

FRAGMENT

CUSTOMVIEWGROUP

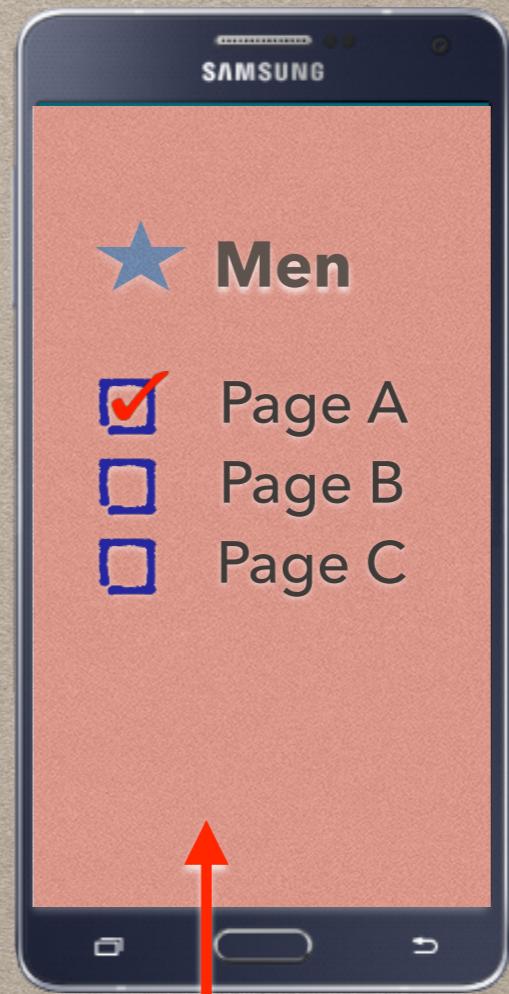
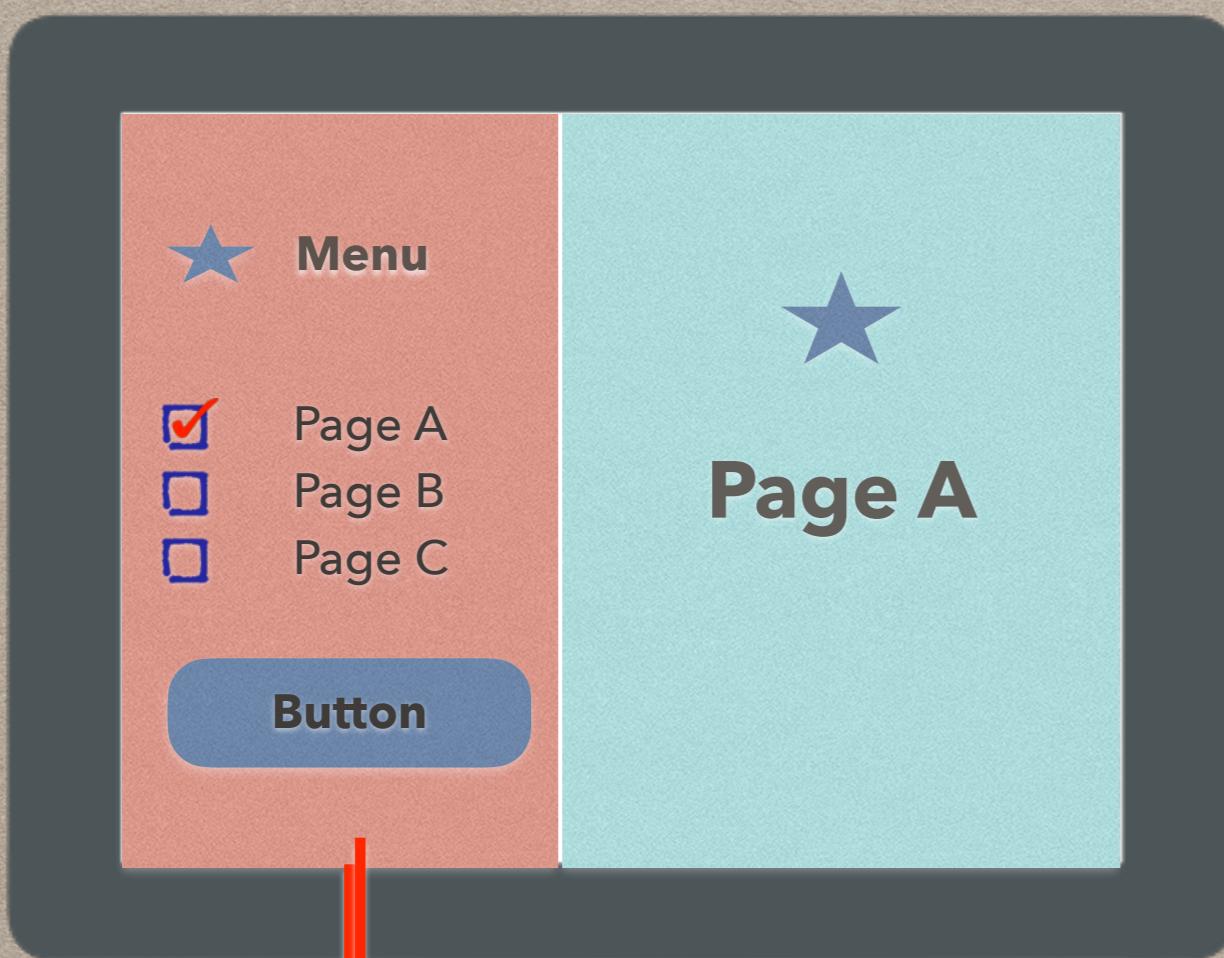
DESIGN PHILOSOPHY

- Android introduced fragments
in Android 3.0 (API level 11)
- primarily to support more dynamic and flexible UI designs on large screens

WHAT IS FRAGMENT ?

Fragment class in Android is used to build dynamic User Interfaces.

