**中国矿业大学计算机学院**

**2017级本科生实验报告**

课程名称 软件工程实践

实验题目 Java语言与集成开发环境

学生姓名 郭子杨

学 号 08173042

专业班级 计科2017-6班

任课教师 刘迎春

目录

[1.开发环境基本情况 3](#_Toc27754805)

[1.1.JDK版本 3](#_Toc27754806)

[1.2.开发工具 3](#_Toc27754807)

[2.Java介绍与语言编程规范 3](#_Toc27754808)

[2.1. 简介 3](#_Toc27754809)

[2.2. 命名规范 3](#_Toc27754811)

[2.3. 注释规范 5](#_Toc27754812)

[2.4. 视觉组织 5](#_Toc27754813)

[2.5. 参考文档 7](#_Toc27754814)

[3.开发环境操作介绍 7](#_Toc27754815)

[3.1. IDEA面板 7](#_Toc27754816)

[3.2. 常用功能介绍 8](#_Toc27754817)

[3.3. File常用操作 9](#_Toc27754818)

[3.4. Navigate常用操作 9](#_Toc27754819)

[3.5. Run常用操作: 10](#_Toc27754820)

[3.6. Code常用操作 10](#_Toc27754821)

[3.7. Settings设置面板 10](#_Toc27754822)

[3.8. 右键类或者方法 12](#_Toc27754823)

[3.9. 快捷操作栏 12](#_Toc27754824)

[4.开源项目基本情况 12](#_Toc27754825)

[4.1.运行截图 12](#_Toc27754826)

[4.2.项目说明 14](#_Toc27754827)

[4.3.功能架构 14](#_Toc27754828)

[4.4.派生开源库 14](#_Toc27754829)

[4.5.第三方库 15](#_Toc27754830)

[5.开源项目需求分析 15](#_Toc27754831)

[5.1.数据层面 15](#_Toc27754832)

[5.2.界面层面 16](#_Toc27754833)

[6.开源项目软件结构 16](#_Toc27754834)

[6.1.软件类图 16](#_Toc27754835)

[6.2.部分实例的时序图 24](#_Toc27754836)

[6.3部分模块用例图 28](#_Toc27754837)

[7.开源项目源码分析 28](#_Toc27754838)

[7.1.项目中各类的功能 28](#_Toc27754839)

[7.2.项目中的重要函数 28](#_Toc27754840)

[7.2.1．Void Putparcelable（） 28](#_Toc27754841)

[7.2.2．Void getDateText（） 28](#_Toc27754842)

[7.2.3．Void MakeIntent（Data，Action） 29](#_Toc27754843)

[7.2.4．Void NewInstance（） 29](#_Toc27754844)

[7.2.5．Int Register(Account,Password1,Password2,Code) 29](#_Toc27754845)

# 1.开发环境基本情况

1.1.JDK版本：12.0.1

1.2.开发工具：IntelliJ IDEA Ultimate 2019.1

IntelliJ IDEA是JetBrains公司于2001年1月发布的Java语言开发集成环境。IntelliJ IDEA有着鲜明的自身特色:

智能代码，在代码选取时提高了代码的重构效率。

方便的版本恢复，仅通过IDEA就可以查看任何工程中文件的历史记录。

方便的版本控制，idea集成了目前常见的版本控制工具插件，包括git、svn、github让开发人员在编程的工程中直接在intellij idea里就能完成代码的提交、检出、解决冲突、查看版本控制服务器内容等等。

自动代码修复机制，当代码出现错误时，及时提醒程序员，程序员可利用相应按键自动修复异常代码。

# 2.Java介绍及语言编程规范

## 简介

## Java是由Sun Microsystems公司于1995年5月推出的Java面向对象程序设计语言和Java平台的总称。由James Gosling和同事们共同研发，并在1995年正式推出。

Java分为三个体系：

JavaSE(J2SE) (Java2 Platform Standard Edition，java平台标准版）

JavaEE(J2EE) (Java 2 Platform,Enterprise Edition，java平台企业版)

JavaME(J2ME) (Java 2 Platform Micro Edition，java平台微型版)。

2005年6月，JavaOne大会召开，SUN公司公开Java SE 6。此时，Java的各种版本已经更名以取消其中的数字"2"：J2EE更名为Java EE, J2SE更名为Java SE，J2ME更名为Java ME。

## 命名规范

* 1. 代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。 反例：\_name / \_\_name / $name / name\_ / name$ / name\_\_
  2. 代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。 说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式 也要避免采用。

正例：alibaba / taobao / youku / hangzhou 等国际通用的名称，可视同英文。

反例：DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3

* 1. 类名使用 UpperCamelCase 风格，但以下情形例外：DO / BO / DTO / VO / AO / PO / UID 等。

正例：MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion

反例：macroPolo / UserDo / XMLService / TCPUDPDeal / TAPromotion

* 1. 方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从驼峰形式。

正例： localValue / getHttpMessage() / inputUserId

* 1. 常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。

正例：MAX\_STOCK\_COUNT

反例：MAX\_COUNT

* 1. 抽象类命名使用 Abstract 或 Base 开头；异常类命名使用 Exception 结尾；测试类命名以它要测试的类的名称开始，以 Test 结尾。
  2. 类型与中括号紧挨相连来表示数组。

正例：定义整形数组 int[] arrayDemo;

反例：在 main 参数中，使用 String args[]来定义。

* 1. POJO 类中布尔类型的变量，都不要加 is 前缀，否则部分框架解析会引起序列化错误。

反例：定义为基本数据类型 Boolean isDeleted 的属性，它的方法也是 isDeleted()，RPC框架在反向解析的时候，“误以为”对应的属性名称是 deleted，导致属性获取不到，进而抛出异常。

* 1. 包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用单数形式，但是类名如果有复数含义，类名可以使用复数形式。

正例：应用工具类包名为 com.alibaba.ai.util、类名为 MessageUtils（此规则参考 spring 的框架结构）

* 1. 杜绝完全不规范的缩写，避免望文不知义。

反例：AbstractClass“缩写”命名成 AbsClass；condition“缩写”命名成 condi，此类随 意缩写严重降低了代码的可阅读性。

* 1. 为了达到代码自解释的目标，任何自定义编程元素在命名时，使用尽量完整的单词 组合来表达其意。

正例：在 JDK 中，表达原子更新的类名为：AtomicReferenceFieldUpdater。

反例：变量 int a 的随意命名方式。

* 1. 如果模块、接口、类、方法使用了设计模式，在命名时需体现出具体模式。 说明：将设计模式体现在名字中，有利于阅读者快速理解架构设计理念。

正例： public class OrderFactory;

public class LoginProxy;

public class ResourceObserver;

* 1. 接口类中的方法和属性不要加任何修饰符号（public 也不要加），保持代码的简洁 性，并加上有效的 Javadoc 注释。尽量不要在接口里定义变量，如果一定要定义变量，肯定是 与接口方法相关，并且是整个应用的基础常量。

正例：接口方法签名 void commit();

接口基础常量 String COMPANY = "alibaba";

反例：接口方法定义 public abstract void f();

说明：JDK8 中接口允许有默认实现，那么这个 default 方法，是对所有实现类都有价值的默 认实现。

* 1. 接口和实现类的命名有两套规则：

A）【强制】对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部 的实现类用 Impl 的后缀与接口区别。 正例：CacheServiceImpl 实现 CacheService 接口。

B）【推荐】如果是形容能力的接口名称，取对应的形容词为接口名（通常是–able 的形式）。 正例：AbstractTranslator 实现 Translatable 接口。

* 1. 枚举类名建议带上 Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。 说明：枚举其实就是特殊的类，域成员均为常量，且构造方法被默认强制是私有。 正例：枚举名字为 ProcessStatusEnum 的成员名称：SUCCESS / UNKNOWN\_REASON。
  2. 各层命名规约：

A) Service/DAO 层方法命名规约

a） 获取单个对象的方法用 get 做前缀。

b） 获取多个对象的方法用 list 做前缀，复数形式结尾如：listObjects。

c） 获取统计值的方法用 count 做前缀。

d） 插入的方法用 save/insert 做前缀。

e） 删除的方法用 remove/delete 做前缀。

f） 修改的方法用 update 做前缀。

B) 领域模型命名规约

a） 数据对象：xxxDO，xxx 即为数据表名。

b） 数据传输对象：xxxDTO，xxx 为业务领域相关的名称。

c） 展示对象：xxxVO，xxx 一般为网页名称。

d） POJO 是 DO/DTO/BO/VO 的统称，禁止命名成 xxxPOJO。

## 注释规范

1. 类、类属性、类方法的注释必须使用 Javadoc 规范，使用/\*\*内容\*/格式，不得使用 // xxx 方式。
2. 所有的抽象方法（包括接口中的方法）必须要用 Javadoc 注释、除了返回值、参数、 异常说明外，还必须指出该方法做什么事情，实现什么功能。
3. 所有的类都必须添加创建者和创建日期。
4. 方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释 使用/\* \*/注释，注意与代码对齐。
5. 所有的枚举类型字段必须要有注释，说明每个数据项的用途。
6. 与其“半吊子”英文来注释，不如用中文注释把问题说清楚。专有名词与关键字保持 英文原文即可。
7. 代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑 等的修改。

