同济大学

**《JAVA语言程序设计》**

**项目设计报告**

**项目设计报告题目：**

基于SprintBoot框架的Java语言学习微信小程序

**班级：** 周四56

**姓名：** 鲁翔辰 **学号：** 1952941

**年级：** 19 **专业：** 信息管理与信息系统

**指导教师：** 贺向东

**日期：** 2022 **年** 1 **月** 6 **日**

整体介绍

本项目的开发目的是完成一个基于知识和习题数据的Java语言学习、检验、测试的平台。前端采用了微信小程序官方的WeUI组件库，集成开发环境使用微信开发者工具；后端采用了Springboot+Maven框架，集成开发环境使用Intellig IDEA。鉴于本人的能力范围，工作重点放在整体架构的搭建和主要功能的实现，而非界面设计，部分次要功能由于时间有限最终被剔除。

声明

由于任课教师没有提供此《Java语言程序设计》的电子版本，且其中大量章节无法在网络上找到相关内容。作为一名程序员，本人认为手动录入知识点数据显然并不合理，也与对Java和程序设计技术的学习无关，因此本人没有完成这项工作，仅录入前几个知识点的数据演示效果。

技术特点

1、由于要求使用《Java语言程序设计》中的知识点，观察后可发现此书的知识点为三层结构，因此在本小程序中也采用三层结构存储与展示知识点：章节-知识块-知识点。如书中有第1章 Java语言概述-1.3 Java程序及其实现-1.3.1 Java Application程序举例。对于知识块本身的内容，处理方式是将其当做第0个知识点。

2、本项目程序拥有高内聚、低耦合的优秀架构，各模块之间基本不存在依赖关系，可以分别独立更新与维护，添加新的功能一般也不会影响到现有的功能。

3、选择题允许有多个选项（2-8），填空题可以有多个空格。但由于填空题的架构原因，难以方便地提供添加功能，所以没有做这个功能。

4、选择题和判断题的选项在生成习题集时会随机排序，但这一排序不会被存储在数据库中，做题记录也不会标注做题时的排序。

5、数据库符合第三范式。

6、本程序均考虑了章节、知识块、知识点、习题等对象的id出现空缺的情况，通过索引主键与具体内容在数据库中的分离来实现，具有较强的容错性。

7、没有完全复用选择题和判断题。

8、由于微信小程序的wx:request方法是异步的，因此不可以在渲染页面时“先后依赖”地请求数据。譬如，module页面加载知识块嵌套知识点的列表时，不能使用先通过一个request请求后端从数据库中查询本章节所有的数据块，成功收到返回值后再用一个循环分别request查询每个数据块中的所有数据点的方法，这可能会导致页面渲染错误以及先后顺序颠倒的问题（因为数据是逐个concat到list中的，而无法保证异步请求按照发出时的顺序依次返回）。所以，本程序采取的解决方法是将存在这种“依赖”关系的数据在后端打包成三维数组或嵌套泛型的格式，前端通过一次请求就可以获得所有数据，弊病就是后端代码可能十分冗长复杂，一定程度上对后续的维护会产生不良影响。

Java后端架构

本项目的Java后端由三个Package构成：background包含一个数据库操作类和三个服务器类，classes包含习题和习题集的模板类，controller包含Springboot框架的controller类。

background包中：

DBOperator类封装了JDBC对SQL Server数据库的query和update方法，减少了其他类操作数据库时的代码量。不将其设为静态类而需创建实例的原因是为了避免循环中操作数据库时反复连接。

StaticServer类包含了大部分的静态方法，主要作为前端与数据库的中介，负责数据的提取与存放等。

UserServer类只有一个login方法，需为每一个在线用户创建一个实例，可作为用户登录后的数据缓存区，但目前未添加相应方法。

ExerciseServer类的对象也是与每个在线用户一一对应，在登录时创建，作为习题生成后的数据缓存区，同时也负责前端与数据库的交互。用户进入章节测验时，ExerciseServer就会生成一个对应的ExerciseSet存放在自己的数组中，之后所有的提取习题、用户提交后批改和将做题记录录入数据库的工作都由前端通过TestController类调用ExerciseServer的方法操作该ExerciseSet完成。

classes包中：

Exercise类是所有习题模板类的父类，也是一个抽象类，提供了各习题类型的方法头，其中init方法为真抽象方法，用多态特性实现了习题对象的初始化。

cjExercise类是Exercise类的子类，选择题类ChoiceExercise和判断题类JudgmentExercise的父类，也是一个抽象类，包含选择题和判断题共有的一些变量和方法，如实例化时是否随机排列选项ifRandomize等。

ChoiceExercise类和BlankExercise类是cjExercise类的子类，分别是选择题和判断题的模板类，用于存储生成测试题目集时从数据库中随机抽取的单题数据，同时提供诸多get和set方法便于ExerciseSet类对其进行操作。