DESIGN PHILOSOPHY



Copy

- Layout

- Java source

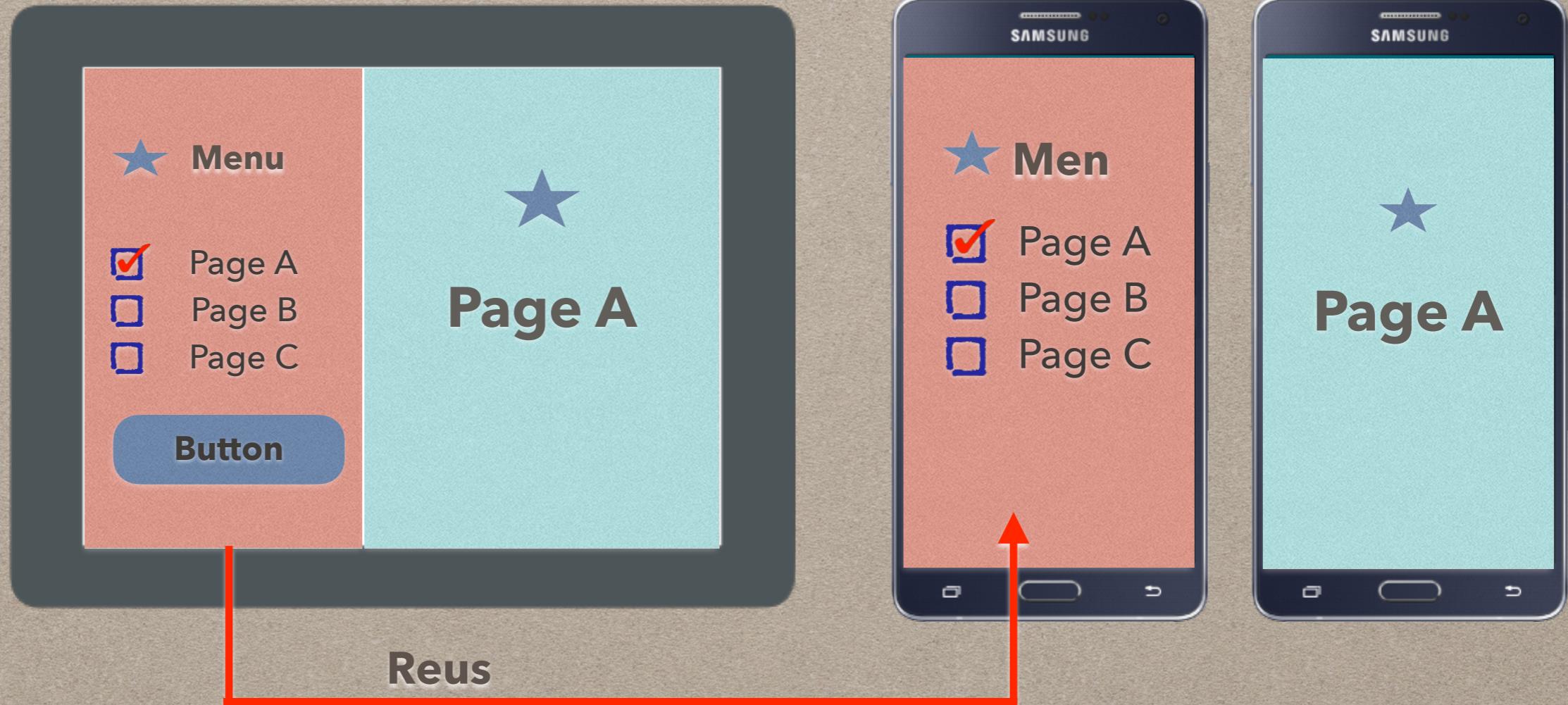


Reuse

- Event
- Logic

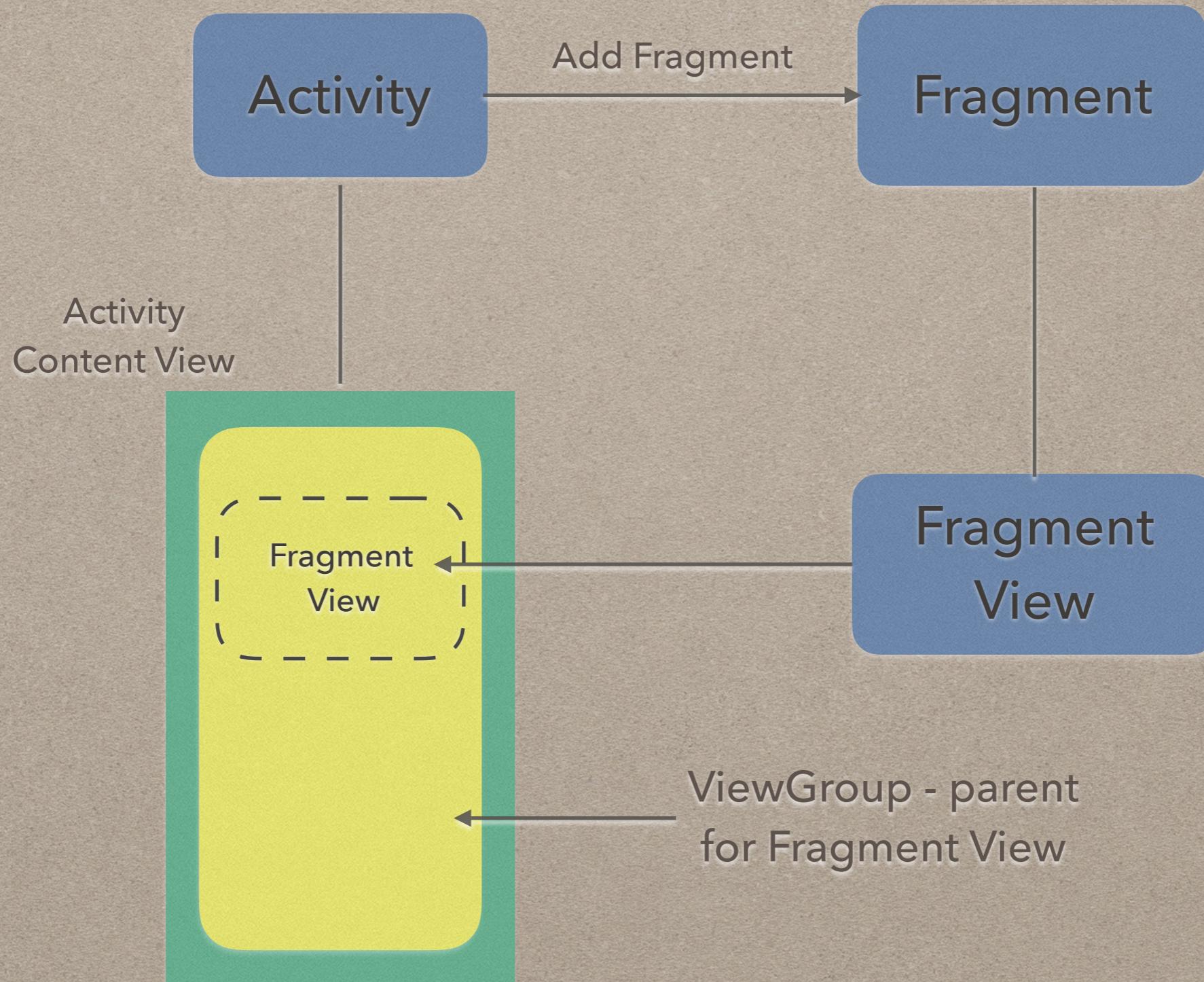


DESIGN PHILOSOPHY

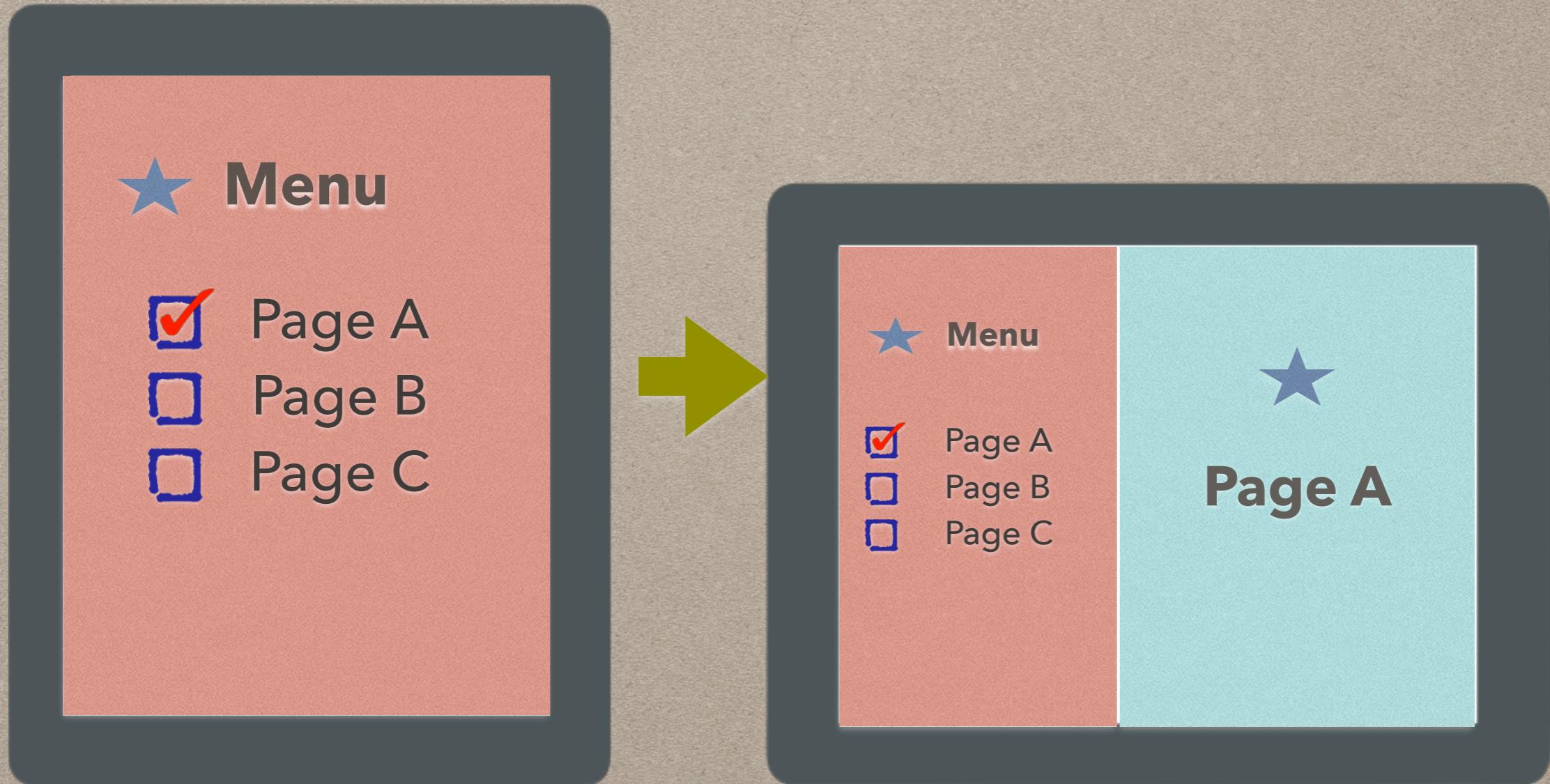


