

**《软件工程》项目报告**

**题目： 儿童学习**

**课程名称： 软件工程**

**专业班级： CS1703**

**学 号： U201714660**

**姓 名： 饶鸿斌**

**同组成员： 朱凯丰**

**尕玛央宗**

**指导教师： 袁凌**

**报告日期： 2019.11.11**

**计算机科学与技术学院**

任 务 书

# 一 总体要求

1. 综合运用软件工程的思想，协同完成一个软件项目的开发，掌软件工程相关的技术和方法；

2. 组成小组进行选题，通过调研完成项目的需求分析，并详细说明小组成员的分工、项目的时间管理等方面。

3. 根据需求分析进行总体设计、详细设计、编码与测试等。

# 二 基本内容

根据给出的题目任选一题，自行组队，设计与开发中软件过程必须包括：

**1. 问题概述、需求分析：**正确使用相关工具和方法说明所开发软件的问题定义和需求分析，比如NABCD模型，Microsoft Visio，StarUML等工具 (20%)；

**2. 原型系统设计、概要设计、详细设计**：主要说明所开发软件的架构、数据结构及主要算法设计，比如墨刀等工具（35%）；

**3. 编码与测试**：编码规范，运用码云等平台进行版本管理，设计测试计划和测试用例（30%）；

**4．功能创新**：与众不同、特别吸引用户的创新（10%）；

**5. 用户反馈**：包括用户的使用记录，照片，视频等（5%）。

**目 录**

[**任务书** I](#_Toc24657625)

[**一 总体要求** I](#_Toc24657626)

[**二 基本内容** I](#_Toc24657627)

[**1 问题定义** 4](#_Toc24657628)

[1.1项目背景与意义 4](#_Toc24657629)

[1.1.1需求，NEED 4](#_Toc24657630)

[1.1.2 Approach，做法 4](#_Toc24657631)

[1.1.3 Benfit，好处 4](#_Toc24657632)

[1.1.4 Competitors，竞争 4](#_Toc24657633)

[1.1.5 Delivery，推广 5](#_Toc24657634)

[1.2 项目基本目标 5](#_Toc24657635)

[1.3 可行性分析 5](#_Toc24657636)

[1.3.1技术可行性 5](#_Toc24657637)

[1.3.2经济可行性 5](#_Toc24657638)

[1.3.3操作可行性 6](#_Toc24657639)

[1.4人员管理和项目进度管理 6](#_Toc24657640)

[**2 需求分析** 7](#_Toc24657641)

[2.1 E-R图、数据流图 7](#_Toc24657642)

[2.2 用例图 9](#_Toc24657643)

[2.3 原型系统设计 9](#_Toc24657644)

[**3 概要设计和详细设计** 13](#_Toc24657645)

[3.1 系统结构 13](#_Toc24657646)

[3.2 类图等 15](#_Toc24657647)

[3.3关键数据结构定义 17](#_Toc24657648)

[3.4 关键算法设计 18](#_Toc24657649)

[3.5 数据管理说明 20](#_Toc24657650)

[**4 实现与测试** 22](#_Toc24657651)

[4.1实现环境与代码管理 22](#_Toc24657652)

[4.2关键函数说明 23](#_Toc24657653)

[4.3 测试计划和测试用例 29](#_Toc24657654)

[4.4 结果分析 32](#_Toc24657655)

[**5 总结** 33](#_Toc24657656)

[5.1 用户反馈 33](#_Toc24657657)

[5.1 全文总结 33](#_Toc24657658)

[**6 体会** 34](#_Toc24657659)

[**附录** 35](#_Toc24657660)

1 问题定义

# 1.1项目背景与意义

## 1.1.1需求，NEED

N：家长在辅导幼儿（通常以小学一年级至三年级为主）学习数学时，常常无法获取优质且适合的习题辅助孩子学习。家长希望可以获得一款app帮助家长在儿童学习时提供习题库等的帮助。

## 1.1.2 Approach，做法

A:创建习题库，并为之分难度等级，提供给不同学习进度的儿童。强制性学习，降低儿童借学习之便玩手机等。

## 1.1.3 Benfit，好处

B:提供更为专业的学习辅导，解放家长在监督孩子学习之余还需提供额外的习题辅导的压力。帮助儿童循序渐进的学习知识。

## 1.1.4 Competitors，竞争

C：目前有大量的学习类app，这些app还附带有社交属性，在粘连用户上做的很好。但是它们并没有集中在小学低年级阶段，主要集中在英语类、初高中类。并且操作相对比较复杂，对儿童不友好。

## 1.1.5 Delivery，推广

D：联系小学进行线下的推广。

# 1.2 项目基本目标

项目的初衷是由于小明上小学后，老师要求每天完成30道口算题，每天出题改题成了一个大负担，小明妈妈想有个可以出题的软件。并且小明现在一年级，小明妈妈希望买一个软件可以用三年的。所以我们项目的基本目标是设计一款随机出题的学习app可以让一到三年级的学生更加方便地做口算题，使家长不必每天花大量时间给孩子出口算题，可以让普通用户通过互联网直接在线做题纠正错误并且解答。

# 1.3 可行性分析

## 1.3.1技术可行性

由于android studio3.5.1具有完整的数据库开发功能。通过端对端的设计、调试和提交,可以控制软件应用的 整个生命周期。本系统客户应用采用android studio3.5.1开发，因此这个小学生在线做题软件在技术上是可行的。

## 1.3.2经济可行性

系统所需的开发环境是要有一台能够联网的计算机,并安装了Windows10 的操作系统，并且要求安装sql server 2017 和visual studio 3.5.1两个 应用软件来设计前台和后台，而一般计算机都可以安装Windows 10的操作系统,和上述两个应用软件,都是完全免费的。至于参考资料在网络上都是可以找到的 没有的的话可以去图书馆借书,这些消费都是大家可以接受的,因此这个小学生在线做题软件在经济上是可行的。

## 1.3.3操作可行性

这个小学生做题软件的操作都是简单易懂的界面操作,是服务于小学生和学生家长的。在操作和使用上是没有什么技术含量的,并且小学三年级的学生基本都会熟练使用手机，所以操作上是可行的。

# 1.4人员管理和项目进度管理

尕玛央宗：原型模型（墨刀）

朱凯丰：UI界面设计

饶鸿斌：系统内部逻辑实现，包括数据库的实现、习题生成和求解等。

2 需求分析

# 2.1 E-R图、数据流图

用户需求：

1. 可以注册、登陆
2. 能做小学1至3年级的数学题，以填空题形式出现
3. 分为练习模式和考试模式，考试模式可以记录错题
4. 错题可以保存下来以供将来复习

我们用ER图来表示实体类型、属性和联系的方法 (如图2-1所示)。

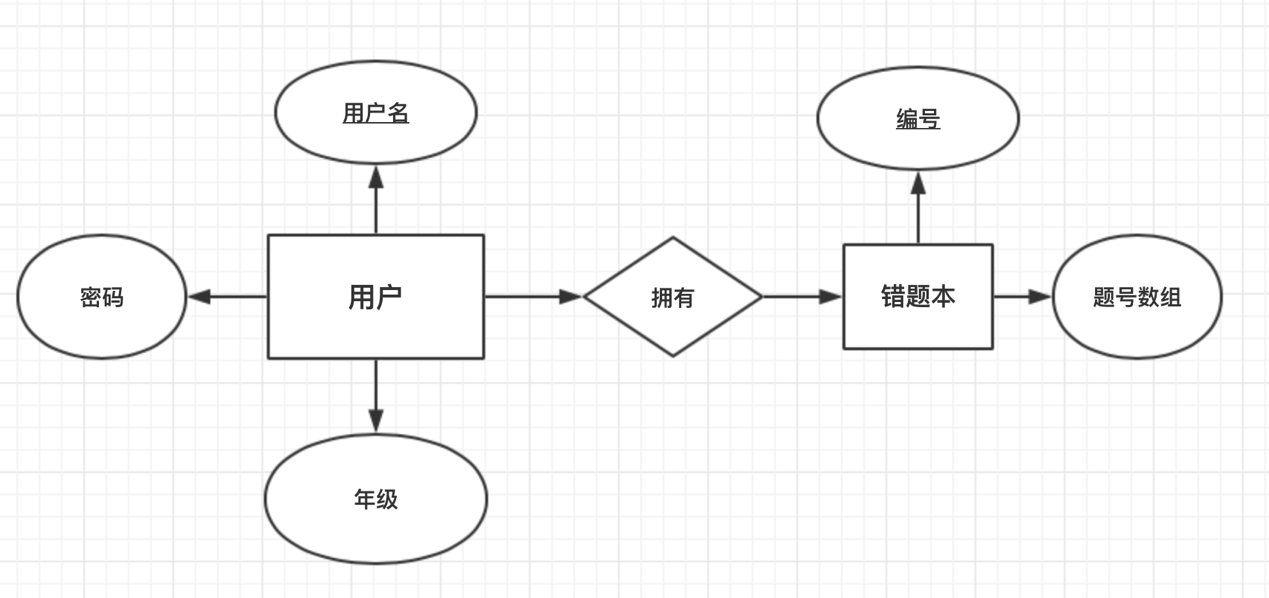


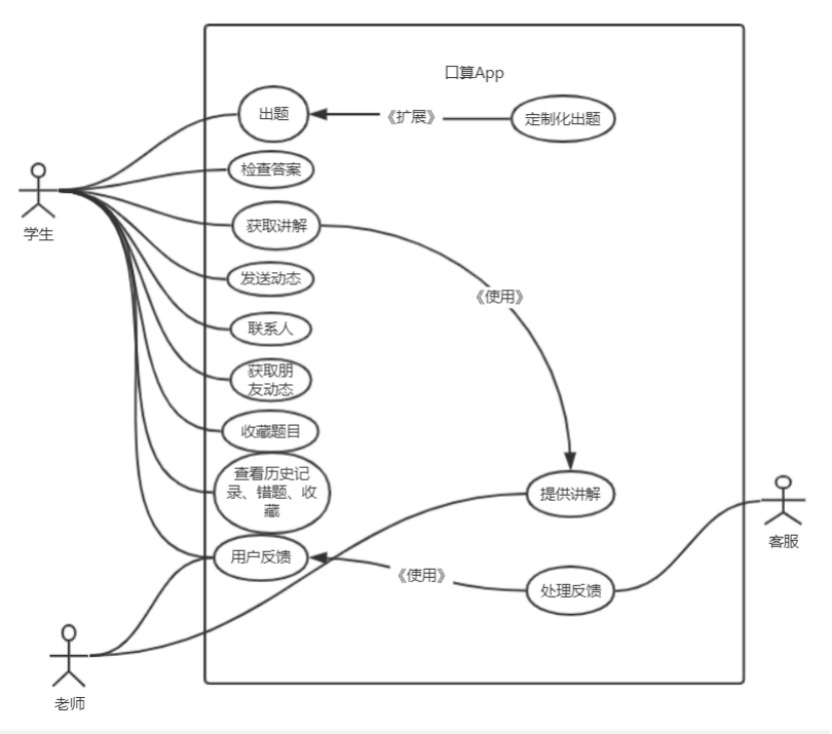
图2-1 E-R图

我们用DFD图描述数据的流动。



图2-1 DFD图

# 2.2 用例图



# 2.3 原型系统设计

1.开始界面



2.登陆注册页面



3.登陆页面和注册页面详情



4难度和年级选择页面



5.难度和年级选择详情



6.做题和题目解析页面



3 概要设计和详细设计

# 3.1 系统结构

在设计儿童学习系统的时候，我们初步分为了三大类：题目模块、主模块、做题模块。在题目模块中为了更有区分度，我们设计了生成不同难度和适配不同年级的题目分类模块。在主模块中我们提供了交互界面和显示界面，该模块是串联所有模块的模块。在做题模块中，我们设计了两个不同模式的做题：练习模式和错题模式，以满足不同学生的不同需求。