## 视觉组织

1. 大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行；如果 是非空代码块则：
   1. 左大括号前不换行。
   2. 左大括号后换行。
   3. 右大括号前换行。
   4. 右大括号后还有 else 等代码则不换行；表示终止的右大括号后必须换行。
2. 左小括号和字符之间不出现空格；同样，右小括号和字符之间也不出现空格；而左大 括号前需要空格。
3. if/for/while/switch/do 等保留字与括号之间都必须加空格。
4. 任何二目、三目运算符的左右两边都需要加一个空格。
5. 采用 4 个空格缩进，禁止使用 tab 字符。 说明：如果使用 tab 缩进，必须设置 1 个 tab 为 4 个空格。IDEA 设置 tab 为 4 个空格时， 请勿勾选 Use tab character；而在 eclipse 中，必须勾选 insert spaces for tabs。
6. 注释的双斜线与注释内容之间有且仅有一个空格。
7. 单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则：
   1. 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。
   2. 运算符与下文一起换行。
   3. 方法调用的点符号与下文一起换行。
   4. 方法调用中的多个参数需要换行时，在逗号后进行。
   5. 在括号前不要换行，见反例。
8. 方法参数在定义和传入时，多个参数逗号后边必须加空格。
9. IDE 的 text file encoding 设置为 UTF-8; IDE 中文件的换行符使用 Unix 格式， 不要使用 Windows 格式。
10. 单个方法的总行数不超过 80 行。
11. 没有必要增加若干空格来使某一行的字符与上一行对应位置的字符对齐。
12. 不同逻辑、不同语义、不同业务的代码之间插入一个空行分隔开来以提升可读性。 说明：任何情形，没有必要插入多个空行进行隔开。
13. 正例： （涉及 1-5 点）
14. **public** **static** **void** main(String[] args) {
15. // 缩进 4 个空格
16. String say = "hello";
17. // 运算符的左右必须有一个空格
18. **int** flag = 0;
19. // 关键词 if 与括号之间必须有一个空格，括号内的 f 与左括号，0 与右括号不需要空格
20. **if** (flag == 0) {
21. System.out.println(say);
22. }
23. // 左大括号前加空格且不换行；左大括号后换行
24. **if** (flag == 1) {
25. System.out.println("world");
26. // 右大括号前换行，右大括号后有 else，不用换行
27. } **else** {
28. System.out.println("ok");
29. // 在右大括号后直接结束，则必须换行
30. }
31. }

## 参考文档

本部分代码规范说明来自《阿里巴巴Java开发手册》。

来源链接：https://github.com/chjw8016/alibaba-java-style-guide

# 3.开发环境操作介绍

## IDEA面板

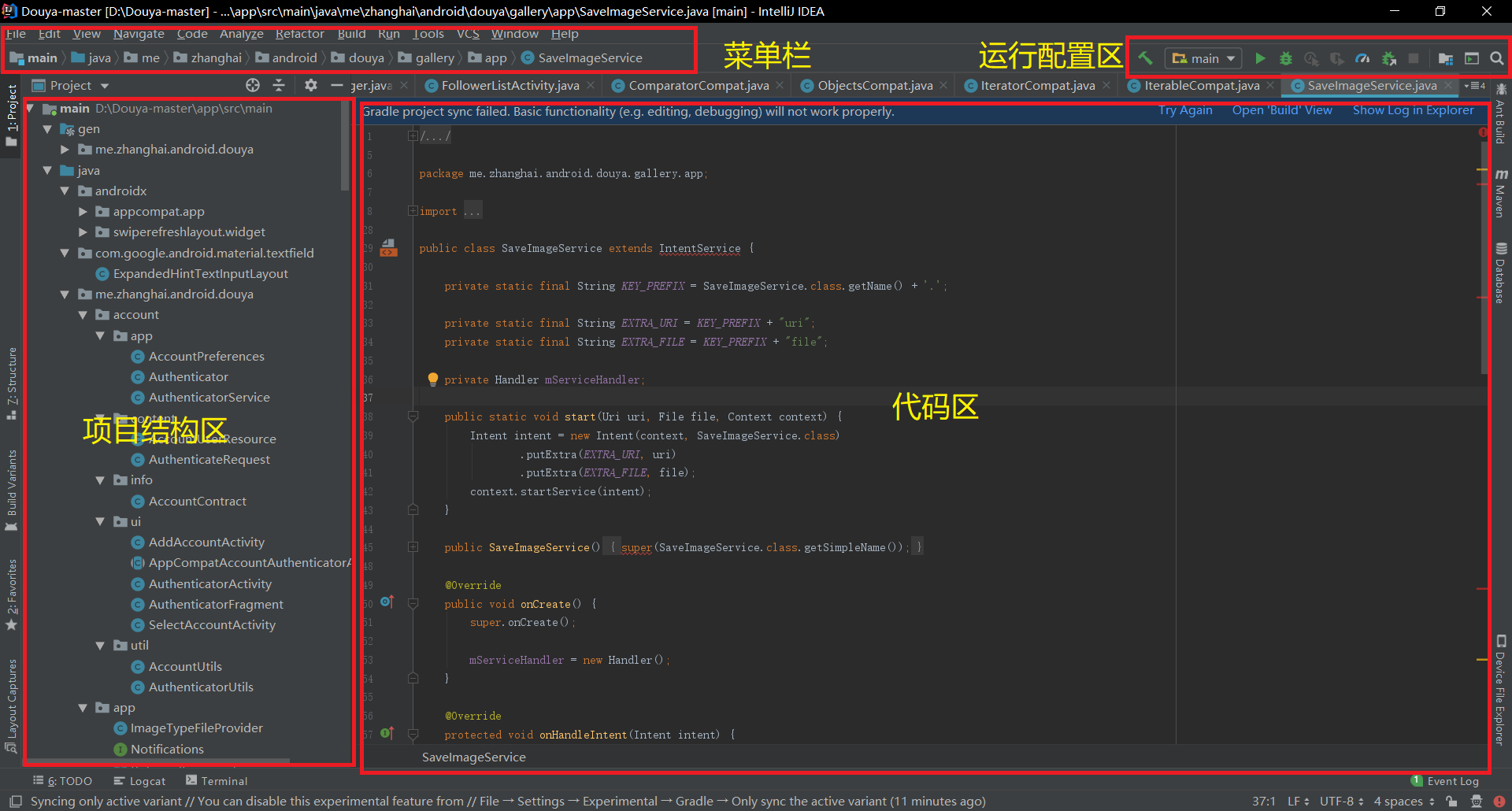


图3-1 IntelliJ IDEA面板

## 常用功能介绍：

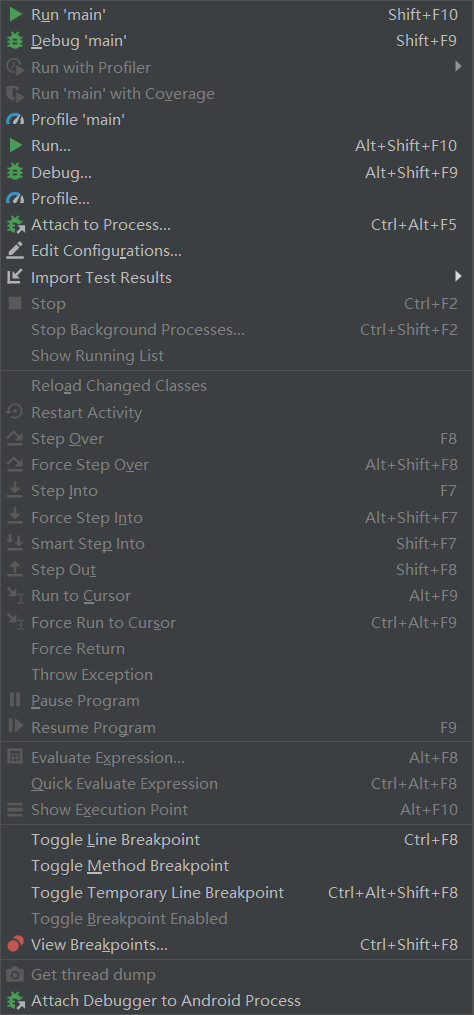
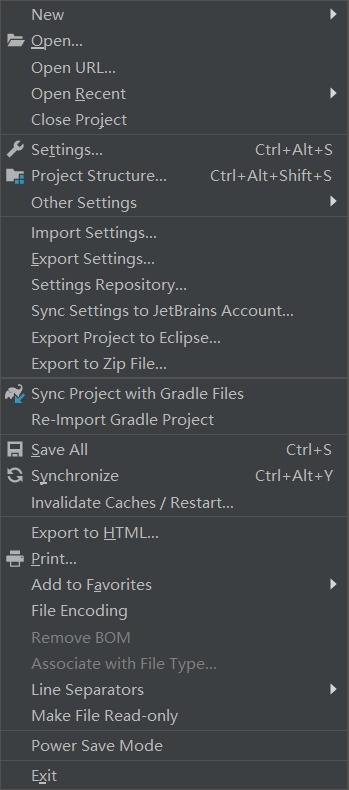


图3-2 File选项 图3-3 Run选项

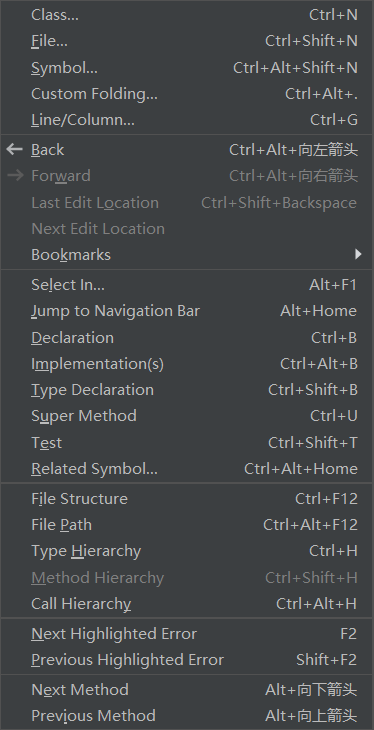
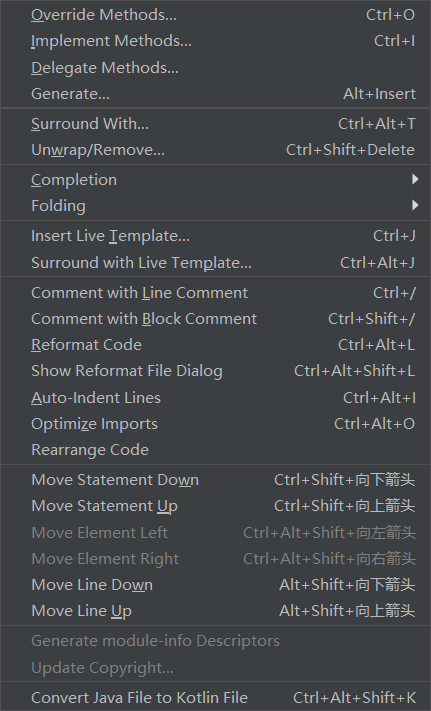