ใช้ Fragment เข้ามาช่วยให้ Reuse ได้อย่างมีประสิทธิภาพ

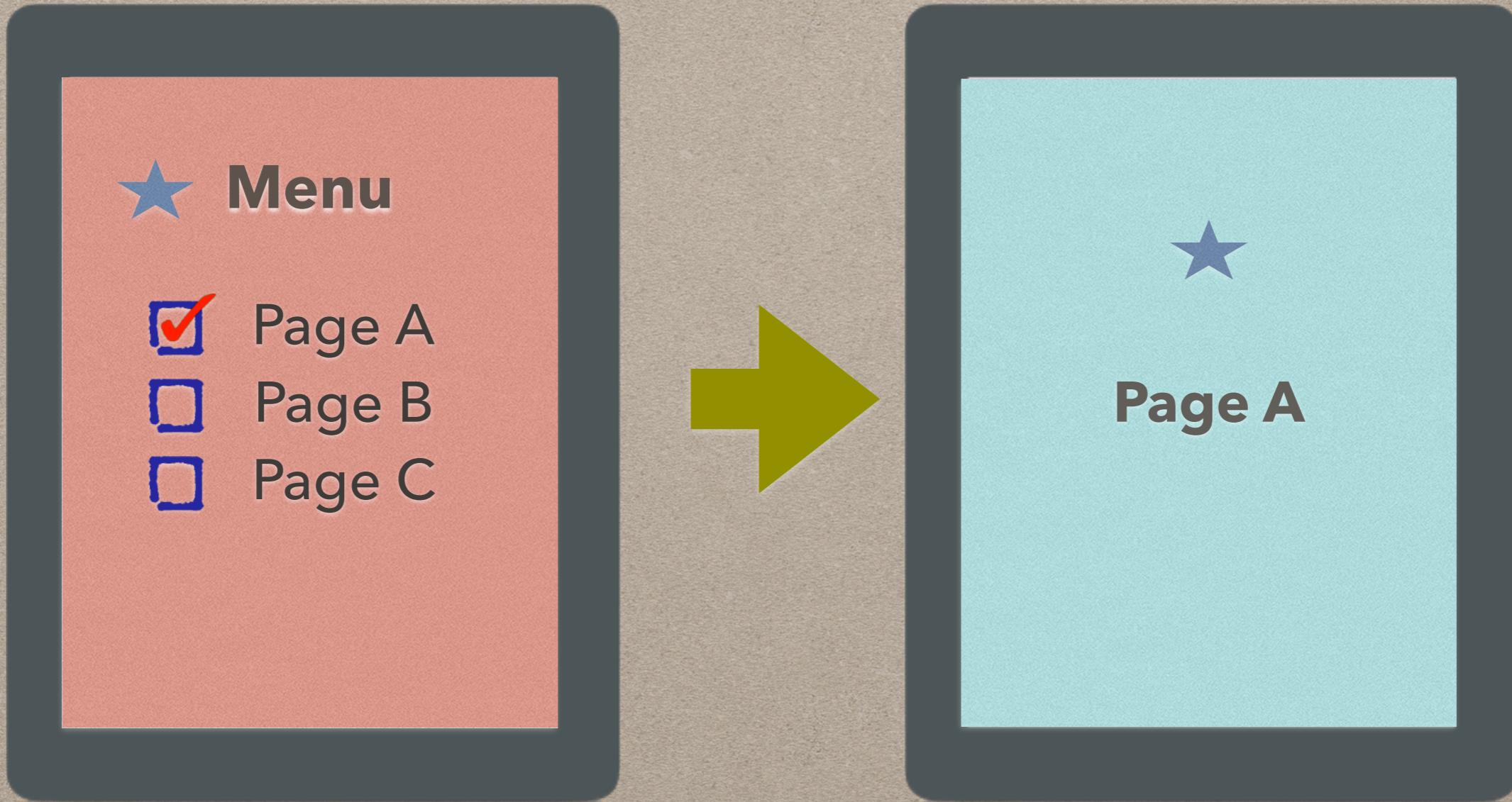
WHAT IS FRAGMENT ?



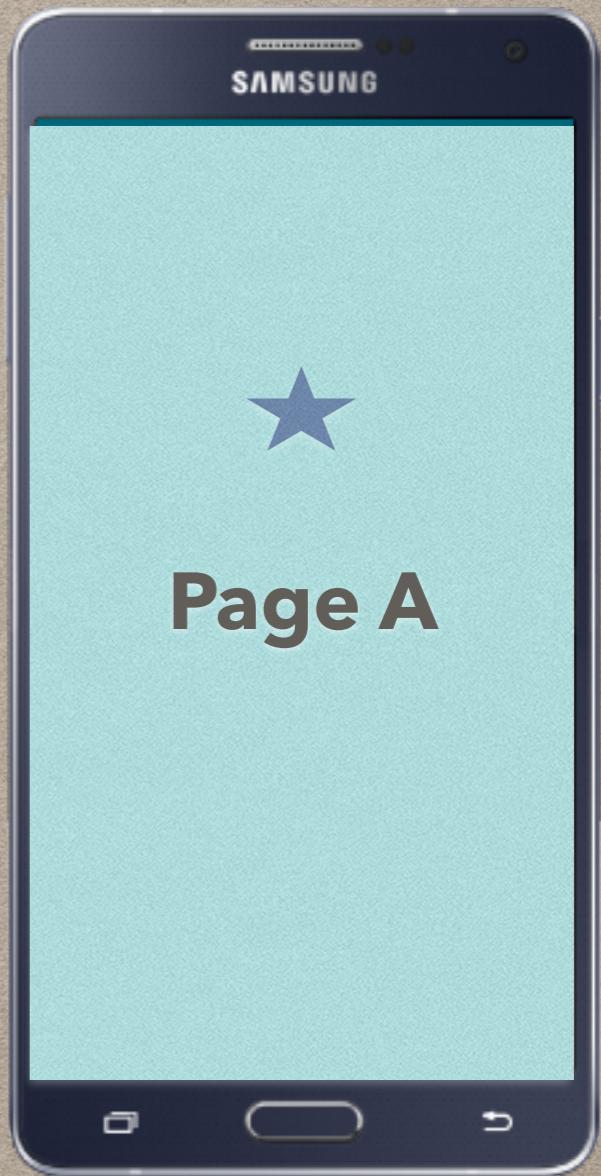
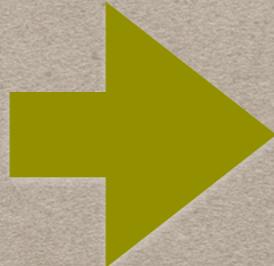
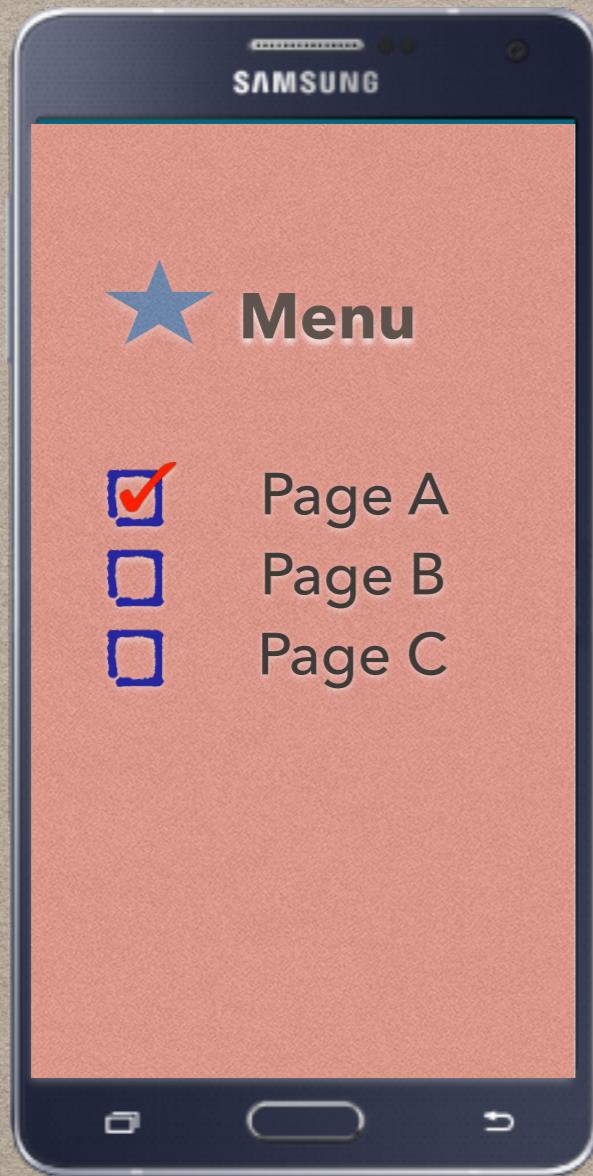
DESIGN PHILOSOPHY



