Міністерство освіти та науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт з навчальної дисципліни «Кросплатформені мови програмування» для студентів денної та заочної форми навчання першого (бакалаврського) рівня вищої освіти галузі «Інформаційні технології».

ЗАТВЕРДЖЕНО
на засіданні кафедри кібербезпеки та програмного забезпечення, протокол № 1 від 26.08.2025 року

Кропивницький
2025

Кросплатформені мови програмування: методичні рекомендації до виконання лабораторних робіт для студентів денної та заочної форми навчання, першого (бакалаврського) рівня вищої освіти, галузі «Інформаційні технології» / М-во освіти і науки України, Центральноукр. нац. техн. ун-т; [уклад. О.В. Коваленко, А.С. Коваленко] – Кропивницький: ЦНТУ, 2025. – 37с.

Укладачі:

Коваленко О.В., докт. техн. наук, доц;

Коваленко А.С., к. т. н., доц.

Рецензенти: Смірнов О. А., докт. техн. наук, професор, завідувач кафедри;
Якименко Н.М., к. ф.-м. наук, доцент.

Android - операційна система для мобільних пристроїв: смартфонів, планшетних комп'ютерів. В даний час саме Android є самою широко використовуваною операційною системою для мобільних пристроїв.

Платформа Android об'єднує операційну систему, побудовану на основі ядра ОС Linux, проміжне програмне забезпечення і вбудовані мобільні додатки. Розробка та розвиток мобільної платформи Android виконується в рамках проекту AOSP (Android Open Source Project) під управлінням OHA (Open Handset Alliance), керує всім процесом пошуковий гігант Google.

Android підтримує фонове виконання завдань; надає багату бібліотеку елементів користувальницького інтерфейсу; підтримує 2D і 3D графіку, використовуючи OpenGL стандарт; підтримує доступ до файлової системи і вбудованою базі даних SQLite.

З точки зору архітектури, система Android представляє собою повний програмний стек, в якому можна виділити наступні рівні:

- **Базовий рівень (Linux Kernel)** - рівень абстракції між апаратним рівнем і програмним стеком;
- **Набір бібліотек і середовище виконання (Libraries & Android Runtime)** забезпечує найважливіший базовий функціонал для додатків, містить віртуальну машину Dalvik і базові бібліотеки Java необхідні для запуску Android додатків;
- **Рівень каркаса додатків (Application Framework)** забезпечує розробникам доступ до API, що надаються компонентами системи рівня бібліотек;
- **Рівень додатків (Applications)** - набір встановлених базових додатків.

Розглянемо компоненти платформи більш докладно.

У підставі компонентної ієрархії лежить ядро ОС Linux 2.6 (кілька урізане), служить проміжним рівнем між апаратним і програмним забезпеченням, забезпечує функціонування системи, надає системні служби ядра: управління пам'яттю, енергосистемою і процесами, забезпечення безпеки, робота з мережею і драйверами.

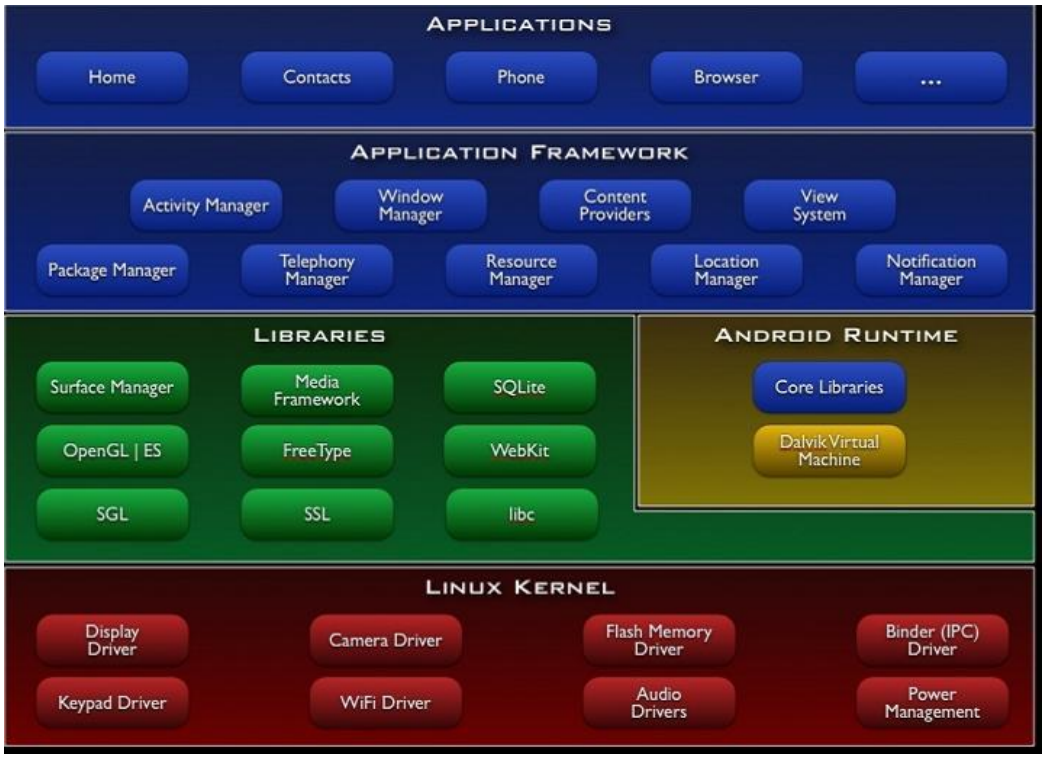


Рис.1. Архітектура Android

Рівнем вище розташовується набір бібліотек і середовище виконання. Бібліотеки реалізують такі функції:

- надають реалізовані алгоритми для вищих рівнів;
- забезпечує підтримку файлових форматів;
- здійснює кодування і декодування інформації (наприклад, мультимедійні кодеки);
- виконує отрисовку графіки і т.д.

Бібліотеки реалізовані на C / C ++ і скомпільовані під конкретне апаратне забезпечення пристрою, разом з яким вони і поставляються виробником в передвстановленому вигляді.

Розглянемо деякі бібліотеки:

Surface Manager	- Композитний менеджер вікон. Вступники команди отрисовки збираються в закадровий буфер, де вони накопичуються, складаючи якусь композицію, а потім виводяться на екран. Це дозволяє системі створювати цікаві безшовні ефекти, прозорість вікон і плавні переходи.

Media Framework	- Бібліотеки, реалізовані на базі PacketVideo OpenCORE. Використовуються для запису і відтворення аудіо та відео контенту, а також для виведення статичних зображень. Підтримуються формати: MPEG4, H.264, MP3, AAC, AMR, JPG і PNG.
SQLite	- Легковага і продуктивна реляційна СУБД, використовується в Android в якості основного движка для роботи з базами даних.
3D бібліотеки	- Використовуються для високооптимізовані отрисовки 3D-графіки, при можливості використовують апаратне прискорення. Бібліотеки реалізовані на основі API OpenGL | ES. OpenGL | ES (OpenGL for Embedded Systems) - підмножина графічного програмного інтерфейсу OpenGL, адаптоване для роботи на вбудованих системах.
FreeType	- Бібліотека для роботи з бітовими картами, для растеризації шрифтів і здійснення операцій над ними.
LibWebCore	- Бібліотеки браузерного движка WebKit, використовуваного також у відомих браузерах Google Chrome і Apple Safari.
SGL (Skia Graphics Engine)	- Відкритий движок для роботи з 2D-графікою. Графічна бібліотека є продуктом Google і часто використовується в інших програмах.
SSL	- Бібліотеки для підтримки однойменного криптографічного протоколу.
Libc	- Стандартна бібліотека мови C, а саме її BSD реалізація, налаштована для роботи на пристроях на базі Linux.

Середовище виконання включає в себе бібліотеки ядра, що забезпечують більшу частину низкоуровневой функціональності, доступної бібліотекам ядра мови Java, і віртуальну машину Dalvik, що дозволяє запускати додатки. Кожне додаток запускається в своєму примірнику віртуальної машини, тим самим забезпечується ізоляція працюючих додатків

від ОС і один від одного. Для виконання на віртуальній машині Dalvik Java - класи компілюються в виконувані файли з розширенням.dex за допомогою інструменту dx, що входить до складу Android SDK. DEX (Dalvik EXecutable) - формат виконуваних файлів для віртуальної машини Dalvik, оптимізований для використання мінімального обсягу пам'яті. При використанні IDE Eclipse і плагіна ADT (Android Development Tools) компіляція класів Java у формат.dex відбувається автоматично.

