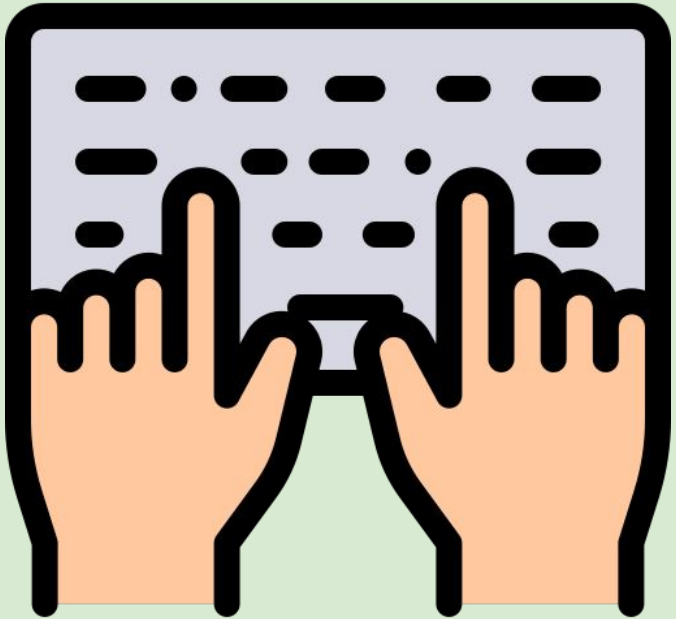
—Lab Notes—

|1/12/2022|



Dr. Xenakis Dionysis

Διαφορές Δραστηριότητας / Fragment

Activity	Fragment
Activity is an application component that gives a user interface where the user can interact.	The fragment is only part of an activity, it basically contributes its UI to that activity.
Activity is not dependent on fragment	Fragment is dependent on activity. It can't exist independently.
we need to mention all activity it in the manifest.xml file	Fragment is not required to mention in the manifest file
We can't create multi-screen UI without using fragment in an activity,	After using multiple fragments in a single activity, we can create a multi-screen UI.
Activity can exist without a Fragment	Fragment cannot be used without an Activity.
Creating a project using only Activity then it's difficult to manage	While Using fragments in the project, the project structure will be good and we can handle it easily.
Lifecycle methods are hosted by OS. The activity has its own life cycle.	Lifecycle methods in fragments are hosted by hosting the activity.
Activity is not lite weight.	The fragment is the lite weight.

ASync

- **onPreExecute()**
  Εκτελείται στο Κύριο Νήμα πριν αρχίσει η λειτουργία παρασκηνίου. Χρησιμοποιείται για την αρχικοποίηση μεγεθών που θα χρησιμοποιηθούν στην εργασία παρασκηνίου(π.χ την αρχικοποίηση μιας μπάρας προόδου).
- **doInBackground()**
  Εκτελείται στο παρασκήνιο και εκεί τοποθετείται όλη η εργασία που πρόκειται να γίνει. Δέχεται τον πρώτο τύπο παραμέτρων της AsyncTask, επί των οποίων θα λειτουργήσει *(π.χ ένα URL για πρόσβαση στο διαδίκτυο)*. Επιστρέφει μία παράμετρο **Result** που μεταφέρεται στην μέθοδο **onPostExecute()** για έξοδο από την εργασία παρασκηνίου.
- **publishProgress()**
  Καλείται από την προηγούμενη και ενημερώνει περιοδικα το Κύριο Νήμα. Οι παράμετροι αυτής της μεθόδου μεταφέρονται στην **onProgressUpdate()**.
- **onProgressUpdate()**
  Ενημερώνει το Κύριο Νήμα όποτε η **doInBackground()** καλεί την **publishProgress()**. Λαμβάνει πληροφορίες από την εργασία παρασκηνίου (π.χ αύξηση της μπάρας προόδου) του δεύτερου τύπου παραμέτρων **(Progress)**.
- **onPostExecute()**
  Εκτελείται από το Κύριο Νήμα όταν ολοκληρωθεί η εργασία παρασκηνίου. Λαμβάνει μια παράμετρο μόλις ολοκληρωθεί η μέθοδος **doInBackground()** (π.χ επιτυχής εργασία) και πραγματοποιεί έξοδο από το παρασκήνιο (π.χ εξαφάνιση της μπάρας προόδου)

