Fragment

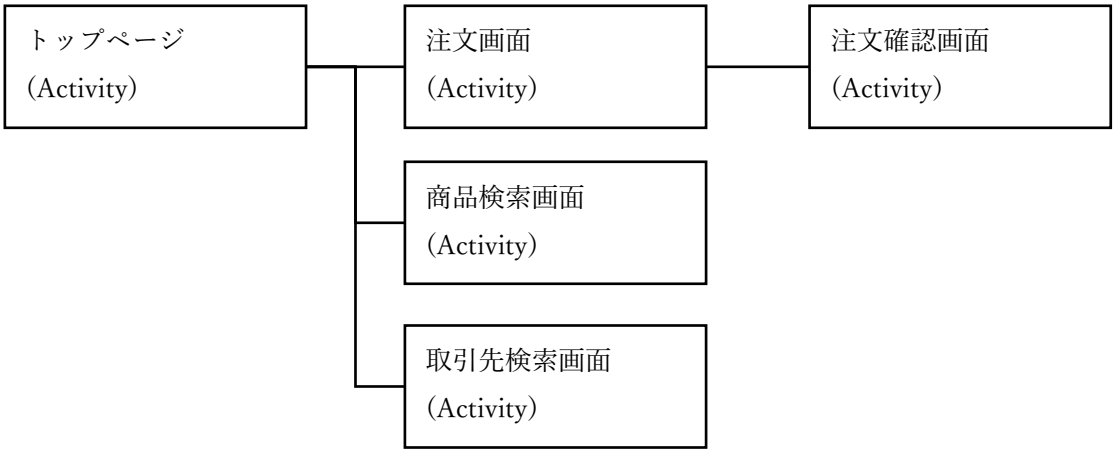
Fragment とは

FragmentAcrivity での UI の挙動や部位を表すものです。1 つのアクティビティに複数のフラグメントを組み合わせてマルチペイン UI を作成したり、複数のアクティビティでフラグメントを再利用したりできます。フラグメントはアクティビティの実行中に追加・削除することができるため、独自のライフサイクルを持ちます。

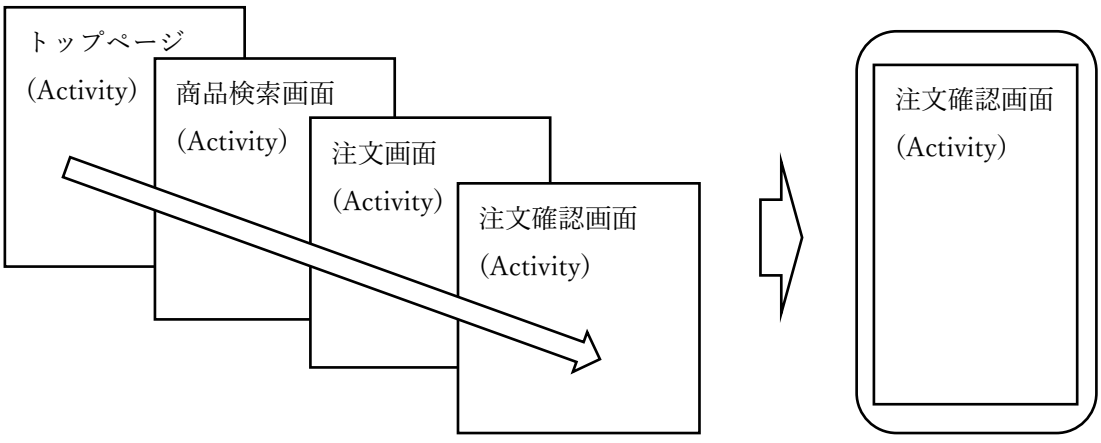
そもそも Android の画面遷移はどうなっている

Android では、画面に対応するプログラムは Activity として処理を行います。その為、複数の画面が必要なアプリケーションはその数だけ Activity を作る必要があります。

・必要な画面分 Activity が必要になる



また、Activity は一つの画面と対応するので、画面遷移を行うと今ある画面に上塗りするように重なりながら画面を展開させていきます。



Activity は 1 つの画面に対応するため、とても分かりやすい仕組みだが共通するデザインの適用や画面の一部分の切り替えなどは出来ないので、同様の処理とデザインを行う Activity を改めて作成する必要がある。