图3-4 Navigate选项 图3-5 Code选项

## File常用操作：

新建project，导入已存在的资源project，从版本控制库导入工程，新建Module，导入已存在的资源Module

Open：打开本地的文件或工程

Open Recent：打开最近已导入过的工程

Close Project：关闭工程

Setting：IDEA配置文件

Project Structure：显示当前工程结构

Import Settings：导入Settins文件

Export Settings：导出Settings文件

将自己的settings文件导出到本地或云盘，下次使用时可以直接导入使用

## Navigate常用操作：

Class：查询类

File：查询文件

Jump to Navigation Bar：跳到导航栏

Declaration：进入光标所在的方法/变量的接口或是定义处

Implementations：方法的实现

Type Declaration：进入光标当前所在属性的类

Type Hierarchy：当前类的分层结构

## Run常用操作:

Run:运行项目或文件

Debug：调试文件

## Code常用操作：

Override Methods:重写方法

Implement Methods:方法继承

## Settings设置面板：

IntellIJ IDEA配置Maven以及修改默认Repository：

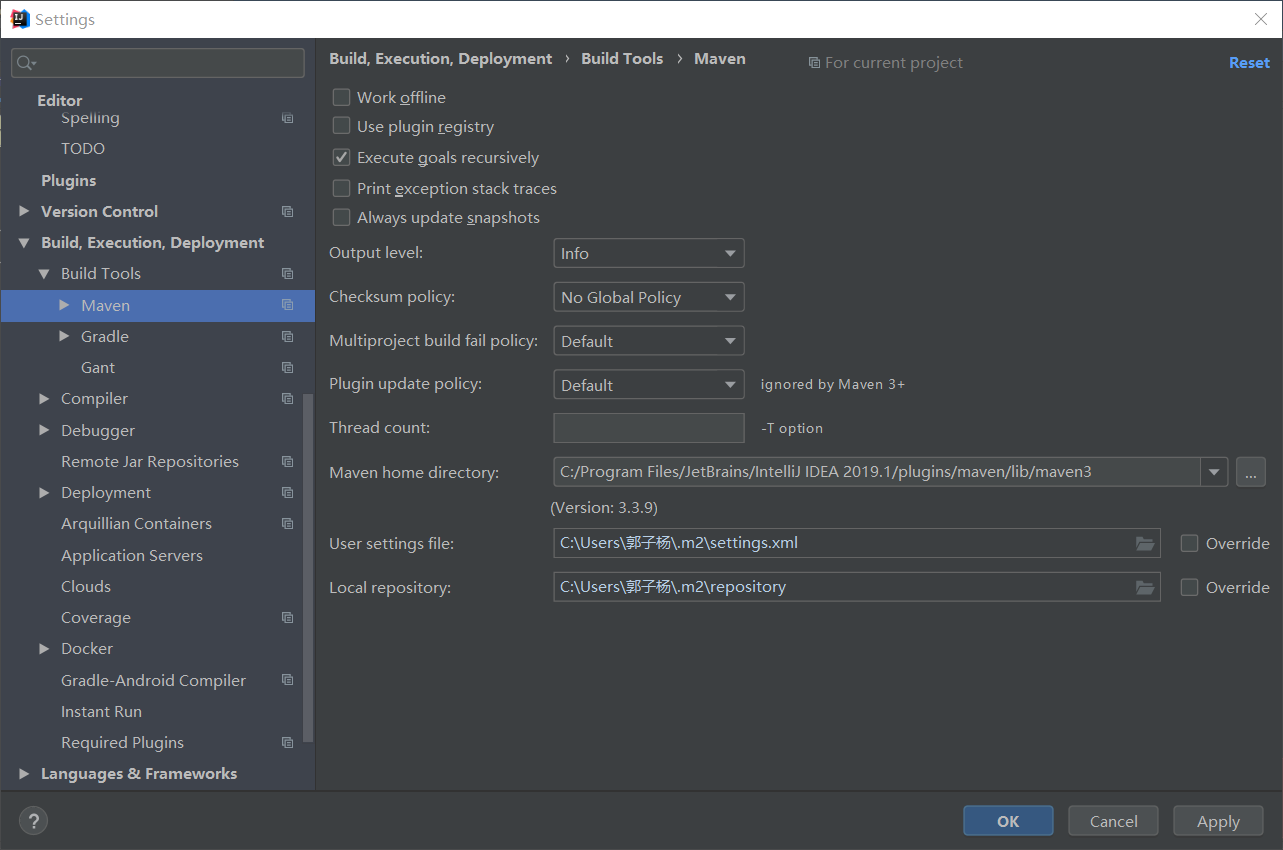


图3-6 Maven设置

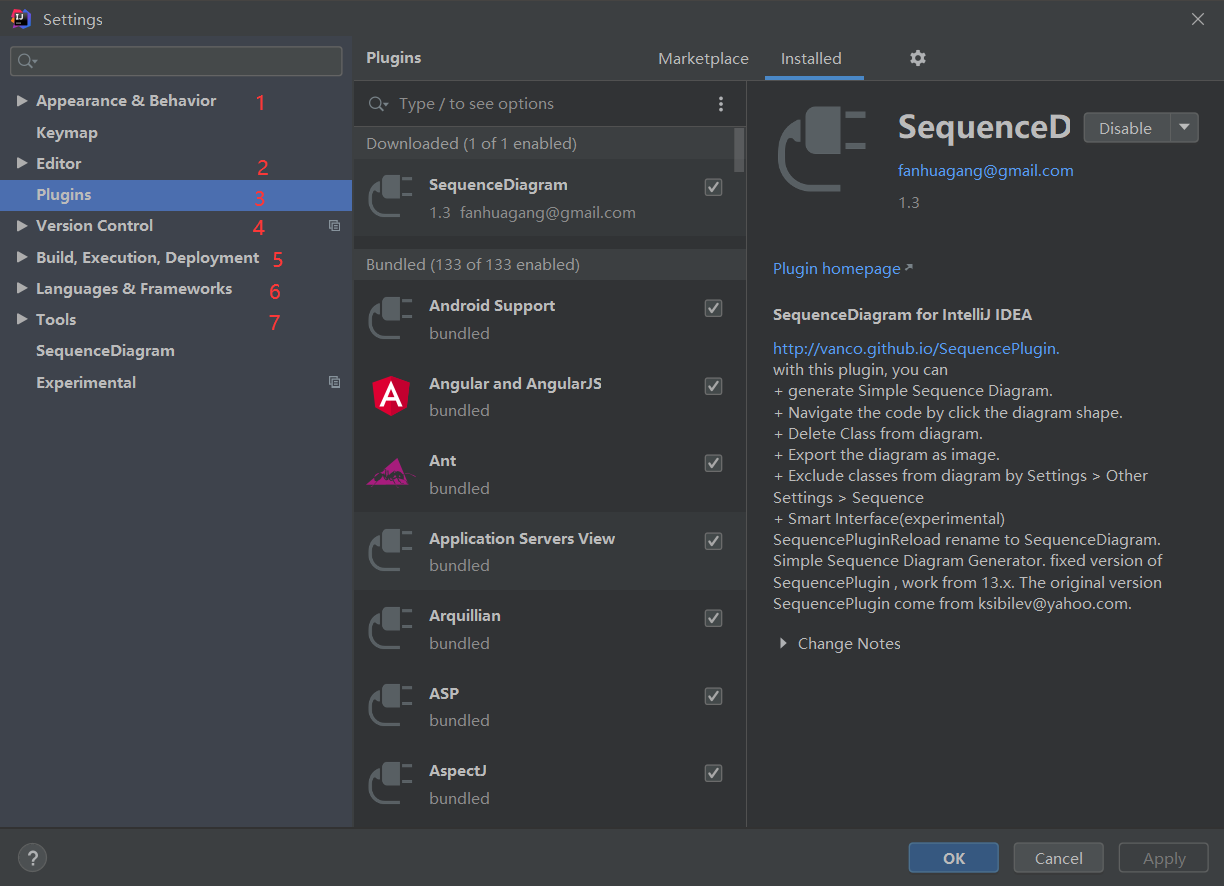
插件管理和下载：

图3-7插件下载

1. 包含Idea的外观设置以及页面按钮的摆放和组合
2. 包含Idea中字体设置以及代码编辑器的风格设置
3. 包含Idea第三方插件的安装与卸载
4. 包含Idea利用其中的git插件与github建立连接或与服务器端建立连接。
5. 包含idea中代码Bulid、编译、运行、debuger、探查器工具。
6. 包含多种可用框架设置以及语言设置，如:JavaScript、Kotlin、OsGi、Spring、TypeScript.
7. 包含多个使用工具，如数据库操作工具、连接服务器端的SSH工具。

## 右键类或者方法：

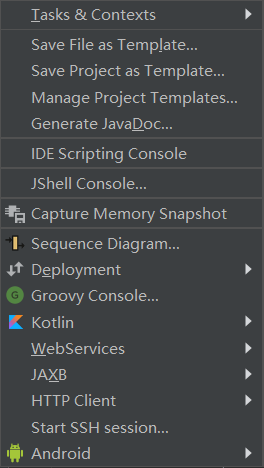
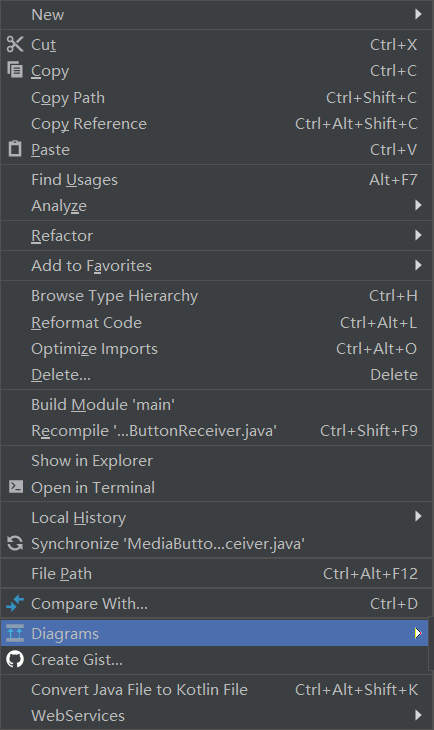
 

图 3-8 右键菜单

右键点击类可以运行和调试(当类里面有Main()函数时)，点击Diagrams时可以查看类图。右键点击方法，可以查看使用位置以及实现重构等等，点击Sequence Diagram可以查看时序图。