Broadcast Receivers

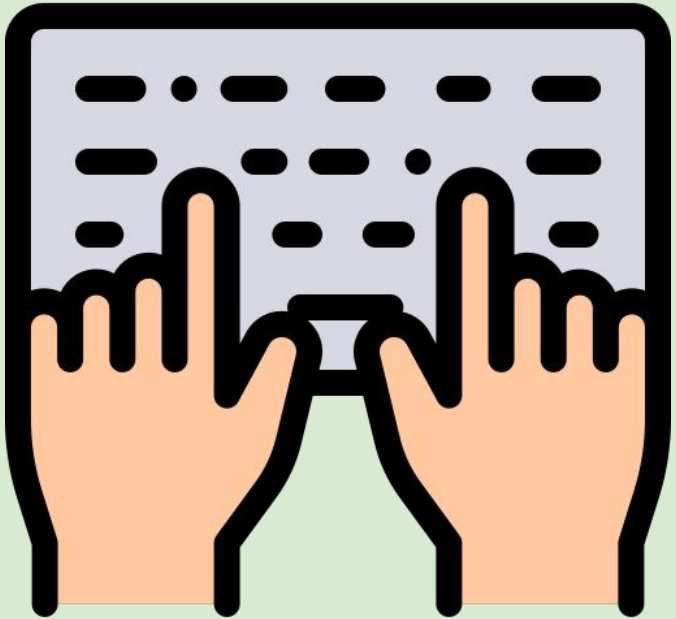
intent? : Η γλώσσα προγραμματισμού Kotlin χρησιμοποιεί το σύμβολο του ερωτηματικού (?) Το χαρακτηριστικό της Null Safety δημιουργήθηκε για να μειώσουν (ή να εξαλείψουν) τον κίνδυνο αναφοράς μηδενικών τιμών(null values) στον πηγαίο κώδικα.

Intent Filters(email)

Η προδιαγραφή Internet RFC 822 ορίζει μια μορφή ηλεκτρονικού μηνύματος που αποτελείται από πεδία κεφαλίδας και ένα προαιρετικό σώμα μηνύματος. Τα πεδία κεφαλίδας περιέχουν πληροφορίες σχετικά με το μήνυμα, όπως τον αποστολέα, τον παραλήπτη και το θέμα

—Lab Notes—

|8/12/2022|



Dr. Xenakis Dionysis

Notifications (1 Channel)

Σημείωση

Χωρίς τη διαμόρφωση των **Καναλιών Ειδοποιήσεων(Notification Channels)**, δεν μπορούμε να δημιουργήσουμε ειδοποιήσεις για εφαρμογές με ***Android API >=26***. Κάθε κανάλι έχει μια συγκεκριμένη συμπεριφορά που θα ισχύει για όλες τις ειδοποιήσεις που αποτελούν μέρος του. Επομένως, κάθε κανάλι θα έχει ένα **αναγνωριστικό καναλιού(Channel ID)** το οποίο θα λειτουργεί ως το μοναδικό αναγνωριστικό για αυτό το κανάλι.

Το **FLAG_UPDATE_CURRENT** καθορίζει ότι, εάν υπάρχει ήδη ένα προηγούμενο PendingIntent, τότε το τρέχον θα το ενημερώσει με το πιο πρόσφατο Intent. Η τιμή 0 είναι ο κωδικός αιτήματος. Εάν το χρησιμοποιήσουμε αργότερα με την ίδια μέθοδο και πάλι θα λάβουμε πίσω την ίδια εκκρεμή Πρόθεση. Για μελλοντική αναφορά η Πρόθεση μεταβιβάζεται εδώ στην Κλάση μας **afterNotification**.

- Status Bar Notification *(2η εικόνα)*
  Εμφανίζεται στην ίδια την Διάταξη *(π.χ ώρα, κατάσταση μπαταρίας)*
- Drawer Notification *(3η εικόνα)*
  Εμφανίζεται στο Drop-Down Μενού
- Heads-Up Notification *(1η εικόνα)*
  Εμφανίζεται στην οθόνη επικάλυψης*(overlay screen, π.χ. ειδοποίηση Whatsapp, FB κτλ)*
- Lock-Screen Notification *(4η εικόνα)*
  Εμφανίζεται στην συσκευή όταν έχει Κλειδωμένη Οθόνη

Notification Manager

Κλάση για να ειδοποιεί τον χρήστη για τα συμβάντα που συμβαίνουν. Έτσι λέμε στον χρήστη ότι κάτι έχει συμβεί στο παρασκήνιο.

