****

**宿舍管理系统**

模型分析

小组成员： 雷泓 1851202 王文政1853928

肖博阳1852672 张明哲1852452

指导教师： 孙萍

专 业： 软件工程

同济大学软件学院

School of Software Engineering, Tongji University

目录

[一、简介 4](#_Toc41768815)

[1.1 项目简介 4](#_Toc41768816)

[1.2 项目进度 4](#_Toc41768817)

[1.3 重定义部分 5](#_Toc41768818)

[二、架构分析 5](#_Toc41768819)

[2.1 逻辑架构分析 5](#_Toc41768820)

[2.2 物理架构分析 6](#_Toc41768821)

[三、分析机制 6](#_Toc41768822)

[四、模型分析 6](#_Toc41768823)

[4.1 类与用例实现 7](#_Toc41768824)

[4.2 类图 8](#_Toc41768825)

[4.2.1学生信息系统 8](#_Toc41768826)

[4.2.2学生管理系统 10](#_Toc41768827)

[4.2.3 宿舍设施报修 12](#_Toc41768828)

[4.2.4宿舍自动贩卖机货物整理 14](#_Toc41768829)

[4.2.5财务统计系统 16](#_Toc41768830)

[4.2.6宿舍资产管理 17](#_Toc41768831)

[4.2.7 宿舍成绩系统 19](#_Toc41768832)

[4.2.8 数据库管理系统 20](#_Toc41768833)

[4.2.9 登录 20](#_Toc41768834)

[4.2.10 注册 21](#_Toc41768835)

[4.2.11 寝室费用系统 22](#_Toc41768836)

[4.2.12 功能室申请 23](#_Toc41768837)

[4.2.13 快递、快件查询 25](#_Toc41768838)

[4.2.14 申请更换宿舍 26](#_Toc41768839)

[4.3 序列图 27](#_Toc41768840)

[4.3.1 信息更新与检索 27](#_Toc41768844)

[4.3.2 行为检测 28](#_Toc41768845)

[4.3.3 学生注册 28](#_Toc41768846)

[4.3.4 缴纳寝室费用 29](#_Toc41768847)

[4.3.5 宿舍设施检查 30](#_Toc41768848)

[4.4 通信图 31](#_Toc41768849)

[4.4.1 缴纳寝室费用 31](#_Toc41768850)

[4.4.2 功能室申请 32](#_Toc41768851)

[五、参考资料与说明 33](#_Toc41768852)

[5.1参考书目 33](#_Toc41768853)

[5.2 参考文章及期刊 33](#_Toc41768854)

[六、组员分工 34](#_Toc41768855)

# 一、简介

## 1.1 项目简介

技术迅速发展的今天，为了方便工作生活各式各样的数据被保存下来以供之后的安排和统计。数据膨胀的时代，逼迫着高校以更加有效率和简便的方式来管理和组织这些数据。本系统以常见的高校宿舍管理系统为框架进行补充和改善，实现对宿舍人员和设备的有效管理。

本系统基于宿舍系统网络的架构，为学生、宿舍管理员和生活老师等提供便捷的服务，达到更高效的管理。主要功能有为宿舍学生提供查询及缴纳寝室费用，查询学生信息等常用服务，为宿舍管理员及生活老师提供宿舍信息查询、宿舍内设施检查保修以及查看设施信息服务，同时为数据库管理员提供清晰的接口以维护和更新数据库。

本系统分为7个子系统：登陆子系统；学生服务子系统；学生信息子系统；设施信息子系统；财务子系统；寝室成绩子系统；数据库管理子系统。

登陆子系统主要参与者为用户，该系统为数据库中已有用户提供登陆服务，验证其身份给予其使用服务权力，为未来用户提供账户注册服务。

学生服务子系统主要参与者为学生，该系统为学生提供一系列如查询缴纳费用、查看公告及申请更换宿舍等常见宿舍服务。

学生信息子系统主要参与者为宿舍管理员和生活老师，该系统提供了查询宿舍各方面信息、发布公告及学生晚归登记等一系列学生信息管理和登记服务，同时提供发布宿舍公告服务。

设施信息子系统主要参与者为宿舍管理员和生活老师，该系统提供了查询宿舍内各设施状态、功能室借用以及备用钥匙借用等宿舍内设施信息管理服务。

财务子系统主要参与者为财务人员，该系统提供了宿舍内总体开销费用和宿舍资产登记服务。

寝室成绩子系统主要参与者为学生、宿舍管理员和生活老师，该系统为学生和生活老师提供查询宿舍得分情况服务；为宿舍管理员提供各宿舍违规登记，卫生情况登记以及录入成绩等管理功能。