## 快捷操作栏：



图 3-9快捷操作栏

快捷操作栏可以快速实现运行、调试、停止等功能。

# 4.开源项目基本情况

开源项目：开源的 Material Design 豆瓣客户端

来自于：https://github.com/zhanghai/Douya

## 4.1.运行截图：

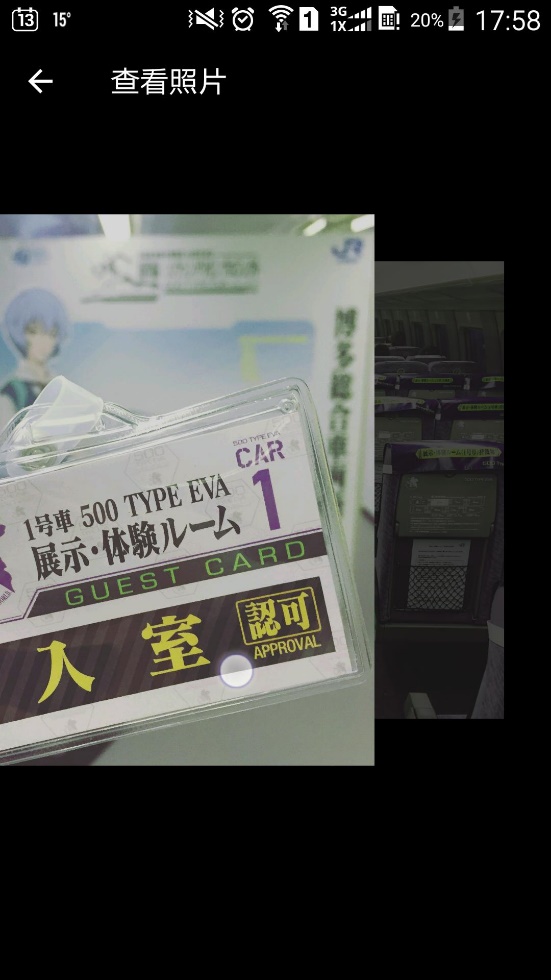


图4-主界面截图 图4-2画廊界面截图



图4-3动态界面截图

## 4.2.项目说明

这是一个模仿豆瓣而制作的项目——豆芽。豆芽的最终目标是为豆瓣中基于友邻的信息获取方式提供在移动端的便利。为了优雅地实现这个目标，豆芽主要遵循以下的原则：

* 遵循 Material Design 规范，且指导思想优先于细节规定。
* 像素完美，但更注重以人为本。
* 实现精确，代码可以自我辩护。
* 行为合格，支持屏幕旋转和平板布局。
* 功能崇尚简约，不打扰用户。
* 行为默认值合理，且用户可调节。
* 积极表现豆瓣特性，如广播、友邻、豆邮等。
* 通过细节设计，提倡用心、考虑到他人的内容。

## 4.3.功能架构

导航采用抽屉一级导航 + 选项卡二级导航的方式。工具栏上将显示全局的动作。

* 工具栏
  + 提醒：所有类别的提醒，可以查看历史提醒
  + 豆邮：用户间的邮件往来，希望鼓励郑重而非聊天。
  + 搜索：立即访问想要的内容。
* 用户：点击后显示个人页面，相当于“我的豆瓣”。
* 首页
  + 友邻广播：友邻互动、友邻推荐、系统定制的推荐。
  + 九点：友邻的日记、博客文章等，有深度的内容。
  + 一刻：全站范围的热门内容推荐。
  + 同城：基于地理位置的内容。
  + 线上活动：基于共同兴趣的内容。
* 读书
  + 分类浏览、首页推荐：入口，以及最有可能发现新内容的地方。
  + 我读：管理自己的读书标记、创造内容。
  + 动态：查看友邻的阅读动态，互动、获得推荐。
  + 豆瓣猜：基于算法的推荐。
* 电影
  + 类似读书。
* 音乐
  + 类似读书。
* 设置：提供应用设置等。

## 4.4.派生开源库

为此项目诞生的五个开源库：

* [MaterialColdStart](https://github.com/zhanghai/MaterialColdStart)，800+ Stars
* [MaterialProgressBar](https://github.com/zhanghai/MaterialProgressBar)，500+ Stars
* [CustomTabsHelper](https://github.com/zhanghai/CustomTabsHelper)，200+ Stars
* [MaterialEditText](https://github.com/zhanghai/MaterialEditText)
* [SystemUiHelper](https://github.com/zhanghai/SystemUiHelper)

## 4.5.第三方库

* [PhotoView](https://github.com/chrisbanes/PhotoView)
* [Glide](https://github.com/bumptech/glide)
* [Gson](https://github.com/google/gson)
* [ButterKnife](https://github.com/JakeWharton/butterknife)
* [DiskLruCache](https://github.com/JakeWharton/DiskLruCache/)
* [ThreeTenABP](https://github.com/JakeWharton/ThreeTenABP/)
* [Retrofit](https://github.com/square/retrofit)
* [EventBus](https://github.com/greenrobot/EventBus)
* [CustomTabsHelper](https://github.com/zhanghai/CustomTabsHelper)
* [EffortlessPermissions](https://github.com/zhanghai/EffortlessPermissions)
* [MaterialEditText](https://github.com/zhanghai/MaterialEditText)
* [MaterialProgressBar](https://github.com/zhanghai/MaterialProgressBar)
* [SystemUiHelper](https://github.com/zhanghai/SystemUiHelper)
* [MaterialColdStart](https://github.com/zhanghai/MaterialColdStart)

# 5.开源项目需求分析

豆瓣从来不是一个单一的网站，而对于豆瓣的用法自然不尽相同。使用豆瓣是为了获取信息，但信息的获取是基于条目和算法，还是基于友邻和人，这个问题在豆瓣的多次改版中长时间悬而未决。在最近的几次更新中，豆瓣的应用选择更改为基于条目推荐而一个重度豆瓣用户，重视的却恰好是基于人的推荐，喜欢的是友邻间的鸡犬相闻，以及闲逛时从条目评论、广播、日记中发现新友邻的惊喜。正如某次“还我旧版”运动中听到的声音，“不管怎么改版，只要友邻们还在就好”。借助豆瓣提供的开放 API，因而产生了改项目。

## 5.1.数据层面

应用除了对少数内容进行缓存，其他内容均直接从网络获取。

* 基于 Android 账户系统提供用户账户和身份认证。
* 使用 Retrofit 及部分自定义增强处理网络请求。
* 使用 Gson 自动填充数据模型。
* 使用 Glide 加载图片。
* 使用 DiskLRUCache 及自定义增强对首页数据进行缓存。
* 使用 EventBus 同步不同页面间对象状态。

## 5.2.界面层面

使用 Support Library 中的 AppCompat、Design、CardView、RecyclerView 进行 Material Design 实现，在必要时引入/自己写作第三方库以实现部分界面元素和效果。

