

# 用户界面开发进阶

字节跳动 Android 工程师  
刘成 ([liucheng@bytedance.com](mailto:liucheng@bytedance.com))

# Overview

- Animation
- Fragment
- Exercise

# Animation

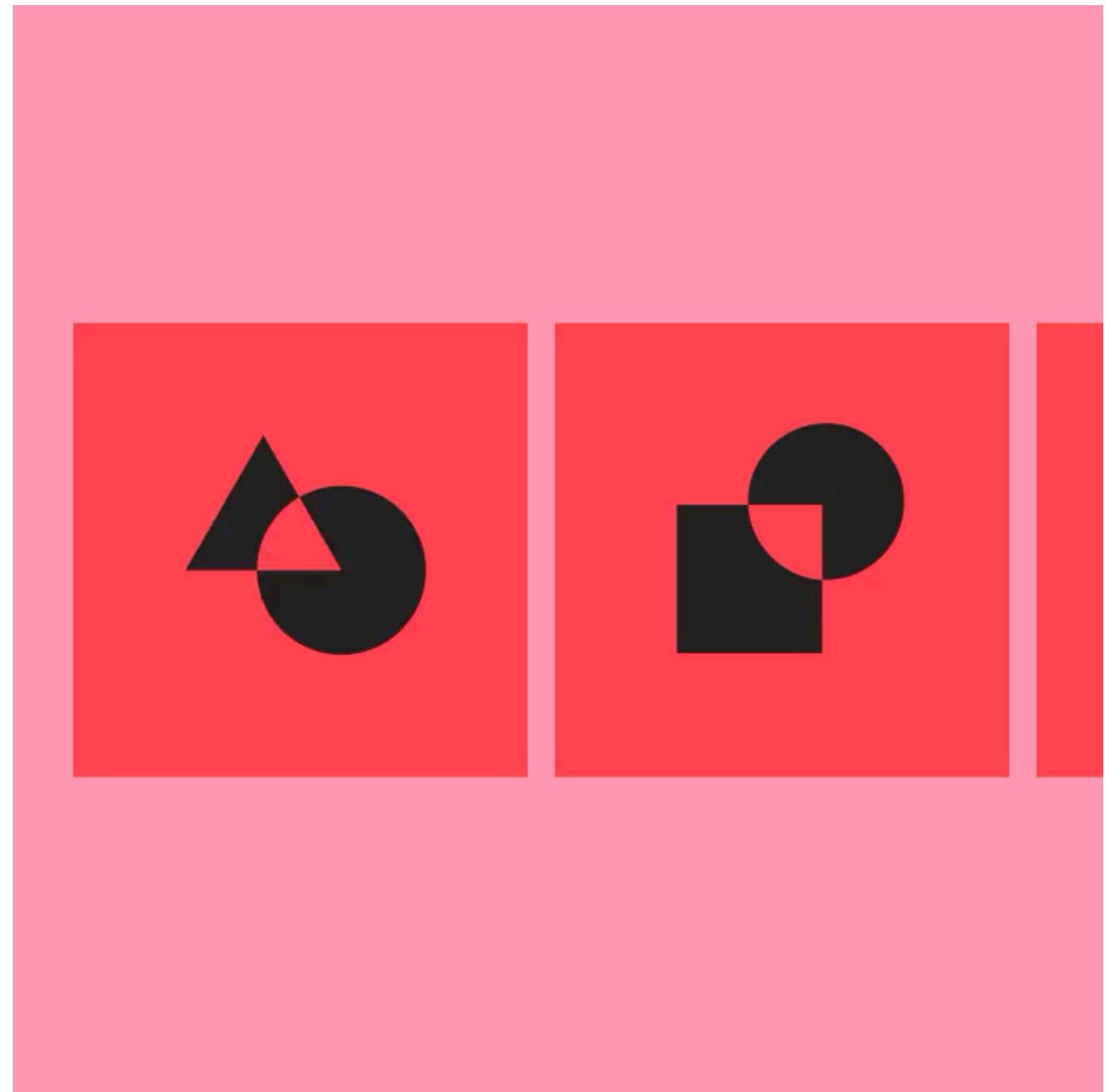




# Animation

- 意义
- 属性动画
- Activity 切换动画
- Drawable 动画

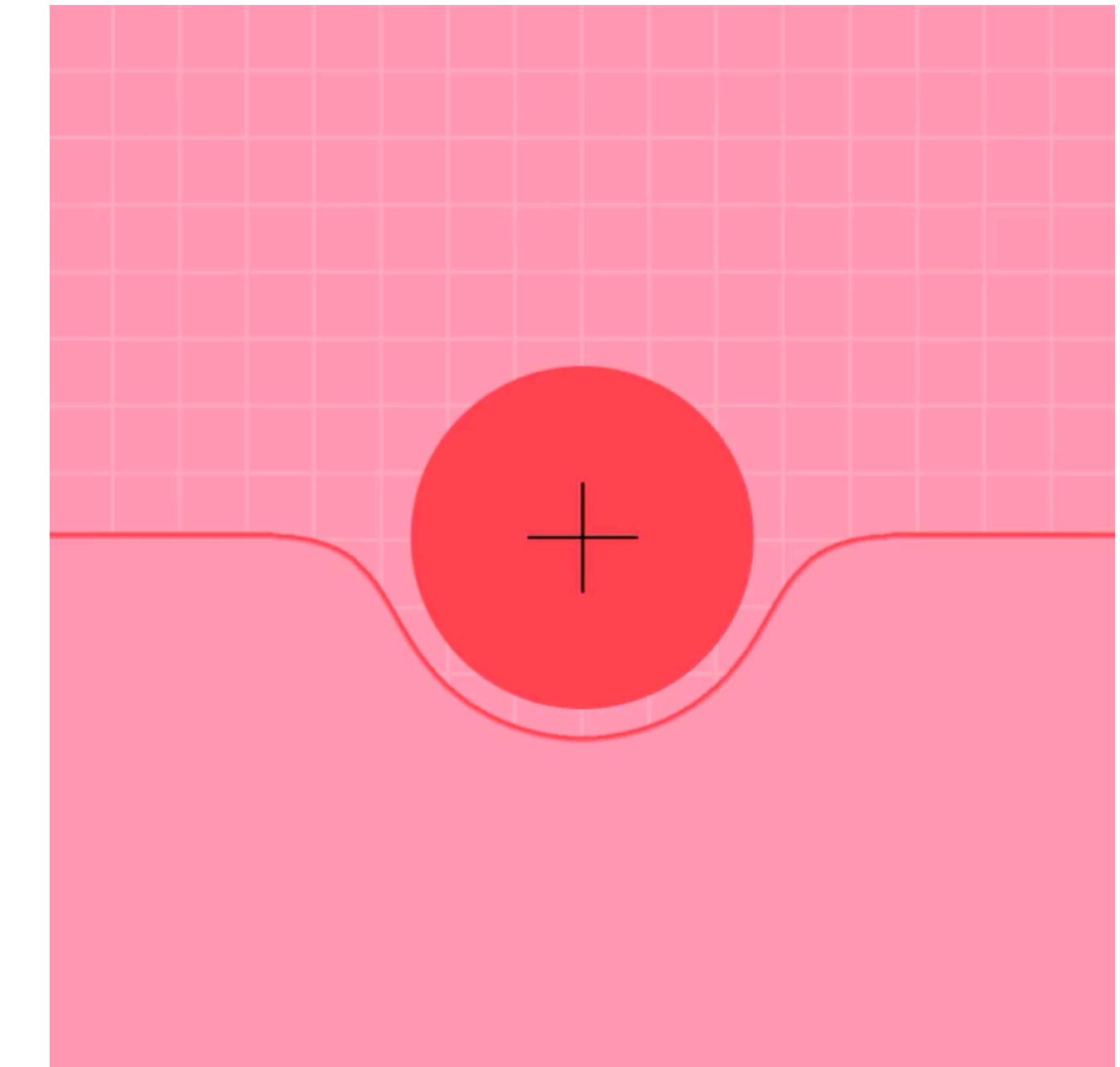
# 意义



Informative

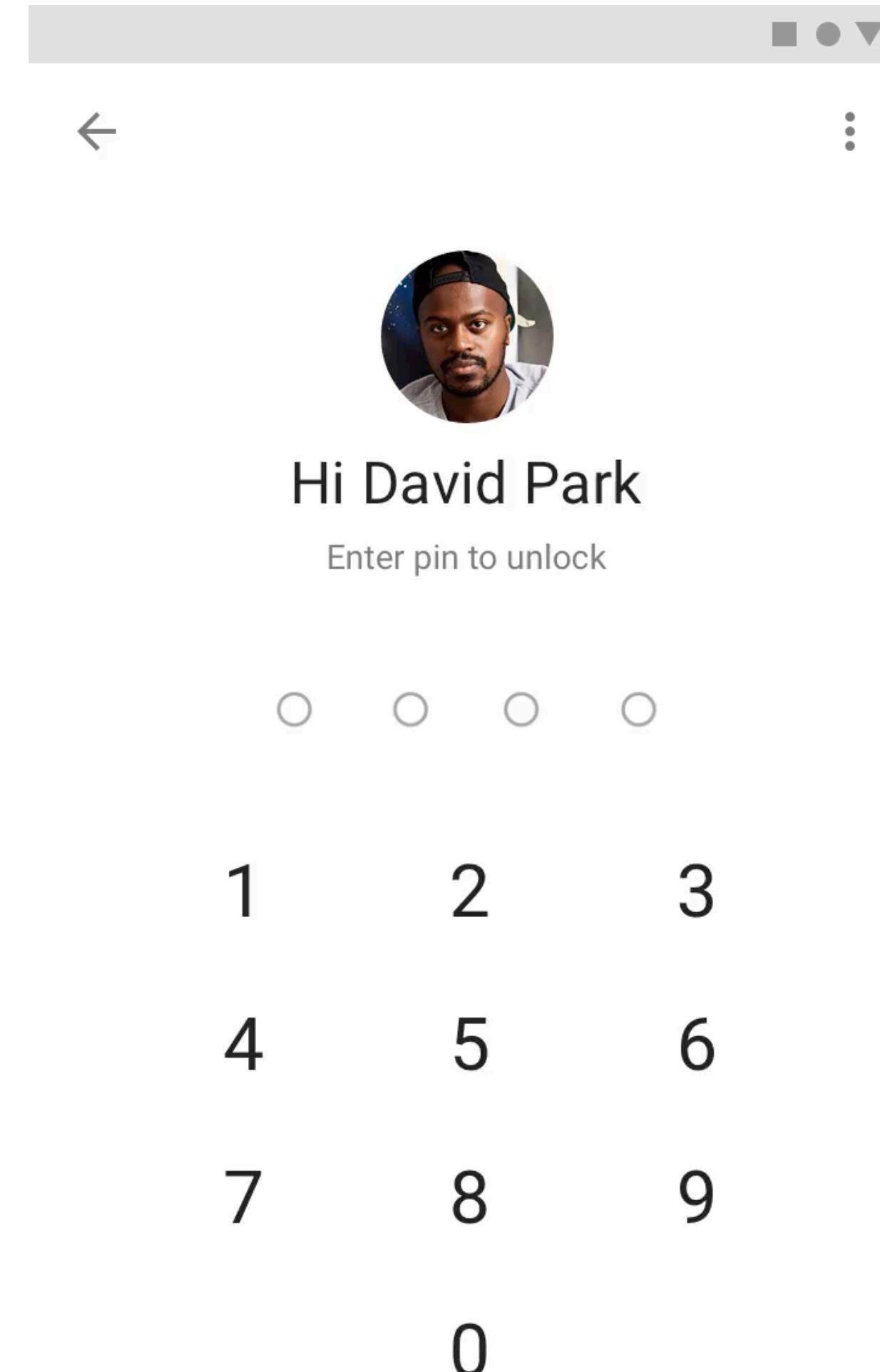


Focused



Expressive

# 意义

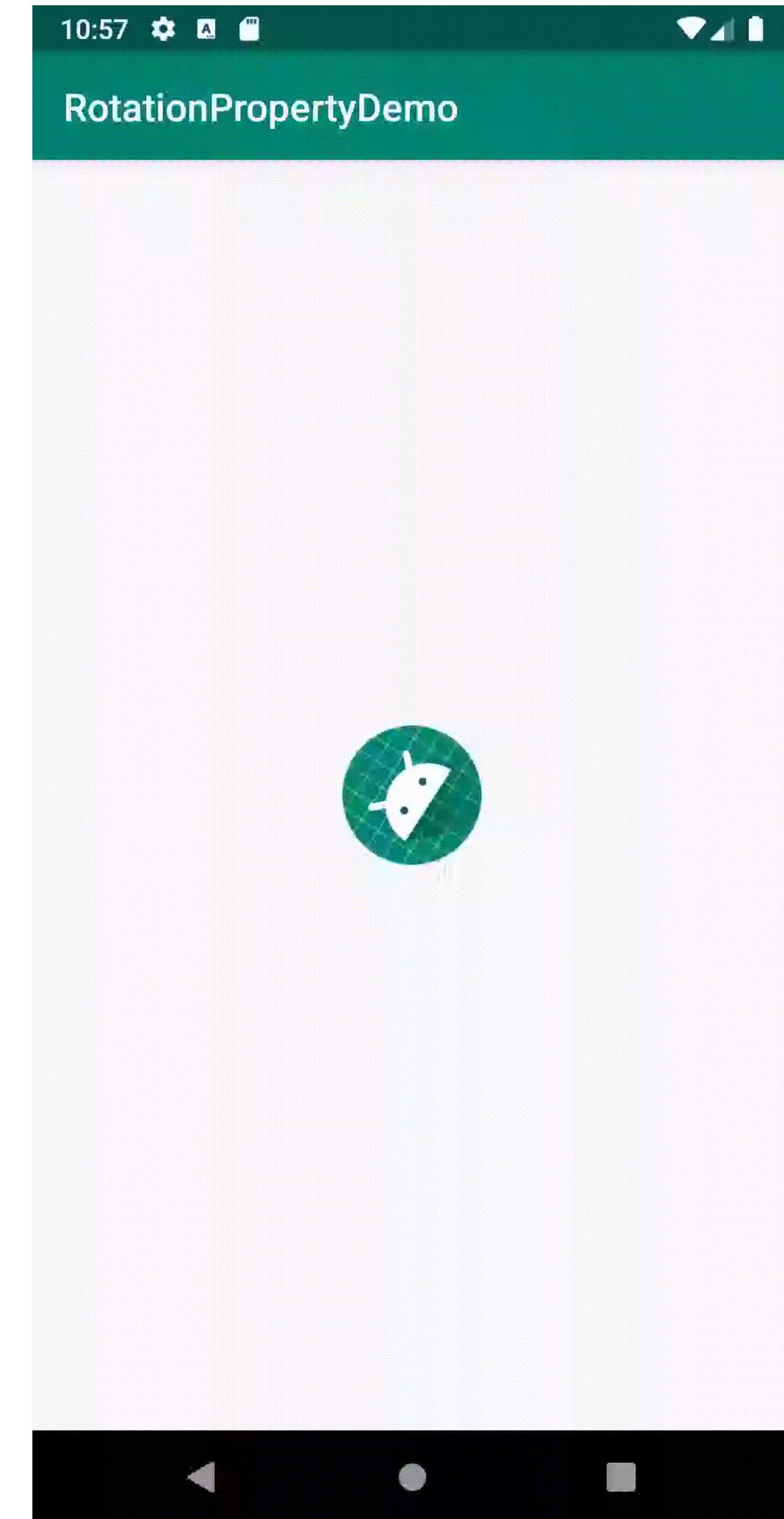


Feedback

User Education

# 属性动画 - 示例，旋转封面，Code

```
ObjectAnimator animator = ObjectAnimator.ofFloat(  
    findViewById(R.id.image_view),  
    "rotation", 0, 360);  
animator.setRepeatCount(ValueAnimator.INFINITE);  
animator.setInterpolator(new LinearInterpolator());  
animator.setDuration(8000);  
animator.setRepeatMode(ValueAnimator.RESTART);  
animator.start();
```



# 属性动画 - 示例，旋转封面，XML

```
<!-- animator/rotate.xml -->
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="8000"
    android:propertyName="rotation"
    android:interpolator="@android:anim/linear_interpolator"
    android:repeatCount="infinite"
    android:repeatMode="restart"
    android:valueFrom="0"
    android:valueTo="360" />
```

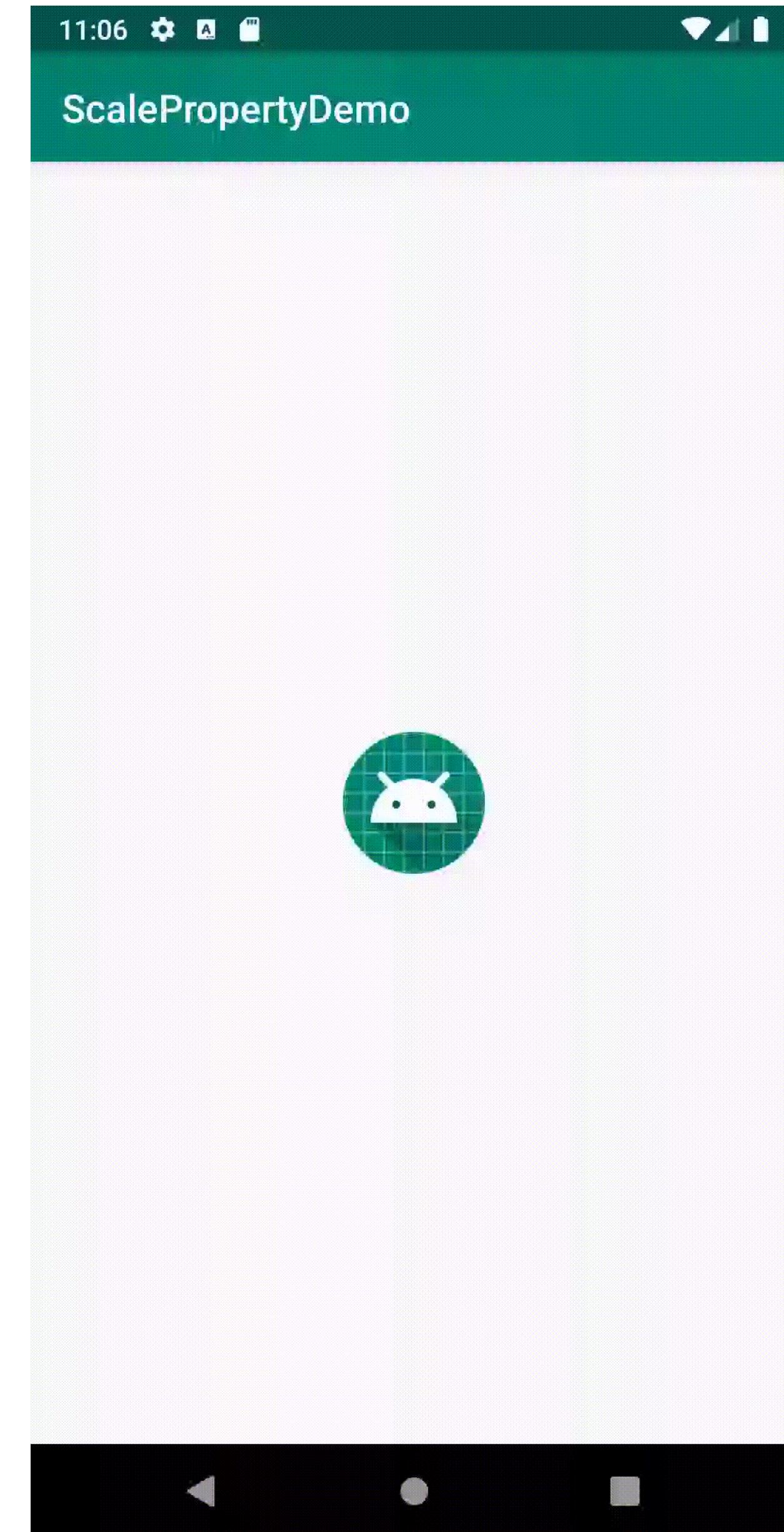
```
Animator animator = AnimatorInflater.loadAnimator(this, R.animator.rotate);
animator.setTarget(findViewById(R.id.image_view));
animator.start();
```

# 属性动画 - 示例，呼吸，Code

```
View imageView = findViewById(R.id.image_view);
ObjectAnimator scaleXAnimator = ObjectAnimator.ofFloat(imageView,
    "scaleX", 1.1f, 0.9f);
scaleXAnimator.setRepeatCount(ValueAnimator.INFINITE);
scaleXAnimator.setInterpolator(new LinearInterpolator());
scaleXAnimator.setDuration(1000);
scaleXAnimator.setRepeatMode(ValueAnimator.REVERSE);

ObjectAnimator scaleYAnimator = ObjectAnimator.ofFloat(imageView,
    "scaleY", 1.1f, 0.9f);