Builder

Κλάση Builder για αντικείμενα ειδοποίησης. Παρέχει έναν βολικό τρόπο για να ορίσουμε τα διάφορα πεδία μιας Ειδοποίησης και να δημιουργήσετε προβολές περιεχομένου χρησιμοποιώντας το πρότυπο διάταξης ειδοποιήσεων της πλατφόρμας.

Notifications (2 Channel)

Δημιουργήστε μια δεύτερη Ειδοποίηση στην εφαρμογή σε 2ο κανάλι **(In-Lab)**

Notifications - Expanded

Δημιουργία αναπτυσσόμενης Ειδοποίησης με μήνυμα κειμένου προς τον χρήστη **(In-Lab)**

Accelerometer

```
@SuppressLint("SetTextI18n")
```

είναι ένας σχολιασμός που χρησιμοποιείται από το εργαλείο Android Lint. Το Lint θα σας ενημερώνει όποτε κάτι στον κώδικά σας δεν είναι βέλτιστο ή μπορεί να διακοπεί.

1. **onCreate()**
   Περιέχει ***δύο(2)*** βασικά private arguments (**lateinit var Sensor Manager και square**). Η πρώτη θα είναι *Sensor type* και η δεύτερη *TextView* που ουσιαστικά πρόκειται για το "τετράγωνο σχήμα" που θα εμφανίζονται οι ενδείξεις του επιταχυνσιομέτρου της συσκευής μας. Μέσα σε αυτήν την μέθοδο θα καλείται επίσης και το επόμενο μέλος της κλάσης μας, η μέθοδος δηλαδή που περιγράφεται μόλις πιο κάτω *(setUpSensorStuff)*.
2. **setUpSensorStuff()**
   Εδώ αρχικοποιούμε τον **Sensor Manager** ως *Sensor Service*, προσδιορίζουμε ποιον αισθητήρα θα χρησιμοποιήσουμε **(Sensor.TYPE_ACCELETOMETER)** και κάνουμε **register** τον **Sensor Listener()**.
3. **onSensorChanged()**
   Σε αυτή τη συνάρτηση "τσεκάρουμε" τον αισθητήρα της συσκευής που θα κάνουμε χρήση, δηλώνουμε τις ***μεταβλητές*** και τις ***τιμές*** αυτών που είναι ***απαραίτητες*** για τον ***προσδιορισμό*** της ***κλίσης*** του κινητού μας στους 3 άξονες και τέλος τα ***χρώματα*** αλλαγής του ***TextView*** αναλόγως την ***μετατόπιση*** του.
4. **onAccuracyChanged()**
   Εδώ θα επιστρέψουμε με μια return τις λειτουργίες του αισθητήρα μας.
5. **onDestroy()**
   Τερματισμός τους αισθητήρα μας για μη άσκοπη χρήση αυτού στην συσκευή μας μετά τον τερματισμό της εφαρμογής μας.

```
class MainActivity : AppCompatActivity(), SensorEventListener {
```

Τα μέλη (members) αυτή της κλάσης δημιουργούνται αυτόματα αφού κάνουμε extend την Main με την Sensor Event Listener και απλώς αργότερα τις "γεμίζουμε" με τις λειτουργίες(μεθόδους κτλ) που θέλουμε να υλοποιήσουμε.

StepCounter

.Context

Αυτή είναι μια αφηρημένη κλάση της οποίας η υλοποίηση παρέχεται από το σύστημα Android. Επιτρέπει την πρόσβαση σε πόρους και κλάσεις για συγκεκριμένες εφαρμογές(για Σένσορες στην δική μας περίπτωση)

Τα **επιταχυνσιόμετρα** μετρούν τη γραμμική επιτάχυνση κατά μήκος ενός ή περισσότερων αξόνων. Ένα **γυροσκόπιο** μετρά τη γωνιακή ταχύτητα.

Εργασία για σπίτι

https://www.tutorialspoint.com/android/android_bluetooth.htm

χειρισμό Bluetooth και εμφάνιση λίστας συζευγμένων συσκευών από το Bluetooth