儿童学习系统分为以下部分：

登录模块：实现不同用户的登录，便于对不同用户进行数据的记录。

题目选择模块：为了满足不同年级和不同层次的儿童学习，我们针对题目进行了不同年级和不同难度的设计，将题目进行了分类。

考试模块：在进行习题练习的时候，为了更好的模拟考试的环境，我们设计了一个考试模块。在考试模块中对用户的做题时间进行了限制，在有限时间内完成的题目会进行打分操作，以便用户更好的了解自己的水平。

错题模块：用户在进行做题的时候，会出现做错题题的情况，为了帮助用户更快的获知自己的错题，我们设计了错题模块，收集用户的错题。

题目生成模块：该模块生成不同类型的数学计算题，其中包括加、减、乘、除等运算符出现次数的选择，计算数出现次数的选择，以实现生成不同难度的习题供不同年级和不同层次的学生使用。

成绩单模块：该模块记录了每次用户在考试中的得分情况，以及每次选择的题目类型。

下图位为系统结构图：

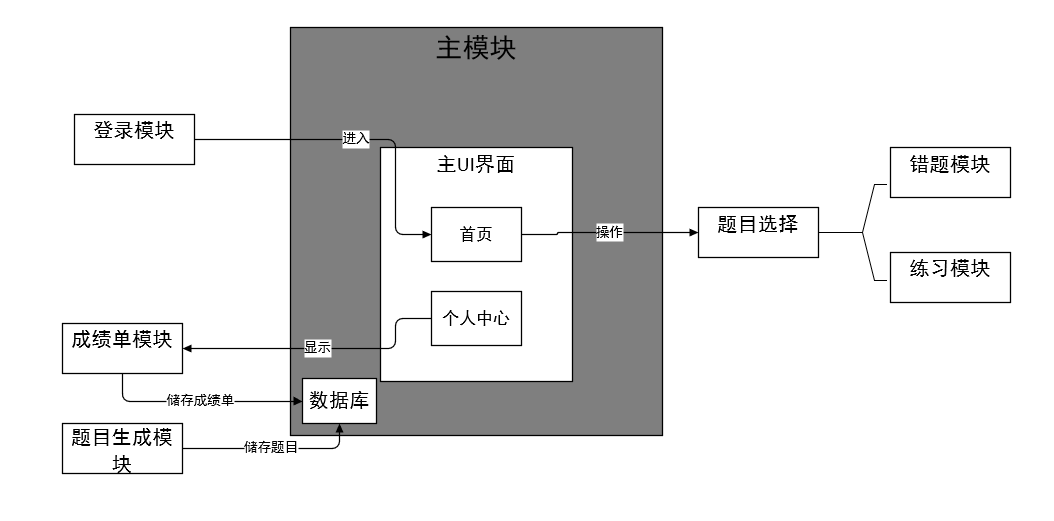


图3-1-1 系统结构图

系统模块的接口设计：

表3-1-1 系统模块的接口api

|  |  |  |
| --- | --- | --- |
| 模块接口 | Api | Api说明 |
| 题目生成模块 | loadQuestion(int grade, int number,  String tableName,  int hard,  int Version) | 参数：grade表示选择的年级  参数：number表示生成题目的数量  参数：tableName表示储存题目的数据库表  参数：hard表示生成题目的难度  参数：Version表示数据库的版本 |
| 登录模块一 | saveLoginInformation(boolean status ,String userName) | 参数：status表示该用户是否登录  参数：userName表示该用户的用户名  由于密码和用户名以键对的形式储存，故密码可通过用户名得知。 |
| 登录模块二（即注册模块） | saveRegisterInfo(String userName,String pwd) | 参数：userName表示该用户的用户名  参数：pwd表示该用户的密码  用户名和密码以键对的形式存储在数据库中。 |

成绩单模块、练习模块、错题模块、题目生成模块均为按钮选择，故不显示其接口。

# 3.2 类图等

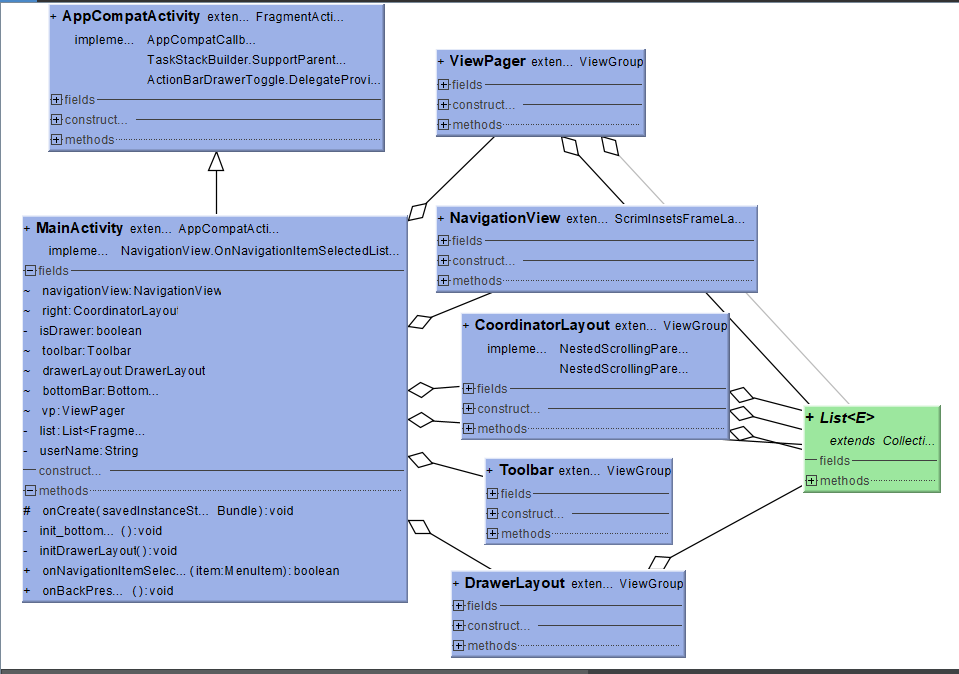


图3-2-1 主模块类图

说明：

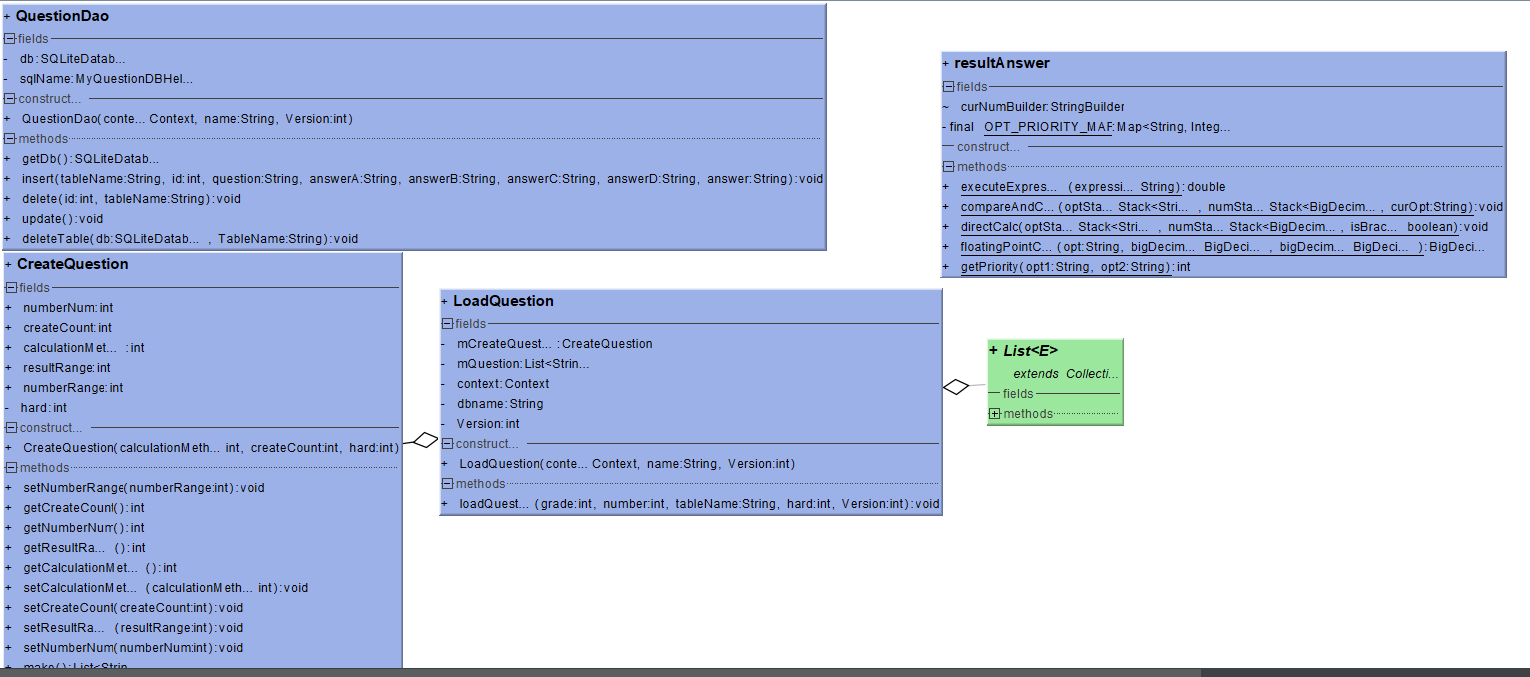
主活动采取了ViewPager来显示多个碎片，NavigationView和DrawerLayout则实现了侧滑栏的效果。