scaleYAnimator.setRepeatCount(ValueAnimator.INFINITE);
scaleYAnimator.setInterpolator(new LinearInterpolator());
scaleYAnimator.setDuration(1000);
scaleYAnimator.setRepeatMode(ValueAnimator.REVERSE);

AnimatorSet animatorSet = new AnimatorSet();
animatorSet.playTogether(scaleXAnimator, scaleYAnimator);
animatorSet.start();
```



# 属性动画 - 示例，呼吸， XML

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:duration="1000"
        android:valueFrom="1.1"
        android:valueTo="0.9"
        android:propertyName="scaleX"
        android:interpolator="@android:anim/linear_interpolator"
        android:repeatMode="reverse"
        android:repeatCount="infinite" />

    <objectAnimator
        android:duration="1000"
        android:valueFrom="1.1"
        android:valueTo="0.9"
        android:propertyName="scaleY"
        android:interpolator="@android:anim/linear_interpolator"
        android:repeatMode="reverse"
        android:repeatCount="infinite" />
</set>
```

# 属性动画 - 特点

- Property: alpha, scaleX, scaleY, rotation, rotationX, rotationY, translationX, translationY, ...
- ObjectAnimator
  - Duration
  - Interpolator: Linear/AccelerateDecelerate/...
  - Repeat Count and Behavior: Infinite, Restart/Reverse
- AnimatorSet: play together or sequentially

# 属性动画 - 特点, XML 语法

```
<set
    android:ordering=[ "together" | "sequentially"]>

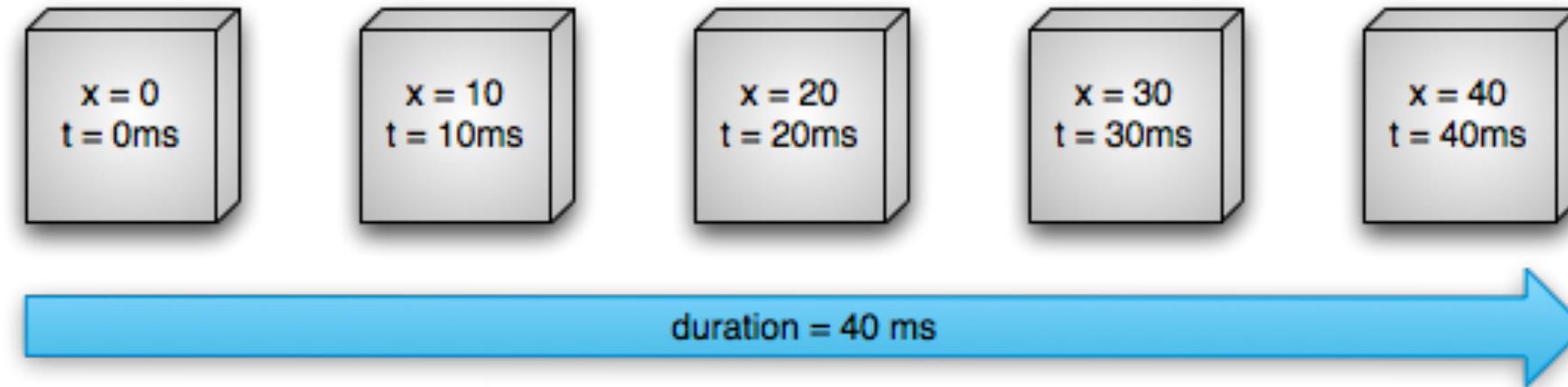
    <objectAnimator
        android:propertyName="string"
        android:duration="int"
        android:interpolator="@[package:]anim/interpolator_resource"
        android:valueFrom="float | int | color"
        android:valueTo="float | int | color"
        android:startOffset="int"
        android:repeatCount="int"
        android:repeatMode=[ "repeat" | "reverse" ]
        android:valueType=[ "intType" | "floatType" ] />