数据库管理子系统主要参与者为数据库管理员，该系统提供了系统配置功能，允许数据库管理员对数据库维护和更新，并进行异常行为检测等活动。

整个宿舍管理系统基本涵盖了学生及老师管理员在管理宿舍时需要进行的各项活动，在真实使用中能够方便使用者。如在网络上可以直接填写申请更换宿舍表，省去大部分与管理员和生活老师的对接工作，而管理员也只需查看系统中的信息，再与老师简单沟通即可完成。且通过数据库的管理和整合，学生和宿舍内各样的数据都能够方便快捷的取出进行统计。

## 1.2 项目进度

之前的文档已完成了项目的基本规划、用例图及用例规约、活动图、术语表 和补充规约等。本文档将进行项目的架构分析，并给出类图、部分时序图和协作图、分析机制等，还有本系统的功能和特点及其操作必须遵守的约束条件。本文档供系统的开发者和利益相关者阅读。

## 1.3 重定义部分

学生之前在缴纳寝室费用时，可以在一定时间内继续尝试支付未成功支付的费用，现在改为每次支付失败后都可直接跳转回主界面，而不是等待10分钟，且系统不记录缴费失败及尝试缴费记录；学生需要从缴费界面重新选择未缴费的项目完成缴费流程。缴费失败指第三方支付平台API传回支付失败信息，这些平台可能允许用户多次尝试支付，但系统并不关心用户具体在该平台上进行尝试的次数。

学生现在使用“学生服务系统”中的操作时，系统不再每次都检测学生是否已缴纳寝室费用，即“且学生已缴纳寝室费用”不再成为前置条件。现在改为“学生已注册并登录系统，系统认证学生身份”，避免了因突发情况无法缴纳费用而使用不了学生服务系统的可能情况。学生仍可以直接通过学生服务系统来缴纳费用，但是否缴纳费用与能否使用系统服务已不直接挂钩。

# 二、架构分析

## 2.1 逻辑架构分析

本次宿舍管理系统采取常用的软件架构——分层模式为基础，分为用户界面层，业务服务层，中间件层以及数据库层；

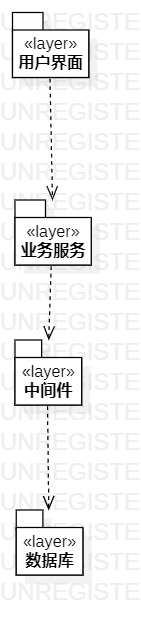
**·用户界面层**：用户界面，负责视觉和用户互动

**·业务服务层**：实现业务逻辑

**·中间件层**：为业务服务提供数据等中间件

**·数据库层**：保存数据

这种架构将软件分成若干个水平层，每一层都有清晰的角色和分工，不需要知道其他层的细节。每一层都能够独立测试，其他层的接口通过模拟进行解决，使得整个系统具有高内聚低耦合特点。