图3-2-2 题目生成模块类图

说明：

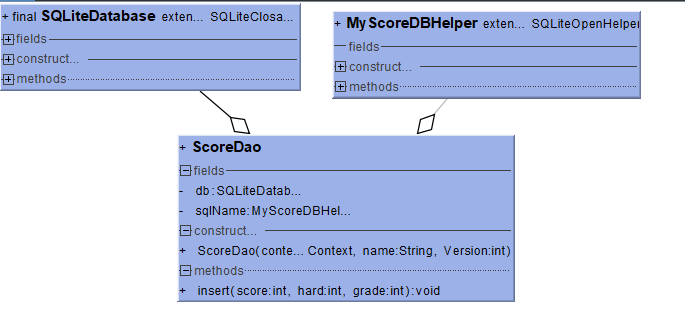
QuestionDao类中实现了对题目数据库的封装，实现了插入题目和删除题目等的操作。ResultAnswer类对题目进行了解答并返回答案。CreateQuestion实现了题目的生成。LoadQuestion实现了对生成的题目和答案进行插入数据库的操作。

图3-2-3 成绩生成模块类图

说明：采取了数据库对成绩进行收集，然后通过主模块中的成绩单显示界面显示出来。

图3-2-4 练习模式和错题模式类图

说明：练习模式和错题模式均采取了viewpager实现习题的切换设计。区别在与错题模式不再显示时间计时器。

# 3.3关键数据结构定义

系统需要处理的数据主要有题目实体类和成绩实体类。

其中题目实体类包括了：

int id //表示的是存储在数据库的id

String answerA、answerB、answerC、answerD //表示的是为该题目所提供的选择答案。为了方便，故在数据库中以String的形式存储。

String rightAnswer //表示的是正确的答案，方便后续错题的登记和成绩的计算

String selectedAnswer //表示的是用户的选择的答案选项

String explaination //表示的是用户对题目的笔记，但是在老师的建议下，笔记就放弃了。

int isWrong //表示的是记录该题目是否做错

int favorvite //表示的是该题目收录为喜欢的题目，但是实用性不强，没有实现。

成绩实体类包括了：

int id //表示的是储存在数据库的id

int score //表示的是此次考试模式的成绩

int hard //表示的是此次考试选择的题目的难度

int grade //表示的是此次考试选择的题目的年级层次

成绩实体类和题目实体类的关联：

在主界面选择年级和难度后在模式选择考试模式后会从相应的数据库中加载题目，此时题目实体类中的 answerA、answerB、answerC、answerD 、rightAnswer会得到填充。在考试模式界面中显示习题，用户在做题过程中会对selectedAnswer 进行填充。

在考试模式结束的时候，系统会对当前所做题目进行分析，错误的题目上的isWrong会进行记载。并且会对成绩进行统计，然后填充到成绩实体类中。



图3-3-1 系统处理的主要数据图

# 3.4 关键算法设计

系统主要算法应用在登录、注册模块中对用户账号的保护上和题目生成不同类别的生成和解答算法上。

登录、注册模块算法：

该模块上采取了MD5加密算法，算法是实现的基础是java提供的security库。MD5的作用是让大容量信息在用数字签名软件签署私人密钥前被"压缩"成一种保密的格式（就是把一个任意长度的字节串变换成一定长的十六进制数字串）。

注册时对用户名以键对（login\_info，用户名）的方式储存在SharedPreferences数据库中。对密码进行MD5加密后的字段和用户名以键对（用户名，加密字段）形式存入SharedPreferences数据库中。

题目生成模块算法：

该算法的实现比较简单，主要运用到的数据类型为list、String。

首先对按照输入的难度和年级确定每道题的数字数和运算符数和答案。例如：一年级的简单题目中题目的数量为50，运算符为加号和减号，每道题的数字数为两个（只有两个数字进行运算），且其范围为5以内的正整数，答案的范围为10以内的正整数。

然后，采取字符串和随机生成数，根据上述确定数字数和运算符的符号随机生成并添加到字符串中。

再然后，对该字符串表示的题目进行求解，符合上述答案范围的字符串添加到链表中表示该题目符合要求。

链表即为此次题目生成的集合，其中链表的大小取决于上述设定的题目的数量。

题目解题模块：

该算法是系统中最为复杂的算法，由于一开始考虑的比较多，我对题目中的参与运算的数字数的多少、正负和符号类型没有加以限制，并针对数学中“（）”表示先行运算的数学符号也没有加以限制。

该算法采用了两个栈，一个对数字进行储存、一个对符号进行储存。当读取的字符串的字符为数字时，压入数字栈；为符号时压入符号栈。当入栈的符号为“）”时执行运算，将数字栈中的两个数字出栈，符号栈中一个符号出栈，运算操作直至符号为“（”时停止，然后将运算结果压入数字栈。当不存在“（”、“）”时，字符串中的字符全部压入栈后，依次从栈顶取出数字和符号进行运算，当符号为同级时运算继续，否者运算停止，将运算结果压入数字栈。

运算中，针对符号，定义其符号的优先级：“（”“）”为最高的优先级；“x”“/”为同优先级，且优先级第二高，“+”、“-”为同优先级，且优先级最低。

算法流程图如下：

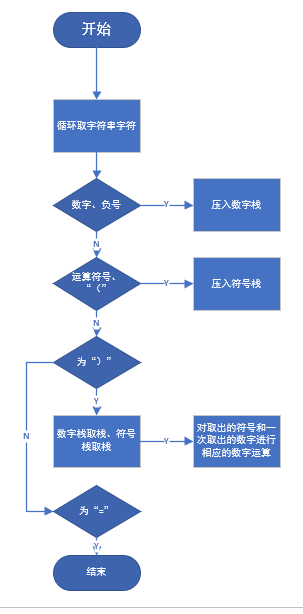


图3-4-1 解题算法流程图

# 3.5 数据管理说明

系统采取了本地的数据库进行数据的储存和访问。用户名和密码采取了SharedPreferences数据库，该数据库的大小较小，故用于用户名和密码的储存。其中用户名和login\_info以键对的形式、用户名和密码的MD5加密字段以键对的形式储存。当登录的时候，取出相对于的用户名和MD5加密字段与用户登录时输入的用户名和密码（采取MD5加密算法生成加密字段）进行匹配，匹配合格则登录成功，反之登录失败。

习题集和成绩单采取了本地SQLite数据库进行储存，该数据库体量合适。对每个用户的不同年级、不同难度习题生成一个对应的数据库以及对该用户的成绩生成一个对应的数据库。

在访问数据库的过程中，我设计了两个访问类TestDao和ScoreDao。类中采取SQL语句对数据库进行打开数据库操作、对表中的数据进行读取的操作。

4 实现与测试

# 4.1实现环境与代码管理

实验环境：

IDE：android studio3.5.1版本。

JRE:1.8.0\_202-release-1483-b03 amd64

JVM:OpenJDK 64-Bit Server VM by JetBrains s.r.o

SDK Platforms：Android 10.0Q

真机测试：小米手机6

Android版本9

运行内存6GB

处理器 骁龙835八核 最高2.45Ghz

码云平台签入记录截图如图4-1所示：

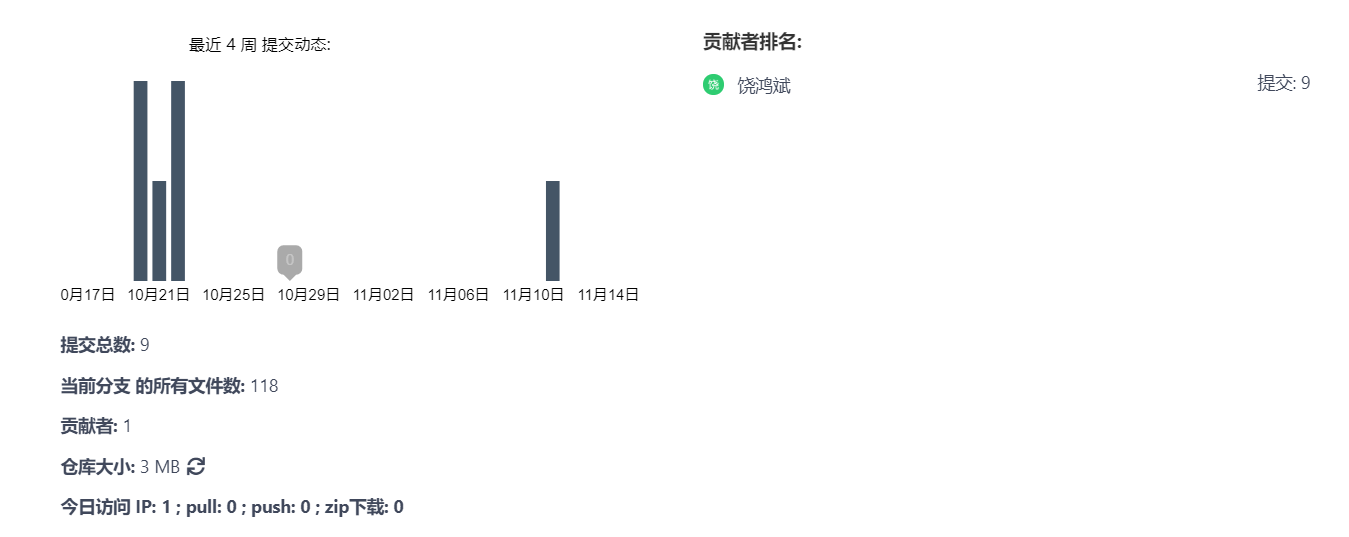


图4-1 码云平台代码签入

# 4.2关键函数说明

该系统主要的函数和类为：MainAcitivty 、ExamActivity、ScoreGradeActivity、Question\_fragment、home\_fragment、Question、Score、TestDao、ScoreDao、LoadQuestion。

1、MainActivity:

