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

TP n° 11

Fragments

0

Ce que l'on va faire dans ce TP et le suivant

Dans ce TP, nous allons implémenter une application géographique, avec une seule activité, sans BD pour faire vite, mais avec des fragments.

L'activité utilise trois fragments :

(1) le fragment `ChoixFragment` qui affiche deux boutons permettant de choisir un de deux autres fragments,

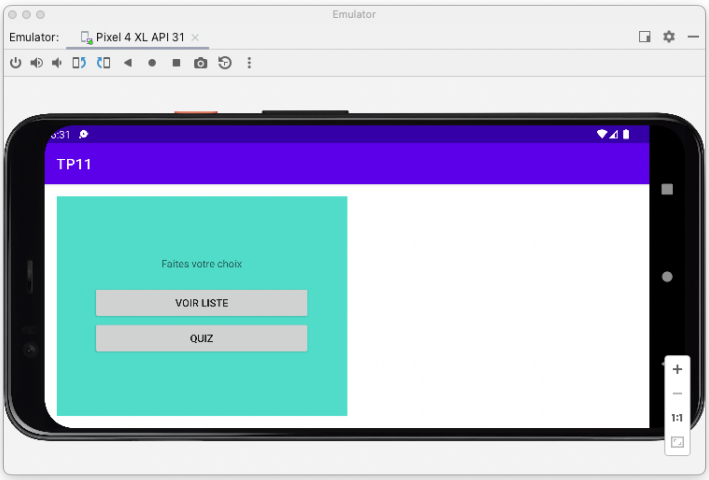
(2) le fragment `PaysFragment` qui affiche la liste de pays avec leurs capitales dans un `RecyclerView`,

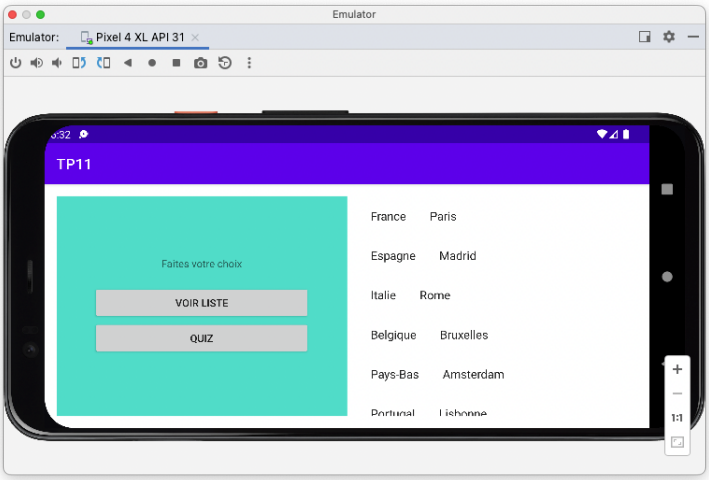
(3) le fragment `QuizFragment` qui permet d'effectuer un quiz de connaissance (quiz de connaissances de capitales).

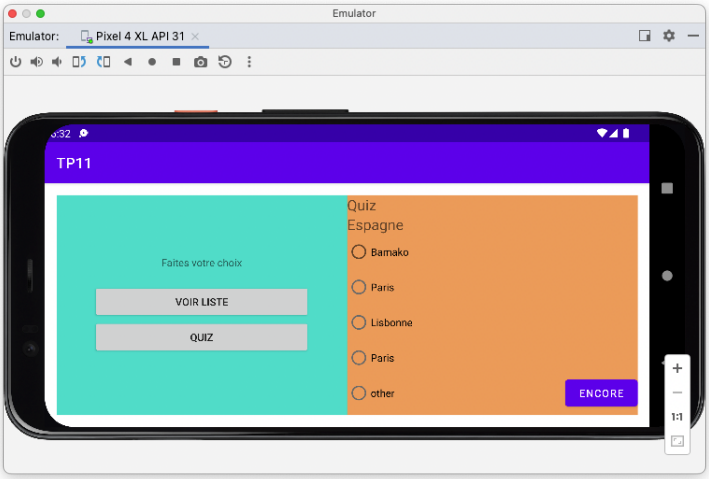
0.1 Mode paysage

Quand le téléphone est en mode paysage, à gauche de l'écran on affichera le fragment `ChoixFragment`. A droite de l'écran initialement on n'affiche rien. L'utilisateur clique sur un bouton de fragment `ChoixFragment` et à droite s'affiche soit le fragment `PaysFragment` soit le fragment `QuizFragment`, en fonction de bouton que l'utilisateur a actionne dans le fragment `ChoixFragment`.

