**应用Twire详细报告**

## 应用简介

**应用名称：**Twire

**应用类型：**Multi-Media

**应用描述：**Twire 是一款适用于 Android 的开源、无广告的 Twitch流媒体播放器。

## 功能场景与详细设计

|  |  |  |
| --- | --- | --- |
| 场景序号 | 场景名称 | 包含组件数 |
| 1 | 热门流媒体显示场景 | 2 |
| 2 | 热门游戏显示场景 | 2 |
| 3 | 搜索场景 | 2 |
| 4 | 设置场景 | 5 |

### 热门流媒体显示场景

**主要功能职责：**打开侧边栏点击 ”TopStreams” 按钮后会在页面显示当前热门的流媒体

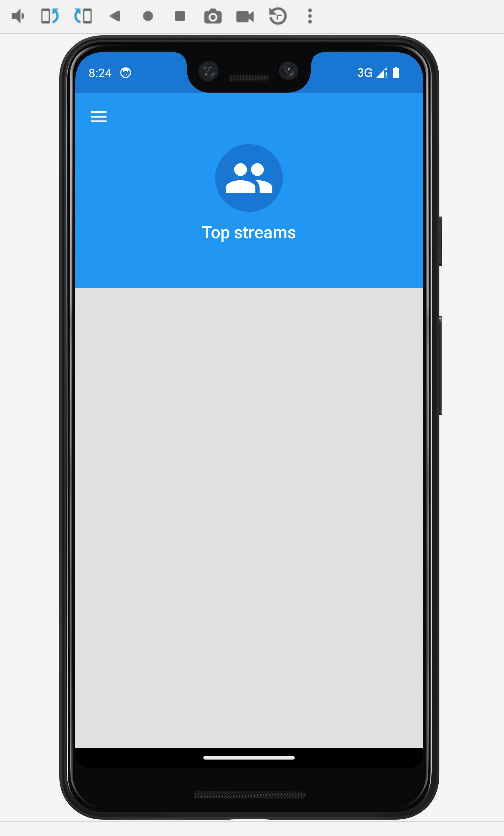
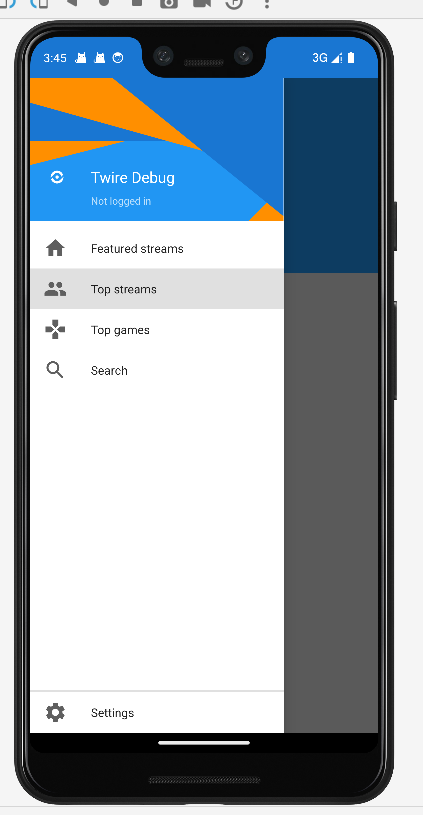
**与其他功能场景的切换：**

* 打开侧边栏，点击 ”Settings” 按钮，可切换至设置场景；
* 打开侧边栏，点击 ”Search” 按钮，可切换至搜索场景；
* 打开侧边栏，点击 ”Top games” 按钮，可切换至热门游戏显示场景；

**包含组件：** MainActivity, TopStreamsActivity

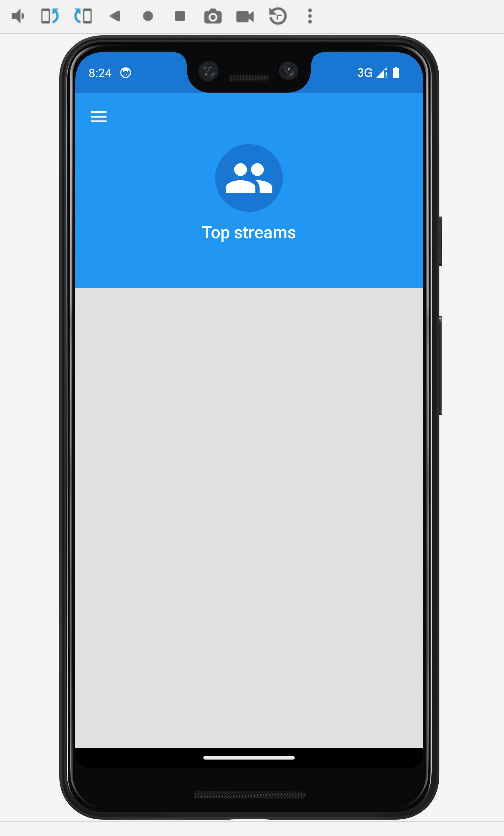
**MainActivity:**

* **功能职责:** 作为应用打开的主界面，默认情况下会显示当前热门的流媒体，并且会通过customizeActivity()进行定制化的界面显示
* **截图**