MainActivity的类关系和函数关系如下图所示：

（所有函数关系截图中图标表示的意思：C表示类，I表示接口，f表示字段，m表示方法。）

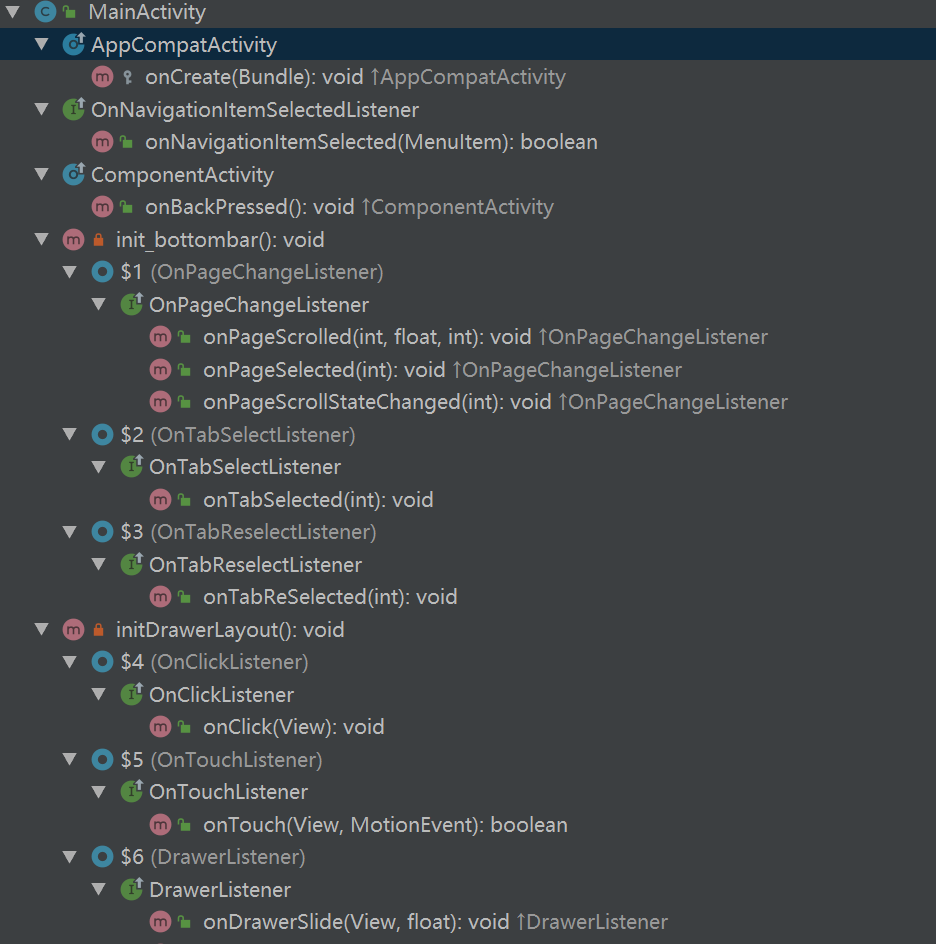


图4-2-1 MainActivity类关系和函数关系图

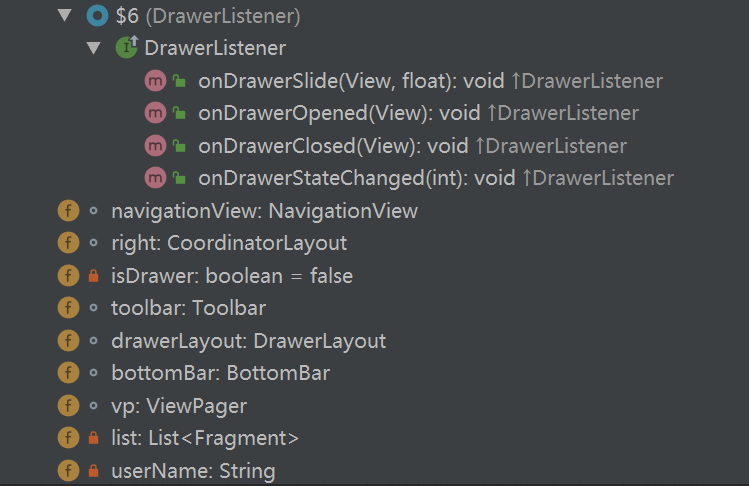


图4-2-2 MainActivity类关系和函数关系图

说明：

initDrawerLayout函数加载了侧滑栏

init\_bottombar函数加载了底部栏

2、LoadQuestion：

作用：加载习题。

LoadQuestion的类关系和函数关系如下图所示：

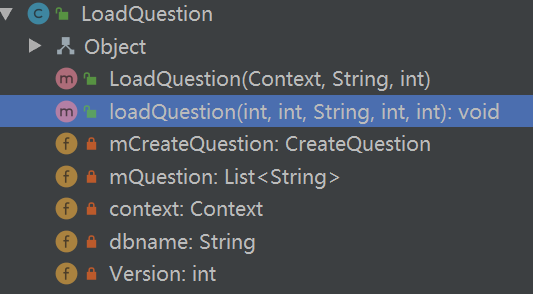


图4-2-3 LoadQuestion类关系和函数关系图

说明：该类中调用了CreateQuestion函数来创建习题。

3、ExamActivity:

作用：加载习题界面并提供上下题、交卷等交互按钮。

ExamActivity的类关系和函数关系如下图所示：

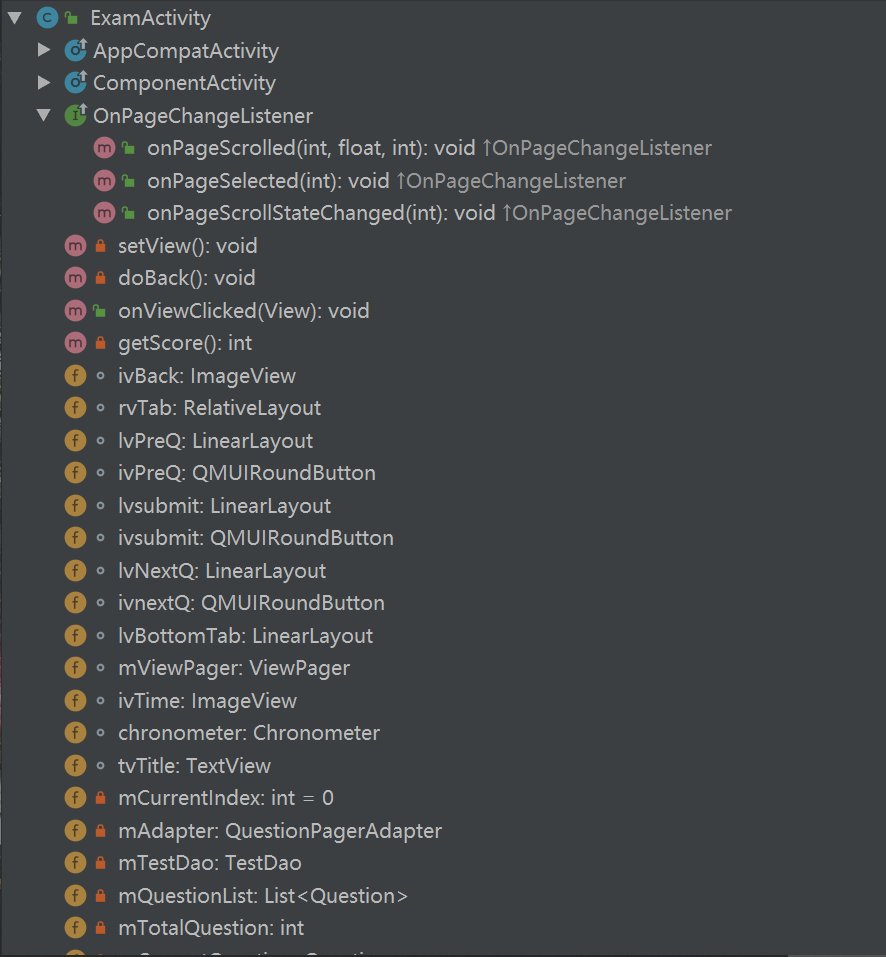


图4-2-4 ExamActivity类关系和函数关系图

说明：

OnPageChangeListener：使用了Viewpager加载不同的习题碎片。

4、Question\_fragment:

作用：呈现出习题和提供的选项。

Question\_fragment的类关系和函数关系如下图所示：

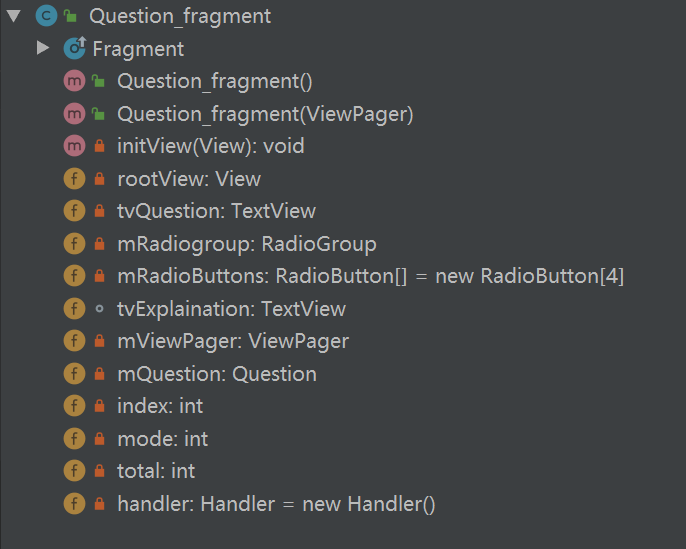


图4-2-5 Question\_fragment类关系和函数关系图

说明：

该碎片采取了Radiogroup来加载提供的习题选项。

5、ScoreGradeActivity:

作用：呈现出成绩单界面。

ScoreGradeActivity的类关系和函数关系如下图所示：

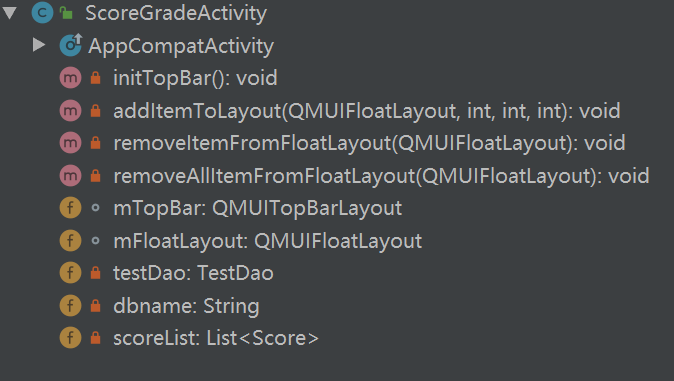


图4-2-6 ScoreGradeAcitivity类关系和函数关系图

说明：

调用addItemToLayout函数动态添加每一次的成绩。为了确保成绩单均能显示出每次的成绩，在进入成绩单界面的时候，调用removeAllItemFromFloatLayout函数清除之前的成绩显示，然后再动态添加成绩。

6、home\_fragment:

