

# PROGRAMMATION DE COMPOSANTS MOBILES (ANDROID)

WIESLAW ZIELONKA



# Alarmes



# Déclencher une alarme

AlarmManager un service d'Android qui gère les alarmes.

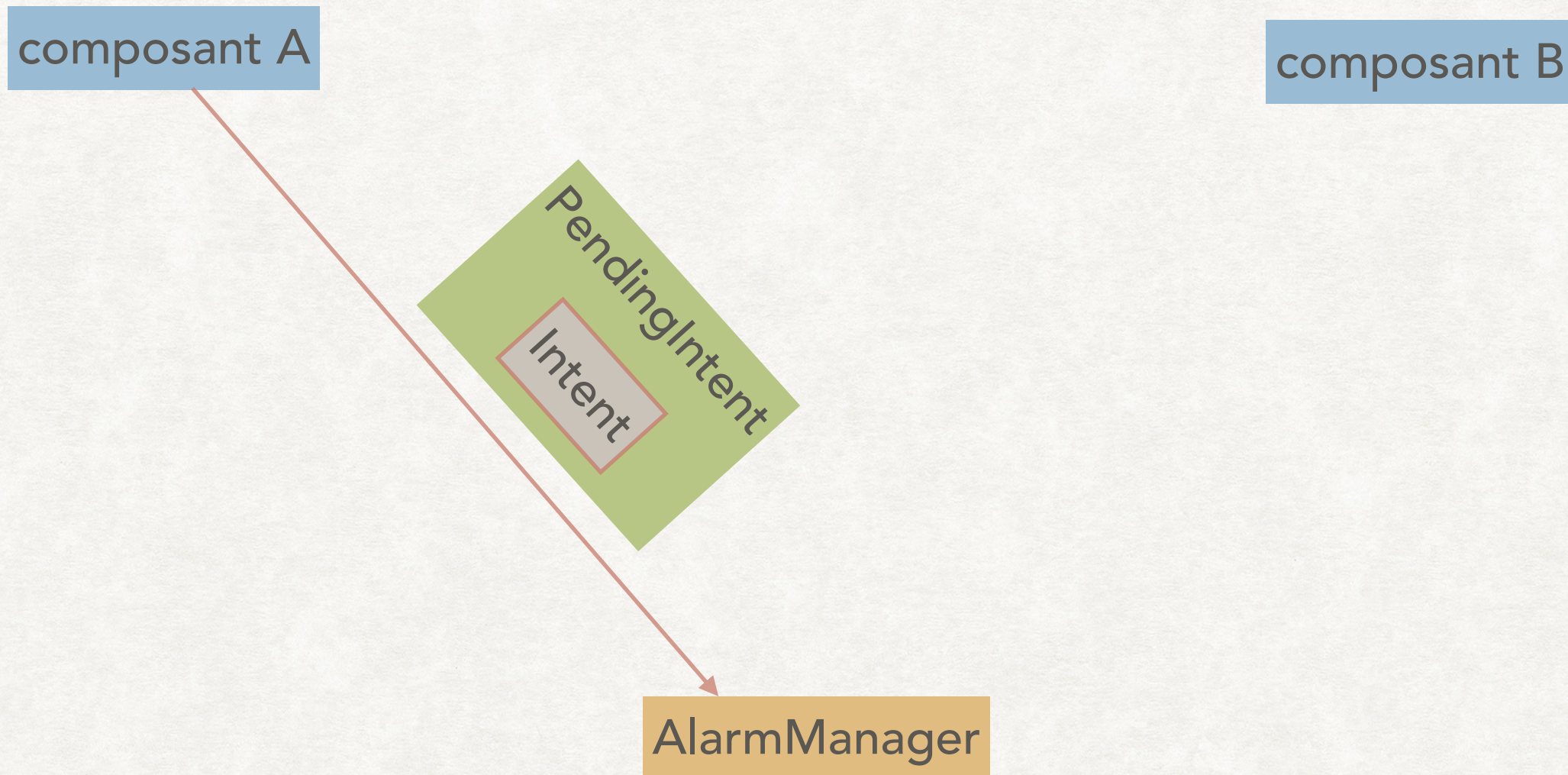
- Amorcer une alarme pour la faire déclencher à un moment donné
- AlarmManager lance l'alarme au moment indiqué et en averti un composant de l'application (Activity, Service, ou BroadcastReceiver)