**TopStreamsActivity:**

* **功能职责:** 显示热门的流媒体内容并在主界面显示
* **截图**



**场景内组件间跳转关系（图像可选）：**

* MainActivity在应用打开后默认显示TopStreamsActivity的内容。

### 热门游戏显示场景

**主要功能职责：**打开侧边栏点击 ”Top games” 按钮后会在页面显示当前热门的流媒体

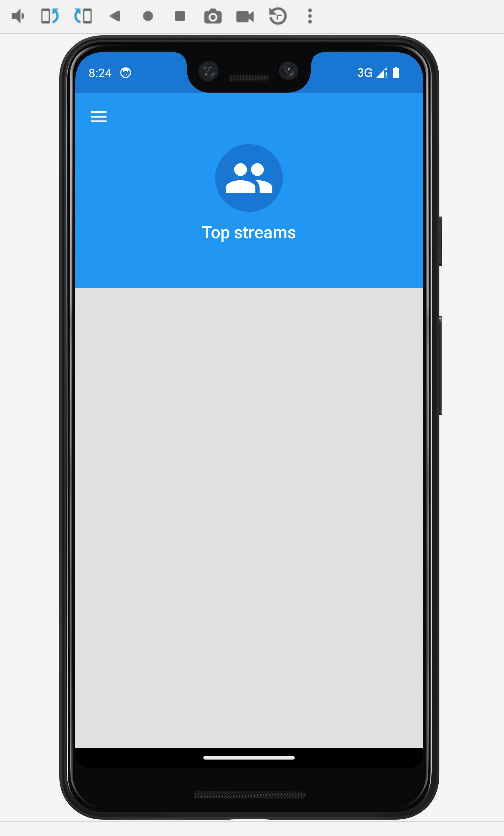
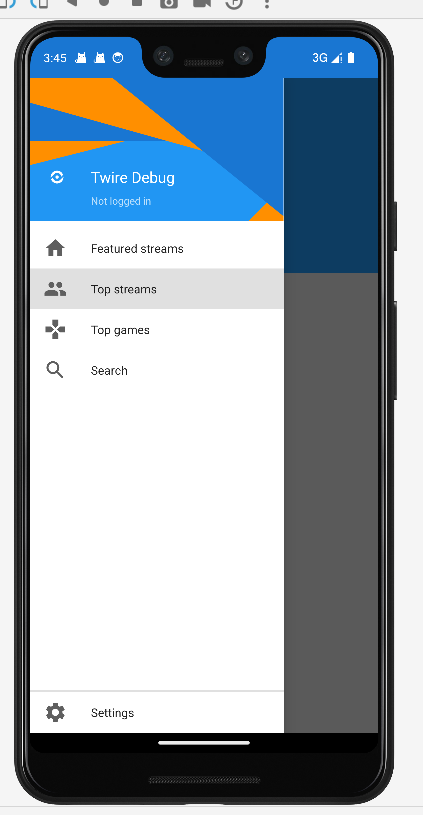
**与其他功能场景的切换：**

* 打开侧边栏，点击 ”Settings” 按钮，可切换至设置场景；
* 打开侧边栏，点击 ”Search” 按钮，可切换至搜索场景；
* 打开侧边栏，点击 ”Top streams” 按钮，可切换至热门流媒体显示场景；

**包含组件：** MainActivity, TopGamesActivity

**MainActivity:**

* **功能职责:** 作为应用打开的主界面，默认情况下会显示当前热门的流媒体，并且会通过customizeActivity()进行定制化的界面显示
* **截图**