Архітектура Android Runtime така, що робота програм здійснюється строго в рамках оточення віртуальної машини, що дозволяє захистити ядро ОС від можливої шкоди з боку інших її складових. Тому код з помилками або шкідливе ПЗ не зможуть зіпсувати Android і пристрій на його базі, коли спрацюють.

На ще більш високому рівні розташовується каркас додатків (Application Framework), архітектура якого дозволяє будь-якому додатком використовувати вже реалізовані можливості інших програм, до яких дозволений доступ. До складу каркаса входять наступні компоненти:

- багатий і розширюваний набір уявлень (**Views**), який може бути використаний для створення візуальних компонентів додатків, наприклад, списків, текстових полів, таблиць, кнопок або навіть вбудованого web-браузера;
- контент-провайдери (**Content Providers**), керуючі даними, які одні додатки відкривають для інших, щоб ті могли їх використовувати для своєї роботи;
- менеджер ресурсів (**Resource Manager**), що забезпечує доступ до ресурсів без функціональності (що не несе коду), наприклад, до строкових даними, графіці, файлів і іншим;
- менеджер сповіщень (**Notification Manager**), що дозволяє додаткам відображати власні повідомлення для користувача в рядку стану;
- менеджер дій (**Activity Manager**), керуючий життєвими циклами додатків, що зберігає історію роботи з діями, що надає систему навігації по діям;

• менеджер місця розташування (**Location Manager**), що дозволяє додаткам періодично отримувати оновлені дані про поточний географічному положенні пристрою.

Application Framework надає в розпорядження додатків в ОС Android допоміжний функціонал, завдяки чому реалізується принцип багаторазового використання компонентів додатків і ОС. Природно, в рамках політики безпеки. І, нарешті, найвищий, найближча до користувача рівень додатків. Саме на цьому рівні користувач взаємодіє зі своїм пристроєм, керованим ОС Android. Тут представлений набір базових додатків, який предустановлен на ОС Android. Наприклад, браузер, поштовий клієнт, програма для відправки SMS, карти, календар, менеджер контактів та ін. Список інтегрованих програм може змінюватися в залежності від моделі пристрою і версії Android. До цього рівня також належать всі користувальницькі додатки.

Розробник зазвичай взаємодіє з двома верхніми рівнями архітектури Android для створення нових додатків. Бібліотеки, система виконання і ядро Linux приховані за каркасом додатків.

Повторне використання компонентів інших додатків призводить до ідеї завдань в Android. Додаток може використовувати компоненти іншого Android додатки для вирішення завдання, наприклад, якщо розроблювальне додаток припускає використання фотографій, воно може викликати додаток, що управляє фотографіями та зареєстроване в системі Android, вибрати з його допомогою фотографію і працювати з нею.

Для поповнення колекції додатків свого мобільного пристрою користувач може скористатися додатком Google Play, яке дозволяє купувати і встановлювати додатки з сервісу Google Play. Розробники, в свою чергу, можуть викладати свої додатки в цей сервіс, Google Play відслідковує появу оновлень програми, повідомляє користувачам цього додатка про оновлення і пропонує встановити його. Також Google Play надає розробникам доступ до послуг та бібліотекам, наприклад, доступ до використання і відображенню Google Maps.

Для установки програми на пристроях з ОС Android створюється файл з розширенням *.apk (Android package), який містить виконувані файли, а

також допоміжні компоненти, наприклад, файли з даними і файли ресурсів. Після установки на пристрій кожне додаток "живе" у своєму власному ізольованому екземплярі віртуальної машини Dalvik.

ОСНОВНІ ВИДИ ANDROID-ДОДАТКІВ

Приступаючи до розробки мобільних додатків добре б мати уявлення про те, які види додатків існують. Справа в тому, що якщо вдасться визначити до якого типу відноситься додаток, то стає зрозуміліше на які моменти в процесі його розробки необхідно звертати основну увагу. Можна виділити наступні види додатків:

- **Додатки переднього плану** виконують свої функції тільки, коли видимі на екрані, в іншому ж випадку їх виконання призупиняється. Такими додатками є, наприклад, ігри, текстові редактори, відеопрогравачі. При розробці таких програм необхідно дуже уважно вивчити життєвий цикл активності, щоб перемикання в фоновий режим і назад проходили гладко(безшовно), тобто При поверненні додатки на передній план було непомітно, що воно взагалі кудись пропадало. Для досягнення цієї гладкості необхідно стежити за тим, щоб при вході в фоновий режим додаток зберігало свій стан, а при виході на передній план відновлювало його. Ще один важливий момент, на який обов'язково треба звернути увагу при розробці додатків переднього плану, зручний і інтуїтивно зрозумілий інтерфейс.

- **Фонові програми** після настройки не припускають взаємодії з користувачем, більшу частину часу перебувають і працюють в прихованому стані. Прикладами таких додатків можуть служити, служби екранування дзвінків, SMS-автовідповідачі. У більшості своїй фонові програми націлені на відстеження подій, породжуваних апаратним забезпеченням, системою або іншими додатками, працюють непомітно. Можна створювати абсолютно невидимі сервіси, але тоді вони будуть некерованими. Мінімум дій, які необхідно дозволити користувачеві: санкціонування запуску сервісу, настройка, призупинення і переривання його роботи при необхідності.

- **Змішані додатки** більшу частину часу працюють у фоновому режимі, проте допускають взаємодію з користувачем після. Зазвичай взаємодія з користувачем зводиться до повідомлення про які-небудь події.

Прикладами таких додатків можуть служити мультимедіа-програвачі, програми для обміну текстовими повідомленнями(чати), поштові клієнти. Можливість реагувати на користувача введення і при цьому не втрачати працездатності у фоновому режимі є характерною особливістю змішаних додатків. Такі програми зазвичай містять як видимі активності, так і приховані(фонові) сервіси, і при взаємодії з користувачем повинні враховувати свій поточний стан. Можливо буде потрібно оновлювати графічний інтерфейс, якщо додаток знаходиться на передньому плані, або ж посилати користувачеві повідомлення з фонового режиму, щоб тримати його в курсі, що відбувається. І ці особливості необхідно враховувати при розробці подібних додатків.

- **Віджети** - невеликі додатки, які відображаються у вигляді графічного об'єкта на робочому столі. Прикладами можуть служити, додатки для відображення динамічної інформації, такої як заряд батареї, прогноз погоди, дата і час. Зрозуміло, складні додатки можуть містити елементи кожного з розглянутих видів. Плануючи розробку програми, необхідно визначити спосіб його використання, тільки після цього приступати до проектування і безпосередньо розробці.

БЕЗПЕКА

Звернемо увагу на організацію виконання додатків в ОС Android. Як вже було зазначено додатки під Android розробляються на мові програмування Java, компілюється в файл з розширенням.apk, після цього файл використовується для установки програми на пристрої, що працюють під управлінням Android. Після установки кожне Android додаток "живе" у своїй власній безпечною "пісочниці", розглянемо, як це виглядає:

- операційна система Android є багатокористувацької ОС, в якій кожен додаток розглядається як окремий користувач;
- за замовчуванням, система призначає кожному з додатком унікальний користувальницький ID, який використовується тільки системою і невідомий додатком;
- система встановлює права доступу до всіх файлів додатка наступним чином: доступ до елементів додатки має тільки користувач з відповідним ID;

- кожному додатку відповідає окремий Linux процес, який запускається, як тільки це необхідно хоча б одному компоненту програми, процес припиняє роботу, коли ні один компонент програми не використовує його або ж системі потрібно звільнити пам'ять для інших(можливо, більш важливих) додатків;
- кожному процесу відповідає окремий екземпляр віртуальної машини Dalvik, у зв'язку з цим код додатку виповнюється ізольовано від інших додатків.

Перераховані ідеї функціонування програми в ОС Android реалізують принцип мінімальних привілеїв, тобто кожному додатку, за замовчуванням, дозволений доступ тільки до компонентів, необхідним для його роботи і ніяким більше. Таким чином забезпечується дуже безпечне середовище функціонування додатків.

Однак, у випадку необхідності додатки можуть отримати доступ до даних інших додатків і системним сервісів(послуг). У випадку, коли двом додаткам необхідно мати доступ до файлів один одного, їм присвоюється один і той же користувальницький ID. Для економії системних ресурсів такі додатки запускаються в одному Linux процесі і ділять між собою один і той же екземпляр віртуальної машини, в цьому випадку додатки також повинні бути підписані одним сертифікатом. У разі ж, коли додатку потрібно доступ до системних даними, наприклад, контактам, SMS повідомленнями, картам пам'яті, камері, Bluetooth, Користувачеві необхідно дати додатком такі повноваження під час встановлення його на пристрій.