## 2.2 物理架构分析

暂无物理架构考虑

# 三、分析机制

四、模型分析

## 4.1 类与用例实现

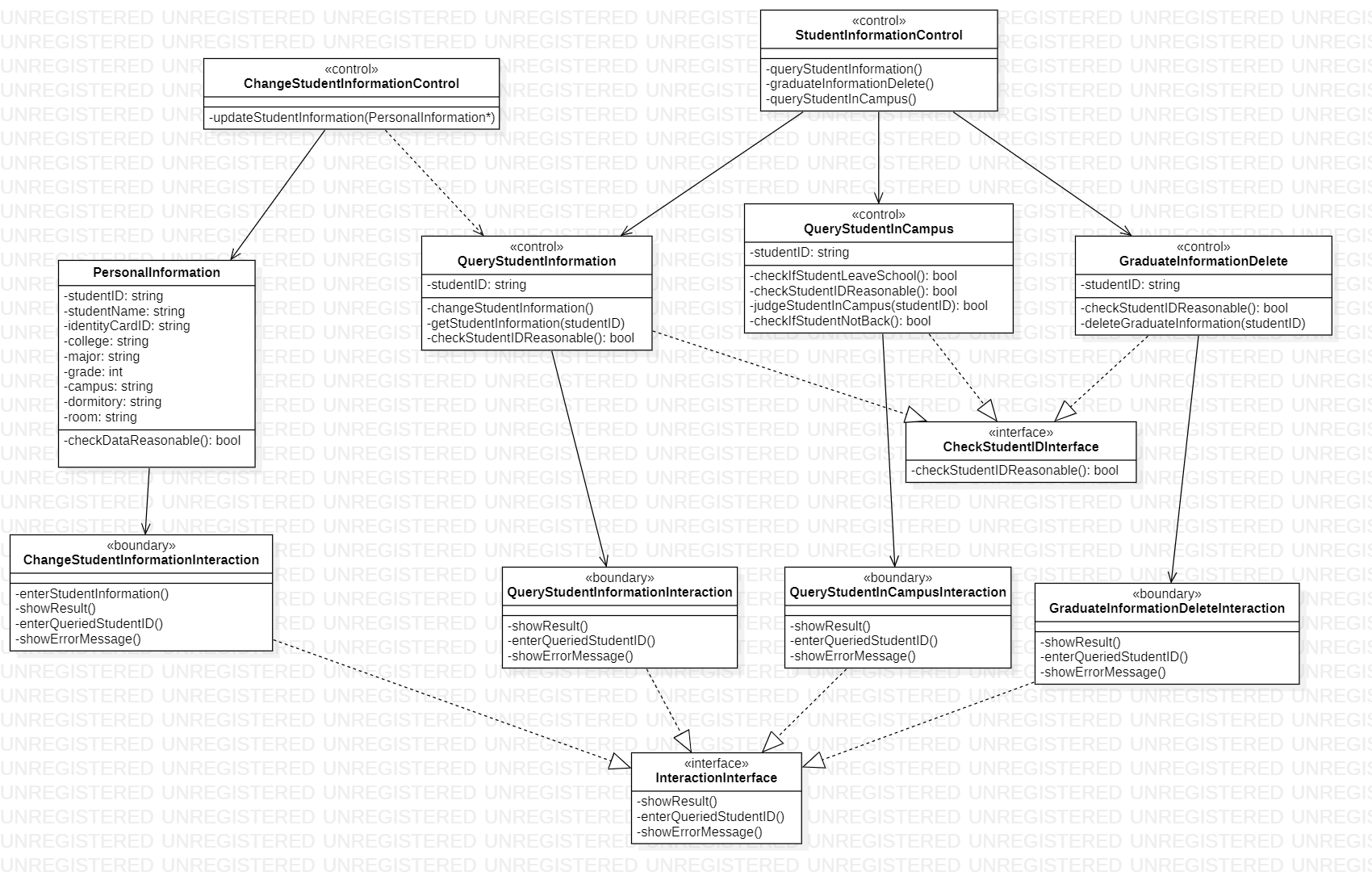
本部分将用表格列出所有用例及其实现时用到的类。

|  |  |
| --- | --- |
| 注册 | Register |
| 登录 | Login |
| 学生信息系统 | StudentsInfoSystem |
| 学生管理系统 | StudentsManageSystem |
| 宿舍设施报修 | FacilitiesRepairsController |
| 宿舍自动贩卖机货物整理 | VendingMachineManagement |
| 财务统计系统 | FinancialStatisticsController |
| 宿舍资产管理 | AssetManagement |
| 宿舍成绩系统 | DormitoryPointController |
| 数据库管理系统 | DatabaseProcess |
| 寝室费用系统 | DormitoryPaymentController |
| 功能室申请 | FunctionRoomApplicationController |
| 快递、快件查询 | MailResearchController |
| 申请更换宿舍 | DormitoryAlterationsController |

## 4.2 类图

这一部分将给出一些类图。由于用例与类之间不可避免的耦合，每个类图及其标题可能 对应**多个用例或一个子系统及其中的所有用例**，具体将在补充说明中解释。补充说明还包括 了对这个类的解释及类的成员属性和方法（即函数）的说明。