**TopGamesActivity:**

* **功能职责:** 显示热门的游戏内容并在主界面显示
* **截图**



**场景内组件间跳转关系（图像可选）：**

* 通过定制化操作可以让MainActivity在主页显示TopGamesActivity的内容。

### 2.3设置显示场景

**主要功能职责：**打开侧边栏点击 ”Settings” 按钮后会在页面显示当前热门的流媒体

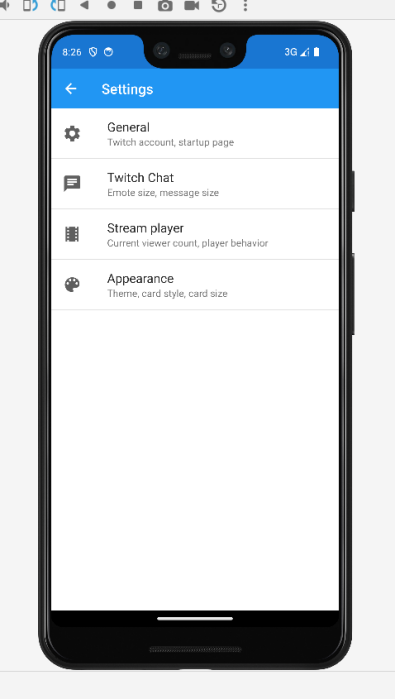
**与其他功能场景的切换：**

* 打开侧边栏，点击 ”Top Games” 按钮，可切换至游戏直播显示场景；
* 打开侧边栏，点击 ”Search” 按钮，可切换至搜索场景；
* 打开侧边栏，点击 ”Top streams” 按钮，可切换至热门流媒体显示场景。

**包含组件：**SettingsActivity、ThemeActivity、SettingsGeneralActivity、SettingsTwitchChatActivity、SettingsStreamPlayerActivity、SettingsAppearanceActivity