作用：呈现出习题选择界面。

Home\_fragment的类关系和函数关系如下图所示：

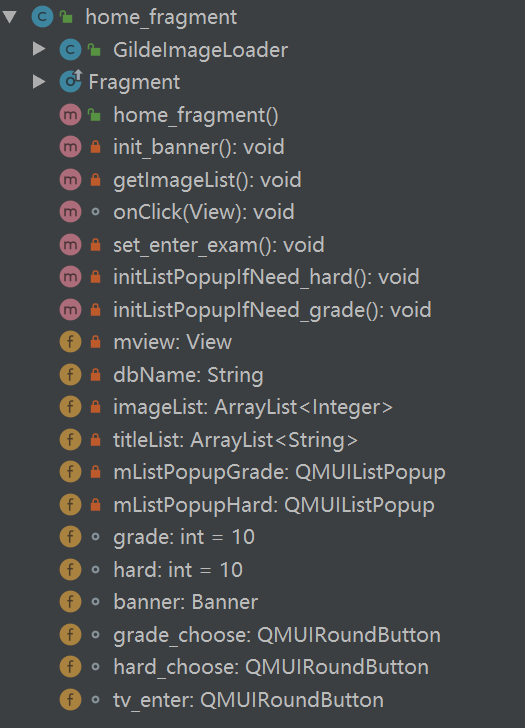


图4-2-7 home\_fragment类关系和函数关系图

说明：

该碎片使用了两个悬浮框（调用initListPopupIfNeed函数）来用作用户的选择年级和难度。调用了init\_banner函数来实现首页的图片轮转效果。

# 4.3 测试计划和测试用例

该系统生成了一个debug版的apk文件，故我们可以在Android手机上测试。

多用户模块:

由于登录后直接进入主界面，故多用户是不方便截图检查的。所以我们在主界面的侧滑栏部分进行检查。

检查截图如下：

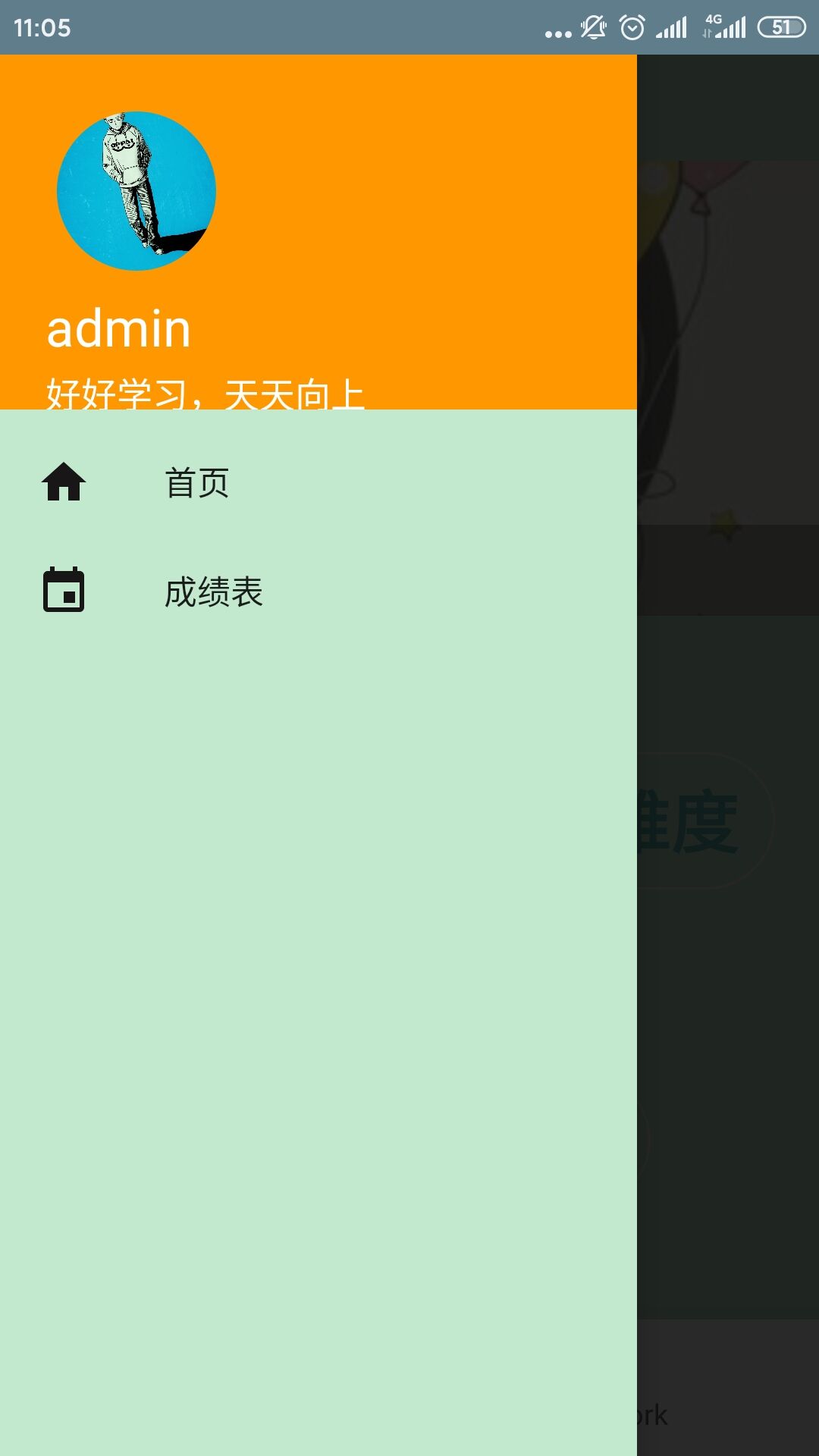
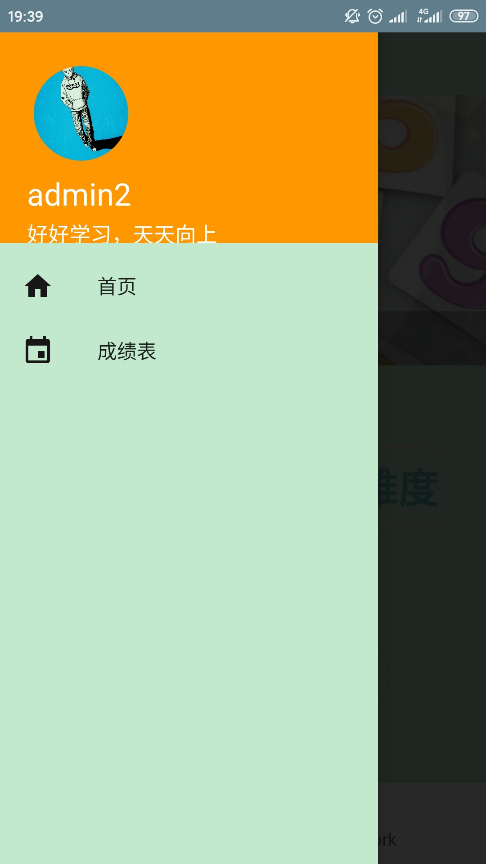


图4-3-1 多用户模块检验图

题目选择模块：

该模块是个交互模块，是个固定显示的模块。

模块显示截图如下：

图4-3-2 题目选择模块检验图

考试模块：

由于数据是内部封装好的，故没有特别设置的测试数据，直接使用数据库中的数据进行测试。该模块显示了计时器，当点击交卷的时候，会显示成绩。

模块显示截图如下：

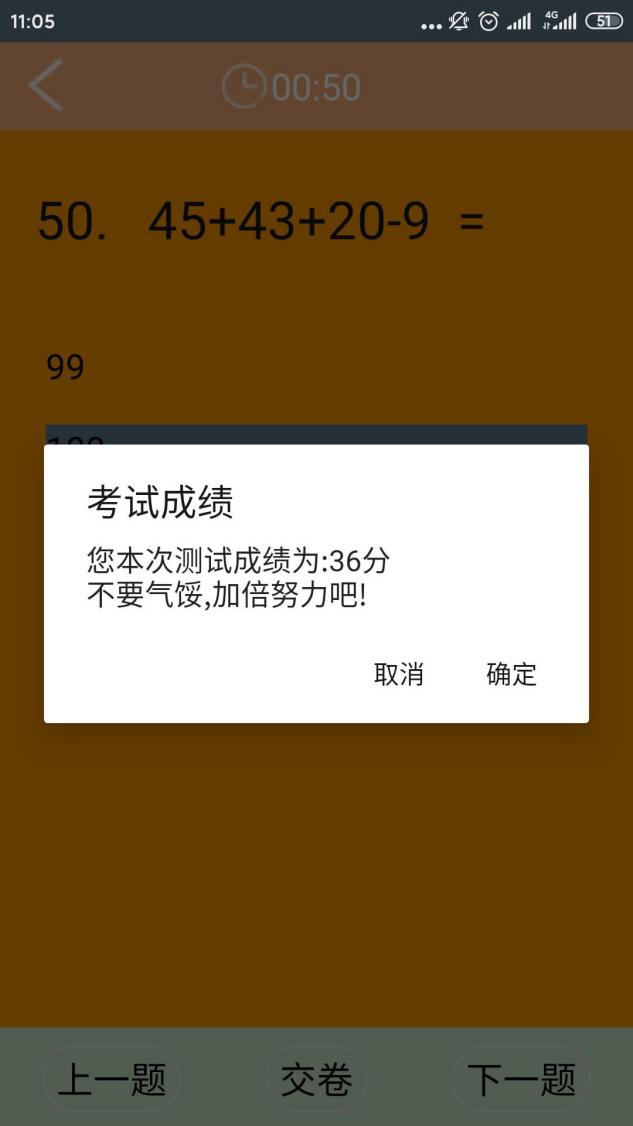
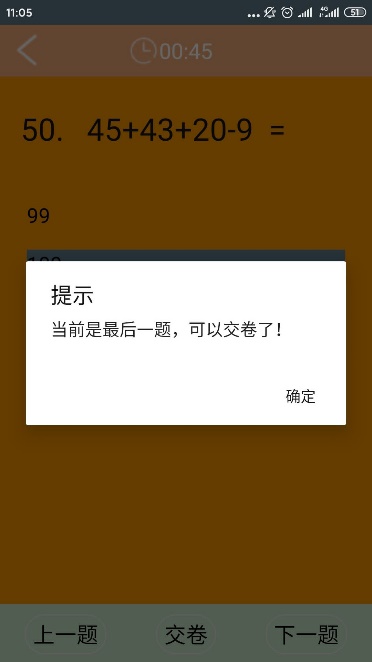


