'яттю мобільної платформи, облік обмеженої зарядки батарей, а також облік можливих нестабільності бездротового мережевого з'єднання.

Знання, що отримані з даного посібника, можуть бути цікавими як з точки зору використання в професійній діяльності, так і з точки зору більш глибокої розуміння роботи мобільного програмно-апаратного комплексу та використання набутого досвіду з науковими методами та підходами.

УДК 004. 451:621.391.2 (075.8)

ISBN 978-617-529-207-5

ЗМІСТ

ВСТУП

Без стільникового зв'язку сучасне людство не може уявити собі й дня. І буквально зовсім недавно головними функціями були тільки дзвінки та відправка повідомлень. Однак сьогодні пріоритети дещо змінюються. Телефон став далеко не просто засобом для комунікації людей, але і багатофункціональним пристроєм, що дозволяє пізнавати світ, проводити дозвілля і навіть заробляти гроші. І, звичайно, все це стало можливо тільки завдяки розвитку Інтернету для мобільних пристроїв і створених з цієї нагоди застосунків. У XXI столітті почався бурхливий розвиток ринку мобільного контенту, зокрема, мобільних застосунків. Здешевити мобільний інтернет-трафік дозволила поява нових технологій – GPRS і EDGE. Користувачі починають викачувати із Всесвітньої мережі величезну кількість картинок, ігор, рингтонів та іншого.

Ринок мобільних пристроїв стали заповнювати смартфони і комунікатори. Вони відрізнялися від звичайних мобільних телефонів наявністю досить розвиненої операційної системи (Windows Phone, Symbian OS, RIM, Android, Mac OS), а також більш широкими можливостями і більшою продуктивністю.

З'являються спеціалізовані сайти з продажу контенту для мобільних пристроїв.

Зараз, крім різних маркет – застосунків, подібні веб-ресурси створюють провідні виробники стільникових телефонів, смартфонів і комунікаторів.

В даний час спеціалізовані портали з розповсюдження застосунків мають відомі по всьому світу корпорації: гігант Apple (App Store), компанія Google (Google Play), компанія Nokia (OVi), виробник смартфонів Blackberry – компанія RIM (BlackBerry App World і Application Center), компанія Sony Ericsson (PlayNow arena) і ряд інших.

Крім самих застосунків, ці онлайн-ресурси продають також різноманітний мобільний контент (музику, відео, картинки, електронні книги і т.д.).

Принцип роботи мобільного телефону

У теоретичній частині ми не будемо углиблюватися в історію створення стільникового зв'язку, про її засновників, хронологію стандартів і т.д. Кому це цікаво – матеріалу предосить як у друкованих виданнях, так і в мережі інтернет.

Розглянемо, що ж із себе представляє мобільний (стільниковий) телефон.

Стільниковий телефон – це прийомо – передавач, що працює на одній із частот у діапазоні 850МГц, 900МГц, 1800МГц, 1900Мгц. Причому приймання й передача рознесені по частотах.

Система GSM складається з 3-х основних компонентів, таких як:

– підсистема базових станцій (BSS – Base Station Subsystem);

– підсистема перемикання/комутації (NSS – Networkswitchings ubsystem);

– центр керування й обслуговування (OMC – Operation and Maintenance Center);

Двома словами працює це наступним чином.

Стільниковий (мобільний) телефон взаємодіє з мережею базових станцій (БС). Вежі БС звичайно встановлюють на дахах будинків або інших споруд, або ж на орендованих вже існуючих вишках усіляких ретрансляторів радіо/ТВ і т.п., а також на висотних трубах котелень і інших промислових споруд.

Телефон після включення весь наступний час моніторить (прослуховує, сканує) ефір на наявність GSM-сигналу своєї базової станції. Сигнал своєї мережі телефон визначає по спеціальному ідентифікатору. Якщо такий є (телефон перебуває в зоні покриття мережі), то телефон вибирає кращу за рівнем сигналу частоту й на цій частоті посилає БС запит на реєстрацію в мережі.

Процес реєстрації по суті є процесом аутентифікації (авторизації). Його суть полягає в тому, що кожна SIM-карта, вставлена в телефон, має свої унікальні ідентифікатори IMSI (International Mobile Subscriber Identity) і Ki (Key for Identification). Ці самі IMSI і Ki заносяться в базу центру аутентифікації (Auc) при випуску виготовлених SIM-карт оператора зв'язку. При реєстрації телефону в мережі ідентифікатори передаються БС, а саме Auc. Далі Auc (центр ідентифікації) передає телефону деяке випадкове число, яке є ключем для виконання обчислень по спеціальному алгоритму. Це обчислення відбувається одночасно в мобільному телефоні й в Auc, після чого обидва результати порівняються. Якщо вони збігаються, то SIM-карта визнається справжньою й телефон реєструється в мережі.

Для телефону ж ідентифікатором у мережі є його унікальний номер IMEI (International Mobile Equipment Identity). Цей номер звичайно складається з 15 цифр у десятковому поданні. Наприклад 35366300/758647/0. Перші вісім цифр описують модель телефону і його походження. Що залишилися – серійний номер телефону й контрольне число.

Даний номер зберігається в енергонезалежній пам'яті телефону. У застарілих моделях цей номер можна перемінити за допомогою спеціального програмного забезпечення (ПЗ) і відповідного

програматора (іноді й дата-кабелю), а в сучасних телефонах він дублюється. Один екземпляр номера зберігається в області пам'яті, яку можна програмувати, а дублікат – у зоні пам'яті OTP (One Time Programming), яка програмується виробником один раз і не має можливості перепрограмування.

Отож, якщо навіть змінити номер у першій області пам'яті, то телефон, при включенні, порівнює дані обох областей пам'яті, і, якщо виявляються різні номери IMEI – телефон блокується. Для чого все це міняти, запитаєте ви? Насправді законодавство більшості країн забороняє це робити. Телефон по номеру IMEI відслідковується в мережі. Відповідно при крадіжці телефону його можна відстежити й вилучити. А якщо встигнути змінити цей номер на будь-який інший, тоді шанси знайти телефон зводяться до нуля. Цими питаннями займаються спецслужби при відповідній допомозі оператора мережі і т.д. Нас цікавить чисто технічний момент зміни номера IMEI.

Справа в тому, що при певних обставинах даний номер може ушкодитися в результаті збою ПО або неправильного його відновлення й тоді телефон абсолютно не придатний для експлуатації. Ось тут на допомогу й приходять усі засоби, щоб відновити IMEI і працездатність апарата. Докладніше цей момент буде розглянутий у розділі програмного ремонту телефону.

Тепер коротенько про передачу голосу від абонента до абонента в стандарті GSM. Насправді це технічно дуже складний процес, який абсолютно відрізняється від звичної передачі голосу по аналогових мережах як, наприклад, домашній провідний/радіо телефон. Чимсь віддалено схожі цифрові радіотелефони DECT, але реалізація однаково інша.

Справа в тому, що голос абонента, перш ніж буде переданий в ефір, зазнає безлічі перетворень. Аналоговий сигнал розбивається на відрізки тривалістю 20мс, після чого перетворюється в цифровий, після чого кодується шляхом застосування алгоритмів шифрування з т.зв. відкритим ключем – система EFR (Enhanced Full Rate – удосконалена система кодування мови, розроблена фінською компанією Nokia).

Усі сигнали кодека обробляються алгоритмом на основі принципу DTX (Discontinuous Transmission) – переривчастої передачі мови. Його корисність полягає в тому, що він управляє передавачем телефону, включаючи його тільки в той момент, коли починається вимова мови й відключає в паузах між розмовою. Усе це досягається за допомогою включеного в кодек VAD (Voice Activated Detector) – детектору активності мови.

У абонента, що приймає всі, перетворення відбуваються у зворотному порядку [1].

Устрій мобільного телефону і його основні функціональні вузли (модулі)

Будь-який мобільний телефон – це складний технічний пристрій, що складається з безлічі функціонально закінчених модулів, які взаємозалежні між собою й у цілому забезпечують нормальну роботу апарату. Вихід з ладу хоча б одного модуля спричиняє мінімум – часткову несправність апарата, максимум – телефон стає повністю непрацездатним.

Призначення й робота окремих вузлів

1. Акумуляторна батарея (АКБ) – основне (первинне) джерело живлення телефону. У процесі експлуатації має одну неприємну властивість – старіння, тобто втрату ємності, збільшення внутрішнього опору. Це необоротний процес і швидкість старіння акумулятора залежить від багатьох факторів, ключовими з яких є правильна експлуатація й зберігання.

Раніше основна маса АКБ для телефонів проводилася за технологіями Nicd (на основі нікелю й кадмію), Nimh (нікель-металгідрид). У цей час дані акумулятори зняті з виробництва. З поширенням АКБ на основі технології Li-Ion (літій-іонний), останні показали краще співвідношення ціни та якості, а також мали ряд переваг, зокрема відсутність т.зв. «ефекту пам'яті». Тривалість терміну служби становить приблизно 3–4 роки. Не дуже давно на ринку з'явилися Li-Pol (літій-полімерні) акумулятори. Вони коштують дешевше літій-іонних, але термін служби в них теж менше – приблизно 2 роки.

Сучасні АКБ є працездатними, якщо в них збереглося не менш 80% від номінальної ємності. На практиці ж зустрічаються АКБ із 50% і менше. Тобто багато користувачів намагаються «вичавити» з акумулятора останні міліампери, через що самі потім і страждають, тому що нерідко зношений акумулятор починає здуватися, що може приводити до поломок корпуса телефону, а іноді навіть до виходу з ладу мережного зарядного пристрою, ланцюгів зарядки телефону, контролера живлення. Так що на АКБ грошей заощаджувати не варто. Телефону теж потрібно гарне живлення.

Особливого догляду АКБ не вимагають. Головне, не допускати переохолодження в зимовий час (до –10°С), тому що прискорюється розряд і старіння, а так само нагрівання до 50–60°С й вище. Це небезпечно – АКБ може попросту здутися й навіть вибухнути (саме для літієвих АКБ це критично)!

АКБ мобільного телефону складається з 2-х частин: власне батареї й маленької плати електроніки-автоматики.

Що стосується плати електроніки, то вона виконує захисну функцію, захищаючи як саму батарею, так і телефон від позаштатних ситуацій, таких як:

– коротке замикання (КЗ) живильних клем акумулятора;

– перегрів батареї в процесі зарядки й експлуатації;

– витрата розряду батареї нижче встановленої мінімально припустимої норми;

– перезаряд батареї.

При виникненні однієї з таких ситуацій, спрацьовує т. зв. електронне реле й вихідні клеми АКБ знеструмлюються.

Як правило, сучасна АКБ має мінімум 3 контактних виводи для підключення до батарейного рознімання мобільного телефону. Це відповідно «+», «–», і «TEMP» (датчик температури, за допомогою якого контролер батареї разом з контролером живлення телефону управляють процесом зарядки батареї, зменшуючи або збільшуючи зарядний струм, а при перегрівах або КЗ взагалі відключають батарею від клем плати електроніки).

Слід зазначити, що в різних виробників розташування контактів може відрізнятися!

Основними характеристиками АКБ є:

– номінальна напруга – як правило 3,6–3,7 Вольт. Для повністю зарядженого акумулятора 4,2–4,3 Вольт;

– ємність – для сучасних телефонів приблизно від 700мА до 2000мА й більш;

– внутрішній опір – чим менше, тем краще (приблизно до 200 міліОм).

2. Контролер живлення – служить для перетворення напруги АКБ у кілька видів напруг для живлення окремих вузлів і пристроїв телефону, таких, як CPU (центральний процесор), RAM і ROM (мікросхеми пам'яті), усіляких підсилювачів, іноді підсвічувань клавіатури й дисплея і т.д., і так само управляє процесом зарядки АКБ. Разом із процесором активує вбудовані в нього або ж зовнішні підсилювачі звуку розмовного динаміка, мікрофона, буззера (поліфонічного гучномовця). Плюс до всього забезпечує обмін даними з SIM-картою.

Конструктивно виконаний у вигляді окремого чипа. Іноді може бути сполучений із процесором (китайські підробки відомих брендів типу Nokia N95 і т.д.).

При нормальній експлуатації телефону контролер живлення рідко виходить із ладу. Найчастіше це трапляється під час зарядки при перегріві або при використанні неоригінального або несправного

зарядного пристрою (ЗУ). Рідше – якщо телефон піддався впливу вологи, отримав сильний удар.

Зовнішній вигляд представлений на рис. 1.1 і може відрізнятися (залежить від конкретної моделі телефону та його виробника).

Рисунок 1 – Мобільний телефон

3. Sim-holder (SIM-коннектор) – тримач SIM-картки. Виходячи з назви – служить для підключення SIM-картки до телефону. Конструкція практично однакова для всіх телефонів, тому що сучасні SIM-картки приведені до одного стандарту. Має в собі 6 (рідко 8) підпружиненних контактів, за допомогою яких здійснюється електричний зв'язок SIM-карти й контролера живлення або процесора. Відрізняються лише конструкцією кріплення (утримання) SIM-карти. До поломок можна віднести облом контактів при частій зміні SIM-карт або ж неправильне їх вилучення, коли користувач починає застосовувати підручні засоби для підковирювання SIM-карти для подальшого захвата пальцями й вилучення із тримача. Часто до цього прибігають наші прекрасні жінки, використовуючи свої довгі, з дорогим манікюром нігті. У підсумку – страждає й телефон і манікюр.

Спеціального догляду коннектор не вимагає. Але бувають випадки (знову таки залежить від користувача), коли контакти окислюються, засмічуються, втрачають свої пружні властивості. У такому випадку допускається **дуже обережно** протерти їх гумкою (ластиком) і **дуже обережно**, злегка, голкою або дерев'яною зубочисткою підігнути контакти нагору.

При описаних вище несправностях SIM-холдера (тримача), телефон не буде «бачити» вашу SIM-карту й постійно буде виводити на дисплей повідомлення типу: «Вставте SIM-карту». Зламані тримачі ремонту не підлягають і вимагають заміни на нові.

4. Мікрофон – служить для перетворення голосу користувача в слабкі електричні сигнали з метою їх подальшого посилення, перетворення й відправлення в ефір. У стільникових телефонах бувають

двох типів: аналогові й цифрові. Останні мають більш складну конструкцію й вимагають більше працезатрат при демонтажі й заміні.

Мікрофони втрачають свої експлуатаційні характеристики або виходять із ладу в основному при забрудненні, попаданні води, при ударах по телефону (особливо це стосується цифрових мікрофонів, тому що вони самі по собі дуже тендітні).

При несправностях мікрофона в телефоні можуть бути такі дефекти:

– другий абонент не чує користувача взагалі;

– другий абонент чує користувача дуже слабко;

– у слуховому (розмовному) динаміці чутний тріск (т.зв. наведення GSM-сигналу). Такий же шум можна почути, коли підносимо стільниковий телефон у режимі розмови або відправлення SMS до працюючого радіоприймача, підсилювача, комп'ютерних колонок і т.д. Мікрофони не ремонтуються й підлягають заміні (крім випадків засмічення отворів, звуководів корпуса мобільного телефону. Їх слід просто очистити від пилу, бруду і т.д.).

5. Динамік (розмовний динамік) – служить для перетворення електричних сигналів у звукові коливання. Тобто працює у зворотному порядку мікрофона. Один абонент говорить у мікрофон, який перетворює голос в електричні сигнали, далі ці сигнали перетворюються (див. опис вище), випромінюються в ефір. Другий абонент уловлює ці сигнали телефоном і чує їх у динаміку телефону.

У більшості телефонів установлено кілька динаміків – окремо розмовний і окремо поліфонічний. Поліфонічний динамік відтворює мелодію при вхідному виклику, смс і т.д. Але є телефони (у більшості – від фірми Samsung), де роль розмовного й поліфонічного виконує той самий динамік. Тільки при відтворенні мелодії або інших сигналів активується додатковий підсилювач потужності звуку. До несправностей динаміків можна віднести часткову несправність і повну. Часткова – це відтворення мови або музики дуже тихо, із хрипами й неприємним дзенькотом. Це можна усунути, але лише в тих випадках, коли, після зовнішнього огляду буде видно, що динамік засмічений сторонніми предметами. Наприклад такими, як дуже дрібна металева стружка, яка любить проникати через спеціально відведені отвори для виходу звуку динаміка. Це обумовлене тим, що динамік у своїй конструкції містить постійний магніт. От він і примагнічує до себе дрібні металеві предмети. Краще замінити такий динамік на новий. По-перше, це заощадить вам час, який ви будете витрачати на чищення, а його вам знадобитися чимало. По-друге, рідко буває, що після чищення динамік працює так само чисто, без викривлень і так само голосно.

Повна – відсутність звуку взагалі. Причина – обрив проводів звукової котушки динаміка. Вирішується тільки заміною динаміка.

6. Спікер (буззер, дзвінок, поліфонічний динамік – це все те саме) – той же динамік, тільки в більшості випадків призначений для відтворення мелодії дзвінка, смс, MP3 і т.д. Але, як говорилося вище, може використовуватися й для розмови. Несправності й способи усунення такі ж, як і для розмовного динаміка.

7. Центральний процесор (CPU) – є основним пристроєм мобільного телефону. Це той же процесор, який є присутнім у будь-якому персональному комп'ютері, ноутбуці і т.д., тільки трошки поменше й примітивніше. Призначений для виконання машинних команд, інструкцій і операцій, передбачених програмним забезпеченням (прошиванням – розм.) телефону, а також чіткої взаємодії з іншими модулями й пристроями й наступного керування ними. Одним словом, процесор – це «мозок», який повністю управляє роботою мобільного телефону. Конструктивно виконаний у вигляді окремого чипа. Відповідає за безліч процесів, що відбуваються під час нормальної роботи телефону. Основні з них це: вивід зображення на дисплей, приймання й обробка сигналів стільникової мережі, приймання й обробка сигналів клавіатурного модуля, керування роботою камери, пристроїв приймання/передачі інформації, процесом зарядки акумулятора (разом з контролером живлення) і багато іншого.

За умови нормальної експлуатації телефону процесор практично ніколи не виходить із ладу і ніякого догляду не вимагає.

У сучасних телефонах, а особливо смартфонах (у перекладі з англ. смартфон – розумний телефон; той ж телефон, тільки має подібність із комп'ютером у виді наявності операційної системи й безліччю встановлюваних програм для виконання тих або інших завдань) часто встановлюється 2 процесори. Один з них виконує ті ж функції, що й у звичайному телефоні, а другий призначений для роботи операційної системи й виконання її програм.

При виході з ладу центрального процесора телефон стає повністю непрацездатним.

8. Flash-пам'ять. Окремий чип (мікросхема), який призначений для зберігання програмного забезпечення телефону (прошивання, firmware), а також даних користувача (контакти, мелодії, фотографії і т.д.). Програмне забезпечення (прошивання, firmware) – це розроблена виробником телефону програма, яка обробляється й виконується процесором. Для користувача – це те, що він бачить на екрані мобільного телефону й ті функції, які йому доступні в конкретній моделі телефону.

Флэш-пам’ять так само рідко виходить із ладу за умови нормальної експлуатації. Але слід пам’ятати, що ці чипи мають хоч і велику, але все-таки обмежену кількість циклів читання/запису інформації.

Флэш-пам’ять є енергонезалежною й зберігає всі записані в неї дані навіть після відключення джерела живлення (наприклад, АКБ).

9. RAM-пам’ять (ОЗП). Служить для тимчасового зберігання даних. У ній проводяться всі процесорні обчислення програмного коду, а також зберігаються результати обчислень і обробки інформації в конкретний теперішній момент (наприклад, прослуховування музики, відтворення відео, робота застосунків, ігор і т.д.). Через непотрібність пам’ять очищається від одних даних і завантажує нові і так постійно.

Слід пам’ятати, що пам’ять ОЗП (оперативного запам’ятовувального пристрою) є енергозалежною і у випадку відключення джерела живлення всі дані, які зберігалися в ОЗУ будуть загублені!

10. Клавіатурний модуль – стандартна цифрова клавіатура для набору номера абонента, тексту SMS-повідомлень та набір додаткових кнопок, які виконують певні функції програмного забезпечення телефону, наприклад регулювання рівня гучності, запуск програм, фотокамери, диктофона і т.д. Для нормальної роботи клавіатурного модуля основне завдання користувача – містити клавіатуру в чистоті й не допускати влучення вологи, бруду й інших предметів. А якщо ні, то кнопки доводиться давити з більшим зусиллям або ж телефон взагалі не реагує на натискання. Відновити роботу клавіатурного модуля можна методом чищення від забруднень. Якщо ж контактні майданчики й з’єднуючі їхні провідники були піддані впливу вологи або ін. рідин і були ушкоджені, то такий клав. модуль підлягає заміні на новий.

11. LCD-дисплей – власне дисплей (екран) телефону. Призначення всім зрозуміле, тому поглиблюватися на цьому не станемо. Основними характеристиками є такі параметри, як:

– роздільна здатність, тобто кількість відтворених пікселів (крапок). Чим вище цей параметр, тим чіткіше і якісніше буде зображення. Для більш-менш сучасних телефонів властиві такі роздільні здатності екрана: 220X176 пікселів, 320X240. Для телефонів з більшими сенсорними екранами: 400X240, 640X360, 800X400.

– кількість відтворених (відображуваних) кольорів. Теж саме, чим більше, тим краще. У застарілих телефонах з кольоровими дисплеями це значення в основному 4096 кольорів. У міру вдосконалювання цей параметр збільшився до 65тис., потім досяг 262тис. Зараз усі сучасні дорогі телефони обладнані дисплеями із глибиною кольору 16млн.

При правильній експлуатації телефону дисплей не вимагає ніякого обслуговування. У деяких випадках, коли телефон використовується в запиленому середовищі або ж просто згодом у корпус набилося багато пилу й сміття, то дисплей необхідно **бережно** протерти мікрофіброю (спеціальна протиральна серветка, яка добре очищає й не залишає слідів і розводів. Її можна придбати в салонах продажу оптики. Деякі види окулярів комплектуються такою протиральною мікрофіброю.). При експлуатації телефону не можна допускати фізичного впливу на дисплей (удари, здавлювання, сильні перегини), а також піддавати впливу прямих сонячних променів і підвищеній температурі. Це приведе до виходу його з ладу.

12. Приємопередатчик – служить для приймання й передачі стільникового GSM-сигналу. Містить у собі багато функціональних елементів (генератори керування напругою приймача й передавача, смугові фільтри, що містять конденсатори, індуктивності і т.д.). Управляється процесором і кварцовим резонатором 26 Мгц.

При несправностях приємо-передавач телефон не зможе зареєструватися в стільниковій мережі й на дисплеї буде відсутній індикатор рівня GSM-сигналу.

13. Підсилювач потужності – призначений для посилення сигналу, вироблюваного приємопередавачем, до рівня потужності, який є необхідним для випромінювання антеною в ефір.

При несправностях підсилювача потужності телефон буде уловлювати сигнал стільникової мережі, але зареєструватися в ній не зможе, тому що не зможе передавати GSM-сигнал.

14. Антенний перемикач (світч) – призначений для сполучення (підключення) прийомного й передавального тракту GSM-модуля до антени телефону. Цим досягається наявність у телефоні однієї загальної антени для приймання й передачі, а також виключається вплив підсилювача потужності на прийомний тракт.

Конструктивно антенні перемикачі виконані на тендітній керамічній підложці й, при падіннях телефону, дуже часто виходять із ладу, тому що підложка тріскається. У таких випадках телефон «не бачить» сигнал стільникової мережі.

15. Антена – призначена для нагромадження енергії, випромінюваною базовою станцією й наступної передачі її в ланцюзі прийомного тракту. При передачі сигналу все навпаки: з передавача сигнал підсилюється підсилювачем потужності й подається в антену, яка випромінює сигнал в ефір [2].

1 Компютерні програми та їх мобільні аналоги

1.1 Основні характеристики

Найбільш істотні відмінності між застосунками для мобільних пристроїв і застосунками для персональних комп'ютерів обумовлені, ймовірно, тим, яким саме чином люди їх використовують. Найчисленніша категорія користувачів використовуєте комп'ютер в основному для наступних видів роботи: перегляд ресурсів в Web, робота з документами (обробка текстових документів, електронних таблиць, малюнків і тому подібного) та спілкування (електронна пошта, миттєвий обмін повідомленнями і так далі). У типових випадках всі ці види діяльності носять довготривалий, дослідницький характер, і користувач періодично переходить від одного з цих основних занять до іншого.

У більшості випадків способи використання настільного і мобільного варіантів одного і того ж застосунка відрізняються між собою, хоча і доповнюють один одного. Як правило, робота користувача з мобільним застосунком носить характер короткочасних сеансів взаємодії, яке або переривається ззовні, або саме викликає переривання сеансу. Зазвичай користувач або реагує на спробу перервати його роботу, або використовує пристрій для негайної передачі запиту певній особі або процесу. Чудовою ілюстрацією цього може служити типова практика використання сучасних мобільних телефонів. Коли ви здійснюєте телефонний дзвінок або відправляєте текстове SMS-повідомлення, ваш абонент отримує сигнал переривання; якщо ж дзвонять або надсилають SMS-повідомленням вам, то переривається ваша робота. Ця ж модель використання переноситься і на будь-яке інший застосунок, що виконується на мобільних пристроях. Мобільний застосунок може вважатися вдало розробленим лише в тому випадку, якщо користуватися ним настільки ж просто і природно, як зв'язуватися з кимось по телефону або обмінюватися SMS-повідомленнями.

Крім того, застосункам для мобільних пристроїв властива яскраво виражена спеціалізація, що забезпечує максимально зручний доступ користувача до обмеженого набору певних можливостей, що й відрізняє пристрої від того універсального дослідницького середовища, створення якого характерно для успішних застосунків, що виконуються на настільних обчислювальних машинах. Оскільки управління мобільними пристроями найчастіше здійснюється однією рукою або за допомогою невеликого екрану, дуже важливо забезпечити користувача можливостями оперативної ідентифікації та швидкого знаходження потрібної йому інформації чи коштів. Можливість швидкої навігації в межах нечисленного набору ключових засобів є важливим показником якості розробки мобільного застосунку.

Робота за настільними комп'ютерами, як правило, ведеться протягом тривалих робочих сеансів. Самими звичайними є годинні сеанси, однак нерідко вони затягуються на кілька годин. У процесі тривалої роботи формуються і розробляються ідеї, а сама робота носить ітеративний характер з властивими йому поверненнями до вже пройдених етапів. Тому більше значення має не скорочення тривалості підготовки до роботи, а надання користувачеві достатнього набору різноманітних засобів, які можуть йому знадобитися в процесі дослідження та обробки відповідної інформації.

Лептопи в основному використовуються так само, як і настільні комп'ютери, однак і в цьому випадку вже можна помітити ознаки, характерні для використання мобільних пристроїв. Більше значення надається скороченню часу запуску (або пробудження) програми та можливості швидкого переходу користувача до тієї роботи, якою він займався до завершення попереднього робочого сеансу. Разом з тим, за характером їх використання лептопи ближче до настільних комп'ютерів, ніж до мобільних пристроїв, що частково пояснюється просто близькістю фізичних розмірів відповідних категорій устаткування. Поки ви дістанете комп'ютер із сумки, відкриєте кришку, запуститься комп'ютер, доберетесь до застосунку, з яким збираєтеся працювати, або підключитесь, якщо це необхідно, до мережі інтернет, користувач мобільного телефону встигне вирішити масу завдань. Тому лептопи дуже далекі від моделі типу «дістань пристрій з кишені та працюй».

Що ж стосується мобільних пристроїв, то аналогічні дії користувач може виконати набагато швидше, і в типових випадках на це йде від декількох секунд до декількох хвилин. Такі операції, як виконання телефонного дзвінка, перегляд розкладу зустрічей або відправка і читання повідомлень, здійснюються в процесі часто повторюваних, але короткочасних сеансів. У разі мобільних пристроїв навіть ігри відносяться до короткочасних видів діяльності. Зазвичай мобільні пристрої використовуються для ігор протягом коротких проміжків часу в перервах між виконанням інших операцій. Ігри призначені для того, щоб допомогти людям скоротати час в очікуванні поїзда, довгого перебування в аеропорту або в очікуванні партнера, який спізнюється на зустріч з вами. Саме фактори короткочасності використання і можливості негайного доступу до необхідних засобів і змушують нас тримати мобільні пристрої в «постійно включеному» стані або в стані «негайної готовності до включення», в яких час, необхідний для отримання доступу до пристрою, можна порівняти з часом, який йде на вилучення пристрою з кишені.

Важливо відзначити, що незважаючи на короткочасність робочих сеансів, дані, з якими працює користувач, нерідко мають довготривалий характер. Користувач мобільного пристрою може захотіти продовжити гру з того місця, на якому вона була перервана під час попереднього сеансу.

У процесі ведення переговорів по телефону або під час особистих зустрічей часто доводиться звертатися до розкладу зустрічей і заносити в нього дату та час чергової зустрічі. Користувач повинен мати можливість швидко отримати доступ до пристрою, перейти до списку особистих контактів, створити новий пункт розкладу та одразу ж продовжити ведення переговорів без будь-якої затримки. Відповідні дані залишаються актуальними протягом тривалого часу і повинні зберігатися довго, але сам застосунок використовується протягом коротких періодів часу в режимі негайного доступу до інформації.

1.2 Ключові моменти при розробці мобільних застосунків

Порівнянню і протиставленню мобільних пристроїв і їх застосунків з їх настільними і серверними аналогами ми відвели в цьому розділі досить багато місця і часу. Тому зараз буде вельми доречно перерахувати все те, що робить мобільний застосунок дійсно якісним.