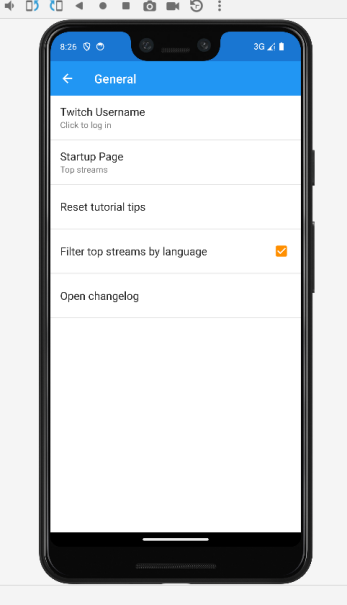
**SettingsActivity:**

* **功能职责:** 作为应用打开设置的主界面，默认情况下会显示设置选项
* **截图**



**SettingsGeneralActivity:**

* **功能职责:** 作为应用打开General的界面，默认情况下会显示General设置选项
* **截图**

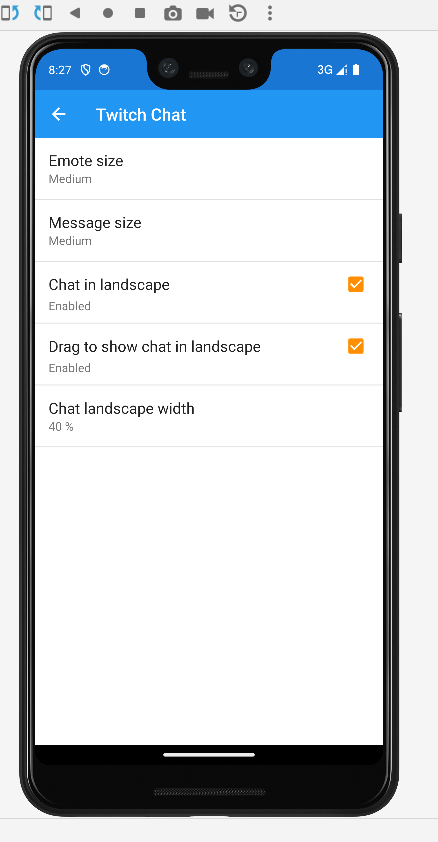


**场景内组件间跳转关系：**

* 通过点击操作可以让SettingsActivity在主页显示SettingsGeneralActivity的内容。

**SettingsTwitchChatActivity:**

* **功能职责:** 作为应用打开TwitchChat的界面，默认情况下会显示TwitchChat设置选项，设置对话的字体、颜色等
* **截图**

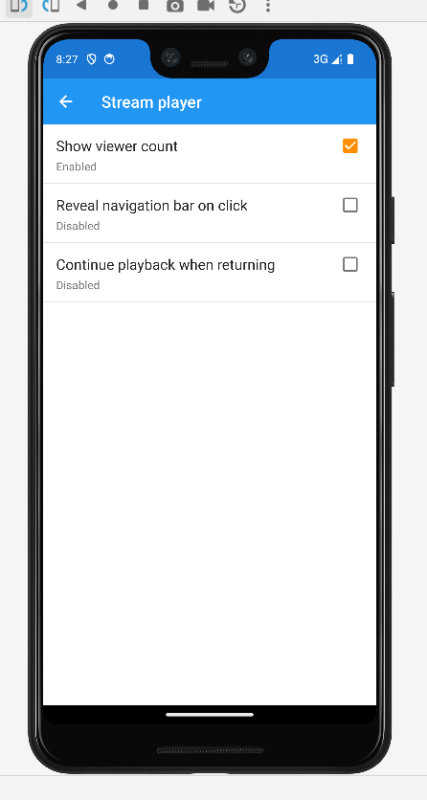


**场景内组件间跳转关系：**

* 通过点击操作可以让SettingsActivity在主页显示SettingsTwitchChatActivity的内容。

**SettingsStreamPlayerActivity:**

* **功能职责:** 作为应用打开StreamPlayer的界面，默认情况下会显示StreamPlayer设置选项
* **截图**

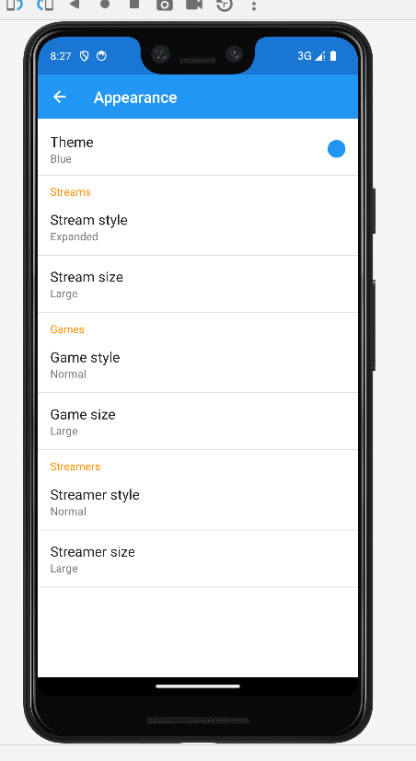


**场景内组件间跳转关系：**

* 通过点击操作可以让SettingsActivity在主页显示SettingsStreamPlayerActivity的内容。

**SettingsAppearanceActivity:**

* **功能职责:** 作为应用打开Appearance的界面，默认情况下会显示Appearance设置选项，设置主题、媒体风格、游戏风格等
* **截图**



**场景内组件间跳转关系：**

* 通过点击操作可以让SettingsActivity在主页显示SettingsAppearanceActivity的内容。

### 2.4搜索显示场景

**主要功能职责：**打开侧边栏点击 ”Search” 按钮后会在页面显示当前热门的流媒体

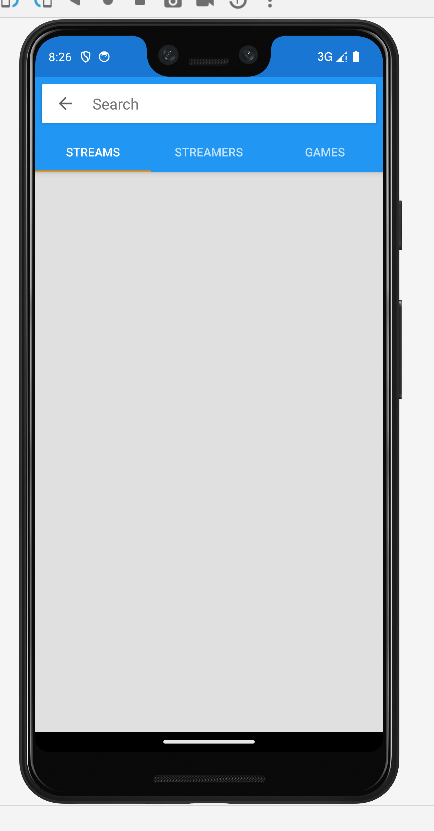
**与其他功能场景的切换：**

* 搜索栏下面有三个按钮，点击 ”Streams” 按钮，可切换至流媒体直播显示场景；
* 搜索栏下面有三个按钮，点击 ”Games” 按钮，可切换至游戏直播显示场景；
* 搜索栏下面有三个按钮，点击 ”Streamers” 按钮，可切换至设置场景。