使用框架的 Shared Element Transition 实现在 Android 5.0 以上的界面过渡动画。

界面实现一般分为 Activity、Fragment、Adapter 三个模块，分别负责作为容器，发起请求、展示数据和用户交互，以及数据/交互绑定。

# 6.开源项目软件结构

## 6.1.软件类图：

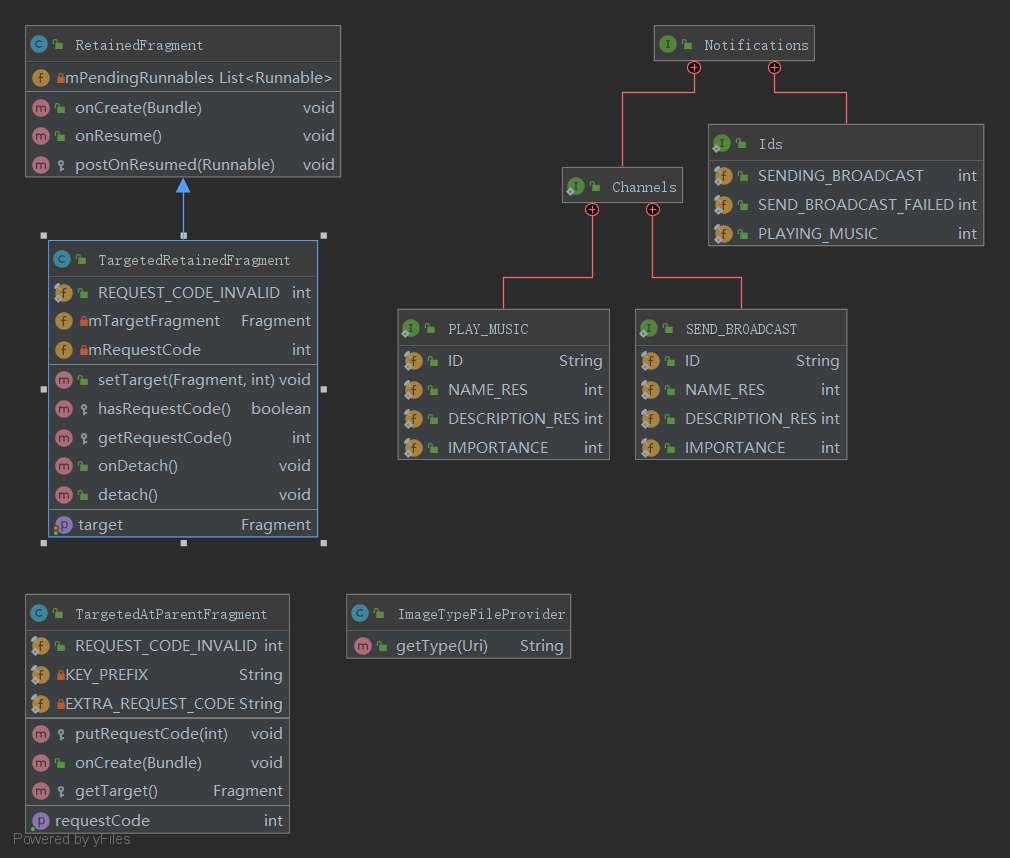
Package app：

图 6-1 项目软件结构（无依赖下）

Package content：

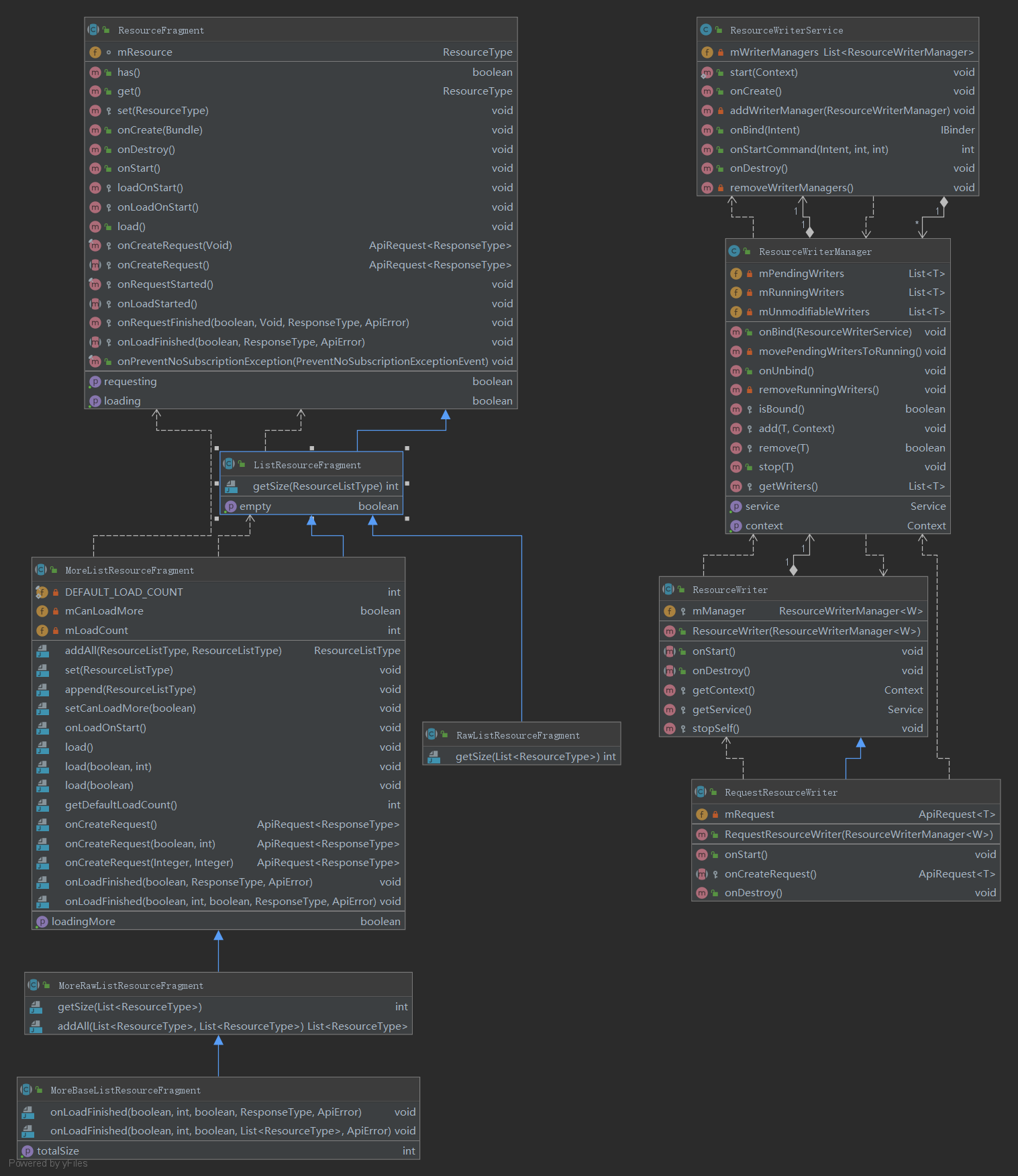


图 6-2 Package content下的类关系图

Package functional：

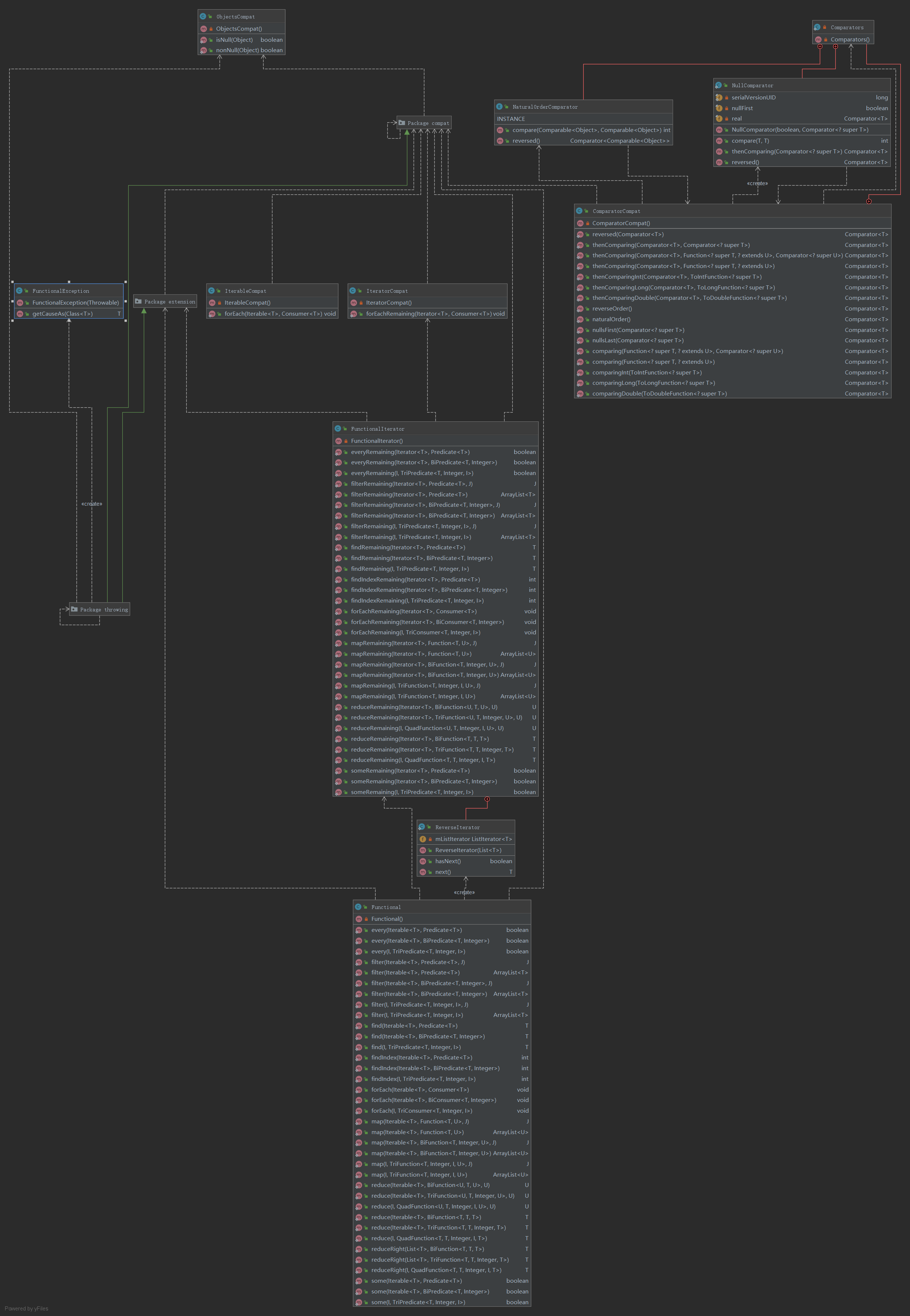


图6-3 Package functional下的类图

Package glide：

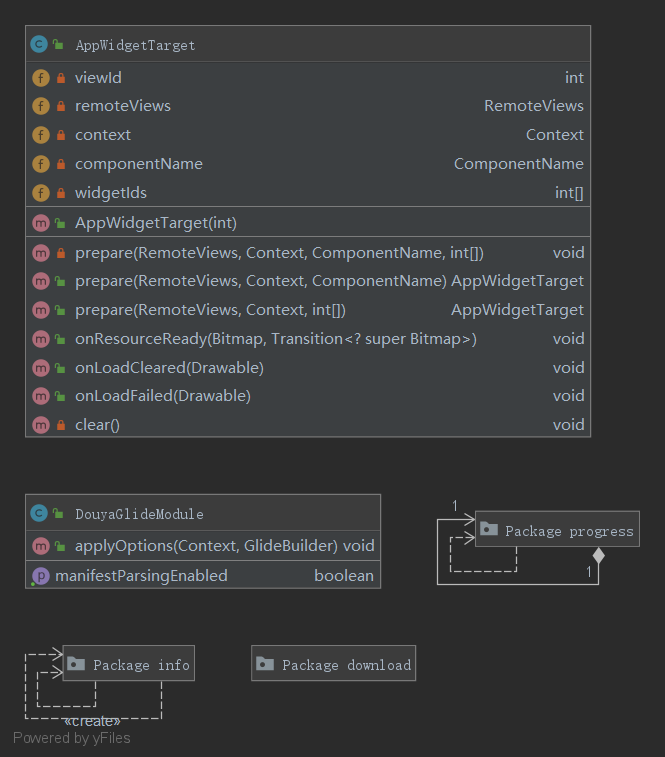


图 6-4 Package glide下的类关系图

Package link：

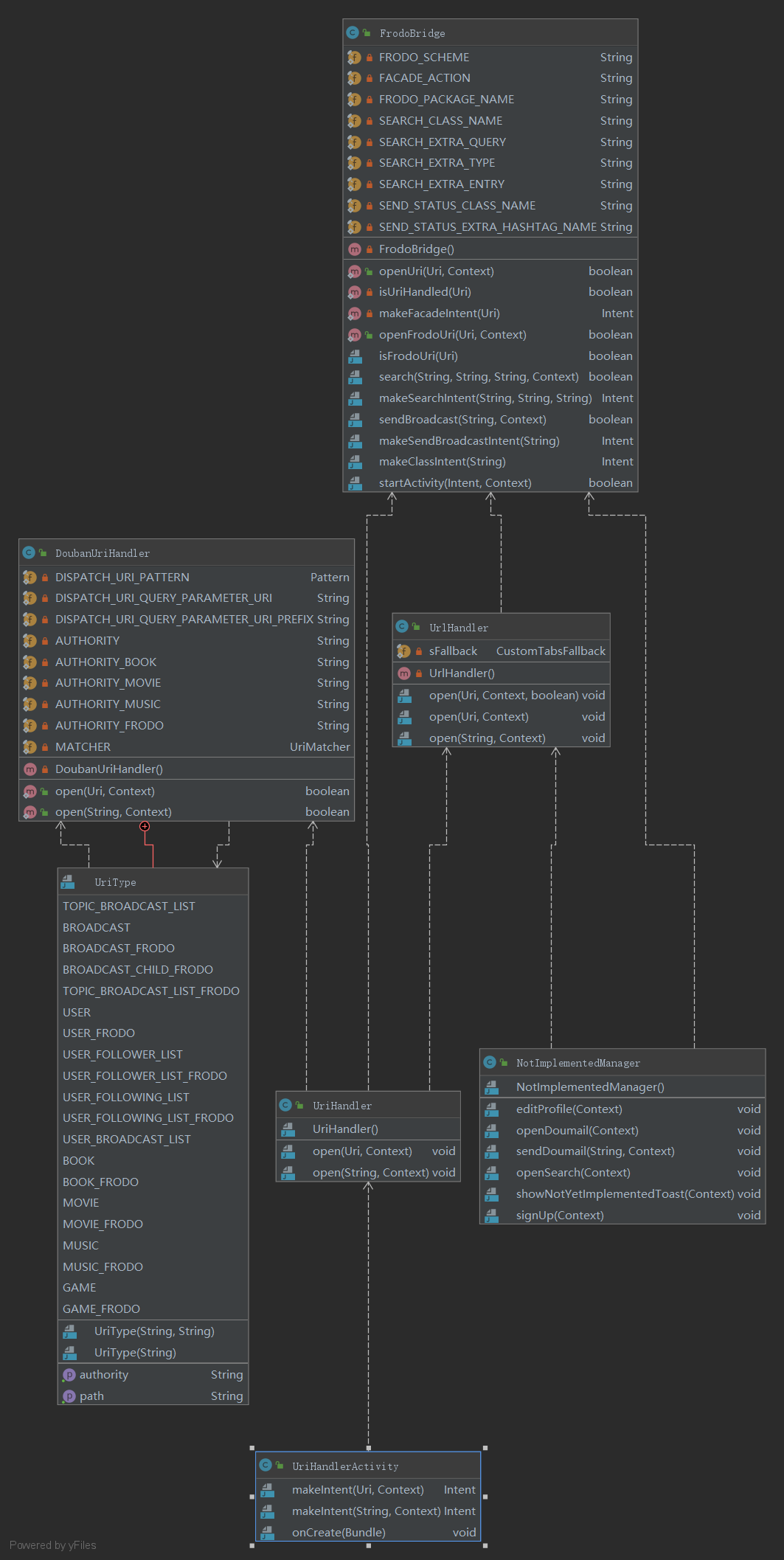


图 6-5 Package link下的类关系图

Package media：

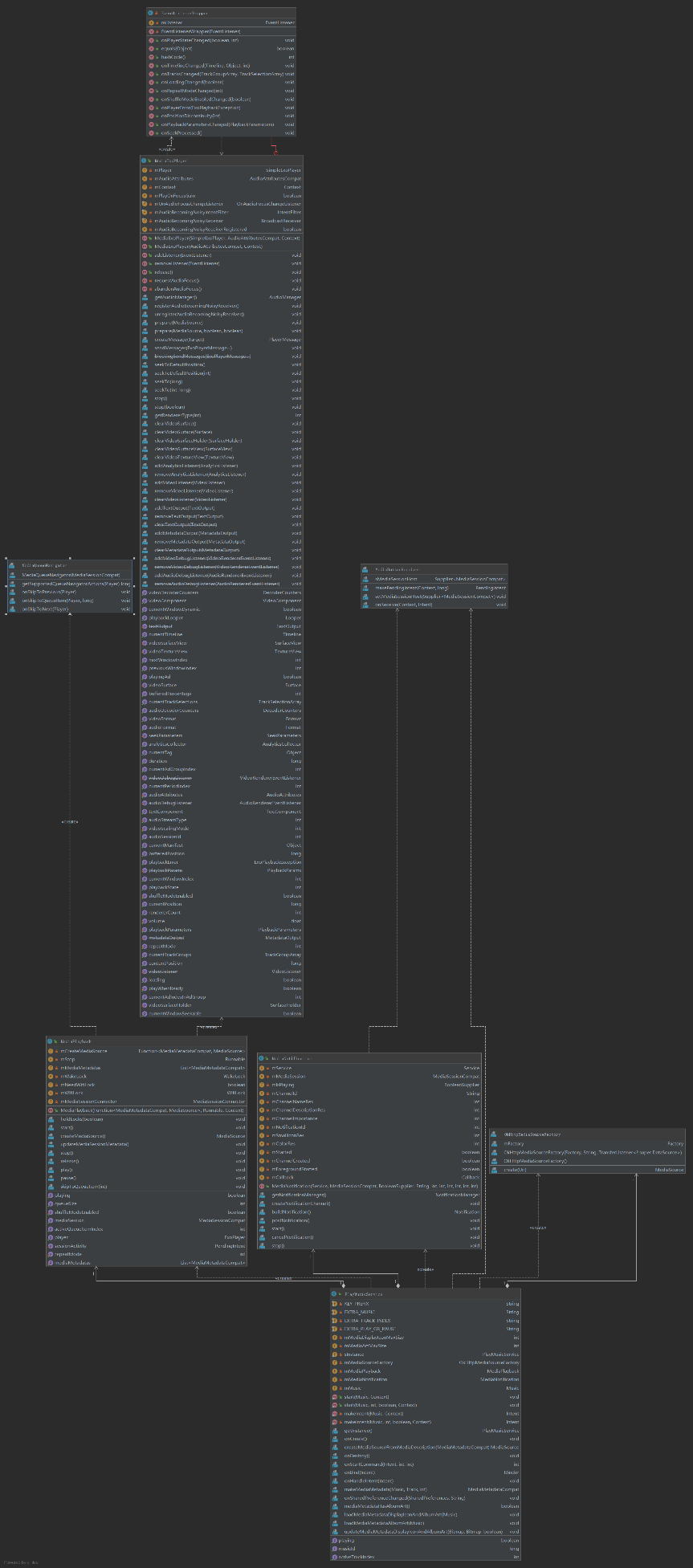


图 6-6 Package media下的类关系图

Package network：

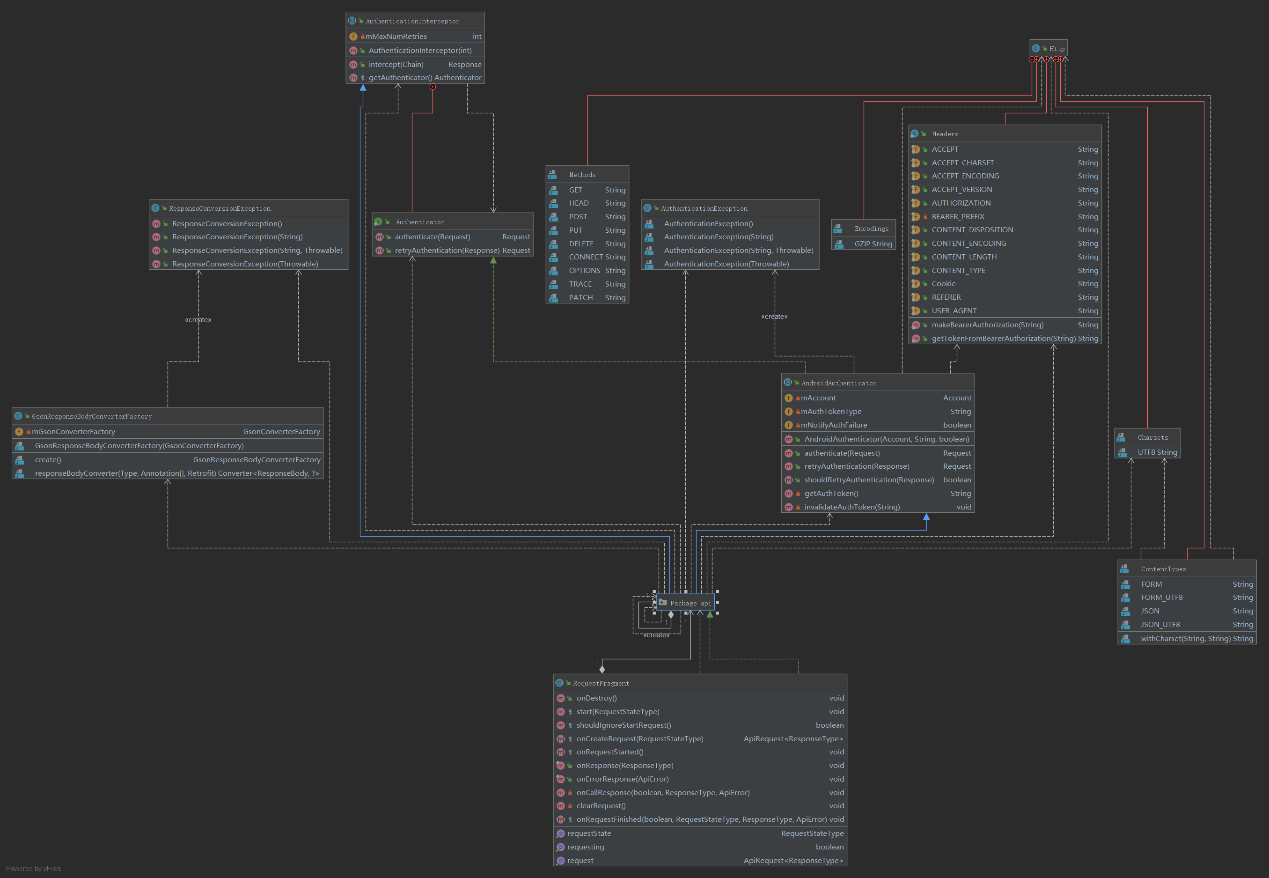


图 6-7 Package network下的类关系图