АРХІТЕКТУРА ПРОГРАМИ, ОСНОВНІ КОМПОНЕНТИ

Ось і прийшла пора поговорити безпосередньо про внутрішню організацію додатків під Android: обговорити їх архітектуру і основні компоненти.

Архітектура Android додатків заснована на ідеї багаторазового використання компонентів, які є основними будівельними блоками. Кожен компонент є окремою сутністю і допомагає визначити загальний поведінку програми.

Система Android вибудувана таким чином, що будь-яке додаток може запускати необхідний компонент іншої програми. Наприклад, якщо додаток припускає використання камери для створення фотографій, зовсім необов'язково створювати в цьому додатку активність для роботи з камерою. Напевно на пристрої вже є додаток для отримання фотографій з камери, досить запустити відповідну активність, зробити фотографію і повернути її в додаток, так що користувач буде вважати, що камера частина додатку, з яким він працює.

Коли система запускає компонент, вона запускає процес програми, якому належить компонент, якщо він ще не запущений, і створює екземпляри класів, необхідних компоненту. Тому на відміну від більшості інших систем, в системі Android програми не мають єдиної точки входу(немає методу main(), наприклад). В силу запуску кожної програми в окремому процесі і обмежень на доступ до файлів, додаток не може безпосередньо активувати компонент іншої програми. Таким чином для активації компонента іншої програми необхідно надіслати системі повідомлення про намір запустити певний компонент, система активує його.

Можна виділити чотири різних типи компонентів, кожен тип служить для досягнення певної мети і має свій особливий життєвий цикл, який визначає способи створення і руйнування відповідного компонента. Розглянемо основні компоненти Android-додатків.

Активності(Activities)

Активність - це видима частина додатки(екран, вікно, форма), відповідає за відображення графічного інтерфейсу користувача. При цьому додаток може мати кілька активностей, наприклад, у додатку, призначеному для роботи з електронною поштою, одна активність може використовуватися для відображення списку нових листів, інша активність - для написання, і ще одна - для читання листів. Незважаючи на те, що для користувача додаток представляється єдиним цілим, все активності додатки не залежать один від одного. У зв'язку з цим будь-яка з цих активностей може бути запущена з іншої програми, що має доступ до активностям цього додатка. Наприклад, додаток камери може запустити активність, що створює нові листи, щоб

відправити щойно зроблену фотографію адресату, зазначеному користувачем.

Сервіси(Services).

Сервіс - компонент, який працює у фоновому режимі, виконує тривалі за часом операції або роботу для віддалених процесів. Сервіс не надає інтерфейсу користувача. Наприклад, сервіс може програвати музику у фоновому режимі, поки користувач використовує інше додаток, може завантажувати дані з мережі, що не блокуючи взаємодію користувача з активністю. Сервіс може бути запущений іншим компонентом і після цього працювати самостійно, а може залишитися пов'язаним з цим компонентом і взаємодіяти з ним.

Контент-провайдери(Content providers).

Контент-провайдер управляє розподіленим безліччю даних програми. Дані можуть зберігатися в файлової системі, в базі даних SQLite, в мережі, в будь-якому іншому доступному для програми місці. Контент-провайдер дозволяє іншим програмам за наявності у них відповідних прав робити запити або навіть міняти дані. Наприклад, у системі Android є контент-провайдер, який управляє інформацією про контакти користувача. У зв'язку з цим, будь-яке додаток з відповідними правами може зробити запит на читання і запис інформації будь-якого контакту. Контент-провайдер може бути також корисний для читання і запису приватних даних програми, не призначених для доступу ззовні.

Приймачі широкомовних повідомлень(Broadcast Receivers).

Приймач - компонент, який реагує на широкомовні повідомлення. Більшість таких оповіщень породжуються системою, наприклад, повідомлення про те, що екран відключився або низький заряд батареї. Додатки також можуть ініціювати широкомовлення, наприклад, розіслати іншим додаткам повідомлення про те, що деякі дані завантажені і доступні для використання. Хоча приймачі не відображають користувача інтерфейсу, вони можуть створювати повідомлення на панелі станів, щоб попередити користувача про появу повідомлення. Такий приймач служить провідником

до інших компонентів і призначений для виконання невеликого обсягу робіт, наприклад, він може запустити відповідний події сервіс.

Усі розглянуті компоненти є спадкоємцями класів, визначених у Android SDK.

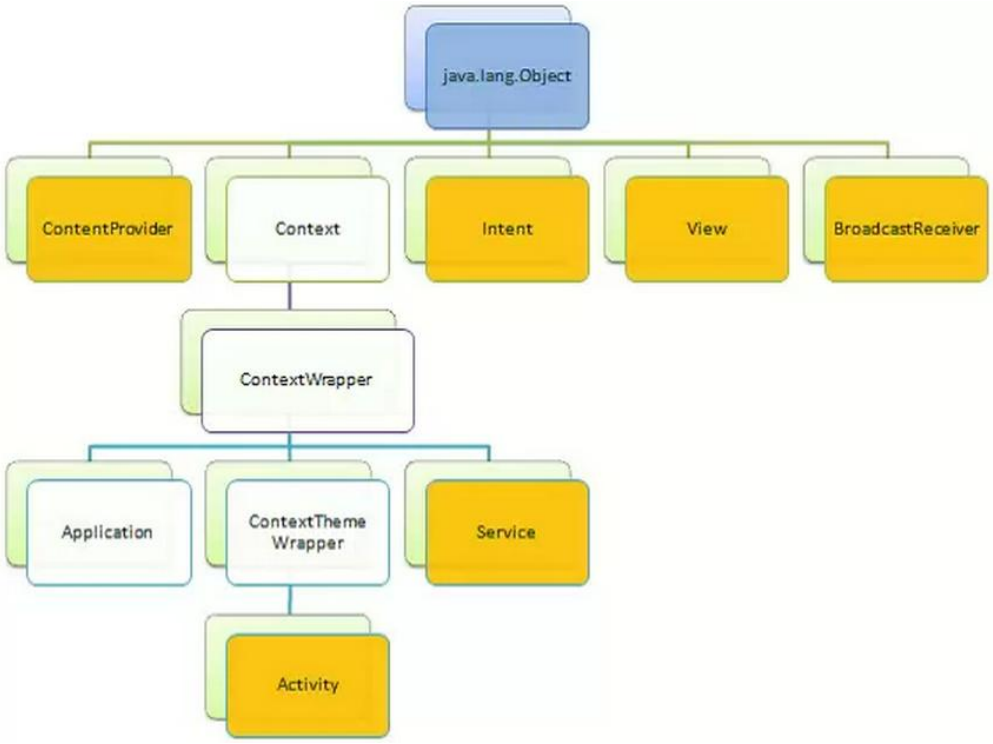


Рис. 2. Ієрархія класів Android SDK

На рис. 2.1 показана ієрархія основних класів Android SDK, з якими зазвичай має справу розробник. Насправді класів набагато більше, жовтим кольором виділені класи, з якими розробник працює безпосередньо, успадковує від них свої класи. Інші класи не менш важливі, але вони рідше використовуються безпосередньо. Для початку розглянемо класи Intent і View.

Клас View є основним будівельним блоком для компонентів інтерфейсу користувача(UI), він визначає прямокутну область екрану і відповідає за промальовування і обробку подій.

Є базовим класом для віджетів (GUI widgets), які використовуються для створення інтерактивних компонентів для користувача інтерфейсу: кнопок, текстових полів і т.п.

А також є базовим класом для класу ViewGroup, який є невидимим контейнером для інших контейнерів і віджетів, визначає властивості розташування компонентів для користувача інтерфейсу.

Інтерфейс Android-додаток являє собою ієрархію UI компонентів (див. рис.3), можна описати цю ієрархію програмно, але більш простим і ефективним способом задати розташування елементів інтерфейсу є XML файл, який надає зручну для сприйняття структуру компоновки(layout file). Під час виконання XML файл автоматично перетворюється в дерево відповідних об'єктів.

Детальніше про клас View, властивості і методах: http://developer.android.com/reference/android/view/View.html.