DESIGN PHILOSOPHY



DESIGN PHILOSOPHY



สามารถมองและแบ่ง FRAGMENT ได้

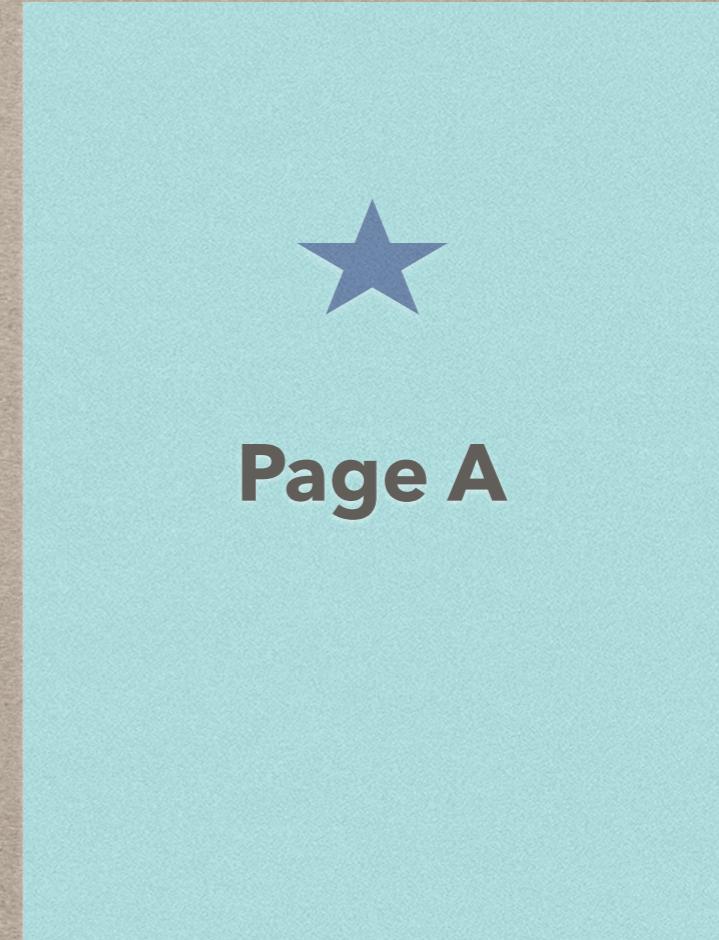
★ Menu

- Page A
- Page B
- Page C

MenuFragment

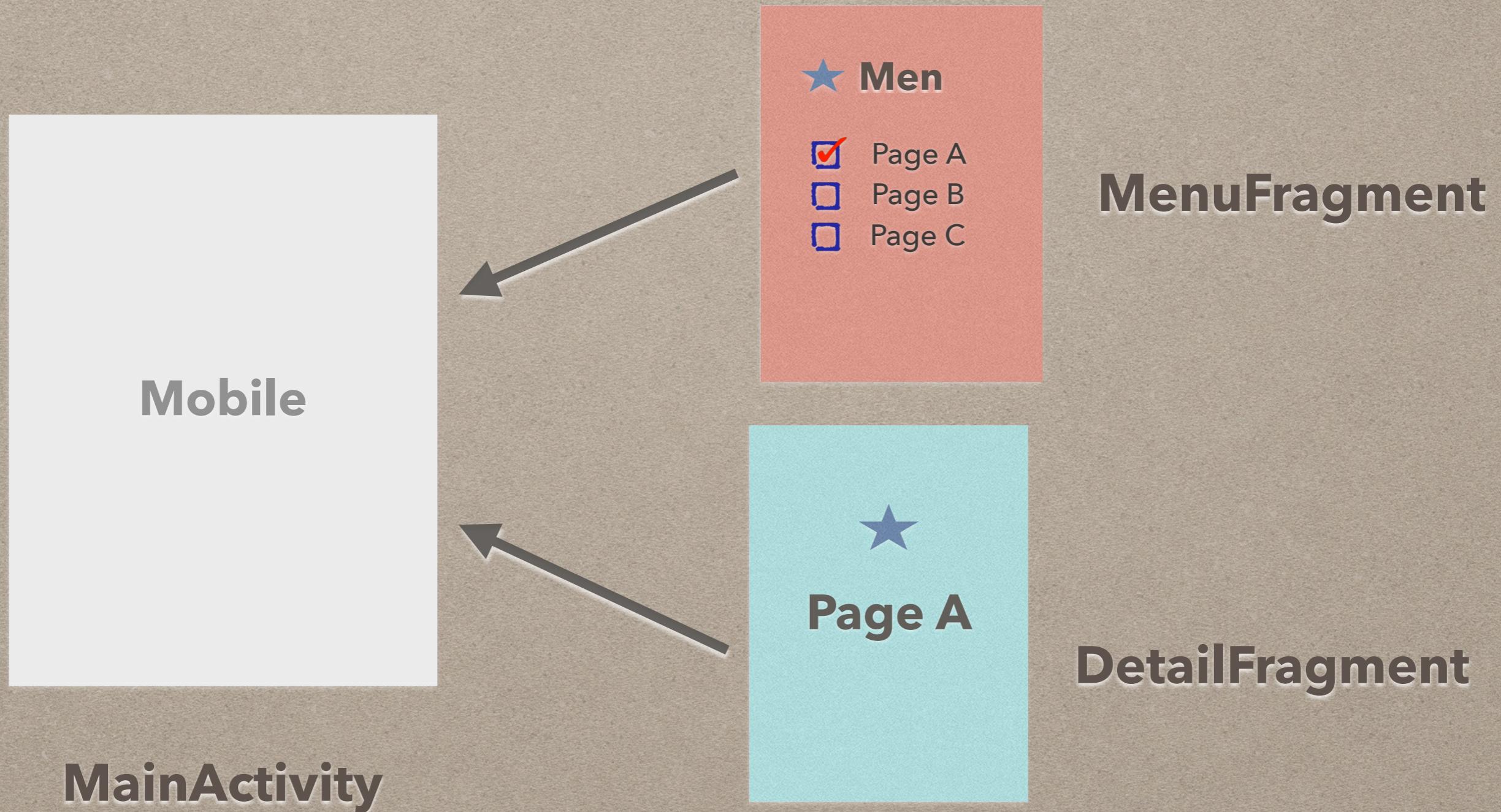


Page A



DetailFragment

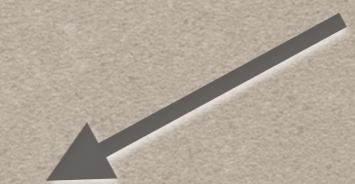
DESIGN PHILOSOPHY



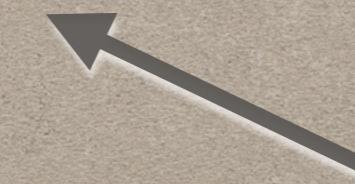
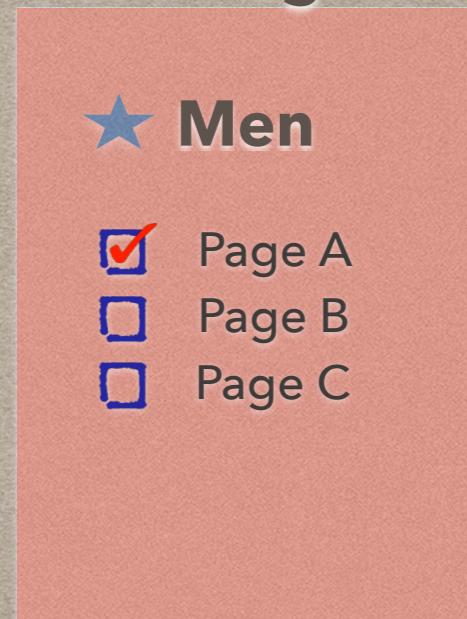
DESIGN PHILOSOPHY