**包含组件：**SearchActivity，MainActivity

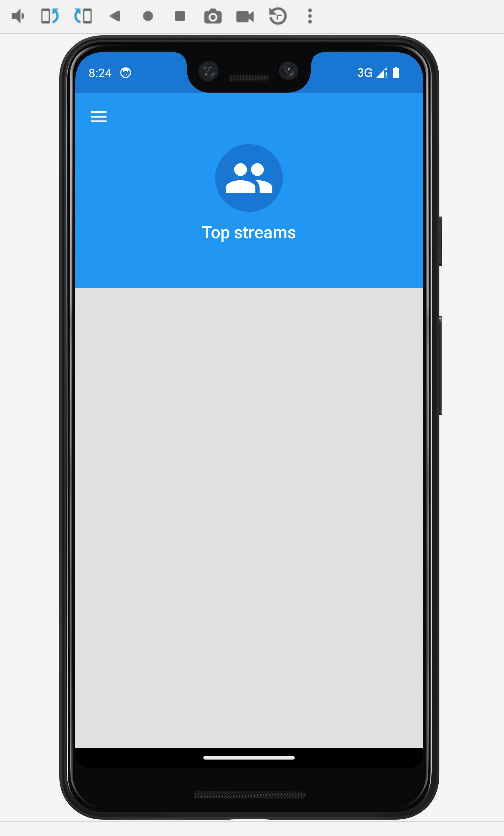
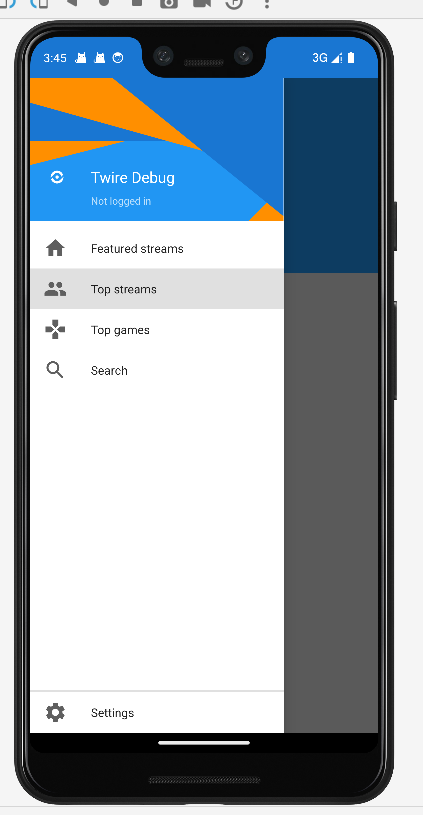
**SearchActivity:**

* **功能职责:** 作为应用打开搜索的主界面，默认情况下会显示设置选项，点击 ”STREAMS” 按钮，可切换至直播场景；点击 ”STREAMERS” 按钮，可切换至主播场景；点击 ”GAMES” 按钮，可切换至游戏场景。
* **截图**



**MainActivity:**

* **功能职责:** 作为应用打开的主界面，默认情况下会显示当前热门的流媒体，并且会通过customizeActivity()进行定制化的界面显示
* **截图**



**场景内组件间跳转关系：**

* 通过点击返回键会回到主页即MainActivity

## 场景内详细设计

### 适配器模式

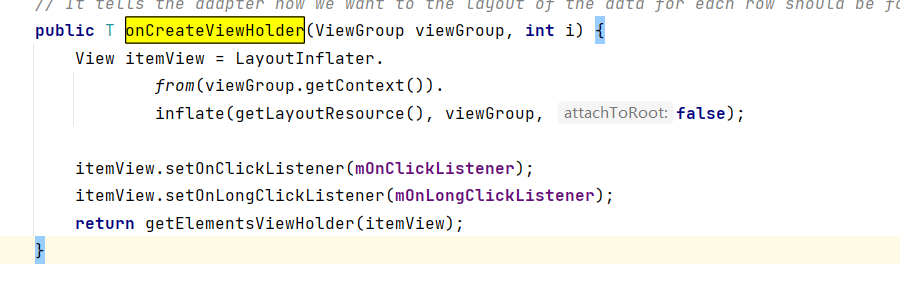
**功能场景位置：**热门流媒体显示场景

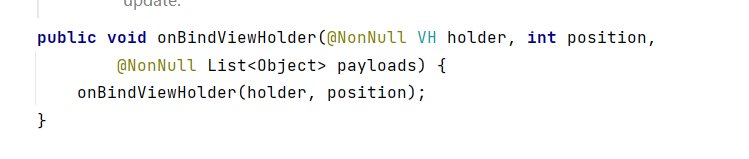
**组件位置：**TopStreamActivity

**涉及类：**TopStreamActivity, StreamAdapter, MainActivityAdapter，RecyclerView.Adapter, StreamInfo, StreamViewHolder

**描述（图像可选）：**

这是一种类适配器的模式。RecyclerView.Adapter是目标接口，它规定了适配器应该实现的方法，例如 onCreateViewHolder () 或 onBindViewHolder()，用于在界面上显示流媒体list的数据。MainActivityAdapter是一个通用的抽象适配器类，它继承RecyclerView并且对方法进行重写。StreamAdapter是适配器同时也是被适配者，它继承MainActivityAdapter获取重写的方法，并且用泛型<StreamInfo, StreamViewHolder>规定了list存放的是流媒体相关信息，同时负责将其中包含的List< StreamInfo >数据适配至StreamAdapter中进行列表显示。





