Lesson 13:

Blur-O-Matic

Instructor: Ahmet Geymen

About this lesson

- Lesson 13:
  - DataStores
  - WorkManagers
  - Workshop
    - Preferences DataStore
    - Blur-O-Matic (Work Manager)

Get started

DataStores

DataStore Types

- Preferences DataStore
- Proto DataStore

Preferences DataStore

- Key-value pairs, without schema
- Async via Coroutines & Flow
- Easy & quick data migrations
- No predefined schema
- No type safety

Proto DataStore

- Stores data as instances of a custom data type.
- Requires defining schema using protocol buffers
- Provides type safety.

WorkManager

WorkManager

- Android Jetpack architecture component
- Recommended solution to execute background work (immediate or deferred)
- Opportunistic and guaranteed execution
- Execution can be based on certain conditions
- Chaining of complex work requests, such as running work in parallel.
- Output from one work request used as input for the next.

When to use WorkManager

- For tasks is not dependent on the app continuing to run after the work is enqueued.
- The tasks run even if the app is closed or the user returns to the home screen.
- Some examples:
  - Periodically querying for latest news stories.
  - Applying filters to an image and then saving the image.
  - Periodically syncing local data with the network.

Important classes to know

- **Worker** - does the work on a background thread, override `doWork()` method
- **WorkRequest** - request to do some work
- **Constraint** - conditions on when the work can run
- **WorkManager** - schedules the WorkRequest to be run

WorkRequests

- Can be scheduled to run once or repeatedly
  - OneTimeWorkRequest
  - PeriodicWorkRequest
- Persisted across device reboots
- Can be chained to run sequentially or in parallel
- Can have constraints under which they will run

Result output from doWork()

Result status	Result status with output
`Result.success()`	`Result.success(output)`
`Result.failure()`	`Result.failure(output)`
`Result.retry()`	

Worker with input and output

```kotlin
class MathWorker(context: Context, params: WorkerParameters):
      CoroutineWorker(context, params)  {

    override suspend fun doWork(): Result {
        val x = inputData.getInt(KEY_X_ARG, 0)
        val y = inputData.getInt(KEY_Y_ARG, 0)
        val result = computeMathFunction(x, y)
        val output: Data = workDataOf(KEY_RESULT to result)
        return Result.success(output)
    }
}
```

WorkRequest Constraints

- setRequiredNetworkType
- setRequiresBatteryNotLow
- setRequiresCharging
- setTriggerContentMaxDelay
- requiresDeviceIdle

Constraints example

```
val constraints = Constraints.Builder()
    .setRequiredNetworkType(NetworkType.UNMETERED)
    .setRequiresCharging(true)
    .setRequiresBatteryNotLow(true)
    .setRequiresDeviceIdle(true)
    .build()

val myWorkRequest: WorkRequest =
OneTimeWorkRequestBuilder<MyWork>()
    .setConstraints(constraints)
    .build()
```

Workshop