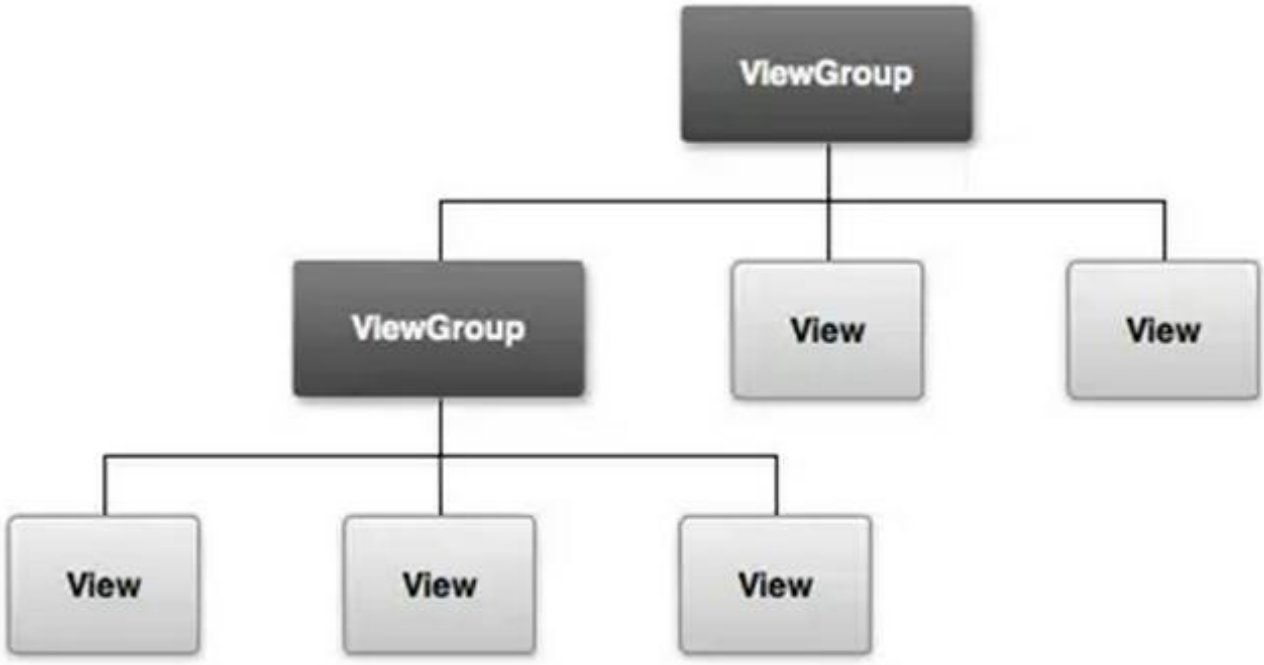


Рис. 3. Ієрархія компонентів, що визначає компоновку інтерфейсу користувача

Об'єкти-екземпляри класу Intent використовуються для передачі повідомлень між основними компонентами додатків. Відомо, що три з чотирьох основних компонентів: активності, сервіси та приймачі широкомовних повідомлень, можуть бути активовані за допомогою повідомлень, які називаються намірами. Такі повідомлення є інструментом пізнього зв'язування компонентів одного або декількох додатків. Екземпляр

класу Intent являє собою структуру даних, що містить опис операції, яка повинна бути виконана, і зазвичай використовується для запуску активності або сервісу. У випадку з приймачами широкомовних повідомлень об'єкт Intent містить опис події, яка сталася або було оголошено.

Для кожного типу компонентів існують свої механізми передачі намірів.

- Щоб запустити активність або викликати у працюючої активності нову дію, необхідно передати об'єкт-намір в метод Context.startActivity() або Activity.startActivityForResult().
- Щоб запустити сервіс або доставити нові інструкції працюючому сервісу, необхідно передати об'єкт-намір в метод Context.startService(). Також об'єкт-намір може бути переданий в метод Context.bindService(), щоб зв'язати між собою викликає компонент і сервіс.
- Щоб доставити об'єкт-намір всім зацікавленим приймачів широкомовних повідомлень, необхідно передати його в будь-який з широкомовних методів: Context.sendOrderedBroadcast(), Context.sendStickyBroadcast(), Context.sendBroadcast().

У кожному випадку система Android у відповідь на намір знаходить відповідний компонент: активність, сервіс або безліч широкомовних приймачів і запускає його якщо необхідно. У цій системі повідомлень не трапляється накладок: повідомлення-намір, відправлене певного компоненту, буде отримано саме цим компонентом і ніким іншим.

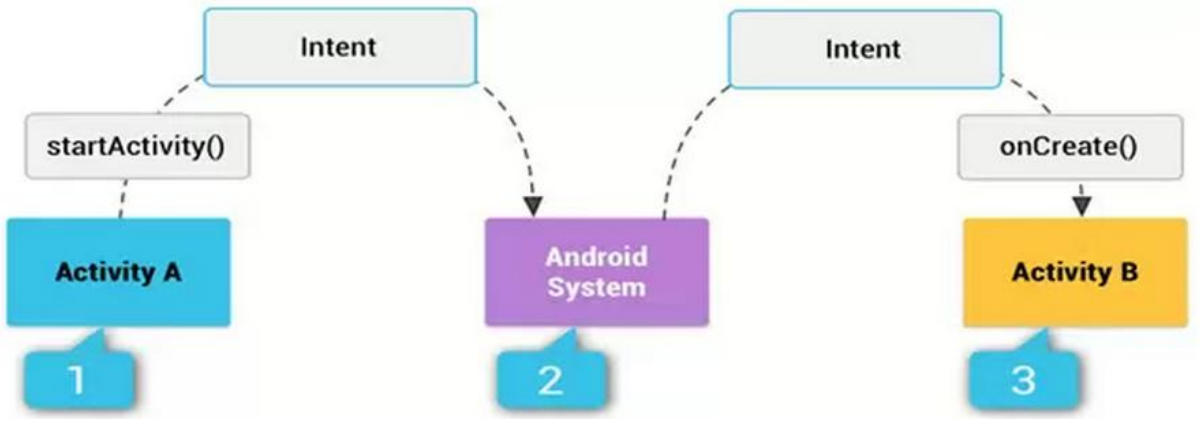


Рис. 4. Передача намірів(Intent)

На рис.4 можна побачити як відбувається передача намірів(Intent), в даному випадку одна активність запускає іншу. Активність А створює намір(Intent) з описом дії і передає його в метод startActivity(). Система Android перевіряє всі додатки на збіг з наміром, коли збіг знайдено, система запускає відповідну активність, для чого викликає метод onCreate() і передає в нього об'єкт – намір Intent.

Детальніше про клас Intent:

– http://developer.android.com/guide/components/intents-filters.html;

– http://developer.android.com/reference/android/content/Intent.html.

Лабораторна робота №1

ТЕМА: ОСНОВИ РОЗРОБКИ МОБІЛЬНОГО ПЗ

МЕТА: Отримати практичні навики розробки мобільних програм під керуванням ОС ANDROID.

ЗНАТИ: Основи Android Studio чи Eclipse

ТЕОРЕТИЧНІ ВІДОМОСТІ. У зв'язку з великим обсягом інформації використовувати електронну документацію (погоджувати з лектором).

ЗАВДАННЯ

Використовуючи наявну електронну документацію створити мобільне ПЗ під керуванням ОС Android. Яке складається з двох кнопок та текстового поля для виведення інформації користувачу.

1. При натискання на кнопку №1 у текстове поле виводиться текст «Hello World!» та П.І.Б. студента розробника. При натискання на кнопку №2 розроблене ПЗ завершує свою роботу.

Рисунок 1 – Приклад вікна ПЗ

2. У ПЗ є поле введення тексту та елемент для відображення результату (TextView). При введені тексту та натисканні кнопк –текст відображається на екрані. Необхідно під час повороту екрана (з книжкової орієнтації в альбомну або навпаки) щоб введені дані не втрачалися.

3. Зробити локалізацію інтерфейсу (укр, англ)

4. Зробити темну і світлу тему яка буде перемикатися в залежності від часових налаштувань.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке Java SDK (JDK)?
2. Навіщо потрібен Android SDK, SDK Manager?
3. Для чого необхідне Android Virtual Device (AVD)?
4. Для чого потрібен файл AndroidManifest.xml?

Лабораторна робота №2

ТЕМА: РОБОТА З КОНТЕКСТНИМ МЕНЮ

МЕТА: Отримати практичні навики розробки мобільних програм під керуванням ОС ANDROID.

ЗНАТИ: Основи Android Studio

ТЕОРЕТИЧНІ ВІДОМОСТІ. У зв'язку з великим обсягом інформації використовувати електронну документацію (погоджувати з лектором).

ЗАВДАННЯ

Використовуючи наявну електронну документацію створити мобільне ПЗ під керуванням ОС Android. Яке складається з двох кнопок, меню та контекстного меню.

1. При натисканні на кнопку №1 на екран виводиться меню з двома пунктами: «Приховано»; «Неприховано» рис.1, а.