</set>
```

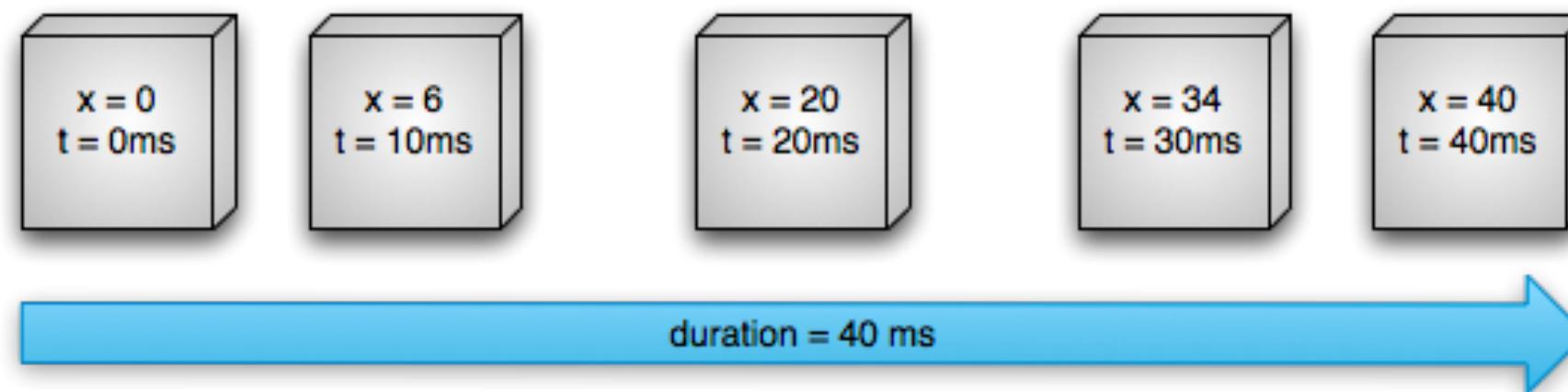
# 背后 - ValueAnimator, 旋转封面

```
final View v = findViewById(R.id.image_view);
ValueAnimator valueAnimator = ValueAnimator.ofFloat(0, 360);
valueAnimator.setRepeatCount(ValueAnimator.INFINITE);
valueAnimator.setInterpolator(new LinearInterpolator());
valueAnimator.setRepeatMode(ValueAnimator.RESTART);
valueAnimator.setDuration(8000);
valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        v.setRotation((float) animation.getAnimatedValue());
    }
});
valueAnimator.start();
```

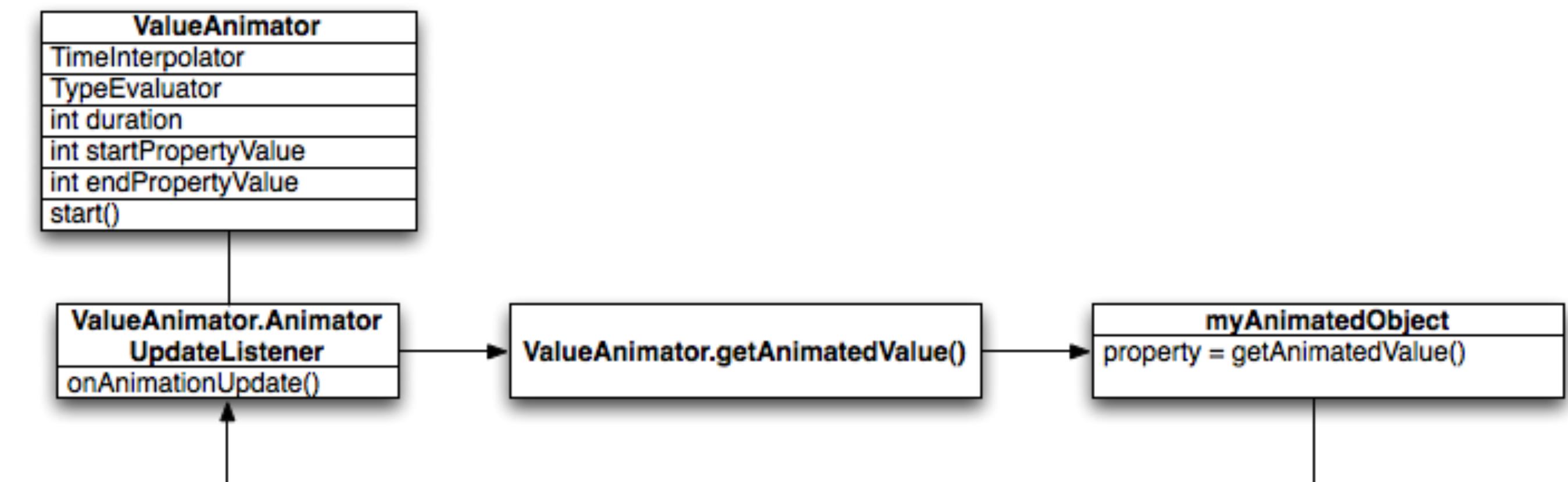
# 属性动画 - 原理



Linear animation



Nonlinear animation



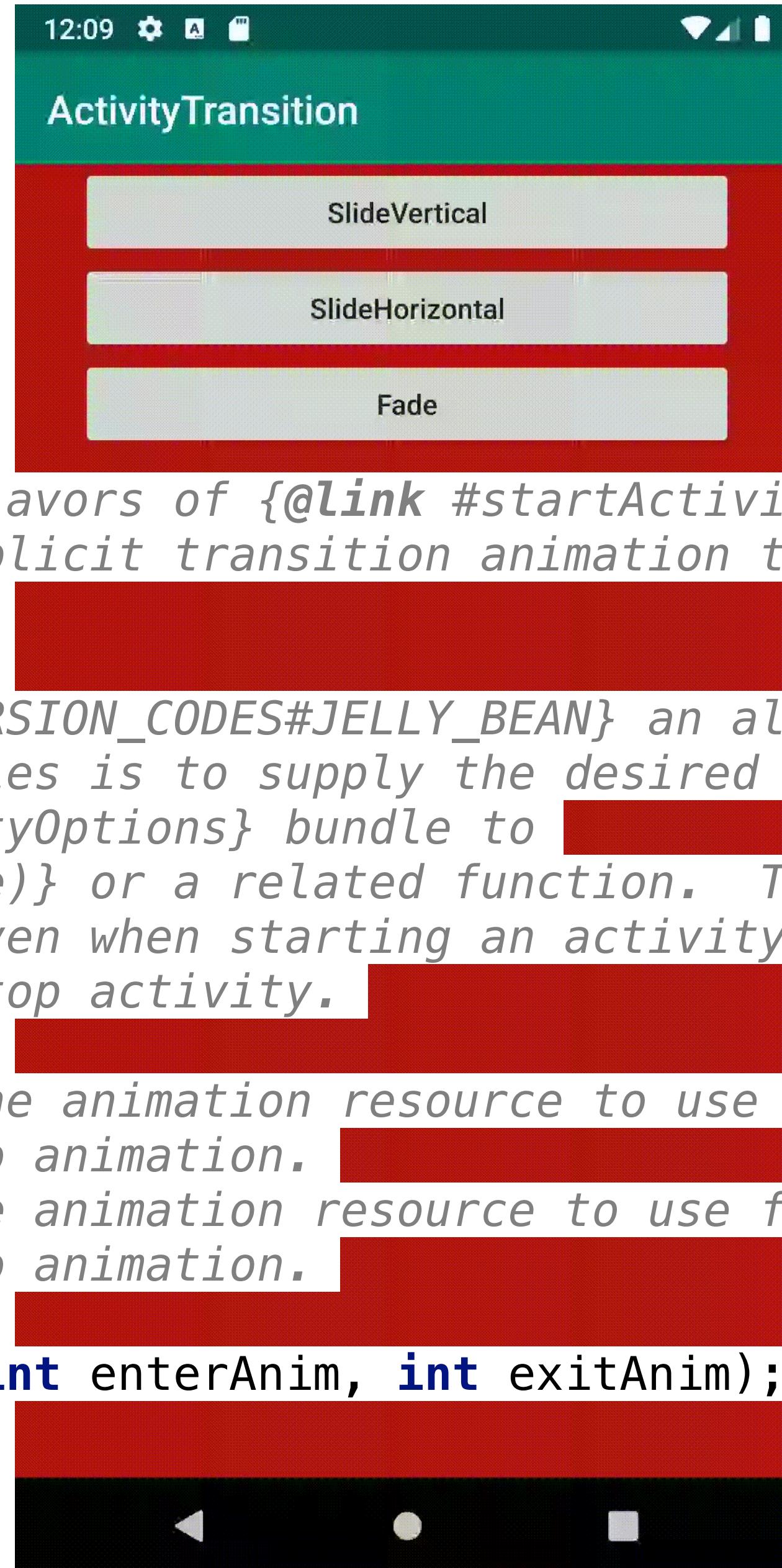
How animations are calculated

# 属性动画 vs 视图动画

- 属性动画: android.animation
- 视图动画: android.view.animation
  - 只能对 View 做动画
  - 只能对 View 的某些属性做动画
  - 只是视觉效果

# Activity 切换动画

```
/**  
 * Call immediately after one of the flavors of {@link #startActivity(Intent)}  
 * or {@link #finish} to specify an explicit transition animation to  
 * perform next.  
 *  
 * <p>As of {@link android.os.Build.VERSION_CODES#JELLY_BEAN} an alternative  
 * to using this with starting activities is to supply the desired animation  
 * information through a {@link ActivityOptions} bundle to  
 * {@link #startActivity(Intent, Bundle)} or a related function. This allows  
 * you to specify a custom animation even when starting an activity from  
 * outside the context of the current top activity.  
 *  
 * @param enterAnim A resource ID of the animation resource to use for  
 * the incoming activity. Use 0 for no animation.  
 * @param exitAnim A resource ID of the animation resource to use for  
 * the outgoing activity. Use 0 for no animation.  
 */  
public void overridePendingTransition(int enterAnim, int exitAnim);
```



# Activity 切换动画 - 示例

```
// 进入动画
startActivity(new Intent(MainActivity.this, activityClass));
overridePendingTransition(android.R.anim.fade_in, android.R.anim.fade_out);

// 退出动画
@Override
public void finish() {
    super.finish();
    overridePendingTransition(android.R.anim.fade_in, android.R.anim.fade_out);
}

// anim/fade_in
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@interpolator/decelerate_quad"
    android:fromAlpha="0.0" android:toAlpha="1.0"
    android:duration="@android:integer/config_longAnimTime" />

