Assigning Background Tasks

1) Service & Intent Service ⇒ Traditionally, work in background was done using this technique.

For Communication, need to use Intent.

Explicit intent { Val i = Intent (this, <filename>)
start Activity (i)

You need to have a facility of registering and then running intents, called implicit intent.

How to define an implicit intent to start an activity?

An activity must specify the type of action that it can perform.

ACTION_VIEW
ACTION_SEND } Activities within apps need to specify that they are capable of this action.

If there are multiple activities capable, then the user is given a choice.

Within the activity entry within the manifest file, needs to specify the type of actions that are possible within the "intent-filter" entry.

<activity class= id= >
intent-filter = " ACTION_VIEW }

Intent needs to be specify:

1) Action
2) Location of data ⇒ URI / Location of a file within the file system
3) Type of data (which can possibly be inferred)
4) Additional optional data necessary/useful for an action

```
val intent = Intent(Intent.ACTION_SEND).apply {
        type = "text/plain"
        putExtra(Intent.ExtraText, "...")
        putExtra(Intent.ExtraSubject, "...")
}
startActivity(intent)
```

WorkManager ⇒ Mechanism of doing periodic tasks.

1) Create a Worker class ⇒ defines some task that needs to be done in background
2) Create a WorkRequest ⇒ assigns the task
3) Channel and Notification to inform the user

1) class PWorker (
    val context: Context
    workerParameters : WorkerParameters
): CoroutineWorker (context, workerParameters) {
    override suspend fun doWork (): Result {
        :
        :
    }

context.appScreenSize to display,
and so on

Define the main task done.

Constraints.builder().setRequiredNetwork Type(...).build()

2) val workReq = OneTimeWorkRequest
    .Builder (PWorker::class.java)
    .setConstraints (constraints) .build ()

WorkManager.getInstance (requireContext ()).enqueue (workReq)

The task is scheduled for execution.

There is a way of registering constraints

---

3) Notifications and Channel

Create a channel from an activity to a Notification Manager. Define a Pending Intent, meaning when some task is completed, an Intent would be sent which would lead to a Notification

```
val channel = NotificationChannel(NOTIFICATION_CHANNEL_ID, name, uiforlance)

val nm: NotificationManager = getSystemService(NotificationManager::class)

nm.createNotificationChannel(channel)
```