Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №2

з дисципліни

«Розробка мобільних застосувань під Android»

Виконала:

студентка 3 курсу ФІОТ

групи ІО-23

Чепенюк Анастасія

Київ 2025

Тема: дослідження роботи з компонентом Fragment.

Мета роботи: дослідити створення та взаємодію з компонентом Фрагмент (Fragment) компоненту Діяльність та набути практичні навички з використання фрагментів для інтерфейсу користувача.

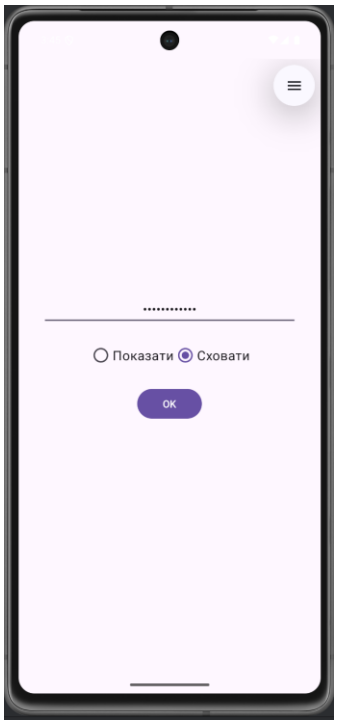
Завдання: Написати програму під платформу Андроїд, яка має інтерфейс, побудований з декількох фрагментів згідно варіанту. Перший фрагмент представляє з себе форму для введення даних та кнопку підтвердження («ОК»), а інший фрагмент відображає результат взаємодії. Тобто другий фрагмент містить тестове поле з результатом та кнопкою «Cancel» (якщо згідно варіанту така існує, якщо ж за варіантом її немає – можете додати за власним бажанням), яка очищає або приховує (або видаляє) другий фрагмент та очищає форму введення з першого фрагменту. Зверніть увагу, що робота з фрагментами відбувається в рамках однієї Діяльності.

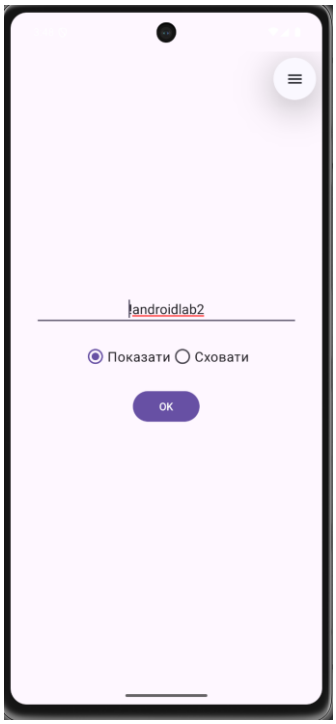
Мій номер в списку: 159
Варіант: 6

6.	Вікно введення паролю містить: текстове поле, дві опції (радіо-батони) режиму введення (відображати введені символи або відображати зірочки) та кнопку «ОК». Вивести введений пароль при натисканні на кнопку «ОК» у деяке текстове поле.

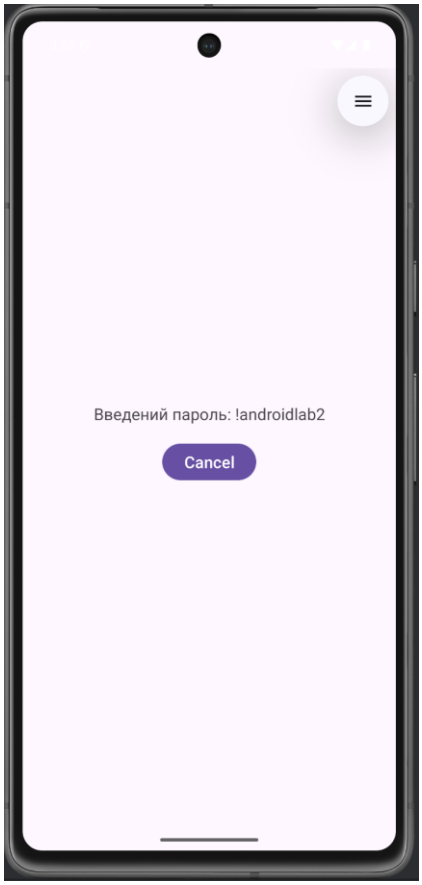
Посилання на код

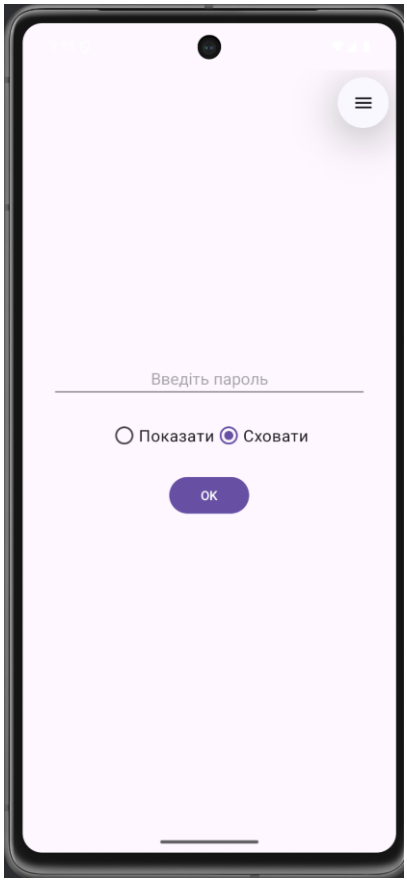
Програма запускається та відразу за замовчуванням текст вводить прихований.