// anim/fade_out
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@interpolator/accelerate_quad"
    android:fromAlpha="1.0"
    android:toAlpha="0.0"
    android:duration="@android:integer/config_mediumAnimTime"
/>
```



# Drawable 动画

- AnimationDrawable
- AnimationVectorDrawable
- Lottie

# 示例 - AnimationDrawable

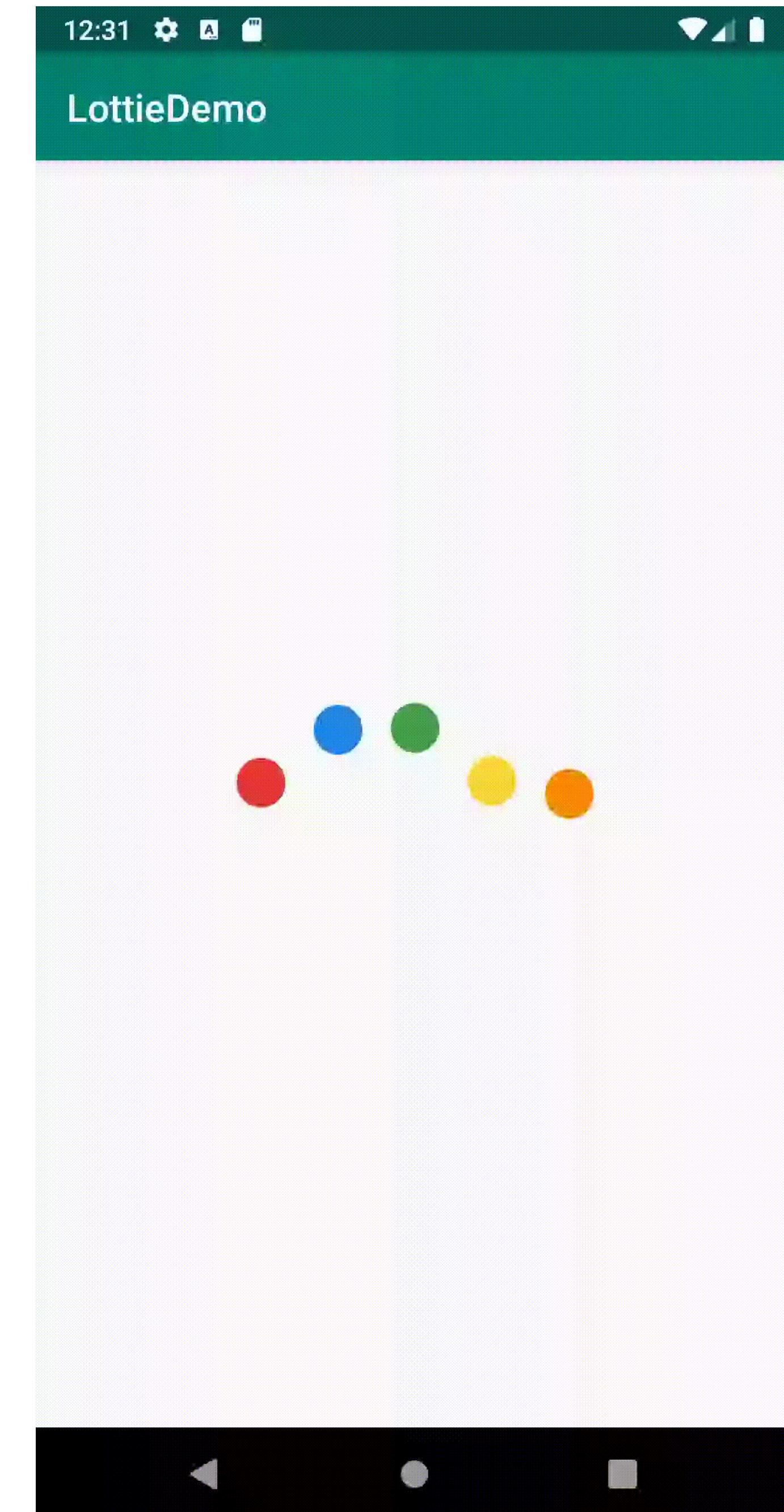
```
// res/drawable/rocket.xml
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
    <item android:drawable="@drawable/rocket_thrust1" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3" android:duration="200" />
</animation-list>

// activity
ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
rocketImage.setBackgroundResource(R.drawable.rocket_thrust);
rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
rocketAnimation.start();
```

# 示例 - Lottie

```
// app/build.gradle, 添加依赖
dependencies {
    ...
    implementation 'com.airbnb.android:lottie:2.7.0'
}