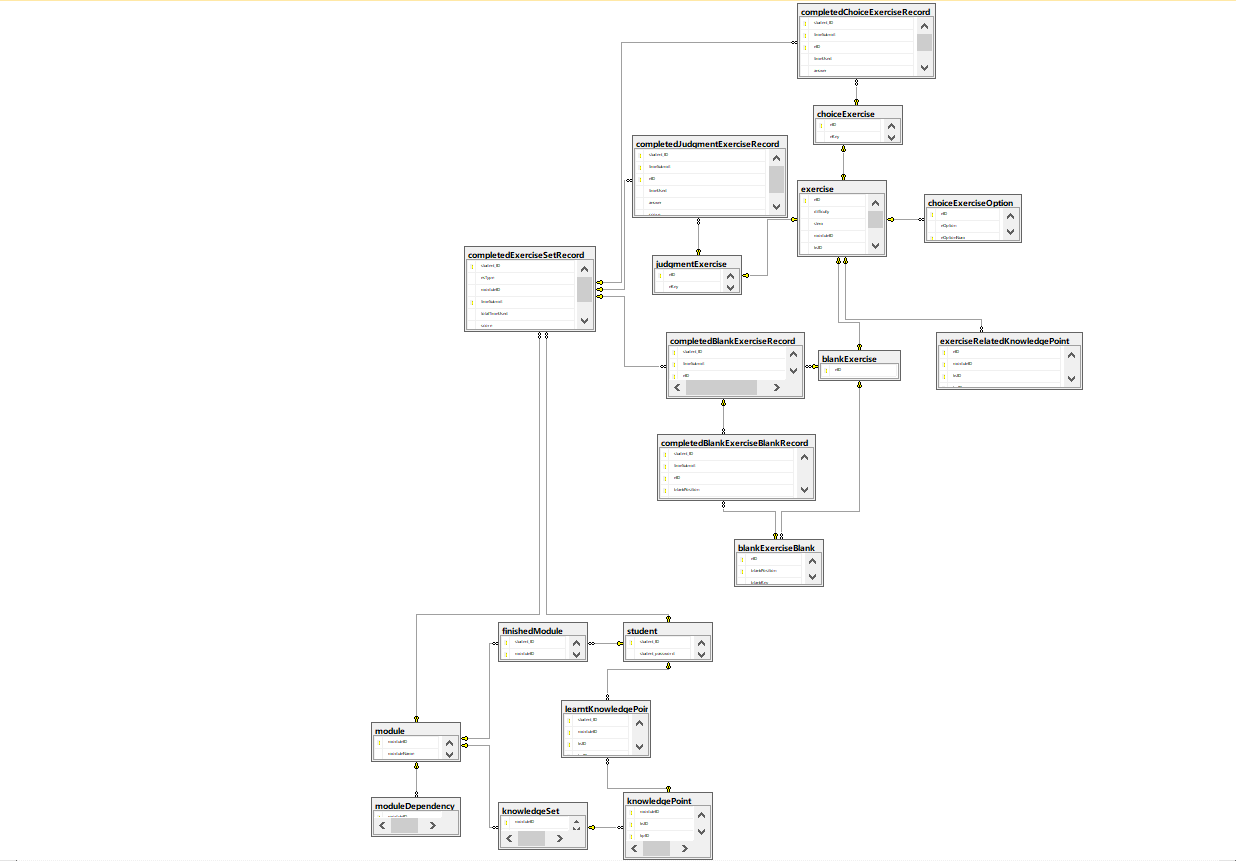
BlankExercise类是Exercise类的子类，是填空题的模板类，可以在其初始化时将数据库中的题干根据空格位置插入空格生成新的题干。

ExericseSet类是习题集，用于缓存一次测验的所有相关数据，有exercise数组存储题目，对其多态地调用子类构造函数来初始化，并有大量方法用于前端与习题数据的交互，此处略。

controller包中：

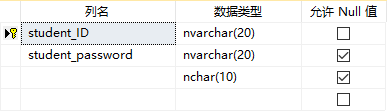
TestController类被注解为Spring MVC的RestController处理器，用于接受微信小程序前端发送的请求，并调用后端的方法实现相应的功能。

数据库架构



数据库中共有19个表，分别用于存储学生账号数据、章节数据及用户完成情况和依赖关系、知识点数据与用户完成情况、习题数据、做题记录数据等。由于本程序允许选择题有多个选项、填空题有多个空格，因此必须为它们的选项和空格专门建表，也需为填空题建答题记录表。其中，字符串类型的字段统一使用nvarchar类型。数据库还有一个触发器，用于避免在exerciseRelatedKnowledgePoint中插入该题目本身的主要知识点（在exercise表中）。

student表



有2个字段，分别存储用户名和密码，每一行代表一个学生用户。

module表：



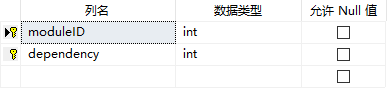
有2个字段，分别存储章节ID和章节名，每一行代表一个章节。

finishedModule表：



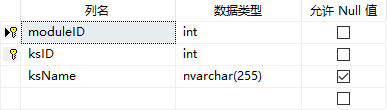
有2个字段，分别存储学生用户名和章节ID，每一行代表某个用户与已通过测验的一个章节的相关关系。

moduleDependency表：



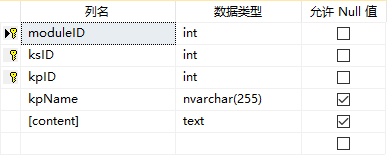
有2个字段，分别存储章节ID和依赖的章节ID，每一行代表某个章节与其依赖的另一个章节的相关关系。

knowledgeSet表：



有3个字段，分别存储章节ID、知识块ID和知识块名称，每一行代表某个章节中的一个知识块。

knowledgePoint表：



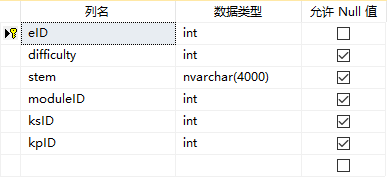
有5个字段，分别存储章节ID、知识块ID、知识点ID、知识点名称和知识点内容，每一行代表某个章节的某个知识块的一个知识点。