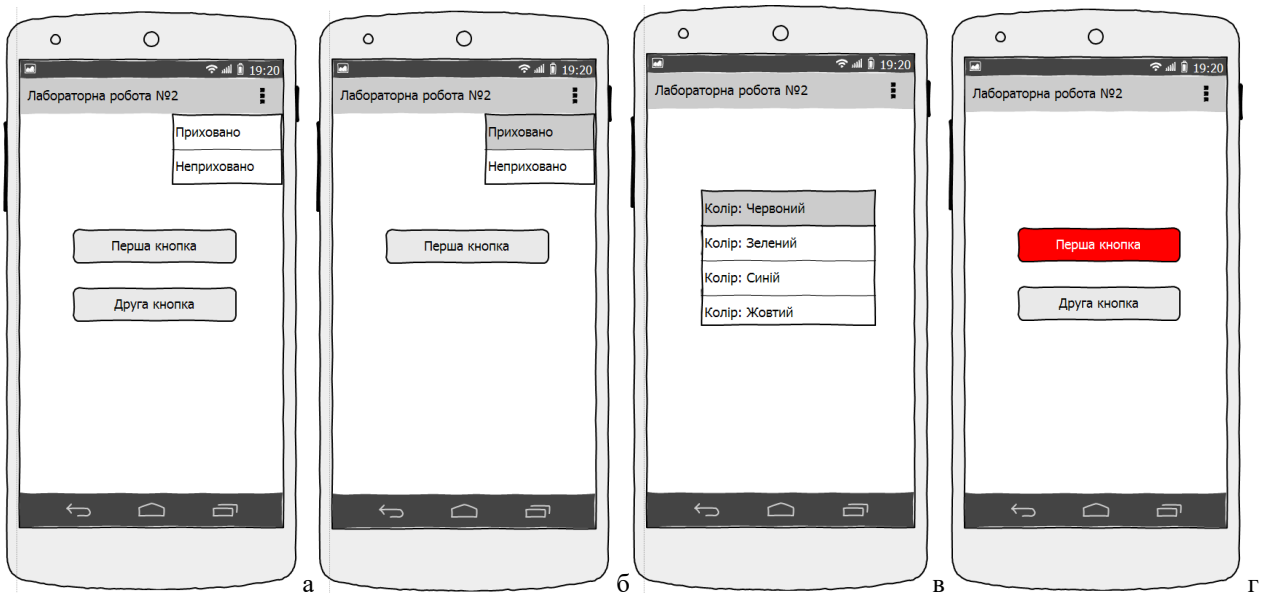
При натисканні пункту «Приховано» рис.1,б приховується кнопка №2, при натисканні «Неприховано» – кнопка виводиться на екран рис.1, а.

При довгому натисканні на кнопку №1 виводиться контекстне меню (рис.1,в) з пунктами обрання кольору кнопки, при обранні кольору відбувається зміна кольору кнопки №1 (рис.1, г).

2. Зробити діалогове вікно (Dialog) підтвердження виходу з програми.

3. Додати пункт меню «Скинути» який повертає значення в початковий стан.

4. Зберігати вибраний колір кнопки №1 при повороті екрана.



а б в г

Рисунок 1 – Приклад вікон ПЗ

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Опишіть елементи екрану та їх властивості?
2. Навіщо потрібен ViewGroup та RelativeLayout?
3. Навіщо потрібен Layout-файл в Activity?
4. Що таке XML представлення?
5. Як змінити орієнтацію екрана?
6. Як працювати з меню?
7. Як створити пункти меню с ID?
8. Як групувати і сортувати пункти меню?
9. Що таке контекстне меню?

Лабораторна робота №3

ТЕМА: ОСНОВИ РОЗРОБКИ БАГАТОЕКРАННОГО МОБІЛЬНОГО ПЗ

МЕТА: Отримати практичні навики розробки мобільних програм під керуванням ОС ANDROID.

ЗНАТИ: Основи Android Studio

ТЕОРЕТИЧНІ ВІДОМОСТІ. У зв'язку з великим обсягом інформації використовувати електронну документацію (погоджувати з лектором).

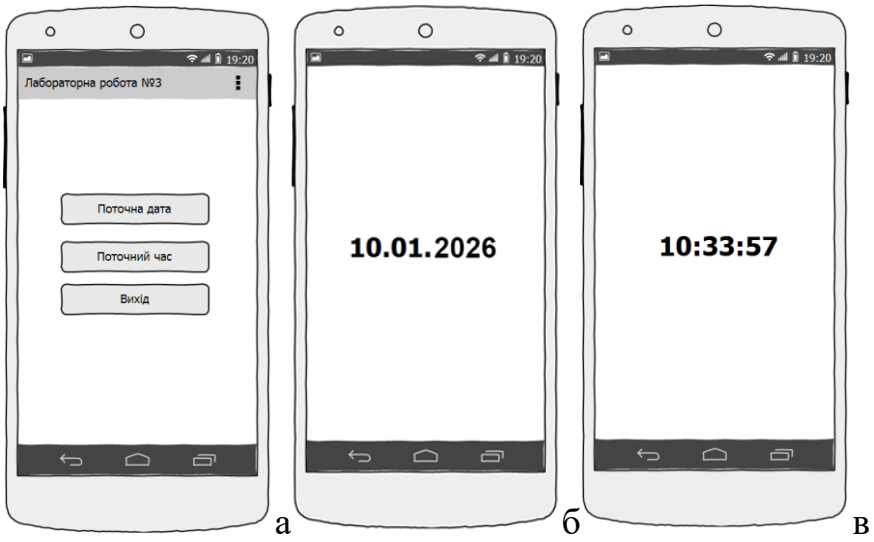
ЗАВДАННЯ

1. Використовуючи наявну електронну документацію створити мобільне ПЗ під керуванням ОС Android. Яке складається з трьох кнопок (рис.1,а). При натискання на кнопку №1 відбувається перехід на новий екран (рис.1,б) в якому відтворюється інформація про поточну дату (формат на вибір). З даного екрану також можна повернутися в попередній. При натискання на кнопку №2 відбувається перехід на новий екран (рис.1,в) в якому відтворюється інформація про поточний час (формат на вибір). З даного екрану також можна повернутися в попередній. При натискання на кнопку №3 розроблене ПЗ завершує свою роботу з діалоговим вікном підтвердження. З підтримкою зміни орієнтації без втрати даних.

2. Створити вікно налаштування де можна налаштовувати формат відображення поточного часу (24h/12h), формат відображення поточної дати короткий (dd.mm.yyyy - 23.02.2025) та повний (dd MMMM yyyy - 23 лютого 2025 р.).

3. Використовувати у налаштуваннях DatePicker і TimePicker (рис.2, рис.3).

4. Забезпечення збереження історії останніх вибраних значень, при перезавантаженні ПЗ.



а б в

Рисунок 1 – Приклад вікон ПЗ

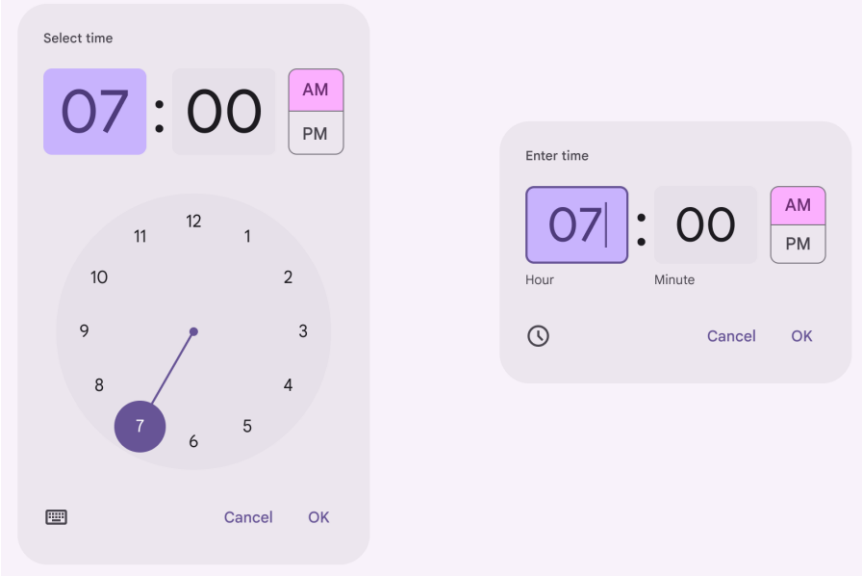


Рисунок 2 – Приклад «A dial and an input time picker»
https://developer.android.com/develop/ui/compose/components/time-pickers