1. Час запуску.

Важливою характеристикою мобільних застосунків є час їх запуску. Оскільки мобільними пристроями зазвичай користуються часто і протягом нетривалих проміжків часу, здатність мобільного застосунку до швидкого запуску є обов'язковою вимогою. Можливо, 6-секундне споглядання заставки текстового процесора, енциклопедії або середовища розробки в очікуванні появи основного вікна відповідної програми і приносить мало задоволення, однак, з урахуванням загальної тривалості типового сеансу роботи з такими застосунками, цей фактор можна вважати малозначним.

У разі ж мобільного застосунку, який користувач збирається використовувати протягом 20 секунд для уточнення або поновлення невеликої порції інформації, втрата 6 секунд представляється недозволеною розкішшю. Розумно керуватися мнемонічним правилом, згідно з яким тривалість сеансу роботи користувача з застосунком повинна набагато перевершувати тривалість запуску цього застосунку. У випадку застосунків для настільних комп'ютерів робочі сеанси тривають довше, і тому користувачі готові миритися з періодами очікування більшої тривалості. У разі ж мобільних застосунків тривалість робочих сеансів менше, і тому висувається вимога, щоб тривалість періоду запуску програми була відповідно меншою.

Тривалість періоду запуску застосунку є важливим чинником і у випадку настільних комп'ютерів, але у випадку мобільних пристроїв, які характеризуються нерегулярністю і короткочасністю робочих сеансів, цей чинник набуває вирішальне значення.

2. Відгук пристрою.

Сприймаючи мобільні пристрої як слухняні механічні іграшки, люди очікують від них відповідної поведінки. Коли користувач впливає на маніпулятор, натискає на кнопку або будь-яким іншим чином маніпулює елементами керування пристрою, він розраховує на фізичну реакцію з його боку. Не отримавши негайної реакції з боку пристрою, нетерплячий користувач починає нервувати і тут же повторює спробу виконання потрібної операції.

Якщо повторна спроба маніпуляції елементом управління, виконання клацання чи іншою аналогічною дією будуть оброблені вашим застосунком, або, помилково, іншим застосунком, то в результаті цього можуть виникнути проблеми. Тому вкрай важливо дбати про те, щоб після виконання яких-небудь дій з пристроєм користувач отримував від нього негайне підтвердження реакції на ці дії в тій чи іншій формі.

Ідеальним варіантом такого підтвердження є виконання запитаної операції; краще цього нічого бути не може. На другому місці в цьому відношенні стоїть підтвердження того, що запит отримано і обробляється у фоновому режимі, в той час як застосунок зберігає здатність сприймати інші запити. Третє місце належить відображенню курсору очікування, який повідомлятиме користувача про те , що запит обробляється; застосунок ні на які подальші запити не реагує, але користувач знає, що роботу з обслуговування його запиту розпочато і виконується. Найгірший з варіантів – це коли користувач не бачить ніякої відповідної реакції з боку пристрою і йому залишається лише здогадуватися про те, сприйнятий його запит чи ні. Як і у випадку інших характеристик, які ми будемо вивчати, ці вимоги не є унікальними для мобільних пристроїв, але в даному випадку вони набувають особливого значення в силу специфіки роботи з цими пристроями і того, чого люди від них чекають.

3. Фокусування уваги на окремих завданнях.

Ще однією характеристикою мобільного застосунку, що визначає його успішність , є ступінь його спеціалізації. Для програми повинен бути чітко визначений набір завдань, ефективне вирішення яких вона має забезпечувати; здійснювані при цьому операції повинні виконуватися швидко і вимагати лише мінімальної кількості клацань, натискань або інших дій з боку користувача.

Саме з цієї причини в пристроях часто передбачаються спеціальні кнопки, що відповідають окремим завданням, які здатен вирішувати даний пристрій (наприклад, кнопка для переходу безпосередньо до бази даних, що зберігає інформацію про контакти, кнопка для перегляду календаря зустрічей або кнопка для зчитування штрих – коду та передачі його певному застосунку).

Ви повинні докласти максимум зусиль до того, щоб якомога точніше визначити коло тих регулярних завдань, вирішення яких має вимагати мінімуму дій з боку користувача. Цією рекомендацією слід керуватися як щодо високорівневих функцій програми, так і щодо низькорівневих завдань, які зазвичай доводиться вирішувати користувачам в процесі роботи з даним застосунком. Наприклад, якщо мобільний застосунок має забезпечувати швидке введення дат, то ви повинні подбати про максимальне спрощення і прискорення зазначеного процесу і вимірюванні його ефективності. Якщо ваш мобільний застосунок повинен забезпечувати можливість вказівки певного місця розташування за допомогою схеми міських вулиць, щоб користувач міг вибрати маршрут руху до кінцевого пункту призначення, то ви повинні подбати про те, щоб це могло бути зроблено якомога швидше.

Одна з поширених помилок, що допускаються розробниками при створенні застосунків для мобільних пристроїв, полягає в тому, що програмний код намагаються зробити якомога більш коротким, щоб тим самим максимально зменшити розмір програми. Хоча подібні цілі і є благородними, вони не повинні досягатися за рахунок зниження продуктивності праці користувачів. Не шкодуйте часу на написання додаткового коду, фокусую чого увагу користувачів на виконанні спеціалізованих завдань і що дозволяє скоротити час отримання кінцевого результату.

4. Налаштування взаємодії із зовнішніми джерелами інформації.

Дуже важливо розуміти, що створення відмінного мобільного застосунку – це не лише написання коду, який правильно виконується на пристрої. Не слід забувати також про зовнішнє програмне забезпечення, з яким взаємодіє ваше мобільний застосунок. Необхідно, щоб у застосунку відповідним чином враховувалися особливості інформаційних джерел, що обслуговують мобільні пристрої, і здійснювалася перевірка того, що інформація надається в тій формі, яка відповідає даному пристрою. Як приклад можна привести служби електронної пошти для мобільних пристроїв. Багатофункціональні поштові застосунки вимагають наявності відповідного програмного забезпечення як на стороні сервера, так і на стороні клієнта. Клієнт здійснює доступ до сервера для отримання інформації щодо

повідомлень, що надійшли і завантажує відповідний вміст в локальний пристрій. Оскільки мережеві з'єднання, використовувані мобільними пристроями, відрізняються нерегулярністю доступу до них, зниженою пропускною здатністю, а в багатьох випадках і більш високою вартістю в порівнянні з тими з'єднаннями, які використовуються персональними комп'ютерами, поштові служби сервера повинні бути пристосовані для задоволення цих вимог. Зазвичай це досягається за рахунок того, що на сервері є засоби, що дозволяють обмежувати обсяг отриманого вмісту, або фільтри, пропускають лише ту інформацію, яка дійсно може знадобитися користувачеві мобільного пристрою. Для забезпечення підтримки ефективних сценаріїв при обслуговуванні мобільних пристроїв може знадобитися розширення функціональних можливостей тих служб на стороні сервера, які спочатку призначалися для обслуговування персональних комп'ютерів. Крім того, повинні бути передбачені такі механізми змін параметрів конфігурації програми для його виконання на серверах, настільних комп'ютерах і самих мобільних пристроях, за допомогою яких користувачі зможуть визначати свої інформаційні запити і налаштовувати інформаційні фільтри таким чином, щоб забезпечувалося виконання відповідних вимог. Проектування мобільних застосунків далеко не обмежується фізичними межами самих мобільних пристроїв.

5. Особливості стилю інтерфейсу.

З урахуванням того, що будь-який мобільний пристрій відрізняється компактністю і самодостатністю, бажання користувачів працювати з інтерфейсом, який не залежить від характеру розв'язуваних за допомогою пристрою завдань, видаються цілком природними. Кожний мобільний пристрій має власний функціональний колорит. Типовий вдало спроектований застосунок сприймається не як розрізнений набір засобів, а як гармонійне розширення можливостей самого пристрою.

З цієї причини при проектуванні застосунків для мобільних пристроїв дуже важливо дотримуватися єдиного стилю. Способи запуску і зупинки застосунків, переміщення в межах інтерфейсу, а також організації діалогів між користувачем і застосунками повинні ретельно продумувати окремо для кожного типу пристроїв. Користувачі мобільних пристроїв підсвідомо підлаштовуються під метафори користувача інтерфейсу, і будь-які відхилення від звичних шаблонів доставляють їм відчутні незручності. Створення звичного робочого середовища багато значить і в разі застосунків для настільних комп'ютерів, але в силу різноманітності можливостей, які пропонуються такими застосунками, вони дозволяють досягнути однієї і

тієї ж мети декількома способами (наприклад, за допомогою сполучень клавіш, клацань мишею, меню і панелей інструментів). У разі ж мобільних пристроїв для вирішення даної задачі часто є тільки один шлях, і користувач мимоволі звикає до одного певного способу. Набагато краще мати чотири різних версії застосунку, кожна з яких відповідає певним властивостям інтерфейсу користувача конкретного пристрою, ніж один загальний застосунок, який не може бути інтегрованим належним чином ні в один цільовий пристрій.

6. Відмінності в архітектурі платформ.

Якщо скористатися аналогією з області архітектури, то настільні та переносні комп'ютери можна уподібнити великим садибам у сільській місцевості. Тоді відносно мобільних пристроїв можна сказати, що вони схожі на окремі чи комунальні міські квартири. Кожен з цих типів жител призначений для задоволення певних потреб і висуває специфічні вимоги щодо найбільш ефективних способів їх використання.

Зазвичай садиби характеризуються великими розмірами, і в них є багато місць, придатних для зберігання різних речей. Одні з таких місць знаходяться неподалік, тоді як дістатися до інших дещо важче. У садибах рідко використовувані речі зазвичай зберігаються на горищах або в підвалах, звідки їх можна періодично витягувати по мірі того, як в них виникає необхідність.

З іншого боку, в умовах міських квартир, місць, придатних для зберігання речей, значно менше. У квартирах зберігають речі, які повинні бути завжди під рукою. Ті ж речі, якими користуються рідко, в квартирі не тримають. Замість того щоб купувати рідко використовувані речі, їх часто вигідніше орендувати.

Те ж саме справедливо і щодо мобільних пристроїв. Як оперативна пам'ять, так і пам'ять для довготривалого зберігання даних орієнтовані на зберігання тієї інформації, яка використовується найчастіше. Рідко використовувані дані повинні зберігатись на серверах, звідки їх можна буде запросити, коли вони будуть потрібні.

У настільних комп'ютерах ОЗУ і довготривала пам'ять розділені. Водночас, в пристроях ОЗУ часто використовуються і як робоча пам'ять застосунків, і як проміжна або довгострокова пам'ять. В якості довгострокових сховищ даних все частіше використовується флеш-пам'ять, зазвичай у вигляді змінних карт пам'яті.

1.3 Перспективи розвитку

Можливості мобільних застосунків давно вийшли за рамки індустрії розваг і їх використання дозволяє мати додаткові преференції у сфері бізнес-інтересів. Мобільні інформаційні технології сприяють

поліпшенню сервісного обслуговування споживчої аудиторії, а також розвитку торгових і виробничих відносин. Крім того, не варто скидати з рахунків комерційний потенціал розважальних застосунків, цей ринок як і раніше надзвичайно привабливий і має зростаючі перспективи. Доцільність замовлення мобільних застосунків для бізнесу можна обґрунтувати відразу декількома факторами:

Користувачі інтернету (потенційні споживачі) все частіше виходять в мережу за допомогою мобільних пристроїв, а ряд застосунків дозволяє фактично прив'язати до себе клієнтів і цільову аудиторію.

Можливість підключатися до серверів, які працюють за галузевим стандартам, і бути на зв'язку з корпоративними мережами, забезпечують практично повну незалежність та оперативне реагування при вирішенні нагальних питань.

Динаміка розвитку ринку мобільних застосунків підтверджує прогнози про високе зростання доходів в короткостроковій перспективі. Причому це не чергова банківська «мильна бульбашка», роздута з чергових махінацій, а реальний продукт, який здатний принести відмінний прибуток.

Забезпечується більш тісний контакт з представниками найбільш платоспроможної цільової аудиторії, так як в основному саме вони є власниками багатофункціональних мобільних пристроїв.

Якщо подивитися на сучасні напівпровідникові технології, то ми практично дійшли до меж можливого. Зараз шар діелектрика в транзисторі становить 10–15 атомів. Теоретична межа – це п'ять-шість атомів, і практично у розвитку традиційної напівпровідникової електроніки ми вже до нього наблизилися. Тому майбутнє за новими технологіями, і такі нові технології розвиваються в надрах лабораторій IBM. Одне з інноваційних напрямів – це наноелектроніка. У 1998 році вчені Дональд Ейтлер і Ерхард Швейцер з лабораторії IBM в Каліфорнії зуміли викласти 35 атомами ксенону на кристалі нікелю логотип IBM. Це одне з перших звершень у галузі нанотехнології, коли людство навчилося керувати структурами фактично на атомарному рівні.

Подальший розвиток цей напрям отримав з винаходом так званих кремнієвих трубок. Кремнієва трубка на атомарному рівні може бути наповнена якимось вмістом і змінює свої якості залежно від геометрії і того, яким матеріалом вона наповнена. Розмір такої трубки становить 1–1,5 нанометра. Фактично це дозволяє створювати інтегральні мікросхеми на принципово іншому рівні і в принципово інший щільності.

Причому ці нанотехнології вже вийшли за межі теоретичних розробок. Так, в лабораторії IBM в Швейцарії вчений Герд Біннінг, який отримав Нобелівську премію за розробку електронного мікроскопа,

звернув увагу на можливість формування в певних полімерах маленьких ямочок, які мають розмір нанометрів. І мало того, ці ямочки можна як створювати, так і сканувати, що дозволило створити принципово нову технологію запису інформації. Щільність запису на таких пристроях пам'яті становить порядку 1 Тбайт на квадратний дюйм, це 25 мільйонів друкованих аркушів на кристалі розміром з поштову марку. Тобто на одному маленькому чипі пам'яті, створеному за цією технологією, можна записати 15–20 Гбайт інформації. І цього року такий запам'ятовуючий пристрій було показано на виставці. Це не означає, що воно вже сьогодні готове до промислового випуску, але через два-три роки він вже буде можливий.

Існують і інші основи для розвитку інформаційних технологій. Такі, наприклад, як квантова обробка інформації, роботи з якої теж ведуться в надрах лабораторій IBM [3].

2 Найпопулярніші мобільні операційні системи та їх характеристика

2.1 Google Android System

Android – операційна система для смартфонів, планшетів і нетбуків. Компанія Google придбала розробника програмного забезпечення Android inc. в 2005 році. Операційна система Android заснована на модифікованому ядрі Linux. Згодом, Google та інші учасники Open Headset Alliance співпрацювали для спільної розробки цієї нової операційної системи. Далі Android Open Source Project (AOSP) доручено підтримання та подальший розвиток платформи. У Android є велика спільнота розробників, які розширюють функціональність пристроїв.

Операційна система (ОС) OS Android має свій офіційний магазин з продажу застосунків – Android Market. Включає він в себе як платні програми, так і безкоштовні. В даний момент, для України доступні для завантаження тільки безкоштовні програми та ігри. Так як OS Android є відкритою, користувачу надається можливість завантажувати застосунки та з інших ресурсів.

Під Android розробники, в основному, пишуть програми на мові Java, що керують пристроєм через розроблені Google бібліотеки.

Офіційно про OS Android стало відомо 5 листопада 2007 року, коли було оголошено підставу Open Headset Alliance – консорціум з 80 компаній. Велика частина коду Android була випущена під ліцензією Apache.

Android програми включають в себе java-застосунки та бібліотеки, які запускаються віртуальною машиною Dalvik з JIT-компілятором (англ. Just-in-time compilation, компіляція «на льоту»). Бібліотеки включають в себе

систему управління, графіку OpenGL ES 2.0, движок WebKit, графічний движок SGL, SSL і бібліотеки Bionic. OS Android складається з 12 мільйонів рядків коду, в тому числі 3-х мільйонів рядків XML, 2.8 мільйонів рядків на C, 2.1 мільйона рядків на Java і 1.75 мільйона рядків на C++. Компанія Android inc. була заснована в жовтні 2003 року в Пало Альто, штат Каліфорнія. Засновниками Android inc. були Енді Рубін, Річ Майнер, Нік Сірс і Кріс Уайт.

У серпні 2005 року компанія Google придбала Android inc. Після цього Android inc. стала дочірньою компанією Google. Після покупки Енді Рубін, Річ Майнер і Кріс Уайт залишилися в Android inc. Після поглинання Android inc., В мережі почали з'являтися чутки про те, що Google хоче вийти на ринок мобільних телефонів.

Отримавши підтримку Google, команда на чолі з Енді Рубіном почала працювати над операційною системою що базується на ядрі Linux. Тоді ж, у грудні 2006 року поповзли чутки про те, що Google планує випустити смартфон під своїм брендом, так званий «Гуглофон».

Всі ці чутки спростував Ерік Шмідт, заявивши наступне: «Сьогоднішня заява носить більш амбітний характер, ніж випуск смартфона під нашим брендом, про що преса спекулює останні тижні. Наше бачення полягає в тому, що найпотужніша платформа повинна віддати свої сили тисячам різних моделей телефонів». Після цих слів 5 листопада 2007 року і був представлений Open Headset Alliance – консорціум з безліччю компаній, до якого увійшли такі гіганти як: Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, TMobile, Texas Instruments та інші.

Назву кожної чергової версії ОС Android представляє собою назву якого-небудь десерту. Перші букви найменувань в порядку версій відповідають літерами латинського алфавіту.

Android Market-інтернет-магазин застосунків для смартфонів на базі Android, яку просуває альянс Open Handset Alliance (OHA) на чолі з Google. Включає в себе ігри, клієнти соціальних мереж, офісні застосунки, застосунки для читання новин, управління фінансами та інші.

22 жовтня 2008 року Google оголосила про відкриття цього онлайн-магазину застосунків для Android.

В Україні на смартфонах перший час був відсутній офіційний магазин застосунків. Ситуація була виправлена 12 січня 2010 року, коли про це повідомили в Samsung Україна. Зрозуміло, що пізніше Android Market з'явився і у інших вендорів – Motorola, HTC, LG і Sony Ericsson, що значно підігріло інтерес до Android-смартфонів в Україні.

Першим офіційним смартфоном в Android Market в Україні став Samsung i5700 Galaxy Spica.

Програми для Android є програмами в нестандартному байт-коді для віртуальної машини Dalvik.

Google пропонує для вільного скачування інструментарій для розробки (Android SDK), який призначений для x86-машин під операційними системами Windows XP, Windows Vista, Mac OS X (10.4.8 або вище) і Linux. Для розробки потрібно JDK 5 або JDK 6.

Розробку застосунків для Android можна вести на мові Java (не нижче Java 1.5). Існує плагін для Eclipse – «Android Development Tools» (ADT), призначений для Eclipse версій 3.3–3.5. Для IntelliJ IDEA також існує плагін, який полегшує розробку Android-застосунків. Повідомляється, що для середовища розробки NetBeans IDE розроблено експериментальний плагін. Архітектура OS Android представлена на рис. 2.1.

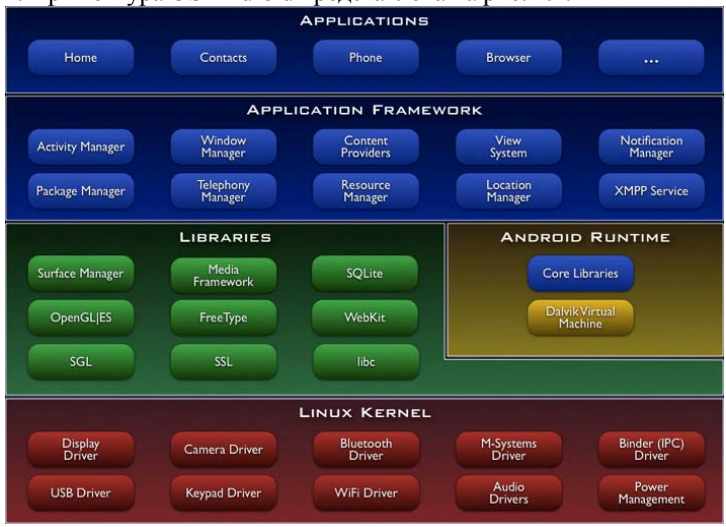

Рисунок 2.1– Архітектура OS Android

Рівень застосунків (Applications):

До складу Android входить комплект базових застосунків: клієнти електронної пошти та SMS, календар, різні карти , браузер, програма для управління контактами і багато іншого. Всі застосунки, що запускаються на платформі Android написані на мові Java.

Рівень каркаса застосунків (Application Framework):

Android дозволяє використовувати всю міць API, використовуваного в застосунках ядра. Архітектура побудована таким чином, що будь-який застосунок може використовувати вже реалізовані можливості іншої

програми за умови, що остання відкриє доступ на використання своєї функціональності. Таким чином, архітектура реалізує принцип багаторазового використання компонентів ОС і застосунків.

Основою всіх застосунків є набір систем і служб.

1. Система уявлень (View System) – це багатий набір уявлень з розширюваною функціональністю, який служить для побудови зовнішнього вигляду застосунків, що включає такі компоненти, як списки, таблиці, поля введення, кнопки і т.п.

2. Контент-провайдери (Content Providers) – це служби, які дозволяють застосункам отримувати доступ до даних інших застосунків, а також надавати доступ до своїх даних.

3. Менеджер ресурсів (Resource Manager) призначений для доступу до строкових, графічних та інших типів ресурсів.

4. Менеджер сповіщень (Notification Manager) дозволяє будь-якому застосунку відображати для користувача повідомлення в рядку статусу.

5. Менеджер дій (Activity Manager) управляє життєвим циклом застосунків і надає систему навігації по історії роботи з діями.

Рівень бібліотек (Libraries):

Платформа Android включає набір C/C++ бібліотек, використовуваних різними компонентами ОС. Для розробників доступ до функцій цих бібліотек реалізований через використання Application Framework. Нижче представлені деякі з них:

1. System C library – BSD – реалізація стандартної системної бібліотеки C (libc) для вбудованих пристроїв, заснованих на Linux.

2. Media Libraries – бібліотеки, засновані на PacketVideo's OpenCORE, призначені для підтримки програвання і запису популярних аудіо- та відео-форматів (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG і т.п.).

3. Surface Manager – менеджер поверхонь управляє доступом до підсистеми відображення 2D- і 3D-графічних шарів.

4. LibWebCore – сучасний движок web-браузера, який надає всю міць вбудованого Android-браузера.

5. SGL – рушій для роботи з 2D-графікою.

6. 3D libraries – рушій для роботи з 3D-графікою, заснований на OpenGL ES 1.0 API.

7. FreeType – бібліотека, призначена для роботи зі шрифтами.

8. SQLite – потужний легковаговий движок для роботи з реляційними БД.

Рівень середовища виконання (Android Runtime):

До складу Android входить набір бібліотек ядра, які надають велику частину функціональності бібліотек ядра мови Java.

Платформа використовує оптимізовану, регістр-орієнтовану віртуальну машину Dalvik, на відміну від неї стандартна віртуальна машина Java-стек-орієнтована. Кожна програма запускається у своєму власному процесі, зі своїм власним примірником віртуальної машини. Dalvik використовує формат Dalvik Executable (*.Dex), оптимізований для мінімального використання пам'яті застосунком. Це забезпечується такими базовими функціями ядра Linux, як організація потокової обробки і низькорівневе управління пам'яттю. Байт-код Java, на якому написані ваші програми, компілюються в dex – формат за допомогою утиліти dx, що входить до складу SDK.

Рівень ядра Linux (Linux Kernel):

Android заснований на ОС Linux версії 2.6, тому платформі доступні системні служби ядра, такі як управління пам'яттю і процесами, забезпечення безпеки, робота з мережею і драйверами. Також ядро служить шаром абстракції між апаратним та програмним забезпеченням [4].

2.2 Apple IOS

Щоб ваш мобільний помічник міг виконувати свої завдання одного «заліза» й батареї недостатньо. Потрібна ще програмна начинка, яка забезпечить потрібний функціонал пристрою, тобто операційна система та застосунки. Так які ж мобільні версії iOS бувають і чим один від одного відрізняються (рис. 2.2)?

Рисунок 2.2 – iOS 6 проти iOS 7

Напевно в світі залишилося зовсім мало людей, які не чули про ігрові ставки, телефонах iPhone та інших чудових продуктах «яблучної» компанії, а більш допитливі розуми напевно чули і про iOS, операційну систему Apple, під якою працюють такі її продукти, як IPad Touch, iPhone і Ipad.

Інтерфейс:

У момент появи першого iPhone, інтерфейс був досить інноваційним для смартфона (рис. 2.3). Великі іконки на які легко і зручно натискати пальцем. Велика клавіатура також зручна для пальців. Кілька віртуальних робочих столів з іконками для організації застосунків. Нічого зайвого. Тільки сторінки з іконками, ніяких вам меню або діалогових вікон.

Рисунок 2.3 – Інтерфейс iPhone

До речі, таке цікаве явище як док (Doc), або віртуальна поличка, куди ставляться найбільш затребувані застосунки, теж вперше було масово поширено в продуктах Apple. Як бачите, в інтерфейсі iOS він теж є і може зберігати до 4 іконок.

Інтерфейс iOS стандартизований і не змінювався від версії до версії. Налаштувати його за своїм смаком, вам також навряд чи вдасться. Ви мало що можете змінити.

Архітектура iOS:

IOS спроектована таким чином, щоб, по-перше, зробити систему стабільною, по-друге, знизити витрату електрики і продовжити життя акумулятора. Навіть якщо це означає, що доведеться відмовитися від частини функціоналу і прикрас.

Кожна програма в iOS запускається в «пісочниці», що виключає можливість впливу однієї програми на інші, як на рівні файлової системи, так і на рівні оперативної пам'яті. На практиці це означає, з одного боку, стабільність системи і те, що iOS не зависає, також якщо застосунок App1 завершується некоректно (просто закривається саме по собі або «глючить»), це ніяк не вплине на роботу застосунків App2 і App3.

Ізоляція застосунків сильно підвищує безпеку iOS, оскільки застосунки можуть звертатися лише до файлів у своїй «пісочниці», так що ваші замітки ніяк не зможуть бути таємно або випадково відправлені, скажімо, в блог через застосунок Livejournal App.

Однак, не обходиться без незручностей. Наприклад, якщо ви завантажили ролик з мережі через спеціальну «гойдалку» ви не зможете переглянути його, якщо в «гойдалці» не має функціоналу відео-плеєра. І перекинути файли з одного застосунку в інші немає ніякої можливості.

Ще одна цікава особливість – це поділ ресурсів. Запущеному застосунку в iOS віддаються всі доступні ресурси, а всі інші програми закриваються. Ось чому таке явище як «гальма» можна вкрай рідко зустріти на пристроях Apple. Більш того, в стані коли всі програми закриті, можна бути впевненим, що «в тілі» не залишилося якогось сервісу, який з'їсть всю батарейку або викачає весь ліміт мобільного трафіку, поки ви обідаєте або спите.

Хоча стабільність та енергозбереження це дуже важливо, але в ряді випадків однозначність зводить нанівець всі плюси такого підходу. Уявіть собі, що ви спілкуєтеся по скайп-чату або через інший месенджер і вам присилають щось на кшталт «Привіт, я там прислав тобі на пошту презентацію...». На жаль, для того, щоб ви змогли дістатися до пошти вам доведеться закрити Skype, і відкрити пошту, потім все спочатку, але у зворотний бік. Точно також можна «докачувати» ролик з YouTube паралельно читаючи книгу або відправляючи твіт.

Втім, буде несправедливо не сказати про деякі, виправдовуючі ситуацію, моменти. Майже у всіх пристойних програмах для спілкування є вбудовані браузери, що дозволять швидко переглянути надіслані посилання, це перше. Друге, майже всі програми, які написані в самій Apple, (браузер, будильник, пошта, замітки...) можуть бути відкриті повторно і ви знайдете їх в тому вигляді, в якому залишили. І, зрозуміло, музику можна слухати завжди [5].

У iOS 7 з'явилася-таки багатозадачність, тобто можливість запускати кілька застосунків паралельно, однак вона працює не з усіма застосунками і не завжди так, як треба.

Остання архітектурна особливість IOS це деякі непорозуміння. Так, не дивлячись на відмінну «залізну» начинку, IOS не підтримує передачу файлів через Bluetooth. Також не можна використовувати Bluetooth-гарнітуру для пристроїв IPod Touch, хоча є можливість приєднати провідний мікрофон .

Програми iOS:

Для пошуку та установки застосунків у Apple, є спеціальний сервіс, який називається AppStore (магазин застосунків, якщо дослівно). Застосунки бувають платними (від 0,99 до 900 $) і безкоштовними. Всі застосунки, перш ніж потрапити в AppStore проходять перевірку на ідеологічну складову і нешкідливість. Така санітарна обробка дозволяє

уникнути шкідництва з боку розробників і захистити особисті дані (і гроші) користувачів від крадіжки.

Більшість застосунків в AppStore не представляють інтересу і часто дублюють один одного. Одних тільки будильників можна знайти з півсотні, не кажучи вже про календарі, тижневики і записні книжки. Однак, серед них є маса корисних, які допоможуть вам організувати свої справи, підкажуть, направлять і розважать, та й взагалі, бродити по AppStore майже також цікаво, як по справжньому магазину.

Крім AppStore також існує магазин Cydia. Cydia можуть користуватися ті, хто попередньо розблокував свій телефон через процедуру джейлбрейка. У Cydia можна знайти масу корисних застосунків (як платних, так і безкоштовних), які не змогли потрапити в AppStore або ніколи туди не прагнули. Оскільки Apple, фільтрує застосунки вельми прискіпливо, багато з програм змушені розміщуватися через цей альтернативний магазин. Втім, не будемо псувати вам задоволення від дослідницької діяльності.