learntKnowledgePoint表：



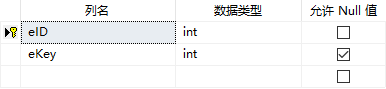
有4个字段，分别存储用户名、章节ID、知识块ID和知识点ID，每一行代表某个用户与其已完成学习的一个知识点的相关关系。

exercise表：



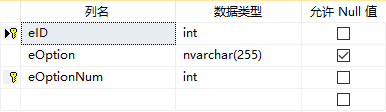
有6个字段，分别存储习题ID、习题难度、题干、主要相关知识点所在章节ID、主要相关知识点所在知识块ID和主要相关知识点ID，每一行代表一道习题，存储的是习题与题目类型无关、各类型习题所共有的数据。

choiceExercise表：



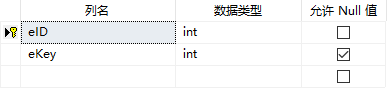
有2个字段，分别存储习题ID和答案，每一行代表一道选择题。

choiceExerciseOption表：



有3个字段，分别存储习题ID、习题选项内容和习题选项编号，每一行代表某一道选择题的一个选项。

judgmentExercise表：



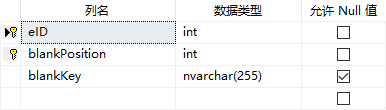
有2个字段，分别存储习题ID和答案，每一行代表一道填空题。

blankExercise表：



有1个字段，存储习题ID。

blankExerciseBlank表：



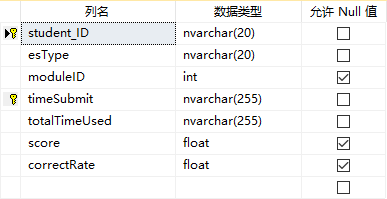
有3个字段，分别存储习题ID、空格位置和空格答案，每一行代表某一道填空题的一个空格。

exerciseRelatedKnowledgePoint表：



有4个字段，分别存储习题ID、相关知识点所在章节ID、相关知识点所在知识块ID和相关知识点ID，每一行代表某一道习题的一个其他相关知识点（非主要相关知识点）。

completedExerciseSetRecord表：



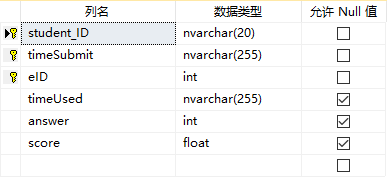
有7个字段，分别存储用户名、测验类型、测验的章节ID、提交时间、总用时、总分和正确率，每一行代表某个学生用户的一次测验的整体结果记录。尽管总分和正确率可以通过其他表中的数据计算得到，此处的数据是冗余的，但由于查询与计算过程十分繁琐，此处将这两个数据直接存入表中。

completedChoiceExerciseRecord表：



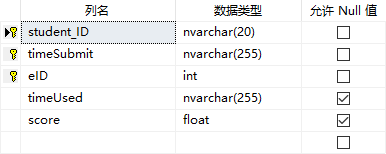
有6个字段，分别存储用户名、提交时间、习题ID、单题用时、用户答案和得分，每一行代表某个学生用户的某次测验的其中一道选择题的做题结果记录。

completedJudgmentExerciseRecord表：



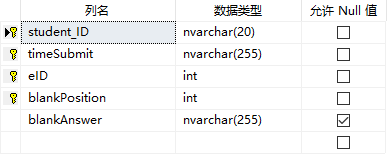
有6个字段，分别存储用户名、提交时间、习题ID、单题用时、用户答案和得分，每一行代表某个学生用户的某次测验的其中一道判断题的做题结果记录。

completedBlankExerciseRecord表：



有5个字段，分别存储用户名、提交时间、习题ID、单题用时和得分，每一行代表某个学生用户的某次测验的其中一道填空题的做题结果记录。

completedBlankExerciseBlankRecord表：



有5个字段，分别存储用户名、提交时间、习题ID、空格位置和用户空格答案，每一行代表某个学生用户的某次测验的某道填空题的一个空格的做题结果记录。

登录与注册





工作需求：

实现用户登录与注册功能。登录时判断用户名是否存在以及密码是否正确，若成功则根据用户名分别进入学生端和教师端，若失败则提示失败原因；注册时判断用户名是否已存在，若成功则注册用户，若失败则提示失败原因。