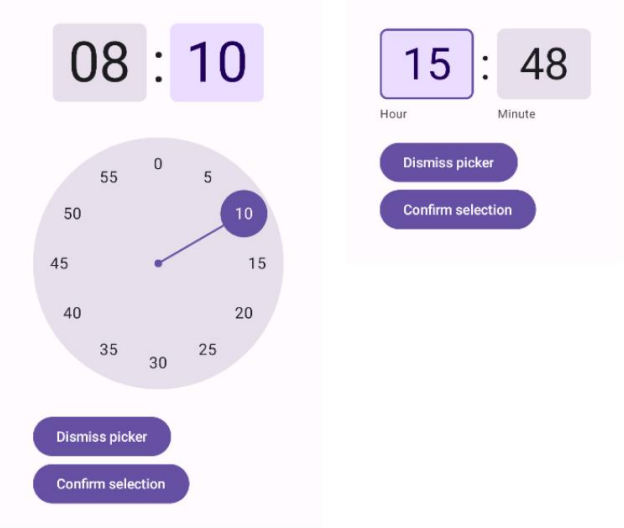


Рисунок 3 – Приклад «A dial time picker, An input time picker»
https://developer.android.com/develop/ui/compose/components/time-pickers

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Навіщо потрібен LinearLayout?
2. Навіщо потрібен TableLayout?
3. Навіщо потрібен RelativeLayout?
4. Навіщо потрібен AbsoluteLayout?
5. Які існують види Layouts. Ключові відмінності і властивості?
6. Що таке абсолютне значення?
7. Які існують Layout параметри для View-елементів?

Лабораторна робота №4

ТЕМА: ОСНОВИ РОБОТИ З ТЕКСТОВИМИ ДАНИМИ

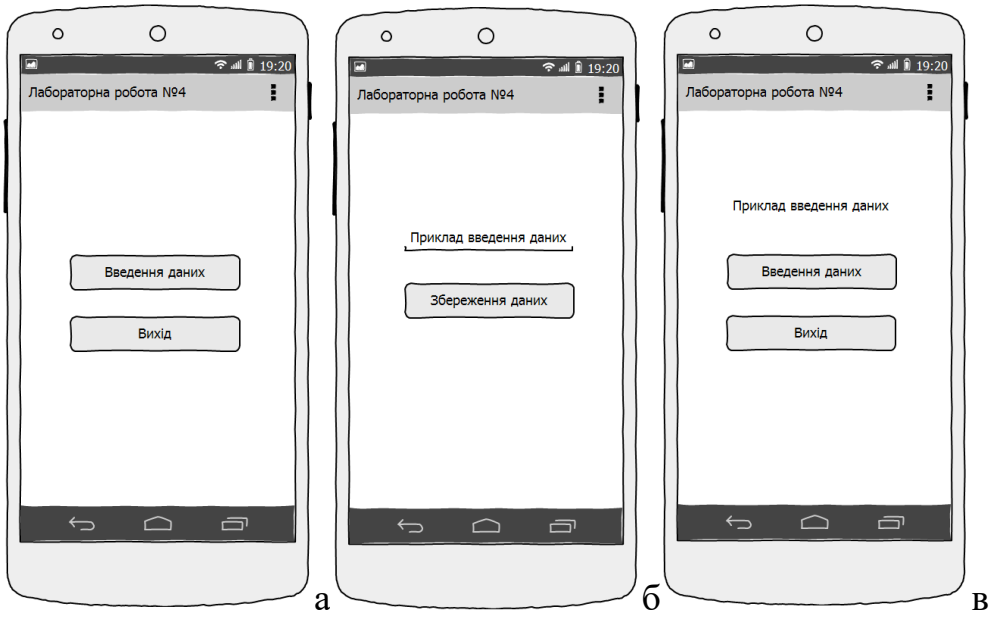
МЕТА: Отримати практичні навики розробки мобільних програм під керуванням ОС ANDROID.

ЗНАТИ: Основи Android Studio

ТЕОРЕТИЧНІ ВІДОМОСТІ. У зв'язку з великим обсягом інформації використовувати електронну документацію (погоджувати з лектором).

ЗАВДАННЯ

1. Використовуючи наявну електронну документацію створити мобільне ПЗ під керуванням ОС Android. Яке складається з двох кнопок та текстового поля (рис. 1,а). При натискання на кнопку №1 відбувається перехід на новий екран в якому відбувається введення текстових даних (рис. 1,б). При натисненні кнопки «Збереженні даних» відбувається повернення до головного вікна програми та виведення збережених даних до текстового поля (рис. 1,в). Отримана інформація повинна відтворитися в текстовому полі головного екрану. При натискання на кнопку №2 розроблене ПЗ завершує свою роботу.



а б в

Рисунок 1 – Приклад вікон ПЗ

2. Через контексне меню зробити перехід до нового екрану в якому відбувається введення даних по полям:

– ПІБ (ліміт символів 30 повного імені);

– E-mail (формат example@email.com стандарт RFC 5321/5322. Дозволені символи: Латинські літери (a-z), цифри, крапка (.), дефіс (-), підкреслення (_) та символ @);

– Мобільний телефон (стандарт E.164. Маска шаблону «+38 (0__) ___-__-__», тільки цифри);

ЗАБЕЗПЕЧИТИ введення та валідацію даних по формі маски.

ЗБЕРІГАТИ останній запис локально (SharedPreferences або у файл).

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке обробники подій?
2. Наведіть приклад як призначити обробник подій Button?
3. Навіщо потрібен метод findViewById?
4. Навіщо потрібен метод setOnClickListener?
5. Розкажіть механізм обробки подій, наведіть приклади?

Лабораторна робота №5

ТЕМА: ОСНОВИ РОБОТИ ЗІ СПИСКАМИ ДАНИХ

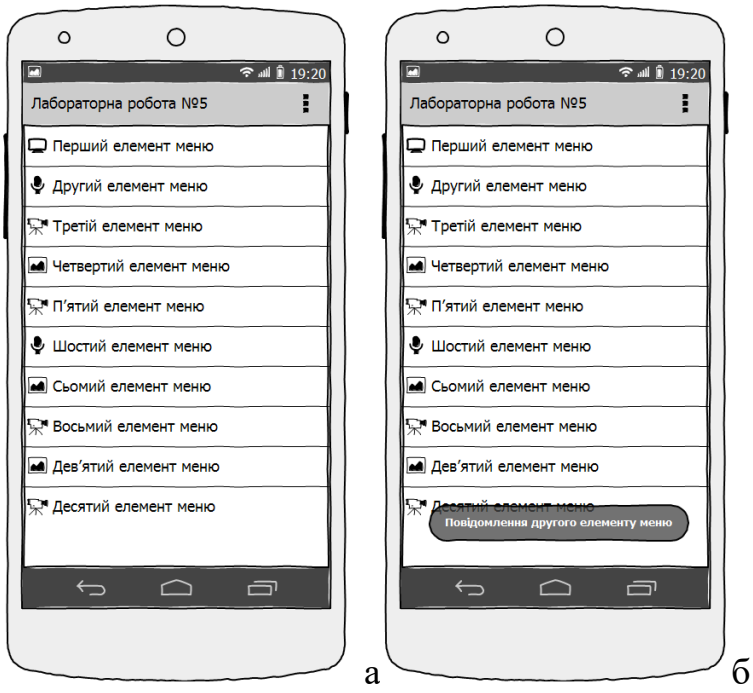
МЕТА: Отримати практичні навики розробки мобільних програм під керуванням ОС ANDROID.

ЗНАТИ: Основи Android Studio

ТЕОРЕТИЧНІ ВІДОМОСТІ. У зв'язку з великим обсягом інформації використовувати електронну документацію (погоджувати з лектором).

ЗАВДАННЯ

1. Використовуючи наявну електронну документацію створити мобільне ПЗ під керуванням ОС Android. Яке складається з списку (ListView). В списку відтворюється наступна інформація (рис. 1,а): будь-які текстові дані в правій частині екрану рядка і будь-яка картинка в лівій частині екрану цього ж рядка. Список повинен містити не менше десяти рядків з описаними вище даними. При натисненні на рядок або на один з його елементів (рис. 1,б) потрібно показати користувачу інформацію по обраному рядку (Toast message).



а б

Рисунок 1 – Приклад вікон ПЗ

2. Реалізувати пошук, фільтрацію та сортування даних за назвою.

2.1 Реалізувати поле для введення даних пошуку елементів списку, з реалізацією приховання (фільтрації) інших даних при введені кожної нової букви. У списку відображаються тільки ті елементи, назва яких починається з введеного рядку. Якщо поле пошуку порожнє, у списку відображаються всі елементи.

