SoftUni Foundation

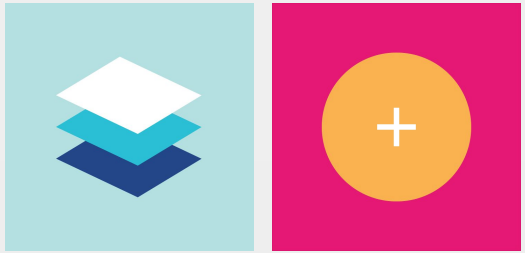
android development

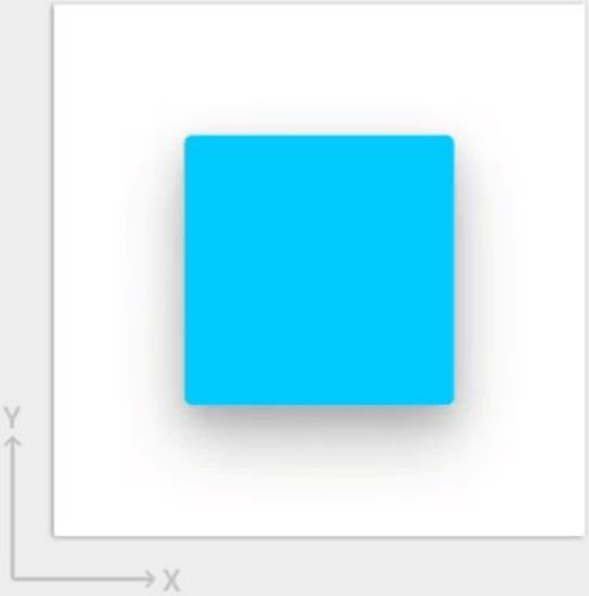
for beginners

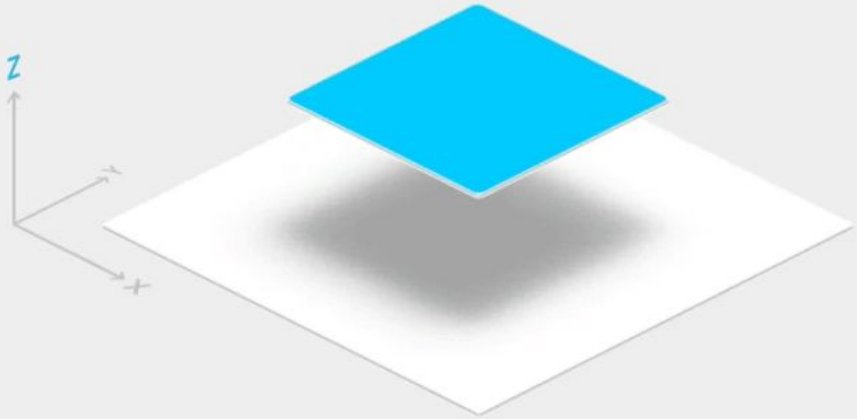
Join at Slido.com with #AndroidSoftUni

Fragments. Advanced UI

- Material design
- Fragments
- Toolbar
- Tabs
- Scenes
- Transitions
- Animate layout changes
- Contextual action bar



