Université Paris Cité **Programmation des composants mobiles**
M1 informatique **Année 2022-2023**

TP n° 10

Alarm, Notification, Service

1 Présentation

On va faire une application qui affichera des jolies phrases en zone de notification. Si l'utilisateur s'y intéresse et clique dessus, il obtient une activité qui explique la phrase. L'intérêt de l'application est dans les techniques utilisées :

- L'activité principale contient un `EditText` où l'utilisateur indique la durée x de l'affichage d'une phrase. Pour éviter l'ennui les phrases doivent changer périodiquement, chaque phrase s'affichera pendant x secondes. L'activité principale lance aussi le service `PhraseService` qui affiche les phrases dans les notifications.
- Le service `PhraseService` de type « foreground » lance l'affichage d'une phrase dans les notifications et amorce une alarme qui sera déclenchée dans x secondes.
- Alarme relance le service qui affichera la phrase suivante dans les notifications. Et après l'affichage le service réamorce l'alarme.

L'activité principale et le service vont communiquer par l'intermédiaire de `SharedPreferences`.

2 On commence

Dans `MainActivity` : quand l'utilisateur clique sur le bouton `start service` on récupère la valeur que l'utlisateur entre dans le `EditText` et on la met dans les `SharedPreferences`. On prépare un intent pour lancer le service `PhraseService` que nous allons écrire dans un moment. Sans surprise, on lance le service avec `startService(intent)`.

Pour créer le service vous suivez `File` → `New` → `Service` → `Service`.

Une fois le service crée vous vous placez dedans et vous suivez `Code` → `Override methods` et vous sélectionnez la méthode `onStartCommand`.

C'est dans `startCommand()` que nous implémentons les actions de service.

Le service récupère les phrases (dans le fichier `arrays.xml` que vous devez copier) et récupère l'indice de la phrase suivante à afficher dans les `SharedPreferences`. Donc `SharedPreferences` contient deux informations, la durée x de l'affichage d'une phrase et l'indice de tableau de phrases. Cet indice n'a pas été initialisé donc on ne le retrouve pas initialement dans les SharedPreferences et dans ce cas on prend l'indice 0.

Le service

(1) prépare une notification avec la phrase à afficher,

(2) augmente l'indice de tableau de phrases (modulo la taille du tableau) et sauvegarde la valeur de l'indice dans `SharedPreferences`.

(3) « demande » à `NotificationManager` d'afficher la notification. Rappelons que pour faire afficher une notification le service exécute

```
startForeground( id, notification )
```

où `id` est un identifiant entier (l'identifiant de la notification). L'exécution de cette méthode a aussi un deuxième effet, le service devient un service « foreground ».

(4) et finalement le service amorce une alarme à déclencher dans x secondes (rappelons que la valeur de x vient de `SharedPreferences`). Alarme, quand elle sera déclenchée doit réactiver `PhrasesService`, en quelque sorte `PhrasesService` prépare un alarme qui est destinée à lui-même.

Rappels. Pour que le service puisse fonctionner il faut ajouter une permission dans `AndroidManifest.xml` :

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
```

Pour les notifications il est nécessaire d'indiquer les icônes, sur moodle j'ai mis deux fichiers png, mais si vous trouvez des icônes plus adaptées c'est tant mieux.

Il est bien possible que l'alarme amorcé avec la fonction `set`, comme en cours, ne tient pas la route, on lui demande de se déclencher dans 5 secondes et il se déclenche dans 5 minutes. Remplacer éventuellement par la méthode `setExact()` qui donne l'alarme exacte mais nécessite une permission spéciale

```
<uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />
```

Vérifiez que même si vous tuez `MainActivity` `PhrasesService` ne s'arrête pas et affiche toujours les phrases dans les notifications.

3 Arrêt de service

Il faut donner la possibilité d'arrêter le service. Ajouter un bouton dans la notification qui doit provoquer l'arrêt de service. (Le plus simple : changer la valeur de période x à une valeur négative et ne plus amorcer l'alarme quand x est négatif. On aimerait bien de supprimer aussi le dernier alarme mais ce n'est pas si simple, il faut pour cela le `PendingIntent` qui l'a amorcée.).

4 Page web

Ajouter un deuxième bouton dans chaque notification. Quand l'utilisateur active ce bouton on cherchera dans Wikipedia l'information concernant la phrase affichée.

C'est simple, on prépare intent qui lance le navigateur de façon suivante :

```
val urlPrefix = "https://fr.m.wikipedia.org/wiki/"
val urlSuffix = ....
val webPage : uri = Uri.parse( "$urlPrefix$urlSuffix")
val browserIntent = Intent(Intent.ACTION_VIEW, webPage)
```

où `urlSuffix` est la phrase recherchée avec les espaces remplacés par le caractère `'_'`.

5 Encore mieux ?

Il semble, et il est, absurde d'utiliser les alarmes juste pour faire une action périodiquement.

Une solution naïve : faire une boucle dans `PhraseService` et dans la boucle s'arrêter pour x secondes avec `Thread.sleep( 1000 * x)`.

Créer un répertoire TP09bis dans votre répertoire de projets android et recopier TP10 dans TP10bis à l'aide de la commande `cp -R TP10/* TP10bis`.

Implémentez la « solution » avec `Thread.sleep()` en supprimant les alarmes. Est-ce que ça marche ?

Une autre solution, plus viable ?