Jailbreak и Unlock:

Не всі люди готові миритися з обмеженнями, які Apple накладає на їхні пристрої. Зокрема при покупці телефону iPhone, ви, швидше за все, придбаєте його у оператора на умовах контракту або у перекупника, який продасть вам телефон, який був до цього придбаний через оператора.

Проблема полягає в тому, що за умовами поставки iPhone оператор зобов'язаний укласти з клієнтом дворічний контракт за яким клієнту слід виплачувати певну суму щомісяця за деяке число розмовних хвилин і мобільного інтернету. Більше того, ви не зможете перейти до іншого оператора зберігши за собою телефон, поки не закінчитися термін дії договору. Те ж правило поширюється і на роумінг. Коротше кажучи ви не можете використовувати телефон інакше як з сім-картою оператора, у якого придбали iPhone.

Втеча з в'язниці (втеча з в'язниці, дослівно) це процедура, що дозволяє отримати повний контроль над iOS, тобто встановлювати будь-які застосунки, переглядати всю файлову систему, змінювати системні налаштування, а також активувати багатозадачність на iPhone 3G і IPod Touch 2-го покоління і багато іншого.

Однак, слід врахувати, що процедура суттєво впливає на стабільність пристрою і знижує рівень безпеки, крім того, акумулятор може почати розряджаються швидше.

Взаємодія з ПК:

Пристрої під керуванням iOS 4 можуть працювати як з Mac'ами, так і з вікнами або Linux. Сумування полягає в тому, що якщо в Mac'ах Itunes – це стандартний застосунок, то в Windows, вам доведеться завантажити

близько 100Мб софта, щоб просто закачати музику в плеєр. Те ж саме стосується й інших операцій, таких як резервне копіювання, оновлення прошивки, синхронізація контактів і т.д. Без ITunes ви не зможете активувати свій iPhone або зареєструватися для роботи з AppStore. До речі, ваші налаштування і музика можуть бути синхронізовані тільки з ОДНИМ пристроєм, також як ваш пристрій може бути синхронізоване для роботи тільки з одним ПК.

Загалом, звучить не дуже, правда? Насправді з цим цілком можна жити, головне звикнути. З Linux справи йдуть трохи краще, як не дивно, ви можете працювати зі своєю музичною колекцією без всяких проблем через такі програми як Rhythmbox (стандартний плеєр Ubuntu), свої календарі та контакти можна синхронізувати через акаунт у Google, а застосунки встановлювати через мобільний інтерфейс AppStore [6].

2.3 Windows Phone

Windows Phone (рис. 2.4) – мобільна операційна система, розроблена Microsoft, вийшла 11 жовтня 2010 року, а 21 жовтня почалися поставки перших пристроїв на базі нової платформи. У Росії телефони з Windows Phone почали продаватися 16 вересня 2011 року, першим з яких став HTC 7 Mozart.

Рисунок 2.4 – Логотип Windows Phone

Операційна система є наступником Windows Mobile, хоча і несумісна з нею, з повністю новим інтерфейсом і – вперше – з інтеграцією сервісів Microsoft: ігрового Xbox Live і медіаплеєра Zune. Презентація системи відбулася в рамках заходу Mobile World Congress 2010 року, що пройшов у Барселоні.

Windows Phone підтримує технологію мультитач і має новий користувальницький інтерфейс під назвою Metro , принципи якого були раніше використані в дизайні інтерфейсу Windows Media Center, Zune і Xbox. Також в Windows Phone знайшли відображення ідеї платформи смартфонів Microsoft Kin.

Microsoft переробила початковий екран: тут більше немає статичних іконок – всі вони замінені на так звані «живі плитки» (або «іконки», «тайли»), які відображають інформацію в режимі реального часу без участі користувача. «Живі плитки» прокручуються по вертикалі і можуть служити як звичайним ярликом для програми, так і ярликом для контакту, замітки, веб-сторінки, медіаконтенту та іншого.

Більшість «плиток» мають квадратну форму, але поряд з ними інтерфейс операційної системи містить і прямокутні (фото, календар).

Одне з нововведень інтерфейсу. «Хаби» (англ. Hub) або, як описано на офіційному сайті, «розділи». Всього в системі є 6 таких розділів, але виробники телефонів можуть створювати свої, наприклад, HTC Hub, відповідно від компанії HTC. Розділ «Контакти» (англ. People Hub) об'єднує всю інформацію, що стосується якоїсь певної людини, в тому числі її записи та коментарі в соціальних мережах, а також фотографії, надаючи централізований доступ до таких мереж, як facebook, Twitter, LinkedIn і Windows Live. З Windows Phone 8 в розділі «Контакти» з'явилася можливість створювати «кімнати»: список певних контактів, яким доступна та чи інша інформація. Розділ «Фото» об'єднує фотографії користувача, що зберігаються в пам'яті пристрою, на комп'ютері та в Інтернеті. За допомогою цього розділу можна опублікувати фотографію в соціальних мережах, помітити на ній людину, а також мати доступ до останніх фотографій від друзів. Розділ «Ігри» відкриває доступ до ігор і сервісу Xbox Live, де можна подивитися свої чи чужі профілі, аватари, досягнення. Розділ «Музика + відео» об'єднує весь мультимедійний контент, що зберігається на комп'ютері користувача, музичні онлайн-сервіси, вбудоване FM-радіо і відкриває доступ до сервісів Xbox Музика і Xbox Video. Розділ «Магазин» дозволяє купувати або завантажувати безкоштовно програми та ігри, а розділ Управління забезпечує доступ до мобільної версії Офіс-Office Mobile, включаючи Word, Excel, OneNote, PowerPoint. За допомогою офісного пакету користувач має можливість відкрити, створити і редагувати документи, а потім синхронізувати зроблені зміни з настільним комп'ютером, використовуючи сервіс OneDrive.

Windows Phone 7:

15 лютого 2010 року в Барселоні на Mobile World Congress 2010 Стівом Балмером вперше була анонсована нова мобільна операційна система – Windows Phone 7. Офіційно була представлена 11 жовтня 2010 року, а 21 жовтня у продажу з'явилися перші смартфони на новій платформі в Європі (Великобританії, Німеччини, Франції, Іспанії та Італії) і Азіатсько-Тихоокеанському регіоні (Сінгапурі, Австралії та Нової Зеландії). Windows Phone 7 була доступна на п'яти мовах.

Windows Phone 7.5:

У лютому 2011 року на Mobile World Congress 2011 корпорація Microsoft вперше анонсувала нове оновлення Windows Phone, а вже у квітні на конференції MIX'2011 в Лас-Вегасі компанія розповіла про подробиці цього оновлення, яке отримало назву Mango. Пізніше компанія офіційно

заявила, що це оновлення операційної системи отримає порядковий номер 7.5 у новій версії, яка вийшла остаточно у вересні 2011 року, більше 500 поліпшень і доповнень, що зробило роботу операційної системи швидше, у системі оновився браузер Internet Explorer Mobile до версії 9, який тепер підтримує HTML5, Canvas, апаратне прискорення. З'явилася можливість використовувати фронтальну камеру. Одна з найважливіших змін для російськомовних користувачів – це локалізація на російську мову – інтерфейс, клавіатура, сумісність з кирилицею. У загальній складності тепер платформа підтримує 24 мови.

У лютому 2012 року на Mobile World Congress 2012 був продемонстрований смартфон Nokia Lumia 610 з черговим оновленням операційної системи, що є частиною Манго, в якому з'явилася підтримка процесорів з частотою 800 МГц та оперативною пам'яттю у 256 Мб. Особливість даного оновлення полягає у можливості випуску на ринок смартфонів бюджетного сегмента. У березні це оновлення було офіційно названо як Windows Phone 7.5 Refresh .

Windows Phone 7.8:

В кінці 2012 року в офіційному блозі Windows Phone розробники повідомили про запланований випуск на початку 2013 року оновлення під номером 7.8, яке включатиме в себе деякі нововведення платформи Windows Phone 8: стартовий екран з налаштованим розміром «живих плиток», новий екран блокування, створення рингтонів, передача даних через Bluetooth. 30 січня 2013 року компанія Nokia офіційно оголосила про початок оновлення своїх апаратів лінійки Lumia до версії 7.8, яка містить обіцяні зміни.

Повноцінного оновлення смартфонів на Windows Phone 7.x до версії 8 не буде, що офіційно підтвердила Microsoft. Офіційна підтримка платформи Windows Phone 7 припиниться через півтора роки після виходу оновлення під номером 7.8, тобто приблизно в середині 2014 року. Версія 7.8 не тільки оновить поточні апарати на Windows Phone 7/7.5, але і буде встановлюватися на бюджетні WP – пристрої [7].

Windows Phone 8

20 червня 2012 року на організованій Microsoft конференції під назвою Windows Phone Summit була анонсована Windows Phone 8. Головна перевага Windows Phone 8 (рис. 2.5) – можливість об'єднати планшети, смартфони та персональні комп'ютери в єдину «екосистему», тобто можливість створення умов для розробників, що полегшують портування програмного забезпечення між цими пристроями. Нове покоління платформи буде засновано на ядрі NT і максимальна роздільна здатність екрану апаратів буде 1280×768 або 1280×720, на

відміну від Windows Phone 7.x, яка заснована на ядрі CE з роздільною здатністю екрану 800×480.

Рисунок 2.5 – Логотип Windows Phone 8

29 жовтня 2012 року Microsoft офіційно представила наступне покоління платформи.

Нововведення Windows Phone 8: Internet Explorer Mobile 10-ї версії з функцією виявлення шкідливих сайтів, підтримка багатоядерних процесорів, технологія обміну даними NFC, слот для карт пам'яті та інше. Користувачі отримали можливість змінювати розмір «живих плиток» стартового екрана (тепер їх три види: маленькі, середні і великі).

Windows Phone 8.1:

Компанія Microsoft 2 квітня 2014 року на конференції BUILD показала версію системи під номером 8.1. Серед нововведень оновленої системи: центр повідомлень, голосовий помічник під ім'ям Кортана, нова клавіатура, підтримка корпоративних VPN.

Магазин застосунків:

Для пристроїв на Windows Phone передбачений інтернет-магазин програм та ігор Windows Phone Marketplace, доступний в 60 країнах. Купівля або установка цих програм можлива через розділ Marketplace на телефоні або через браузер. В кінці червня 2012 року Microsoft офіційно заявила, що в Windows Phone Marketplace кількість застосунків перевалила за 100 тисяч.

Для покупки музики, відео, підкастів, телевізійних передач існує інтернет-магазин Zune Marketplace, який доступний на телефоні через розділ «Музика + відео» або через спеціальну програму Zune Software, що встановлено на комп'ютері з операційною системою Windows. Zune Software – це повноцінний програвач, каталогізатор, який синхронізує пристрій і персональний комп'ютер, а також використовується для оновлення операційної системи смартфонів. Для Mac OS X компанія Microsoft випустила аналогічний застосунок – Windows Phone 7 Connector, який, крім іншого, дозволяє синхронізувати Windows Phone з частиною бібліотеки iTunes (фільми, музику, аудіо і відео підкасти) та iPhoto.

Встановлення програм та ігор на Windows Phone можливе тільки з офіційного інтернет-магазину Windows Phone Marketplace. Для ентузіастів можливе розблокування телефону за допомогою затвердженого Microsoft сервісу ChevronWP Labs, в результаті чого на

смартфон можна встановлювати саморобні програми або використовувати його для тестування в обхід Marketplace та офіційного розблокування телефону в якості розробника.

У вересні 2012 року Microsoft офіційно перейменувала Windows Phone Marketplace в Windows Phone Store, оновивши і поліпшивши пошук застосунків. З 30 листопада, після чергового оновлення Магазину Windows Phone, для пристроїв на платформі Windows Phone 8 з'явилася можливість встановлювати програми та ігри з SD-карти вручну.

Для розробки застосунків і ігор використовується Silverlight або XNA. Microsoft випустила інструментарій розробника Windows Phone SDK, для якого необхідні Visual Studio 2010 Express for Windows Phone, Expression Blend 4 for Windows Phone та XNA Game Studio 4.0. Формат файлів установки змінився з .cab, використовуваний в Windows Mobile, на новий.

В кінці 2012 року Microsoft офіційно повідомила про доступність більше 150 тисяч застосунків в Windows Phone Store (рис. 2.6) і про функціонування магазину в 191 країні. Влітку 2013 року кількість застосунків в магазині досягла 165 тисяч. У серпні цього ж року кількість завантажень в Windows Phone Store досягла 2 мільярди.

Рисунок 2.6 – Windows Phone Store

Windows Phone SDK.

Розробка під Windows Phone 7:

Всі програми для Windows Phone 7 створюються з використанням керованого коду. NET. В даний час C# – єдина підтримувана мова програмування, вільно доступна для завантаження Microsoft Visual Studio 2010 Express для Windows Phone включає XNA Game Studio 4.0 та екранний емулятор телефону, а також інтегрується з Visual Studio 2010.

Візуальні елементи і анімація для застосунків Silverlight можуть створюватися у Microsoft Expression Blend. Платформи Silverlight і XNA для Windows Phone 7 мають ряд спільних бібліотек, тобто деякі бібліотеки XNA можуть використовуватися в програмі Silverlight і

навпаки. Але не можна створювати програму, що поєднує в собі візуальні елементи обох платформ.

Як правило, Silverlight використовується для програм, які можна класифікувати як застосунки або утиліти. Опис компоновки елементів управління і панелей користувальницького інтерфейсу в цих програмах виконується за допомогою розширюваної мови розмітки застосунків (Extensible Application Markup Language, XAML). У файлах виділеного коду можуть реалізовуватися операції з ініціалізації і деяка логіка, але основним їх призначенням є обробка подій елементів управління. Silverlight дозволяє реалізовувати в Windows Phone стиль насичених інтернет-застосунків (Rich Internet Applications, RIA), включаючи мультимедіа та Web. Для Windows Phone створена версія Silverlight 3, в яку не ввійшли деякі можливості, що не підходять для телефону, але компенсовані низкою доповнень [8].

Головне призначення XNA – створення високопродуктивних ігор. Для 2D-ігор спрайт описуються за допомогою растрових зображень; для 3D ігор створюються тривимірні моделі. Дія гри, що включає переміщення графічних об'єктів по екрану і запит користувацького вводу, оброблюється вбудованим ігровим циклом XNA.

Зручно провести межі і прийняти, що Silverlight використовується для застосунків, а XNA – для ігор, але це не повинно накладати обмеження. Поза всяких сумнівів, Silverlight може застосовуватися для реалізації ігор, і традиційні застосунки можуть створюватися на XNA, хоча це буде пов'язане зі значними труднощами.

Silverlight підходить для ігор з невеликими вимогами до графіки, чи тих, що використовують векторну, а не растрову графіку, або для ігор, темп яких визначається реакцією користувача, а не таймером.

3 Розробка мобільного мультимедіа застосунку на базі платформи Android

У застосунках під Android використовуються бази даних SQLite, які являють собою один з п'яти способів зберігання даних в Android. Ми будемо розглядати тільки бази даних SQLite, оскільки це і є основа побудови робочої і функціональної програми. Після освоєння цього посту ви зможете вставляти дані з таблиць в бази даних, і проводити їх відбір.

Почнемо зі створення нового проекту (рис. 3.1) під назвою PROSTO-FM. Дані, які необхідно внести в форму створення нового проекту наступні:

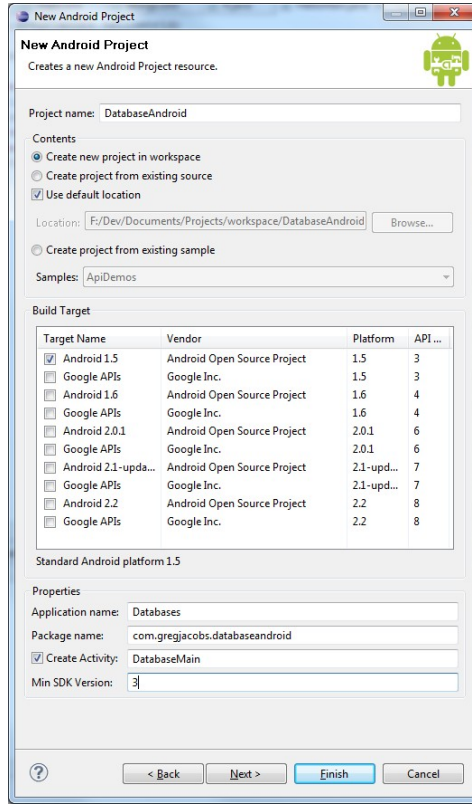

Рисунок 3.1 – Створення нового проекту

Назва проекту: TEST
Мета збірки: Android 1.5
Ім’я програми: TEST
Ім’я пакету: com.gregjacobs.randomquotes
Створити активність: QuotesMain
Мін SDK Версія: 3

Після введення даних і натискання на кнопку «Готово» приступимо до створення файлу класу в нашому пакеті під назвою com.gregjacobs.randomquotes. Для цього потрібно натиснути правою кнопкою миші на пакеті, вибрати в меню «Новий» (рис. 3.2), потім «Клас». У вікні потрібно заповнити лише поле «Ім’я», ввівши в нього DBAdapter. Далі тиснемо «Готово» і отримуємо базовий файл для класу, який нам належить трохи видозмінити. У цьому розділі спочатку

наведемо код, а потім пояснимо основні його частини та призначення головних функцій. В додаток цього разу буде приведено текстові файли, щоб ви могли завантажити їх і порівняти зі своїм варіантом. Почнемо з файлу DBAdapter.java:

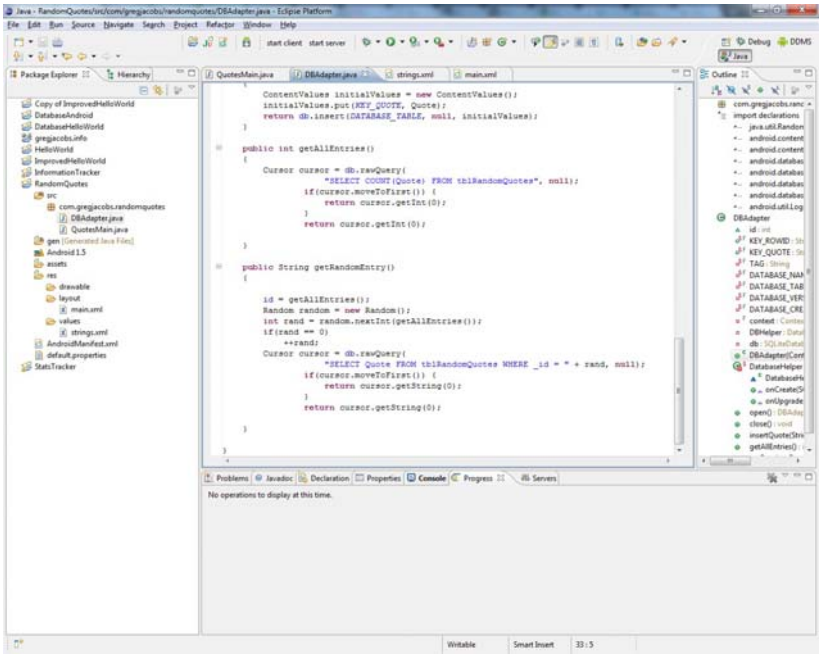

Рисунок 3.2 – Вигляд головного вікна програми

Почнемо з імпорту всіх інструментів, які знадобляться для створення і функціонування нашої бази даних SQ Lite (лістинг 3.1).

Можливо, професійним програмістам не знадобляться подальші пояснення, що означає кожен рядок, але для початківців вони будуть корисними. Отже, ContentValues дає можливість зберігати набір значень для оператору вставки, і дає доступ до середовища застосунку. Курсор, напевно, найпотрібніший імпорт з усіх. Курсор дозволяє отримати доступ до даних, отриманих з БД. SQL Exception дозволяє ініціювати SQL виключення при появі помилки. Ці повідомлення дозволяють зрозуміти, чим конкретно викликана помилка. SQLiteDatabase дає можливість керувати базою даних SQLite, використовуючи методи. SQLiteOpenHelper являє собою клас-помічник в управлінні БД.

«public class DBAdapter

```
{
int id = 0;
public static final String KEY_ROWID = «_id»;
public static final String KEY_QUOTE = «Quote»;
private static final String TAG = «DBAdapter»;
private static final String DATABASE_NAME = «Random»;
private static final String DATABASE_TABLE = «tblRandomQuotes»;
private static final int DATABASE_VERSION = 1;
private static final String DATABASE_CREATE =
«create table tblRandomQuotes (_id integer primary key autoincrement,
«
+ «Quote text not null );»;
private final Context context;
private DatabaseHelper DBHelper;
private SQLiteDatabase db; »
```

Лістинг 3.1 – Код програми для визначення використовуваних змінних

Тут ми визначаємо всі потрібні нам змінні, починаючи з назви бази даних і закінчуючи запитом створення самої бази даних. Ми використовуємо остаточні змінні так як їх значення не будуть змінюватися в ході роботи програми, але при цьому, виносимо змінні на зразок імен таблиці в окремі змінні, щоб полегшити модифікацію програми надалі.

```
«public DBAdapter(Context ctx)
{
this.context = ctx;
DBHelper = new DatabaseHelper(context);
}
vate static class DatabaseHelper extends SQLiteOpenHelper
{
DatabaseHelper(Context context)
{
super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db)
{
db.execSQL(DATABASE_CREATE);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion)
```

```
{
Log.w(TAG, «Upgrading database from version «+ oldVersion
+ «to «
+ newVersion + «, which will destroy all old data»);
db.execSQL(«DROP TABLE IF EXISTS tblRandomQuotes»);
onCreate(db); }
} »
```

Лістинг 3.2 – Код програми для визначення конструкторів

У вищевказаному фрагменті коду (лістинг 3.2) ми визначаємо конструктор, який буде передавати контекст пропозиції нашому помічникові, DatabaseHelper. Клас DatabaseHelper розширює можливості нашого SQLiteOpenHelper, який покращує функціонал управління базою даних SQLite. Головна функція OnCreate, яку нам ще належить використовувати далі, дозволяє виконати SQL-запит по створенню бази даних.

```
«public DBAdapter open() throws SQLException
{
db = DBHelper.getWritableDatabase();
return this;
}
public void close()
{
DBHelper.close();
}»
```

Лістинг 3.3 – Функції відкриття та закриття бази даних

У коді вище (лістинг 3.3) є дві ключові функції відкриття і закриття бази даних. На ці функції можна зсилатися при виклику їх у нашому головному .java файлі.

```
«public long insertQuote(String Quote)
{
ContentValues initialValues = new ContentValues();
initialValues.put(KEY_QUOTE, Quote);
return db.insert(DATABASE_TABLE, null, initialValues);
}»
```

Лістинг 3.4 – Функція для додавання цитат до бази даних

Вище описана функція (лістинг 3.4) обробляє наші цитати, коли ми викликаємо їх в головному .java файлі. Також ця функція готує цитати для введення в БД, розміщуючи рядок «Цитата» в ContentValues під назвою initialValues, які потім вставляються в таблицю БД.

```
«public int getAllEntries()
{
Cursor cursor = db.rawQuery(
«SELECT COUNT(Quote) FROM tblRandomQuotes», null);
if(cursor.moveToFirst()) {
return cursor.getInt(0);
}
return cursor.getInt(0);
}»
```

Лістинг 3.5 – Функція запиту

Ця функція (лістинг 3.5) буде виконувати запит на кількість введених в БД цитат для передачі цього значення як максимально можливого генератору випадкових чисел. Таким чином вдасться уникнути помилки (обраний номер цитати ніколи не буде перевищувати максимально можливий). Ми в основному використовуємо в роботі тип запиту rawQuery, бо багатьом не дуже подобається, як Android обробляє свої запити, але вражає можливістю використання повнофункціональних SQL-запитів. Умова «якщо» змусить покажчик прийняти перший результат (якщо буде знайдено кілька результатів). Коли умова «якщо» не дійсна, в якості результату візьметься значення першої позиції (лістинг 3.6).

```
«<?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/Quote»
/>
<EditText
android:id=«@+id/Quote»
android:layout_width=«fill_parent»
android:layout_height=«wrap_content»
/>
<Button
android:id=«@+id/go»
android:layout_width=«fill_parent»
```

```
android:layout_height=«wrap_content»
android:text=«@string/press»
/>
<Button
android:id=«@+id/genRan»
android:layout_width=«fill_parent»
android:layout_height=«wrap_content»
android:text=«@string/genRan»
/>
</LinearLayout>»
```

Лістинг 3.6 – Приклад коду для візуального оформлення першої версії програми

3.1 Засоби розробки мобільних застосунків

У попередньому розділі була описана загальна архітектура платформи Java 2 ME, в цій главі вашій увазі буде представлено два інтегрованих середовища для розробки мобільних застосунків – це SUN ONE Studio 4 Mobile Edition TA J2ME Wireless Toolkit 2.1. Обидва середовища поширюється безкоштовно і розроблені в надрах компанії Sun Microsystems. Ці інструментарії, на відміну від інших середовищ програмування, адаптовані саме для створення застосунків для мобільних пристроїв.

Перш ніж приступити до інсталяції SUN ONE Studio 4 Mobile Edition і J2ME Wireless Toolkit 2.1, необхідно встановити набір інструментальних засобів Java 2 SDK, SE vl.4.2_03 (Software Development Toolkit Standard Edition). Це остання на даний момент версія бібліотеки, що надає, програмні засоби для створення застосунків на Java. До складу цієї бібліотеки також входить Java 2 Runtime Environment, SE vl.4.2_03, яка необхідна для роботи застосунків написаних на Java .

3.1.1 Установка Java 2 SDK SE

Подвійний клік лівої кнопки миші на файлі j2sdk-l_4_2_03-windows-i586-p з каталогу \SDK, що знаходиться на компакт-диску, приведе до розпаковування архіву. Далі з’явиться вікно з ліцензійною угодою, після чого, натиснувши на кнопку «Next», ви попадете у вікно Java 2 SDK SE vl.4.2_03-Custom Setup.

У цьому діалоговому вікні вам буде запропоновано вибрати каталог і тип установки. Нам потрібні всі компоненти, тому натисніть кнопку «Next». У наступному вікні вам необхідно вибрати браузер для інтеграції Java. Зупиніть свій вибір на необхідному і підтвердіть свої дії,

після чого почнеться процес установки, в кінці якого вам запропонують перезавантажити комп'ютер. Після установки Java 2 SDK можна приступати до інсталяції SUN ONE Studio 4 Mobile Edition і J2ME Wireless Toolkit.

Відразу хотілося б зупинитися на деяких особливостях, що виникають при інсталяції. Обидва середовища програмування незалежні один від одного, але в комплект SUN ONE Studio 4 Mobile Edition входить J2ME Wireless Toolkit версії 1.0.4 01, хоча на даний момент вже є J2ME Wireless Toolkit версії 2.1. Така плутанина обумовлена саме тим, що обидва продукти незалежні один від одного і розвиваються паралельно [9]. Розглянемо установку потужнішого середовища SUN ONE Studio 4 Mobile Edition .

3.1.2 Середовище програмування SUN ONE Studio 4 Mobile Edition

На компакт-диску в директорії \SDK знаходиться файл ffj_nic_win32 – це програма установки інтегрованого середовища розробки SUN ONE Studio 4 Mobile Edition. Після подвійного натиснення лівої кнопки миші на цьому файлі, ви побачите вікно з вітанням, а натиснувши на кнопку «Next», ви попадете у віконце з ліцензійною угодою. Ознайомившись з угодою і натиснувши на кнопку «Next», ви побачите діалогове вікно SUN ONE Studio 4 update 1, Mobile Edition Setup.

У цьому і наступному вікнах необхідно визначитися з директорією, в якій розміщуватиметься встановлюваний інструментарій, після чого з'явиться діалогове вікно, де перераховуються компоненти установки. Шлях, що задає директорію для установки SUN ONE Studio 4 Mobile Edition, не повинен містити пробілів. Вікно вибору компонентів містить елементи для установки. Інсталюється не лише J2ME Wireless Toolkit, але і Java 2 Platform Micro Edition, яка також може поставлятися окремо і лежить на компакт-диску в каталозі \ SDK (в разі, якщо ви вважаєте за краще працювати з командним рядком).

Два подальших вікна носять більше інформаційний характер, просто натискайте кнопку «Next» і майстер установки почне свою роботу з інсталяції середовища програмування SUN ONE Studio .4 Mobile Edition на комп'ютер.

3.1.3 Створення проекту в SUN ONE Studio 4 Mobile Edition

Для створення проекту в середовищі SUN ONE Studio 4 Mobile Edition необхідно обрати в меню команду Project => Project Manager. У вікні менеджера проектів, що з'явилося, натисніть кнопку «New».

Замість вікна менеджера проектів з'явиться невелике діалогове вікно Create New Project, де в полі Project Name необхідно задати ім'я

проекту, наприклад Demo, і натиснути на кнопку «ОК». Після цих дій на екрані з'явиться вікно конфігурації створеного проекту.