图4-3-3 考试模块检验图

成绩单模块：

该模块使用到的数据为上图中考试模式中得到的成绩。其他的成绩是提前测试的成绩。

模块显示截图如下：

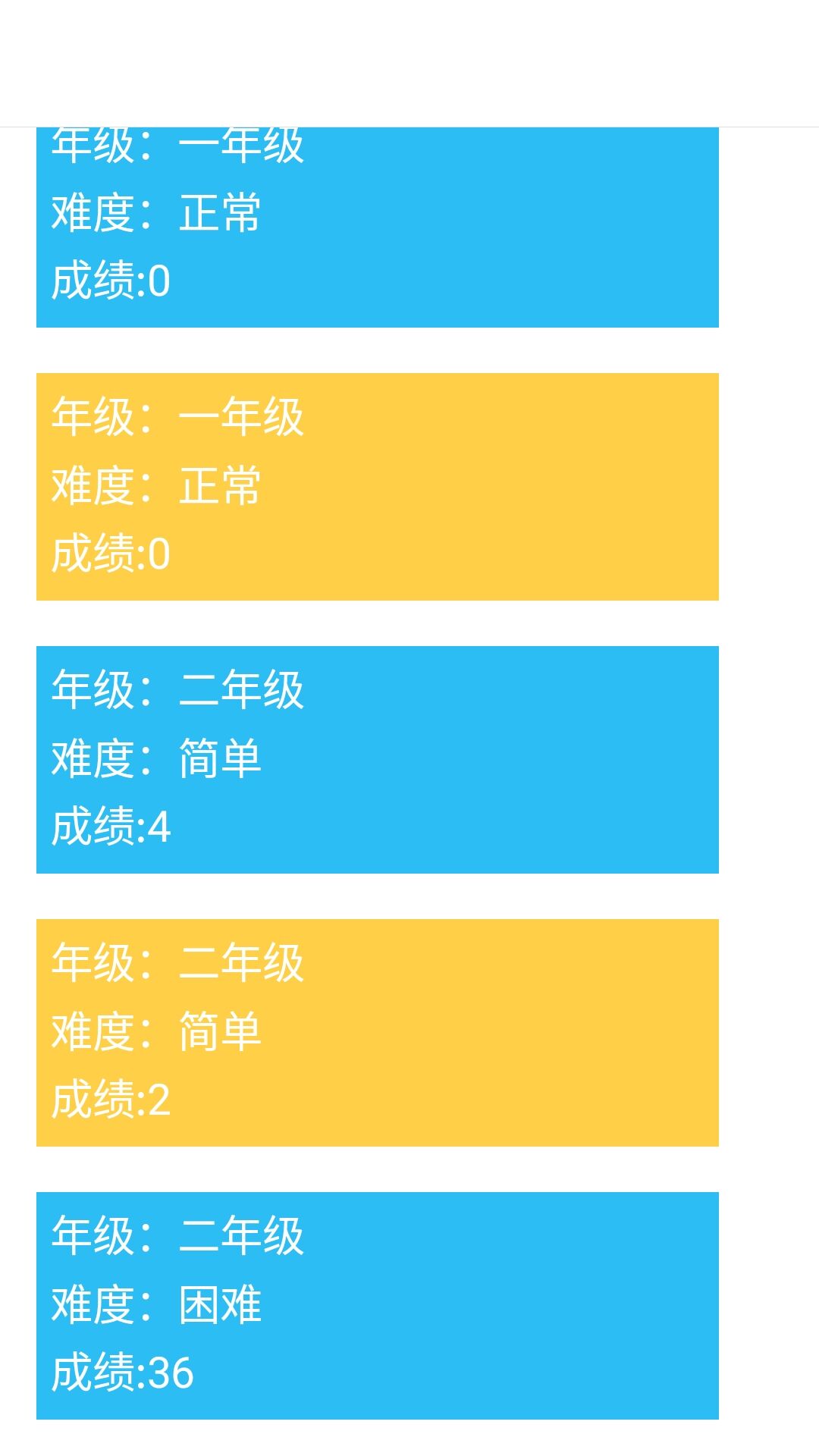


图4-3-4 成绩单模块检验图

# 4.4 结果分析

由测试可知，该系统实现了多用户登录、每个用户拥有自己的数据库的功能。也实现了习题的分类选择和习题练习的两种模式：考试模式和错题模式。故该系统基本实现了用户的基本需求。

5 总结

# 5.1 用户反馈

* **优点**：
* 1.可以选择题目的难度和年级，满足不同用户的需求。
* 2.分为练习模式和考试模式，丰富用户的体验。
* 3.页面简洁，操作简单，适合小学生使用。
* 4.题目变化多，范围广。
* 5.有错题本，可以很方便地复习错题。
* 6.解析详细，对学习有很大帮助。
* 7.用户在考试模式的成绩会在个人中心记录，可以查看自己的历史考试情况，看到进步的脚印。
* **用户建议**：
* 1.可以加入除了填空题之外的题型。
* 3.希望加入计时功能。
* 4.能否扩展到更多年级。

# 5.1 全文总结

主要工作如下：

（1）实现了习题生成模块、登录模块、习题选择模块、练习模块、错题模块和成绩单模块。

（2）实现了数据库对成绩单和习题集的储存。

（3）实现了系统的内部逻辑，包括界面的切换等。

6 体会

本次作业是为了开发一个app便于小学生的学习和使用。在开发app的过程中，我一开始是希望仿照市场上成熟的相关的app，但是可行性不高。无论是UI的设计和内部逻辑的实现都存在很大的问题。所以我们决定仿照一部分。由于我的工作是内部逻辑和算法的实现，任务量比较大，所以该系统目前还存在很多的问题。实现的功能也十分有限。

在开发的过程中，也遇到了很多问题。但是大多数的问题都是可以依靠谷歌解决。其中让我最感到棘手的是实现SQLite数据库的时候，在更新库的过程中不断出现错误，这也导致我放弃一个用户只采取一个库的方式，而是创建多个库来储存不同分类的习题和成绩。其次，我在设计数据库的过程中，没有采取本地读取数据的方式而是随机生成习题的方式，这导致了用户的可练习的习题数量有限，这是我在设计之初没有考虑到位的地方。此外，这个系统的架构的设计还是存在不小的问题，类和类之间的耦合度太高，在进行重构和修改的时候，容易出现问题。

小组之间的沟通也不是很充分，在系统的交互上没有考虑到用户的感受，比如袁老师提出的小学阶段不存在小数和负数的情况，对于口算题，不要提供其答案和不要采取选择题的方式出题······这些都是我们小组在设计之初考虑不周的地方。

最后，我十分感谢我小组的两名成员和负责辅导我们软件工程大作业的老师们！

附录

关键代码部分：

1：CreateQuestion：

package Dao;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
**//按需生成题目，由于是小学三年级的题目，故答案为小数和负数的题目将被去除**public class CreateQuestion {  
 **//每道题的数字数，比如 2 , 3 ,4** public int **numberNum**;  
 **//生成题的数量** public int **createCount**;  
 **//运算类型** public int **calculationMethod**;  
 **//数值范围** public int **resultRange**;  
 public int **numberRange**;  
 private int **hard**;  
 public void setNumberRange(int numberRange) {  
 this.**numberRange** = numberRange;  
 }  
 public int getCreateCount() {  
 return **createCount**;  
 }  
 public int getNumberNum() {  
 return **numberNum**;  
 }  
  
 public int getResultRange() {  
 return **resultRange**;  
 }  
  
 public int getCalculationMethod() {  
 return **calculationMethod**;  
 }  
  
 public void setCalculationMethod(int calculationMethod) {  
 this.**calculationMethod** = calculationMethod;  
 }  
  
 public void setCreateCount(int createCount) {  
 this.**createCount** = createCount;  
 }  
  
 public void setResultRange(int resultRange) {  
 this.**resultRange** = resultRange;  
 }  
  
 public void setNumberNum(int numberNum) {  
 this.**numberNum** = numberNum;  
 }  
 public CreateQuestion(int calculationMethod,int createCount,int hard)  
 {  
 this.**createCount** =createCount;  
 this.**calculationMethod** = calculationMethod;  
 this.**hard** = hard;  
 }  
 **//生成题目** public List<String> make() {  
 Random r = new Random();  
 int j=0;  
 if (**calculationMethod**==1)  
 {  
 if (**hard**==1)  
 {  
 **numberNum**=2;  
 **numberRange**=5;  
 **resultRange**=10;  
 j=1;  
 }  
 else if(**hard**==2) {  
 **numberNum** = 2;  
 **numberRange** = 10;  
 **resultRange** = 20;  
 j = 1;  
 }  
 else if(**hard**==3)  
 {  
 **numberNum** = 2;  
 **numberRange** = 20;  
 **resultRange** = 40;  
 j = 1;  
 }  
 }  
 else if (**calculationMethod**==2) {  
 if (**hard** == 1) {  
 **numberNum** = 3;  
 **numberRange** = 20;  
 **resultRange** = 50;  
 j = (r.nextInt(1) == 0) ? 1 : 3;  
 }  
 else if(**hard**==2) {  
 **numberNum** = 4;  
 **numberRange** = 50;  
 **resultRange** = 100;  
 j = (r.nextInt(1) == 0) ? 1 : 3;  
 }  
 else if(**hard**==3)  
 {  
 **numberNum** = 4;  
 **numberRange** = 50;  
 **resultRange** = 100;  
 j = 3;  
 }  
 }  
 else if (**calculationMethod**==3) {  
 if (**hard** == 1) {  
 **numberNum** = 2;  
 **numberRange** = 20;  
 **resultRange** = 100;  
 j = 3;  
 }  
 else if (**hard**==2)  
 {  
 **numberNum** = 3;  
 **numberRange** = 50;  
 **resultRange** = 200;  
 j = 3;  
 }  
 else if (**hard**==3)  
 {  
 **numberNum** = 4;  
 **numberRange** = 120;  
 **resultRange** = 500;  
 j = 3;  
 }  
 }  
 char[] methods = new char[**numberNum** - 1];  
 int[] nums = new int[**numberNum**];  
 List<String> l = new ArrayList<>();  
 for (int count = 0; count < **createCount**; ) {  
 for (int i = 0; i < methods.**length**; i++) {  
 int temp=r.nextInt(j);**//生成1,2,3,0:0为加法，1为减法，2为乘法，3为除法** switch (temp)  
 {  
 case 0:  
 methods[i] = '+';  
 break;  
 case 1:  
 methods[i] = '-';  
 break;  
 case 2:  
 methods[i] = '÷';  
 break;  
 default:  
 methods[i] = 'x';  
 break;  
 }  
 }  
 nums[0]=r.nextInt(**numberRange**);  
 for (int i=1;i<**numberNum**;i++)  
 {  
 nums[i]=r.nextInt(**numberRange**);  
 while (methods[i-1]=='÷'&&nums[i]==0)**//排除除0的情况** {  
 nums[i]=r.nextInt(**resultRange**);  
 }  
 }  
 if(resultAnswer.executeExpression(create(methods,nums))%1==0&&resultAnswer.executeExpression(create(methods,nums))>=0)**//保证生成题目的答案是非负数** {  
 l.add(create(methods, nums));  
 count++;  
 }  
 }  
 return l;  
 }  
 String create(char[] methods, int[] nums) {  
 StringBuilder sb = new StringBuilder();  
 sb.append(nums[0]);  
 for (int i = 0; i < methods.**length**; i++) {  
 sb.append(methods[i]).append(nums[i + 1]);  
 }  
 return sb.toString();  
 }  
  