Package reflected：

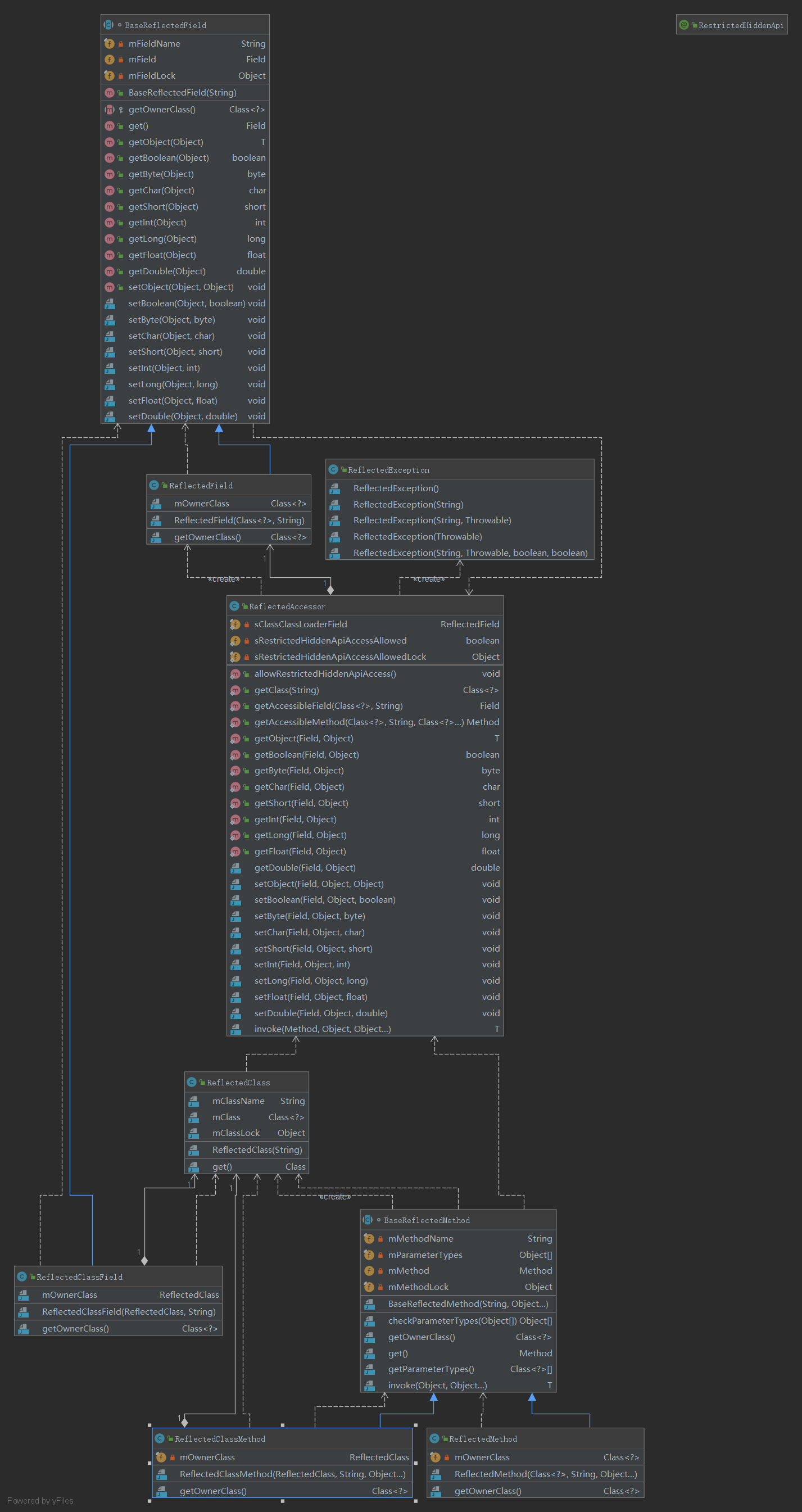


图 6-8 Package reflected下的类关系图

## 6.2.部分实例的时序图如下：

RebrodcastList中的时序图：

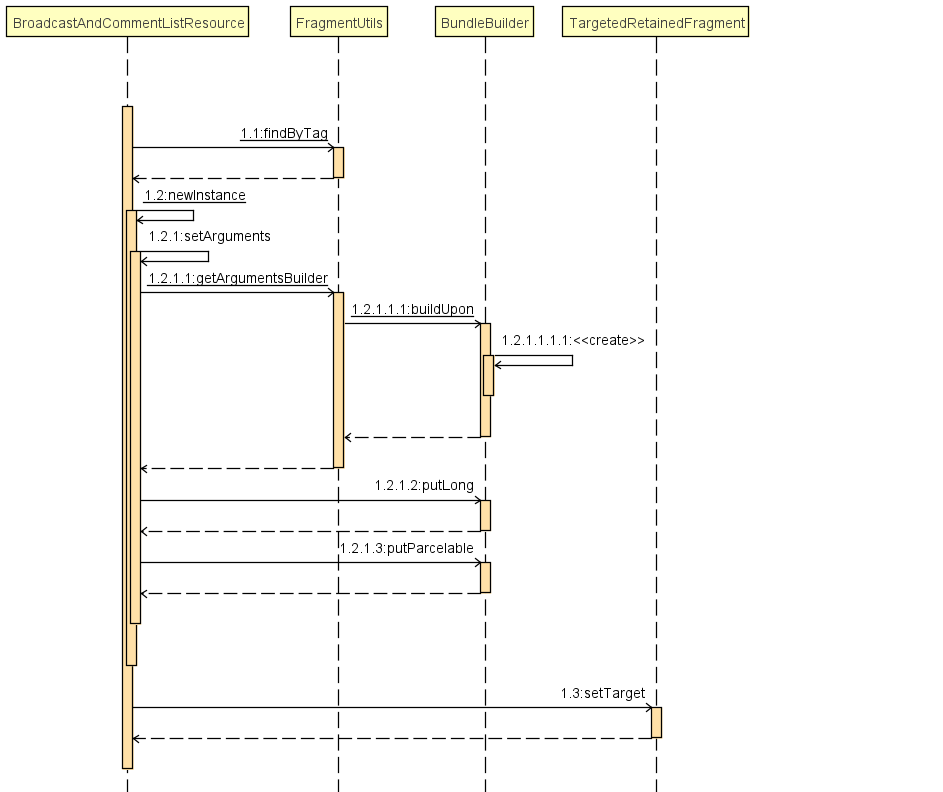


图6-9 RebrodcastList

CalenderAppWidgetProvider中的时序图：

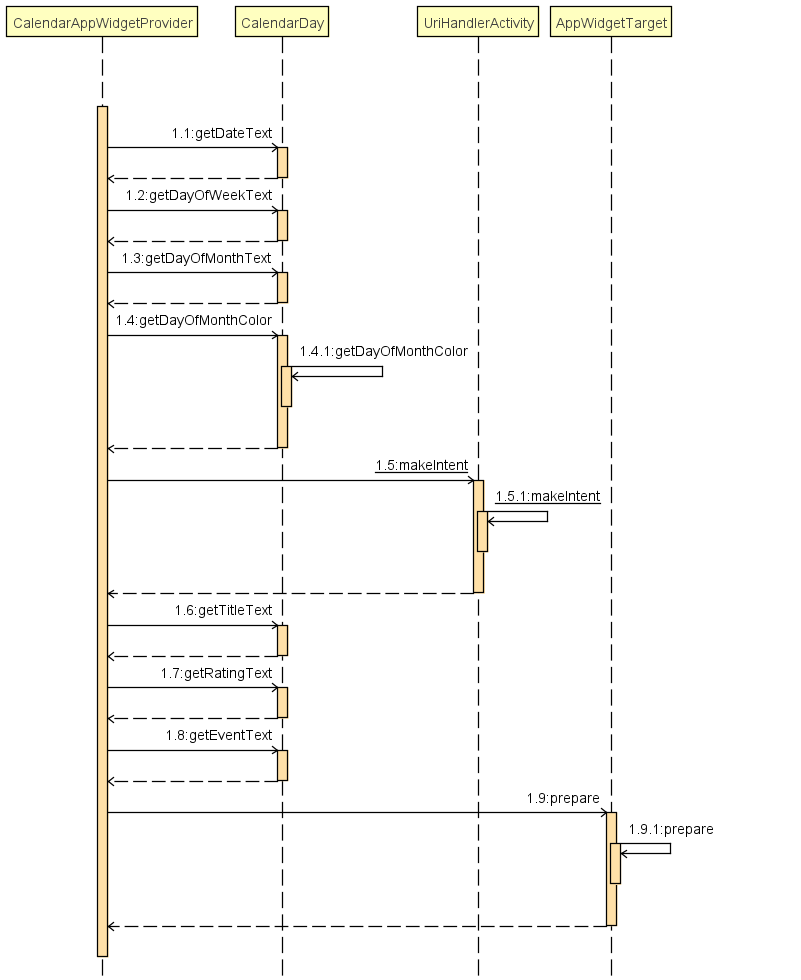


图6-10 CalenderAppWidgetProvider

FollowerList中的时序图：

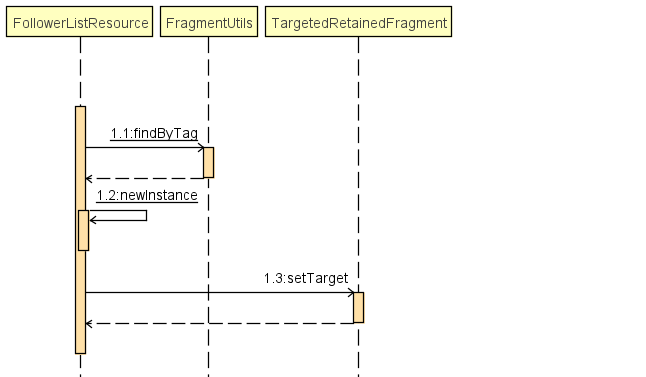


图6-11 FollowerList

FolloweUserManager中的时序图：

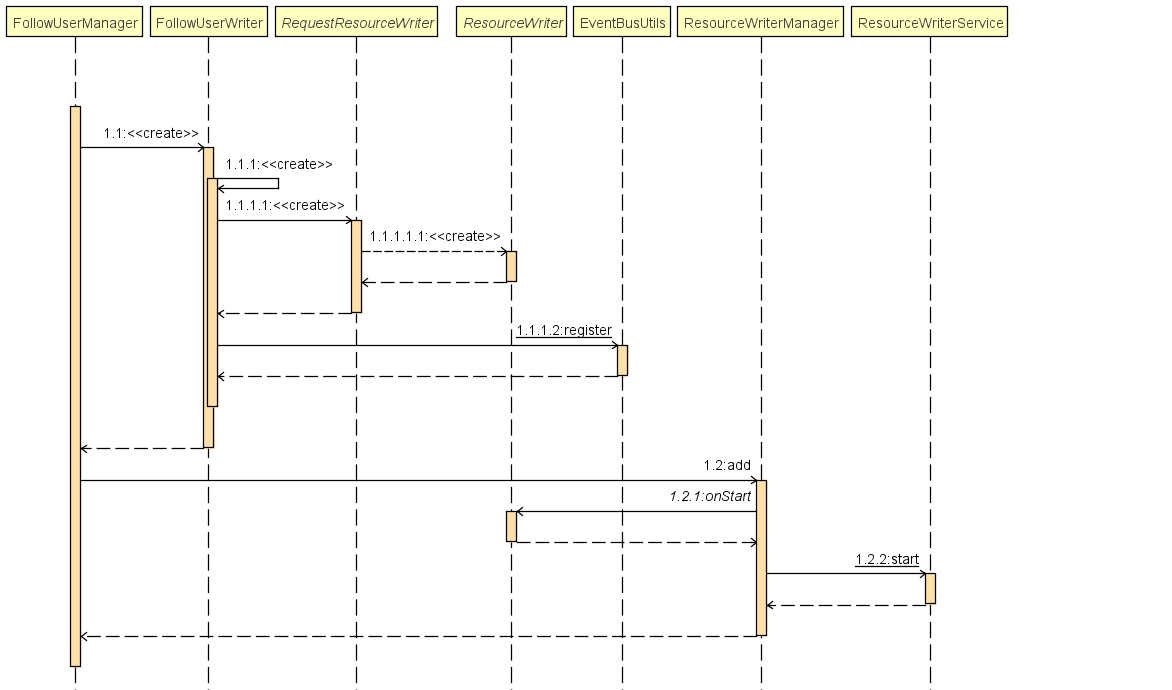


图6-12 FolloweUserManager

SaveImageService中的时序图：

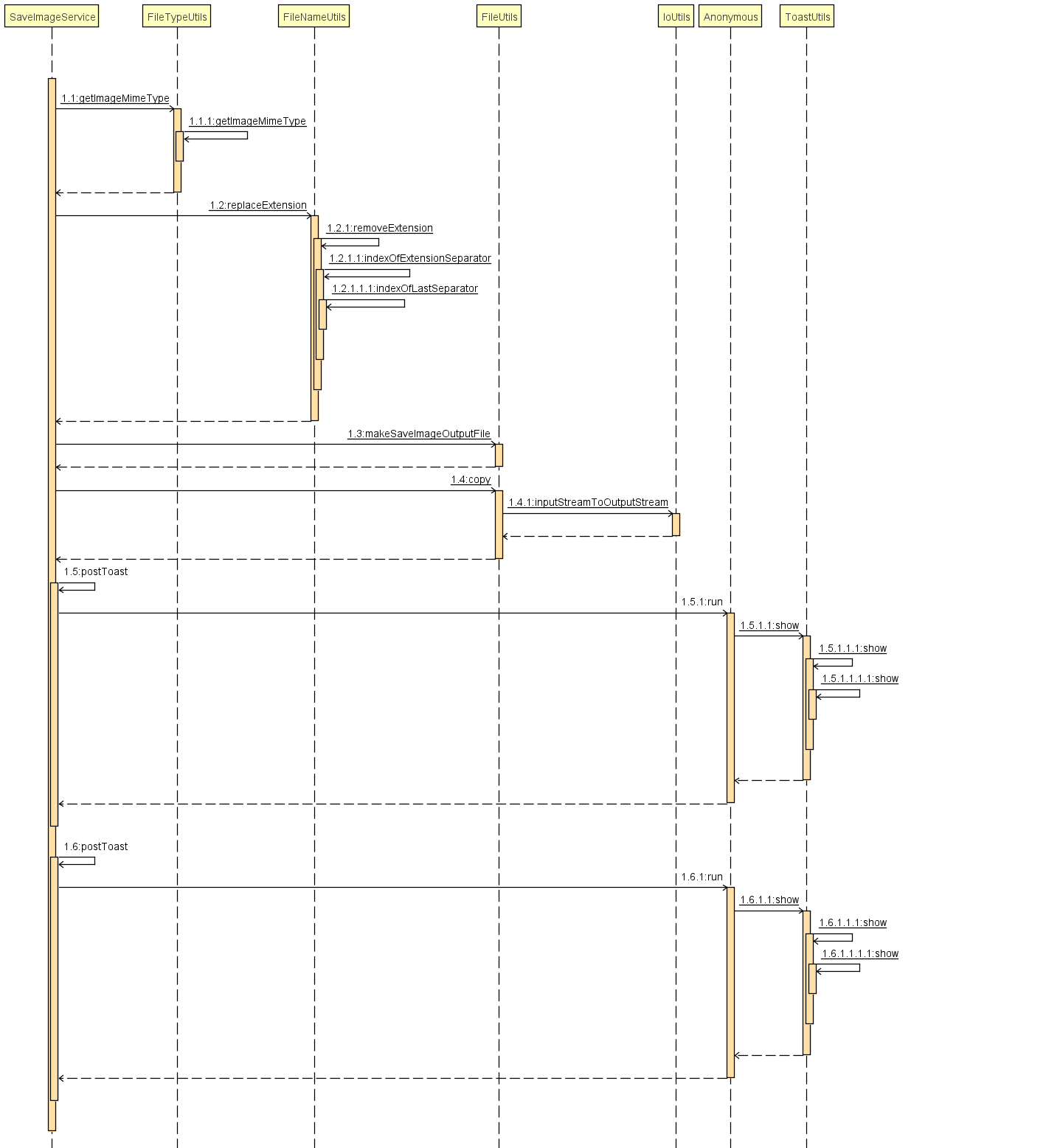


图6-13 SaveImageService

## 6.3部分模块用例图

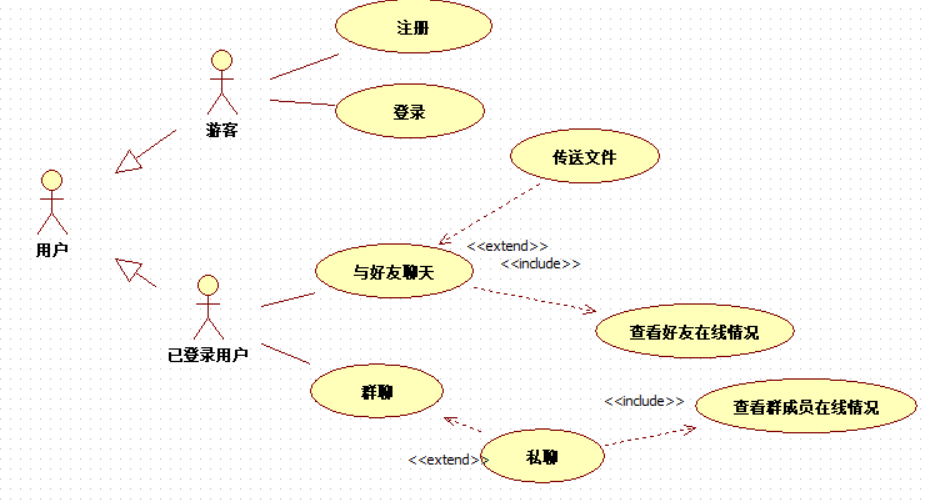


图6-14用户聊天用例图

# 7.开源项目源码分析

## 7.1.项目中各类的功能

类BundleBuilder 负责保存与回调数据

类FragmentUtils 负责生成APP主界面