У вікні конфігуратора проектів є два пункти: Java 2 Standard Edition (J2SE)H Mobile Information Device Profile (CLDC\MIDP), виконані у виді списку обираних елементів. Вибравши один з двох пунктів можна конфігурувати проект для створення стандартної програми на Java 2 SE, або застосунок для мобільних телефонів, якщо вибрати пункт Mobile Information Device Profile (CLDC\MIDP). Обираємо пункт для створення мобільних застосунків і натискаємо кнопку «Finish». Після цього у нижній частині вікна «Explorer» до вкладок «Файлові системи», «Виконання» і «Java-doc», додасться ще одна вкладка «Project Demo». Після цього необхідно виконати монтування файлової системи для конфігурованого проекту. На вкладку системи закладок що з'явиться при натисканні правою кнопкою миші на написі «Файлова система». У відповідь на ці дії з'явиться низка випадаючих меню; оберіть команду Mount => Archive (JAR, Zip). Обравши команду Archive (JAR, Zip), перейдіть до вікна New Wizard-Archive (JAR, Zip), за допомогою якого виконується монтаж необхідної для проекту бібліотеки Java API. Всі бібліотеки скомпоновані у вигляді архівів, наприклад бібліотека для профілю MIDP 1.0 міститься в архіві midpapi.zip. Ситуація тут декілька заплутана. Як вже говорилося до складу інструментарію SUN ONE Studio 4 Mobile Edition входить також середовище J2ME Wireless Toolkit версії 1.0.4 01, що містить архів API для профілю MIDP 1.0.

Під час установки пізнішої версії цього інтегрованого середовища ви зможете підключати API і для профілю MIDP 2.0, обравши відповідний файл з директорії, в яку було встановлено середовище J2ME Wireless Toolkit 2.1 пізнішої версії. У наступній главі буде розглянуто установку і налаштування різних телефонних емуляторів і програмного забезпечення для програмування телефонів всіляких марок.

Після монтування файлової системи, необхідно також виконати монтаж директорії, де буде міститися початковий код створеного проекту. Монтаж директорії відбувається за тією ж схемою, що і монтаж файлової системи, але у випадаючому меню потрібно обрати команду Mount Local Directory і у вікні New Wizard, що з'явилося, вказати місце, де зберігатиметься вихідний код програми, після чого натискайте кнопку «Finish». Створивши робочий каталог для застосунку, його необхідно додати в проект. Натисніть правою кнопкою миші на щойно доданій директорії у вкладці «Файлові системи» вікна «Explorer» і в меню, що з'явилося, виберіть команду Tools => Add to Project.

3.1.4 Створення застосунків в SUN ONE Studio 4 Mobile Edition

Завершивши створення проекту, можна переходити до написання коду застосунку. Для цього перейдіть на вкладку створеного проекту Demo, у вікні Explorer і виберіть команду «New File» у меню «File», або скористайтеся гарячими клавішами Ctrl+N. На екрані з'явиться вікно New Wizard, де відбувається вибір шаблону створюваного застосунку.

У ролі демонстраційної програми оберемо шаблон HelloMIDlet, який формує код простої програми, що має назву мідлет і що позиціюється в Java 2 ME як застосунок, написаний спеціально для мобільного пристрою. Натиснувши на кнопку «Next», ви попадете в наступне вікно New Wizard – HelloMIDlet, де необхідно вказати майбутню директорію знаходження мідлета і задати ім'я пакету, або скористатися стандартним пакетом, що надається за замовчуванням. Вказавши пакет і обравши директорію, натисніть на кнопку «Finish». Після цього автоматично згенерується простий код прикладу HelloMIDlet проекту Demo, який з'явиться в текстовому редакторі Source Editor.

3.1.5 Компіляція і запуск програми в SUN ONE Studio 4 Mobile Edition

У ролі демонстраційного прикладу ми використаємо проект Demo, створений автоматично SUN ONE Studio 4 Mobile Edition раніше. Знайдіть вихідний код HelloMIDlet проекту Demo в директорії, яку ви задали для розміщення програм в SUN ONE Studio 4 Mobile Edition або візьміть код HelloMIDlet з компакт-диску в теці \Code\ HelloMIDlet, помістіть його в директорію C:\WTK21\apps\Demo\src. Потім натисніть кнопку Build на панелі інструментів для компіляції і компоновки всього проекту. У робоче вікно SUN ONE Studio 4 Mobile Edition додадуться дві строчки.

Тут ми, звичайно, унеможливлюємо появу помилок при компіляції, але в реальному програмуванні застосунків без цього не обійтися: програміст де-небудь та забуде поставити крапку з комою. Після компіляції і компоновки проекту натисніть кнопку Run, і на екрані з'явиться емулятор за замовчуванням DefaurtColorPlione, SUN ONE Studio 4 Mobile Edition.

На екран емулятора буде виведено ім'я проекту. Натиснувши на клавішу емулятора «Select» або «Launch», ви попадете в робочий цикл програми, і на екрані емулятора з'явиться напис «Test string». Натиснувши на клавішу «Exit» можна вийти з застосунку.

3.1.6 Упаковка програм

При створенні проекту і подальшої компіляції вихідного коду, у вас з'являться два файли: JAD-файл і файл маніфесту. Якщо ви працюєте з

SUN ONE Studio 4 Mobile Edition, то обидва файли знаходитимуться в робочому каталозі проекту.

3.1.7 Файл маніфесту

Файл маніфесту MANIFEST.MF описує можливі атрибути створюваного застосунку. При створенні проекту з J2ME Wireless Toolkit в попередніх пунктах згадувалося про діалогове вікно Settings для проекту Demo, розділене на сім вкладок, в кожній з яких вказуються різні атрибути створюваного застосунку. На основі заданих атрибутів в діалоговому вікні Settings для проекту «Demo» і відбувається генерація файлу маніфесту, а також JAD-файлу.

3.1.8 Файл JAD

Як уже згадувалося, в робочому каталозі проекту Demo» знаходитиметься ще один файл Demo.jad. JAD-файл в мобільних застосунках ще називають дескриптором застосунку (Java Application Descriptor). Цей файл використовується телефоном під час роботи програми для здобуття інформації про наявні класи, зображення, піктограми і звукові файли всієї програми. На основі отриманої інформації відбувається управління внутрішніми ресурсами застосунку. Якщо ви переміститеся у робочий каталог проекту Demo і знайдете згенерований файл Demo.jad, то побачите іконку у вигляді телефону з лівого боку від назви файлу. Зробивши подвійний клік лівою кнопкою миші на файлі Demo.jad, ви запустите емулятор телефону незалежно від того, чи відкрите у даний момент одне з середовищ програмування чи ні. Це ще раз вказує на те, що JAD-файл використовується для управління роботою програми. Але є і ще одна цікава особливість JAD-файлу. JAD-файл теж містить опис атрибутів застосунку і вони багато в чому дублюються у файлі маніфесту MANIFEST.mf. У програмах Java 2 ME, JAD-файл – це дескриптор застосунку і використовується для управління роботою програми. Сервіс телефону перед запуском програми звертається саме до JAD-файлу, визначаючи тим самим можливість роботи цієї програми на цьому телефоні. Якщо один з параметрів буде недопустимим для цієї моделі телефону, то застосунок не буде запущений.

3.1.9 Файл JAR

У мові Java існує можливість архівації файлів застосунку в один файл з розширенням *.jar. Файл JAR – це архів, що містить супутні класи і графічні зображення всього застосунку. При написанні програм на Java під різні комп'ютерні операційні системи, програміст вільний сам обирати, чи буде він поширювати свій застосунок у заархівованому

вигляді, або в оригінальному. Ситуація з поширенням та перенесенням програм під мобільні телефони радикально протилежна. Телефони обмежені в своїх ресурсах і в середньому пам'ять, відведена під Java програму, вагається від 30 до 80 кілобайт. Ці цифри обов'язково треба враховувати при створенні мобільних застосунків. Розмір в 40–50 кілобайт вважається оптимальним. У зв'язку з цим всі мобільні програми зобов'язані поширюватися у заархівованому вигляді, тобто в JAR-файлі [10].

3.2 Телефонні емулятори

Інтегровані засоби розробки застосунків для мобільних телефонів розглянуті в попередній главі мають в своєму складі декілька емуляторів мобільних пристроїв неіснуючих марок телефонів. Емулятор телефону – це застосунок, що програмно емулює роботу реального телефону, враховуючи всі технічні характеристики цього пристрою. Більшість виробників мобільних телефонів мають в своєму арсеналі набори інструментальних засобів для програмування телефонів, так званих SDK (Software Developer Kit) і безліч вбудованих модулів для емуляції певних моделей. Найбільша кількість програмного забезпечення надається компанією Nokia.

Лінійка всіляких моделей телефонів, що випускаються одним виробником, зазвичай ділиться по цінових категоріях. Чим дорожче модель – тим більше функціональних можливостей включено в цей пристрій, а значить, передбачена потужніша архітектура телефону. Як правило, всі відмінності зводяться до декількох параметрів:

– наявність або відсутність операційної системи в телефоні;

– можливість роботи з застосунками написаними на Java 2 ME і C++;

– кількість пам'яті;

– роздільно здатність дисплея телефону.

У зв'язку з цим дуже корисно мати якомога більше програмних засобів, що емулюють реальні мобільні пристрої, для того, щоб можна було протестувати створювану програму. Якщо програмувати застосунок на Java 2 ME, використовуючи профіль MIDP 20 або MIDP 1.0, то на всіх наявних моделях телефонів, де передбачена підтримка Java з відповідним профілем, ваш застосунок працюватиме. Але, на жаль, можуть виникнути проблеми в графічному відображенні елементів призначеного для користувача інтерфейсу, унаслідок того, що присутній ряд очевидних відмінностей. У візуальному представленні цих елементів. Також кожен з виробників телефонів дотримується своєї системи навігації, що, безумовно, ускладнює розробку програмного

забезпечення. Тому наявні набори SDK і емуляторів окремо взятих моделей телефонів, набагато полегшують роботу програміста.

У своїй більшості програмне забезпечення різних виробників інтегрується у візуальні середовища від компанії Sun Microsystems і Borland, або існують у вигляді окремих програм. Далі буде детальніше вивчення специфіки створення застосунків за допомогою класів Java 2 ME [11].

3.3 Механізм роботи застосунків Java 2 ME

З цього розділу починається безпосередньо опис роботи з кодом. Специфіка Java 2 ME застосунків декілька своєрідна, але зовсім не складна. Досить розібратися в загальній моделі побудови програм і все відразу стане зрозуміло.

3.3.1 Мідлет

Застосунок, написаний для мобільного телефону, може складатися з різної кількості класів. Одні класи, відповідають за завантаження ресурсів, інші за обробку даних, треті виконують ще якісь додаткові функції, як програміст ви маєте право вибирати будь-яку зручну для вас модель побудови програми. У результаті, створені вами класи, об'єднані в одне ціле, складатимуть одну програму або застосунок. Всі застосунки, сформовані для роботи в середовищі Java мобільних телефонів, носять назву мідлет. Мідлет – це програма, написана для мобільного телефону з використанням платформи Java 2 ME. Визначати кількість класів програми привілей програміста, але серед всіх класів однієї програми існує один основний клас, з якого починається робота всієї програми. У цьому класі описується код, що відповідає за управління процесом створення інтерфейсу користувача, оголошення набору даних необхідних для роботи всього застосунку, створюються об'єкти наявних класів, і що найголовніше, він є відправною крапкою в роботі застосунка. Такий клас в Java 2 ME носить назву основний клас мідлета.

Робочі функції, що виконуються цим класом, практично ідентичні методу main (). Пам'ятаєте запис, з яким починався робочий процес застосунків написаних на Java 2 SE:

public static void main (String () args)

На мобільних пристроях аналогічні дії покладені на підклас класу MIDI et, що виробляє управління робочим процесом всього застосунку. В доповнення до основного класу, може створюватися ряд класових обмежень (лістинг 3.7) для реалізації поставленого перед вами завдання.

```
/**
Клас HelloMIDlet.java
*/
```

```
// імпорт пакетів
import javax.microйdition.midlet.*; import j avax.microйdition.lcdui.* ;
// створюється клас HelloMIDlet
public class HelloMIDlet extends MIDlet implements CommandListener {
// команда виходу з програми private Command exitCommand;
// дисплей телефону private Display mydisplay;
// конструктор класу HelloMIDlet public HelloMIDlet()
{
mydisplay = Display.getDisplay(this);
// вихід з програми
exitCommand = new Command(«Вихід», Command.SCREEN, 2);
}
// вхідна точка застосунку public void startAppO {
// об'єкт класу TextBox
TextBox t = new TextBox(«HelloMIDlet»,»Текст»,256, 0);
// додає команду виходу t.addCommand(exitCommand);
// вуст анавливает обробник подій для команди виходу t.setCommandListener(this); // відображає поточний екран телефону mydisplay.setCurrent(t);
>
// пауза в роботі застосунку public void pauseApp() {// вихід з програми
public void destroyApp (boolean unconditional) {// обробник подій для команд застосунку public void Commandaction (Command з, Displayable s) {
// обробка команди вихід if. (з == exitCommand) {
// звільняє ресурси і вивантажує з пам'яті мидлет
destroyApp(false); notifyDestroyed();
}
} }
```

Лістинг 3.7 – Код програми

Перше, що впадає в очі при детальному розгляді коду мідлету – це три методи: startApp і, pauseApp t і destroyApp 1. Програмісти, які знайомі з аплетами, знайдуть явну схожість в структурі мідлета і аплета, навіть назви в деякій мірі схожі. У аплеті також є подібні методи: start(), stop() і destroy(). Можна сказати, що основний клас мідлета – це деякий симбіоз аплета і методу main(), що грає основну керуючу роль, в застосунку для мобільних пристроїв. Створюючи свої класи і реалізовуючи їх код, ви повинні покласти основні робочі функції на основний клас мідлета.

3.3.2 Інтерфейс користувача

Коли ми розглядали механізм роботи прикладу з попереднього лістингу, я думаю, ви помітили деякий поекранний принцип відображення інформації на дисплеї. Перший екран показував список доступних застосунків, після вибору одного з них ви потрапляли на екран цього застосунку. Натискує кнопку Вихід, відбувалося повернення до екрану вибору. У Java 2 ME програмах така схема поекранного відображення інформації є основною. Якщо ви ніколи не звертали уваги на це при роботі зі своїм телефоном, то саме час узяти його і перевірити джойстиком. У застосунках для мобільних телефонів, заснованих на екранах в Java 2 ME, відсутні вікна і фрейми на відміну від Java 2 SE. Телефони обмежені в системних ресурсах і всіляких прикрасах, вони не дозволені для цих маленьких пристроїв. Тому при створенні кінцевого продукту варто з особливою ретельністю продумувати основні складові майбутнього застосунку. Ґрунтуючись на поекранному відображенні інформації, необхідно створювати інтуїтивно зрозумілу структуру застосунку, утворюючи при цьому чітку екранну навігацію. Якщо користувач заблукає у вашій програмі, він просто видалить її з пам'яті телефону і ніколи до неї більше не повернеться, а ви втратите потенційного покупця.

Як вже наголошувалося, екран телефону представлений класом Display. Кожен мідлет може мати лише один об'єкт класу Display, що повертає мідлету за допомогою методу getDisplay(), визначаючи тим самим поточний дисплей телефону для мідлета.

Платформа Java 2 ME володіє пакетом javax.microedition.lcdui, що включає класи для роботи з інтерфейсом користувача UI (user interface). Велика кількість класів, що входять в цей пакет, детально розглядатимуться в наступній главі. Найголовнішим класом інтерфейса користувача є клас Displayable. З основи абстрактного класу Displayable відбувається побудова основної частини графічного інтерфейсу застосунку.

Від класу диспетчера Display залежить, який з класів Displayable відображуватиметься на екрані. У свою чергу лише один клас Displayable може бути одноразово показаний на екрані. Тобто, об'єкт класу TextBox, грубо кажучи, існує в своєму екрані, об'єкт класу List – в своєму і обидва об'єкти не можуть існувати разом на одному екрані, визначаючи тим самим правило поекранного віддзеркалення інформації на дисплеї телефону.

Далі в ієрархії структури інтерфейсу користувача йдуть два абстрактні класи: Screen та Canvas. На цій стадії відбувається розділення класів інтерфейсу користувача на високорівневий клас, призначений класу Screen і всій його подальшій ієрархії спадкоємства і

низькорівневий клас Canvas. Обидва класи розділені на високорівневий і низькорівневий інтерфейси користувача.

Високорівневий інтерфейс містить засоби для роботи з інтерфейсом користувача, створені на основі класів шаблонів, використання яких призводить до побудови жорстко заданого інтерфейсу. Наприклад, задіяний у вихідному коді HelloMIDlet проекту Demo клас TextBox не може жодним чином змінити екран телефону. Екран, представлений класом TextBox – це текстовий контейнер, в якому можна здійснювати вивід, видалення і редакцію тексту і не більш того. Тобто класи високорівневого інтерфейсу – це жорстко задана модель відображення інтерфейсу користувача на екрані телефону, за допомогою яких програміст організовує навігацію, списки, меню, текстові контейнери, групи вибираних елементів і так далі.

Низькорівневий інтерфейс надає графічні засоби для малювання на екрані різних графічних елементів і обробку команд, що посилаються з клавіш телефону. Низькорівневий інтерфейс дозволяє малювати на екрані телефону, додаючи тим самим в застосунок красиві графічні елементи у вигляді таблиць, зображень, полів, заставок та інше. Таке розділення класів існує суто в теоретичному вигляді і ніщо вам не заважає комбінувати елементи обох інтерфейсів в одній програмі, створюючи красивий інтерактивний застосунок.

Розглянемо деякі характеристики класів Alert, TextBox, Form та List, що представляють високорівневий інтерфейс.

Alert – надає можливість в створенні повідомлень про помилки у вигляді окремих екранів; TextBox – масив даних, містить текстову інформацію з можливістю її редакції; Form – екранний контейнер, в якому можна розмістити різні елементи інтерфейсу за допомогою додаткових підкласів класу Item; List – список елементів, що дозволяє організувати вибір різних операцій.

Далі вивчатимуться класи високорівневого інтерфейсу, за допомогою яких створюються списки, групи елементів, текстові поля і безліч інших елементів інтерфейсу користувача.

3.4 Класи інтерфейсу користувача

У Java 2 ME є пакет javax.microedition.lcdui, визначений для класів призначеного інтерфейсу користувача. Як вже наголошувалося раніше, класи призначеного для інтерфейсу користувача розділені на високорівневий і низькорівневий інтерфейси. У цьому розділі будуть послідовно розглянуті всі класи високорівневого інтерфейсу користувача. Кожен з розділів містить інформацію про один конкретний клас, що надає ряд можливостей в оформленні інтерфейсу користувача.

Використовуючи можливості цих класів, ви зможете створювати в застосунку списки, групи елементів, завантажувати в програму зображення, використовувати рядок, що біжить, призначати шрифт тексту і багато що інше. У розділах по кожному класу призначеного для інтерфейсу користувача, аналізуються конструктори, основні методи і константи класу, за допомогою яких в кінці кожного розділу створюється застосунок, що ілюструє роботу цього класу, і наводиться лістинг всієї програми. При розгляді методів і констант класів використовується лише основні компоненти. Для детального аналізу складових одного з класів зверніться до застосунку 2, який виконаний у вигляді довідника по Java 2 ME.

Єдине, про що необхідно пам'ятати при проектуванні і створенні застосунку інтерфейсу користувача – це про використовуваний профіль MIDP. У деякі класи, що є у складі профілю MIDP 1.0, були додані нові методи, константи і навіть класи з профілю MIDP 2.O. Наприклад, в класі ChoiceGroup для профілю MIDP 1.0 доступні тринадцять методів, а вже в профілі MIDP 2.0 їх налічується сімнадцять, тобто, додано ще чотири нових. У тій же документації по Java 2 ME якоїсь чіткої грані що розділяє два профілі не існує, але при розгляді методів і констант згадується про приналежність до одного з профілів. Тому при створенні застосунків, наприклад під профіль, MIDP 1.0, необхідно уважно планувати розробку програм і не задіюючи компоненти профілю MIDP 2.0. Якщо ви розробляєте програму для профілю MIDP 2.0, то можете користуватися всіма наявними компонентами незалежно від приналежності до одного з профілів.

3.4.1 Клас Form

Основним екранним класом прикладів з попередніх розділів служив екран, представлений класом Form. Як ви розумієте цю не обов'язкову умову, але я вибір класу Form не випадковим. Річ у тому, що реалізація класу Form виконана у вигляді контейнера, що дозволяє вбудовувати в себе різні компоненти інтерфейсу користувача. Це, мабуть, єдиний і найпотужніший по своїх можливостях клас. Сама по собі назва Form має на увазі створення деякої форми для заповнення екрану телефону. У попередніх прикладах ми використовували, так звану, порожню форму цього класу у вигляді чистого екрану. Згодом, при згадці терміну форма, матиметься на увазі екран телефону, представлений об'єктом класу Form для інтеграції класів інтерфейсу користувача.

Клас Form має два конструктори, необхідних при створенні об'єкту цього класу. Першим конструктором ми вже користувалися, і виглядає він досить просто:

public Form (String title)

Параметри конструктора класу Form:

– title – заголовок з'являється у верхній частині створеного вікна.

Другий конструктор класу Form має вже два параметри, і дозволяє вбудувати компоненти інтерфейсу в порожню форму, public Form (String title, Item [] items).

Параметри конструктора класу Form:

– title – заголовок вікна;

– items – масив компонентів розміщуваних у класі Form.

У класі Form існує набір методів, за допомогою яких можна додати, видалити або вставити компоненти інтерфейсу. Клас Form має дванадцять методів, за допомогою яких можна маніпулювати компонентами абстрактного класу Item. Всього налічується вісім компонентів призначеного для користувача інтерфейсу в ієрархії класу Item. Тобто ви створюєте екран, за який відповідає клас Form, і інтегруєте наявні у вашому розпорядженні компоненти класу Item. Про сам клас Item і його ієрархії ми поговоримо в наступному розділі цієї глави, після аналізу класу Form.

3.4.2 Клас Item

Абстрактний суперклас Item має ієрархію з восьми підкласів. Кожен підклас представляє один з елементів інтерфейсу користувача, наприклад, клас TextField, створює текстові поля для введення пароля, адреси електронної пошти або просто числових значень. Всі вісім класів, по суті, встановлюють компоненти інтерфейсу користувача, які вбудовуються у форму визначену класом Form. На рис. 3.3 зображена ієрархія абстрактного суперкласу Item.

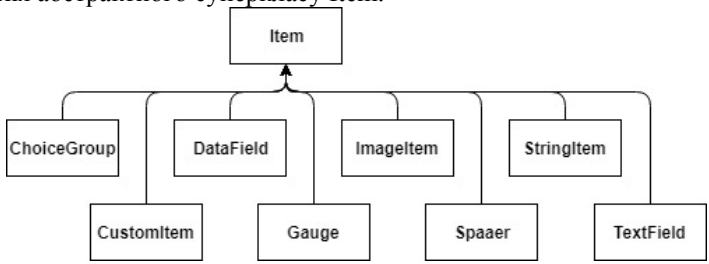

Рисунок 3.3 – Ієрархія суперкласу Item

Choice Group – це група зв'язаних елементів для подальшого вибору передбачуваних дій;

CustomItem – за допомогою цього класу можна додавати різні графічні елементи у форму;

Date Field – клас, за допомогою якого є можливість редагувати час і дату;

Gauge – допускає графічне відображення діаграм, процесів завантаження;

Imageltem – здійснює показ зображення на екрані телефону;

Spacer – задає визначений по розміру простір;

String Item – за допомогою цього класу можна створити довільний текст. Цей клас не допускає редагування, він лише відображує інформацію;

TextField – надає текстові поля для редакції. Любий із розглянутих класів успадковується з суперкласу Item і може бути доданий на екран, створений класом Form Кожен компонент класу Item містить з лівого боку область, де за бажання можна відображувати зображення у вигляді іконки. При переміщенні компонента, іконка також переміщається разом з компонентом. Клас Item за допомогою наявних в його складі директив задає, в основному, формат відображення для будь-якого компонента. Формат визначає задану ширину, висоту або вирівнювання компонентів у формі, а також клас Item має безліч методів що здійснюють контроль над компонентами.

3.4.3 Клас Alert

Використання класу Alert в Java 2 ME застосунках обумовлено виникненням різних позаштатних ситуацій. В основному класі Alert застосовується для створення екрану, який інформує користувача про помилку, що сталася в застосунку або будь-якому іншому повідомленні інформаційного характеру. Екран, визначений класом Alert може містити строкове повідомлення про помилку, що сталася, або текстовий рядок із заданим зображенням. У зв'язку з цим, клас Alert має два конструктора об'єктів цього класу, що використовуються в застосунку. Перший конструктор містить один параметр типу String, задаючи рядок тексту для повідомлення. Розглянемо перший конструктор класу Alert.

public Alert (String title);

Параметри конструктора public Alert :

– title – рядок тексту.

Другий конструктор класу Alert має вже чотири параметри, представляючи цікавіший вигляд створюваного екрану.

public Alert (String title, String alertText, image alertlmage, AlertType alertType)

Параметри конструктора public Alert :

– title – назва створеного екрану;

– alertText – текст повідомлення;

– alertlmage – зображення;

– alert/Type – тип повідомлення, визначуваний класом

AlertType. Існує п’ять типів повідомлень:

– static AlertType ALARM – тривога;

– static AlertType CONFIRMATION – попередження про можливу дію, яку користувач повинен виробити;

– static AlertType ERROR – помилка;

– static AlertType INFO – інформаційне повідомлення ;

– static AlertType WARNING – попередження.

Створюючи об’єкт класу Alert, ви можете вибрати необхідний тип повідомлень або інформаційних повідомлень, формуючи органічні, зручні застосунки, що передбачають будь-які варіанти розвитку подій.

3.5 Програмування графіки та техніка створення ігор

Високорівневі класи, розглянуті в попередньому розділі, дають можливість створювати застосунок інтерфейсу користувача. По суті, ці класи виконані у вигляді шаблонів, використовуючи які ви можете створювати списки, форми, шрифт, групи елементів, рядки, що біжать. Але використання таких класів-шаблонів трохи спрощує інтерфейс програми, позбавляючи можливості використання графіки в програмах на Java2ME. Інколи в застосунку необхідно намалювати таблицю, лінію, квадрат, тобто скористатися графікою для створення насиченої і приваблюваної програми. Для таких цілей в платформі Java2ME існують так звані класи низькорівневого інтерфейсу – це класи Canvas і GameCanvas (клас GameCanvas буде розглянутий в наступному розділі), а також клас Graphics, за допомогою якого здійснюється безпосереднє промальовування графіки на екрані телефону.

Ринок мобільних телефонів розвивається стрімкими темпами. Все більше телефонів мають підтримку технології Java. Віяння ігрової індустрії захопило і мобільні телефони, тому платформа Java 2 ME позиціює більшою мірою як ігрова платформа для мобільних телефонів. При розробці ігор під профіль MIDP 1.0 програміст стикається з масою проблем у вигляді написання великої кількості власних класів для створення ігрового процесу, малювання графіки, рівнів і так далі. У профіль MIDP 2.0 додано п’ять ігрових класів, мобільних ігор, що значно спрощують створення, це:,

– Game Canvas – абстрактний клас, що містить основні елементи ігрового інтерфейсу;

– Layer – абстрактний клас, що відповідає за рівні що представляються в грі;

– LayerManager – менеджер рівнів;

– Sprite – представляє на екрані спрайтове зображення;

– TiledLayer – відповідає за створення фонових зображень.

Всі вищеперелічені класи доступні лише в профілі MIDP 2.0. Також можна скористатися цими класами і в програмуванні телефонів Siemens під профіль MIDP 1.0, імпортувавши пакет com.siemens.mp.color_game. Компанія Siemens входить до експертної групи MIDP Expert Group і, на базі вже наявних ігрових класів від компанії Siemens були створені ігрові класи для профілю MIDP 2.0.

При використанні ігрових класів профілю MIDP 2.0, побудова мобільної гри заснована на системі рівнів. Формуючи гру, програміст створює необхідну йому кількість рівнів. Кожен з рівнів містить набір графічних елементів, наприклад, можна створити рівень з фоновим зображенням, рівень з перешкодами, рівень з ігровими бонусами, рівень з головним персонажем гри. Після цього все наявні рівні компонуються воєдино, накладаються один рівень на інший і промальовувалися на екрані телефону. Контроль над всіма рівнями, здійснюється за допомогою класу LayerManager – менеджера рівнів. Велика кількість методів, що надаються ігровими класами, дозволяє відстежувати всілякі зіткнення, переміщення, анімацію і безліч інших функцій, що значно спрощує процес створення мобільних ігор.

У цьому розділі, спочатку ви познайомитеся з устроєм всіх ігрових класів, а потім буде створений ряд прикладів, що ілюструє роботу і взаємодію ігрових класів профілю MIDP 2.O.

3.5.1 Клас Layer

Абстрактний клас Layer задає основні властивості для всіх створених рівнів гри.

Клас Layer має два підкласу TiledLayer і Sprite. При створенні будь-яких інших підкласів класу Layer необхідно реалізувати метод paint () у цих класах, аби мати можливість малювати створені рівні на екрані телефону, що представляється класом Graphics. За допомогою методів класу Layer можна задавати розмір і позицію рівня.

– int getHeight () – отримує висоту екрану;

– int getwidth() – отримує ширину екрану;

– int getx() – отримує горизонтальну координату рівня;

– int getY() – отримує вертикальну координату рівня;

– void move(int dx, int dy) – переміщує рівень на dx и dy;

– Q abstract void paint(Graphics g) – малює рівень;

– void setPosition (int x, int в) – встановлює рівень в позицію, позначену в координатах x і в.

Для створення об’єкту потрібно скористатися конструктором класу LayerManager.

G LayerManager() – створює рівень.

А щоб додати рівні в гру необхідно використовувати наступні методи:

– Q void append(Layer 1) – додає рівень в менеджер рівнів;

– Q Layer getLayerAt (int index) – отримує рівень із заданим індексом.