<com.airbnb.lottie.LottieAnimationView
    android:id="@+id/animation_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    app:lottie_autoPlay="true"
    app:lottie_loop="true"
    app:lottie_rawRes="@raw/material_wave_loading" />
```



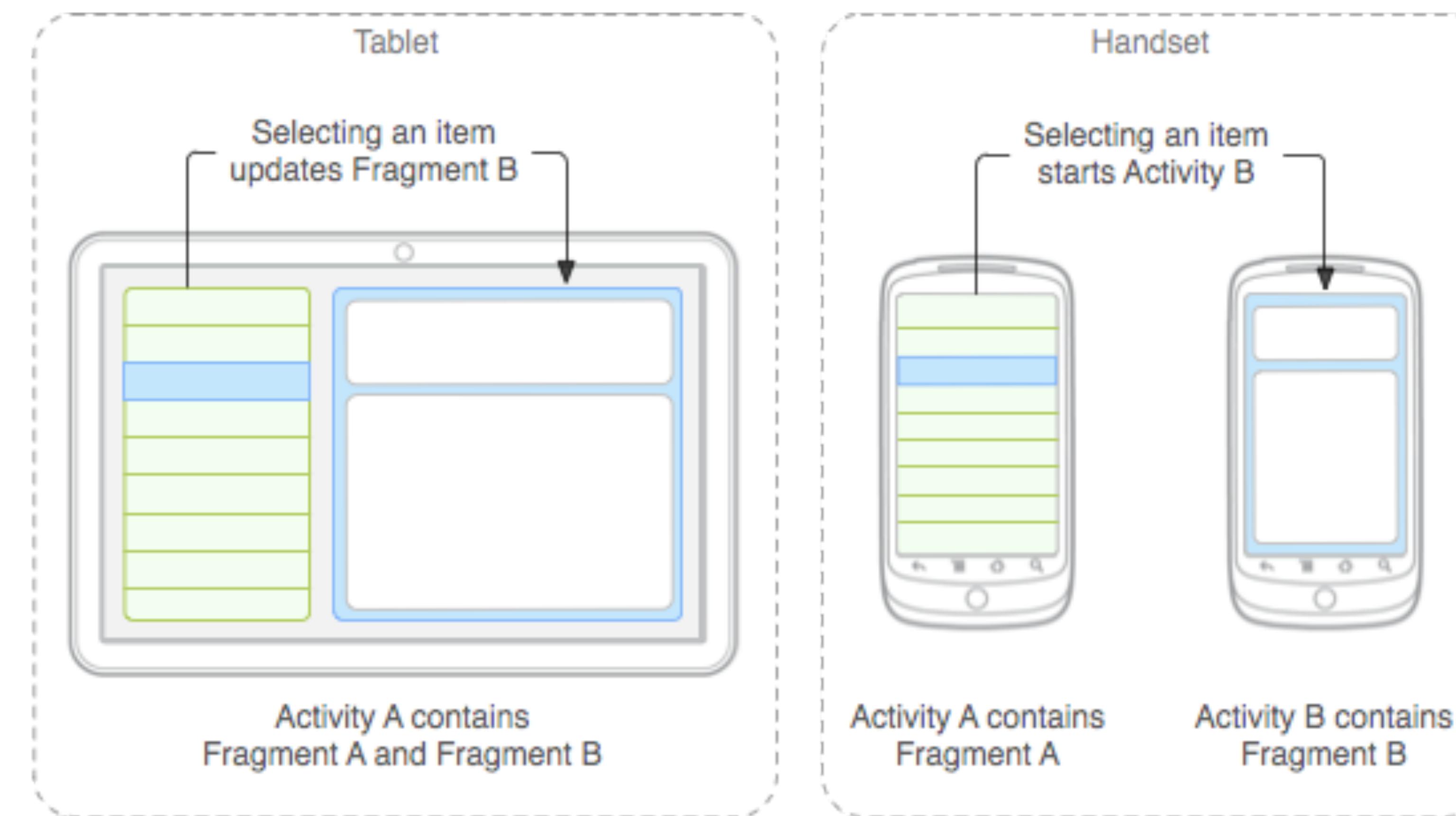
# Fragment



# Fragment

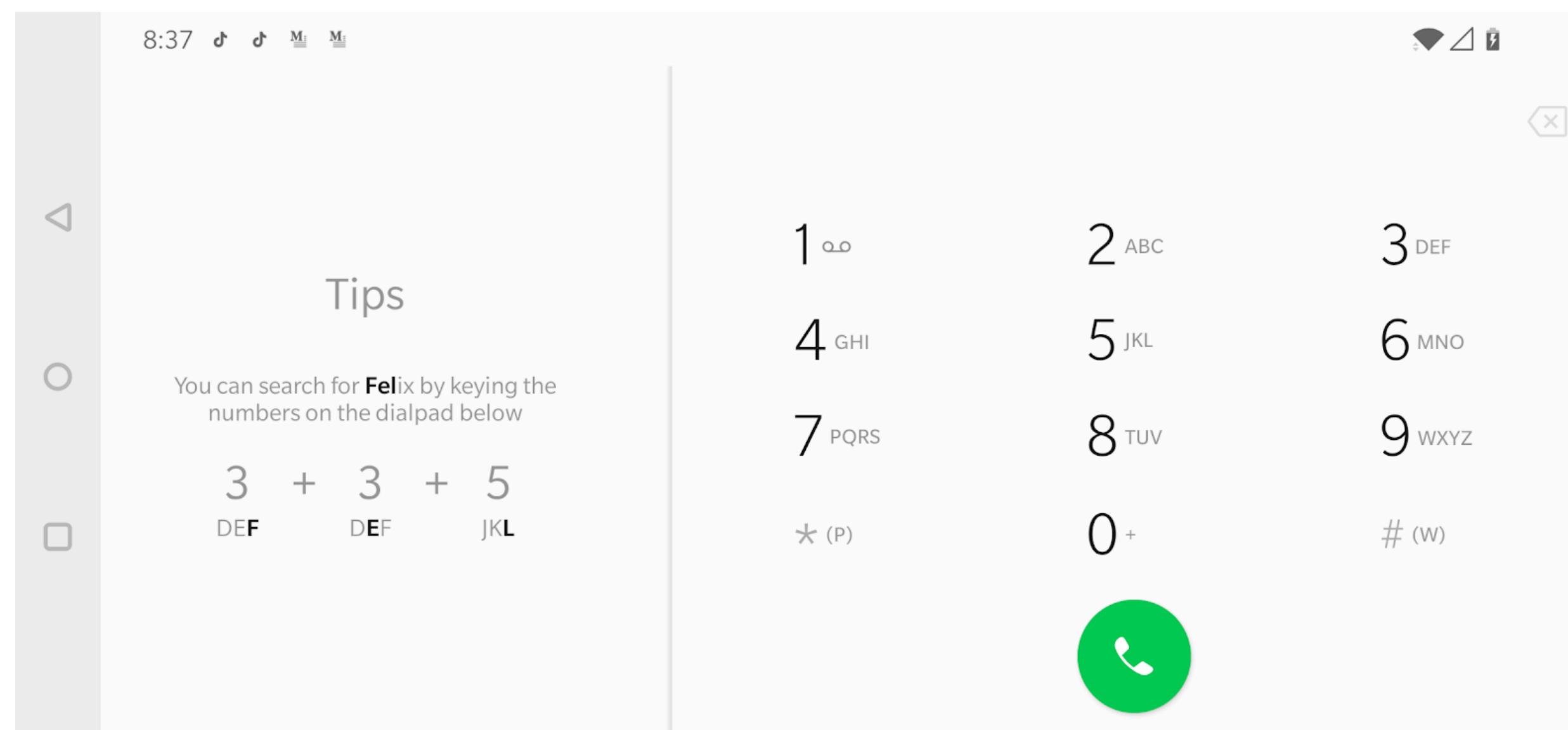
- 概念和作用
- 生命周期和基本用法
- 结合 ViewPager 创建多 Tab 界面
- 如何和 Activity 通信

# Fragment - UI 重用



# Fragment - Responsive Design

8:36 ⚡ ⚡ M M

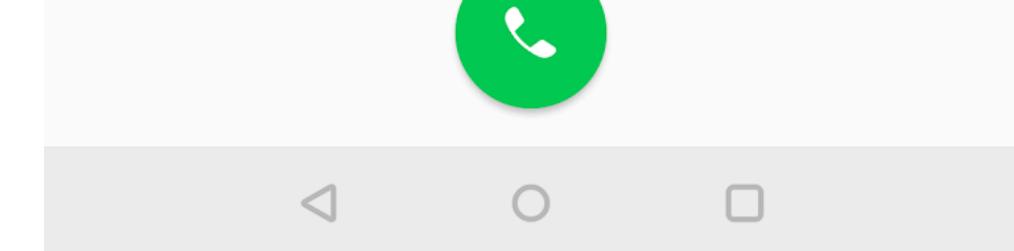


## Tips

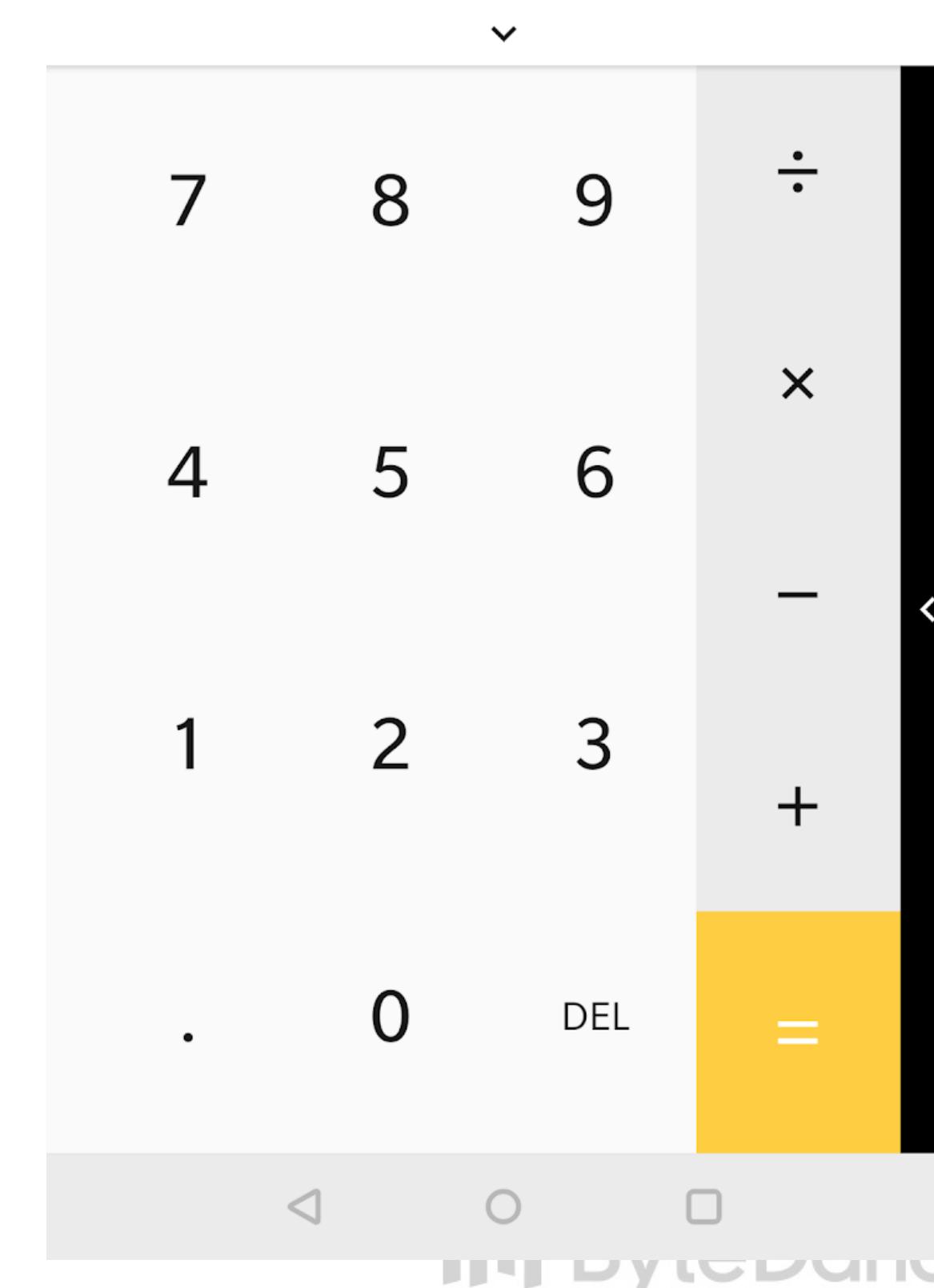
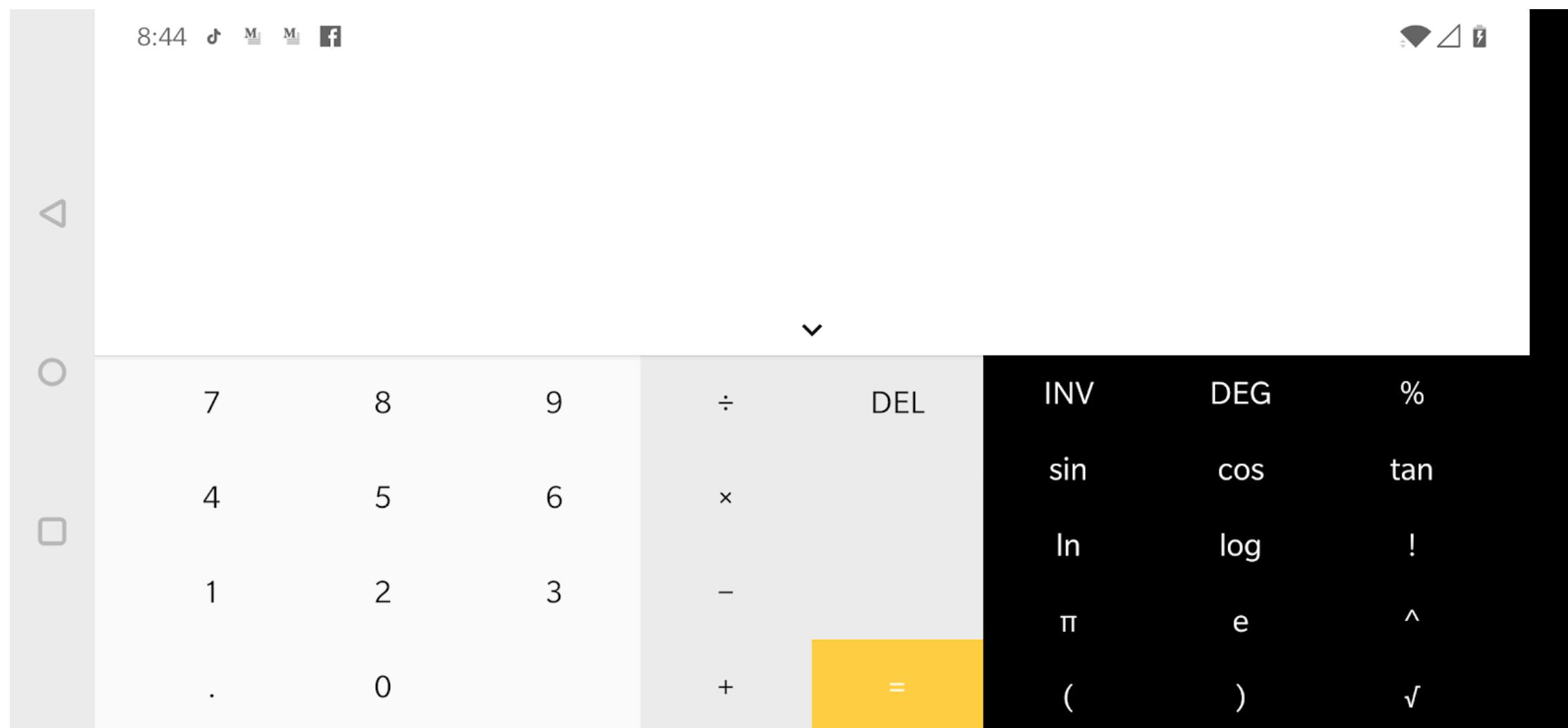
You can search for **Felix** by keying the numbers on the dialpad below

3 + 3 + 5  
DEF DEF JKL

1 <small>Q</small>	2 <small>ABC</small>	3 <small>DEF</small>
4 <small>GHI</small>	5 <small>JKL</small>	6 <small>MNO</small>
7 <small>PQRS</small>	8 <small>TUV</small>	9 <small>WXYZ</small>
*	0 <small>+</small>	# <small>(W)</small>



# Fragment - Responsive Design



# Fragment - Why

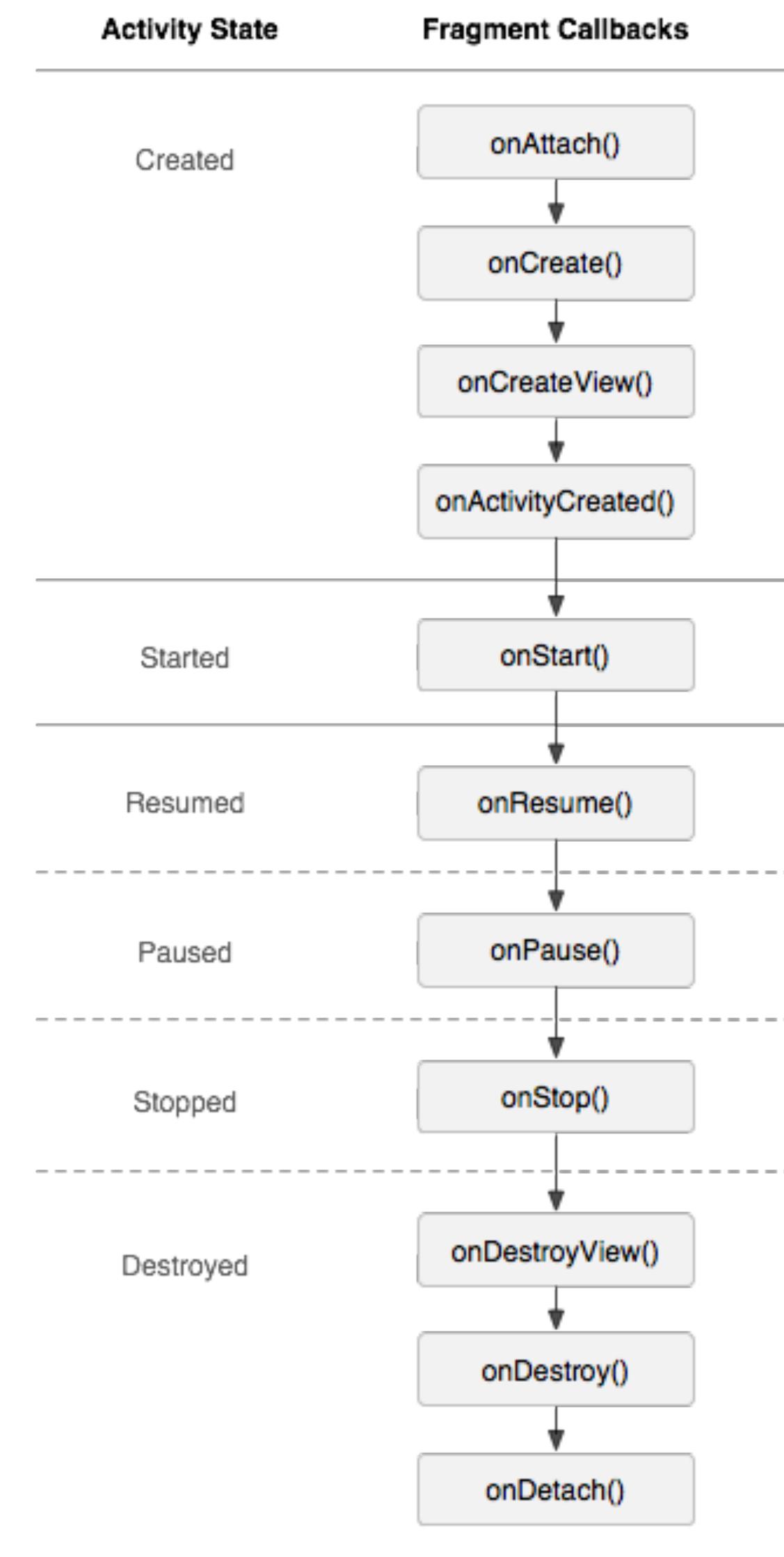
- Activity 模块化
- 相比 View, 带有生命周期管理
- 可重用, 灵活

# 使用哪个？

Library	Package
Support Library	<code>android.support.v4.app.Fragment</code>
AndroidX Library	<code>androidx.fragment.app.Fragment</code>
Native	<code>android.app.Fragment</code>

# 生命周期

- onAttach/onDetach
- onCreate/onDestroy
- onCreateView/onDestroyView
- onActivityCreated
- onStart/onStop
- onResume/onPause



# 示例 - Lifecycle

- 定义 fragment 布局文件
- 定义 fragment 类
- 在 activity 布局文件中嵌入 fragment

# 示例 - Lifecycle - 1

- fragment\_hello.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/hello_fragment" />
</FrameLayout>
```

# 示例 - Lifecycle - 2

- HelloFragment.java

```
public class HelloFragment extends Fragment {

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

# 示例 - Lifecycle - 3

- **activity\_fragment.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <fragment
        android:id="@+id/hello_fragment"
        android:name="com.example.chapter3.demo.fragment.HelloFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

# 动态添加/删除 Fragment

- Fragment 容器
  - 定义 Fragment 的位置和大小
- FragmentManager
  - 动态添加/替换/删除 Fragment
  - FragmentTransaction

# 示例 - 动态修改 Fragment - 1

- 在 activity 布局文件中定义 fragment 容器

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

# 示例 - 动态修改 Fragment - 2

- 使用 FragmentManager 添加 Fragment