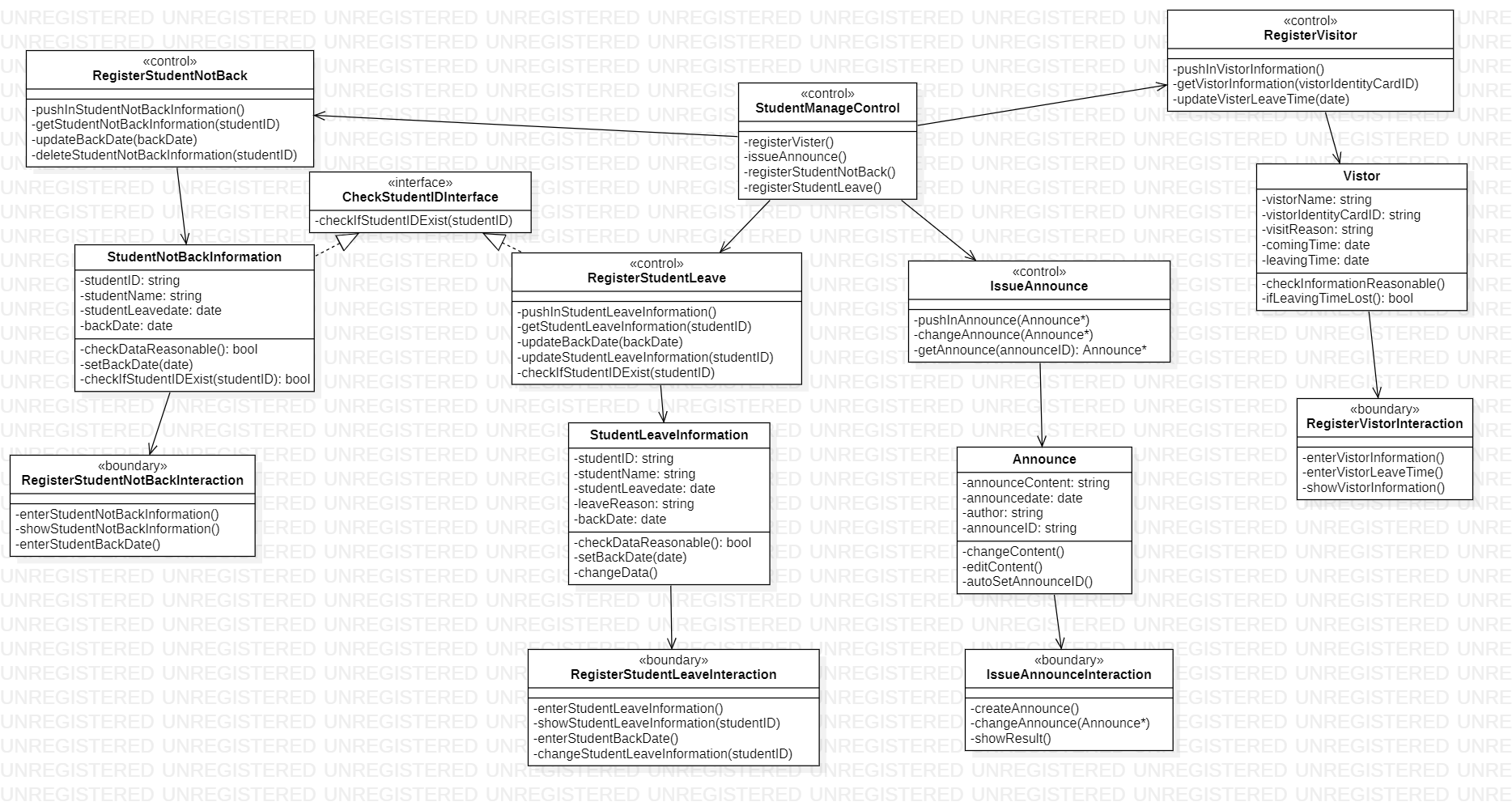
### 4.2.1学生信息系统



本图说明了学生信息系统中的四个用例，包括毕业生信息删除、学生在校情况查询、宿舍人员信息查询及其扩展用例宿舍人员信息变更。管理人员可以在该页面使用毕业生信息删除、学生在校情况查询、宿舍人员信息查询功能，并且可以在宿舍人员信息查询中使用宿舍人员信息变更功能。学生信息系统控制类StudentInformationControl用于控制各个功能的使用，并使用控制类QueryStudentInformation、QueryStudentInCampus、GraduateInformationDelete、ChangeStudentInformationControl来具体控制各个子功能的具体操作。