2.2 Реалізувати сортування елементів списку за назвою у алфавітному порядку.

Передбачити елемент керування для запуску сортування (кнопка або пункт меню). Після сортування список відображає елементи у впорядкованому вигляді без втрати інших даних (текст, зображення). Забезпечити коректну спільну роботу фільтрації та сортування: сортування працює для повного списку; сортування працює для відфільтрованого списку; очищення пошуку повертає повний список у коректному стані.

ПЕРЕВІРИТИ РОБОТУ ЗАСТОСУНКУ ДЛЯ ТАКИХ ВИПАДКІВ:

– знайдено декілька елементів;

– знайдено один елемент;

– збігів немає;

– повторне сортування після фільтрації;

– очищення пошуку після сортування.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як використовувати один обробник для декількох View-елементів?
2. Як Activity використовувати в якості обробника?
3. Навіщо потрібна папка res/values, що в ній можна зберігати і як використовувати?
4. Що таке Log дані та випливаючи повідомлення?
5. Як малювати екран програмно, а не через layout-файл?
6. Як додавати компоненти на екран прямо з програми?

Лабораторна робота №6

ТЕМА: ОСНОВИ РОБОТИ З РІЗНОМАНІТНИМИ МАСИВАМИ ДАНИХ

МЕТА: Отримати практичні навики розробки мобільних програм під керуванням ОС ANDROID.

ЗНАТИ: Основи Android Studio

ТЕОРЕТИЧНІ ВІДОМОСТІ. У зв'язку з великим обсягом інформації використовувати електронну документацію (погоджувати з лектором).

ЗАВДАННЯ

Використовуючи наявну електронну документацію створити мобільне ПЗ під керуванням ОС Android (основа 5 ЛР).

1. Яке складається з списку (ListView). В списку відтворюється наступна інформація (рис. 1,а): будь-які текстові дані в правій частині екрану рядка і будь-яка картинка в лівій частині екрану цього ж рядка. Список повинен містити не менше десяти рядків з описаними вище даними.

2. Забезпечити сортування та фільтрацію списку.

При натисненні на рядок списку переходить в нове вікно (рис. 1,б). В новому вікні повинна відтворюватися наступна інформація:

– Текстове поле з відображенням обраного елементу меню;

– Картинка 1;

– Картинка 2;

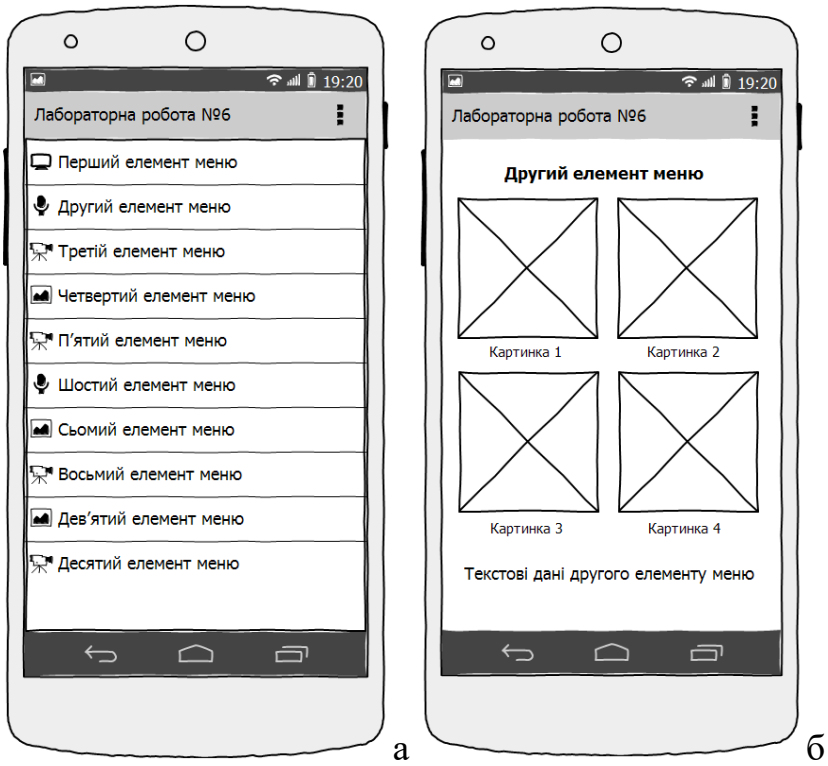
– Картинка 3;

– Картинка 4;

– Багаторядкове поле з довільним текстом до кожного елементу меню.

Наприклад в списку знаходяться дані про ваших друзів (картинка і текстова інформація).

Після натиснення на рядок з другом відкривається нове вікно з розширеною інформацією: 4 фото та інформація про них.



а б

Рисунок 1 – Приклад вікон ПЗ

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як використовувати один обробник для декількох View-елементів?
2. Як Activity використовувати в якості обробника?
3. Навіщо потрібна папка res/values, що в ній можна зберігати і як використовувати?
4. Що таке Log дані та випливаючи повідомлення?
5. Як малювати екран програмно, а не через layout-файл?
6. Як додавати компоненти на екран прямо з програми?

Лабораторна робота №7

ТЕМА: ОСНОВИ МАНІПАЛЮВАННЯ ВІКНАМИ ПЗ

МЕТА: Отримати практичні навики розробки мобільних програм під керуванням ОС ANDROID.

ЗНАТИ: Основи Android Studio

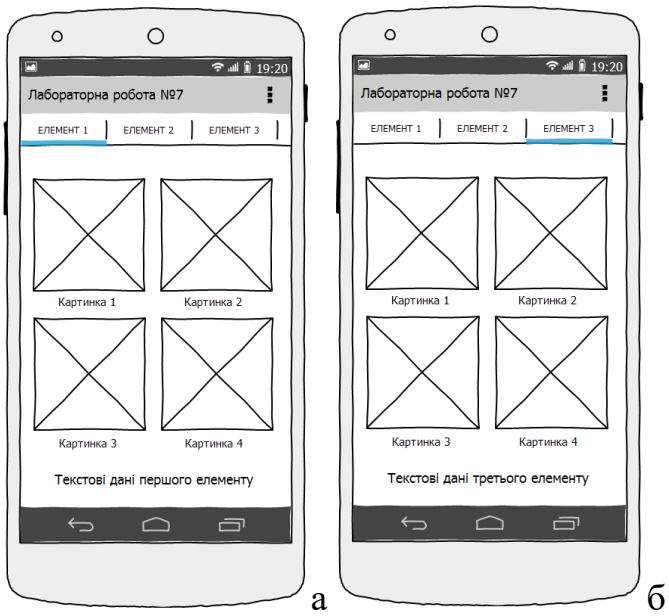
ТЕОРЕТИЧНІ ВІДОМОСТІ. У зв'язку з великим обсягом інформації використовувати електронну документацію (погоджувати з лектором).

ЗАВДАННЯ

Використовуючи наявну електронну документацію створити мобільне ПЗ під керуванням ОС Android. Яке складається з екранів (на основі ViewPager) з можливістю перегортання їх вліво чи вправо.

На кожному екрані відображаються дані з 6 ЛР, тобто: Текстове поле з відображенням обраного елементу меню; Картинка 1; Картинка 2; Картинка 3; Картинка 4; Багаторядкове поле з довільним текстом до кожного елементу меню.

<u>РЕАЛІЗУВАТИ через ViewPager2+FragmentStateAdapter.</u> Приклад https://developer.android.com/develop/ui/views/animations/screen-slide-2



а б

Рисунок 1 – Приклад вікон ПЗ

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке Task?
2. Як фіксувати Activity в стані Paused?
3. Що таке виклик Activity, використовуючи неявний виклик?
4. Як читати action з Intent?
5. Як передавати дані за допомогою Intent?
6. Як викликається Activity з поверненням результату?
7. Навіщо потрібен requestCode і resultCode в onActivityResult?
8. Що таке Uri і Intent-атрибут дата?
9. Як викликати системні програми (браузер, карта)?

Лабораторна робота №8

ТЕМА: РОБОТА З JSON ТА XML ДАНИМИ

МЕТА: Отримати практичні навики розробки мобільних програм під керуванням ОС ANDROID.

ЗНАТИ: Основи Android Studio

ТЕОРЕТИЧНІ ВІДОМОСТІ. У зв'язку з великим обсягом інформації використовувати електронну документацію (погоджувати з лектором).

ЗАВДАННЯ

Використовуючи наявну електронну документацію створити мобільне ПЗ під керуванням ОС Android.