MainActivity



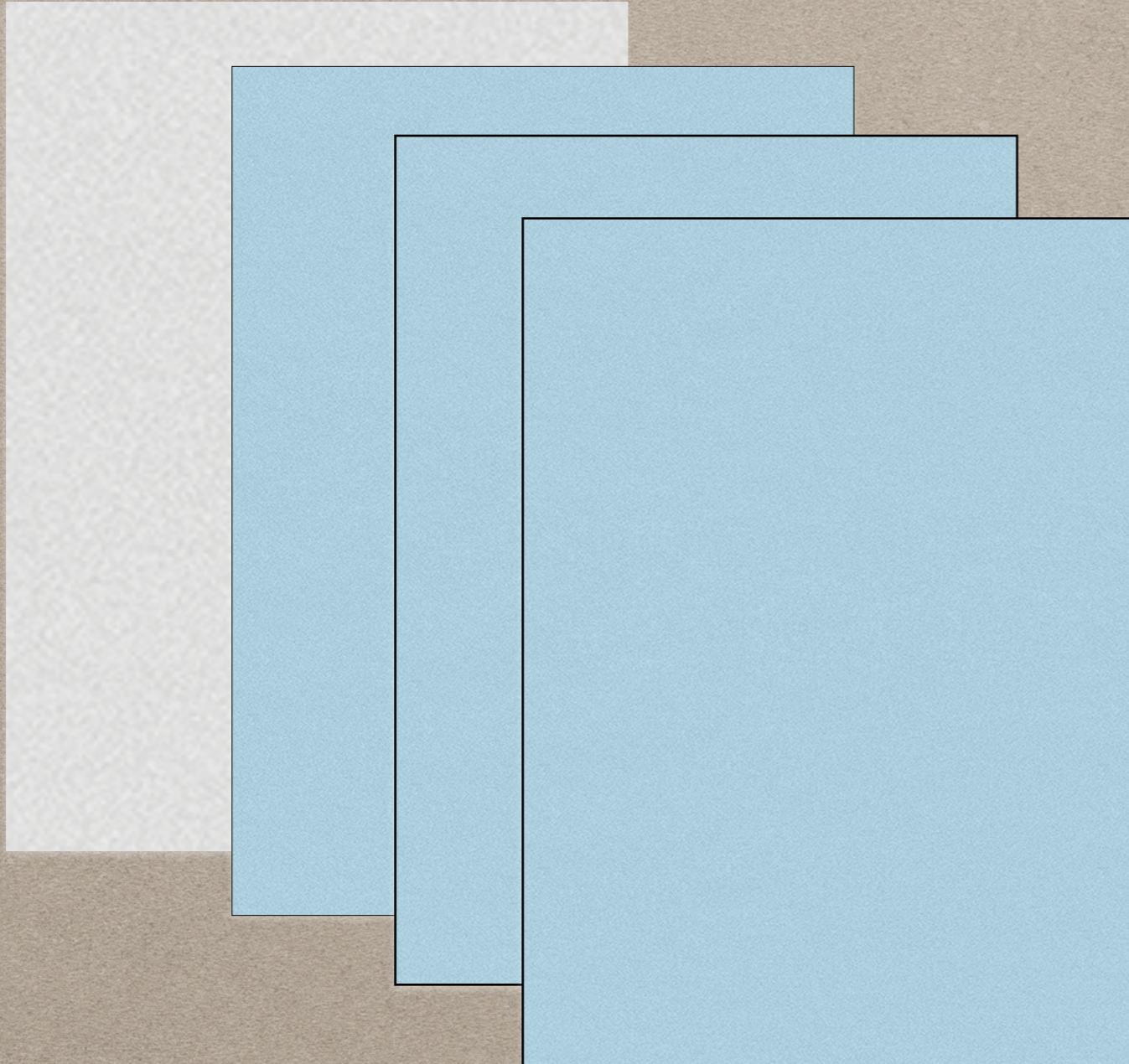
MenuFragment



DetailFragment

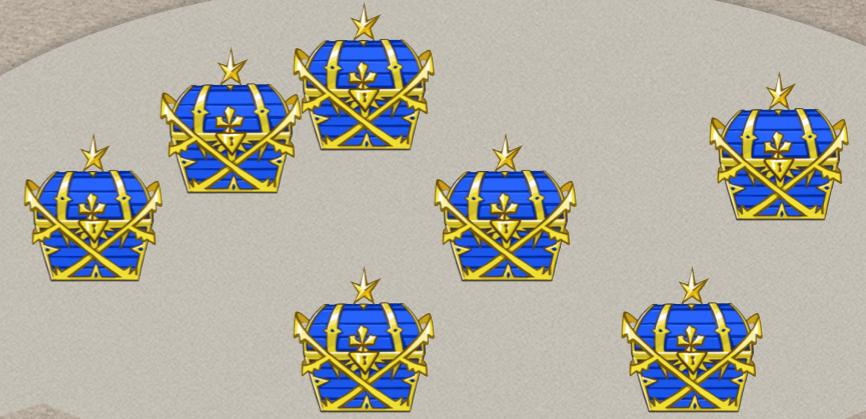
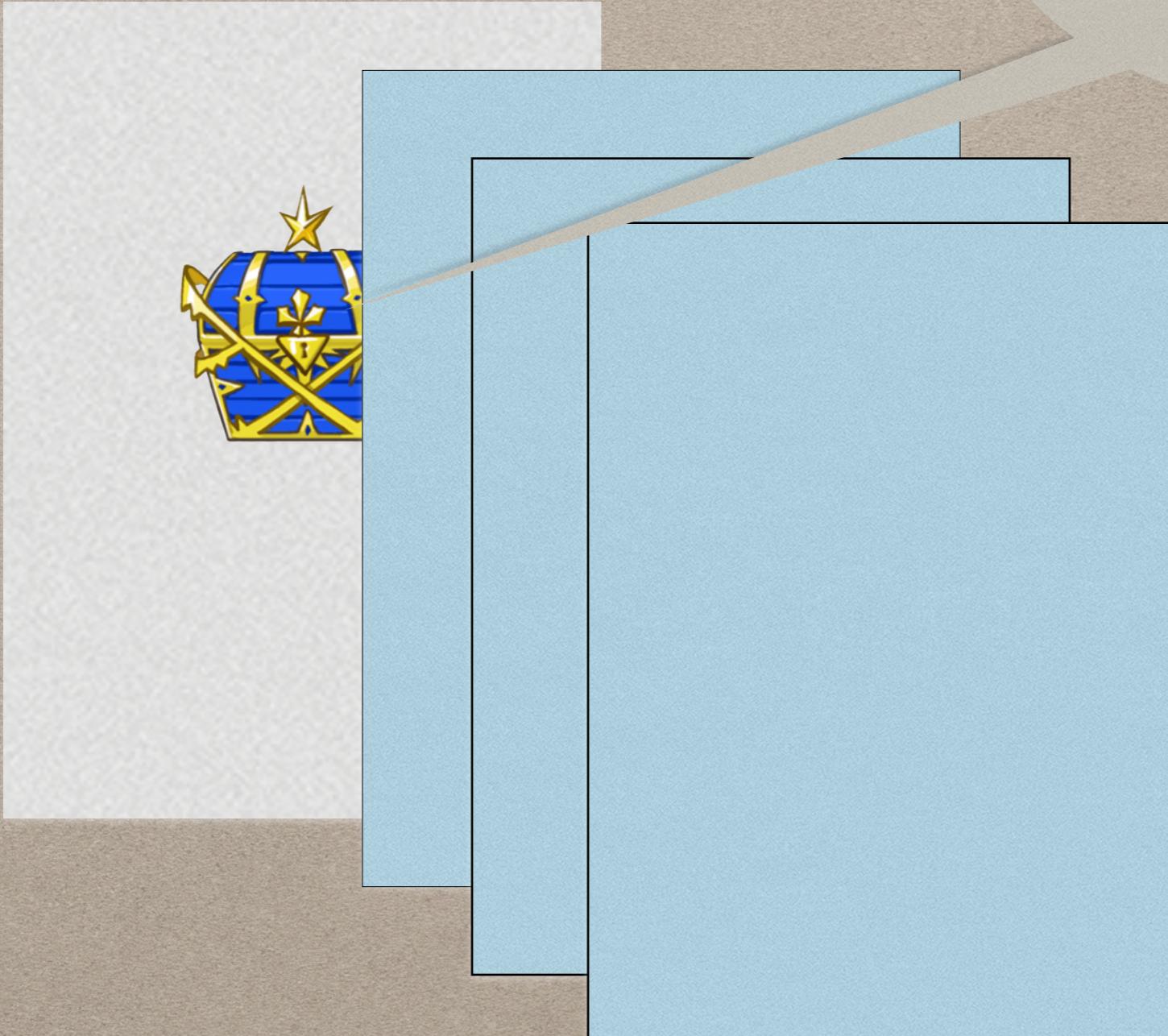