}

2、resultAnswer:

package Dao;  
  
import java.math.BigDecimal;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Stack;  
**//计算题目，得到答案。**public class resultAnswer {  
 StringBuilder **curNumBuilder** = new StringBuilder(80);  
 private static final Map<String, Integer> ***OPT\_PRIORITY\_MAP*** = new HashMap<String, Integer>() {  
 private static final long ***serialVersionUID*** = 6968472606692771458L;  
 {  
 put("+", 2);  
 put("-", 2);  
 put("x", 3);  
 put("÷", 3);  
 put("(", 0);  
 put(")", 7);  
 put("=", 20);  
 }  
 };  
 public static double executeExpression(String expression) {  
 StringBuilder curNumBuilder = new StringBuilder(80);  
 Stack<String> optStack = new Stack<>(); **// 运算符栈** Stack<BigDecimal> numStack = new Stack<>(); **// 数值栈** for (int i = 0; i < expression.length(); i++) {  
  
 char c = expression.charAt(i);  
 if (c != ' ') { **// 空白字符直接丢弃** if ((c >= '0' && c <= '9') || (c == '-'&&!(expression.charAt(i-1)>='0'&&expression.charAt(i-1)<='9')) || c == '.') {  
 curNumBuilder.append(c); **//** }  
 else {  
 **//数字压栈** if (curNumBuilder.length() > 0) {  
 numStack.push(new BigDecimal(curNumBuilder.toString()));  
 curNumBuilder.delete(0, curNumBuilder.length());  
 }  
 **//curOpt为运算符** String curOpt = String.valueOf(c);  
 if (optStack.empty()) {**// 运算符栈栈顶为空则直接入栈** optStack.push(curOpt);  
 } else {  
 if (curOpt.equals("(")) {**// 当前运算符为左括号，直接入运算符栈** optStack.push(curOpt);  
 } else if (curOpt.equals(")")) {**// 当前运算符为右括号，触发括号内的字表达式进行计算** directCalc(optStack, numStack, true);  
 } else if (curOpt.equals("=")) {**// 当前运算符为等号，触发整个表达式剩余计算，并返回总的计算结果** directCalc(optStack, numStack, false);  
 return numStack.pop().doubleValue();  
 } else {  
 **// 当前运算符为加减乘除之一，要与栈顶运算符比较，判断是否要进行一次二元计算** compareAndCalc(optStack, numStack, curOpt);  
 }  
 }  
 }  
 }  
 }  
 if (curNumBuilder.length() > 0) { **// 如果追加器有值，说明之前读取的字符是数值，而且此时已经完整读取完一个数值** numStack.push(new BigDecimal(curNumBuilder.toString()));  
 }  
 if(!optStack.empty())  
 directCalc(optStack, numStack, false);  
 return numStack.pop().doubleValue();  
 }  
  
 public static void compareAndCalc(Stack<String> optStack, Stack<BigDecimal> numStack, String curOpt) {  
 String peekOpt = optStack.peek();  
 int priority = getPriority(peekOpt, curOpt);  
 if (priority == -1 || priority == 0) {  
 String opt = optStack.pop(); **// 当前参与计算运算符** BigDecimal num2 = numStack.pop(); **// 当前参与计算数值2** BigDecimal num1 = numStack.pop(); **// 当前参与计算数值1** BigDecimal bigDecimal = floatingPointCalc(opt, num1, num2);  
 **// 计算结果当做操作数入栈** numStack.push(bigDecimal);  
 if (optStack.empty()) {  
 optStack.push(curOpt);  
 } else {  
 compareAndCalc(optStack, numStack, curOpt);  
 }  
 } else {**// 当前运算符优先级高，则直接入栈** optStack.push(curOpt);  
 }  
 }  
  
 public static void directCalc(Stack<String> optStack, Stack<BigDecimal> numStack, boolean isBracket) {  
 String opt = optStack.pop();**// 当前参与计算运算符** boolean simple\_word = false;  
 if(opt.equals("("))**//防止出现（-2）的情况** {  
 simple\_word=true;  
 if(!optStack.empty())  
 {  
 opt=optStack.pop();  
 }  
 else  
 return;  
 }  
 BigDecimal num2 = numStack.pop(); **// 当前参与计算数值2** BigDecimal num1 = numStack.pop(); **// 当前参与计算数值1** BigDecimal bigDecimal = floatingPointCalc(opt, num1, num2);  
 **// 计算结果当做操作数入栈** numStack.push(bigDecimal);  
 if (isBracket&&!optStack.empty()) {  
 if ("(".equals(optStack.peek())) {**// 括类型则遇左括号停止计算，同时将左括号从栈中移除** optStack.pop();  
 }  
 else if(simple\_word==false){  
 directCalc(optStack, numStack, isBracket);  
 }  
 }  
 else {  
 if (!optStack.empty()) {  
 **// 等号类型只要栈中还有运算符就继续计算** directCalc(optStack, numStack, isBracket);  
 }  
 }  
 }  
  
 public static BigDecimal floatingPointCalc(String opt, BigDecimal bigDecimal1, BigDecimal bigDecimal2) {  
 BigDecimal resultBigDecimal = new BigDecimal(0);  
 switch (opt) {  
 case "+":  
 resultBigDecimal = bigDecimal1.add(bigDecimal2);  
 break;  
 case "-":  
 resultBigDecimal = bigDecimal1.subtract(bigDecimal2);  
 break;  
 case "x":  
 resultBigDecimal = bigDecimal1.multiply(bigDecimal2);  
 break;  
 case "÷":  
 resultBigDecimal = bigDecimal1.divide(bigDecimal2,6, BigDecimal.***ROUND\_HALF\_UP***);  
 break;  
 default:  
 break;  
 }  
 return resultBigDecimal;  
 }  
  
 public static int getPriority(String opt1, String opt2) {  
 int priority = ***OPT\_PRIORITY\_MAP***.get(opt2) - ***OPT\_PRIORITY\_MAP***.get(opt1);  
 return priority;  
 }  
}

3、QuestionDao：

package Dao;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.sqlite.SQLiteDatabase;  
  
import java.util.List;  
  
import exam.MyQuestionDBHelper;  
  
**//对数据库进行封装操作**public class QuestionDao {  
 private SQLiteDatabase **db**;  
 private MyQuestionDBHelper **sqlName**;  
 public QuestionDao(Context context, String name,int Version) {  
 **sqlName** = new MyQuestionDBHelper(context,name,Version);  
 }  
 public SQLiteDatabase getDb() {  
 **db**=**sqlName**.getWritableDatabase();  
 return **db**;  
 }  
 **//添加数据---题目** public void insert(String tableName, int id, String question, String answerA, String answerB, String answerC, String answerD,String answer) {  
 **db** = **sqlName**.getWritableDatabase();  
 ContentValues contentValues = new ContentValues();  
 contentValues.put("ID", id);  
 contentValues.put("question", question);  
 contentValues.put("answerA", answerA);  
 contentValues.put("answerB", answerB);  
 contentValues.put("answerC", answerC);  
 contentValues.put("answerD", answerD);  
 contentValues.put("answer",answer);  
 **db**.insert(tableName, null, contentValues);  
 **db**.close();  
 }  
 **//删除数据** public void delete(int id,String tableName)  
 {  
 **db** = **sqlName**.getWritableDatabase();  
 **db**.delete(tableName,"ID=?",new String[]{"id"});  
 **db**.close();  
 }  
 **//更新数据** public void update()  
 {  
 }  
 public void deleteTable(SQLiteDatabase db,String TableName)  
 {  
 db=**sqlName**.getWritableDatabase();  
 String sql="DELETE FROM "+TableName;  
 db.execSQL(sql);  
 db.close();  
 }  
  
}

4、ScoreDao:

package Dao;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.sqlite.SQLiteDatabase;  
import exam.MyScoreDBHelper;  
**//数据库命名规范：userName+"ScoreGrade.db"**public class ScoreDao {  
 private SQLiteDatabase **db**;  
 private MyScoreDBHelper **sqlName**;  
 public ScoreDao(Context context, String name, int Version) {  
 **sqlName** = new MyScoreDBHelper(context,name,Version);  
 }  
 public void insert(int score,int hard,int grade)  
 {  
 **db**= **sqlName**.getWritableDatabase();  
 ContentValues contentValues = new ContentValues();  
 contentValues.put("score",score);  
 contentValues.put("hard",hard);  
 contentValues.put("grade",grade);  
 **db**.insert("ScoreGrade",null,contentValues);  
 **db**.close();  
 }  
}

5、TestDao:

package Dao;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.util.Log;  
  
import java.lang.annotation.Target;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
import entity.Question;  
import entity.Score;  
  