1. Підготувати та наповнити змістом (50 значень) два ідентичних файли даних з однаковою структурою різного формату JSON та XML, з наступними полями:

– id;

– title або name;

– description;

– image (назва ресурсу або ідентифікатор) ;

– додаткові поля на вибір студента для забезпечення показу повного вмісту даних (див. пункт 7.).

2. РЕАЛІЗУВАТИ у першому вікні зчитування даних з файлу JSON – відкрити файл, розпарсити вміст, перетворити дані у список об’єктів моделі (скорочений вміст даних).

3. РЕАЛІЗУВАТИ у другому вікні зчитування даних з файлу XML– відкрити файл, розпарсити вміст, перетворити дані у список об’єктів моделі (скорочений вміст даних). Забезпечити перехід між вікнами.

4. ВІДОБРАЗИТИ отримані дані у списку (ListView або RecyclerView) замість статичних масивів.

5. ЗАБЕЗПЕЧИТИ однакове відображення даних незалежно від формату джерела.

6. РЕАЛІЗУВАТИ обробку помилок - файл відсутній, файл пошкоджений або має неправильний формат, порожній список даних.

7. РЕАЛІЗУВАТИ при виборі елемента списку відкрити екран деталей або сторінку ViewPager2 з даними вибраного елемента (повний вміст даних).

8. ПЕРЕВІРИТИ коректність роботи ПЗ для обох форматів: кількість елементів збігається; назви та описи відображаються коректно; переходи на детальний екран (повний вміст даних) працюють; застосунок не завершується з помилкою.

9. ЗАБЕЗПЕЧИТИ пошук і сортування для даних, завантажених з JSON та XML.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке JSON і XML, та для яких задач у мобільному застосунку їх зазвичай використовують?
2. Які основні структурні елементи має JSON (об’єкт, масив, ключ, значення) і як вони записуються?
3. Які основні структурні елементи має XML (тег, атрибут, текстовий вузол, вкладені елементи) і як вони організовуються?
4. У чому полягає різниця між поданням даних у JSON через пари ключ значення та в XML через елементи й атрибути?
5. Які типи даних підтримує JSON, і як у JSON подаються рядки, числа, логічні значення та null?
6. Які вимоги до коректності XML документа (наявність кореневого елемента, закривальні теги, правильна вкладеність)?
7. Які типові помилки виникають під час парсингу JSON та XML у застосунку Android, і як їх виявляти?
8. У чому переваги JSON порівняно з XML для обміну даними в мобільних застосунках, і в чому переваги XML порівняно з JSON?
9. Як у JSON та XML подається колекція однотипних об’єктів (наприклад список товарів або список записів)?
10. Чому важливо мати однакову логічну структуру даних у JSON і XML файлах, якщо застосунок підтримує завантаження з обох форматів?

Рекомендована література

Базова

1. Murphy M. L. Elements of Android Jetpack. [Електронний ресурс]. 2021. URL: https://commonsware.com/Jetpack/Jetpack-FINAL.pdf
2. Murphy M. L. Exploring Android. FINAL Version [Електронний ресурс]. 2021. URL: https://commonsware.com/AndExplore/AndExplore-FINAL.pdf
3. Murphy, M. L. Elements of Android Room. FINAL Version [Електронний ресурс]. 2021. URL: https://commonsware.com/Room/Room-FINAL.pdf
4. Google. Android Developers, офіційна документація та навчальні матеріали з розробки під Android [Електронний ресурс]. URL: https://developer.android.com/?hl=en
5. Google. Курси з Android розробки (включно з навчальними шляхами та codelabs) [Електронний ресурс]. URL: https://developer.android.com/courses?hl=en
6. Google. Курс Android Basics with Compose, базова розробка застосунків на Kotlin та Jetpack Compose [Електронний ресурс]. URL: https://developer.android.com/courses/android-basics-compose/course?hl=en
7. Google. Jetpack Compose, документація UI тулкіта та приклади [Електронний ресурс]. URL: https://developer.android.com/develop/ui/compose/documentation?hl=en
8. Google. Архітектура Android застосунку, рекомендації щодо шарів, ViewModel, repository та залежностей [Електронний ресурс]. URL: https://developer.android.com/topic/architecture?hl=en
9. Google. Android API Reference, довідник класів та API платформи [Електронний ресурс]. URL: https://developer.android.com/reference?hl=en
10. Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85091278920&origin=resultslist **(Scopus)**.
11. Коваленко О.В. Методи та засоби управління безпекою додатків. Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44. Режим доступу: http://nbuv.gov.ua/UJRN/Ikszt_2018_4_7 **(Фахове видання. Категорія «Б»)**
12. Коваленко О.В. Удосконалений метод управління ризиками розробки програмного забезпечення на основі напівмарковської моделі прийняття рішень. Сучасні інформаційні системи. – Випуск 2 (3). – Харків. – 2018. – С. 41-48. Режим доступу: http://repository.kpi.kharkov.ua/handle/KhPI-Press/40480 **(Фахове видання. Категорія «Б»)**
13. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125. Режим доступу: http://journals.nupp.edu.ua/sunz/article/view/1146 **(Фахове видання. Категорія «Б»)**

14. Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп'ютерних систем для підвищення безпеки даних: **монографія** / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 295 с.

Допоміжна

15. Google. Тестування Android застосунків, підходи та інструменти (unit, instrumented, UI tests) [Електронний ресурс]. URL: https://developer.android.com/training/testing?hl=en
16. Google. Room, офіційна документація з роботи з локальною БД через ORM над SQLite [Електронний ресурс]. URL: https://developer.android.com/training/data-storage/room?hl=en
17. Google. DataStore, сучасне зберігання налаштувань та даних (Preferences, Proto) [Електронний ресурс]. URL: https://developer.android.com/topic/libraries/architecture/datastore?hl=en
18. Google. Практики безпеки Android застосунків, рекомендації щодо даних, дозволів і взаємодій [Електронний ресурс]. URL: https://developer.android.com/privacy-and-security/security-best-practices?hl=en
19. Google. Baseline Profiles, оптимізація продуктивності та швидкості запуску [Електронний ресурс]. URL: https://developer.android.com/topic/performance/baselineprofiles/overview?hl=en
20. Google. Профілювання в Android Studio, інструменти аналізу CPU, пам'яті, енергії та мережі [Електронний ресурс]. URL: https://developer.android.com/studio/profile?hl=en
21. Kotlin. Coroutines overview, основи корутин та асинхронності в Kotlin для Android [Електронний ресурс]. URL: https://kotlinlang.org/docs/coroutines-overview.html
22. Gradle. Gradle User Manual, документація системи збірки для Android проєктів [Електронний ресурс]. URL: https://docs.gradle.org/
23. Google. Налаштування Firebase для Android проєкту, підключення SDK та конфігурація [Електронний ресурс]. URL: https://firebase.google.com/docs/android/setup?hl=en
24. Google. Довідка Play Console, створення та первинне налаштування застосунку для публікації [Електронний ресурс]. URL: https://support.google.com/googleplay/android-developer/answer/9859152?hl=en
25. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 с.
26. Poul Klausen. JAVA 17: more about Java and Android software development. Bookboon. 2018. 257 с.
27. Peter Sommerhoff. Kotlin for Android App Development. Addison-Welsey Professional. 2019. 435 с.
28. Alexey Soshin. Kotlin Design Patterns and Best Practices. Packt Publishing. 2022. 513 с.
29. Pierre-Olivier Laurence, Amanda Hinchman-Dominguez. Programming Android with Kotlin. O'Reilly Media. 2022. 355 с.
30. Kevin D. Moore, Carlos Mota, Saeed Taheri. Kotlin Multiplatform by Tutorials. Razeware LLC. 2022. 400 с.

31. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 с.
32. Filip Babić, Luka Kordić, Nishant Srivastava. Kotlin Coroutines by Tutorials. Razeware LLC. 2022. 287 с.
33. Irina Galata, Victoria Gonda, Joe Howard, Ellen Shapiro. Kotlin Apprentice. Razeware LLC. 2021. 491 с.
34. René Cacheaux & Josh Berlin. Advanced iOS App Architecture. Razeware LLC. 2022. 334 с.
35. Ehab Amer, Alexis Gallagher, Matt Galloway, Eli Ganim, Ben Morrow, Cosmin Pupăză. Swift Apprentice. Razeware LLC. 2021. 555 с.
36. Neil Smyth. SwiftUI Essentials. Payload Media. 2022. 526 с.
37. Will Grant. 101 UX Principles. Packt Publishing. 2022. 432 с.