实现解析：

前端：

登录页面的输入框使用自定义样式，注册页面的输入框使用WeUI的表单结构样式；按钮均使用WeUI的Button样式。

后端：

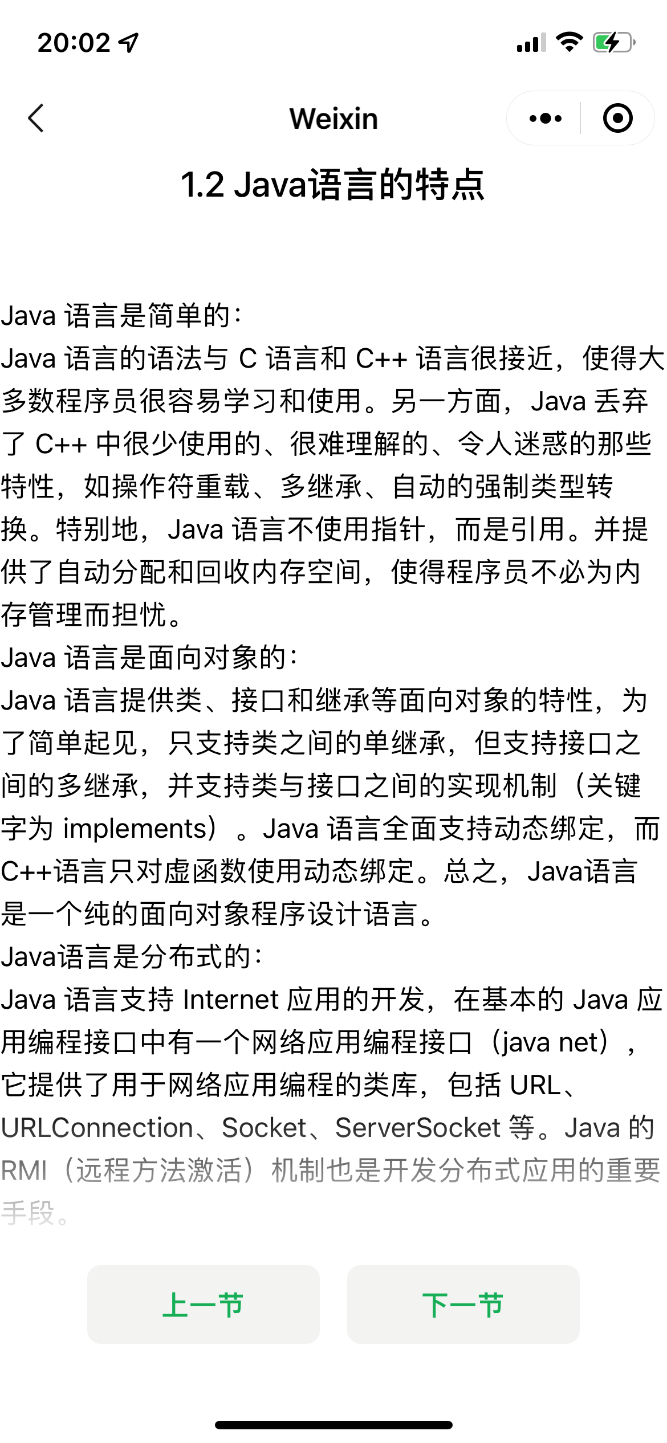
登录时，首先调用StaticServer中的方法获得状态，若UserServers和ExerciseServers中已存在当前登录用户ID的对象，则将其移除，并创建相应的对象。

按章节学习（学生端）









工作需求：

首先，在章节列表界面展示所有的章节，并标注为已完成、待学习和未解锁三种状态，用户可以进入已完成的章节和待学习的章节，但不能进入未解锁的章节，并会向用户提示章节的依赖关系。

进入章节后，按知识块-知识点的层级关系展示章节中的所有知识点，并可以自由进入任何一个知识点学习。知识块可以折叠/展开，已完成学习的知识点显示为绿色，完成的知识块右侧打钩。最下方提供进入章节测试的按钮，若本章的知识点尚未全部学完，则会提示，但仍可进入测试。

进入知识点后，上方展示知识点名称，中间为知识点内容，可以上下滚动，底部有三个悬浮按钮，分别用于上下翻页和用户完成学习（完成学习后该按钮即消失）。

实现解析：

前端：

moduleList页面显示时，向后端发送三个请求，分别用于获取已完成章节、待学习章节和未解锁章节的列表（包含章节ID和章节名），使用WeUI的“带说明、跳转的列表项”样式，通过wx:for循环分别循环渲染三个列表。用户单击未解锁章节时，使用WeUI的Dialog样式弹出警告对话框。

module页面显示时，向后端发送请求getKnowledgeSetsAndPoints，一次性获取知识块-知识点的所有数据，通过wx:for二重循环渲染列表。kindToggle函数用于知识块的折叠/展开。用户点击本章测试时，若尚未完成所有知识点，同样使用WeUI的Dialog样式弹出警告。