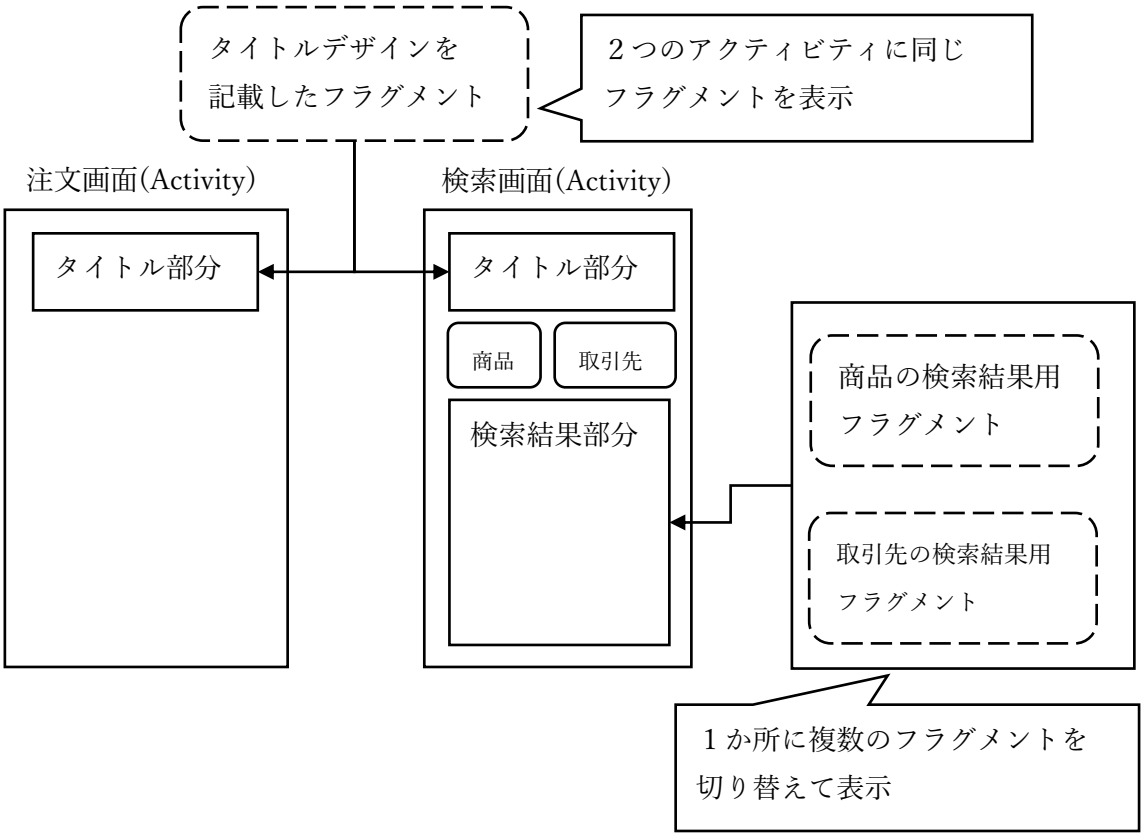
Fragment は Activity で使用する部品

先程の Activity ではできない「共通するデザインの適用」や「画面の一部分の切り替え」を行うために Fragment という機能があります。Fragment とは Activity の中で使う部品で、レイアウトと処理をひとまとめにしたものと言えます。

Fragment の利用方法は主に以下の 2 つです。

1．複数の Activity で 1 つの Fragment を使いまわす。
2．Activity 上の一部を Fragment で入れ替える。

例えば、タイトルなど共通するデザインなどは Fragment にして複数の Activity で呼び出すことで画面ごとにタイトルのデザインをする必要がなくなります。また、検索画面などで Fragment を活用すれば商品検索と取引先検索を別画面に分けなくても 1 画面での実装が可能になります。