类BroadcastAndCommentListResource 负责生成今日广播内容

类TargetedRetainedFragment 负责确认发送目标

类CalendarAPPWidgetProvider 负责生成日历内容

类CalendarDay 负责生成今日推送

类UriHandlerActivity 负责注册验证服务

类AppWidgetTarget 负责桌面小部件的生成

类FollowerListResource 负责生成关注者的信息

类RequestResourceWriter 负责请求作者信息

类SaveImageService 负责保存图片

## 项目中的重要函数

### 7.2.1．Void Putparcelable（）

Android中实现序列化有两个选择：一是实现Serializable接口（是JavaSE本身就支持的），一是实现Parcelable接口（是Android特有功能，效率比实现Serializable接口高效，可用于Intent数据传递，也可以用于进程间通信（IPC））。实现Serializable接口非常简单，声明一下就可以了，而实现Parcelable接口稍微复杂一些，但效率更高，推荐用这种方法提高性能。需要在多个部件(Activity或Service)之间通过Intent传递一些数据，简单类型（如：数字、字符串）的可以直接放入Intent。复杂类型必须实现Parcelable接口。

### 7.2.2．Void getDateText（）

获取数据文本，包括日期，颜色，标题，时间，次数，事件信息等

### 7.2.3．Void MakeIntent（Data，Action）

这个命令可以在终端构造特定格式的Intent并且发送这个广播，这个命令的参数很丰富，可以满足大部分Intent构造需求。Intent是一个任务或功能的抽象，它会被APP在未来某个时刻执行。它告诉APP须要做的事情。Actions:也就是Intent须要做的事情。在软件中用于打开一个URL，编辑某些数据。一个Action是一个简单字符串常量描写叙述应该完毕的事情。Data:即Intent需用使用的资源。在Android中它会用一个URI(唯一资源标识符)或者Uri对象来表示。数据类型须要依据Action而变。将 Action和Data组合在一起。Android 就能够知道这个Intent是干什么的以及用什么来干的问题。

### 7.2.4.Void NewInstance（）

在开发中，我们经常可以看到一些接口的设计，实现接口的类加载一般都会采用newInstance。先调用Class加载方法加载某个类，然后实例化。可以在调用class的静态加载方法forName时获得更好的灵活性，提供给了一种降耦的手段。

### 7.2.5.Int register(Account,Password1,Password2,Code)

判断账号:定义一个函数，用于用户注册时检查用户名是否被占用,此时不需要对密码进行判断,可以打开存储文件将账号全部提取,用于注册时判断,下面程序选择用字典储存.判断密码:此时输入注册密码,然后对密码是否和法进行判断,然后再次输入密码判断;两次输入的密码是否相同,然后定义函数对注册无误的账号密码写入到存储文件中保存.登录验证:登录前输入账号和密码时需要对账号密码进行验证,此时需要定义一个函数,打开存储文件进行验证,当输入的账号和密码匹配,则登录成功登录次数:登录次数不超过3次,此时在for循环或者while循环中控制登录次数即可.