```
public class DynamicAddFragmentActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dynamic_add_fragment);

        getSupportFragmentManager()
            .beginTransaction()
            .add(R.id.fragment_container, new HelloFragment())
            .commit();
    }
}
```

# ViewPager + Fragment

- 常用于实现可滑动的多个视图
- 容器，类似于 ListView/RecyclerView
- 需要通过 Adapter 配置内容
- 内容一般通过 Fragment 来实现
- 可配合 TabLayout 或三方库添加 Title

# 示例 - ViewPager - 1

- 在布局 xml 中添加 ViewPager

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.v4.view.ViewPager
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</FrameLayout>
```

# 示例 - ViewPager - 2

- 通过 Adapter 配置页面的 Fragment

```
public class ViewPagerActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_pager);
        ViewPager pager = findViewById(R.id.view_pager);
        pager.setAdapter(new FragmentPagerAdapter(getSupportFragmentManager()) {
            @Override
            public Fragment getItem(int i) {
                return new HelloFragment();
            }

            @Override
            public int getCount() {
                return 3;
            }
        });
    }
}
```

# 示例 - ViewPager + TabLayout - 1

- 在布局 xml 中继续添加 TabLayout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">  
  
    <android.support.design.widget.TabLayout  
        android:id="@+id/tab_layout"  
        android:layout_width="match_parent"  
        android:layout_height="40dp" />  
  
    <android.support.v4.view.ViewPager  
        android:id="@+id/view_pager"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent" />  
  
</LinearLayout>
```

# 示例 - ViewPager + TabLayout - 2

- 在代码中对 ViewPager 和 TabLayout 建立关联

```
ViewPager pager = findViewById(R.id.view_pager);
TabLayout tabLayout = findViewById(R.id.tab_layout);
pager.setAdapter(new FragmentPagerAdapter(getSupportFragmentManager()) {
    @Override
    public Fragment getItem(int i) {
        return new HelloFragment();
    }

    @Override
    public int getCount() {
        return PAGE_COUNT;
    }

    @Override
    public CharSequence getPageTitle(int position) {
        return "Hello " + position;
    }
});
tabLayout.setupWithViewPager(pager);
```

# Fragment/Activity 之间的通信

- 构造 Fragment 时传递参数 (setArguments/getArguments)
- 通过接口和回调

# 示例 - 通信 - 传参

```
public final class ColorFragment extends Fragment {
    private static final String KEY_EXTRA_COLOR = "extra_color";

    public static ColorFragment newInstance(int color) {
        ColorFragment cf = new ColorFragment();
        Bundle args = new Bundle();
        args.putInt(KEY_EXTRA_COLOR, color);
        cf.setArguments(args);
        return cf;
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
                           @Nullable ViewGroup container,
                           @Nullable Bundle savedInstanceState) {
        int color = Color.BLUE;
        Bundle args = getArguments();
        if (args != null) {
            color = args.getInt(KEY_EXTRA_COLOR, Color.BLUE);
        }
        View view = inflater.inflate(R.layout.fragment_color, container, false);
        view.setBackgroundColor(color);
        return view;
    }
}
```

# 示例 - 通信 - Listener - 1

```
public final class ColorPlusFragment extends Fragment {

    public interface Listener {
        void onCollectColor(int color);
    }

    private Listener mListener;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof Listener) {
            mListener = (Listener) context;
        }
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
                           @Nullable ViewGroup container,
                           @Nullable Bundle savedInstanceState) {
        ...
        // fire event when needed
        if (mListener != null) {
            mListener.onCollectColor(color);
        }
        return view;
    }
}
```

# 示例 - 通信 - Listener - 2

```
public class ViewPagerCommunicationActivity extends AppCompatActivity
    implements ColorPlusFragment.Listener {

    ...

    @Override
    public void onCollectColor(int color) {
        mCollectAdapter.addColor(color);
    }