3.5.2 Клас TiledLayer

За допомогою класу TiledLayer створюється фон ігрової сцени.

int getSize() – отримує загальну кількість рівнів;

void insert (Layer 1, int index) – підключає новий рівень в заданий індекс;.

void paint(Graphics g, int x, int y) – представляє поточний менеджер рівнів в заданих координатах;

void remove(Layer 1) – видаляє рівень з менеджера рівнів.

Припустимо, у вас є чотири рівні: фон, гравець, перешкоди і артефакти. Для того, щоб зв'язати всі чотири рівні воєдино, створюється об'єкт класу LayerManager і визивається метод append ().

Наприклад:

La'yerManager Im = new LayerManager ();

Im.append(fon);

lm.append(igrok);

lm.append(prep);

lm.append(artf);

Тепер чотири перераховані рівні відображаються на ігровому полі.

3.5.3 Клас Sprite

Механізм роботи з об'єктом класу Sprite ідентичний моделі роботи з класом TiledLayer. Але якщо клас TiledLayer в основному відповідає за фонове зображення, то за допомогою класу Sprite малюються на екрані основні анімовані герої, космічні кораблі, машини, люди, звірі, артефакти і так далі. Зображення, що завантажується в гру, може бути виконане у вигляді анімаційної послідовності. Кількість малюнків в анімаційній послідовності необмежено, а відлік походить від нуля. Розташовуватися малюнки можуть як у вигляді рядку, так і у вигляді колонки, залежно від ваших переваг. Кожен малюнок анімаційної послідовності називається фреймом. Фрейм може бути будь-якого розміру по ширині і висоті, але всі фрейми мають бути однакових розмірів. Розмір фрейму має бути відомий, тому що він використовується при створенні об'єкту класу Sprite. Є три

конструктори класу Sprite кожен з яких можна використовувати при створенні об'єкту класу Sprite:

– Sprite(Image image) – створює не анімований спрайт;

– Sprite(Image image, int frameWidth, int frameHeight) – створює анімаційний спрайт, узятий із заданого фрейма;

– Q Sprite(Sprite s) – створює спрайт з іншого спрайту.

Для переходу по фреймах вихідного зображення, а також визначення зіткнення між об'єктами використовуються методи класу Sprite.

boolean collidesWith (Sprite s, boolean pixelLevel) – вираховує зіткнення між спрайтом;

boolean collidesWith(TiledLayer t, boolean pixelLevel) – определяет зіткнення між спрайтом і перешкодою, намальованою за допомогою классуTiledLayer;

int getFrame() – отримує поточний фрейм;

void nextFrame() – здійснює перехід на наступний фрейм;

void paint (Graphics g) – малює спрайт;

void prevFrame () – здійснює перехід, на попередній фрейм;

void setFrame(int sequencelndex) – встановлює заданий фрейм;

void setFrameSequence (int [] sequence) – встановлює певну фреймову послідовність;

void setlmage(Image img, int frameWidth, int frameHeight) – змінює зображення спрайту на нове зображення;

void setTransform(int transform) – виробляє трансформацію спрайту.

public void defineReferencePixel (int x, int y) – змінює опорну позицію спрайту, переносячи її в крапку з координатами x і y.

Метод defineReferencePixel () змінює опорну позицію спрайту, але для чого це потрібно? Опорна позиція для спрайту задається лівим верхнім кутом, але не завжди це зручно, тому опорну позицію можна перенести, в центр спрайту. Наприклад, якщо спрайт зроблений у вигляді машини, то при обертанні довкола своєї осі, якщо опорна позиція перенесена в центр, обертання буде правдоподібним. Але якщо не перевизначити позицію, то обертання станеться в точці лівого верхнього кута спрайту і виглядати це буде не цілком природно, неначе у машини пробите одне з коліс. Для цього викликається метод defineReferencePixel () із заданими координатами і перевизначається опорна позиція, наприклад в центр спрайту:

defineReferencePixel(frameWidth / 2, frameHeight / 2);

– nodsetTransform () виробляє трансформацію спрайту, обертаючи або дзеркально відображуючи спрайт довкола своєї осі за допомогою наступних констант:

– static int TRANS_MIRROR;

– static int TRANS_MIRROR_ROT180;
– static int TRANS_MIRROR_ROT270;
– static int TRANS_MIRROR_ROT90;
– static int TRANS_NONE;
– a static int TRANS_ROTl80;
– D static int TRANS_ROT270;
– D static int TRANSROT90.

Розглянувши класи GameCanvas, Layer, Sprite, TiledLayer і Layer-Manager, зробимо за допомогою їх невеликий застосунок.

3.5.4 Створення фонового зображення

За допомогою класу TiledLayer можна створювати будь-яку кількість рівнів, накладаючи їх один на одного, а за допомогою менеджера рівнів, представленого класом LayerManager, відстежувати всі наявні рівні. Як приклад буде створений фон на основі елементів розмітки ігрового поля. Фонове зображення завантажується з файлу fon.png. Само зображення виконане у вигляді шести вічок розміром 15x15 пікселями (лістинг 3.8), [9].

```
/** лістинг клас MainGame */
Import javax.microedition.lcdui.*;
Inport javax.microйdition.midlet.*;
public class MainGame extends MIDlet inplements CommandListener { //
команда виходу
private Command exitMidlet = new Command (‘Вихід’,Command.
EXIT,0) ;
// об’єкт класу MyGameCanvas
privâte MyGameCanvas mr;
public void startAppO {
// обробляємо виняткову ситуацію try{
/ / ініціалізували об’єкт класу MyGameCanvas mr = new
MyGameCanvas() ;
// запускаємо потік mr.start () ;
// додаємо команду виходу mr.addCommand(exitMidlet);
mr.setCommandListener(this);
/ / відображаємо поточний дисплей
Display.getDisplay(this).setCurrent(mr);
}catch (java.io.IOException zxz) {};
}
public void pauseApp () {}
public void destroyApp (boolean unconditional )
{
```

```
// зупиняємо потік if(mr != null) mr.stopO;
}
public void commandAction(Command c, Displayable d){
if (c == exitMidlet) {
destroyApp(false); notifyDestroyedO ;
}
} )
/**
Файл MyGameCanvas. java
клас MyGameCanvas */
import java.io.IOException; import javax.microйdition.lcdui.* ; import javax.microйdition.lcdui.game.* ;
public class MyGameCanvas extends GameCanvas implements Runnable {
// створюємо об'єкт класу TiledLayer private TiledLayer fonPole; // створюємо об'єкт класу LayerManager private LayerManager lm;
// логічна змінна для виходу з циклу boolean z;
public MyGameCanvas() throws IOException {
/ / звертаємося до конструктора суперкласу Canvas super(true);
// ініціалізували fonPole fonPole = Fon(); // створюємо менеджер рівнів
lm = new LayerManager ();
```

Лістинг 3.8 – Код прикладу фонового зображення

4 Розробка застосунків для мобільних платформ Iphone, Android, Blackberry

4.1 Вступ

Останні кілька років у світі спостерігається все більше поширення пристроїв на базі мобільних платформ. З'являються нові мобільні платформи. У цей час основна частина ринку поділена між декількома мобільними платформами. З погляду різних аналітичних агентств, найбільше активно серед них у найближчі кілька років будуть розвиватися iphone, Android і Blackberry.

По своїй продуктивності мобільні пристрої вже досягли рівня настільних комп'ютерів. Раніше до платформ, під керуванням яких працювали, наприклад, мобільні телефони, пред'являлися жорсткі вимоги. Невід'ємною частиною мобільних платформ був набір застосунків, ретельно налагоджених, протестованих виробником і завантажених на мобільний пристрій разом із самою платформою. Не

було широко поширене завантаження сторонніх застосунків. Для нових типів мобільних платформ, навпаки, завантаження сторонніх застосунків є однієї з основних можливостей. Сторонні розроблювачі одержали можливість створення й установки своїх застосунків на мобільні платформи. У той же час компанією Apple був створений новий механізм масового поширення мобільних застосунків – App Store. Механізм виявився успішним і був узятий на озброєння іншими компаніями-розробниками мобільних платформ. Цей механізм дозволив стороннім розроблювачам мати прибуток від створення й поширення мобільних застосунків.

Зараз у сегменті ринку мобільних застосунків працюють окремі розробники й групи розробників, безпосередньо орієнтовані на розробку мобільних застосунків, а також It-компанії з багаторічним досвідом розробки програмного забезпечення, які відкрили новий напрямок своєї діяльності.

Розробка мобільних застосунків у багатьох аспектах подібна з розробкою застосунків для настільних комп'ютерів. У той же час розробка мобільних застосунків ставить перед програмістами нові завдання. До таких завдань, зокрема, відносяться більш уважне керування пам'яттю мобільної платформи, облік обмеженого заряду батареї, а також облік можливої нестабільності бездротового мережного з'єднання.

Одержання знань по розробці застосунків для мобільних платформ може бути цікаво як з погляду застосування в професійній діяльності, так і з погляду більш глибокого розуміння роботи мобільного програмно-апаратного комплексу.

Мобільні пристрої на базі нових мобільних платформ, у тому числі смартфони, стають частиною повсякденного життя. Вони використовуються для спілкування. Поряд зі стандартними можливостями мобільних телефонів, як-от: дзвінки, обмін текстовими повідомленнями через SMS або в соціальній мережі – створюються застосунки, що інтегрують функції обміну інформацією, що працюють одночасно з декількома соціальними мережами. Усе активніше відбувається обмін відеоінформацією на основі камер, що вбудовуються в смартфони. У подорожах виявляється дуже затребувана функціональність GPS-навігатора. Росте інтерес до інтеграції інформації від GPS-навігатора смартфона в соціальні мережі. У розроблювачів з'являється можливість використовувати власні застосунки у своєму повсякденнім житті, що дозволяє їм по-новому глянути на свій продукт. При розробці мобільних застосунків цей фактор може додатково мотивувати й збільшити інтерес до процесу навчання [12].

4.2 Розробка мобільних застосунків

Однієї з перших тенденцій, що супроводжують швидкий розвиток мобільних платформ, став перенос на них практично всіх технологій, розвинених для настільних комп'ютерів [13]. У розробці застосунків намітилися кілька напрямків. Серед них варто відзначити ігрові застосунки, застосунки для роботи із соціальними мережами й бізнес-застосунки, орієнтовані на документообіг у середніх і великих компаніях. По способу розробки виділилися два основні виду застосунків: «native» застосунки й web-застосунки. «Native» застосунками створюються повністю на базі власного API мобільної платформи. Web-застосунки включають клієнтський застосунок, на основі невеликої кількості «native» коду, що одержує контент від віддаленого серверного застосунка, що й відображає його з використанням стандартних web-технологій, у тому числі HTML, Java Script. Віддалений серверний застосунок може бути створене заново або може бути задіяний наявний серверний застосунок з відомим API.

Ігрові застосунки пишуться на «native» коді для досягнення максимальної продуктивності. Для застосунків, що працюють із соціальними мережами, створюється клієнтський застосунок, що використовує для одержання даних стандартний API сервера тієї або іншої соціальної мережі. Для розробки бізнес-застосунків використовується як «native» код, так і web-технології.

При розробці мобільного застосунку програміст повинен розуміти, як його застосунок використовує пам'ять платформи. Кілька корисних підходів для зменшення використання пам'яті при розробці мобільних застосунків представлені в статті «On Mobile Java Memory Consumption» [14]. Зазначені підходи в першу чергу орієнтовані на платформу Java ME. Однак вони можуть бути поширені й на інші мобільні платформи. Підходи включають відмову, по можливості, від використання маленьких класів в одному коді, тому що класи вимагають додаткової пам'яті, і, коли необхідний окремий оброблювач команд для елемента користувацького інтерфейсу, рекомендується використовувати основний об'єкт як оброблювача, навіть якщо потрібна додаткова логіка. Інший підхід полягає в тому, щоб уникати залежностей між класами й підбирати розмір таких структур даних, як вектори й рядку. Необхідно приділяти увагу успадкуванню й переконуватися, що будь-який клас, який успадковує методи, використовує ці методи. Усі ці підходи сприяють збільшенню продуктивності мобільного застосунку. Їхнє застосування буде сприяти формуванню більш компетентних програмістів.

Іншою проблемою для мобільних пристроїв є велика різноманітність їх екранів і роздільних здатностей. Практичне рішення для створення «гнучкого» графічного користувацького інтерфейсу для платформи Java ME показане в статті «Better Midlets by design» [15]. Воно засноване на приділенні уваги тому, як пункт меню відображається на екрані при використанні високорівневого API платформи Java ME, а у випадку використання низькорівневого API вимагає більшого часу тестування на різних пристроях.

4.3 Мобільні платформи

У даний момент серед досить широкої різноманітності мобільних платформ на світовому ринку виділяються iphone, Android і Blackberry. Платформи iphone і Blackberry розробляються й підтримуються на ринку в основному тільки компаніями-розроблювачами. Вихідні коди цих платформ закриті. На противагу iphone і Blackberry, платформа Android, розроблена компанією Google, підтримується альянсом компаній OHA (Open Handset Allianse), серед яких присутні такі компанії, як HTC, Intel, Samsung. Вихідні коди цієї платформи відкриті. Це дозволяє різним компаніям-розробникам мобільних апаратних систем використовувати Android у якості базової програмної платформи. Кожна компанія може доробити платформу Android у відповідності зі своїми вимогами.

Кожна з перерахованих вище мобільних платформ має свою програмну мову й інструментарій для розробки застосунків. Важливим питанням для програміста, що вирішив спробувати свої сили в розробці мобільних застосунків, є вибір мобільної платформи. І дійсно, програмна мова й інструментарій для платформ iphone, Android і Blackberry суттєво відрізняються. Перехід між ними багато в чому вимагає повторного вивчення нової платформи.

4. 4 Мови й інструментальні засоби

4.4.1 Iphone

Використовується мова Objective-C. Розробка ведеться на настільній операційній системі Mac OS X. У якості інтегрованого середовища розробки використовується Xcode. В Xcode інтегровано застосунок для пошуку витоків і спостереження за виділенням пам’яті Instruments і застосунок Interface Builder для створення графічного інтерфейсу. Для симуляції роботи застосунку на мобільному пристрої служить iphone Simulator. З використанням iphone Simulator не можна перевірити роботу деяких специфічних API, наприклад роботу з акселерометром, геопозиціонуванням і камерою. iphone Simulator

входить до складу iphone SDK (Software Development Kit). Iphone SDK містить фреймворки й бібліотеки для розробки застосунків під реальний пристрій iphone і iphone Simulator. Xcode і iphone SDK можна завантажити on-line [16] з офіційного сайту розроблювачів застосунків для платформ компанії Apple. Для завантаження інструментарію потрібна реєстрація. Реєстрація припускає участь в одній із програм для розроблювачів. Найбільш простою є програма ios Developer Program. Вона безкоштовна й дозволяє завантажити й використовувати інструментарій розробки, а також завантажувати й тестувати власні застосунки на iphone Simulator. Можливість завантаження застосунків на реальний пристрій відсутня. Для одержання безкоштовної можливості завантаження й тестування на реальному пристрої можна використовувати програму ios University Program. Однак потрібне підтвердження того, що учасник програми дійсно буде використовувати реальний пристрій iphone лише з метою університетського навчання.

На офіційному сайті розроблювачів застосунків для платформ компанії Apple також можна знайти докладні інструкції з установки й використанню інструментарію.

4.4.2 Android

Споконвічно була можливість розробляти застосунки тільки з використанням мови Java у формі JNI (Java Native Interface), розробленого компанією Google. Застосунки для платформи Android збираються в байткод для виконання віртуальною машиною Dalvik. При розробці застосунків можливо використовувати сторонні jar-файли, зібрані для віртуальної машини Java. У процесі складання байткод віртуальної машини Java перезбирається в байткод віртуальної машини Dalvik спеціальною утилітою, що входить до складу інструментарію розробки застосунків.

У червні 2009 року вийшов перший офіційний реліз Android NDK (Native Development Kit). Розроблювачі одержали можливість розробляти власні бібліотеки мовою C для платформи Android. Використання коду мовою C у критичних по продуктивності ділянках застосунку суттєво підвищує його швидкодію. Розробка може вестися на інструментальних машинах під керуванням Windows, Linux, Mac OS X. Існують два способи розробки: з використанням консольних утиліт і з використанням ADT plug-in для Eclipse. Для імітації роботи на реальному пристрої Android служить емулятор Android, створений на базі віртуальної машини Qemu. Емулятор Android входить до складу Android SDK. Також в Android SDK входять засоби профілювання, налагодження й створення графічного інтерфейсу застосунків.

Завантажити Android SDK і Android NDK можна on-line [17] з офіційного сайту розроблювачів для платформи Android. На тій же сторінці Internet розташоване посилання на покрокову інструкцію з установки інструментарію.

Використання інструментарію розробки під платформу Android - емулятора, а також завантаження й тестування на реальному пристрої не вимагає оплати.

4.4.3 Blackberry

Інструментальна машина для створення застосунків може працювати під керуванням однієї з операційних систем: Windows, Linux, Mac OS X. Розробка застосунків ведеться мовою Java. Використовується платформа Java Micro Edition (Java ME) у конфігурації Connected Limited Device Configuration (CLDC) і Mobile Information Device Profile (MIDP). Платформа Java ME у зазначеній конфігурації використовується також для досить широкого набору інших платформ. Крім того, платформа Java ME простіше платформи Java Standard Edition (SE), вивчення якої часто входить у програму навчання комп'ютерним наукам. Це спрощує початок розробки мобільних застосунків під платформу Blackberry для студентів факультетів комп'ютерних наук. Для ознайомлення з Java ME рекомендується використовувати Sun Microsystem Java Wireless Toolkit для CLDC. У ній є вбудований емулятор роботи застосунків на мобільних пристроях. Для розробки застосунків як для Java ME, так і для Blackberry, використовується Blackberry Java Development Environment (JDE). JDE існує у вигляді окремого IDE і у вигляді plug-in для Eclipse. Blackberry JDE дозволяє використовувати такі функціональні можливості апаратної платформи Blackberry, як камера, медіа, акселерометр. JDE включає необхідні інструменти для конвертування існуючих Java ME Midlets у формат (.cod) для платформи Blackberry. Завантажити інструментарій для розробки застосунків для платформи Blackberry можна після переходу по відповідних до посилання на сторінці [18] офіційного сайту платформи Blackberry.

5 Інструментарій кросплатформної розробки

Відмінності в мові й інструментарії розробки для розглянутих мобільних платформ визначають істотні часові й економічні витрати при розробці та підтримці кросплатформних застосунків. А завдання створення мобільного застосунку, здатного працювати на декількох платформах, залишається актуальним. Існує ряд кросплатформних інструментів, які дозволяють використовувати той самий код для

декількох мобільних платформ. Для розробки застосунків під конкретну платформу повинен бути встановлений SDK для цієї платформи. До кросплатформних інструментів, прикладом, ставляться Rhodes і Phonegap [19].

5.1 Rhodes

Інструмент створений компанією Rhomobile (рис. 5.1). Доступний у тому числі й для платформ iphone, Android і Blackberry. Дозволяє створювати кросплатформні застосунки з використанням HTML, CSS, Javascript і Ruby. Призначений для програмістів, які мають досвід web-розробок й прагнуть створювати мобільні застосунки без необхідності вивчати інструментарій і мови для кожної мобільної платформи. Rhodes у першу чергу призначений для створення бізнес-застосунків з набором екранів, що складаються зі стандартних графічних елементів. Не призначений для розробки ігор, де потрібна висока продуктивність, а також застосунків, що включають специфічні графічні елементи. Rhodes є відкритим продуктом.

Рисунок 5.1 – Програмні продукти фірми Rhomobile

5.2 Phonegap

Відкритий фреймворк для створення «native» застосунків з використанням HTML, CSS і Javascript, у тому числі, для платформ iphone, Android і Blackberry. Проект спонсорується компанією Nitobi. Дозволяє через Javascript Apis використовувати дані контактів, геопозиціонування, камери й акселерометра, які не доступні для web-застосунків. Дозволяє легко конвертувати існуючі мобільні web-застосунку для роботи в Phonegap (рис. 5.2). Phonegap Apis міняються досить часто, що пояснюється прагненням задовольняти стандартам, що розвиваються, W3C. Як і все кросплатформні фреймворки не дуже підходить для створення застосунків, що вимагають інтенсивних математичних обчислень або 3D-анімації.

Рисунок 5.2 – Фреймворк «Phonegap»

6 Мобільні пристрої в курсах комп'ютерних наук

На цей момент для навчання розробці мобільних застосунків створені навчальні програми в багатьох університетах і інститутах по усьому світу. Серед них можна назвати Smartphone Programming Course for Undergraduates «Mobile Application Development» в Wireless Institute, University of Notre Dame [20], Mobile Software Engineering Undergraduate Degree Programme, University of Glasgow [21], University of Illinois computer science project Mobile Learning Communities (MLC), [22].

6.1 Nordsecmob

Nordsecmob – магістерська програма по комп'ютерній безпеці й мобільним технологіям. Програма дає студентам широкі знання по різних аспектах сучасних архітектур обміну даними і їх розвитку. Ключовими областями навчального плану є мережні застосунки й сервіси, інформаційна безпека й робота з мережею з використанням мобільних пристроїв. Магістри будуть здатні розробляти свої власні рішення для мобільного Internet.

Програма Nordsecmob є одним з магістерських курсів вищої європейської якості, обраних European Commission Erasmus Mundus Programme. Програми Erasmus Mundus призначені для студентів будь-якої країни світу. Учасниками програми вже стали багато студентів з України.

На сайті програми Nordsecmob [23] можна знайти повні тексти магістерських дисертацій її учасників. Частина магістерських досліджень виконана на мобільній платформі Android. Необхідно відзначити, що базовим університетом програми є Aalto University School of Science (раніше Helsinki University of Technology, TKK), Finland, а одним зі спонсорів програми є компанія Nokia. Із цими факторами зв'язана помітна орієнтованість програми на платформи Symbian і Meego (Maemo), що розвиваються компанією Nokia.

6.2 CMER

Centre for Mobile Education and Research (CMER) [24] створений на базі Department of Computing and Information Science, University of

Guelph, Canada. Місією CMER є залучення на передовий рубіж прикладних досліджень у розробці новітніх застосунків і сервісів для сприяння розробок в області мобільних пристроїв, а також інтеграція мобільних пристроїв у навчальний план комп'ютерних наук. CMER частково організований компанією RIM, виробником мобільної платформи Blackberry.

В CMER розроблений академічний цикл для інтеграції мобільних пристроїв у комп'ютерний світ (CMER Academic Kit for Integrating Mobile Devices into Computing Education). На сайті CMER представлені докладні матеріали по академічному циклу. Опис циклу може бути знайдене в статті Mahmoud Q.H. і ін. «An Academic Kit for Integrating Mobile Devices into the CS Curriculum» [25].

Цикл включає навчальні матеріали для повного вступного курсу по розробці мобільних застосунків, а також конкретні навчальні модулі для інтеграції мобільних пристроїв у курси програмної інженерії, розробки ігор, web – сервісів, інформаційної безпеки й операційних систем. В описі циклу демонструються приклади його успішного впровадження в навчання в Department of Computing and Information Science, University of Guelph. В описі циклу відзначається, що у фактичному стандарті вищої комп'ютерної освіти ACM Computing Curricula [26] для США й Канади курс Wireless and Mobile Computing є факультативним у рамках галузі знань «Net-Centric Computing». Затверджується необхідність включення курсу по розробці мобільних застосунків у сучасний обов'язковий навчальний план для комп'ютерних наук.

6.3 Відкриті платформи для мобільних пристроїв

У [27] розміщено ресурс, на якому концентруються матеріали про застосування ОПП при розробці застосунків для мобільних пристроїв і, насамперед, для тих, що інтенсивно розвивають мобільні Інтернет-пристрої (Mobile Internet Devices – MID). Ресурс фокусується на Інтернет планшетах Nokia (NIT) на основі ОПП maemo і класичних платформах на базі ОС Symbian (S60). На ресурсі в першу чергу представляються навчальні матеріали (або посилання на них), призначені для освоєння методів розробки застосунків студентами й початківцями, роботи в середовищах відкритих програмних платформ (у тому числі й мобільних) [28].

7 Мобільні платформи як об'єкт наукових досліджень

Бурхливий індустріальний розвиток мобільних платформ у цей час має своєю основою економічний інтерес як великих компаній, так і окремих програмістів. Розробка застосунків для мобільних платформ по більшій

частині базується на накопиченому досвіді й «кращих практиках». Але систематичний науковий аналіз у цій новій сфері програмування проводився в набагато меншій ступені в порівнянні, наприклад, з розробкою застосунків для настільних комп'ютерів. У той же час при розробці мобільних застосунків є цілий ряд особливостей, які немає необхідності враховувати при розробці інших типів застосунків.

У своїй статті «Software Engineering Issues for Mobile Application Development» [29] Wasserman виділяє дві області досліджень в інженерії мобільного ПО: користувацький досвід (The User Experience) і не функціональні вимоги (Non-functional Requirements). З користувацьким досвідом автор зв'язує наступні можливі напрямки досліджень:

– Як визначити, які функції повинні бути присутнім у мобільній версії традиційного застосунку? Чи існують техніки, які можуть гарантувати максимальне повторне використання коду серед різних версій?

– Які необхідні зусилля для створення «native» мобільних застосунків (або їх наборів для різних платформ) у порівнянні з мобільними web-застосунками? Чи є вимірювана відмінність у задоволенні користувачів або продуктивності між «native» і web-застосунками?

– Як дизайнер ПО інтегрує різні форми введення й сенсори в дизайн застосунку?

З не функціональними вимогами автор зв'язує наступні можливі напрямки досліджень:

– Чи повинні застосунки бути спроектовані по-різному залежно від швидкості передачі даних у мережі, у якій вони будуть використовуватися? В Азії деякі країни надають швидкість передачі даних 50 МБ/с або вище в той час, як типові швидкості, приміром, у США, навіть для мереж 3G, нижче 1 МБ. Як розроблювач повинен створювати застосунок, щоб максимально збільшити час роботи батареї й використання ресурсів?

Це лише невеликий перелік можливих напрямків досліджень.

В зв'язку з бурним індустріальним розвитком мобільних платформ вивчення розробок мобільних додатків актуальне як з точки зору вищої освіти, так і з точки зору науки. Закордоном і в Україні до цього часу прийняті стандарти вищої освіти в галузі інженерії мобільних пристроїв. Є цілий ряд прикладів, в основному в Європі та Північній Америці, успішного впровадження курсів інженерії мобільних пристроїв та розробки додатків для них в стандартні курси комп'ютерних наук. Має сенс на базі українського та зарубіжного досвіду інтегрувати вивчення інженерії мобільних пристроїв і, зокрема, розробку мобільних додатків в базові курси комп'ютерного формування

ВУЗів. У зв'язку з бурхливим розвитком платформ iPhone, Android та BlackBerry в першу чергу слід розглянути можливості зазначеної інтеграції на основі саме цих платформ [30].

Висновок

Оскільки практично всі сьогодні носять з собою смартфони і різні мобільні пристрої, мобільні програми дозволяють мати необхідну інформацію під рукою. Ці програми дозволяють ефективно інтегрувати інформацію з соціальними мережами, сайтом компанії, мультимедійним контентом і засобами комунікації. Будь-яка необхідна інформація може бути представлена в одному застосунку і актуалізована для конкретної людини, місця і часу.

Такий спосіб концентрованого надання інформації на персональному мобільному пристрої дозволяє значно скоротити відстань між компанією, брендом і джерелом інформації з клієнтом, споживачем або співробітником. Це наділяє користувача застосунку додатковими можливостями, які недоступні йому при умовах відсутності подібної програми. Компанія ж отримує можливість постійної комунікації з користувачем за допомогою одного каналу, одночасно включає в себе функції кількох [31].

Наявність власних брендових і корпоративних мобільних застосунків, що виконують різні складні функції, – це сьогодні просто необхідність для підвищення ефективності компанії в умовах сучасного ринку.

Крім того, зараз розробником програм для мобільних пристроїв може стати абсолютно кожен.

Література

Основна

1. Маковеева М. М. Системы связи с подвижными объектами / Маковеева М. М., Шинаков Ю. С. – М. : Радио и связь, 2002. – 440 с.

2. Невдяев Л. М. Мобильная связь 3-го поколения. Серия изданий «Связь и бизнес» / Невдяев Л. М. – М. : МЦНТИ: 000 «Мобильные коммуникации» 2000. – 208 с.

3. Майер Р. Android 2: программирование для планшетных компьютеров и смартфонов / Ретро Майер ; пер. с англ.– М. : Эксмо, 2011. – 672 с.

4. Дэрси Л. Android за 24 часа. Программирование приложений под операционную систему Google / Дэрси Л., Кондер Ш. – М. : Рид Групп, – 464 с.

5. Махер А. Программирование для iPhone / Махер Али ; пер. с англ. – М. : Эксмо. 2010. – 368 с.

6. Здзиарски Дж. iPhone. Разработка приложений с открытым кодом / Здзиарски Дж. ; пер. с англ. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2009. – 368 с. : ил.

7. Петзольд Ч. Программируем Windows Phone 7 / Петзольд Ч.; пер. англ. – Redmond, Washington 98052-6399: Microsoft Press, 2011. – 695 с.

8. Байдачный С.С. Silverlight 4: Создание насыщенных Web-приложений / Байдачный С.С. – М.: СОЛОН-ПРЕСС, 2010. – 288 с. : ил.