Les images ci-dessous présentent (en mode paysage) l'activité avec juste le fragment `ChoixFragment`, l'activité avec le fragment `ChoixFragment` à gauche et le fragment `PaysFragment` à droite et finalement l'activité avec `ChoixFragment` et `QuizFragment` à droite.


Emulator
Emulator: Pixel 4 XL API 31
TP11
Faites votre choix
VOIR LISTE
QUIZ
1:1


Emulator
Emulator: Pixel 4 XL API 31
TP11
Faites votre choix
VOIR LISTE
QUIZ
France Paris
Espagne Madrid
Italie Rome
Belgique Bruxelles
Pays-Bas Amsterdam
Portugal Lisbonne
1:1


Emulator
Emulator: Pixel 4 XL API 31
TP11
Faites votre choix
VOIR LISTE
QUIZ
Quiz
Espagne
Bamako
Paris
Lisbonne
Paris
other
ENCORE
1:1

0.2 Mode portrait

Quand on sera en mode portrait, on ne verra qu'un fragment, initialement `ChoixFragment`. Quand l'utilisateur clique sur un bouton de ce fragment, le fragment sera remplacé soit par `PaysFragment` soit par `QuizFragment` en fonction de bouton activé par l'utilisateur.

1 Préparation

Créez une application avec une `Empty Activity` comme d'habitude. Dans `build.gradle(module)` activez le `ViewBinding` et ajoutez deux lignes aux dépendances

```
def fragment_version = "1.5.4"
implementation "androidx.fragment:fragment-ktx:$fragment_version"
```

Pour l'activité `MainActivity`, on fournit deux layouts, un pour le mode portrait et l'autre pour le mode paysage : créez le mode paysage en allant sur design et en choisissant l'icône avec le téléphone qui tourne, puis, "create landscape variation" et vous recopiez le code fourni dans le XML.

Ensuite créez une data class Kotlin `Pays.kt` avec deux attributs `nom` et `capitale`.

Créer un nouveau fichier kotlin `ListePays.kt` et copiez dedans le code fourni, regardez un peu le code.

Le fait qu'on ait déclaré un `object` au lieu d'une classe garantit que la variable `liste` ne sera initialisée qu'une fois. On accède à cette liste tout simplement avec `ListePays.liste`

Pour simplifier cette liste est la liste de tous les pays utilisés dans le quiz.

Pour communiquer avec les fragments l'unique activité de l'application utilisera un `ViewModel`, vous pouvez dans la foulée générer une classe `MainViewModel` dérivée de `ViewModel`.

2 Premier Fragment — le choix

Créez le premier fragment `ChoixFragment` (choisissez "Fragment/Fragment (Blank)") et recopiez le layout fourni.

On va maintenant s'occuper de ce premier Fragment puis l'attacher à l'activité.

Comme dans le cours, on construit le fragment avec son layout :

```
class ChoixFragment : Fragment(R.layout.fragment_choix)
```

La méthode `newInstance` générée automatiquement dans le fragment possède deux paramètres, nous en aurions besoin trois donc ajouter le troisième paramètre de type `String`.

Le fragment est composé d'un `TextView` et de deux boutons. Le but de fragment est de signaler à l'activité quand un de bouton est activé par l'utilisateur. Pour que le fragment soit au moins un peu réutilisable on passera le texte de bouton et le texte affiché dans le `TextView` comme le paramètres de `newInstance()`.

2.1 Création de ViewBinding pour le fragment

Comme toujours on préfère `ViewBinding` par rapport aux appels à `findViewById()`.
Il faut réécrire la méthode `onCreateView` [1].