INTRO FRAGMENT



Fragment สามารถ
วางช้อนกันหลายชั้น
บน Activirty ได้

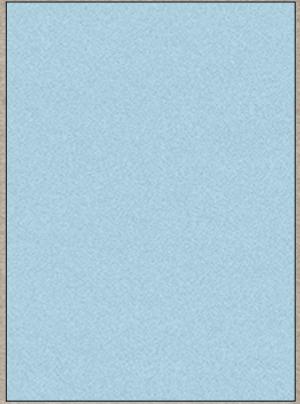
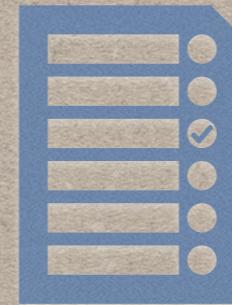
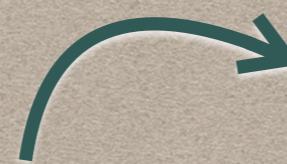
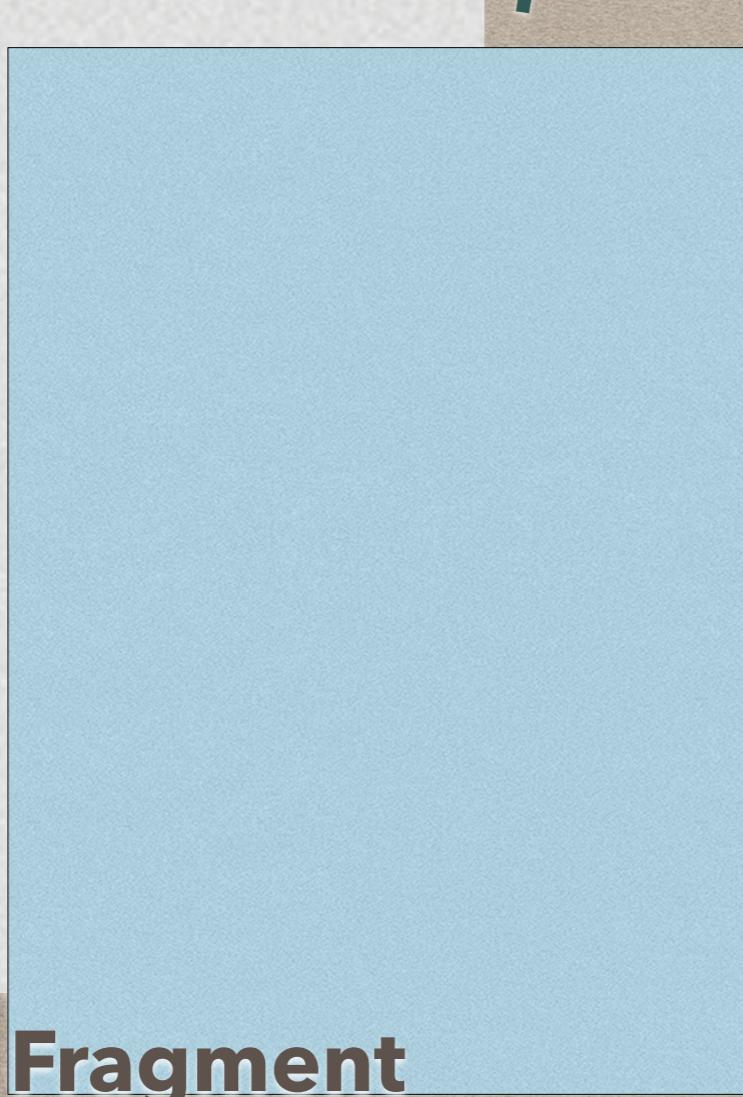
INTRO FRAGMENT



เมื่อ fragment ข้อนั้น
ก็จะแยกจาก Fragment
ที่อยู่ลึกสุด user ไม่
เห็น Fragment ก็จะ
ตามเก็บไว้เฉพาะ
ประวัติ ทำให้มีกิน
ram

INTRO FRAGMENT

Activity



Fragment

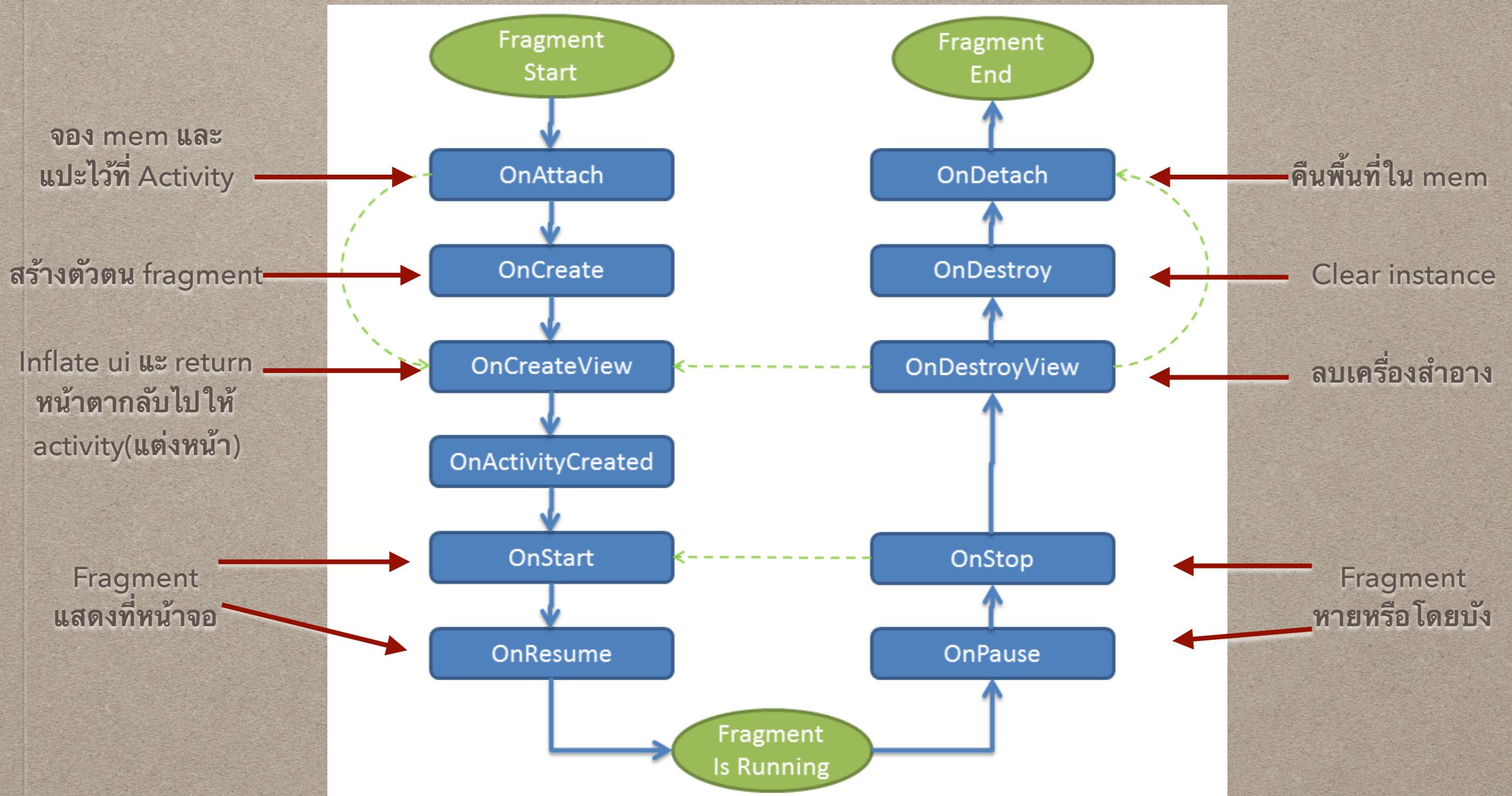
เมื่อเรากด back กลับ
ไปจนถึงตัวที่ติด มัน
จะทำการไปเช็ค
ประวัติแล้วทำแปลง
ให้กลับมาเป็น
Fragment ที่เราใช้
งานต่อได้



FRAGMENT

Reusable!

FRAGMENT LIFECYCLE



ADDING A FRAGMENT TO AN ACTIVITY

- **Declaratively** - Fragments สามารถประกาศไว้ใน .xml files โดยใช้ tag **<Fragment>**
- **Programmatically** - Fragments สามารถประกาศเป็น instance ได้โดยการใช้ **FragmentManager**