9. Горнаков С. Г. Программирование мобильных телефонов на Java 2 ME – ДМК прес, Москва, 2005 р, – 311 с.

10. С. Хашими, С. Разработка приложений для Android / С. Хашими, С. Коматинени, Д. Маклин – СПб. : Питер, 2011. – 736 с. : ил.

11. Мобільний портал «Mobile Arsenal» [Електронний ресурс]. – Режим доступу : http://www.mobile-arsenal.com.ua/ glossary/bluetooth_profiles /goep/.

12. Голощапов А. Л. Google Android: программирование для мобильных устройств / Голощапов А. Л. – СПб. : БХВ-Петербург, 2011. – 448 с. : ил. + CD-ROM – (Профессиональное программирование).

13. Mahmoud, Q.H., and Popowicz, P., «A Mobile Application Development Approach to Teaching Introductory Programming», 40th ASEE/IEEE Frontiers in Education Conference, Session T4F, October 27–30, Washington, DC.

14. Hartikainen, V., Liimatainen, P.P., and Mikkokek, T., «On Mobile Java Memory Consumption.» Procceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-based Processing (PDP2006), Montbeliard-Sochaux, France, pp. 333–339.

15. Kontio, M., «Better MIDlets by design.» IBM developerWorks, www.ibm.com/developerworks/wireless/library/wi-design.html, просматривалось 24 января 2011.

16. iOS Dev Center: https://developer.apple.com/devcenter/ios/index.action#downloads.

17. Download the Android SDK: http://developer.android.com/sdk/index.html.

18. Key Information for Developing Java Applications: http://us.blackberry.com/developers/javaappdev.

19. Allen, S., and Graupera, V., and Lundrigan, L, «Pro Smartphone Cross-Platform Development».

20. A Smartphone Programming Course for Undergraduates: http://wireless.nd.edu/wi-nd-mobileapps-white-paper.pdf.

21. Mobile Software Engineering Undergraduate Degree Programme: http://www.gla.ac.uk/undergraduate/degrees/ mobilesoftwaree ngineering.

22. University of Illinois computer science project Mobile Learning Communities (MLC): http://cs.illinois.edu/news/2010/Jan12-1.

23. NordSecMob: http://nordsecmob.tkk.fi.

24. CMER: http://cmer.cis.uoguelph.ca.

25. Mahmoud, Q.H., Ngo, T., Niazi, R., Popowicz, P., Sydoryshyn, R., Wilks, M., and Dietz, D., «An Academic Kit for Integrating Mobile Devices into the CS Curriculum.», Proceedings of the 14th ACM-SIGCSE Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2009), Paris, France, pp. 40–44. или http://cmer.cis.uoguelph.ca/cmer-ak/AcademicKit-V1.0/publications/ ITICSE 2009-Paper.pdf.

26. ACM Computing Curricula (ACM-CC 2001): http://www.acm.org/education/education/education/curric_vols/cc2001.pd, (оновлен в 2008).

27. Открытые платформы для мобильных устройств: http://oss.fruct.org.

28. ФГОС ВПО по специальности 090301 Компьютерная безопасность: http://www.edu.ru/mon-site/pro/fgos/vpo/pv090301s.pdf.

29. Anthony I. Wasserman, «Software Engineering Issues for Mobile Application Development», FoSER 2010, November 7–8, 2010, Santa Fe, New Mexico, USA, http://www.cmu.edu/silicon-valley/wmse/wasserman-foser2010.pdf.

30. Мобільний портал «Mobile Arsenal» [Електронний ресурс].– Режим доступу : http://www.mobile-arsenal.com.ua/glossary/bluetooth_profiles /goep/.

31. Степаненко О.О. Система зберігання та формування списків покупок smart shopping list для Android-смартфонів / О.О. Степаненко, Є.М. Федорченко, П.М. Фердман // Тиждень науки : Збірник тез доповідей щорічної науково-практичної конференції серед студентів, викладачів, науковців, молодих учених і аспірантів ЗНТУ, 18-21 квітня 2017 р. – Запоріжжя : ЗНТУ, 2017. – С. 792 – 793.

32. Федорченко Є. М. Мобільний Android додаток «Аптечна довідка» / О.О. Степаненко, Є.М. Федорченко, С.О. Філатов // Тиждень науки: Збірник тез доповідей щорічної науково-практичної конференції серед студентів, викладачів, науковців, молодих учених і аспірантів ЗНТУ, 18–21 квітня 2017 р. – Запоріжжя : ЗНТУ, 2017. – С. 794–795.

Додаток А ОЗНАЙОМЛЕННЯ З АРХІТЕКТУРОЮ МОБІЛЬНОГО ЗАСТОСУНКУ

Мета: створення середи розробки мобільного застосунку на базі Android, створення екземпляру емулятору мобільного пристрою; вивчення структуру, принципів роботи та інсталяція застосунку на емулятор та мобільний пристрій.

Короткі теоретичні відомості

А.1 Створення середи розробки

Для розробки застосунків на платформі Android необхідно завантажити та встановити наступне програмне забезпечення, доступне для вільного завантаження:

1. ***Java Development Kit (JDK)*** – комплект розробника застосунків мовою Java, що включає в себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і середовище виконання *Java Runtime Environment, JRE* (доступне для завантаження за адресою: http://java.sun.com/javase/downloads/index.jsp).
2. **Eclipse IDE** – інтегроване середовище розробки для Java, що має можливість установки плагіну для розробки Android-застосунків (доступно для завантаження за адресою: http://www.eclipse.org/downloads/).
3. **Android SDK (Software Development Kit)** – набір бібліотек та інструментів для розробки Android-застосунків, включаючи емулятор мобільного пристрою (доступне для завантаження за адресою: http://developer.android.com/sdk/index.html). Android SDK включає в себе:

– *API Android SDK* – API-бібліотеки Android, що надаються для розробки застосунків;

– документація SDK;

– *Development Tools* – інструментальні засоби для розробки, що надають можливість компілювати та налагоджувати застосунки, що створюються;

– *Sample Code* – типові застосунки, що демонструють деякі можливості Android.

4. ***Android Development Tools (ADT)*** – плагін, що автоматизує процес побудови застосунків для Android, інтегруючи інструменти розробки безпосередньо в середовище розробки Eclipse.

Плагін ADT додає в Eclipse наступні компоненти:

– майстер створення проекту – *New Project Wizard,* котрий спрощує створення Android-проектів та формує шаблон проекту;

– редактор *Layout Editor* – для розробки графічного інтерфейсу застосунку;

– різні редактори ресурсів для створення, редагування та перевірки вірності XML-ресурсів розробника.

Установка всього програмного забезпечення полягає у завантаженні, розпакуванні архіву та запуску виконавчого файлу з розширенням *.exe. Після встановлення JDK, Eclipse IDE і Android SDK необхідно встановити плагін ADT. Для встановлення плагіну запускаємо Eclipse та обираємо пункт меню **Help | Install New Software**.

У діалоговому вікні, що з'явиться, натиснути кнопку Add. У полі **location** ввести «https://dl-ssl.google.com/android/eclipse/». Після виконання всіх інструкцій з установки перезапустіть середу Eclipse.

Тепер потрібно зв'язати Eclipse з каталогом Android SDK. Виберіть у головному меню Eclipse пункт **Window | Preferences**, після пункт **Android** та в полі **SDK location** вказати каталог, в якому розташований Android SDK.

А.2 Інструменти для розробки на налагодження застосунків

Окрім емулятору, SDK також включає безліч інших інструментальних засобів для налагодження та установки створюваних застосунків:

– *android* – інструмент розробки, що запускається з командного рядка, що дозволяє створювати, видаляти і

конфігурувати віртуальні пристрої, створювати і оновлювати Android SDK новими платформами, доповненнями і документацією;

– *Dalvik Debug Monitor Service (DDMS)* – інтегрований з **Dalvik Virtual Machine**, стандартної віртуальної машиною платформи Android, цей інструмент дозволяє керувати процесами на емуляторі або пристрої, а також допомагає у налагодженні застосунків. За допомогою цього сервісу можна завершувати процеси, обирати певний процес для налагодження, генерувати трасовані дані, здійснити моніторинг використання пам'яті мобільного пристрою, переглядати інформацію про потоки, робити скріншоти емулятора або пристрою та багато іншого;

– *Hierarchy Viewer* – візуальний інструмент, який буде налагоджувати і оптимізувати користувальницький інтерфейс застосунку, що розробляється. Він показує візуальне дерево ієрархії класів View, аналізує швидкодію перемальовування графічних зображень на екрані і може виконувати ще багато інших функцій для аналізу графічного інтерфейсу застосунків;

– *Layoutopt* – інструмент командного рядка, який допомагає оптимізувати схеми розмітки та ієрархії розміток в створюваному застосунку. Необхідний для вирішення проблем при створенні складних графічних інтерфейсів, які можуть зачіпати продуктивність програми;

– *Draw 9*-patch – графічний редактор, який дозволяє легко створювати NinePatch-графіку для графічного інтерфейсу розроблюваних застосунків;

– *Sqlite3* – інструмент для доступу до файлів даних SQLite, створених і використовуваних Android-застосунками;

– *Traceview* – цей інструмент видає графічний аналіз трасованих логів, які можна генерувати з застосунків;

– *Mksdcard* – інструмент для створення образу диску, який можна використовувати в емуляторі для симуляції наявності зовнішньої карти пам'яті (наприклад, карти SD).

Для використання інструментів, що надаються Android SDK, необхідно налаштувати змінні оточення: додати в змінну оточення Path шляхи до підкаталогів Android SDK (наприклад, C:\eclipse\adt\sdk\platform-tools та C:\eclipse\adt\sdk\tools).

А.3 Android-емулятор мобільного пристрою

Android Virtual Device (віртуальний пристрій Android) – це емулятор, що запускається на звичайному комп'ютері. Емулятор використовується для проектування, налагодження і тестування застосунків в реальному середовищі виконання.

Перш ніж запустити Android-емулятор пристрою, необхідно створити екземпляр AVD. AVD визначає системне зображення та параметри пристрою, що використовуються емулятором.

Створити екземпляр AVD можна двома способами:

1. В командному рядку утилітою android .

2. За допомогою візуального інструменту Android SDK and AVD Manager. Його можна запустити з кореневого каталогу Android SDK або в середовищі Eclipse , обравши пункт меню **Window | Android Virtual Device Manager**. У вкладці *Android Virtual Devices* необхідно натиснути кнопку **New** і у вікні *«Create new Android Virtual Device»* задайте конфігурацію для створюваного емулятору пристрою (рис. А.1):

– *Name* – ім'я створюваного пристрою;

– *Device* – тип пристрою;

– *Target* – версія Android SDK, що підтримується емулятором Android-пристрою;

– *Hardware keyboard present* – якщо прапорець встановлено, то для набору тексту в емуляторі можливо буде використовувати не тільки віртуальну клавіатуру, але й клавіатуру комп'ютера;

– *Display a skin with hardware controls* – показує кнопки керування віртуальним пристроєм в вікні емулятору (home, menu, back і т.ін.). Рекомендовано відмітити.

– *Memory Options* – ємність оперативної пам'яті та пам'яті, що використовується застосунком;

– *Internal Storage* – вбудована пам'ять мобільного пристрою;

– *SD Card* – встановлює віртуальну карту пам'яті, для якої можна в текстовому полі Size задати необхідну ємність;

– *Emulator Option* – «*Use Host GPU*» (Graphics processing unit) для прискорення роботи емулятору, «Snapshot» – при встановленні прапорця, з'являється можливість зберігати стан системи в файл при закритті емулятора та відновлювати його при запуску, оминаючи тривале завантаження.

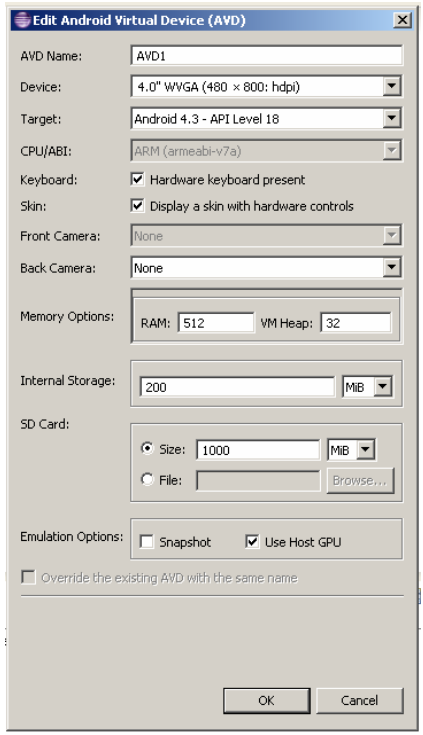

Рисунок А.1 – Створення нового AVD

Для зміни параметрів пристроїв у вікні *Android Virtual Device Manager* у вкладці *Device Definitions* необхідно натиснути кнопку *New Device*.

– *Name* – ім'я пристрою;

– *Screen Size* і *Resolution* – відповідно розмір діагоналі (у дюймах) і дозвіл (в пікселях) екрана. На їх основі розраховується щільність пікселів;

– *Sensors* – датчики пристрої: акселерометр (*Accelerometer*), гіроскоп (*Gyroscope*), GPS і датчик наближення (*Proximity Sensor*);

– *Cameras* – камери пристрою: *Front* – передня, *Rear* – задня;

– *Input* – пристрої введення: *Keyboard* – клавіатура, *DPad* – джойстик, *Trackball* – трекбол;

– *RAM* – розмір оперативної пам'яті за замовчуванням;

– *Size* – узагальнений розмір (клас). Розміри дисплеїв в Android залежно від діагоналі поділяються на 4 класи: *small* (< 3.55 in), *normal* (3.55–5 in), *large* (5 – 7.5in), *xlarge* (> 7.5 in);

– *Screen Ratio* – співвідношення сторін екрану: «*long*» для широкоформатних дисплеїв (наприклад, WQVGA, WVGA, FWVGA), «*nolong*» – для звичайних (наприклад, QVGA, HVGA, VGA);

– *Density* – клас щільності пікселів. На його основі розраховується величина dp. Вона використовується для однакового відображення графічних об'єктів на дисплеях з різною щільністю пікселів. Наприклад, картинка розміром 100×100px буде виглядати маленької на екрані з високою щільністю і великою на екрані з малою, а от 100×100dp виглядає приблизно однаково скрізь.

Існує всього 7 класів щільності: *nodpi* – величина dp прирівнюється до 1px, графіка залежить від щільності, *ldpi* – низька щільність пікселів ~ 120dpi, *mdpi* – середня щільність пікселів ~ 160dpi, *hdpi* – висока щільність пікселів ~ 240dpi, *xhdpi* – дуже висока щільність ~ 320dpi, *xxhdpi* – надвисока щільність >400dpi, *tvdpi* – середнє між *mdpi* і *hdpi* (~ 213dpi);

– *Buttons* – визначає якими будуть кнопки управління (home, menu та ін): *hardware* – фізичні (відобразяться у вікні емулятора) або *software* – програмні (як в планшетах).

– *Device State* – орієнтація екрану (*Portrait* – портретна або *Landscape* – альбомна) і навігації по екрану (*DPad* і *Trackball*).

А.4 Програмний стек Android

Детальний опис програмного стеку Android SDK представлено на рисунку А.2.

Центром платформи Android є ядро Linux, що відповідає за драйвера пристроїв, доступ до ресурсів, управління енергоспоживанням і вирішенням інших завдань ОС.

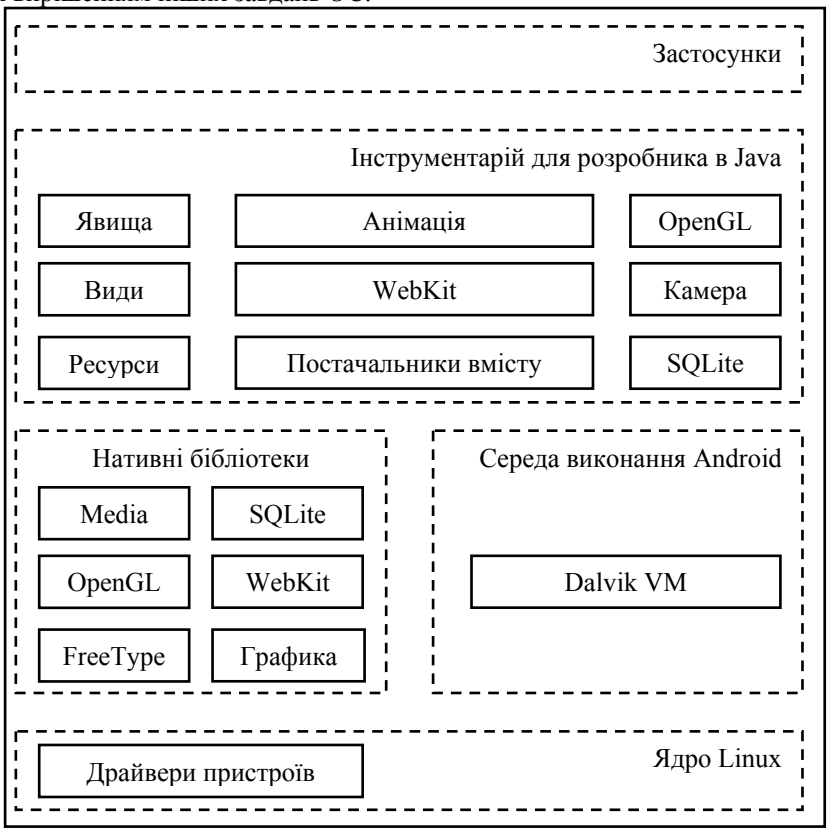

Рисунок А.2 – Опис програмного стеку Android SDK

На наступному рівні, вище ядра, знаходиться ряд бібліотек C/C++, зокрема OpenGL, WebKit, FreeType, Secure Sockets Layer (SSL), бібліотека часу виконання C (libc), SQLite і Media. Системна бібліотека C, заснована на Berkeley Software Distribution (BSD), налаштована для роботи з вбудованими пристроями, що працюють під Linux. Медіа-бібліотеки працюють на основі PocketVideo OpenCORE. Ці бібліотеки відповідають за запис та відтворення аудіо- та відеоформатів. Бібліотека Surface Manager контролює доступ до системи відображення даних і підтримує 2D і 3D. Бібліотека WebKit відповідає за підтримку браузерів.

Бібліотека FreeType підтримує шрифти. SQLite – це реляційна база даних, яка знаходитися на самому пристрої. Крім того, SQLite – це незалежна розробка з відкритим кодом, яка не пов'язана безпосередньо з Android. Можна використовувати інструменти, призначені для SQLite, і при роботі з базами даних Android.

Більша частина застосунків з цього набору звертається до зазначених кореневих бібліотек через **Dalvik Virtual Machine (Dalvik VM)**, що виконує на платформі Android роль шлюзу. Dalvik оптимізована для одночасного використання декількох екземплярів VM. Коли застосунки Java звертаються до цих кореневих бібліотек, кожний застосунок працює з власним примірником віртуальної машини.

В основних бібліотеках прикладного інтерфейсу програмування на Java містяться функції для телефонії, роботи з ресурсами, місцезнаходженням, користувальницькими інтерфейсами, постачальниками вмісту (даними), а також диспетчери пакетів (що відповідають за встановлення, безпеку і т.д.). Розробка застосунків ведеться на підставі прикладного інтерфейсу програмування Java. В Android підтримується бібліотека Google Skia, що призначена для роботи з 2D-графікою, що написана на C/C++ і бібліотека роботи з 3D – графікою на основі скороченої версії OpenGL ES, оптимізованої для роботи з вбудованими системами. Для роботи з медіа, на платформі Android підтримується більшість поширених форматів аудіо, відео та зображень. В області бездротового зв'язку Android має в своєму розпорядженні спеціальні API для підтримки Bluetooth, EDGE, 3G, Wi-Fi і глобальної системи мобільного зв'язку (GSM), залежними від обладнання.

A.5 Пакети Java для Android

Пакети, що входять до складу Android SDK: android.app (модель застосунків), android.bluethooth, android.content (постачальник вмісту для узагальнення обміну та зберігання даних), android.database (реферативна база даних), android.gesture (для роботи з жестами користувача), android.graphics, android.hardware (робота з камерою), android.net (мережеві API на рівні сокетів), android.location (інформація про місцезнаходження об'єкта), android.media, android.opengl, android.os (служби операційної системи: обмін інформацією між процесами, облік змін в файлах, використання потоку повідомлень тощо), android.preference (можливість користувача керувати налаштуваннями свого застосунку), android.provider (набір постачальників вмісту, що відносяться до інтерфейсу android.content), android.sax (набір API для XML), android.speech (робота з розпізнаванням мови), android.telephony (інформація про телефонний виклик), android.text (введення, обробка,

оформлення тексту), android.units, android.view, android.webkit (класи, що відносяться до веб – браузеру), android.widget (класи елементів управління користувальницького інтерфейсу), com.google.android.maps (класу для роботи з картами Google). Опис даних пакетів детально описаний в документації SDK та розглянуто в рекомендованому переліку літератури по виконанню лабораторних робіт.

А.6 Структура Android-проекту

А.6.1 Створення Android-проекту

Запускаємо Eclipse та обираємо **File|New|Project...**, в списку обираємо **Android|Android Application Project** і натискаємо кнопку **Next**. У відкритому діалоговому вікні майстра *New Android Project* заповніть наступні поля:

– ***Application name*** (ім'я застосунку, котре буде відображатися в його заголовку) – *SimpleProject*;

– ***Project name*** (ім'я проекту і ім'я каталогу, що буде містити проектні файли) – *SimpleProject*;

– ***Package name*** (ім'я пакету) – *com.example.simpleproject*;

– ***Minimum Required SDK*** обирається мінімальна підтримувана платформа, мінімальний рівень API;

– ***Target SDK*** – вказівка компілятору зібрати застосунок для обраного рівня API (виставляється автоматично залежно від обраної версії SDK);

– ***Compile With*** – тут за замовчуванням ставиться сама остання версія Android;

– ***Theme*** – тема застосунку.

Заповнивши усі поля, натискаємо на кнопку **Next** для переходу в наступне вікно.

– залишаємо прапорець у полі **Create custom launcher icon**, щоб мати можливість встановити власний значок для програми;

– залишаємо прапорець у полі **Create activity**;

– не ставимо прапорець у полі **Mark this project as library**, вона призначена для створення бібліотек;

– залишаємо прапорець у полі **Create Project in Workspace** – всі ваші проекти будуть зберігатися в спеціальній папці. Або ви можете задати свій шлях для проекту .

У наступному вікні майстра можна вибрати і налаштувати свій значок для програми. Після цього натискаємо двічі **Next** та **Finish**.

Щоб побачити застосунок в дії, виконуємо команду **Run** (рис. А.3). При цьому середовище розробки автоматично інсталює його на Android-емулятор.

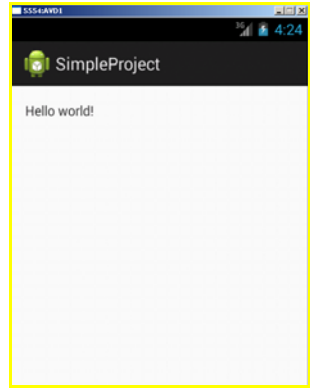

Рисунок А.3 – Вікно програми SimpleProject

А.6.2 Зміст Android-проекту

ADT – плагін при створенні Android-проекту організує структуру у вигляді дерева каталогів, як і будь-який інший Java-проект (рис. А.4):

– ***scr*** – каталог, що містить вихідний код програми;
– ***assets*** – довільний збір каталогів і файлів;
– ***res*** – каталог, що містить ресурси застосунку;
– *drawable* – каталог, який містить зображення або файли дескрипторів зображень;
– *anim* – каталог, що містить XML – дескриптори, в який описується анімація, що використовується програмою;
– *layout* – каталог, в якому містяться види даної програми (вони створюються за допомогою XML – дескрипторів, а не шляхом написання коду);
– *menu* – каталог, що містить файли XML – дескрипторів, в який описується меню, що використовується в програмі;
– *values* – каталог, що містить інші ресурси, використовувані програмою (рядки, стилі, кольори), що визначається XML – дескрипторами;
– *xml* – каталог, що містить додаткові XML – файли, використовувані застосунком;
– *raw* – каталог, що містить додаткові дані, які використовуються застосунком;
– ***AndroidManifest.xml*** – файл опису програми Android. У цьому файлі визначені явища, постачальники вмісту, служби та приймачі намірів (intent receiver) даного застосунку. Цей файл також можна

використовувати для декларативного пояснення прав доступу, необхідних для роботи з застосунком, і для надання специфічних прав доступу іншими застосунками, які користуватимуться службами даного. Крім того, у файлі можуть міститися інформація про інструментування (instrumentation detail), яка може використовуватися при тестуванні даного чи іншого застосунку.

Розглянемо файл **MainActivity.java**.

```
package com.example.simpleproject;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
```

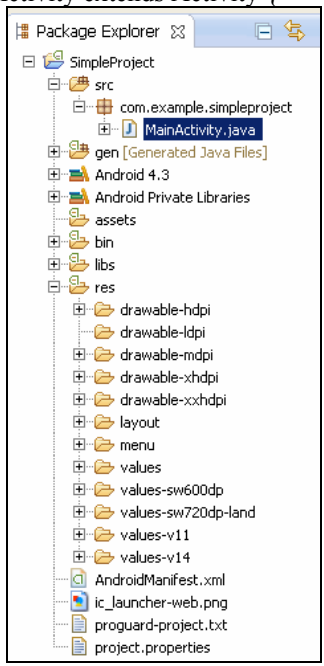

Рисунок А.4 – Структура програми SimpleProject

```
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
public boolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
```

return true;
}
}

У першому рядку йде назва пакету (з Package Name). Далі – рядки імпорту необхідних класів для проекту. Потім йде оголошення самого класу, який успадковується (extends) від абстрактного класу Activity. У самому класі є метод **onCreate()** – він викликається, коли застосунок створює і відображає розмітку активності .

Рядок **super.onCreate (savedInstanceState)** – це конструктор батьківського класу, що виконує операції для роботи активності.

Другий рядок **setContentView (R.layout.activity_main)** – підключає вміст з файлу розмітки. Як аргумент ми вказуємо ім'я файлу без розширення з папки **res/layout/**(за замовчуванням **activity_main.xml**).

Ви можете перейменувати файл або створити свій файл з ім'ям lab.xml, тоді код буде виглядати так: setContentView (R.layout.lab);

Якщо ваш застосунок буде складатися з безлічі екранів, то ви будете створювати для кожного екрана свою розмітку і підключати її описаним вище чином.

Перегляд і редагування властивостей видів застосунку можливо з файлу **activity_main.xml**: оберіть потрібний об'єкт в графічній версії файлу і переглядайте властивості даного об'єкта (праворуч).

Додавання (зміна) рядків, зображень та інших ресурсів можливо з **/res/values:** для цього наприклад оберіть файл strings.xml і натисніть на кнопку Add. Майстер створення запропонує обрати тип елементу, а після його ім'я і значення.

А.6.3 Структура файлу AndroidManifest.xml

Файл маніфесту інкапсулює всю архітектуру Android-застосунку, його функціональні можливості і конфігурацію. Розглянемо призначення кожного з них.

– **<manifest>** – це кореневий елемент файлу, що містить наступні атрибути: xmlns:android (простір імен Android) з постійним значенням «http://schemas.android.com/ apk/res/android», package (ім'я пакету застосунку), android:versionCode (внутрішній номер версії) і android:versionName (номер користувальницької версії);

– **<uses-permission>** – описує дозволи безпеки, які потрібно надати вашому пакету. Кількість не обмежена. Наприклад, <uses-permission android:name = «android.permission. RECEIVE_SMS»/>;

– **<permission>** – оголошує дозвіл безпеки, що може використовуватися до певних компонентів або функціональності даного застосунку, а також права доступу інших застосунків до застосунку, що

розробляється, використовуючи атрибути: *android:name* (елемент дозволу), *android:label* (ім'я дозволу, що відображається користувачеві), *android:description* (опис), *android:icon* (іконка розширення), *android:permissionGroup* (визначає приналежність до групи дозволів) і *android:protectionLevel* (рівень захисту);

– **<permission-tree>** – оголошує базове ім'я для дерева дозволів (простір імен);

– **<permission>** – визначає ім'я для набору логічно зв'язаних рішень (категорія дозволів);

– **<instrumentation>** – визначає можливість контролювати взаємодію застосунку з системою;

– **<uses-sdk>** – визначає сумісність програми з зазначеної версією платформи Android (атрибут *minSdkVersion*);

– **<uses-configuration>** – вказує необхідну для застосунку апаратну і програмну конфігурацію мобільного пристрою;

– **<uses-feature>** – визначає функціональність, яка потрібна для роботи застосунку;

– **<supports-screens>** – визначає здатність екрану, необхідну для функціонування пристрою;

– **<application>** – кореневий елемент, що містить оголошення компонентів застосунку, доступних в пакеті. Цей елемент може також включати глобальні та/або задані за замовчуванням атрибути для програми, такі як мітки, значок, тема, вимоги дозволу, і т.п. Елемент містить наступні дочірні елементи: ***<activity>*** з атрибутами android:name, android:label та інші (кожен *<activity>* може містити елемент *<Intent-filter>,* що надає для компонентів-клієнтів можливість отримання Intent оголошеного типу, відфільтровуючі ті, що не значимі для компоненту, і містить в першу чергу дочірні елементи <action>, <category>, <data>), ***<activity-alias>*** – псевдонім Activity, ***<receiver>*** з внутрішнім елементом intent – filter (дозволяє застосунку повідомляти про заміну даних або про діях, які відбуваються, навіть якщо програма не виконується в даний час), ***<service>*** з внутрішнім елементом intent – filter (працює у фоновому режимі), ***<provider>-***компонент, який управляє постійними даними і відкриває до них доступ іншим застосункам, ***<uses-library>*** – визначає загальнодоступну бібліотеку для компіляції застосунку.