2.2 onCreateView

Dans `onCreateView()` :

```
val v = super.onCreateView(inflater, container, savedInstanceState)
binding = FragmentChoixBinding.bind(v!!)
return v
```

où `binding` est déclaré comme un attribut du fragment :

```
lateinit var binding : FragmentChoixBinding
```

2.3 onViewCreated

Vous devez réécrire aussi la méthode `onViewCreated()` [2]. C'est ici qu'il faut écrire le code qui met le texte dans les boutons et dans le `TextView` – les trois String que `newInstance` a mis dans le bundle `arguments`.

On obtient ce bundle par l'appel à `requireArguments`.

C'est aussi ici qu'on installe les listeners sur les deux boutons.

Quand l'utilisateur clique sur un de ceux boutons de `ChoixFragment` il faut que le fragment informe l'activité.

Et il n'y a qu'un seul moyen de le faire [3]

dans le `ViewModel` de l'activité ajouter un attribut `MutableLiveData<String?>` (ou `MutableLiveData<Int>`, peu importe).

Dans les listeners de boutons `ChoixFragment` met utilise ces objets `MutableLiveData` pour y mettre l'information quel bouton a été activé.

Ensuite il suffit que l'activité installe un observer sur ce `MutableLiveData` pour être informé de l'activation de bouton dans le fragment.

Mais comment le fragment trouve une référence vers `ViewModel` de l'activoité ? Dans le fragment ajoutez un attribut :

```
private val viewModel : MainViewModel by activityViewModel()
```

où `MainViewModel` est le nom de `ViewModel` de l'activité et l'attribut `viewModel` est automatiquement initialisé avec la référence vers le `ViewModel` de l'activité.

1. dans le menu de AndroidStudio : `code` → `Override methods` et choisir la méthode `onCreateView` pour que android studio puisse générer le code
2. faites générer le code par AndroidStudio comme pour la méthode `onCreateView`
3. dans le cours on a indiqué deux possibilité de communications avec l'activité. Il s'avère que la communication par les méthodes callback est désavouée par Google, il reste la méthode par le `ViewModel` de l'activité.

2.4 Affichage de `ChoixFragment`

Dans la méthode `onCreate` de `MainActivity`, faites afficher le fragment `ChoixFragment` (on ne demande pas de gérer le backstack).

3 Deuxième Fragment — la liste de pays

On crée un nouveau Fragment avec "new/Fragment/Fragment(List)". Dans le questionnaire, on remplace "Item" par `Pays`, et comme "Fragment class name" on met `ListePaysFragment` et "finish". On a deux classes engendrées : `ListePaysFragment` et `MyPaysRecyclerViewAdapter` et deux fichiers XML : `fragment_liste_pays_list.xml` et `fragment_liste_pays.xml`. Il a aussi créé un sous-package `placeholder` que l'on supprime.

On remplace `fragment_liste_pays.xml`, qui représente le layout d'un item, par celui fourni.

Ensuite, on va adapter le code Kotlin : Dans `MyPaysRecyclerViewAdapter`,

— on ajoute comme paramètre de constructeur `val listePays : List<Pays>` pour passer la liste de tous les pays,
— on remplace `DummyItem` par `Pays`.
— on corrige `ViewHolder` et `onBindViewHolder` pour afficher le pays

L'adapter doit être créé dans le ViewModel de l'activité.

Dans `ListePaysFragment` on réécrit la méthode `onCreateView` :

— on récupère la référence vers la liste de pays depuis le `ViewModel` de l'activité et on crée adapteur en passant cette liste comme le paramètre de constructeur.
— on attache l'adapter au `RecyclerView` (sans oublier le `LayoutManager`.

Notez que `ListePaysFraglent` n'a pas besoin de paramètres pour être instancié, donc vous pouvez supprimer les paramètres de `newInstance`.

Complétez l'activité pour que l'appui du bouton voir liste, attache un nouveau fragment `ListePaysFragment` et testez (dans toutes les configurations). Vous aurez une liste non cliquable. On y reviendra peut-être.