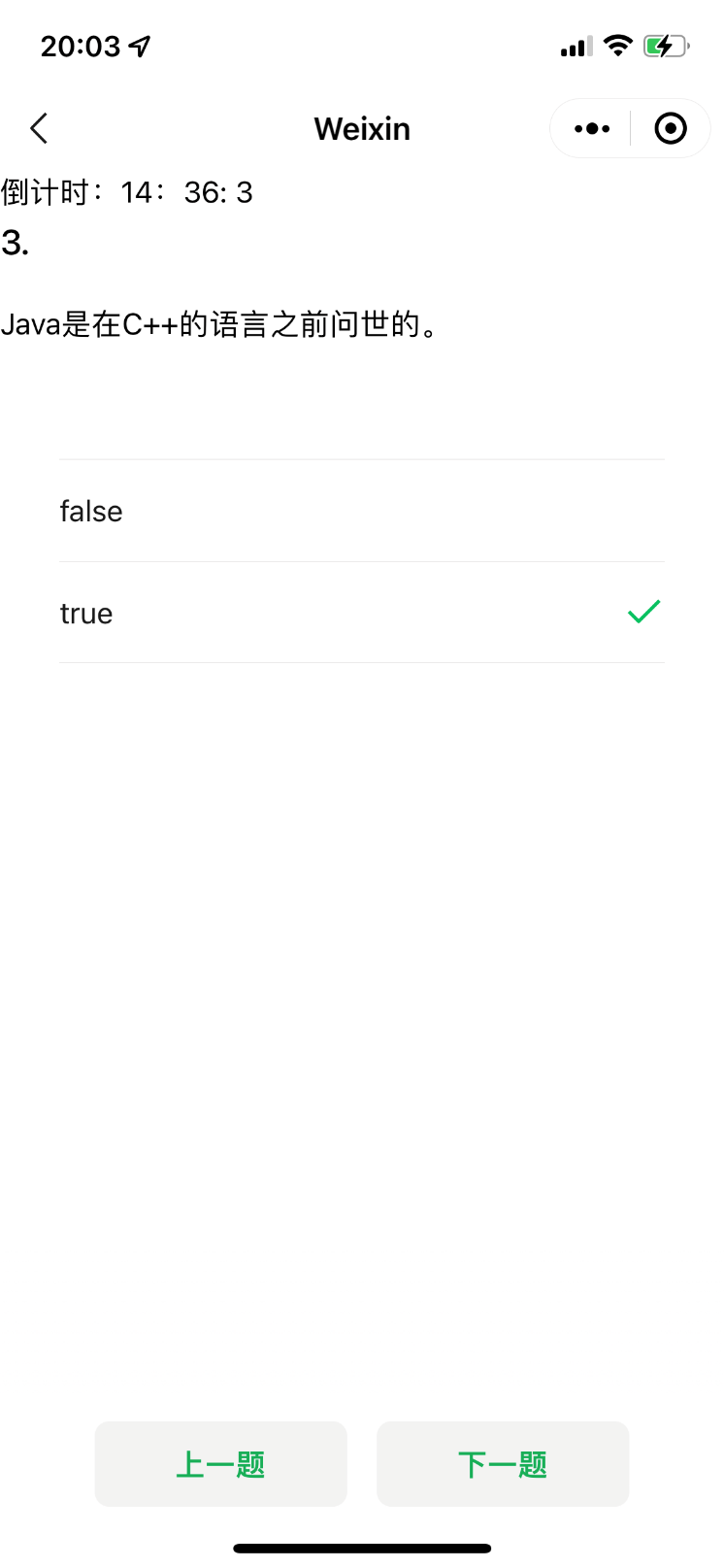
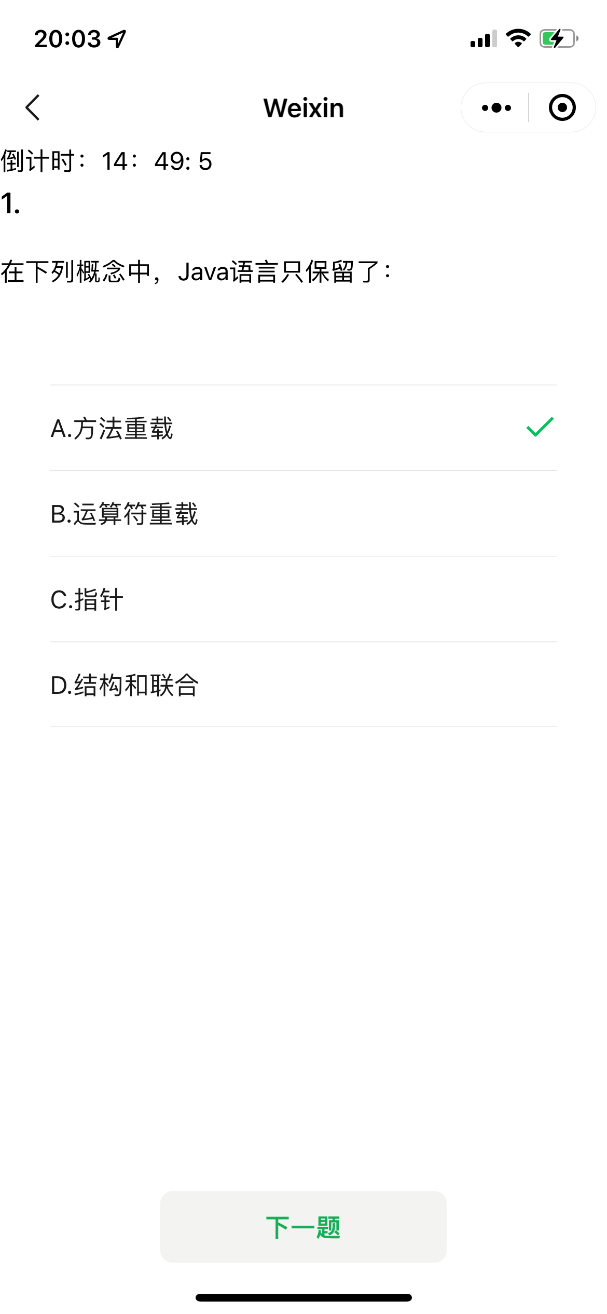
knowledgePoint页面显示时，向后端发送请求getMinMaxkp，获取当前章节的最大最小知识块编号和当前知识块的最大最小知识点编号；发送请求getKnowledgePoint，获取知识点的名称和内容。