### 3.2模板方法模式

**功能场景位置：**热门游戏显示场景

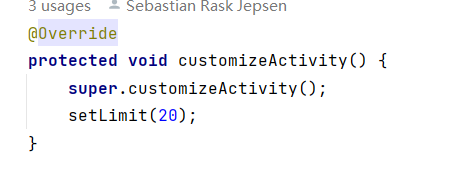
**组件位置：**TopGameActivity

**涉及类：**TopGameActivity, LazyMainActivity, MainActivity

**描述（图像可选）：**

TopGameActivity出现了模板方法模式的使用。在 MainActivity 中，定义了一系列抽象方法，例如 constructAdapter、customizeActivity、getActivityIconRes 等， GameActivity 继承LazyMainActivity，LazyMainActivity继承MainActivity， GameActivity重写了MainActivity定义的这些抽象方法来完成具体的操作。





### 3.3工厂方法模式

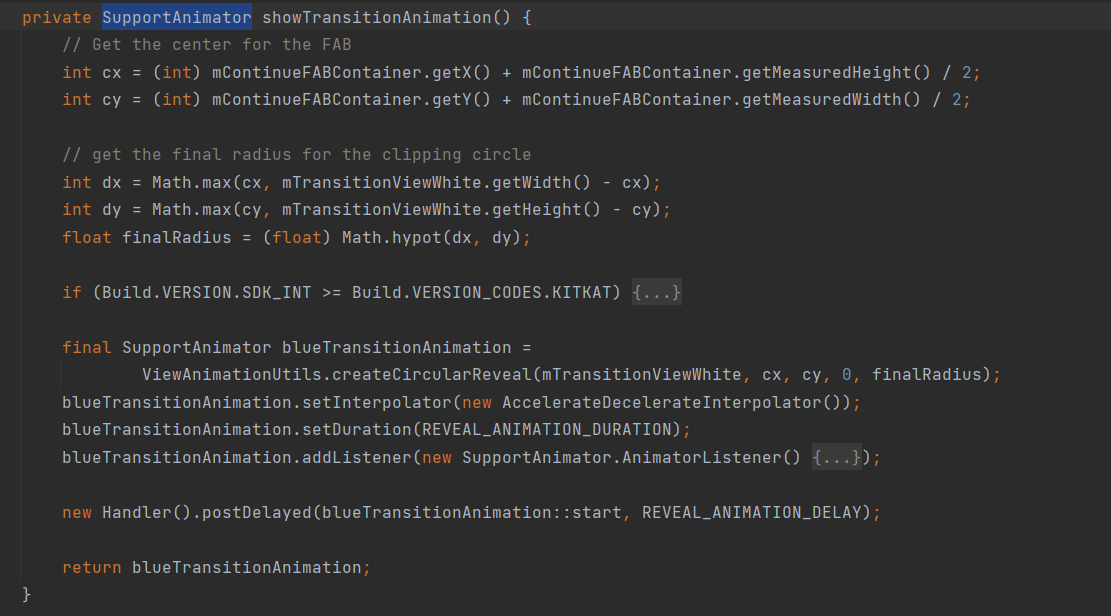
**功能场景位置：**setup场景内

**组件位置：**ConfirmSetupActivity

**涉及类：**ConfirmSetupActivity、SupportAnimator

**描述：**

工厂模式:类的创建依赖工厂类，定义一个创建对象的抽象方法，这样一旦需要增加新的功能，工厂类是ConfirmSetupActivity，用它创建SupportAnimator。



### 3.4备忘录模式

**功能场景位置：**setting场景内

**组件位置：**SettingsActivity

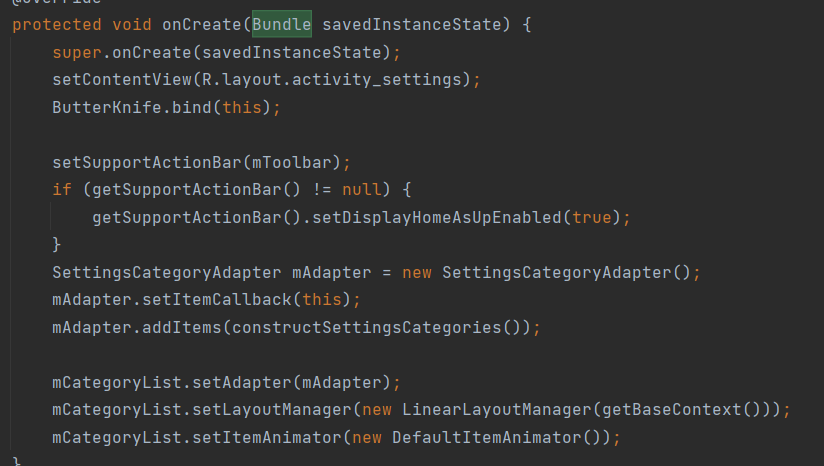
**涉及类：**SettingsActivity、Bundle、ThemeActivity