4 Troisième Fragment — le quiz `QuizFragment`

On démarre

On va faire le fragment de quiz. Lui non plus n'a pas d'argument (pas d'argument de `newInstance()` dans le compagon object).

Par contre vous aurez besoin de `binding` (voir la section 2.1 ou le cours).

`QuizFragment` affichera un pays et la liste de 4 capitales (le cinquième `RadioButton other` permet d'indiquer que aucune ville proposée n'est pas capitale du pays affiché dans le fragment.

Les données pour le quiz seront stockées dans le ViewModel de l'activité. Cela implique que dans le ViewModel de l'activité il faut :

— écrire la méthode `getpays()` qui sélectionne un pays au hasard et le met dans un attribut, appelons-le `currentPays`, qui est `MutableLiveData<pays?>`. `QuizFragment` affichera le nom de ce pays.
— écrire une méthode `getCapitales()` qui sélectionne 4 pays au hasard et met la liste de ce pays dans un attribut de type `MutableLiveData<List<Pays>?>`. `QuizFragment` affichera les capitales de ces pays à côté de RadioButtons

Pour que l'affichage dans `QuizFragment` suive les modifications des objets LiveData décrits ci-dessus il faut que le fragment installe des observers sur ces LiveData.

Le bon endroit pour le faire c'est dans `onViewCreated` du fragment.

Installation d'observeur dans le fragment. Il s'avère que l'installation de l'observers sur un objet LiveData qui est dans le ViewModel de l'activité se déroule un peu différemment :

```
viewModel.correntPays.observe( viewLifeCycleOwner){ //le code de l'observer }
```

c'est-à-dire le paramètre de `observe` ce n'est plus `this`.

5 On affiche

Il est temps de gérer le choix et l'affichage des fragments pour les deux orientations du téléphone.

Sauf initialisation tout se passe dans le observer qui permet de suivre les clicks de boutons

Rappelons que `ChoixFragment` "informe" l'activité de clicks sur un bouton via un MutableLiveData défini dans le ViewModel, voir la section 2.1.

En fonction de l'orientation vous ajouterez `ListPaysFragment` ou `FragmentQuiz` soit dans l'unique frame du layout portrait, soit dans le second frame de layout paysage.

On teste l'orientation avec

```
if (resources.configuration.orientation == Configuration.ORIENTATION_LANDSCAPE)...
```

Si vous voulez tester la présence d'un fragment vous pouvez le faire comme dans le cours avec

```
val fragment = supportFragmentManager.findFragmentByTag( "le_tag")
if( fragment == null )...
```

Ou, mieux, vous pouvez mémoriser la présence d'un quiz dans un attribut.

Si l'affichage des fragments fonctionne pour les deux orientations du téléphone, faites encore un test : lancer le quiz an paysage et tournez le téléphone. Corrigez le problème éventuel.

Vous pouvez revenir à la programmation du Quiz.

On termine le quiz (si on veut)

Il faudra mettre un écouteur sur le bouton "encore", et un autre sur le `RadioGroup`.

Dans le listener, on fait un `when` pour savoir quel bouton radio a été appuyé, on récupère l'indice de la liste correspondant et on compare avec l'indice du pays sélectionné.

Informez l'utilisateur si son choix de capitale est correct.

Gestion de la backStack

On veut que lorsqu'on est en mode portrait, l'appui de la touche "back" élimine le dernier fragment et revienne sur le précédent ou éteigne l'application s'il n'en restait qu'un. On remarque qu'à tout instant, dans la pile, on a soit le menu seul, soit le menu avec un autre fragment dessus.

En mode paysage, s'il y a un fragment à droite, la touche "back" doit laisser la partie droite vide : on n'empile pas de fragments sur la droite (ou plutôt à chaque fois qu'on en met un autre, on dépile le précédent). Quand il n'y a pas de fragment à droite, on éteint l'application.

Gérez la pile et redéfinissez la méthode `onBackPressed()`. On ne gèrera pas ce qui se passe quand le téléphone tourne.

S'il vous reste du temps

On veut que lorsqu'on appuie sur un pays de la liste, on lance un fragment qui permet soit de corriger le nom du pays ou la capitale, soit de le supprimer.