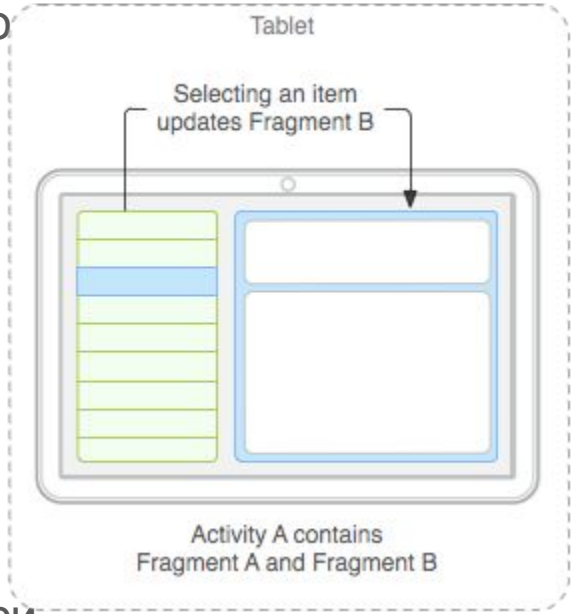

Y
X


Z
Y
X

фрагменти

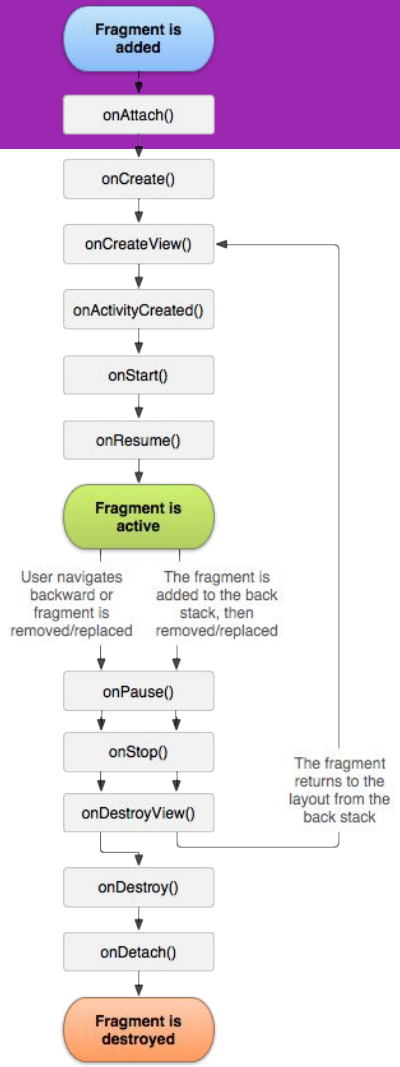
Същност

- Може да се разглеждат като под-активитита
- Имат си свой жизнен цикъл
- Закачват се за активити
- Едно активити може да има много фрагменти
- Фрагментите могат да се добавят и махат докато активитито е пуснато
- Живота на фрагмента зависи от живота на активитито за което е закачен



Живот

- Основната разлика с Активити е метода onCreateView, който трябва да върне инфлейтнатото View на фрагмента
- Останалите методи са същите
- Фрагментът задължително трябва да има празен конструктор, за да може да се пресъздава свободно



Закачване

- Статичния метод използва хмл:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
</LinearLayout>
```

- Динамичния използва джава код:

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
transaction.addToBackStack(null);
fragmentTransaction.commit();
```

Връзка с Активити

- Чрез извикване на методи директно
- Чрез създаване на listener

toolbar

Toolbar

- Т.нар. ActionBar е добавен във версия 11 на Андроид, във версия 21 е заменен с ToolBar
- ActionBar-а беше труден за персонализиране, добавяше се само от програмния код, не беше част от йерархията на елементите, което ограничаваше използването му
- Toolbar-а решава тези проблеми. Той е обикновено View, което може да бъде навсякъде в йерархията, същевременно може да се зададе да бъде използвано като ActionBar

Стъпки за използването му

- Освен ако няма да поддържате само версия 21 и нагоре , вместо <Toolbar /> трябва да напишете <android.support.v7.widget.Toolbar /> и да добавите библиотеката compile 'com.android.support:appcompat-v7:21.0.+' в грейдъл файла си
- За да зададете това като ActionBar на Activity-то трябва да добавите и `setSupportActionBar(toolbar);`
- За да не си пречат дефолтния и вашия екшън бар, трябва да използвате темата `Theme.AppCompat.Light.NoActionBar,` тя се сменя от res/values/styles
- Когато използваме елементи от AppCompat библиотеката, нашето Activity трябва да наследява AppCompatActivity, а не просто Activity, с цел те да работят

Actions

- Бутоните, които се намират от дясната страна на екшън бара се наричат Action Buttons
- Те трябва да бъдат конкретни действия за екрана - търсене, центриране на екрана, изпращане и т.н.
- Те не трябва да водят до други екрани или по друг начин да са част от навигацията
- Когато има повече екшън бутони от място, те влизат в Overflow Menu, три точки, които отварят падащ списък с останалите екшъни
- Екшън бутоните се дефинират в отделен хмл в папка menu