Create Fragment

- **fragment_main.xml**
- **MainFragment.java**

FRAGMENT .XML

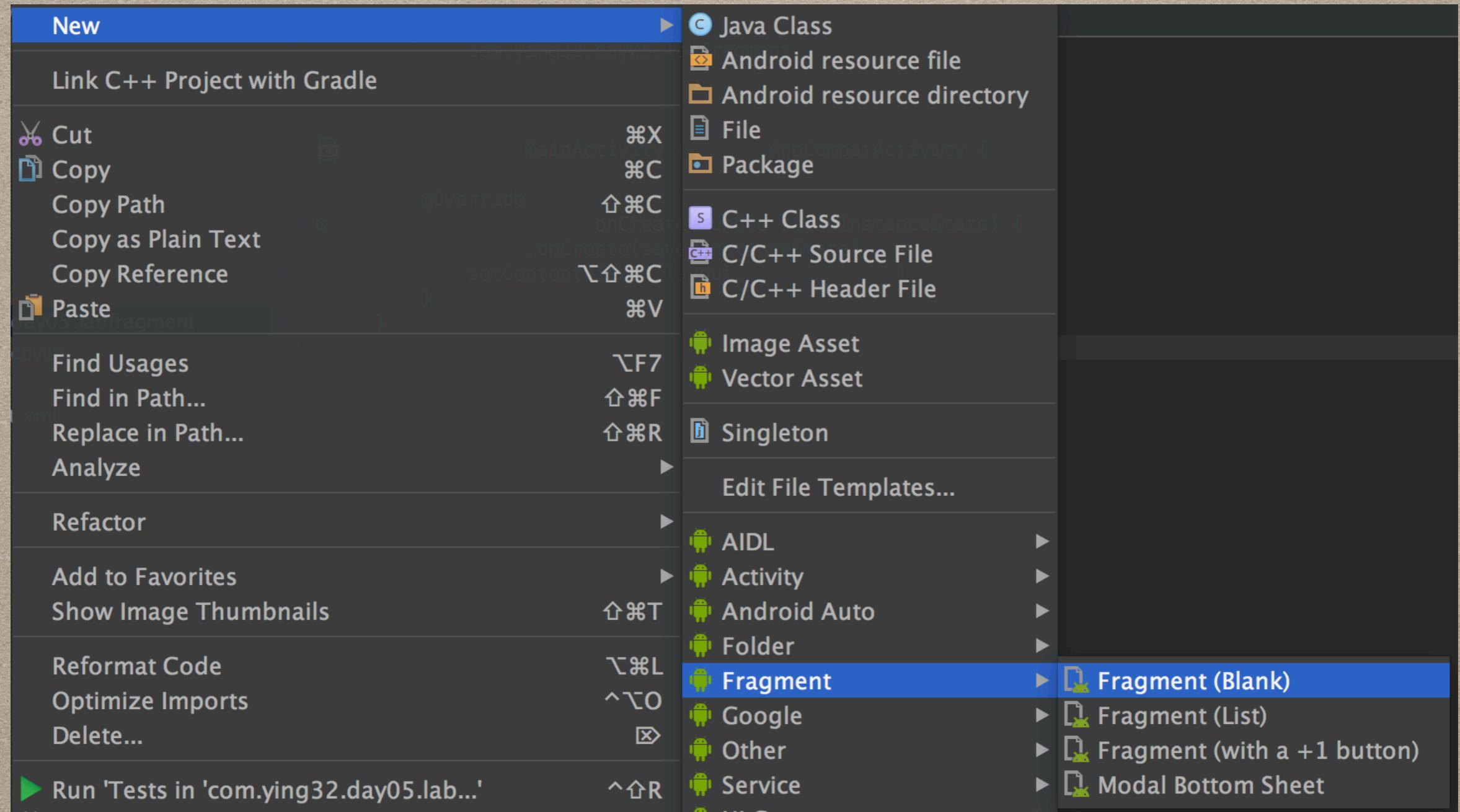
**Click ขวาที่ layout > new > layout resourceCreate
First Fragment file**

**วิธีการตั้งชื่อ [ประเภท]_[ชื่อ layout].xml
fragment_main.xml**

WORKSHOP

Add Fragment to Activity by <Fragment> tag .xml

CREATE FRAGMENT



CREATE FRAGMENT

Creates a blank fragment that is compatible back to API level 4.

Fragment Name:

BlankFragment

Create layout XML?

Fragment Layout Name:

fragment_blank

Include fragment factory methods?

Include interface callbacks?

MANUAL CREATING A FRAGMENT

- 1. Create Fragment Layout .xml**
- 2. Create Fragment class**
- 3. Extends Fragment**
- 4. Override onCreateView**
- 5. Inflate fragment layout to Fragment class**

OnCreateView

```
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                           Bundle savedInstanceState) {  
    return inflater.inflate(R.layout.fragment_main, container, false);  
}
```

View ที่ Fragment ถืออยู่

* Fragment is not View but it's just a class that hold Views.

Create Fragment Complete

ADDING A FRAGMENT BY XML

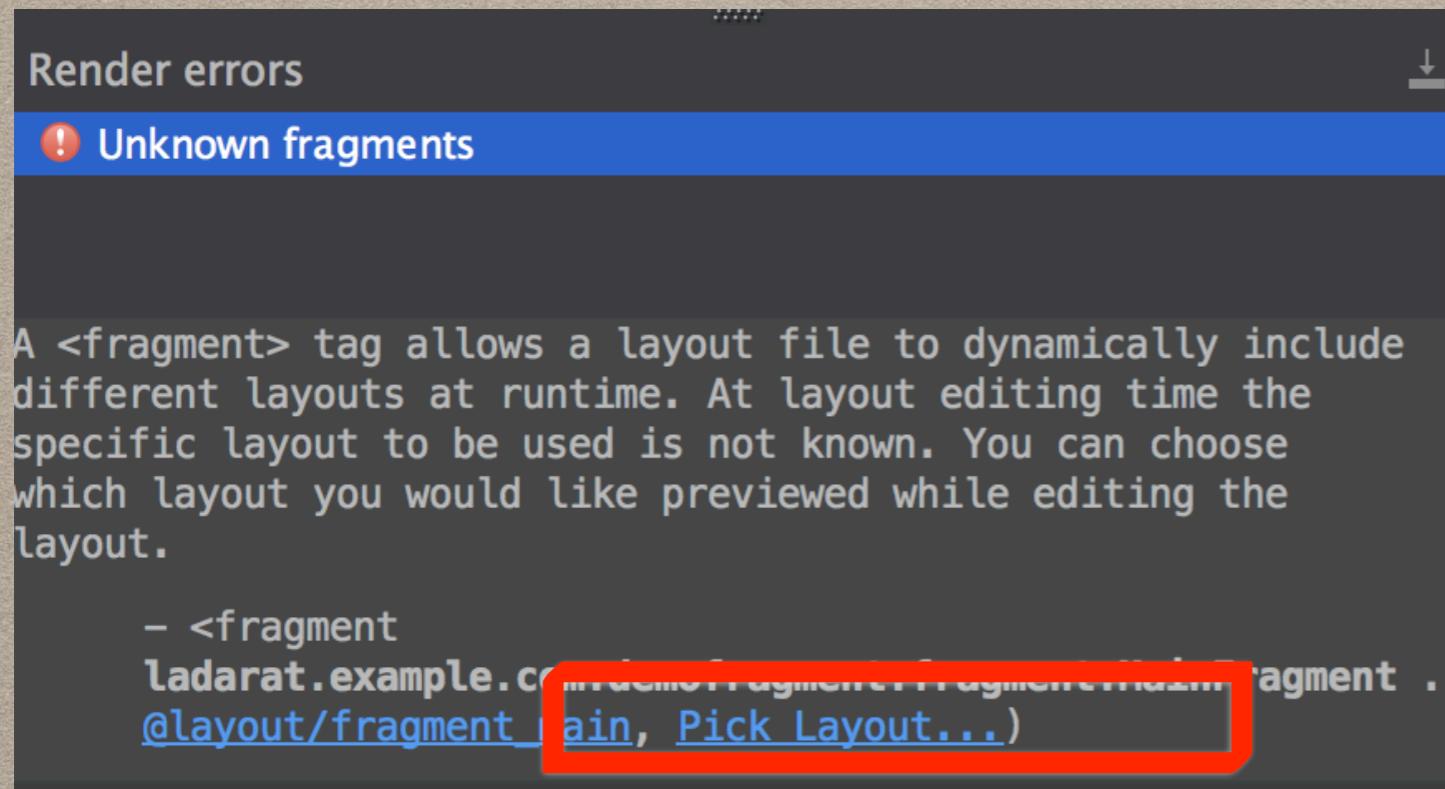
ทำการเพิ่ม เข้าไปใน activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.ying32.day05.labfragment.MainActivity">

    <fragment
        android:id="@+id/fragmentContainer"
        android:name="com.ying32.day05.labfragment.MainFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

ADDING A FRAGMENT BY XML



สำหรับคนที่ติดเรื่อง
Preview

ADDING A FRAGMENT BY JAVA CODE

**สร้าง Fragment และเปลี่ยน Layout Activity ผ่านโค้ด
FragmentSupportManager**

FRAGMENT TRANSACTION

Fragment Transaction : class ทำหน้าที่แสดงผล UI
Fragment บน View ของ Activity

ขั้นตอนการใช้งาน

1. สร้าง FragmentTransaction
2. กำหนดคำสั่ง
3. สั่งให้เริ่มทำงานด้วยคำสั่ง Commit

FRAGMENT TRANSACTION

ตัวอย่าง โค้ด

```
getSupportFragmentManager()
    .beginTransaction()
    .add(R.id.fragmentContainer, new MainFragment())
    .commit();
```