    ...
}
```

# 示例 - Master Detail

- Portrait
  - Master Activity: Item List
  - Detail Activity: Item Detail
- Landscape
  - One Activity: List & Detail



# Reference



# Android Developer

- 开发者官网：<https://developer.android.com>
- 开发者官网中文站：<https://developer.android.google.cn/>

# Android Studio

- AS 快捷键：[https://developer.android.google.cn/studio/  
intro/keyboard-shortcuts](https://developer.android.google.cn/studio/intro/keyboard-shortcuts)
- AS 用户指南：[https://developer.android.google.cn/  
studio/intro](https://developer.android.google.cn/studio/intro)

# Animation

- Material Design - Motion: <https://material.io/design/motion/>
- Property Animation: <https://developer.android.google.cn/guide/topics/graphics/prop-animation>

# Animation

- Lottie Android 使用指南: <https://airbnb.io/lottie/#/android>
- LottieFiles 查找 lottie 资源: <https://lottiefiles.com/>

# Fragment

- Fragment Overview: <https://developer.android.google.cn/guide/components/fragments>

# Exercise



# 准备工作

- 本章内容的讲义和课后作业都放在 Github 上，对应仓库地址是 <https://github.com/bytedance-android/Chapter-3>。请同学们 fork 到自己的 github 仓库后，clone 到本地，修改代码完成练习，最终将修改 push 到自己的仓库中，并发送邮件到 liucheng@bytedance.com.

# 准备工作 - 1

1. 打开 <https://github.com/bytedance-android/Chapter-3>, 并 fork 到自己的仓库

bytedance-android / Chapter-3

No description, website, or topics provided.

Manage topics

4 commits 1 branch 0 releases 1 contributor