Завдання на роботу

Створити Android-проект за допомоги конструкторів та мастерів, змінивши властивості виду: встановити фон застосунку на обраний, змінити надпис, додати зображення. Приклад наведено на рис. А.5.

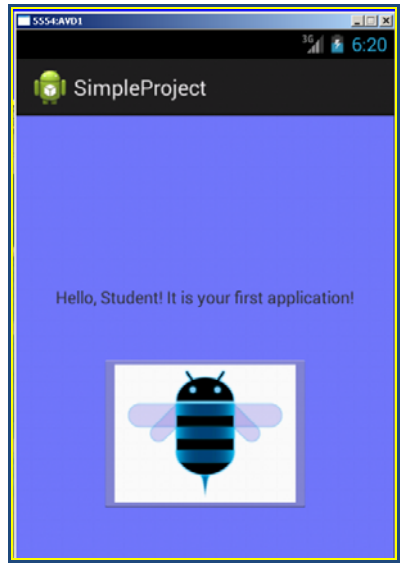

Рисунок А.5 – Вікно програми після зміни її властивостей

Контрольні питання

1. Програмний стек Android.
2. Dalvik Virtual Machine.
3. Відповідність версії платформи та рівень API.
4. Пакети Java для Android.
5. Створення AVD екземпляру (емулятору).
6. Каталоги ресурсів проекту.
7. Каталог констант застосунку /res/values/.
8. Файл AndroidManifest.xml.

Додаток Б ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА. ОСНОВНІ ВІДЖЕТИ ТА ОБРОБКА ПОДІЙ

Мета роботи: ознайомитися з основними віджетами, їх компоновкою та обробкою подій, викликаних користувачем.

Короткі теоретичні відомості

Б.1 Компоновка елементів управління

Компоновка – це архітектура розташування елементів інтерфейсу користувача для окремого вікна (**Activity**). Компоновка визначає структуру розташування елементів (тобто віджетів, **View**) у вікні та містить всі елементи, які надаються користувачу програми. При створенні компоновки треба враховувати те, що екрани мобільних пристроїв мають меншу роздільну здатність, наприклад, ніж монітори, різноманітність мобільних пристроїв з різними розмірами та щільністю пікселів, різні типи сенсорних екранів.

В android-застосунку графічний інтерфейс користувача формується за допомогою об'єктів **View** та **ViewGroup**. Клас View є базовим класом для ViewGroup та складається з набору об'єктів View. Об'єкти View – це основні модулі для створення графічного інтерфейсу користувача. Клас View є базовим для елементів управління (віджетів) – текстових полів, кнопок та ін. Клас ViewGroup представляє контейнер, який служить ядром для підкласів, що називаються компоновки (**layouts**). Ці класи формують розташування віджетів на формі та містять дочірні елементи View та ViewGroup (рис. Б.1). Таким чином для кожного Activity формується дерево ієрархії вузлів View та ViewGroup.

При запуску програми система отримає посилання на кореневий вузол дерева та використовує її для відображення графічного інтерфейсу користувача на мобільному пристрої. Система також аналізує елементи дерева ієрархії, додаючи їх до елементів-батьків. Для цього в методі **onCreate()** треба викликати метод **setContentView()** передаючи в якості параметру посилання на ресурс компоновки. Наприклад, якщо компоновка знаходиться в файлі mail.xml: // ініціалізація компоновки public void onCreate (Bundle savedInstanceState)

```
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
```

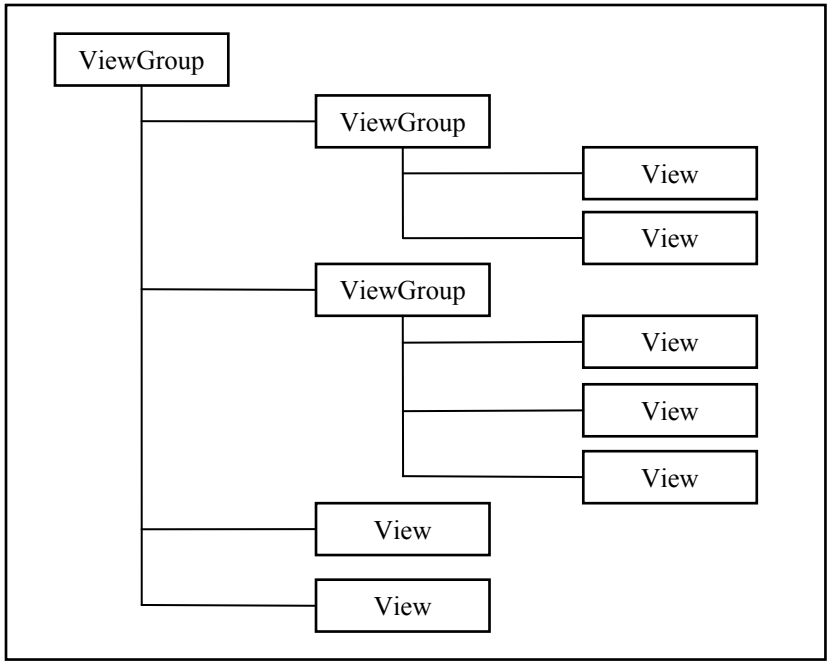

Рисунок Б.1 – Приклад дерева вузлів View та ViewGroup
для Activity

Б.2 Типи компоновок

Для створення вікон існує декілька стандартних типів компоновки, які є підкласами ViewGroup:

– FrameLayout;
– LinearLayout;
– TableLayout;
– RelativeLayout.

Розглянемо їх використання детальніше.

Б.2.1 Компоновка FrameLayout

FrameLayout є найпростішим типом компоновки. В основному це пустий простір на екрані, яке можна заповнити тільки єдиним дочірнім елементом View чи ViewGroup. Всі дочірні елементи додаються до верхнього лівого кута екрану (рис. Б.2).

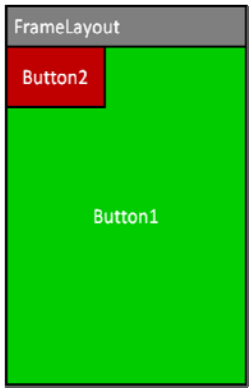

Рисунок Б.2 – Компоновка FrameLayout

В цій компоновці неможна визначити різноманітне місцезнаходження для дочірнього елементу. Всі дочірні елементи додаються поверх попередніх, частково чи повністю затіняючи їх. Виходячи з цього, батьківський елемент займає весь простір екрану та має наступні атрибути: layout_width = «match_parent» та layout_height=«match_parent». Дочірній елемент має дані атрибути з будь-яким параметром: *«match_parent»***(**відповідає батьку та приймає значення відповідного параметра як у батька) чи *«wrap_content»* (стандартний розмір елемента). Елементи інтерфейсу будуть розглянуті далі, проте ***основними атрибутами для кожного виду є наступні:*** **android:id, android:layout_width, android:layout_height, android:text.**

Компоновка FrameLayout використовується рідко, оскільки не дозволяє створити складні вікна з множиною елементів. Вона найчастіше використовується для оверлеїв. Наприклад, якщо у вікні є зображення, що займає весь екран (чи карта, кадр з відеокамери та ін.), а на ньому є елементи керування (наприклад, елемент масштабування, елемент пошуку, індикатор часу чи стану та ін.).

Наприклад, створимо компоновку FrameLayout, розмістивши на неї дві кнопки як на рисунку Б.2.

<FrameLayout xmlns:android=http://schemas.android.com/apk/res/android»
android:layout_width=«match_parent»
android:layout_height=«match_parent»>
<Button
android:id=«@+id/b1»
android:layout_heigth=«match_parent»
android:layout_width=«match_parent»

android:text=«Button1»
/>
<Button
android:id=«@+id/b2»
android:layout_heigth=«wrap_content»
android:layout_width=«wrap_content»
android:text=«Button2»
/>
</FrameLayout>

Б.2.2 Компоновка LinearLayout

Компоновка LinearLayout вирівнює всі дочірні елементи в одному з напрямів – вертикально чи горизонтально, в залежності від значення атрибуту android:orientation: *«horizontal»*, *«vertical»*.

Наприклад, створимо компоновку LinearLayout, розмістивши на неї кнопки вертикально як на рис. Б.3.

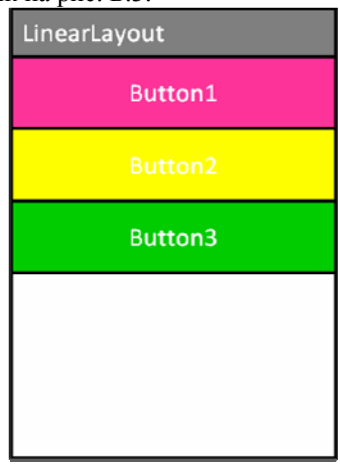

Рисунок Б.3 – Компоновка LinearLayout

<LinearLayout xmlns:android=http://schemas.android.com/ apk/res/android»
android:layout_width=«match_parent»
android:layout_height=«match_parent»
android:orientation=«vertical»>
<Button
android:id=«@+id/b1»
android:layout_heigth=«wrap_content»
android:layout_width=«match_parent»
android:text=«Button1»

```
/>
<Button
android:id=«@+id/b2»
android:layout_heigth=«wrap_content»
android:layout_width=«match_parent»
android:text=«Button2»
/>
</LinearLayout>
```

Компоновка LinearLayout має атрибут android: layout_weight, який визначає вагу дочірнього елементу, тобто надає можливість розширюватися (за замовчуванням, вага має нульове значення).

Оскільки компоновки можуть бути вкладеними, то у вікні можна розташувати наприклад декілька лінійних компоновок з будь-якою орієнтацією їх елементів. Таке застосування є найбільш розповсюдженим засобом при створенні інтерфейсу користувача для Android-застосунків.

Б.2.3 Компоновка TableLayout

Компоновка TableLayout позиціонує всі дочірні елементи у строки та стовбці. TableLayout не відображує лінії розділу чарунок. TableLayout може мати строки з різним числом чарунок. При формуванні компоновки деякі чарунки можна залишати порожніми. При створенні компоновки для строк використовуються об’єкти TableRow. Чарунка може бути об’єктом View чи ViewGroup.

Компоновка TableLayout на практиці застосовується рідко, зазвичай замість неї використовують декілька компоновок LinearLayout. TableLayout зручно використати, якщо розміщення елементів представлено у вигляді таблиці (рис. Б.4).

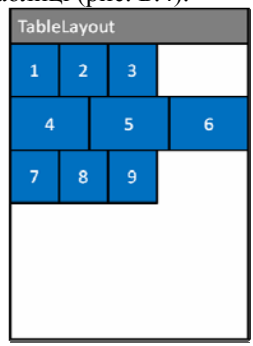

Рисунок Б.4 – Компоновка TableLayout

Наприклад, створимо компоновку TableLayout, розмістивши на неї кнопки як на рис. Б.4.

```
<TableLayout xmlns:android=http://schemas.andro
id.com/apk/res/android»
android:layout_width=«match_parent»
android:layout_height=«match_parent»>
<TableRow
androd:id=«@+id/TR1»
android:layout_heigth=«wrap_content»
android:layout_width=«match_parent»>
<Button
android:id=«@+id/b1»
android:layout_heigth=«wrap_content»
android:layout_width=«wrap_content»
android:text=«1»
/>
…
</TableRow>
<TableRow
androd:id=«@+id/TR2»
android:layout_heigth=«wrap_content»
android:layout_width=«match_parent»>
<Button
android:id=«@+id/b4»
android:layout_heigth=«wrap_content»
android:layout_width=«wrap_content»
android:text=«4»
/>
…
</TableRow>
…
</TableLayout>
```

Б.2.4 Компоновка RelativeLayout

Компоновка RelativeLayout (рис. Б.5) дозволяє дочірнім об’єктам визначати свою позицію відносно вказаного за ідентифікатором елемента. В RelativeLayout дочірні елементи розташовані так, що якщо перший елемент розташовано у центрі екрану, тоді інші елементи будуть вирівняні відносно центру екрану. При об’яві компоновки в XML-файлі, елемент, на який будуть посилатися інші об’єкти для позиціонування, треба об’явити раніш ніж елементи, які звертаються до

нього за його ідентифікатором. В компоновці RelativeLayout розташування елемента можна визначати відносно іншого елемента, на який посилаються через його ідентифікатор. Наприклад, якщо хочемо отримати ліворуч розташований елемент TextView від елемента button2, то: android:layout_toLeftOf=«@id/Button2». Розташування відносно вказаного: android:layout_above, android:layout_below, android:layout_toLeftOf, andr oid:layout_toRightOf. Вирівнювання: android:layout_ alignBaseline, android:layout_alignBottom, android: layout_alignLeft, android:layout_alignRight, andro id:layout_alignTop. Позиціонування відносно контейнеру: android:layout_alignParentBottom, android:layout_ alignParentLeft, android:layout_alignParentRight, android:layout_alignParentTop, android:layout_ centerHorizontal, android:layout_centerInParent, android:layout_centerVertical.

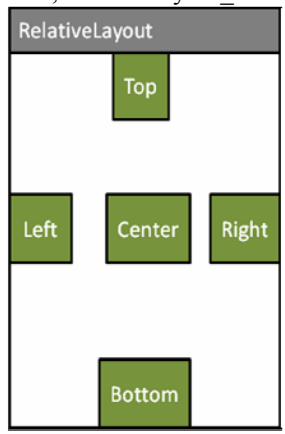

Рисунок Б.5 – Компоновка RelativeLayout

Б.3 Базові віджети

Віджет – це об’єкт View, який служить інтерфейсом для взаємодії з користувачем. Тобто відмет – це елемент керування. Android забезпечує набор готових віджетів, таких як кнопки, текстові поля, перемикачі, полоси прокрутки, індикатори, слайдери, адаптери даних та ін.

Для кожного віджета потрібно визначити його атрибути. Основними атрибутами є наступні: **id, layout_width, layout_height, text, gravity.**

Ідентифікатор елемента є унікальним та задається у вигляді android:id=«@+id/view1», де view1 – унікальний id елемента; символ «@» означає, що синтаксичний аналізатор XML повинен розгорнути дану строку і визначити цей вираз як ресурс ідентифікатора; символ «+»

означає, що це нове ім’я ресурсу, яке потрібно бути створено і додано у ресурси в файл R.java, який середовище Android автоматично генерує.

Ширина елемента задається у вигляді android:width=«wrap_content» (стандартний розмір) чи android:width=«match_parent» (розмір відповідає батьківському елементу, тобто якщо елемент-батько займає весь простір екрану за шириною, то і у дочірнього елемента ширина буде займати ширину екрана).

Висота елемента android:width також може приймати значення «wrap_content» чи «match_parent» в залежності від потрібного застосування.

Текст елемента задається у вигляді android:text=«something» чи android:text= «@string/res_str», де res_str – ім’я ресурсу (ресурс можна задати в каталогу /res/values).

Розташування елементу задається за допомогою android:gravity (*«left», «right», «center»* та ін.).

Б.3.1 Віджет TextView

TextView застосовується для відображення тексту без можливості його редагувати користувачем, крім того він служить для відображення текстових даних в контейнерних відметах для відображення списків (рис. Б.6). Від TextView успадковуються інші елементи управління, на яких повинен бути відображений текст (кнопки, прапорці, перемикачі). У елемента є багато атрибутів для роботи з текстом, наприклад:

– android:textSize (розмір в px, dp, sp, in, pt, mm);

– android:textStyle («normal», «bold», «italic»);

– android:textColor (колір тексту в форматі #RRGGBB чи #AARRGGBB);

– android:autoLink (створення посилання зі значеннями «web», «phone», «email», «map», «all», «none»).

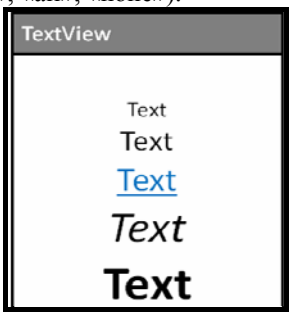

Рисунок Б.6 – Використання віджету TextView

Атрибути тексту можна задати програмно (в файлі *.java):

```
// id елемента T1
TextView T1=(TextView) findViewById(R.id.T1);
// встановлення тексту
T1.setText(«something…»);
```

Б.3.1 Віджет EditText

EditText застосовується для відображення тексту з можливістю його редагування (рис. Б.7). Основний метод класу – getText() з поверненим значенням EditText. За допомогою атрибуту android:hint=«Enter text…» встановлюється підказка для введення тексту. В класі визначено метод для виділення тесту з певних позицій – setSelection(). Більшість методів роботи з текстом та його форматування успадковані від базового класу TextView.

Рисунок Б.7 – Віджет EditText

Б.3.2 Полоси прокрутки

Полоси прокрутки в Android представлені віджетами **ScrollView** та **HorizontalScrollView**, які є контейнерними елементами і успадковуються від ViewGroup. Ці елементи – контейнери типу FrameLayout, це означає, що в них можна розмістити тільки одне дочірнє представлення. Цей дочірній елемент, в свою чергу, може бути контейнером зі складною ієрархією об'єктів. В якості дочірнього елементу для полос прокрутки зазвичай використовують LinearLayout з вертикальною чи горизонтальною орієнтацією елементів.

Віджет ScrollView підтримує тільки вертикальну прокрутку, тому для створення вертикальної та горизонтальної прокруток використовують обидва віджета. Звичайно ScrollView використовують в якості кореневого елементу, а HorizontalScrollView – дочірнього.

```
// файл компоновки main.xml
<ScrollView
android:id=«@+id/s_ver»
android:layout_width=«wrap_content»
android:layout_height=«wrap_content»>
<HorizontalScrollView
android:id=«@+id/s_hor»
android:layout_width=«match_parent»
android:layout_height=«match_parent»>
// елемент (наприклад, TextView)
<TextView
```

```
android:id=«@+id/t»
android:layout_width=«wrap_content»
android:layout_height=«wrap_content»
android:isScrollContainer=«true» />
</HorizontalScrollView>
</ScrollView>
// файл project.java
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class Project extends Activity{
public void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
TextView t=(TextView)findViewById(R.id.t);
t.setText(«…»); //ввести чи завантажити
}      }
```

Б.3.3 Відображення графіки

Відображення графіки виконується за допомогою віджету **ImageView**. Класс ImageView має змогу завантажувати зображення з різних джерел, таких як ресурси програмного застосунку або зовнішні файли. В цьому класі існує декілька методів завантаження зображення (в файлі проекту *.java):

– setImageResource(int resId) – завантажує зображення за його ідентифікатором ресурсу;

– setImageURI(Uri im_uri) – завантажує зображення за його URI;

– seyImageBitmap(Bitmap bitmap) – завантажує растрове зображення.

Крім того ImageView визначені методи для установки розміру зображення – setMaxHeight(), setMaxWidth(), getMinimumHeight(), getMinimumWidth(), а також його масштабування – getScaleType(), setScaleType().

Для завантаження зображення в XML-файлі компоновки використовується атрибут **android:scr**. Ресурси зображення частіше за все поміщають в /res/drawable/ проекту.

```
//1 вар. (завантаження pic1 в *.xml)
<ImageView
android:id=«@+id=image1»
android:layout_width=«wrap_content»
android:layout_height=«wrap_content»
```

```
android:scr=«@drawable/pic1»>
//2 вар.(об'ява в *.xml,завантаження pic2 в *.java)
//файл *.xml
<ImageView
android:id=«@+id=image2»
android:layout_width=«wrap_content»
android:layout_height=«wrap_content»>
//файл *.java
import android.widget.ImageView;
…
public class Progect extends Activity{
…
final ImageView image2 =(ImageView) findViewById (R.id.image2);
image2.setImageResource(R.drawable.pic2);
…
}
```

Б.4 Обробка подій. Кнопки та прапорці

Після додавання віджетів в інтерфейс користувача, потрібно організувати взаємодію віджетів з користувачем. Для цього необхідно визначити обробник подій та зареєструвати його для даного елементу.

Клас View містить в собі колекцію вкладених інтерфейсів, які мають назву **On…Listener()**, в кожному з яких об'явлено єдиний абстрактний метод. Цей метод необхідно перевизначити в вашому класі. Його буде визивати система Android, коли з екземпляром View, до якого було підключено слухача подій, стане взаємодіяти з користувачем.

Клас View містить наступні інтерфейси:

- OnClickListener;
- OnLongListener;
- OnFocusChangeListener;
- OnKeyListener;
- OnTouchListener;
- OnCreateContextMenuListener.

Якщо потрібно, щоб віджет отримав повідомлення про виконану дію користувачем, потрібно в класі вікна (Activity) реалізувати інтерфейс обробника подій (наприклад, OnClickListener) та визначити метод його зворотного виклику (наприклад, onClick()), де буде міститися код обробки події, та зареєструвати слухач події за допомогою, наприклад, setOnClickListener.

Б.4.1 Клас Button

Клас Button (кнопка) – найчастіше використовуваний елемент керування (рис. Б.8). Зазвичай кнопка потребує написання коду обробки події натиснення onClick.

Рисунок Б.8 – Віджет Button

```
// файл *.xml
…
<Button
android:id=«@+id/b1»
android:height=«wrap_content»
android:width=«match_parent»
android:text=«Button 1»>
// 1 вар. обробки натиснення кнопки (файл *.java)
…
import android.widget.*;
import android.view.View;
public class Project extends Activity{
public void onCreate(Bundle state){
    super.onCreate(state);
    setContentView(R.layout.main);
    final Button b1 = (Button) findViewById(R.id.b1);
    final Button b2 = (Button) findViewById(R.id.b2);
    b1.setOnClickListener(new View.OnClickListener(){
    public void onClick(View v){
    // дії після натиснення кнопки 1
    }
    });
    b2.setOnClickListener(new View.OnClickListener(){
    public void onClick(View v){
    // дії після натиснення кнопки 2
    }
    });
    }
    }
    АБО
    // 2 вар. обробки натиснення кнопки (файл *.java)
    …
    import android.widget.*;
    import android.view.View;
    public class Project extends Activity{
    public void onCreate(Bundle state){
```

```
        super.onCreate(state);
        setContentView(R.layout.main);
        final Button b1 = (Button) findViewById(R.id.b1);
        final Button b2 = (Button) findViewById(R.id.b2);
        b1.setOnClickListener(b1_click);
        b2.setOnClickListener(b2_click);
        }
public OnClickListener b1_click=new OnClickListener(){
        public void onClick(View v){
        // дії після натиснення кнопки 1
        }
};
public OnClickListener b2_click=new OnClickListener(){
        public void onClick(View v){
        // дії після натиснення кнопки 2
           }
};
} АБО
// 3 вар. обробки натиснення кнопки (файл *.java)
// найліпший в перелічених варіантів
…
import android.widget.*;
import android.view.View;
import android.view.View.OnClickListener;
public class Project extends Activity implements OnClickListeber{
public void onCreate(Bundle state){super.onCreate(state);
        setContentView(R.layout.main);
        final Button b1 = (Button) findViewById(R.id.b1);
        final Button b2 = (Button) findViewById(R.id.b2);
        b1.setOnClickListener(this);
        b2.setOnClickListener(this);
        }
public void OnClick (View v){
        swith(v.getId()){
        case R.id.b1:
        // дії після натиснення кнопки 1
        break;
        case R.id.b2:
        // дії після натиснення кнопки 2
        break;
           }
}
}
```

Б.4.2 Клас CheckBox

Елемент CheckBox (прапорець) – це перемикач з двома станами (рис. Б.9). Для програмного відстеження зміни стану елемента необхідно реалізувати інтерфейс CompoundButton. OnCheckedChangeListener.

Рисунок Б.9 – Віджет CheckBox

```
// файл *.xml
…
<CheckBox
android:id=«@+id/cb1»
      android:height=«wrap_content»
      android:width=«match_parent»
      android:text=«CheckBox 1»>
// файл *.java
…
import android.widget.CheckBox;
import android.widget.CompoundButton;
public class Project extends Activity implements
CompoundButton.OnCheckedChangeListener{
public void onCreate(Bundle state){
      super.onCreate(state);
      setContentView(R.layout.main);
      final CheckBox cb1 = (CheckBox) findViewById (R.id.cb1);
      final CheckBox cb2 = (CheckBox) findViewById (R.id.cb2);
      cb1.setOnCheckedChangeListener(this);
      cb2.setOnCheckedChangeListener(this);
}
public void OnCheckedChange (CompoundButton v, Boolean isChecked){
      swith(v.getId()){
      case R.id.cb1:
      if(isChecked){// дії вибору CheckBox 1}
      break;
      case R.id.cb2:
      if(isChecked){// дії вибору CheckBox 2}
      break;
         …
} } }
```

Б.4.3 Клас RadioButton

Віджети RadioButton (перемикачі) зазвичай використовують в складі групи контейнеру RadioGroup. Контейнер RadioGroup успадковується від ViewGroup і може бути використаний в якості кореневого елементу компоновки вікна, якщо на екрані є тільки група перемикачів, або в якості вкладеного в інший контейнер, наприклад в LinearLayout. Перемикачі дають змогу користувачу обрати тільки один із запропонованих варіантів. Основний метод зміни стану – toggle(), який інвертує стан перемикача. Крім того, від базового класу успадковуються інші методи, наприклад, isChecked() та setChecked().

```
//файл *.xml
<RadioGroup
Android:orientation=«vertical»
Android:layout_width=«match_parent»
Android:layout_height=«match_parent»>
<RadionButton
Android:id=«@+id/r1»
Android:layout_width=«wrap_content»
Android:layout_height=«wrap_content»
Android:text=«Mode 1»>
…
</RadioGroup>
//файл *.java
…
import android.widget.*;
import android.view.View;
import android.view.View.OnClickListener;
public class Project extends Activity implements OnClickListeber{
public void onCreate(Bundle state){
super.onCreate(state);
setContentView(R.layout.main);
final RadioButton r1 = (RadioButton) findViewById(R.id.r1);
final RadioButton r2 = (RadioButton) findViewById(R.id.r2);
final RadioButton r3 = (RadioButton) findViewById(R.id.r3);
r1.setOnClickListener(this);
r2.setOnClickListener(this);
r3.setOnClickListener(this);
}
public void OnClick (View v){
swith(v.getId()){
case R.id.r1:
```

```
        // дії після натиснення кнопки 1
        break;
        …
} }}
```

Завдання на роботу

Створіть Android-проект простого текстового редактору з кнопками зміни розміру шрифту, вибору одного з типів шрифту (за допомогою RadioGroup) та вертикальної полоси прокрутки. В якості поля введення тексту використати EditText. В якості компоновки рекомендовано використати вкладені LinearLayout. Зразок текстового редактору показано на рисунку Б.10.

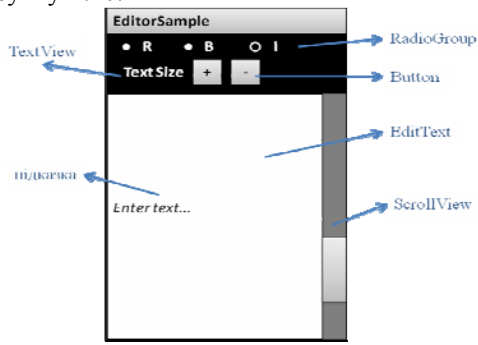

Рисунок Б.10 – Простий текстовий редактор

Контрольні питання

1. Компоновка елементів управління на Android.
2. Типи компоновок. Їх переваги та недоліки.
3. Поняття Activity та View.
4. Перелік основних атрибутів для віджетів.
5. Як створити новий ідентифікатор елементу?
6. Способи встановлення тексту віджету TextView.
7. Встановлення полос прокрутки.
8. Методи завантаження зображення з ImageViw.
9. Механізм обробки подій Android-застосунку.
10. Застосування віджету Button.
11. Застосування віджету CheckBox.
12. Застосування віджету RadioButton.

Додаток В ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА СПИСКИ

Мета роботи: ознайомитися із віджетами-списками та адаптерами даних.

Короткі теоретичні відомості

В.1 Віджети-списки та прив'язка даних

Для відображення даних у віджетах застосовуються адаптери, які призначені для з'єднання списку даних і відображає ці дані віджета. Самим простим адаптером для використання при з'єднанні даних є шаблонний клас **ArrayAdapter<T>**, наприклад:

String[] db = {«s1», «s2», «s3»};

ArrayAdapter<String> adapter = new ArrayAdapter <String>(this, android.R.layout.simple_list_item_1, db);

Конструктор класу ArrayAdapter приймає наступні параметри:

– об'єкт Context – екземпляр класу, який реалізує Activity;

– ідентифікатор ресурсу застосунку, вбудовані системні ідентифікатори ресурсу – константи, які визначені в класі android.R.layout (simple_list_item, simple_spinner_ dropdown_item, simple_gallery_item, simple_list_item_ checked…);

– масив чи список типу List<T> об'єктів для відображення у віджеті.

В.1.1 Віджет AutoCompleteTextView

Віджет AutoCompleteTextView – це текстове поле з авто заповненням і можливістю редагуванням введеного тексту. Віджет є підкласом EditText, він дозволяє використовувати всі можливості редагування тексту. Віджет має властивість вказування мінімального числа символів, які повинен ввести користувач, для роботи функції автозаповнення списку – android:completionThreshold.

Для в'язання віджету з даним необхідно встановити адаптер через setAdapter().