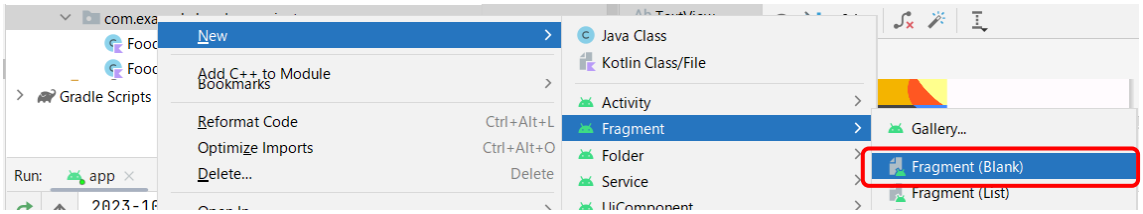
ハンズオン　フラグメントを生成する

1. 新しいプロジェクトを作成する

テンプレート	Empty Views Activity
Name	FragmentMenu
Language	Kotlin
Minimum SDK	API26 (Android8.0)

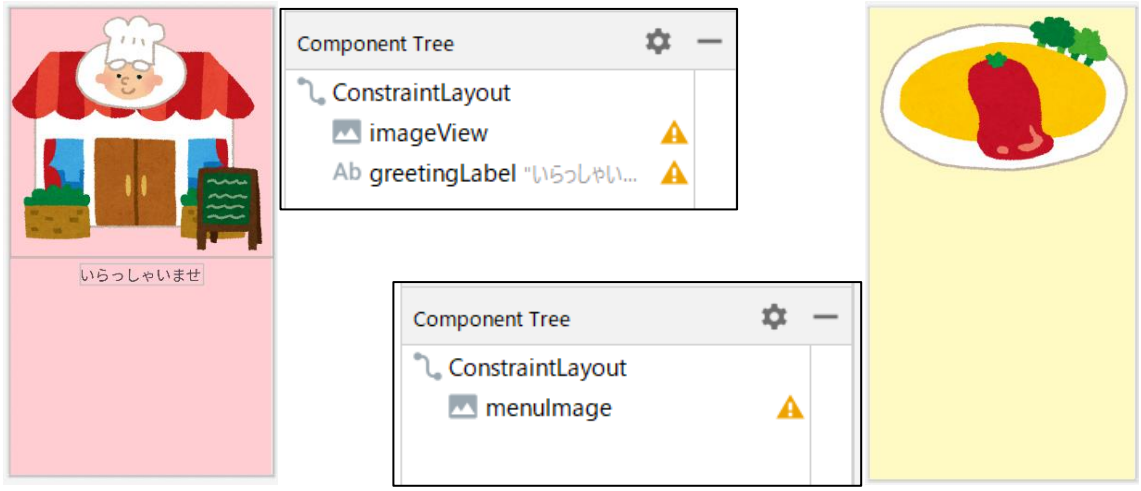
2. アプリで使用する画像をプロジェクトの drawable フォルダに格納する