При натисканні кнопки «ОК» переходимо на другий фрагмент, що містить тестове поле з результатом та кнопкою «Cancel». Кнопка «Cancel» очищає другий фрагмент та очищає форму введення з першого фрагменту.





КОНТРОЛЬНІ ПИТАННЯ

1. Призначення та можливості компоненту Фрагмент.

Фрагмент у Android — це модульний компонент із власним життєвим циклом, який представляє частину користувацького інтерфейсу в активності. Фрагменти були введені, щоб краще адаптувати додатки для різних розмірів екранів, особливо для планшетів. Головне призначення фрагментів — це створення модульної архітектури додатку, де окремі частини екрану можуть бути розроблені, повторно використані та керовані незалежно.

Фрагменти дозволяють ефективно використовувати простір екрану: на телефоні фрагменти можуть відображатися послідовно, а на планшеті — одночасно. Вони підтримують власний макет, логіку та навіть меню, але завжди повинні бути інтегровані в активність і не можуть існувати самостійно. Завдяки цьому розробники можуть створювати адаптивні інтерфейси та повторно використовувати компоненти в різних частинах додатку.

2. Життєвий цикл компонента Фрагмент.

Життєвий цикл фрагмента тісно пов'язаний з життєвим циклом активності, але має додаткові етапи. Коли фрагмент приєднується до активності, викликається метод onAttach(), який встановлює зв'язок між ними. Далі йде onCreate(), де фрагмент ініціалізується, але інтерфейс ще не створено. Створення інтерфейсу відбувається в методі onCreateView(), де розробник надає макет фрагмента. Після цього викликається onViewCreated(), де можна взаємодіяти з вже створеними елементами інтерфейсу.
Коли інтерфейс готовий, викликаються методи onStart() та onResume(), які роблять фрагмент видимим та активним. При втраті фокусу фрагментом викликається onPause(), а коли фрагмент стає невидимим — onStop(). Якщо інтерфейс фрагмента руйнується (наприклад, при заміні фрагмента), викликається onDestroyView(). Повне знищення фрагмента супроводжується викликом onDestroy(), а відокремлення від активності — onDetach(). Цей життєвий цикл дозволяє коректно керувати ресурсами та станом фрагмента.

Також важливою відмінністю є те, що фрагмент має метод onDestroyView(), який викликається, коли інтерфейс фрагмента руйнується, але сам фрагмент як об'єкт продовжує існувати. Це дозволяє фрагменту бути відокремленим від свого інтерфейсу, що особливо корисно при заміні фрагментів.

Життєвий цикл фрагмента тісно пов'язаний з життєвим циклом його активності. Фрагмент не може бути у більш активному стані, ніж його активність. Коли активність проходить через свої методи життєвого циклу, відповідні методи викликаються і для всіх її фрагментів. Наприклад, коли активність переходить в стан onPause(), усі її фрагменти також отримують виклик onPause().

3. Способи створення компонента Фрагмент.

Фрагменти в Android використовуються для створення гнучких і багаторазових інтерфейсів. Вони можуть бути створені двома основними способами:

1. **Статичне створення (XML-розмітка)**

Цей підхід передбачає додавання фрагмента безпосередньо у файл макета (.xml). Цей метод підходить, коли структура інтерфейсу відома заздалегідь і фрагмент не потрібно змінювати динамічно.

2. **Динамічне створення (у коді через FragmentManager)**

Динамічне створення використовується, коли необхідно додавати, замінювати або видаляти фрагменти під час виконання програми.

Кроки динамічного створення:

1. Створюємо клас фрагмента (успадковує Fragment або AppCompatDialogFragment).
2. Використовуємо FragmentManager для керування фрагментами.
3. Використовуємо FragmentTransaction для додавання (add()), заміни (replace()) або видалення (remove()) фрагментів.

4. Опишіть способи управління компонентом Фрагмент.

Управління фрагментами здійснюється через FragmentManager, який надає API для маніпуляції фрагментами під час виконання програми. Основні операції включають додавання фрагмента (add()), заміну одного фрагмента іншим (replace()), видалення фрагмента (remove()), а також приховування (hide()) та показ (show()) фрагментів.