INTRO FRAGMENT

การใช้งานแบบออกเป็น 2 แบบ

1. Replace : เอาร์ตัวใหม่มา ผลักตัวเก่าทิ้งออกไป

2. Backstact : การเรียงช้อนกันเป็นชั้นๆ

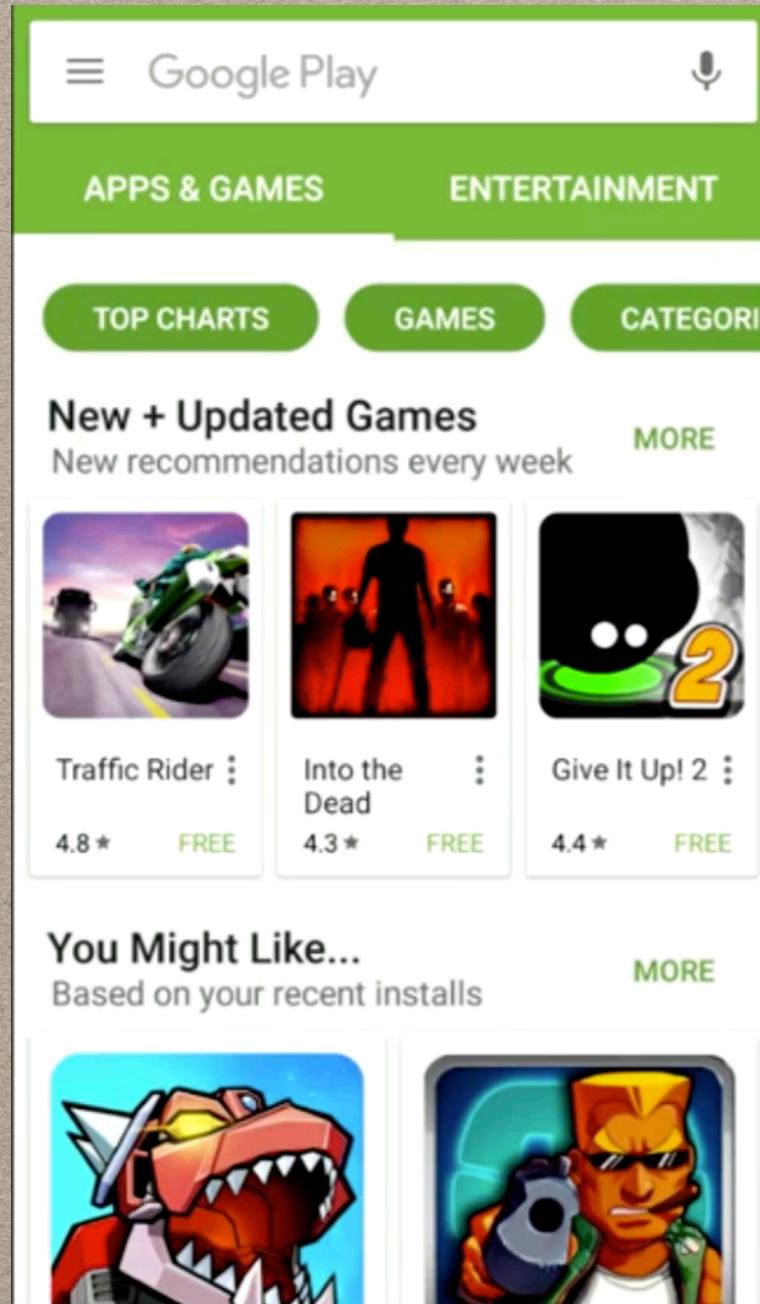
FRAGMENT : BACKSTACT

การซ่อน Fragment ตามลำดับการเปิด เมื่อเวลาเรากด back จะค่อยๆ เจอตัวก่อนหน้าไปเรื่อยๆ และค่อยๆ ทำลายตัวบนไปเรื่อยๆ จนถึงชั้นสุดท้าย Activity แล้วกด back อีกครั้ง จะเป็นการออกจาก Application

FRAGMENT : REPLACE

เป็นการทำงานโดยการสร้าง fragment ใหม่ไปแทนที่ตัวเก่า mode นี้ใน 1 Activity จะมี 1 Fragment จะไม่มีการเก็บประวัติของ fragment ก่อนหน้าไว้เลย

ตัวอย่าง APPLICATION ที่ใช้ FRAGMENT



มีการสร้าง Fragment มา
หลายหน้าแล้วก็ สลับ
Fragment โดยทุกอย่างทำงาน
อยู่บน Activity เป็นการจัดการ
Fragment แบบ Backstack

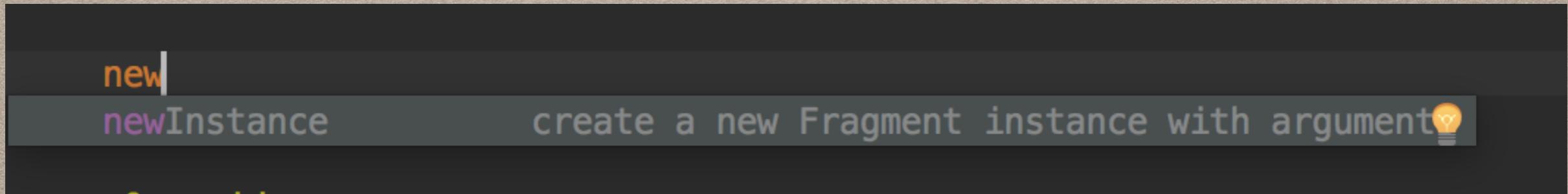
**Fragment help you
Scale Your Code**

ARGUMENT OF FRAGMENT

Argument คือ Object ที่ผูกกับ Fragment เมื่อ
Fragment ด้วย Argument ยังอยู่ และเมื่อ Fragment พิ่น
กลับมาก็สามารถอ่านค่าจาก Argument ขึ้นมาได้ใหม่

USE NEWINSTANCE IN FRAGMENT

Create newInstance



Argument จะถูกเก็บ ในรูปแบบ Bundle

SAVE ARGUMENT

```
public static MainFragment newInstance(String message) {  
    Bundle args = new Bundle();  
    MainFragment fragment = new MainFragment();  
    args.putString("message", message);  
    fragment.setArguments(args);  
    return fragment;  
}
```

READ ARGUMENT

```
String message;  
@Override  
public void onCreate(@Nullable Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    message = getArguments().getString("message");  
}  
}
```

เมื่อมีการเรียกให้ Fragment พื้นขึ้นมาอิก message ก็จะมีค่ากลับเข้ามาทุกครั้ง

CALL NEWINSTANCE FROM ACTIVITY

```
|  
|FragmentManager fragmentManager = getSupportFragmentManager();  
|fragmentManager.beginTransaction()  
|    .replace(R.id.fragmentContainer, new MainFragment().newInstance("My name is Ying"))  
|    .commit();  
|
```

ACCESS FRAGMENT FROM ACTIVITY

Binding View at Fragment

```
TextView helloTextView = rootView.findViewById(R.id.helloTextView);
```

Find Fragment At Activity

```
getSupportFragmentManager().findFragmentById(R.id.fragmentContainer)
```

```
getSupportFragmentManager().findFragmentByTag("MainFragment");
```



```
.add(R.id.fragmentContainer, fragment, "MainFragment")
```

ACCESS FRAGMENT FROM ACTIVITY

Binding View at Fragment

```
TextView helloTextView = rootView.findViewById(R.id.helloTextView);
```

Find Fragment At Activity

```
getSupportFragmentManager().findFragmentById(R.id.fragmentContainer)
```

```
getSupportFragmentManager().findFragmentByTag("MainFragment");
```



```
.add(R.id.fragmentContainer, fragment, "MainFragment")
```

SaveInstance state Fragment

អមុនជារ Fragment

```
@Override  
public void onSaveInstanceState(Bundle outState) {  
    super.onSaveInstanceState(outState);  
  
}  
  
@Override  
public void onActivityCreated(@Nullable Bundle savedInstanceState) {  
    super.onActivityCreated(savedInstanceState);  
  
    if(savedInstanceState != null){  
        // restore state  
    }  
}
```

ACCESS PARENT ACTIVITY VIEW IN FRAGMENT

```
TextView buttoActivity = getActivity().findViewById(R.id.activityButton);  
buttoActivity.setText("Access from Fragment");
```

Fragment : BackStack

```
FragmentManager fragmentManager =  
getSupportFragmentManager();  
fragmentManager.beginTransaction()  
    .replace(R.id.fragmentContainer, fragment)  
    .addToBackStack(null)  
    .commit();
```

WORKSHOP REPLACE

- Replace จะทำการ pop Fragment ตัวบนสุด และ เอาร์ว
เองพร้อมสีพื้นหลังของ Activity มาวางข้างบนสุดของ stack
- Add จะทำการเพิ่ม Fragment ขึ้นไปบน stack เฉพาะของ
ตัว Fragment (เห็นภาพซ้อนกัน)

Workshop 04_ReplaceBackStack

WORKSHOP POP FRAGMENT

```
getSupportFragmentManager().popBackStack();
```

WORK AT HOME

- ใช้ Fragment ในการจัดการ DotView หน้าใน MainActivity
- เพิ่มหน้า EditDotFragment สำหรับแก้ไข สี, X, Y และ รัศมี
 - กดที่ Dot คือ ลบ Dot
 - กดแซ่(LongPass) คือการไป Edit Dot