Laboratoire 6

Les fragments

Objectifs d'apprentissage

- Créer et utiliser les fragments
- Comprendre les fragments statiques
- Comprendre les fragments dynamiques
- Comprendre le cycle de vie des fragments
- Comprendre la communication inter-fragments
- Cibler différents périphériques dans une même application

(Réf. https://developer.android.com/guide/components/fragments.html)

Un fragment représente une portion d'interface que vous pouvez attacher à une activité, il a son propre cycle de vie, gère ses propres événements. Ils peuvent être ajoutés aux activités de façon statique ou de façon dynamique. Vous pouvez combiner plusieurs fragments dans une activité. Les fragments ont été introduits sur Android avec l'arrivée des tablettes sur le marché. Ils permettent de rendre les applications compatibles à la fois sur les smartphones et sur les tablettes. Ils ont permis de mieux exploiter la grande surface offerte par les tablettes.

Un fragment peut être créé en dehors de toute activité mais lui est étroitement lié une fois qu'il y est attaché à cette activité. Son cycle de vie devient dépendant du cycle de vie de l'activité hôte. Le schéma ci-dessous définit le cycle de vie d'un fragment que vous pouvez comparer au cycle de vie d'une activité.

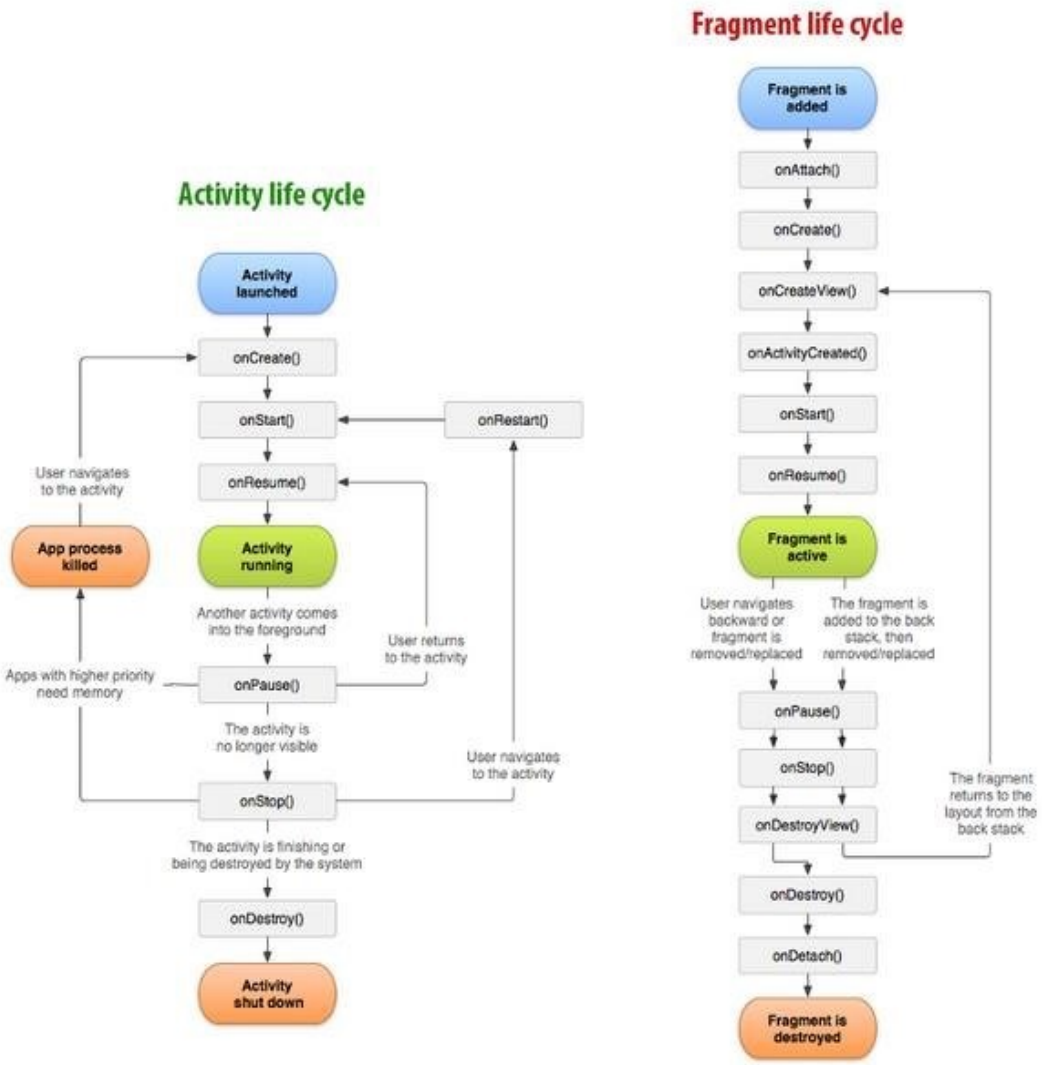
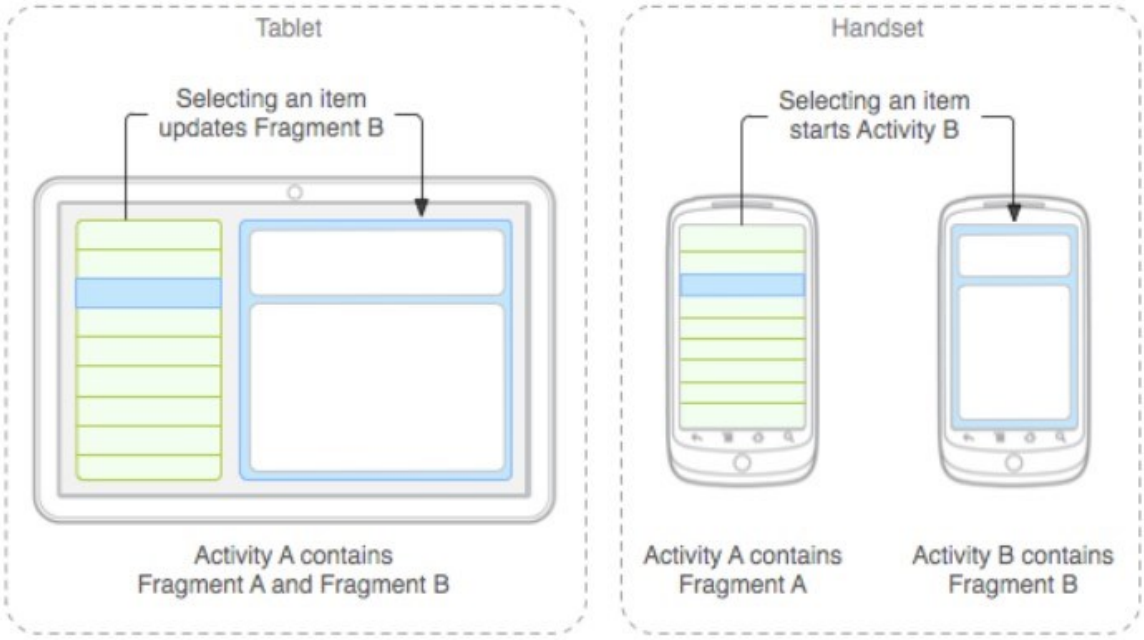


