Coroutines:

<https://github.com/mdSrahman0/kotlin-coroutines-start>

Coroutines are a way to run long running background applications on a background thread (not the main), without having to implement traditional callbacks. The problem with callbacks is they can become difficult to read and don’t allow for exceptions.

Kotlin allows you to transform callback style code into sequential code, which is far easier to read, as well as having exceptions. We use the keyword suspend to make a function available to a coroutine. When such a function is called, it’s execution is put on hold (suspended) until the result is ready, then it resumes. During this suspension, it unblocks the thread that it’s running on so other functions can run.

Notify Me

<https://github.com/mdSrahman0/NotifyMe>

I learned how to create an application that displays a notification to the user. The notification was created using a notification builder.

There are three buttons: notify me that displays a notification, update me that updates the notification text and icon, and cancel me, which cancels the notification.

Another cool feature was allowing the user to open the application from the notification, even if the application was closed.

NotificationScheduler

<https://github.com/mdSrahman0/NotificationScheduler>

The purpose of this app was to implement a JobService that provides a notification telling the user that the job is running. We allow the user to input constraints to schedule a job.

I created a radio button group, containing three radio buttons. This meant only one selection can be made at a time. I then created two switch views, which set another constraint (do the job only when device is idle or when it’s charging). Lastly, I implemented a seekbar, allowing the user to set a deadline between 0 to 100 seconds to execute the task.

Accessibility

<https://github.com/mdSrahman0/SimpleAccessibility_start>

Before starting this app, I had no idea what talkback was. It is a built-in screen reader that verbally tells a visually impared user what they are interacting with. Clicking an element once will tell the user the description, and clicking it twice will actually select it.

In this app, there are different buttons and an imagebutton that I added talkback functionality to.

SensorSurvey

<https://github.com/mdSrahman0/SensorSurvey>

For this app, we use the sensor manager to find all available sensors, then register listeners for sensor data. Afterward, we create a second app that retrieves data from light and proximity sensors.

Music Player

<https://github.com/mdSrahman0/musicplayer-devices-master>

Given two music files, play the selected song even when the app is closed and the screen is turned off. We used a MediaBrowwerServiceCompat, which is a service that provides media content and playback controls to the user.