|  |  |
| --- | --- |
| 方法 | 说明 |
| StudentInformationControl.queryStudentInformation() | 调用查询学生信息功能 |
| StudentInformationControl. graduateInformationDelete() | 调用毕业生信息删除功能 |
| StudentInformationControl. queryStudentInCampus() | 调用查询学生在校情况功能 |
| QueryStudentInformation. checkStudentIDReasonable(): bool | 判断被查询的学生ID是否存在 |
| QueryStudentInformation. changeStudentInformation() | 调用更改学生信息功能 |
| QueryStudentInformation. getStudentInformation(in studentID) | 获取参数作为学号的学生个人信息 |
| ChangeStudentInformationControl.updateStudentInformation(in PersonalInformation\*) | 将参数中的学生个人信息类对象中的信息更新入数据库中 |
| PersonalInformation. checkDataReasonable(): bool | 根据现实情况及学校规定，查看对象中保存的学生个人信息是否符合要求或实际情况 |
| ChangeStudentInformationInteraction. enterStudentInformation() | 要求管理员输入新的学生个人信息 |
| ChangeStudentInformationInteraction. enterQueriedStudentID() | 要求管理员输入要更改的学生ID |
| ChangeStudentInformationInteraction. showResult() | 根据程序运行的结果，将程序执行成功或失败告知管理员。如果成功则显示更改后的结果，失败则显示错误信息 |
| ChangeStudentInformationInteraction. showErrorMessage() | 程序执行失败时，显示错误信息以用于纠错 |
| QueryStudentInformationInteraction. showResult() | 根据程序运行的结果，将程序执行成功或失败告知管理员。如果成功则显示被查询学生的个人信息，失败则显示错误信息 |
| QueryStudentInformationInteraction. enterQueriedStudentID() | 要求管理员输入要查询信息的学生的ID |
| QueryStudentInformationInteraction. showErrorMessage() | 程序执行失败时，显示错误信息以用于纠错 |
| QueryStudentInCampus. checkIfStudentNotBack(): bool | 在数据库中查询学生当前是否有晚归不归记录未销 |
| QueryStudentInCampus. checkIfStudentLeaveSchool(): bool | 在数据库中查询学生当前是否在长时间离校状态中（如休学，病假等） |
| QueryStudentInCampus. checkStudentIDReasonable(): bool | 检查输入的学生ID是否合理 |
| QueryStudentInCampus. judgeStudentInCampus(in studentID): bool | 判断学生当前是否在校 |
| QueryStudentInCampusInteraction. showResult() | 根据程序运行的结果，将程序执行成功或失败告知管理员。如果成功则显示被查询学生是否在校，失败则显示错误信息 |
| QueryStudentInCampusInteraction. enterQueriedStudentID() | 要求管理员输入要查询在校情况的学生的ID |
| QueryStudentInCampusInteraction. showErrorMessage() | 程序执行失败时，显示错误信息以用于纠错 |
| GraduateInformationDelete. checkStudentIDReasonable(): bool | 检查输入的学生ID是否正确以及该学生是否为毕业生 |
| GraduateInformationDelete. deleteGraduateInformation(in studentID) | 在数据库中删除该学生的住宿信息 |
| GraduateInformationDeleteInteraction. showResult() | 根据程序运行的结果，将程序执行成功或失败告知管理员。如果成功则显示该学生的住宿信息已被删除，失败则显示错误信息 |
| GraduateInformationDeleteInteraction. enterQueriedStudentID() | 要求管理员输入要删除的毕业生的ID |
| GraduateInformationDeleteInteraction. showErrorMessage() | 程序执行失败时，显示错误信息以用于纠错 |

### 4.2.2学生管理系统



本系统用于管理员管理学生住宿相关的各项事务，包含了登记学生晚归不归信息、登记学生长时间离校、登记外来访问人员、发布公告通知等。控制类StudentManageControl用于管理各个功能的调用，控制类RegisterStudentNotBack，RegisterStudentLeave，IssueAnnounce， RegisterVisitor用于控制各个子功能的实现。

实体类StudentNotBackInformation类保存了学生晚归不归信息，StudentLeaveInformation类记录了学生因故长时间离校的信息，Announce类保存了通知内容及相关信息（发布日期，发布人，通知识别ID），Vistor类保存了来访人员的记录