**描述：**

发起人角色：进入设置页面时候记录当前时刻的内部状态信息，提供创建备忘录和恢复备忘录数据的功能，实现其他业务功能，它可以访问备忘录里的所有信息。

备忘录角色：Bundle负责存储发起人的内部状态，在需要的时候提供这些内部状态给发起人。

管理者角色：SettingsActivity对备忘录进行管理，提供保存与获取备忘录的功能，但其不能对备忘录的内容进行访问与修改



### 3.5迭代器模式

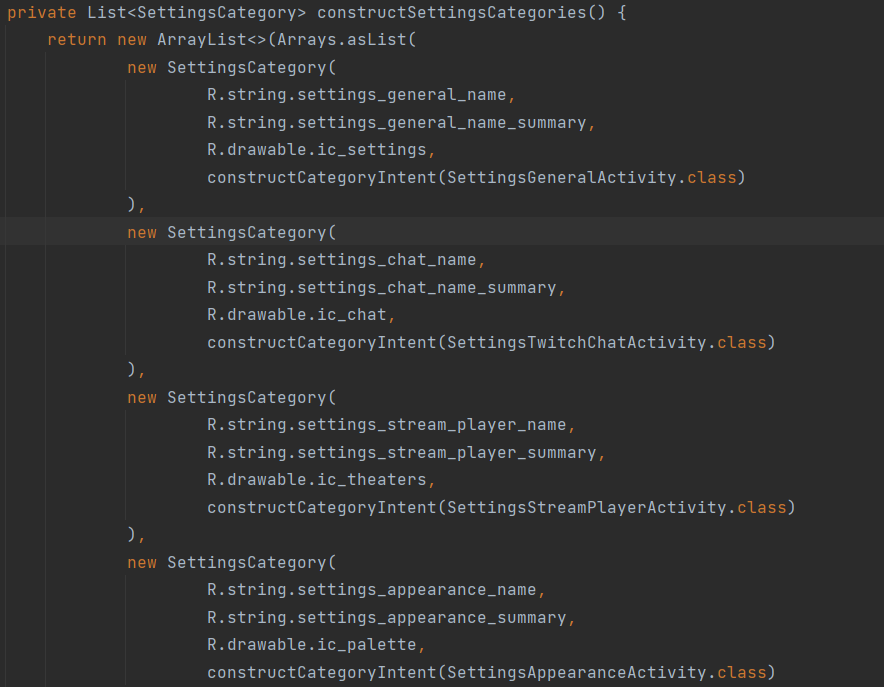
**功能场景位置：**setting场景内

**组件位置：**SettingsActivity

**涉及类：**SettingsActivity、SettingsCategory

**描述：**

SettingsActivity使用列表保存SettingsCategory的实例



### 3.6生成器模式

**功能场景位置：**setup场景内

**组件位置：**LoginActivity

**涉及类：**SupportAnimator、LoginActivity、SetupBaseActivity

**描述：**

生成器模式将一个对象分解为各个组件、将对象组件的构造封装起来、可以控制整个对象的生成过程。产品：具体生产器要构造的复杂对象AnimatorListener；

具体生产器：SupportAnimator实现SetupBaseActivity接口的类，具体生成器将实现Builder接口所定义的方法；指挥者：指挥者是LoginActivity类，该类需要含有SupportAnimator接口声明的变量。指挥者的职责是负责向用户提供具体生成器，即指挥者将请求具体生成器类来构造用户所需要的AnimatorListener对象，如果所请求的具体生成器成功地构造出AnimatorListener象，指挥者就可以让该具体生产器返回所构造的AnimatorListener对象。

