Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3
на тему «Дослідження способів збереження даних»
З дисципліни «Розробка мобільних застосувань під Android»

Виконала:
студентка 3 курсу
групи ІС-21
Бєльчик Софія

Київ-2025

Мета роботи - дослідити способи збереження даних (база даних, файлова система, тощо) та отримати практичні навички щодо використання сховищ даних.

Варіант - 14

Завдання:
Написати програму під платформу Андроїд, яка доповнює програму, що розроблена за лабораторною роботою 2, роботою зі сховищами.
Тобто при натисканні на кнопку «ОК» додатково:
- здійснюється запис результату взаємодії з інтерфейсом до сховища (файл або базу даних);
- користувач інформується відповідним повідомленням щодо успішності запису.
Також інтерфейс необхідно доповнити кнопкою «Відкрити», натискання на яку призводить до переходу на іншу Діяльність, у якій відображається вміст даних, що зберігаються у сховищі. Якщо дані відсутні (сховище пусте) відобразити відповідне повідомлення.

14.	Вікно містить згорнутий список книг (автори), групу опцій (роки видання), тобто радіо-батони, та кнопку «ОК». Вивести інформацію щодо вибору.

Посилання гітхаб:https://github.com/SofiaBielchik/task3.git

1) Початковий стан (порожнє сховище)

Рис. 1. Екран першого Activity без введених даних

- Інтерфейс першого фрагмента (InputFragment) показує:
    - AutoComplete-поле для введення автора (порожнє).
    - Кнопки «ОК» та «Відкрити» (ще не натискали).
    - Поле textViewResult (порожнє).
- Файл ще не існує або порожній.

2) Натискання «Відкрити» при порожньому сховищі

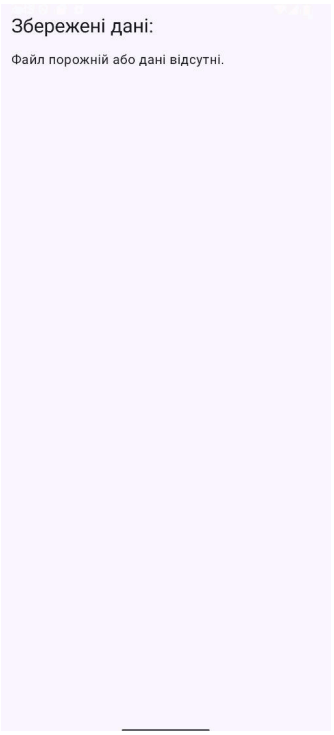

Рис. 2. StorageActivity: «Файл порожній або дані відсутні»

- Коли користувач натискає «Відкрити» без попереднього збереження, запускається StorageActivity.
- Метод displayStoredData() намагається відкрити файл, але отримує виняток (файл не знайдено), тому в TextView відображається: Файл порожній або дані відсутні.

3) Збереження першого запису

Рис. 3. Введено автора «Ліна Костенко» і вибрано рік «1840», після натискання «OK»

- Користувач у полі ввів «Ліна Костенко», позначив радіокнопку «1840» та натиснув «OK».
- З'явився фрагмент ResultFragment, де показано результат.
- Одразу після цього сплив Toast: «Дані успішно збережено».
- У полі textViewResult (у InputFragment) відображено останній запис, щоб користувач бачив свій вибір навіть після повернення.

4) Перегляд збережених даних у StorageActivity

Рис. 4. StorageActivity: перелік усіх збережених записів

- Користувач натиснув кнопку «Відкрити» у InputFragment після кількох збережень.
- StorageActivity відкриває файл books.txt і читає всі рядки.
- Дані виведені у тому порядку, як їх було збережено (append).
- Якщо користувач знову натисне «Back», повернеся до MainActivity та зможе продовжувати вводити й зберігати нові записи.

Висновки:

У лабораторній роботі № 3 було успішно реалізовано механізм збереження даних у внутрішньому сховищі (файлі books.txt) та відображення переліку збережень у новій Activity (StorageActivity).

- При натисканні «OK»
    - Виконується перевірка заповненості полів (автор і роки).

    - Якщо дані неповні — спливає AlertDialog.

Інакше — запис у файл books.txt (режим MODE_APPEND) рядка виду Автор: <author>, Рік видання: <year>

    - Показується Toast — «Дані успішно збережено».
    - Виконується заміна фрагмента на ResultFragment, де відображається останній вибір.

- При натисканні «Відкрити»
    - Запускається StorageActivity, яка читає файл books.txt.
    - Якщо файл порожній або не існує — відображає «Файл порожній або дані відсутні».
    - Інакше — виводить усі збережені записи у тому порядку, як вони додавалися.

Виконано серію скріншотів, що демонструють роботу програми у різних станах. Ця робота покращила навички роботи з файлами та внутрішнім сховищем на платформі Android, а також продемонструвала, як організувати перехід між Activity для відображення даних.

Контрольні запитання:

1) **Як організована робота з налаштуваннями (ключ–значення)?**
   – Використовують SharedPreferences: отримуємо екземпляр через getSharedPreferences("ім’я", MODE_PRIVATE);
   – Запис:
   val prefs = getSharedPreferences("myprefs", MODE_PRIVATE)
prefs.edit().putString("key", "значення").apply()
   – Читання:
val value = prefs.getString("key", "дефолт")
– Підходить для невеликих налаштувань (пара «ключ–значення»), швидкий доступ, автоматичне збереження в приватній папці.