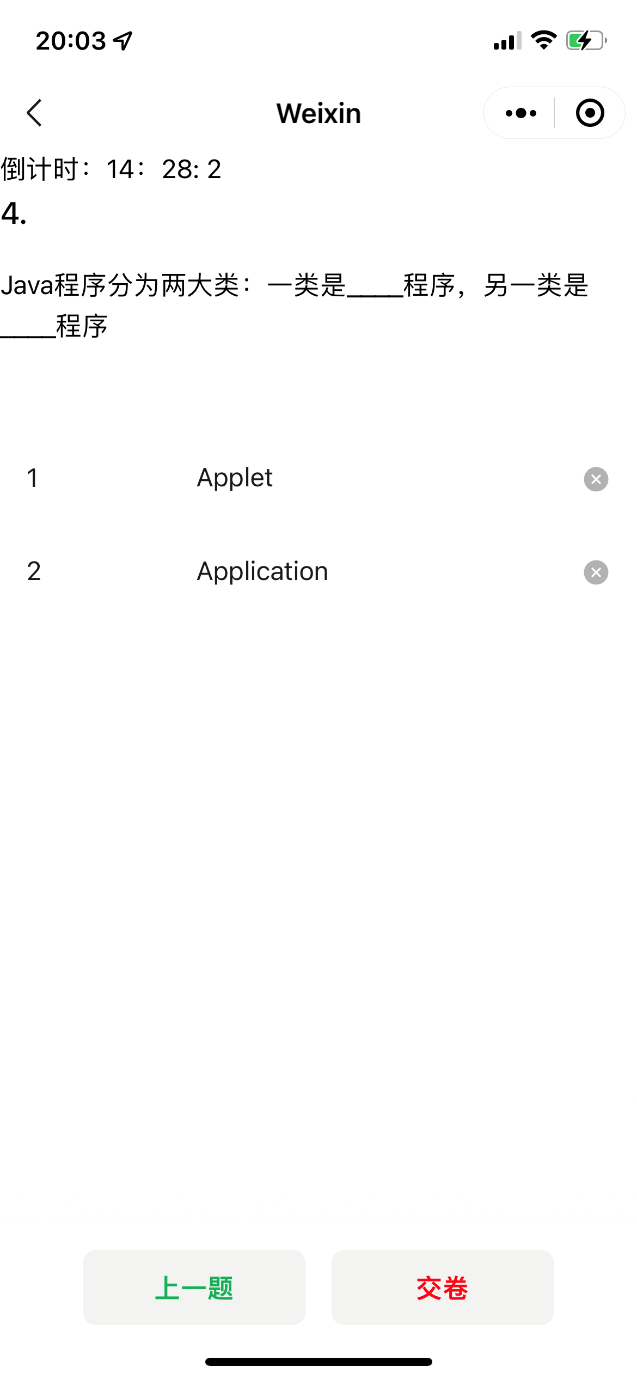
后端：

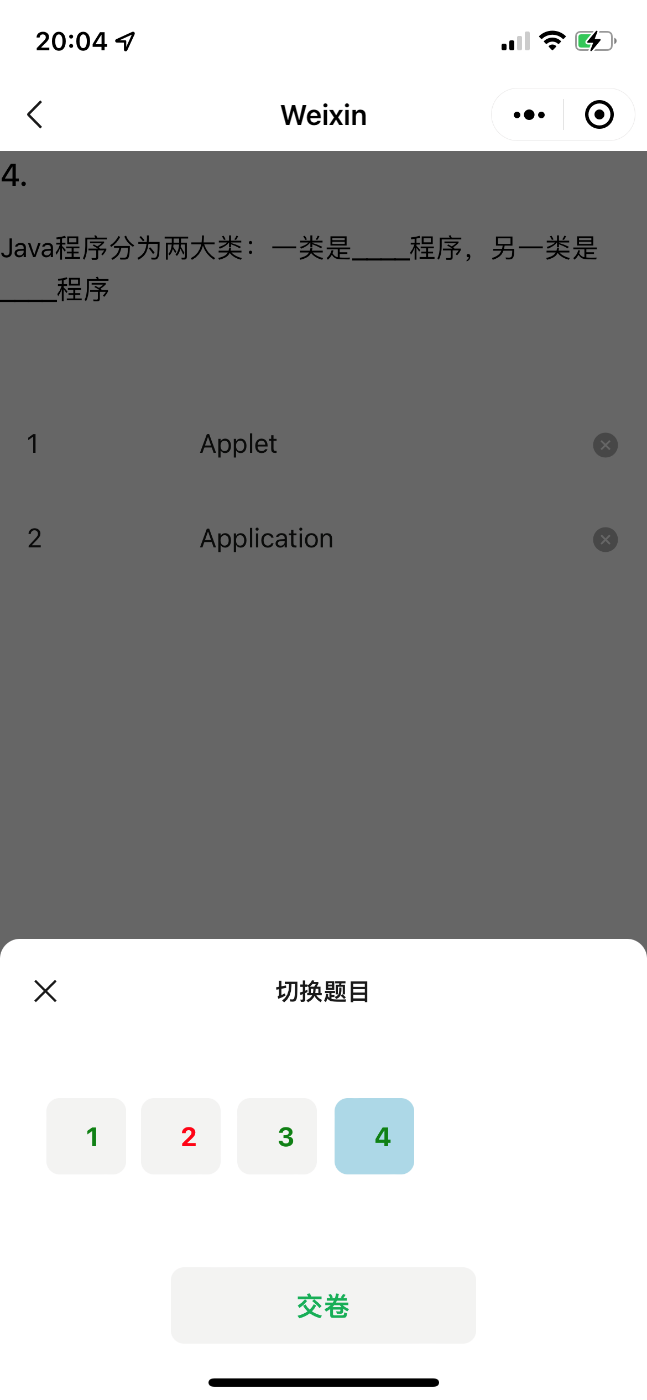
本模块的前后端交互通过StaticServer和ExerciseServer完成。获取章节列表的三个方法在StaticServer中，直接查询数据库获取数据，存入String数组返回。getKnowledgeSetsAndPoints方法在ExerciseServer中，查询数据库获取知识块-知识点，并根据用户ID查询数据库获取已学过的知识点等数据，整合成一个HashMap<String,Stirng[]>>类型的ArrayList返回。其他方法同样在StaticServer中，原理较简单，此处不再赘述。