3. 新しい Fragment を 2 つ生成する



Fragment Name	TopFragment	MenuFragment
Source Language	Kotlin	Kotlin

4. フラグメント画面レイアウトをデザインする



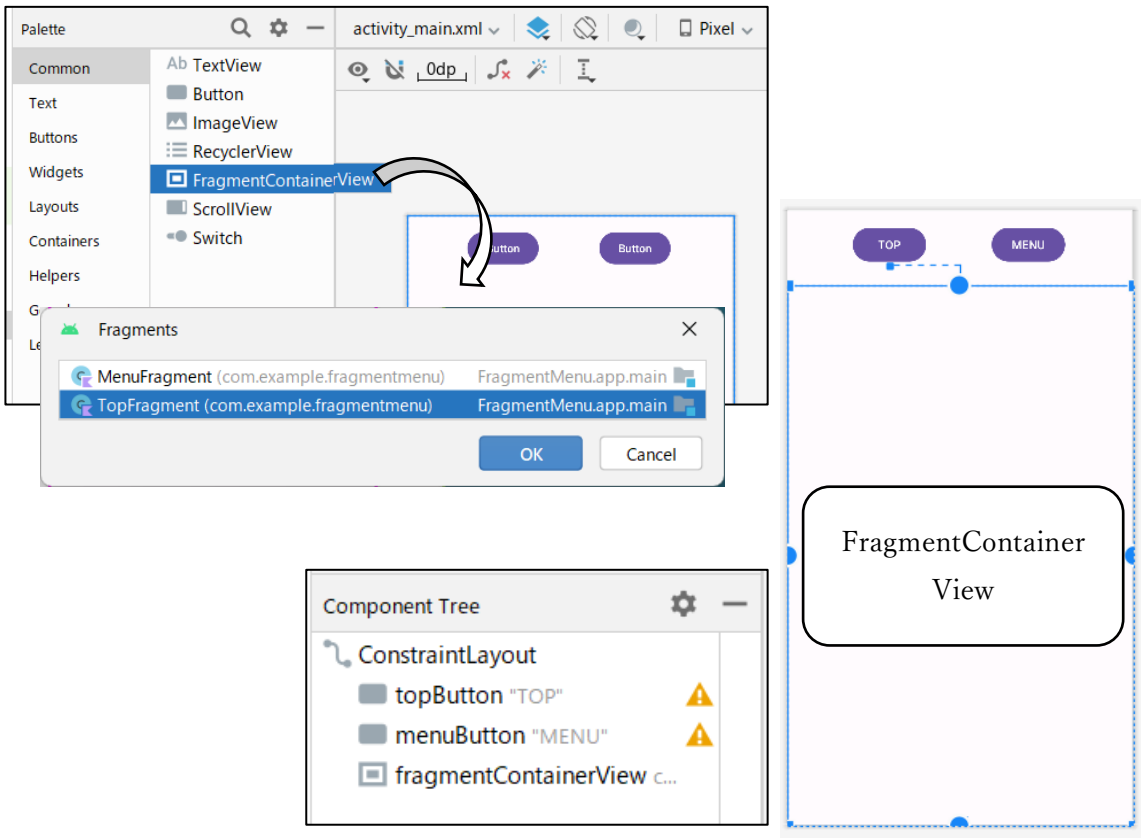
・TopFragment

コンポーネント	属性	設定値
ConstraintLayout	Background	#FFCDD2
ImageView	Id	welcomImage
	adjustViewBounds	True
	srcCompat	@drawable/store
TextView	Id	greetingLabel
	textAppearance	Large
	text	いらっしゃいませ

・MenuFragment

コンポーネント	属性	設定値
ConstraintLayout	Background	#FFF9C4
ImageView	Id	menuImage
	adjustViewBounds	True
	srcCompat	@drawable/omeletrice

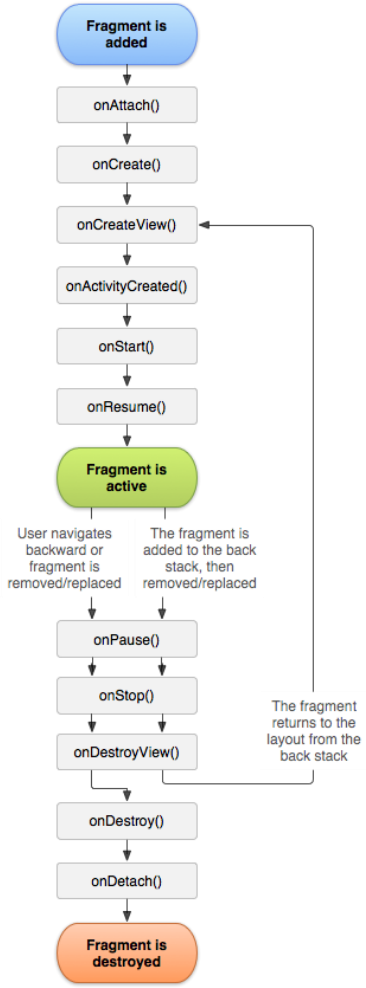
5. アクティビティの画面デザインを行い、フラグメント領域を準備する



コンポーネント	属性	設定値
Button	Id	topButton
	Text	TOP
Button	Id	menuButton
	Text	MENU
FragmentContainerView	Id	fragmentContainerView
	Layout_width	0dp(match constraint)
	Layout_height	0dp(match constraint)

フラグメントのライフサイクル

Fragment には、onCreate や onStop などアクティビティと似たようなライフサイクルメソッドが存在しており、Android OS が適切にライフサイクルを管理しています。
通常は、少なくとも次のライフサイクルメソッドを実装する必要があります。