2) Типи файлових сховищ та причини їх використання

    - Internal Storage (приватні файли застосунку, /data/data/<package>): конфіденційні дані, не потребують дозволів.

- External Storage (зовнішня пам’ять / SDкарта / спільна пам’ять пристрою): для великих файлів (фото, відео), доступні іншим програмам, вимагають WRITE_EXTERNAL_STORAGE.
- Cache Directory (getCacheDir() і getExternalCacheDir()): тимчасові файли, які система очищує автоматично.

3) Процес роботи з файлами та файловою системою

1. Запис у Internal Storage:
   openFileOutput("books.txt", MODE_APPEND).bufferedWriter().use {

   it.appendLine("рядок для запису")
}

2. Читання:
   try {
openFileInput("books.txt").bufferedReader().use { reader ->
 val all = reader.readText()
 // якщо empty → повідомлення «файл порожній»
 }
} catch (e: FileNotFoundException) {
 // файл не існує → «порожній або відсутні дані»
}

Файли розташовані у приватній папці застосунку, при видаленні програми — автоматично знищуються.

4) Робота з SQLite через SQLiteOpenHelper

– Створюємо клас, що успадковується від SQLiteOpenHelper:

```
class DBHelper(ctx: Context) : SQLiteOpenHelper(ctx, "books.db", null, 1) {
 override fun onCreate(db: SQLiteDatabase) {
  db.execSQL("CREATE TABLE books(_id INTEGER PRIMARY KEY,
author TEXT, year INTEGER);")
 }
 override fun onUpgrade(db: SQLiteDatabase, old: Int, new: Int) {
  db.execSQL("DROP TABLE IF EXISTS books")
  onCreate(db)
 }
```

}

– Переваги: автоматичне створення та оновлення схеми, централізоване керування версіями.
– Недоліки: доводиться вручну писати SQL-запити, складно підтримувати складні міграції.

5) Робота з БД за допомогою Room

– Entity: позначає таблицю.

@Entity data class Book(@PrimaryKey(autoGenerate=true) val id: Int, val author: String, val year: Int)

– DAO: інтерфейс з анотаціями (@Insert, @Query, @Delete).
– Database: клас, що успадковується від RoomDatabase, містить abstract fun bookDao(): BookDao.
– Переваги: SQL-запити перевіряються під час компіляції, автоматичні міграції (за потреби), інтеграція з LiveData/Flow.
– Недоліки: потрібно вчитися анотаціям, трохи вища складність і розмір APK.

6) Характеристики екранів мобільних пристроїв

– Роздільна здатність (Resolution): кількість пікселів по горизонталі й вертикалі (наприклад, 1080×2400).
– Щільність пікселів (Density): mdpi ≈ 160 dpi, hdpi ≈ 240 dpi, xhdpi ≈ 320 dpi, xxhdpi ≈ 480 dpi тощо.
– Фізичний розмір (Size): в дюймах (діагональ 5.5″, 6.5″ тощо).
– Aspect Ratio: співвідношення сторін (16:9, 18:9, 19.5:9 і т. д.).
– Orientation: портрет (portrait) або ландшафт (landscape).
– Touch-sampling rate: частота опитування сенсорного шару (наприклад, 120 Hz, 240 Hz).

7) Класифікація та відмінності технологій екранів

1. LCD (TFT, IPS): бюджетніший варіант, гарна кольоропередача, трохи вища затримка, енергоспоживання.
2. OLED / AMOLED / Super AMOLED: кожен піксель світиться самостійно, глибокий чорний, висока контрастність, швидкий час відгуку, енергоефективність (особливо при темних темах).
3. Retina (Apple): IPS із високою щільністю (> 300 dpi).
4. High Refresh Rate Displays (90 Hz, 120 Hz): плавні анімації, швидший відгук.

8) Поняття й характеристика сенсорних екранів

– Резистивні: двошарова плівка, працюють через тиск, недорогі, але нижча прозорість і чутливість; не підтримують мультитач.

– Ємнісні (Capacitive): виявлення дотику через зміну ємності; висока чутливість, підтримка мультитач, висока прозорість.

– Surface Acoustic Wave (SAW): ультразвукові хвилі в склі, чутливі до вологи, здатні працювати з рукавичками, але дорогі.

– Infrared: ІЧ-промені між діодами по краях; працюють з будь-яким стилусом/пальцем, але рамка екрану ширша.

9) Загальна класифікація сенсорних екранів

- Резистивні (Resistive)
- Ємнісні (Capacitive)
- SAW (Surface Acoustic Wave)
- Infrared (Optical Touch)
- Active Stylus (Capacitive Stylus)

10) Рекомендації щодо інтерфейсів для сенсорних екранів

- Розмір елементів: мінімум 48 dp × 48 dp, щоб зручно торкатися.
- Відстань між елементами: не менше 8 dp, щоб уникнути випадкових тапів.
- Зворотний зв’язок: ефект ripple, зміна кольору при натисканні, щоб користувач бачив, що елемент активовано.
- Контрастність і читаність: не менше 16 sp для основного тексту, чіткий контраст із фоном.
- Підтримка жестів: tap, swipe, pinch-to-zoom, scroll; усі анімації мають бути плавними.
- Адаптація до різних екранів: використовувати dp/sp, ConstraintLayout, ресурси layout-land/ та layout-port/.
- Material Design: застосування готових компонентів (MaterialButton, TextInputLayout), дотримання гайдлайнів (AppBar, BottomNavigation, FloatingActionButton).