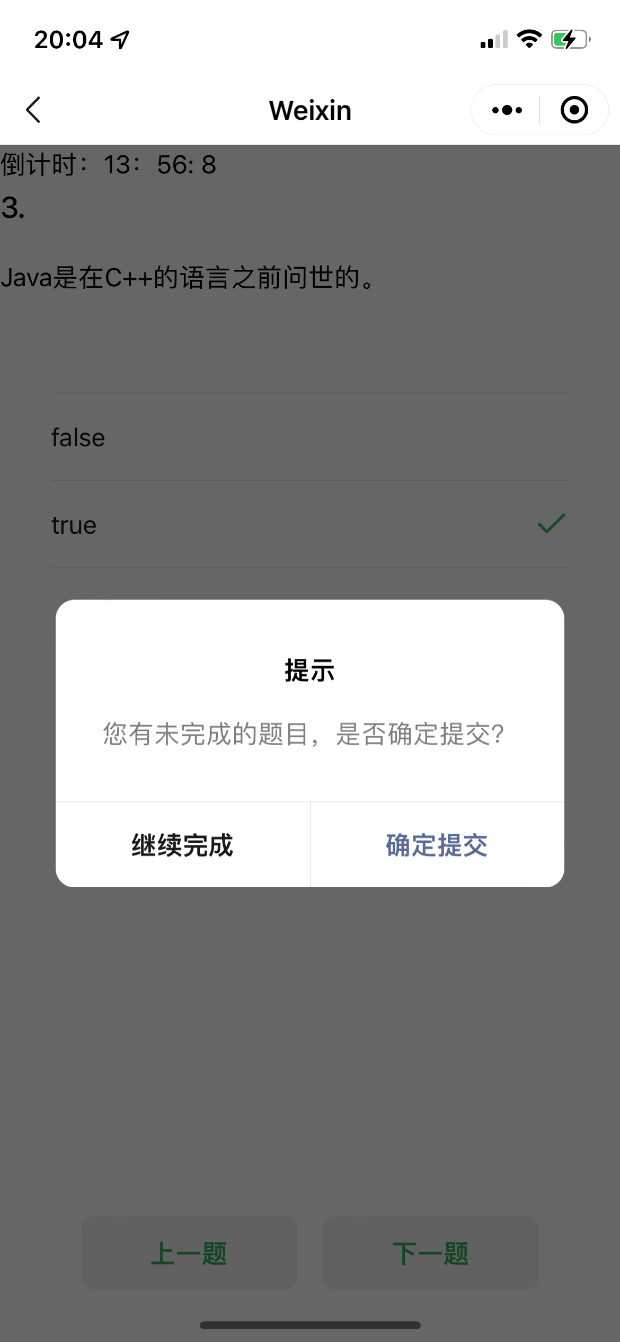
章节测验（学生端）













工作需求：

对于每一个章节，提供章节测验功能，通过（达到90分以后）则当前章节变为已完成，同时根据依赖关系解锁后续章节。为立即显示结果，测验题型仅有客观题，包括选择题、判断题和填空题。选择题允许有2-8个选项，填空题空格数量不限，且选择题和判断题展示题目时选项随机排列。除了通过按钮翻页，在移动端还可以通过上拉触底调出快速切换功能，此功能内可显示已完成的题目和未完成的题目。测验有时间限制，时间到自动交卷。交卷时会记录测验的总用时和每道题的用时（停留在该题页面上的总时长）。交卷后计算出分数，展示学习评价，提示后续操作，并提供测验回顾功能。

实现解析：

前端：

一次测验的题目共用一个exercise页面，通过请求后端更新数据实现翻页。从exerciseAccess页面进入exercise页面前，先调用后端的generateExercise方法生成题目集，并通过options将章节ID作为参数传给exercise。exercise加载时，首先将exerciseNum（当前题目编号）赋值为0，并向后端发送请求获得此测验题集的题目数量，再调用renewExercise更新页面和开始计时。renewExercise需要完成请求题目数据并根据题目类型排列选项的功能，代码逻辑较复杂。交卷时，请求后端的submitModuleExerciseSet方法，将数组转化为JSON格式发送给后端。测验回顾功能即查询数据库中当前用户与时间戳的做题记录并跳转至做题记录页面。

后端：

生成题目集时，从数据库中主要相关知识点属于本章节的所有习题里面随机抽题，并通过算法分别随机设定选择题、判断题和填空题的数量。随机抽题的方法是全组合，而非全排列。

获取题目数据的方法是ExerciseServer的getExercise，传递题目编号为参数，根据章节ID来获取题目集中该编号的题目。submitModuleExerciseSet方法调用了exerciseServer的correctAndRecordExercises方法，根据用户的答案进行批改并录入数据库，因流程较长，此处不详细阐述。

做题记录（学生端）









工作需求：

章节测验模块已实现在用户完成每一次测验后，记录用户的提交时间、做题总用时和每一题的用时、总分和每一题的得分以及正确率等数据。做题记录模块则向用户提供查询每一次测验的功能，展示最近的30次测验，上方有搜索栏，可诉诸数据查询。选择题和判断题将正确选项标绿，若做错则将用户的选项标红；填空题展示用户的答案，若正确则标绿，若错误则标红并展示正确答案。同时，每道题下方提供查询相关知识点的功能，用户可以直接在做题记录界面浏览题目的主要相关知识点和其他相关知识点。