// файл *.xml

<LinearLayout xmlns:android = «http://schemas.android. com/apk/res/android»
android:layout_width=«match_parent»
android:layout_height=«wrap_content»
android:orientation=«vertical» >
<TextView
android:id=«@+id/t1»
android:layout_width=«match_parent»

```
android:layout_height=«wrap_content»
android:text=«Choice:» />
<Button
android:id=«@+id/b1»
android:layout_width=«wrap_content»
android:layout_height=«wrap_content»
android:text=«OK» />
<AutoCompleteTextView
android:id=«@+id/mylist»
android:layout_width=«match_parent»
android:layout_height=«wrap_content»
android:completionThreshold=«3» />
</LinearLayout>
// файл *.java
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;
import android.text.*;
public class MainActivity extends Activity implements TextWatcher,
OnClickListener{
private TextView t1;
private AutoCompleteTextView mylist;
private String[] db = {«Jacob Anderson», «Joseph Godwin», «Joshua
Harrison», «Emma Lawson»};
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
t1=(TextView)findViewById(R.id.t1);
mylist = (AutoCompleteTextView) findViewById(R.id.mylist);
mylist.addTextChangedListener(this);
mylist.setAdapter(new ArrayAdapter<String> (this,
android.R.layout.simple_dropdown_item_1line, db));

Button b1 = (Button) findViewById(R.id.b1);
b1.setOnClickListener(this);
}
public void afterTextChanged(Editable s) {
// TODO Auto-generated method stub
}
```

```
public void beforeTextChanged(CharSequence s, int start, int count, int after)
{
// TODO Auto-generated method stub
}
public void onTextChanged(CharSequence s, int start, int before, int count) {
// TODO Auto-generated method stub
}
public void onClick(View arg0) {
// TODO Auto-generated method stub
if (arg0.getId()==R.id.b1) t1.setText(mylist.getText());
}
}
```

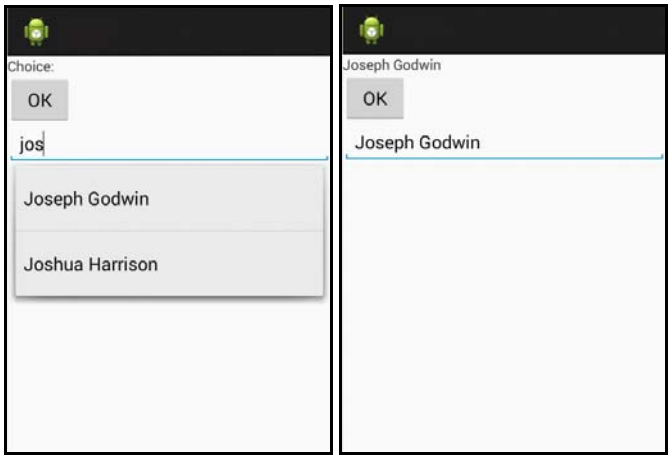

Рисунок В.1 – Застосування віджету AutoCompleteTextView

В.1.2 Віджет MultiAutoCompleteTextView

Віджет MultiAutoCompleteTextView – це текстове поле з авто заповненням і можливістю редагуванням введеного тексту, який розширює функціональність AutoCompleteTextView. Віджет показує авто заповнення для кожної підстроки тексту, яка розділена знаком пунктуації. Роздільник задається явно викликом метода setTokenizer():

```
MultiAutoCompleteTextView mylist = (MultiAutoCompleteTextView)
findViewById(R.id.mylist);
mylist.addTextChangedListener(this);
mylist.setAdapter(new ArrayAdapter<String> (this,
android.R.layout.simple_dropdown_item_1line, db));
//встановлення роздільника
```

```
Mylist.setTokenizer (new MultiAutoCompleteTextView.
CommaTokenizer());
```

B.1.3 Віджет ListView

Віджет має вигляд вертикального списку із прокруткою. Зв'язані зі списком дані ListView отримує від об'єкту ListAdapter. На відміну від AutoCompleteTextView в якості базового класу використовується **ListActivity** замість Activity. Для з'єднання об'єкта ListActivity із даними необхідно розробити клас, який реалізує інтерфейс **ListAdapter**. Android забезпечує два стандартних адаптера списку: SimpleAdapter (статичне з'єднання даних невеликого обсягу), SimpleCursorAdapter (великий масив даних).

```
// файл *.xml
<LinearLayout xmlns:android = «http://schemas.android.
com/apk/res/android»
android:layout_width = «match_parent»
android:layout_height=«match_parent»
android:orientation=«vertical» >
<TextView
android:id=«@+id/t1»
android:layout_width=«match_parent»
android:layout_height=«wrap_content»/>
<ListView
android:id=«@android:id/list»
android:layout_width=«match_parent»
android:layout_height=«wrap_content» />
</LinearLayout>
// файл *.java
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.app.ListActivity;
import android.widget.*;
import android.view.View;
public class MainActivity extends ListActivity {
private String[] db = {«Jacob Anderson», «Joseph Godwin», «Joshua
Harrison», «Emma Lawson»};
private TextView t1;
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
```

```
t1=(TextView)findViewById(R.id.t1);
setListAdapter(new ArrayAdapter<String> (this,
android.R.layout.simple_list_item_1, db));
}
public void onListItemClick(ListView p, View v, int position, long id){
t1.setText(db[position]);
}
}
```

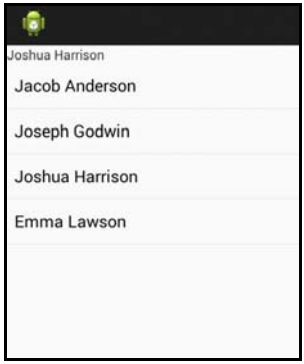

Рисунок В.2 – Застосування віджету ListView

В.1.4 Створення списку із заданою компоновкою

Крім використання стандартних віджетів-списків можна створити список, визначив власну компоновку однієї строки цього списку. Такий підхід використовується для створення таблиць. Для з'єднання табличних даних використовуються класи SimpleAdapter та SimpleCursorAdapter. Конструктор SimpleAdapter має вигляд:

SimpleAdapter(this, list, R.layout.main, new String[] {items.item1, items.item2, …}, new int[] {R.id.item1, R.id.item2, …})

Приклад. Створимо список-довідник, в якому визначено ПІБ та номер телефону. В файлі **main.xml** опишемо два віджета TextView з іменами name та phone (компоновка горизонтальна). Для зберігання строки, яка представляє контакт, створимо окремий файл **Items.java**. Створення списку заданої компоновки представлена на рис. В.3.

```
// Items.java
package com.samples.ui.lab;
import java.util.HashMap;
public class Items extends HashMap<String, String>{
private static final long serialVersionUID=1L;
public static final String NAME = «name»;
```

```
public static final String PHONE = «phone»;
public Items(String name, String phone){
super();
super.put(NAME, name);
super.put(PHONE, phone);
}}
// lab.java
package com.samples.ui.lab;
import java.util.ArrayList;
import android.app.ListActivity;
import android.os.Bundle;
import android.widget.*;
public class lab extends ListActivity{
public void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
ArrayList<Items> list = new ArrayList<Items>();
// заповнення контактів
list.add(new Items(«Jacob Anderson», «420044»));
list.add(new Items(«Joseph Godwin», «336754»));
list.add(new Items(«Joshua Harison», «325588»));
list.add(new Items(«Emma Lawson», «425522»));
// адаптер даних
ListAdapter a = new SimpleAdapter(this, list, R.layout.main, new String[]
{Items.NAME, Items.PHONE}, new int[] {R.id.name, R.id.phone});
setListAdapter(a);
}}
```

Рисунок В.3 – Створення списку заданої компоновки

В.1.5 Віджет Spinner

Віджет Spinner – це аналог ComboBox для Android. Віджет має два режиму android:spinnerMode: розгортаємого списку («dropdown») та діалогу («dialog»). Застосовується інтерфейс AdapterView.OnItemSelectedListener.

Приклад. В файлі проекту main.xml визначимо елементи TextView (id=t1) та Spinner (id=s1). Віджет Spinner (режим діалогу) представлений на рисунку В.4.

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
public class lab extends Activity implements AdapterView.OnItemSelectedListener{
private TextView t1;
private final String[] db={…};
public void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
t1=(TextView)findViewById(R.id.t1);
final Spinner s1=(Spinner) findViewById(R.id.s1);
s1.setOnItemSelectedListener(this);
ArrayAdapter<String> a = new ArrayAdapter<String> (this, android.R.layout.simple_spinner_item, db);
a.setDropDownViewResource(android.R.layout.simple_
spinner_dropdown_item);
s1.setAdapter(a);
}
public void onItemSelected(AdapterView<?> p, View v, int position, long id){
t1.setText(db[position]);
}
public void onNothingSelected(AdapterView<?> p){
t1.setText(««);
}
}
```

Рисунок В.4 – Віджет Spinner (режим діалогу)

3.1.6 Відображення графіки в списках

Для відображення графіки в списках існують віджети **GridView** (рис. В.5)**, Gallery** (рис. В.6)**, SlidingDrawer**.

Для застосування віджетів необхідно наслідувати клас BaseAdapter. В класі адаптера масив даних повинен містити ідентифікатори графічних ресурсів, які будуть розміщуватися в каталозі res/drawable/. Наприклад:

private static final Integer[] im = {R.drawable.photo1, R.drawable.photo2,…};

Приклад створення класу адаптера **MyImageAdapter.java**.

```
// MyImageAdapter.java
import android.view.View;
import android.content.Context;
import android.view.ViewGroup;
import android.widget.*;
public class MyImageAdapter extends BaseAdapter{
private Context c;
private static final Integer[] im = {R.drawable.p1, R.drawable.p2, R.drawable.p3, R.drawable.p4, R.drawable.p5, R.drawable.p6};
public MyImageAdapter(Context cnt){c = cnt;}
public int getCount() {return im.length; }
public Object getItem(int arg0) {return im[arg0];}
public long getItemId(int pos) {return im[pos];}
public View getView(int position, View convertView, ViewGroup parent) {
ImageView v;
if(convertView == null){
v = new ImageView(c);
v.setLayoutParams(new GridView.LayoutParams(85,85));
v.setScaleType(ImageView.ScaleType.CENTER_CROP);
v.setPadding(2, 2, 2, 2);
}
else v=(ImageView) convertView;
v.setImageResource(im[position]);
return v;
}
}
```

1. Віджет **GridView**.

```
// main.xml
<TextView
android:id=«@+id/t1» …/>
<GridView
```

```
android:id=«@+id/gv»
android:layout_width=«match_parent»
android:layout_height=«match_parent»
android:numColumns=«auto_fit»
android:columnWidth=«100px»
android:stretchMode=«columnWidth»
android:gravity=«center»
android:verticalSpacing=«35px»
android:horizontalSpacing=«5px» />
…
//main.java
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.*;
public class Main extends Activity
implements AdapterView.OnItemClickListener{
private TextView t1;
private MyImageAdapter ia;
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
t1 = (TextView) findViewById(R.id.t1);
final GridView gv = (GridView) findViewById(R.id.gv);
ia = new MyImageAdapter(getApplicationContext());
gv.setAdapter(ia);
gv.setOnItemClickListener(this);
}
public void onItemClick(AdapterView<?> a0, View a1, int a2, long a3) {
// TODO Auto-generated method stub
t1.setText(«Select: «+a2+» from «+a0.getCount()+»; id=«+ia.getItem(a2));
}}
```

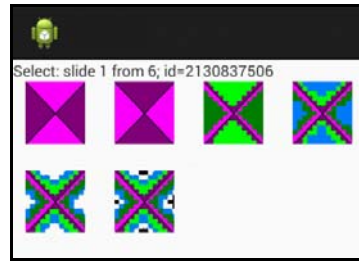

Рисунок В.5 – Віджет GridView

2. Віджет **Gallery.**

Для застосування даного віджету відкоригуємо файл MyImageAdapter.java:

```
public View getView(int position, View convertView, ViewGroup parent) {
ImageView v;
v = new ImageView(c);
v.setImageResource(im[position]);
v.setPadding(20, 20, 20, 20);
v.setLayoutParams(new Gallery.LayoutParams(140, 140));
v.setScaleType(ImageView.ScaleType.FIT_XY);
return v;
}
```

А також у файлі Main.java змінемо строку створення об'єкту:

```
final Gallery gv = (Gallery) findViewById(R.id.gv);
```

Рисунок В.6 – Віджет Gallery

Завдання на роботу

Створіть Android-проект, який містить в собі список типу ListView. Кожен елемент списку містить текст та зображення. Результат обраного користувачем елементу списку відображається в окремому текстовому полі.

Примітка. Для розробки списку складної компоновки необхідно створити файли *.xml головної форми та окремої строки списку, що складається з текстового поля та зображення (наприклад, activity_main.xml та item.xml); створити декілька *.java файлів: 1) клас, який містить типи списку (наприклад, клас myobject, що містить строку string str та ідентифікатор зображення int image); 2) клас адаптеру списку (наприклад, ItemAdapter.java) з наступним конструктором:

```
public ItemAdapter(Context _c, ArrayList<myobject> _ob) {
c = _c;
ob = _ob;
layoutinf = (LayoutInflater) c.getSystemService (Context.LAYOUT_I
NFLATER_SERVICE);
}
```

та методом getView, який ініціює ресурси списку:

```
public View getView(int pos, View cv, ViewGroup parent) {
View v = cv;
if (v=null){
v=layoutinf.inflate(R.layout.item,parent,false);
}
myobject obs = getObject(pos);
((TextView) v.findViewById(R.id.t)).setText(obs.str);
((ImageView)v.findViewById(R.id.im)).setImageResource(obs.image);
return v;
}
```

3) головний клас, який встановлює адаптер списку, його ресурси та виконує операцію вибору елементу списку за допомогою методу setOnItemClickListener (android.widget.AdapterView.*).

Контрольні питання

1. Як прив’язати (встановити) дані до списку?
2. Як встановити адаптер даних?
3. Які параметри приймає адаптер ArrayAdapter?
4. Об’єкт Context.
5. Об’єкт LayoutInflater.
6. Віджет AutoCompleteTextView.
7. Віджет MultiAutoCompleteTextView.
8. Віджет ListView.
9. Віджет Spinner.
10. Відображення графіки у списках.

Додаток Д ОРГАНІЗАЦІЯ ОБМІНУ ДАНИХ У ГІБРИДНИХ МОБІЛЬНИХ ЗАСТОСУНКАХ

Мета роботи: ознайомитися з організацією процесів та взаємодією компонентів в Android-застосунках за допомогою об'єктів Intent. Навчитися передавати дані у викликаний Activity та зворотне отримання інформації із викликаного Activity з використанням extra-параметрів об'єктів Intent.

Короткі теоретичні відомості

Д.1 Процеси в системі Android

Система Android запускає процес, який містить єдиний потік для виконання, коли хоча б один з компонентів застосунку (або весь застосунок) буде запитано. Всі компоненти застосунку виконуються в цьому процесі та потоку за замовчуванням.

Всі компоненти ініціюються в основному потоку процесу. Окремі потоки для кожного екземпляру не створюються. Отже, всі методи зворотного виклику, які визначені в компоненті та викликані системою, завжди працюють в основному потоку процесу. Виходячи з цього, компонент не повинен виконувати в методах зворотного виклику довготривалі операції (завантаження файлів з мережі, цикли обчислення і т.д.) або блокувати системний виклик, бо це блокує будь-які компоненти в цьому процесі. Для таких операції породжують окремі потоки.

Система Android може завершити процес у випадку нестачі пам'яті або якщо пам'ять затребувана більш важливими процесами. Прикладні компоненти, які виконуються в таких процесах, будуть знищені. Процес буде пере запущено для компонентів у випадку їх повторного виклику.

При виборі процесу для знищення Android оцінює відносну важливість цього процесу з точки зору користувача. Процеси з низькою важливістю знищуються в першу чергу. Існує п'ять рівнів в ієрархії важливості. Наступний список представляє їх в порядку зменшення важливості:

– *активний процес* (Foreground Process). Процес вважається активним, якщо з ним взаємодіє користувач, процес має об'єкт Service чи BroadcastReceiver при виконанні методу зворотного виклику для цього об'єкту;

– *видимий процес* (Visible Process) – компонент із цього процесу може викликатися користувачем. Це може бути процес Activity, який не знаходиться в фокусі, але ще видимий користувачу;

– *сервісний процес* (Service Process) – процес, в якому виконується Service та який не відноситься до жодної з перелічених вище категорій;

– *фоновий процес* (Background Process) – процес, в якому виконується Activity, яку в теперішній час не бачить користувач;

– *пустий процес* (Empty Process) – не містить жодних активних компонентів. Цей процес зберігається тільки як кеш, для того щоб зменшити час запуску виклику компонента.

Д.2 Стани Activity

Activity може знаходитися в трьох станах:

– *активний* (active) – Activity знаходиться на передньому плані екрана мобільного пристрою;

– *призупинений* (paused) – Activity не має фокусу, але все ще видимо користувачу;

– *зупинений* (stopped) – Activity повністю закрито іншими

Activity при переході з одного стану в інший отримує повідомлення через захищені методи: *onCreate()* – при створенні Activity, *onRestart()* – при відновленні роботи Activity, *onStart()* – викликається безпосередньо при видимості Activity, *onResume()* – викликається безпосередньо при початку взаємодії користувача з Activity, *onPause()* – викликається, коли система намагається запустити інший Activity, *onStop()* – викликається, коли Activity стає невидимим, *onDestroy()* – при знищенні Activity. Ці методи можна реалізувати в класі Activity, для того щоб виконати певні дії при зміні стану даного Activity. Наприклад: protected void onPause(){super.onPause(); … }

Д.3 Запуск Activity з використанням об'єктів Intent

Компоненти Android-застосунку, в тому числі Activity, запускаються через об'єкти **Intent**. Це засіб пізнього зв'язування під час виконування між компонентами одного чи декількох застосунків. В кожному випадку система Android знаходить відповідний Activity, для того щоб відповісти на Intent та ініціалізує його у випадку необхідності. Об'єкти Intent можна розділити на дві групи:

– *явний Intent* – визначає цільовий компонент за ім'ям;

– *неявний Intent* – не називає адресата.

Явний Intent використовують для повідомлень в межах застосунку, наприклад, коли один Activity запускає інший Activity із цього застосунку.

Неявний Intent використовують для запуску компонентів інших застосунків. В файлі маніфесту застосунку зазвичай декларується фільтр Intent.

В системі Android всі Activity зберігаються в стеку. Коли один Activity запускає інший, новий Activity розміщається в стек і стає активним Activity. Під час роботи користувача, поточний Activity виштовхується із стеку та його заміщує попередній Activity, який знов відображується на екрані.

Д.4 Запуск Activity за допомоги явного об'єкту Intent

Об'єкт Intent являється структурою даних, яка містить абстрактний опис виконуваної операції. Для того щоб викликати інший Activity, в об'єкт Intent треба передати ім'я цього Activity. Ім'я встановлюється методами *setComponent()*, *setClass()* або *setClassName()*. Для запуску Activity, об'єкт Intent передають в метод *Context.startActivity()*. Цей метод приймає єдиний параметр – об'єкт Intent, який описує Activity, що запускається. Наприклад, викликати Activity з ім'ям NewActivity в коді програми можна наступним чином:

```
Intent intent = new Intent();
// встановлюємо ім'я викликаємого компоненту
intent.setClass(getApplicationContext(),
NewActivity.class);
// запускаємо компонент
startActivity(intent);
```

Для того щоб застосунок мав змогу бачити другий Activity, необхідно вказати його в файлі маніфесту застосунку. Для цього необхідно вибрати в представленні **Project Explorer** файл **AndroidManifest.xml** та у відкритому редакторі маніфесту перейдіть на вкладку **Application**. На панелі **Application Nodes** натисніть кнопку **New** та у відкритому діалоговому вікні оберіть елемент **Activity**. На панелі **Application Nodes** буде додано новий вузол Activity. Якщо його виділити, праворуч відобразиться панель із заголовком **Attributes for Activity**. На цій панелі в строчці **Name** натисніть кнопку **Browse** та в відкритому вікні оберіть клас нової Activity (NewActivity). Після цих дій в файлі AndroidManifest.xml з'явиться застосунковий елемент <activity> з атрибутом android:name = «NewActivity».

Д.5 Виклик стандартних Activity для застосунку

Для виклику системних компонентів в класі Intent визначено набір констант Standard Activity Actions, що визначають дії запуску стандартних Activity. Наприклад, дія ACTION_DIAL ініціалізує звернення по телефону у вигляді вікна набору телефонного номера, а в класах MediaStore та MediaStore.Audio.Media визначені константи дії для запуску вікон з медіаплеєром, пошуку музики та запису з мікрофону; в класах DownloadManager – завантаження файлів через

Інтернет, RingtoneManager – для запуску вікна, що встановлює режим дзвінка та ін. Викликати стандартне Activity можна наступним чином:

```
Intent intent = new Intent(«Intent.ACTION_DIAL»);
startActivity(intent);
```

Д.6 Обмін даними між Activity

Інколи є потреба повернути результат Activity, коли він закривається. Для того щоб запустити Activity та отримати результат його виконання, необхідно викликати метод startActivityForResult(Intent, int) зі другим параметром, який ідентифікує запит. Результат повертається через метод onActivityResult(int, int, Intent), що визначений в батьківському Activity.

Крім того, можна передавати застосунку параметри (extra-параметри) у Activity, що викликається. Це пари ключ-значення для інформації, яку треба надати викликаємому компоненту. Об'єкт Intent має ряд методів **put…()** для вставки різного типу додаткових даних та аналогічного набору методів **get…()** для читання даних. Extra-параметри встановлюються та читаються як об'єкти класу Bundle із використанням методів **putExtras()** і **getExtras()**.

Наприклад, запустити Activity с ім'ям класу EditContactActivity та передати йому два додаткових параметра із головного вікна зі списком контактів ContactListActivity можна наступним чином:

```
// ідентифікатор запиту
private static final int IDM_EDIT = 102;
private long mId = –1;
…
// в класі ContactItem визначений список контактів з
// полями NAME та PHONE, а також визначені методи
// getName(),getPhone(),setName(String),setPhone(String)
ArrayList<ContactItem> mList;
private ListAdapter mAdapter;
…
mList = new ArrayList<ContactItem> ();
…
mId=this.getSelectedItemId();
ContactItem mItem = mList.get((int)mId);
Intent intent = new Intent();
// додавання extra-параметрів
intent.putExtra(ContactItem.NAME, mItem.getName());
intent.putExtra(ContactItem.PHONE, mItem.getPhone());
```

```
// визначення класу запускає мого Activity
intent.setClass(this, EditContactActivity.class);
// виклик Activity
startActivityFotrResult(intent, IDM_EDIT);
```

Коли дочірній Activity закриється, в ньому можна викликати метод setResult(int), для того щоб повернути дані в батьківський Activity. Цей метод повертає код результату закриття Activity, який може бути стандартним результатом, що визначений константами

RESULT_CANCELE, RESULT_OK або RESULT_FIRST_USER.

```
private EditText mName;
private EditText mPhone;
…
Intent t = new Intent();
// вставляємо ім'я та телефон людини
t.putExtra(ContactItem.NAME,mName.getText().toString());
t.putExtra(ContactItem.PHONE,mPhone.getText().toString());
// повертаємо результат у викликающий Activity
setResult(RESULT_OK, t);
finish();
```

Крім того, дочірній Activity довільно повернути об'єкт Intent, який містить будь-які додаткові дані. Вся ця інформація в батьківському Activity з'явиться через метод зворотного виклику Activity.onActivityResult(), разом із ідентифікатором, який було передано в метод startActivityForResult() при виклику Activity:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){
super.onActivityResult(requestCode, resultCode, data);
if (resultCode == RESULT_OK){
Bundle extras = data.getExtras();
switch(requestCode){
case IDM_ADD:
mList.add(new ContactItem(
extras.getString(ContactItem.NAME),
extras.getString(ContactItem.PHONE)));
break;
case IDM_EDIT:
mList.set((int)mId, new ContactItem(
extras.getString(ContactItem.NAME),
extras.getString(ContactItem.PHONE)));
break;
}
```

…
}}

Треба зауважити, що для коректної взаємодії Activity, в файлі маніфесту необхідно додати окрім головного Activity інші Activity, додавши атрибути android:name,android:label:

```
<application …>
<activity android:name=«.ContactListActivity»
android:label=«@string/app_name»>
…
</activity>
<activity android:name=«.NewContactActivity»
android:label=«@string/title_add»>
</activity>
<activity android:name=«.EditContactActivity»
android:label=«@string/title_edit»>
</activity>
</application>
```

Д.7 Intent-фільтри та запуск завдань

Intent-фільтри декларують обмеження компонента по прийому неявних об'єктів Intent, які він має змогу обробляти. Якщо компонент не має жодних фільтрів, він може приймати тільки явні об'єкти Intent. Компонент з фільтрами може приймати як явні так і неявні Intent. В Intent декларуються тільки три складові об'єкту Intent: дія, дані та категорія. Наприклад, в будь-якому застосунку є головний Activity, який встановлюється як точка входу для завдання:

```
<activity
android:name=«.ContactListActivity»
android:label=«@string/app_name»>
<intent-filter>
<action android:name=«android.intent.action.MAIN» />
<category android:name=«android.intent.category.
LAUNCHER» />
</intent-filter>
</activity>
```

Для того щоб Intent запустив компонент, котрому належить фільтр, він повинен чітко пройти всі три тести. Проте, якщо у об'єкта є декілька Intent-фільтра, то у випадку не проходження одного з них, проходить перевірку на наступний Intent-фільтр.

Наприклад, створимо окремий застосунок для виклику попереднього застосунку ContactListActivity з прикладу, описаного у пункті 4.6, та зробимо зміни в файлі маніфесту:

```
<activity
android:name=«.ContactListActivity»
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=«com.samples.app.contact.
VIEW_CONTACTS» />
<category android:name=«android.intent.category.
DEFAULT» />
</intent-filter>
</activity>
```

В коді класу нового проекту буде створюватися об’єкт Intent з визначеною дією: startActivity(new Intent(«com.samples.app.contact.VIEW_ CONTACTS»));

Завдання на роботу

Створіть Android-проект, який реалізує механізм обміну даними за варіантом (таб. Д.1).

Таблиця Д.1 – Теми проектів мобільних застосунків

Номер варіанту	Тема проекту
1	Телефонна книга
2	Журнал обліку викладача
3	Кулінарна книга рецептів
4	Здоровий спосіб життя. Фітнес
5	Щоденник справ (записник)
6	Психологічне тестування
7	Медіа-програвач
8	Програма створення списку покупок
9	Тестування з англійської мови
10	Handmade

Контрольні питання

1. Пріоритети процесів в Android.
2. Як ініціюються в потоці методи одного компоненту?
3. Стани знаходження Activity.
4. Який ланцюг повідомлень захищених методів отримує Activity до взаємодії його з користувачем?
5. Для чого використовуються об’єкти Intent?
6. Види об’єктів Intent.
7. Які необхідно зробити зміни в файлі маніфесту для коректної роботи декількох Activity?
8. Механізм запуску нової Activity та передача йому батьківських параметрів.
9. Механізм повернення даних нової Activity до батьківської.
10. Intent-фільтри та запуск завдань.

Alexander Stepanenko, Ievgen Fedorchenko

Bases of programming applications on the base of different mobile operating systems and platform

The training manual is written in accordance with the curriculum for the training of specialists in higher education institutions studying in the field of «Software Engineering» on the basis of the course developed by the authors of the course on lectures on educational disciplines «Mobile Operating Systems» and «Development of Applications for Mobile Devices» for Higher educational institutions students using material that reflects current achievements in this field.

The manual consists of 4 chapters and appendices, each of which is sufficiently independent and completed, and at the same time they are all interrelated both in material development from theoretical positions to the applied aspects and in terms of methodology of presentation.

The first chapter discusses devices' construction, hardware platform and technology used in smartphones. The principles of mobile devices functioning are given, the computer programs used and their mobile analogues are considered.

The second section is dedicated to the most popular mobile operating systems and their characteristics. It describes their history of occurrence, a brief description is given as well.

The third section outlines the steps for developing a mobile multimedia application based on the most widely used android platform. The process of java 2 sdk se and sun one studio 4 installation and developed mobile applications debugging are described. The general description of java 2 me applications' functioning mechanisms is given. The user interface and classes necessary for its high-quality design are considered in-depth.

The fourth section provides information on developing applications for mobile platforms such as iphone, android and blackberry.

The following sections provide information on cross-platform development tools and mobile devices in computer science courses. Additionally, the section about mobile platforms as an object of scientific research is dedicated specially for students studying for a «master» degree.

The appendices provide practical implementation of the sections.

We believe that the manual can be recommended for publication.

Навчальне видання

СТЕПАНЕНКО Олександр Олексійович
ФЕДОРЧЕНКО Євген Миколайович

ОСНОВИ ПРОГРАМУВАННЯ ДОДАТКІВ НА БАЗІ РІЗНИХ МОБІЛЬНИХ ОПЕРАЦІЙНИХ СИСТЕМ ТА ПЛАТФОРМ

Навчальний посібник

Комп'ютерний набір *Степаненко О.О.*
Федорченко Є.М.
Дизайн обкладинки *Лобачова Л.В.*
Верстання *Гринь Д. В.*

Оригінал-макет підготовлено
в редакційно-видавничому відділі ЗНТУ

Підписано до друку 16.10.2018. Формат 60×84/16. Ум. друк. арк. 7,21.
Тираж 300 прим. Зам. № 1114.

Запорізький національний технічний університет
Україна, 69063, м. Запоріжжя, вул. Жуковського, 64
Тел.: (061) 769–82–96, 220–12–14

Свідоцтво суб'єкта видавничої справи ДК № 2394 від 27.12.2005.