Laboraufgaben zum Thema reaktive Programmierung mit Kotlin Flow

Vorbereitung: Clone das folgende GitHub-Repository auf deinen Rechner:
https://github.com/neufst/Learning-Reactive-Programming-With-Kotlin-Flow.git

Aufgabe 1: Theorie

Aufgabe 1.1: Reactive Manifesto

Nenne die vier Prinzipien des Reactive Manifesto und beschreibe diese kurz.

Aufgabe 1.2: Grundlagen Reaktiver Programmierung

Aus welchen drei Bestandteilen (Pattern oder Programmierparadigma) ist die reaktive Programmierung aufgebaut? Erläutere die einzelnen Bestandteile kurz.

Aufgabe 1.3: Operatoren

Nennen vier Operatoren und erläutere die Funktion.

Aufgabe 2: Suspend Functions

Aufgabe 2.1: Starten einer suspend Function

Rufe die Funktion *datasource.getLatestValue* auf und lasse den Wert in der Konsole ausgeben.
Achte darauf, die Signatur der Main-Methode nicht zu verändern.

Aufgabe 2.2: Zeitmessung

Mit der Funktion *measureTimeMillis* lässt sich die benötigte Zeit innerhalb des eingeschlossenen Blocks messen. Zur Erläuterung folgendes Beispiel:

```
val time = measureTimeMillis {
    // run code for time measuring here
}
println("Time needed in milliseconds: $time")
```

Rufe die Funktion *datasource.getLatestValue* nun mehrfach auf und lasse die Werte ausgeben, ohne dass die Ausführungszeit um vielfaches steigt.

Aufgabe 2.3: Asynchrone Verarbeitung

datasource.getLatestValue soll zweifach Aufgerufen werden, dessen Werte anschließend Zusammenaddiert werden sollen. Dabei ist darauf zu achten, dass die Ausführung von *datasource.getLatestValue* parallel geschieht, sodass sich die Ausführungszeit nicht verdoppelt.
Lasse auch hier wieder die benötigte Zeit ausgeben.

Aufgabe 3: Arbeiten mit Kotlin Flow

Bei der Bearbeitung der Aufgaben könnte folgender Auszug aus einem Kotlin Coroutines Cheat Sheet hilfreich sein:

Flow

- Flow is an asynchronous stream of values. Uses `Emit` and `Collect` to send and collect data
- Flow runs in the context that is provided the collector
- The code inside a flow isn't run until the flow is collected
- Flows can be cancelled with `withTimeoutOrNull`
- Useful functions:
  - **transform** - customize data, such as emitting a header first
  - **take** - only take a certain amount of the flow
  - **flowOn** - change the context of preceding code
  - **zip** - combine multiple flows
  - **catch** - catch any exceptions of preceding code
  - **onCompletion** - perform any final tasks after the flow is done

Quelle: *https://dev.to/touchlab/kotlin-coroutines-cheat-sheet-5872*

Aufgabe 3.1: Flowstream ausgeben

Gegeben ist ein Flow, welcher die Fibonacci-Folge nacheinander in einem Flow als Stream emittiert.
Lasse die ersten 12 Fibonaccizahlen in der Konsole ausgeben.

Aufgabe 3.2: Combine

Gegeben sind zwei Flows Numbers und Letters, die unterschiedlich schnell Werte emitieren. Kombiniere diese beiden Flows in einen neuen Flow mit der Funktion *combine* und lasse die Werte ausgeben. Notiere dir die Werte und versuche das Ergebnis (grafisch) nachzuvollziehen.

Tipp: Auf der Seite https://rxmarbles.com werden Reaktive Operatoren grafisch dargestellt. Der hier verwendete Operator ist combine latest

Aufgabe 3.3: Map

Mappen einer Datenbank-Entität in eine Domain-Entität.
Mappe die Datenbankentiät *Person* in die Domainentität *Person*.
Rufe den gemappten Flow mit collect auf und lasse ihn in der Konsole ausgeben.

Aufgabe 3.4: Debounce

Weitergabe einer Texteingabe, wenn eine längere Zeit keine weiteren Eingaben getätigt wurden.

Aufgabe 3.5: Vergleich Flow - Async/Await

Gegeben ist ein Code, welcher Mitarbeiter abruft und diese Aufgaben zuweist. Dieser ist mit Async/Await umgesetzt und soll jetzt stattdessen mit einem Flow umgesetzt werden.

Aufgabe 4: Flow in Android

Vorbereitung: Starte Android Studio und öffne das Projekt ***Aufgabe4***. Nachdem der Gradle-Task durchlaufen ist, führe die App erstmalig aus.

Aufgabe 4.1: MutableStateFlow

Beim Anlegen eines neuen Tasks ist es nicht möglich, Title und Description einzugeben. Stelle sicher, dass im ViewModel der UI-State des AddEditTaskScreen auf den neuen, eingegebenen Wert aktualisiert wird.

Aufgabe 4.2: Transformation eines cold flow in einen hot flow

Beim Drehen des Gerätes wird der View-Lifecycle durchlaufen. In der Task-Ansicht der App kommt es dadurch vor, dass die Tasks nach dem Drehen erneut geladen werden.
Finde eine Lösung, welches dies verhindert.

Aufgabe 4.3: Combine

In der View Task Details werden keine Daten dargestellt.
Erweitere die Combine-Funktion im TaskDetailViewMode, sodass der ausgewählte Task im UiState mit enthalten ist und dargestellt wird.