Транзакції фрагментів можуть бути анімовані за допомогою стандартних переходів або власних анімацій. Важливою концепцією є back stack — історія транзакцій, яка дозволяє користувачу повертатися назад через кнопку "Назад". Транзакції додаються до back stack методом addToBackStack().

FragmentManager також дозволяє знаходити фрагменти за ID або тегом, що корисно для відновлення після зміни конфігурації. Він надає методи для моніторингу життєвого циклу фрагментів через FragmentLifecycleCallbacks.

Для сучасних додатків рекомендується використовувати однофрагментну архітектуру з Navigation Component, де одна активність містить усі фрагменти, а навігація керується через навігаційний граф.

5. Опишіть способи взаємодії між Фрагментами.

Через Activity – фрагменти взаємодіють через активність (interface для передачі даних).
Через ViewModel – спільний ViewModel, що зберігає стан та передає дані.
Через FragmentResultListener – використовується в FragmentManager для передачі даних між фрагментами.
Через Bundle – передача даних при створенні фрагмента (setArguments()).

6. Наведіть поняття системи, малої системи та мобільної платформи.

Система — це сукупність взаємопов'язаних елементів, що функціонують як єдине ціле. Вона характеризується цілісністю, наявністю структури та ієрархії, взаємодією компонентів та емерджентністю — появою властивостей, які не притаманні окремим компонентам.

Мала система має обмежені ресурси та відносно невелику кількість компонентів, виконує специфічні функції в рамках більшої системи. Мобільні пристрої є прикладом малих систем, оскільки мають обмежені обчислювальні ресурси, пам'ять та енергоспоживання.

Мобільна платформа — це комбінація апаратного та програмного забезпечення для функціонування мобільних додатків. Вона включає операційну систему (Android, iOS), програмні інтерфейси (API), засоби розробки (SDK), апаратні компоненти та інфраструктуру для розповсюдження додатків. Мобільна платформа визначає середовище виконання, доступні ресурси, правила для додатків, їх життєвий цикл та способи взаємодії з користувачем.

7. Опишіть типи мобільних застосунків.

Мобільні додатки можна класифікувати за способом розробки на нативні, кросплатформні, веб-додатки, гібридні та прогресивні веб-додатки (PWA).

Нативні додатки розробляються спеціально для конкретної платформи (Java/Kotlin для Android, Swift/Objective-C для iOS), мають максимальну продуктивність і доступ до всіх функцій пристрою, але вимагають окремої розробки для кожної платформи.

Кросплатформні додатки (React Native, Flutter, Xamarin) дозволяють писати код один раз для різних платформ, але мають обмеження в доступі до нативних функцій і знижену продуктивність.

Веб-додатки — це оптимізовані для мобільних пристроїв веб-сайти, які не потребують встановлення, але мають обмежений доступ до функцій пристрою.

Гібридні додатки — це веб-додатки в нативній оболонці, які можна встановлювати з магазинів додатків та мають доступ до частини нативних API через плагіни.

За функціональним призначенням мобільні додатки поділяються на ігрові, соціальні мережі, додатки для бізнесу та продуктивності, e-commerce, подорожі та навігація, освітні, здоров'я і фітнес, медіа та розваги.

8. Наведіть класифікацію та загальну характеристику середовищ розробки мобільних застосунків.

Середовища розробки (IDE) для мобільних додатків поділяються на платформо-специфічні (Android Studio для Android, Xcode для iOS) та кросплатформні (Visual Studio з Xamarin, VS Code з розширеннями).

За функціональністю IDE можуть бути повнофункціональними (з усіма необхідними інструментами для розробки, тестування і розгортання), спеціалізованими (для конкретних аспектів розробки) або легкими редакторами коду з розширеннями.

Основні характеристики IDE включають інструменти редагування коду (автодоповнення, підсвічування синтаксису, рефакторинг), інструменти побудови UI, інтеграцію з системами контролю версій та збірки, засоби тестування та налагодження, інструменти для розгортання додатків.

9. Наведіть класифікацію та загальну характеристику мобільних платформ.

Основними мобільними платформами є Android (Google) з часткою ринку 70-75% та iOS (Apple) з часткою 25-30%. Також існують нішеві платформи, такі як HarmonyOS (Huawei), та історичні платформи, які більше активно не розвиваються (Windows Phone, BlackBerry OS).

Android — відкрита платформа на базі Linux, з модульною архітектурою, високою адаптивністю до різних пристроїв, але значною фрагментацією версій. Розробка ведеться на Java/Kotlin в Android Studio, додатки поширюються через Google Play.

iOS — закрита платформа від Apple, тісно інтегрована з апаратним забезпеченням, з менш фрагментованою екосистемою та вищим рівнем безпеки. Розробка ведеться на Swift/Objective-C в Xcode, додатки поширюються через App Store.

Платформи відрізняються бізнес-моделлю, підходом до розробки додатків, цільовою аудиторією, методами оновлення системи та рівнем безпеки. Вибір платформи для розробки залежить від цільової аудиторії, бюджету, вимог до функціональності та безпеки.