Technical Requirements – Fresh Start Draft

Challenge is accessible on Memberspot: https://pl-coding.mymemberspot.io/library/jx3b7Qik9ip5qpNI8IF2/2BiP9k9ZAdvPhLG5wf7P/siG8KbEAUbi9PHh9v6Sp/details

🎭 Scenario

The app includes a simple screen for creating a new note. The user opens it and starts typing an idea, a reminder, or a quick thought. A Keep draft toggle allows the user to decide whether this draft should be preserved if the app is closed and opened again. In this challenge, the key goal is to make sure the app behaves predictably: the draft is either restored or cleared, exactly as the user expects.

👉 GitHub Repository

The implementation for this mini-challenge is provided in a GitHub repository: https://github.com/PL-Coding-GmbH/Campus-SpringValidation/tree/challenge/fresh-start-draft

The repository contains multiple branches. Each branch corresponds to a **separate mini-challenge** in this series.

Clone the repository, switch to the branch for the current challenge, and work with the existing code. Your task is to focus on writing tests - the implementation itself should not be modified.

🎯 Feature Goal

The goal of this challenge is to write unit tests that validate draft persistence behavior across process death. You must verify that draft content is restored or cleared correctly based on the Keep draft toggle state, and that draft state behaves predictably during app restart. The focus is on persistence logic, not UI implementation.

⚙️ App Behavior Overview

The New Note screen allows the user to create a new note and control whether its draft should be preserved when the app is terminated.

Note Editing

- The screen contains two input fields:
  - Title — a single-line text input;
  - Description — a multi-line text input.
- Both fields are empty by default.
- Any entered text is treated as a **draft** until the note is saved.

Keep Draft Toggle

- The **Keep draft** toggle determines whether the note draft should be preserved after the app is terminated.
- The toggle is **OFF by default**.
- The toggle state affects whether draft data is restored on the next app launch.

Save Action

- The Save Note button completes the note creation flow.
- After pressing Save Note, the draft is no longer considered active.
- The actual note-saving logic is **out of scope** for this challenge.

App Start & Process Death Behavior

- When the app is launched, it checks whether a draft should be restored.
- Draft restoration depends on the Keep draft toggle:
  - if Keep draft is **enabled**, the draft may be **restored**;
  - if Keep draft is **disabled**, the draft is **not restored**.
- This behavior applies after a **process death**, when the app is started again from scratch.

🧪 Validation Tests Requirements

In this mini-challenge, you must write unit tests that verify the draft persistence logic inside the ViewModel.

Process death is simulated by creating a new ViewModel instance using the same persistent storage. All assertions must be performed on the recreated ViewModel, as it represents the restored state after process death.

1️⃣ Initial State

Test name:

- initialState_isEmptyByDefault

Setup:

- Create a new ViewModel instance.

Verify:

- The title value is **empty**.
- The description value is **empty**.
- The keepDraft value is **false**.

2️⃣ Draft Is Not Restored When Keep Draft Is OFF

Test name:

- draft_isNotRestored_whenKeepDraftIsOff

Setup:

- Enter values into title and description.
- Set keepDraft = false.

Action:

- Simulate process death by creating a new ViewModel instance using the same persistent storage.

Verify:

- The keepDraft value remains **false**.
- The title value is **empty**.
- The description value is **empty**.

3️⃣ Draft Is Restored When Keep Draft Is ON

Test name:

- draft_isRestored_whenKeepDraftIsOn

Setup:

- Enter values into title and description.
- Set keepDraft = true.

Action:

- Simulate process death by creating a new ViewModel instance using the same persistent storage.

Verify:

- The keepDraft value remains **true**.
- The restored title matches the **previously entered value**.
- The restored description matches the **previously entered value**.

4️⃣ Keep Draft Toggle - Toggling OFF Preserves Current Input

Test name:

- draft_isNotCleared_whenKeepDraftToggledOff

Setup:

- Enter values into title (e.g., "My Note Title") and description (e.g., "My Note Description").
- Set keepDraft = true.

Action:

- Toggle keepDraft back to false.

Verify:

- The title value remains unchanged.
- The description value remains unchanged.
- The keepDraft value is false.

> 💡 **Note:**
> This test verifies that toggling the "Keep Draft" setting does not clear the current input fields. The toggle should only affect whether the draft is persisted across process death, not the current editing session.

ℹ️ Notes

- Tests must be written as **unit tests**.
- Process death is simulated by creating a new ViewModel instance.
- Test names must be used **exactly as specified**.

🤔 What's Allowed?

- Standard Android / Jetpack libraries.
- Writing unit tests for ViewModel or domain logic.

⚠️ What's not important

- UI testing or UI implementation details.
- How the draft is visually displayed on the screen.
- Writing additional tests beyond those specified in the requirements.
- Covering extra edge cases not mentioned in the challenge.

🔗 Useful Links for This Challenge

- Fundamentals of testing Android apps
- Testing Basics
- The Ultimate Guide to Android Testing

🏆 Submission & Rewards

- Successfully submitting this challenge via the **/submit-challenge** command on Discord grants you **100 XP**.
- Your submission must include:
  a. A **Gist link** with your implementation.
  b. A **screenshot** of the test run results showing:
    - which tests passed,
    - which tests failed,
    - and the names of the executed tests.

> **Note:**
>
> Some tests in this challenge are **expected to fail** due to intentionally incorrect behavior in the app implementation.
>
> Your goal is to write correct tests, not to make all tests pass.

💡

📩 How to Submit a Mini-Challenge

- In any Discord channel, type **/submit-challenge**.
- Attach your screen recording demonstrating the implementation according to the challenge requirements.
- Supported formats: MP4, MOV, AVI, MKV, WEBM, PNG, JPEG, JPG, GIF.
- The total file size must **not exceed** 50 MB.
- If additional materials are required (e.g. screenshots), attach up to 4 additional image files in the command pop-up before submitting.
- Press **Enter** to send the files.
- In the bot flow, select **Mini-Challenge**.
- Choose the month this mini-challenge belongs to (each month includes five mini-challenges).
- Select the exact **challenge name** you are submitting.
- Submit challenge.