メソッド	概要
onCreate	フラグメントの作成時にシステムが呼び出します。アプリ内でフラグメントが停止、一時停止して再開されたときにフラグメントの必須コンポーネントを初期化する必要があります。
onCreateView	フラグメントが初めて UI を描画するタイミングでシステムがメソッドを呼び出します。このメソッドからレイアウトを設定した View を戻り値に設定する必要があります。
onPause	ユーザがフラグメントを離れるときにシステムがこのメソッドを呼び出します。(フラグメントが戻ってこない場合があるので) 通常ここで、維持すべき変更点を保存しておきます。

ハンズオン　自動生成されたライフライクルメソッドを整理する

テンプレートをもとに生成されたコードは、パラメータを２つ受け取るように作られています。今回はフラグメント設定時にパラメータが不要のためコードの整理を行います。

1. 宣言されている定数を削除する

```
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"
```

2. クラス変数を削除する

```
private var param1: String? = null
private var param2: String? = null
```

3. onCreate メソッド内の引数取得部分を削除する

```
arguments?.let {
    param1 = it.getString(ARG_PARAM1)
    param2 = it.getString(ARG_PARAM2)
}
```

4. シングルトンのインスタンスメソッドの引数設定部分を削除する

```
putString(ARG_PARAM1, param1)
putString(ARG_PARAM2, param2)
```

5. インスタンスメソッドの引数を削除する

```
@JvmStatic
fun newInstance() =
```

フラグメントをアクティビティに追加する

フラグメントは、アクティビティに UI の一部を提供し、アクティビティの全体的なビュー階層の一部として埋め込まれます。また、アクティビティの実行中はいつでもフラグメントをアクティビティレイアウトに追加できます。アクティビティ内でフラグメントの管理をするには、FragmentManager を使用する必要があります。そこから FragmentTransation でトランザクション（追加、削除、置換など）を実行するができます。

ハンズオン　フラグメントの切り替えを行う

ここでは MENU ボタンをクリックしたときにフラグメントの内容を切替えます。

1. MENU ボタンのクリックイベントリスナーを実装する

```
menuButton = findViewById(R.id.menuButton)
menuButton.setOnClickListener{
}
```

2. クリックイベントで MenuFragment のオブジェクト変数を宣言する

```
val menuFragment = MenuFragment.newInstance()
```

3. フラグメントマネージャーからトランザクションを呼び出す

```
val transaction = supportFragmentManager.beginTransaction()
```

4. フラグメントの置換処理を行う

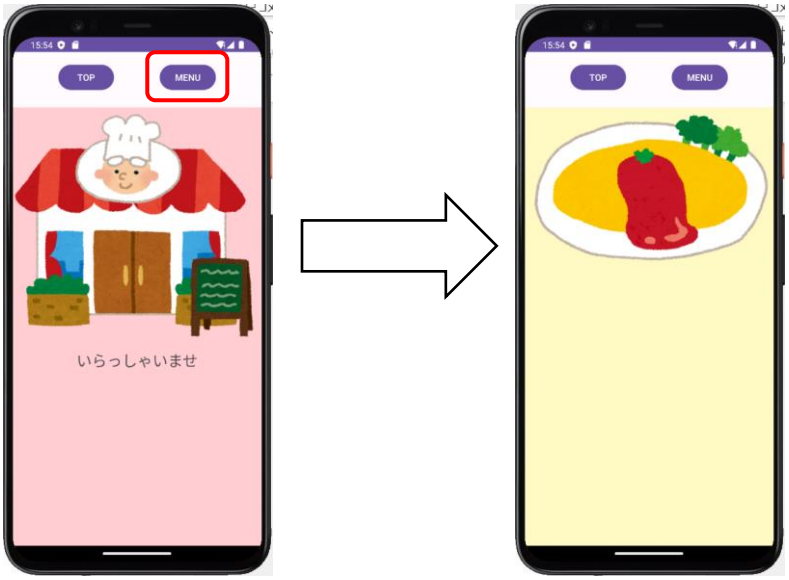
```
transaction.replace(R.id.fragmentContainerView, menuFragment)
```

5. バックスタックにフラグメントを格納する

```
transaction.addToBackStack(null)
```

6. 変更内容を確定させる

```
transaction.commit()
```



切り替られた Fragment はどうなる？

Fragment は、切り替えられるので Activity のようにライフサイクルを持ち、バックスタックも利用可能になっています。バックスタックに入った Fragment は必要に応じて取り出せるため再生成する必要が無い場合に便利です。