**//查询指定数据库获得所有问题，返回问题的list集合**public class TestDao {  
 private SQLiteDatabase **db**;  
 private String **dataName**;  
 private Context **context**;  
 private String **TableName**;  
 public TestDao(Context context,String dataName)  
 {  
 **db** = SQLiteDatabase.openDatabase(context.getDatabasePath(dataName).getPath(),null,SQLiteDatabase.***OPEN\_READWRITE***);  
 this.**dataName**=dataName;  
 this.**context**=context;  
 }  
 public SQLiteDatabase getDb()  
 {  
 return **db**;  
 }  
  
 public void setTableName(String tableName) {  
 **TableName** = tableName;  
 }  
 **//取出成绩** public List<Score> getScore()  
 {  
 if (**db**==null||!**db**.isOpen())  
 {  
 **db** = SQLiteDatabase.openDatabase(**context**.getDatabasePath(**dataName**).getPath(),null,SQLiteDatabase.***OPEN\_READWRITE***);  
 }  
 **//sql语句** String sql = ("select \* from "+**TableName**);  
 List<Score>scoreList = new ArrayList<Score>();  
 **//查询数据** Cursor cursor = **db**.rawQuery(sql,null);  
 if(cursor.getCount()>0)  
 {  
 cursor.moveToFirst();  
 for(int i=0;i<cursor.getCount();i++)  
 {  
 cursor.moveToPosition(i);  
 Score score = new Score();  
 score.**score** = cursor.getInt(cursor.getColumnIndex("score"));  
 score.**grade**=cursor.getInt(cursor.getColumnIndex("grade"));  
 score.**hard**=cursor.getInt(cursor.getColumnIndex("hard"));  
 score.**id** = cursor.getInt(cursor.getColumnIndex("ID"));  
 scoreList.add(score);  
 }  
 }  
 **db**.close();  
 **db**=null;  
 return scoreList;  
 }  
 **//取出题目** public List<Question> getQuestions()  
 {  
 if (**db**==null||!**db**.isOpen())  
 {  
 **db** = SQLiteDatabase.openDatabase(**context**.getDatabasePath(**dataName**).getPath(),null,SQLiteDatabase.***OPEN\_READWRITE***);  
 }  
 **//sql语句** String sql = ("select \* from "+**TableName**);  
 List<Question> list = new ArrayList<Question>();  
 **//查询数据** Cursor cursor = **db**.rawQuery(sql,null);  
 if(cursor.getCount()>0)  
 {  
 cursor.moveToFirst();  
 for(int i=0;i<cursor.getCount();i++)  
 {  
 cursor.moveToPosition(i);  
 Random r = new Random();  
 int j= r.nextInt(3);  
 Question question = new Question();  
 question.**question** = cursor.getString(cursor.getColumnIndex("question"));  
 if (j==0) {  
 question.**answerA** = cursor.getString(cursor.getColumnIndex("answerA"));  
 question.**answerB** = cursor.getString(cursor.getColumnIndex("answerB"));  
 question.**answerC** = cursor.getString(cursor.getColumnIndex("answerC"));  
 question.**answerD** = cursor.getString(cursor.getColumnIndex("answerD"));  
 }  
 else if (j==1)  
 {  
 question.**answerB** = cursor.getString(cursor.getColumnIndex("answerA"));  
 question.**answerA** = cursor.getString(cursor.getColumnIndex("answerB"));  
 question.**answerC** = cursor.getString(cursor.getColumnIndex("answerC"));  
 question.**answerD** = cursor.getString(cursor.getColumnIndex("answerD"));  
 }  
 else if (j==2)  
 {  
 question.**answerC** = cursor.getString(cursor.getColumnIndex("answerA"));  
 question.**answerB** = cursor.getString(cursor.getColumnIndex("answerB"));  
 question.**answerA** = cursor.getString(cursor.getColumnIndex("answerC"));  
 question.**answerD** = cursor.getString(cursor.getColumnIndex("answerD"));  
  
 }  
 else if (j==3)  
 {  
 question.**answerD** = cursor.getString(cursor.getColumnIndex("answerA"));  
 question.**answerB** = cursor.getString(cursor.getColumnIndex("answerB"));  
 question.**answerC** = cursor.getString(cursor.getColumnIndex("answerC"));  
 question.**answerA** = cursor.getString(cursor.getColumnIndex("answerD"));  
 }  
 question.**rightAnswer** = cursor.getString(cursor.getColumnIndex("answer"));  
 question.**isWrong** = cursor.getInt(cursor.getColumnIndex("isWrong"));  
 question.**explaination** = cursor.getString(cursor.getColumnIndex("explaination"));  
 question.setSelectedAnswer(null);  
 question.**id** = cursor.getInt(cursor.getColumnIndex("ID"));  
 list.add(question);  
 }  
 }  
 **db**.close();  
 **db**=null;  
 return list;  
 }  
 public List<Question> getWrongQuestion()  
 {  
 if(**db**==null||!**db**.isOpen()){  
 **db** = SQLiteDatabase.openDatabase(**context**.getDatabasePath(**dataName**).getPath(),null,SQLiteDatabase.***OPEN\_READWRITE***);  
 }  
 Cursor cursor = **db**.rawQuery("select \* from " + **TableName** + " where isWrong=?",new String[]{"1"});**//1表示的是错题** List<Question> wrongList = new ArrayList<Question>();  
 if(cursor.getCount() > 0) {  
 cursor.moveToFirst();  
 for(int i = 0; i < cursor.getCount(); i++)  
 {  
 cursor.moveToPosition(i);  
 Question question = new Question();  
 Random r = new Random();  
 int j= r.nextInt(3);  
 question.**question** = cursor.getString(cursor.getColumnIndex("question"));  
 if (j==0) {  
 question.**answerA** = cursor.getString(cursor.getColumnIndex("answerA"));  
 question.**answerB** = cursor.getString(cursor.getColumnIndex("answerB"));  
 question.**answerC** = cursor.getString(cursor.getColumnIndex("answerC"));  
 question.**answerD** = cursor.getString(cursor.getColumnIndex("answerD"));  
 }  
 else if (j==1)  
 {  
 question.**answerB** = cursor.getString(cursor.getColumnIndex("answerA"));  
 question.**answerA** = cursor.getString(cursor.getColumnIndex("answerB"));  
 question.**answerC** = cursor.getString(cursor.getColumnIndex("answerC"));  
 question.**answerD** = cursor.getString(cursor.getColumnIndex("answerD"));  
 }  
 else if (j==2)  
 {  
 question.**answerC** = cursor.getString(cursor.getColumnIndex("answerA"));  
 question.**answerB** = cursor.getString(cursor.getColumnIndex("answerB"));  
 question.**answerA** = cursor.getString(cursor.getColumnIndex("answerC"));  
 question.**answerD** = cursor.getString(cursor.getColumnIndex("answerD"));  
  
 }  
 else if (j==3)  
 {  
 question.**answerD** = cursor.getString(cursor.getColumnIndex("answerA"));  
 question.**answerB** = cursor.getString(cursor.getColumnIndex("answerB"));  
 question.**answerC** = cursor.getString(cursor.getColumnIndex("answerC"));  
 question.**answerA** = cursor.getString(cursor.getColumnIndex("answerD"));  
 }  
 question.**rightAnswer** = cursor.getString(cursor.getColumnIndex("answer"));  
 question.**isWrong** = cursor.getInt(cursor.getColumnIndex("isWrong"));  
 question.**explaination** = cursor.getString(cursor.getColumnIndex("explaination"));  
 question.setSelectedAnswer(null);  
 question.**id** = cursor.getInt(cursor.getColumnIndex("ID"));  
 wrongList.add(question);  
 }  
 }  
 **db**.close();  
 **db**=null;  
 return wrongList;  
  
 }  
 **//更新错题** public boolean updateQuestion(String where,int wrongNum)  
 {  
 if(**db**==null||!**db**.isOpen()){  
 **db** = SQLiteDatabase.openDatabase(**context**.getDatabasePath(**dataName**).getPath(),null,SQLiteDatabase.***OPEN\_READWRITE***);  
 }  
 ContentValues contentValues = new ContentValues();  
 contentValues.put("isWrong",wrongNum);  
 int rows=**db**.update(**TableName**,contentValues,"answerA=?",new String[]{where});  
 if (rows!=-1)  
 return true;  
 return false;  
 }  
 }

6、Question:

package entity;  
  
import androidx.annotation.NonNull;  
  
import java.io.Serializable;  
**//存储问题的形式**public class Question implements Serializable {  
 public int **id**;  
 public String **question**;  
 public String **answerA**;  
 public String **answerB**;  
 public String **answerC**;  
 public String **answerD**;  
 public String **rightAnswer**;  
 public String **selectedAnswer** =null;  
 public String **explaination**;  
 public int **isWrong**;  
 public int **isFavorite**;  
  
 public int getId() {  
 return **id**;  
 }  
 public void setId(int id) {  
 this.**id** = id;  
 }  
 public String getQuestion() {  
 return **question**;  
 }  
 public void setQuestion(String question) {  
 this.**question** = question;  
 }  
 public String getAnswerA() {  
 return **answerA**;  
 }  
 public void setAnswerA(String answerA) {  
 this.**answerA** = answerA;  
 }  
 public String getAnswerB() {  
 return **answerB**;  
 }  
 public void setAnswerB(String answerB) {  
 this.**answerB** = answerB;  
 }  
 public String getAnswerC() {  
 return **answerC**;  
 }  
 public void setAnswerC(String answerC) {  
 this.**answerC** = answerC;  
 }  
 public String getAnswerD() {  
 return **answerD**;  
 }  
 public void setAnswerD(String answerD) {  
 this.**answerD** = answerD;  
 }  
  
 public String getRightAnswer() {  
 return **rightAnswer**;  
 }  
 public void setRightAnswer(String rightAnswer) {  
 this.**rightAnswer** = rightAnswer;  
 }  
 public String getSelectedAnswer() {  
 return **selectedAnswer**;  
 }  
 public void setSelectedAnswer(String selectedAnswer) {  
 this.**selectedAnswer** = selectedAnswer;  
 }  
 public String getExplaination() {  
 return **explaination**;  
 }  
 public void setExplaination(String explaination) {  
 this.**explaination** = explaination;  
 }  
  
 @NonNull  
 @Override  
 public String toString() {  
 return "Question{"  
 +  
 "id=" + **id** +  
 ", question='" + **question** + '\'' +  
 ", answerA='" + **answerA** + '\'' +  
 ", answerB='" + **answerB** + '\'' +  
 ", answerC='" + **answerC** + '\'' +  
 ", answerD='" + **answerD** + '\'' +  
 ", rightAnswer=" + **rightAnswer** +  
 ", selectedAnswer=" + **selectedAnswer** +  
 ", explaination='" + **explaination** + '\'' +  
 ", isWrong=" + **isWrong** +  
 ", isFavorite=" + **isFavorite** +  
 '}';  
 }  
}

7、Score:

package entity;  
  
import androidx.annotation.NonNull;  
  
import java.io.Serializable;  
  
**//表示成绩**public class Score implements Serializable {  
 public int **score**;  
 public int **hard**;  
 public int **grade**;  
 public int **id**;  
  
 @NonNull  
 @Override  
 public String toString() {  
 return super.toString();  
 }  
  
 public int getGrade() {  
 return **grade**;  
 }  
  
 public int getHard() {  
 return **hard**;  
 }  
  
 public int getScore() {  
 return **score**;  
 }  
  
 public void setGrade(int grade) {  
 this.**grade** = grade;  
 }  
  
 public void setHard(int hard) {  
 this.**hard** = hard;  
 }  
  
 public void setScore(int score) {  
 this.**score** = score;  
 }  
  
 public void setId(int id) {  
 this.**id** = id;  
 }  
  
 public int getId() {  
 return **id**;  
 }  
}