TP
ANDROID - Prise en main - Single Activity App

La première étape consiste à installer Android Studio sur vos machines. Suivez les instructions disponibles ici: `https://developer.android.com/studio`

Exercice 1.1 (Conception d'interfaces - un premier exercice)

1. Créez un nouveau projet `+Create New Project`.
2. Sélectionnez une application Phone and Tablet de type `Basic Activity`. Sélectionnez `Java` en language (ou Kotlin si vous désirez tester ce langage). Utilisez l'API 26 (Oreo) par défaut (d'autres sont possibles mais nécessitent plus de ressources). Le squelette vous fournit une `Activity` et deux `Fragements`. Les fragments sont instanciés par le graphe de navigation `res/navigation/nav_graph.xml`.
3. Dans le `FirstFragment` créez une IHM avec des composants graphiques permettant la saisie des éléments suivants, organisés à l'aide d'un `ConstraintLayout` :
   - nom, prenom
   - date de naissance (en utilisant les widgets fournis par Android)
   - ville de naissance
4. Ajoutez à cette interface un bouton avec le texte "Valider" qui prend toute la largeur de l'écran, et qui soit positionné en permanence en bas de l'écran (vérifiez que c'est bien le cas en modifiant la résolution et/ou l'orientation du terminal).
5. On désire maintenant executer une action particulière lorsque le bouton "Valider" est utilisé. Dans un fragment, plusieurs modalités :
   - soit vous utilisez un listener `View.onClickListener` sur votre bouton. Pour cela, vous récupérez l'objet `View` correspondant à votre bouton via la méthode `findViewById(R.id.votre_id)`. Il faut caster cet objet `View` en objet `Button`. Et ensuite, `Button.setOnClickListener()` doit être appelé sur votre bouton. Dans le listener vous définissez la méthode `onClick()` qui sera appelée lorque le bouton est activé.
   - soit vous utilisez les mécanismes de databinding (cf cours).
6. L'action sur le bouton valider doit consister à afficher à l'écran les données saisies par l'utilisateur à l'aide de la classe `Toast`. Exemple: `Toast.makeText(getActivity().getApplicationContext(), "votre texte", Toast.LENGTH_SHORT ).show();`. Pour récupérer les données saisies dans les champs, il faut accèder aux composants graphiques via la méthode `findViewById()` comme réalisé sur cet exemple :

```
1 public void onViewCreated(@NonNull View view, Bundle saved) {
2         ...
3   EditText text = view.findViewById(R.id.editTextTextPersonName);
4   // get the text
5   String s = text.getText().toString();
```

7. créez ensuite un menu en XML (dans le répertoire `res/menu` cf documentation Android) et permettant:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

Figure 1: Utilisation d'un string-array pour stocker les valeurs d'un composant Spinner.

- l'appel d'une fonction de remise à zéro des données des champs de saisie l'IHM dans le `FirstFragment`.
- l'appel d'une fonction qui fait apparaître un composant graphique permettant la saisie d'un numéro de téléphone, également dans le `FirstFragment`. Pour cela, créez un composant graphique non visible, que vous rendez ensuite visite lors de cet appel.

Pour afficher ce menu spécifique au fragment, il est nécessaire d'ajouter la méthode suivante dans votre `FirstFragment` :

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        // Inflate the menu; this adds items to the action bar
        // if it is present.
        inflater.inflate(R.menu.menu\_fragment1, menu);
        super.onCreateOptionsMenu(menu, inflater);
    }
```

Il faut également ajouter `setHasOptionsMenu(true);` dans la méthode `onCreateView()` du Fragment, qui informe Android qu'un menu de Fragment vient s'ajouter au menu de l'activity.

Enfin, pour appeler des actions relatives à ces items de menu, il faut ajouter `onCreateOptionsMenu()` (cf `MainActivity`).

Exercice 1.2 (Conception d'interfaces - Spinner & Intents)

1. Ajoutez à votre interface la possibilité de saisir le département de naissance. Utilisez un composant de type Spinner.

   - Pour pré-saisir en XML des valeurs dans le composant Spinner, vous pouvez définir l'attribut `android:entries` de votre Spinner: `android:entries="@array/departements"`.
   - La valeur de l'attribut spécifie que les valeurs du Spinner sont stockées dans un tableau nommé `departements` décrit par un fichier XML. Ce fichier XML doit être placé dans le répertoire `res/values` de votre projet et doit être formaté comme dans la figure 1.

2. Ajoutez une option au menu qui permet d'invoquer un browser web en utilisant le mécanisme des Intents implicites (communication entre les applications) pour afficher la page wikipedia correspondant à la ville de naissance de l'utilisateur. Pour utiliser un Intent implicite:

   - il faut spécifier l'action à réaliser et les données sur lesquelles les réaliser. Par exemple l'instruction `Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://fr.wikipedia.org"));` spécifie une action de type "voir" sur l'adresse `"http://fr.wikipedia.org/"`.
   - ensuite, il faut lancer l'intent via la méthode `startActivity(intent);`

Exercice 1.3 (Android View Binding et DataBinding)

Android permet désormais le binding de données avec les composants graphiques via le pattern MVVM. Parcourez la documentation et un ou deux exemples `https://developer.android.com/topic/libraries/data-binding`. Pour le mettre en place :

- ajoutez dans le `build.gradle` de votre module `Basic.Activity.App`:

```
    buildFeatures {
            dataBinding true
        }
    }
```

- Définissez ensuite une classe `Client.java` qui va permettre le stockage de vos données, avec les getter/setter nécessaires. C'est votre Model (dans le pattern MVVM).

- Créez ensuite une classe `ClientViewModel` qui va interfacer la `View` avec le `Model`, comme dans cet exemple :

```
public class ClientViewModel  extends BaseObservable {
    private Client model;
    @Bindable
    public String getName() {
        // access the model
    }
    public void setName(String nom) {
        this.model.nom = nom;
        // BR.name is automatically generated
        notifyPropertyChanged(BR.name);
    }
}
```

- Il est ensuite nécessaire de déclarer le `ViewModel` dans le code du Fragment en XML (dans une balise `<layout>`).

```
   <data>
        <import type="android.view.View"/>
        <variable
            name="viewModel"
            type="com.example.basicactivity.ClientViewModel"/>
    </data>
```

Une classe `FragmentBinding` est générée automatiquement, où `Fragment` represente le nom de votre fichier XML. Dans le fichier XML, l'interface exploite ensuite ce `ViewModel`. `@{viewModel.name}` fait un lien unidirectionnel (model vers vue). `@={viewModel.name}` fait un lien bidirectionnel.

```
   <EditText
        android:id="@+id/editTextTextPersonName"
        android:text="@={viewModel.name}" />
```

- Enfin, le chargement des `Layout` doit être réalisé à l'aide des `inflater` fournis par le mécanisme de databinding (ici dans `onCreateView()`)

```
  FragmentBinding binding = DataBindingUtil.inflate(inflater,
    R.layout.fragment_first, container, false);
  View view = binding.getRoot();
  return view;
```

- Vérifiez enfin que vous pouvez afficher vos données dans votre composant `Toast`.

Exercice 1.4 (Utilisation du deuxième Fragment)

L'objectif est ici de partager les données saisies entre le premier Fragment et le deuxième Fragment. Aidez-vous de l'exemple `ExampleNavigationMMM` dans le share et utilisez votre `ViewModel` pour partager les données entre les deux Fragments.