1. en quoi consiste l'action "lancer alarme" ?
2. en quoi consiste la reception de l'alarme par un composant ?

"Lancer l'alarme" --> AlarmManager lance un Intent à destination d'un composant (ou un Intent envoyé en broadcast).



# amorcer l'alarme





# déclencher l'alarme

composant A

composant B

AlarmManager

Intent

```
graph TD; AlarmManager[AlarmManager] -- Intent --> composantB[composant B];
```

The diagram illustrates the process of triggering an alarm. It features three main components: 'composant A' (a blue box on the left), 'composant B' (a blue box on the right), and 'AlarmManager' (an orange box at the bottom center). A red arrow originates from the 'AlarmManager' box and points towards the 'composant B' box. A grey box labeled 'Intent' is positioned along this arrow, indicating the data being passed. 'composant A' is not directly involved in this specific action.



# alarme

Pourquoi PendingIntent ?

Pour créer les PendingIntent on utilise les méthodes static de la classe PendingIntent :

```
PendingIntent.getBroadcast( context : context,  
reqCode : Int, intent : , flags: Int ) : PendingIntent
```

```
PendingIntent.getActivity( context : context,  
reqCode : Int, intent : , flags: Int ) : PendingIntent
```

```
PendingIntent.getService( context : context,  
reqCode : Int, intent : , flags: Int ) : PendingIntent
```

`pendingIntent.cancel()` invalide un `pendingIntent` (le composant qui détient le `pendingIntent` ne pourra plus utiliser)



# amorcer l'alarme

```
/* intent à mettre dans pendingIntent. AlarmManager va ensuite l'envoyer vers le service MyService */
```

```
val intent = Intent(this, MyService::class.java)
intent.putExtra(MyService.MEDIA_STATE, MyService.START)
```

```
val flag = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
    PendingIntent.FLAG_IMMUTABLE
else
    PendingIntent.FLAG_UPDATE_CURRENT
```

```
/* création de pendingIntent avec le intent dedans */
```

```
val pendingIntent =
    PendingIntent.getService(this, REQUESTED_ID, intent, flag)
```

```
/* récupérer la référence vers AlarmManager */
```

```
val alarmManager = getSystemService(Context.ALARM_SERVICE) as AlarmManager
```

```
/* alarm sera déclenché dans sec secondes (à peu près, ce n'est pas une alarme exacte */
```

```
alarmManager.set(
    AlarmManager.ELAPSED_REALTIME_WAKEUP,
    SystemClock.elapsedRealtime() + sec * 1000,
    pendingIntent
)
```

```
/* SystemClock.elapsedRealtime() retourne le temps depuis le dernier démarrage de l'appareil
```

```
AlarmManager.ELAPSED_REALTIME_WAKEUP indique que le temps compté depuis le dernier démarrage de l'appareil */
```



# amorcer l'alarme

Déclencher l'alarme à une date précise :

// amorcer l'alarme pour 14h00 aujourd'hui

```
val calendar: Calendar = Calendar.getInstance().apply {  
    timeInMillis = System.currentTimeMillis()  
    set(Calendar.HOUR_OF_DAY, 14)  
}
```

// Avec setInexactRepeating(), on demande de répéter l'alarme chaque jour à 14h

```
alarmManager.setInexactRepeating(  
    AlarmManager.RTC_WAKEUP,  
    calendar.timeInMillis,  
    AlarmManager.INTERVAL_DAY,  
    intent  
)
```