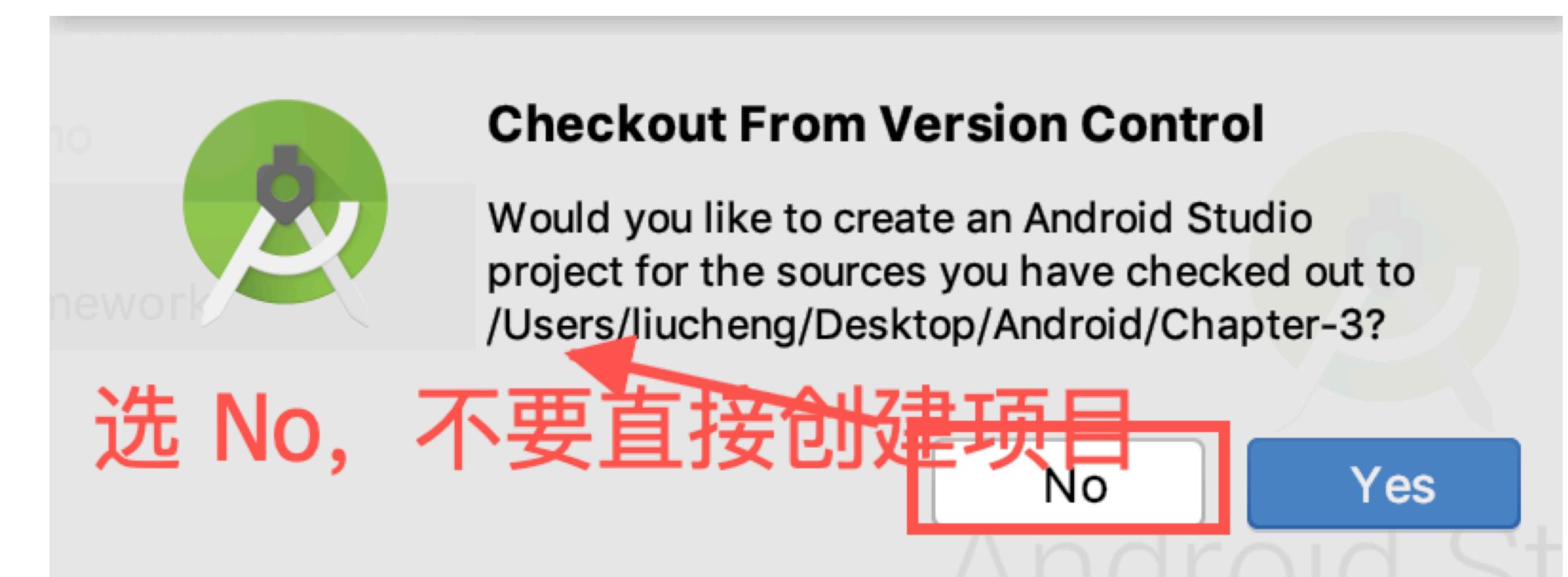
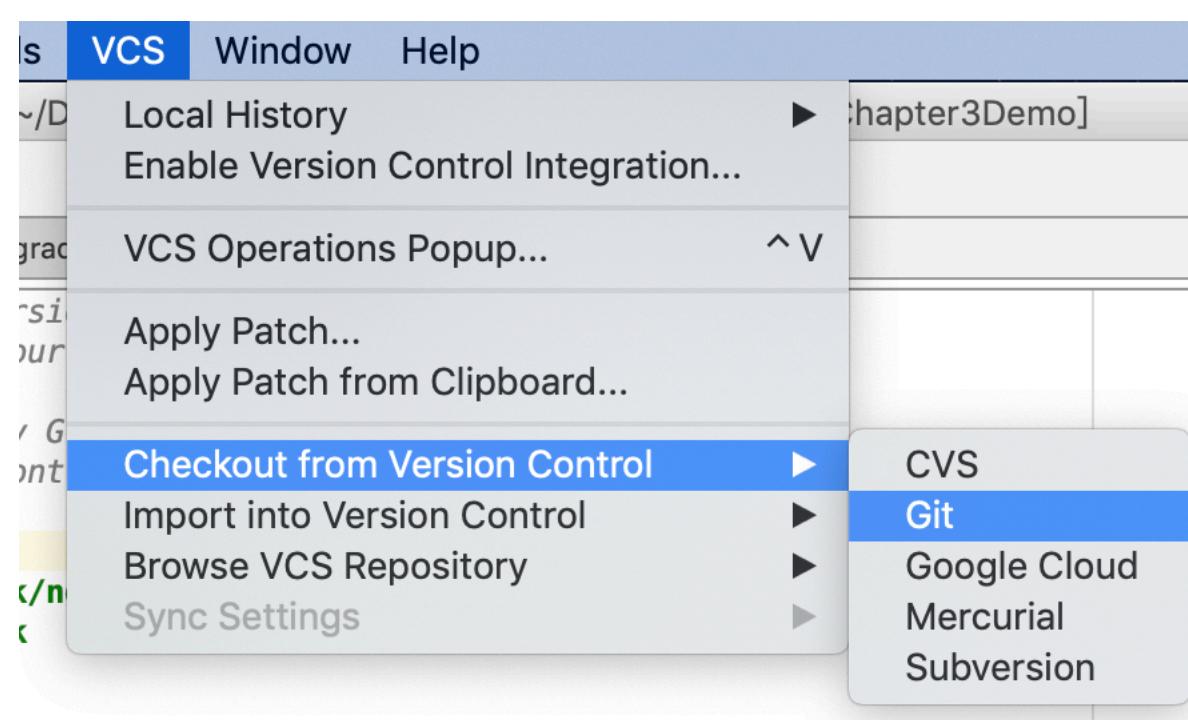
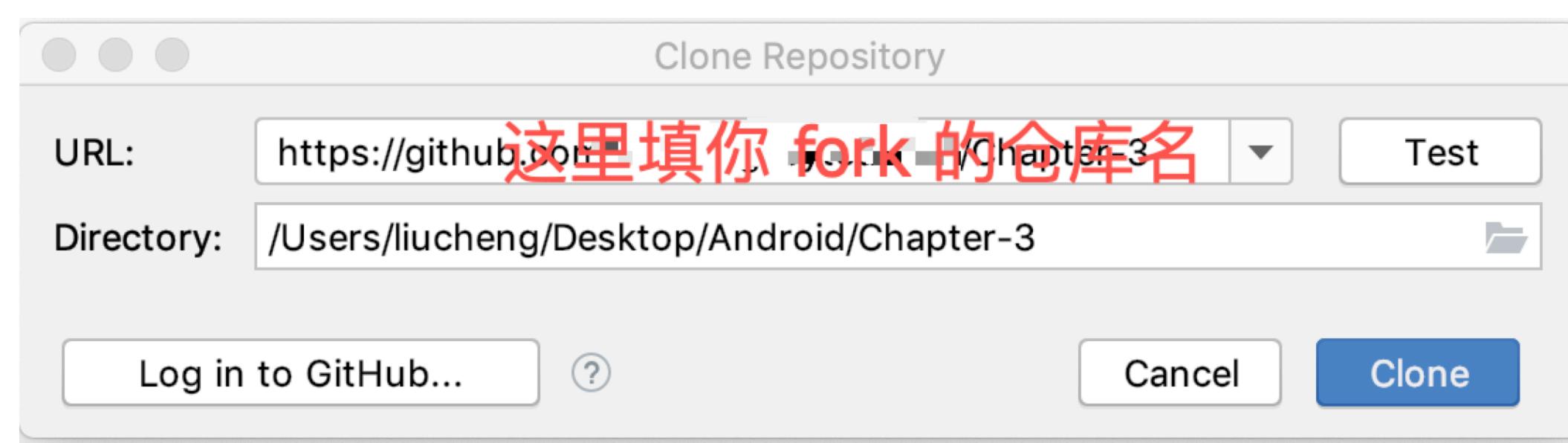
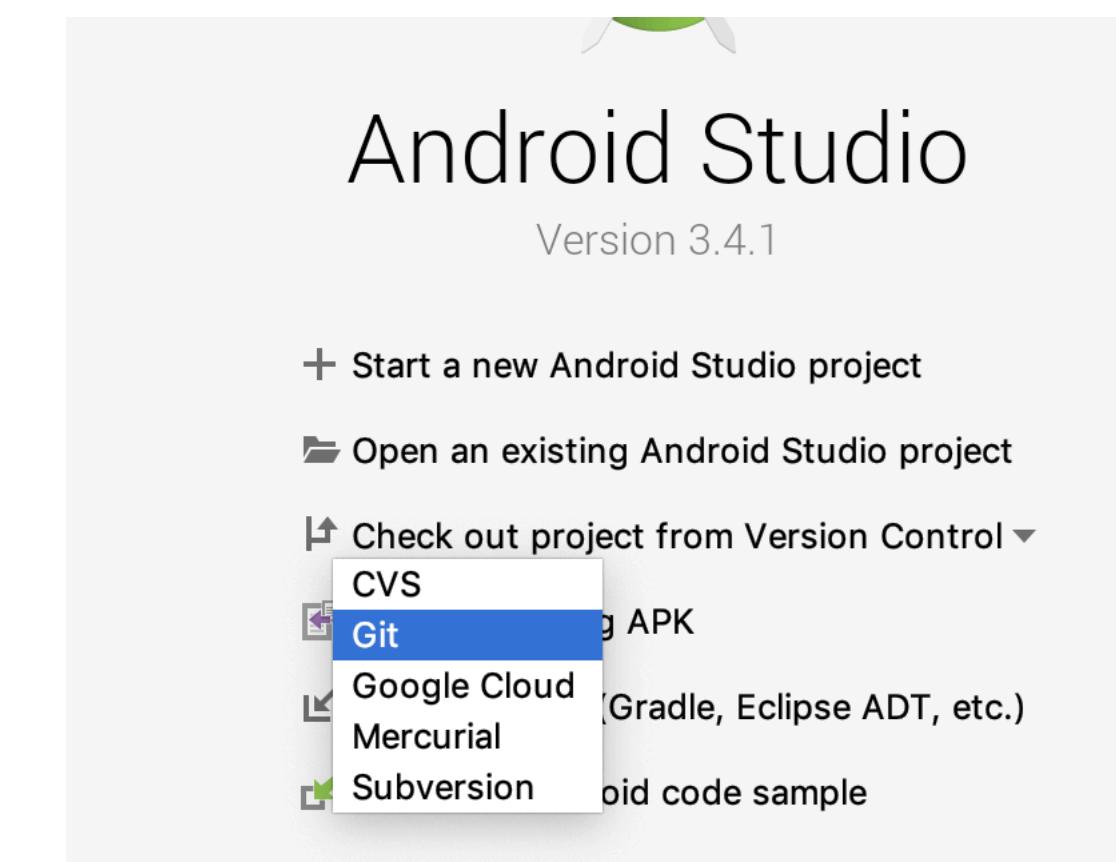
Branch: master ▾ New pull request Create new file Upload files Find File Clone or download ▾

liucheng-bytedance add Homework Latest commit 40865ca 15 minutes ago

File	Commit Message	Time
Homework	add Homework	15 minutes ago
.gitignore	add Homework	15 minutes ago
Chapter-3.pdf	update ppt	20 minutes ago
README.md	Initial commit	5 days ago

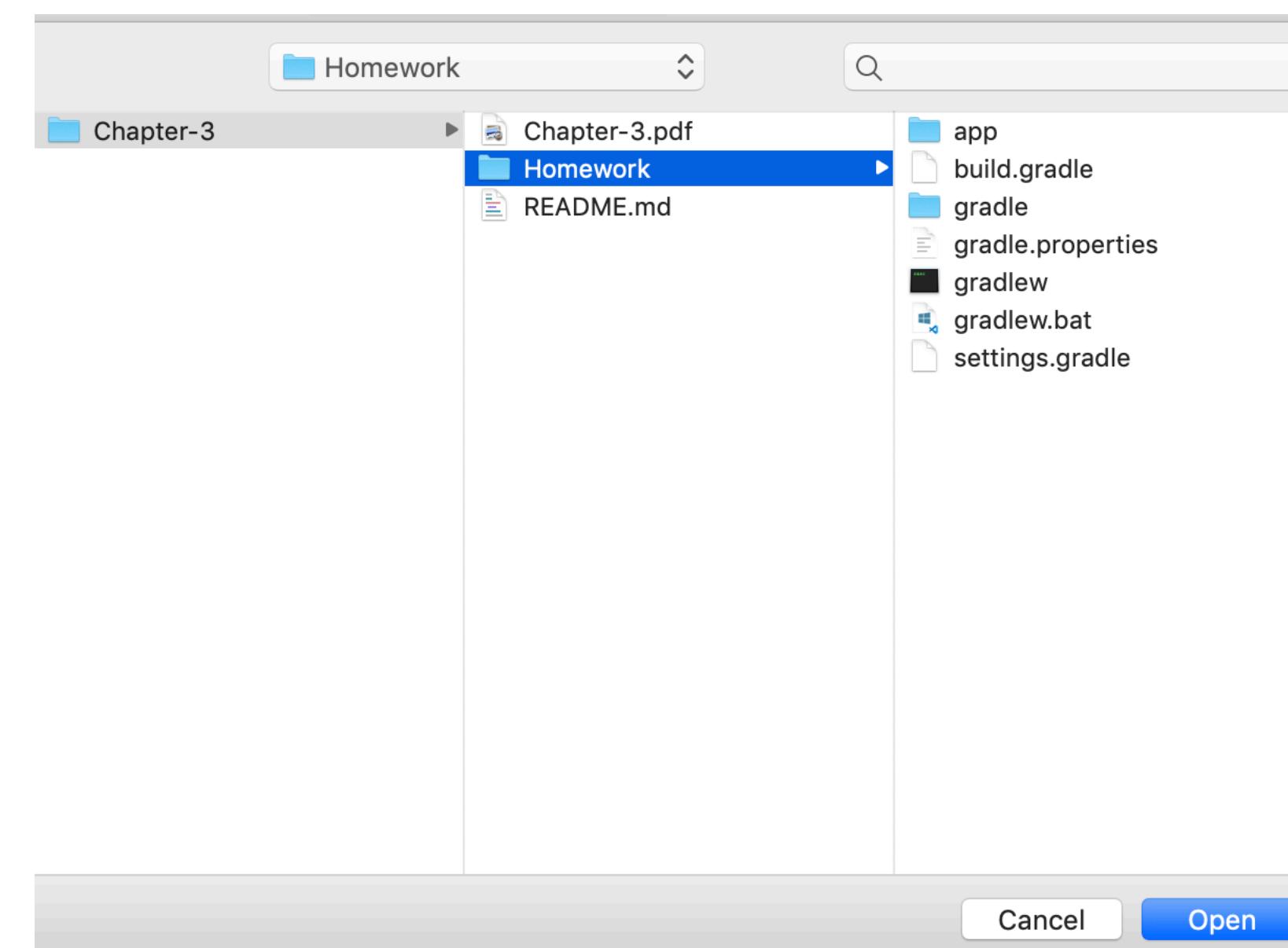
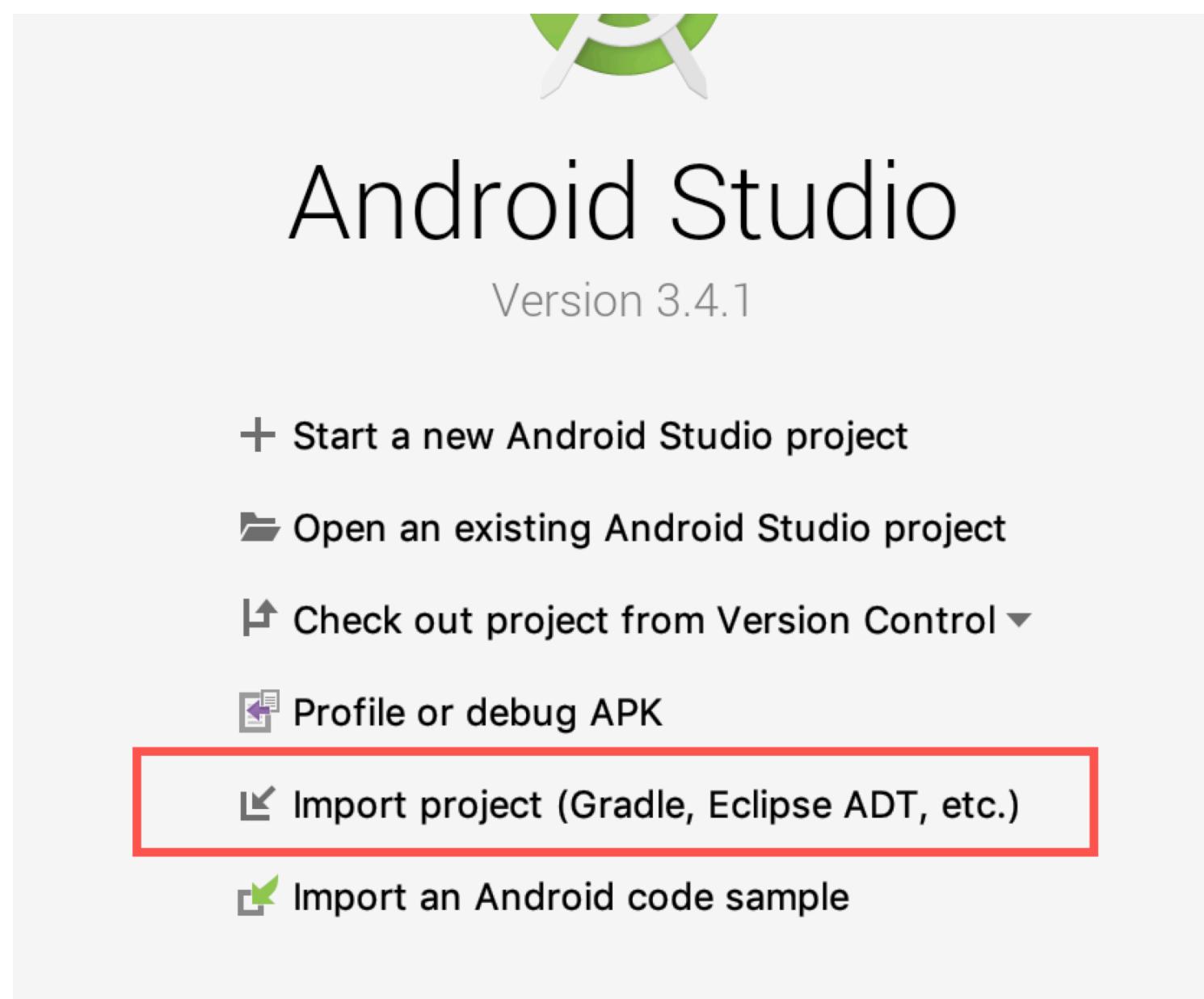
# 准备工作 - 2

2. 使用 Android Studio 将自己仓库中的 Chapter3 clone 到本地，完成后注意不要点 Yes 直接创建项目



# 准备工作 - 3

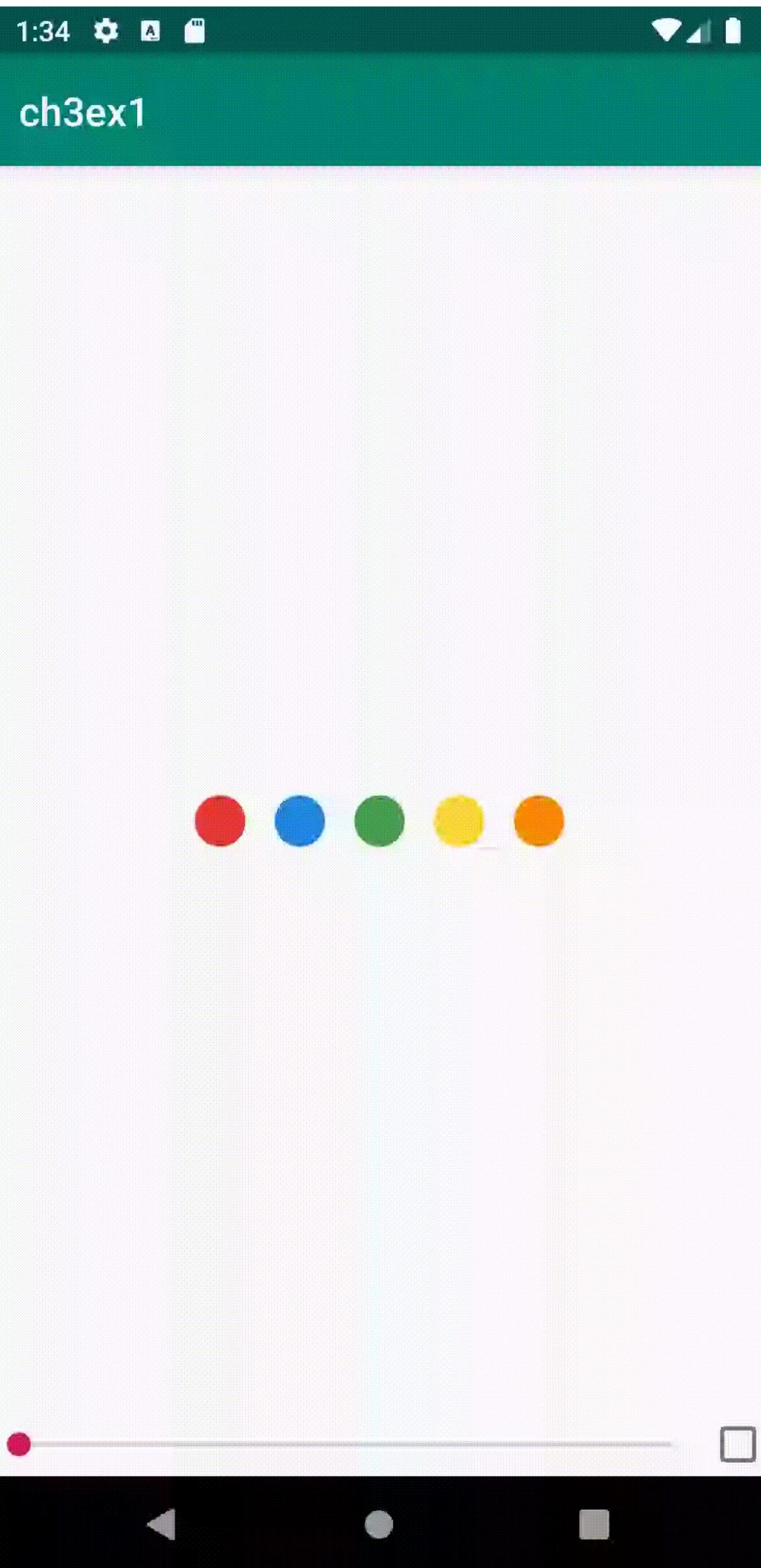
3. 导入刚才 clone 到本地的项目中的 Homework 目录，试运行能正常启动 app 即可



# 练习1 - ch3ex1

- 引入 Lottie 库实现简单的图标动画
- 1. 在 activity\_main.xml 中添加 LottieAnimationView
- 2. 在SeekBar 的回调中修改 LottieAnimationView 的进度

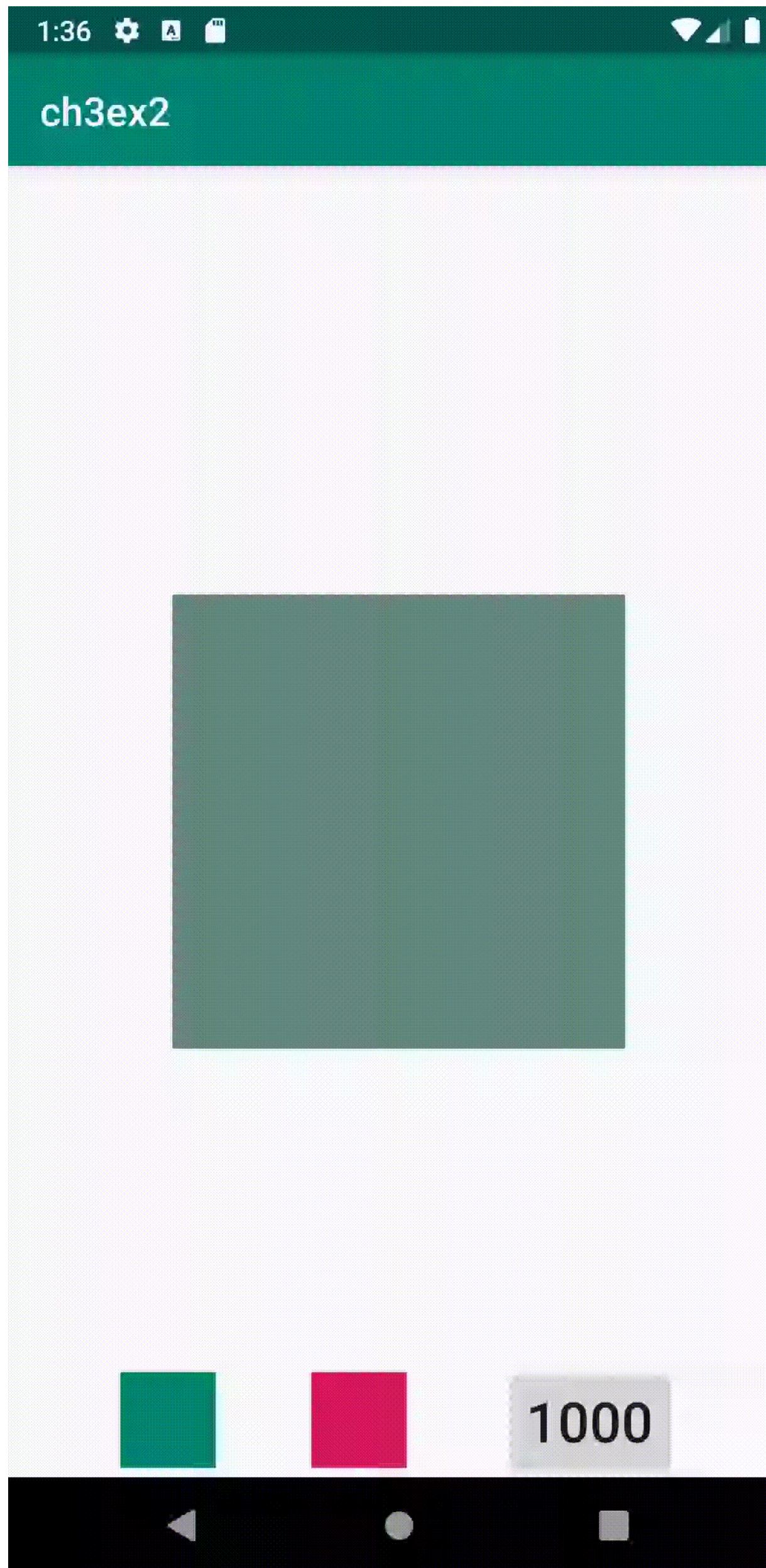
# 练习1 - ch3ex1



## 练习2 - ch3ex2

- 使用属性动画，练习 AnimatorSet 和 scale/fade 等基本动画样式
- 1. 添加 scale 动画
- 2. 添加 alpha 动画
- 3. 组合到 AnimatorSet 中

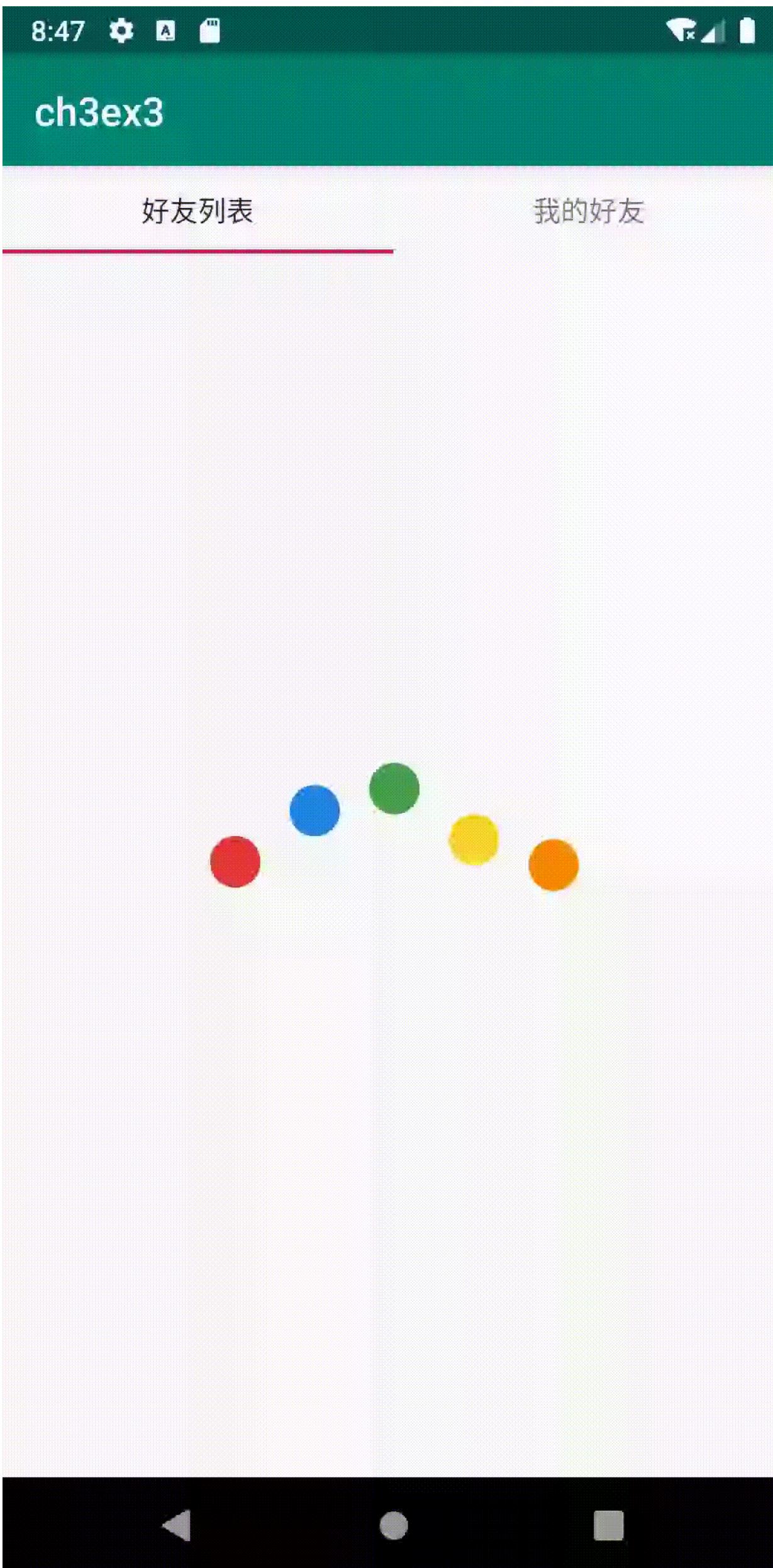
# 练习2 - ch3ex2



## 练习3 - ch3ex3

- 使用 ViewPager 和 Fragment 做一个简单版的好友列表界面
- 1. 使用 ViewPager 和 Fragment 做个可滑动界面
- 2. 使用 TabLayout 添加 Tab 支持
- 3. 对于好友列表 Fragment，使用 Lottie 实现 Loading 效果，在 5s 后展示实际的列表，要求这里的动效是淡入淡出

# 练习3 - ch3ex3





THANKS.



刘成 ([liucheng@bytedance.com](mailto:liucheng@bytedance.com))