Figure 1: Cycle de vie des Fragments et des activités

Les fragments permettent une réutilisation plus facile dans différentes mises en page. Par exemple, vous pouvez construire des mises en page avec un seul panneau pour des téléphones et plusieurs panneaux pour des tablettes. Ce n'est pas limité aux tablettes, vous pouvez aussi supporter les différentes mises en page suivant l'orientation (portrait ou paysage) du téléphone. Sur une tablette, l'activité principale contient les deux fragments dans sa mise en page, sur un téléphone, elle ne contient que l'activité principale.



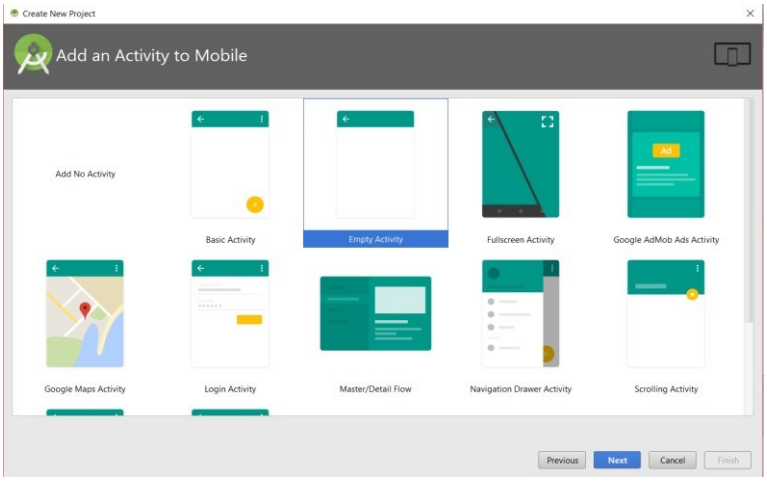
Création d'un Fragment

Pour créer différentes mises en page avec des fragments, vous pouvez utiliser une activité qui affiche deux fragments sur les tablettes et un seul sur les téléphones et utiliser des activités séparées pour héberger chacun des fragments sur un téléphone.
Pour créer un fragment, vous devez créer une classe héritant de la classe Fragment.
La classe Fragment a un ensemble de méthodes de callbacks similaires à la classe Activity, telles que onCreate(), onStart(), onPause() et onStop(). Généralement, vous n'aurez qu'à redéfinir les méthodes onCreate(), onCreateView(), onPause(), onStop(). Il existe d'autres classes héritant de la classe Fragment comme DialogFragment, ListFragment, PreferenceFragment que vous pouvez utiliser aussi suivant vos besoins.

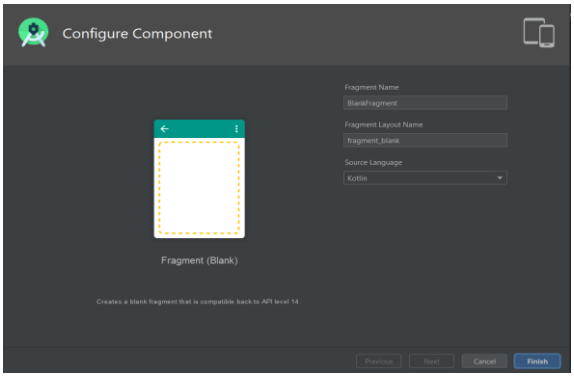
Fragment statique

Dans ce premier exemple que vous allez mettre en place, vous allez utiliser les fragments de façon statique. Vous allez en ajouter deux à votre activité MainActivity.

Créer un projet Laboratoire 6 dans Android Studio en choisissant l'option "Empty Activity".



Créez deux fragments (ColoringFragment et ColoredFragment) en cliquant sur File -> New -> Fragment(Blank) pour ce faire. Vous allez les ajouter de façon statique plus tard à votre activité.



Modifiez le fichier de layout de l'activité en lui ajoutant deux fragments (ColoringFragment et ColoredFragment) dans le sens de la verticale. Le premier occupant la première partie de l'écran(haut) et le deuxième occupant la deuxième partie (bas). Donnez un **identifiant *coloringFragment*** au fragment du haut et ***coloredFragment*** au fragment du bas. Vos fragments doivent remplir la largeur de l'écran. Votre fichier de Layout ressemblera au contenu ci-dessous.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:id="@+id/coloringFragment"
        android:name="ca.qc.cgodin.laboratoire6.ColoringFragment"
        android:layout_width="0dp"
```

```
        android:layout_height="300dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <fragment
        android:id="@+id/coloredFragment"
        android:name="ca.qc.cgodin.laboratoire6.ColoredFragment"
        android:layout_width="0dp"
        android:layout_height="300dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/coloringFragment" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Ajoutez les méthodes de rappel additionnelles aux deux fragments et à l'activité hôte ***MainActivity*** que vous observez sur la ***figure 1*** (***Cycle de vie des fragments et des activités***).
Ajoutez des Logs dans les méthodes de rappel des fragments ainsi que l'activité hôte pour afficher les noms des méthodes appelées avec les noms de classes correspondantes comme aux labos précédents.
Lancez l'application et observez les appels aux méthodes de callback.

Communication entre les fragments

Pour permettre à deux fragments de pouvoir communiquer entre eux, il est très important de passer par l'activité hôte.
Vous allez dans cette partie modifier le fragment du haut que nous appelons fragment colorant (***coloringFragment***) pour envoyer un message au fragment du bas que nous appelons fragment coloré (***coloredFragment***). Quand l'utilisateur va cliquer sur le layout racine (ConstraintLayout) du **fragment colorant**, vous allez envoyer un message au fragment coloré pour lui appliquer une couleur aléatoire provenant du **fragment colorant**. Pour appliquer le principe du couplage faible, vous allez passer par l'activité hôte pour envoyer un message du **fragment colorant** au **fragment coloré**. En vertu de ce principe, vous ne devez **jamais** créer un objet fragment à l'intérieur d'un autre fragment pour lui envoyer un message.
Pour ce faire, vous devez créer une interface dans le fragment colorant qui devra être implémentée par toute activité hébergeant ce fragment pour permettre à un autre fragment de lui envoyer un message comme suit :

```
interface OnColoringFragmentInteractionListener{
    fun onSendColorFragmentInteraction()
}
```