Up Navigation

- Вместо иконката на приложението, в горния ляв край на екшън бара може да се намира стрелка наляво
- Тя води до предишното в йерархията активити
- Има разлика между това да натиснете НАЗАД и да натиснете стрелката в екшън бара
- НАЗАД винаги ви води на активитито, което сте били отворили преди сегашното, докато стрелка нагоре ВИНАГИ трябва да води нагоре в йерархията на приложението ви
- Засичането на клик на стрелка нагоре, както и на другите екшън бутони става в метода `onOptionsItemSelected`
- Засичането на клик на бутона НАЗАД става в метода `onBackPressed`

tabs

Създаване

- За да използваме лолипоп елементите за табове, трябва да си добавим библиотеката `compile 'com.android.support:design:23.0.1'`
- Всеки таб е нов фрагмент
- Смяната между фрагментите става чрез ViewPager - елемент, който сменя децата си при swipe
- За да го използваме, трябва да добавим и адаптер

view binding

Що

- Модерният начин за свързване на лейаута с кода
- Добавя се в gradle файла

```
dataBinding {

   enabled = true

}
```

- Лейаутите трябва да са обгърнати от <layout> таг
- Това е достатъчно за да се генерира автоматично обект, който държи всички вюта в лейаута
- Достъпва се чрез:

```
FragmentFeedBinding binding = DataBindingUtil.inflate(inflater, R.layout.fragment_feed, container, false);
```

сцени

Що

- Позволяват анимирането на обекти от едно състояние в ново
- Тези състояния се наричат сцени
- Сцената се получава от лейаут файл `Scene.getSceneForLayout()`
- За да анимирате между две сцени ви трябва Transition обект
  - AutoTransition
  - Fade
  - ChangeBounds
  - Клас наследяващ Transition

Transition

- `TransitionManager.go(mEndingScene, new Fade());` пуска транзишъна
- `removeTarget()` казва на Transition-а кои елементи от сцената да не се анимират
- За да имате повече от един транзишън

```
<transitionSet xmlns:android="http://schemas.android.com/apk/res/android"
    android:transitionOrdering="sequential">
    <fade android:fadingMode="fade_out" />
    <changeBounds />
    <fade android:fadingMode="fade_in" />
</transitionSet>
```

- Може да анимираме и без сцени, чрез `TransitionManager.beginDelayedTransition()`
- То трябва да се извика непосредствено преди промяната на елемента в кода

Позволяване на Activity Transition

- В стиловете

```
<item name="android:windowSharedElementEnterTransition">@transition/change_image_transform</item>
<item name="android:windowSharedElementExitTransition">@transition/change_image_transform</item>

@transition/change_image_transform:
<transitionSet xmlns:android="http://schemas.android.com/apk/res/android">
  <changeImageTransform/>
</transitionSet>
```

- Или в кода:
  - `Window.setSharedElementEnterTransition()`
  - `Window.setSharedElementExitTransition()`

Shared Element Activity Transition

1. Трябва в двата лейаута да посочите общия елемент чрез `android:transitionName` атрибута.
2. Да го зададете като опция на Активитито и да го стартирате

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
// create the transition animation - the images in the layouts
// of both activities are defined with android:transitionName="robot"
ActivityOptions options = ActivityOptions
        .makeSceneTransitionAnimation(MainActivity.this, findViewById(R.id.image), "lion");
// start the new activity
startActivity(intent, options.toBundle());
```

selectors

Selectors

- Селекторите указват какъв фон да се използва за елемента, спрямо състоянието му
- Дефинират се в drawable/ папката

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/selsected_state"    //използвай тази картинка
        android:state_pressed="true"                      //когато елемента е натиснат
        android:state_enabled="true"/>                   // и е в активно състояние
    <item android:drawable="@drawable/disabled_state"
        android:state_pressed="false"
        android:state_enabled="false"/>
    <item android:drawable="@drawable/unselsected_state" />
</selector>
```

- Може да се използват готови селектори, например ?android:attr/selectableItemBackground
- За да покажем елемент чрез разширяващ се кръг използваме

```
Animator anim =
    ViewAnimationUtils.createCircularReveal(myView, cx, cy, 0, finalRadius);
revealElemt.startAnimation(anim);
```

Ресурси

https://developer.android.com/design/index.html

http://developer.android.com/training/appbar/action-views.html

http://developer.android.com/guide/components/fragments.html

https://www.youtube.com/watch?v=S3H7nJ4QaD8

Homework #1

Преработете задачата с червения апп от домашното за Views&Layouts, така че да използва RecyclerView, CardView, view binding, tabs & bottom navigation.