### 3.7观察者模式

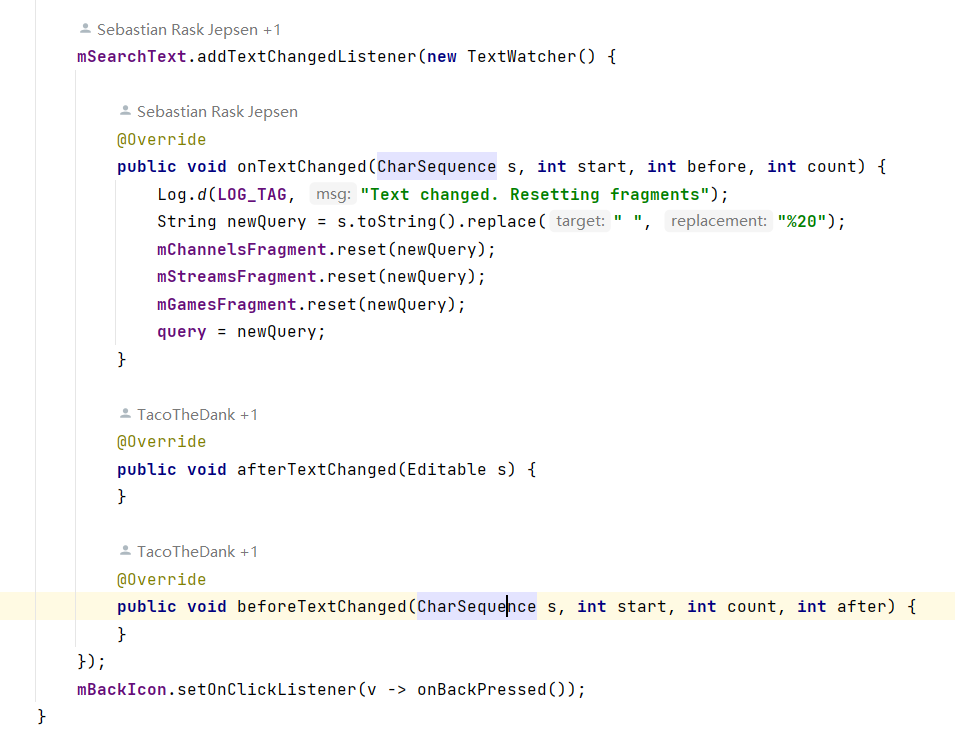
**功能场景位置：**search场景内

**组件位置：**SearchActivity

**涉及类：**SearchActivity、TextWatcher, EditText

**描述：**

TextWatcher接口的使用是一个观察者模式的示例。TextWatcher定义了一组回调方法，当EditText文本内容发生变化时，通知监听器执行相应的操作。其中，mSearchText是被观察者，这个匿名内部类是一个观察者。当搜索框中的文本发生变化时，被观察者mSearchText会触发TextWatcher接口中的onTextChanged()方法对搜索框进行改变。



## 其他详细设计

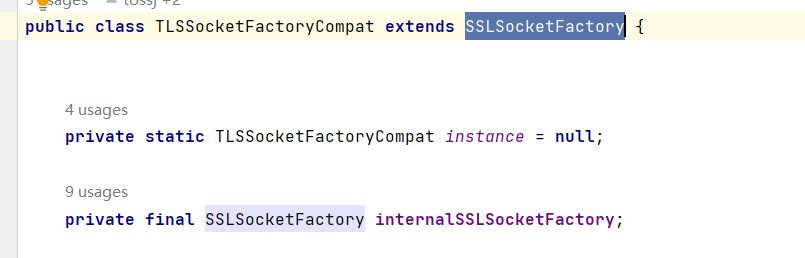
### 4.1单例模式

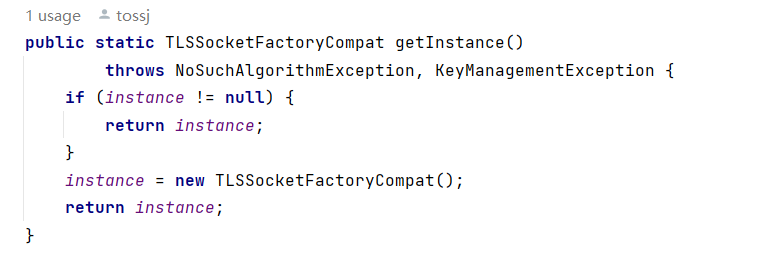
**位置：**TLSSocketFactoryCompat.java

**涉及类：**TLSSocketFactoryCompat, SSLSocketFactory

**描述（图像可选）：**

TLSSocketFactoryCompat出现了懒汉式单例模式的使用。TLSSocketFactoryCompat继承SSLSocketFactory类。TLSSocketFactoryCompat中，设立私有变量instance为null，提供getInstance保证这个变量只能从这个方法访问。在getInstance()方法中，通过检查instance变量是否为null，来确保只有一个TLSSocketFactoryCompat实例被创建并且在后续调用时被返回，这是典型的懒汉式单例模式。





### 4.2迭代器模式

**位置：**ChatEmoteManager.java

**涉及类：**ChatEmoteManager

**描述（图像可选）：**

ChatEmoteManager出现了迭代器模式的使用。loadCustomEmotes方法中通过获取迭代器对channelsets进行遍历访问与处理。通过hasNext查看set中是否还有下一个，通过next获取下一个元素，是经典的迭代器模式的使用。