Ajoutez les variables suivantes au début de la classe ***ColoringFragment***

```
private lateinit var mListener: OnColoringFragmentInteractionListener
private val TAG = ColoringFragment::class.java.canonicalName
```

Pour obliger l'activité hôte à implémenter cette interface, redéfinissez la méthode de Callback ***onAttach*** comme suit dans le fragment colorant. Le casting impose sa volonté d'implémentation de l'interface à l'activité.

```
override fun onAttach(context: Context) {
    super.onAttach(context)
    Log.i(TAG, "${javaClass.simpleName}: entered ${object
{}.javaClass.enclosingMethod.name}")
    try {
        // Set the OnColoringFragmentInteractionListener for communicating
with the hosting Activity
        mListener = context as OnColoringFragmentInteractionListener

    } catch (e: ClassCastException) {
        throw ClassCastException("$context must implement
OnColoringFragmentInteractionListener")
    }
}
```

Créez les variables sivantes dans ColoringFragment:

```
private lateinit var mListener: OnColoringFragmentInteractionListener
private val TAG = ColoringFragment::class.java.canonicalName
private lateinit var coloringFragmentLayout: FrameLayout
```

Votre objectif premier est de colorer le fragment de bas à partir du fragment de haut en cliquant sur le layout racine de ce dernier. Vous devez pour ce faire ajouter un listener sur le layout racine du fragment colorant pour envoyer un message au fragment coloré. Dans la fonction onCreateView, vous devez récupérer le layout attaché au fragment et ensuite lui ajouter un listener pour envoyer un message au fragment coloré. Modifiez la fonction onCreateView comme suit:

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    // Inflate the layout for this fragment
    val coloringFragmentLayout = inflater.inflate(R.layout.fragment_coloring,
container, false)

    coloringFragmentLayout.setOnClickListener {
        mListener.onSendColorFragmentInteraction(Random.nextInt())
    }
    return coloringFragmentLayout
}
```

Dans la classe MainActivity représentant l'activité hôte dans ce cas précis, créez un champ coloringFragment de type ColoringFragment et un champ coloredFragment de type ColoredFragment.

```
private lateinit var coloringFragment: ColoringFragment
private lateinit var coloredFragment: ColoredFragment
```

La classe MainActivity doit implémenter l'interface

OnColoringFragmentInteractionListener de coloringFragment. Modifiez la déclaration de la classe MainActivity comme suit :

```
class MainActivity : AppCompatActivity(),
ColoringFragment.OnColoringFragmentInteractionListener {
```

Dans la classe MainActivity, implémentez la méthode onSendColorFragmentInteraction comme suit :

```
override fun onSendColorFragmentInteraction(bgColor: Int) {
    Log.i(javaClass.simpleName, "onSendColorFragmentInteraction")
    coloredFragment.setLayoutBackgroundColor(bgColor)
}
```

Créer un champ du type coloredFragment (fragment coloré) dans MainActivity

```
ColoredFragment coloredFragment;
```

Initialisez les objets coloringFragment et coloredFragment en ajoutant les deux lignes suivantes dans la méthode onCreate:

```
coloringFragment =
supportFragmentManager.findFragmentById(R.id.coloringFragment) as
ColoringFragment
coloredFragment =
supportFragmentManager.findFragmentById(R.id.coloredFragment) as
ColoredFragment
```

Dans le fragment coloré (ColoredFragment), implémentez la méthode ***setLayoutBackgroundColor***:

```
fun setLayoutBackgroundColor(color: Int) {
    coloredFragmentLayout.setBackgroundColor(color)
}
```

Dans la classe ColoredFragment, créez le champ coloredFragmentLayout

```
private lateinit var coloredFragmentLayout: View
```

Initialisez l'objet coloredFragmentLayout dans la méthode onCreateView en la modifiant comme suit

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    // Inflate the layout for this fragment
    coloredFragmentLayout = inflater.inflate(R.layout.fragment_colored,
container, false)
    return coloredFragmentLayout
}
```

Placez un bouton "Remerciements" dans le fragment coloré pour envoyer un message au fragment colorant.

En vous inspirant de l'approche de communication du fragment colorant au fragment coloré, faites l'inverse en envoyant un message du fragment coloré au fragment colorant affichant dans ce dernier un message "Merci de m'avoir coloré".

Modifier le programme pour que ce bouton soit actif dans le fragment coloré après avoir été effectivement coloré par le fragment colorant.

Ajoutez une liste de 7 couleurs au fragment de haut en utilisant un spinner. Modifiez le programme pour que quand l'utilisateur sélectionne une couleur que le fragment de bas prenne cette couleur.