实现解析：

前端：

搜索栏用wxs语法的字符串匹配算法将用户搜索内容与章节名、测验类型、提交时间和成绩做比较并返回匹配成功的结果。

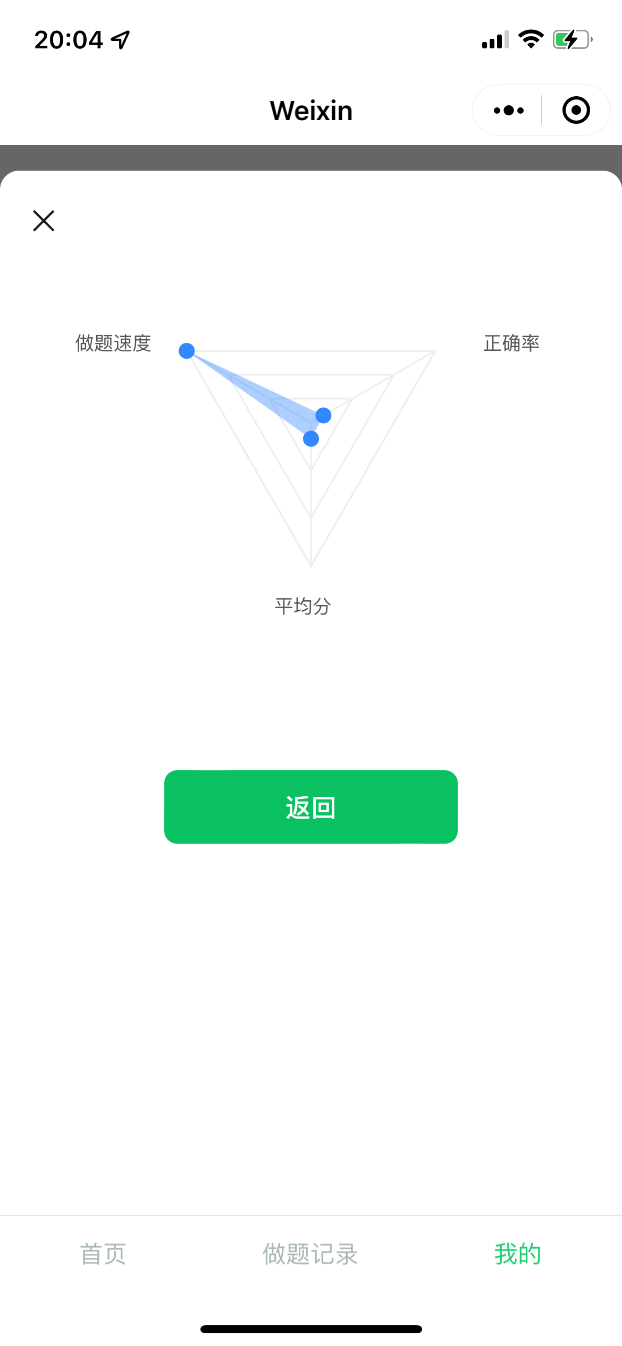
exerciseReview页面中，相关知识点功能采用WeUI的Half-screen Dialog样式两层嵌套，其他主要是与后端的数据交互，方法较为常规。

后端：

做题记录功能的前后端数据交互主要使用HashMap类型，方法较多，代码冗长，此处不再赘述。

个人主页（学生端）





工作需求：

为用户提供查询已完成章节数、已学习知识点数、测验次数、平均分、平均正确率、平均用时等生涯数据的功能，并制作雷达图，提供退出登录的功能。

实现解析：

前端：

上方列表展示数据，雷达图在微信小程序的canvas上绘制。

后端：

常规数据交互，略

推荐学习（学生端）

工作需求：

随机推送一个用户未完成学习的章节中未学过的一个知识点，提供“换一个”功能。

实现解析：

前端：

与按章节学习中的知识点界面相同。

后端：

常规数据交互，略

添加习题（管理员端）



工作需求：

为便于后续维护和扩充习题库，为管理员提供添加习题的功能，首先选择习题对应的章节、知识块、知识点（即其主要相关知识点的索引），然后选择习题类型，根据不同的习题类型（填空题因过于繁琐未提供）分别选定题目的各项属性，选择题可设定选项数量等，完成后单击添加按钮即可加入数据库。

实现解析：

前端：

选择章节、知识块和知识点的部分为三个互相关联的picker选择器，为确保无bug，逻辑较为复杂，详见代码，此处不详细解释。选项数量和正确选项也是两个picker，并根据选项数量的值动态渲染列表，提供输入各选项内容的输入框。管理员用户点击添加按钮后根据题目类型分别调用不同的后端方法传送Map。

后端：

此部分的主要工作在前端完成，前端发送打包成Map的数据，在后端解包并录入数据库。

小结及回顾

经过两周的之所以选择开发微信小程序，是因为本人对这一平台较有兴趣，之前也曾有过接触微信小程序开发的经历，希望在完成此次Java大作业的过程中能进一步学习相关的知识，并且通过做出一个前后端各部分较完整的项目熟悉实际业务的开发流程。除此之外，由于微信小程序与后端的交互有唯一且封装好的wx:request方法，因此就无需学习和选择大量繁琐的Java前后端交互技术。

尽管没有难以克服的困难，但在整个开发过程中还是遇到不少问题，例如微信小程序如何真机测试、前后端的数据交互采用何种格式以及数据库如何设计等。本人采用的是自顶向下的开发方式，首先用两天的时间完整地构思了知识库、习题库的架构，然后再着手进行开发，这就避免了过程中大幅度调整架构的情况，事实上也不可能有时间这么做。

通过完成此次Java大作业，本人主要学了Java的面向对象开发技术、Java后端与数据库的交互技术、微信小程序前端与Java后端的交互技术以及微信小程序的前端开发技术等。更重要的是，本人对如何将这些前后端技术结合起来构建一个完整项目有了较清晰的了解，这对本人今后作为一名程序员的职业生涯一定会带来很大的帮助。