|  |  |
| --- | --- |
| 方法 | 说明 |
| StudentManageControl. registerVister() | 调用登记访客功能 |
| StudentManageControl. issueAnnounce() | 调用发布通知功能 |
| StudentManageControl. registerStudentNotBack() | 调用登记学生晚归未归功能 |
| StudentManageControl. registerStudentLeave() | 调用登记学生长时间离校功能 |
| RegisterStudentNotBack. pushInStudentNotBackInformation() | 将学生晚归未归信息保存至数据库中 |
| RegisterStudentNotBack. getStudentNotBackInformation(in studentID) | 根据参数中的学生ID，在数据库中获取该学生的晚归未归信息 |
| RegisterStudentNotBack. updateBackDate(in backDate) | 更新学生的晚归未归信息，包括修改信息或填入回校时间等 |
| RegisterStudentNotBack. deleteStudentNotBackInformation(in studentID) | 根据参数中的学生ID，删除相应的学生晚归未归记录 |
| StudentNotBackInformation. checkDataReasonable(): bool | 检查各个属性是否合理及符合规范，在每一次管理员输入后调用 |
| StudentNotBackInformation. setBackDate(in date) | 输入学生的返校时间 |
| StudentNotBackInformation. checkIfStudentIDExist(in studentID):bool | 查询学生ID是否合理及是否在晚归未归状态中 |
| RegisterStudentNotBackInteraction. enterStudentNotBackInformation() | 新增一个学生晚归未归记录 |
| RegisterStudentNotBackInteraction. showStudentNotBackInformation() | 输出学生晚归未归记录 |
| RegisterStudentNotBackInteraction. enterStudentBackDate() | 管理员输入学生返校时间 |
| RegisterStudentLeave. pushInStudentLeaveInformation() | 将学生长时间离校信息插入数据库中保存 |
| RegisterStudentLeave. getStudentLeaveInformation(in studentID) | 根据参数中的学生ID从数据库中获取学生长时间离校信息 |
| RegisterStudentLeave. updateBackDate(in backDate) | 更新学生返校时间 |
| RegisterStudentLeave. updateStudentLeaveInformation(in studentID) | 更新学生长时间离校信息 |
| RegisterStudentLeave. checkIfStudentIDExist(in studentID) | 检查学生ID是否存在及合理 |
| StudentLeaveInformation. checkDataReasonable(): bool | 检查对象中保存的学生长时间离校信息是否合理且符合规范 |
| StudentLeaveInformation. setBackDate(in date) | 记录学生返校时间 |
| StudentLeaveInformation. changeData() | 更改对象中记录的信息 |
| RegisterStudentLeaveInteraction. enterStudentLeaveInformation() | 要求管理员输入学生离校信息 |
| RegisterStudentLeaveInteraction. showStudentLeaveInformation(studentID) | 根据参数中的学生ID显示其长时间离校信息（如果有的话） |
| RegisterStudentLeaveInteraction. enterStudentBackDate() | 管理员输入学生返校时间 |
| RegisterStudentLeaveInteraction. changeStudentLeaveInformation(in studentID) | 管理员更改学生长时间离校信息 |
| IssueAnnounce. pushInAnnounce(in Announce\*) | 将通知信息插入数据库保存 |
| IssueAnnounce. changeAnnounce(in Announce\*) | 根据参数，修改相应的通知 |
| IssueAnnounce. getAnnounce(in announceID): Announce\* | 根据通知ID获得相应的通知类对象 |
| Announce. changeContent() | 修改通知内容 |
| Announce. editContent() | 编辑通知内容 |
| Announce. autoSetAnnounceID() | 类对象自动设定通知ID，用于区分，保证与其他通知的ID互斥 |
| IssueAnnounceInteraction. createAnnounce() | 管理员新增一个通知 |
| IssueAnnounceInteraction. changeAnnounce(in Announce\*) | 管理员修改参数所指定的通知 |
| IssueAnnounceInteraction. showResult() | 展示新增的通知或修改后的通知 |
| RegisterVisitor. pushInVistorInformation() | 将访客信息插入数据库中保存 |
| RegisterVisitor. getVistorInformation(in vistorIdentityCardID) | 根据来访者身份证号获取其来访信息（如果有的话） |
| RegisterVisitor. updateVisterLeaveTime(in date) | 更新来访者离开时间 |
| Vistor. checkInformationReasonable() | 检查来访者信息是否正确且符合规范 |
| Vistor. ifLeavingTimeLost(): bool | 检查因为各种原因，管理员是否未能记录来访者离开时间 |
| RegisterVistorInteraction. enterVistorInformation() | 管理员输入来访者信息 |
| RegisterVistorInteraction. enterVistorLeaveTime() | 管理员输入来访者离开时间 |
| RegisterVistorInteraction. showVistorInformation() | 展示输入后的来访者信息，在每一次输入完成后调用 |

### 4.2.3 宿舍设施报修



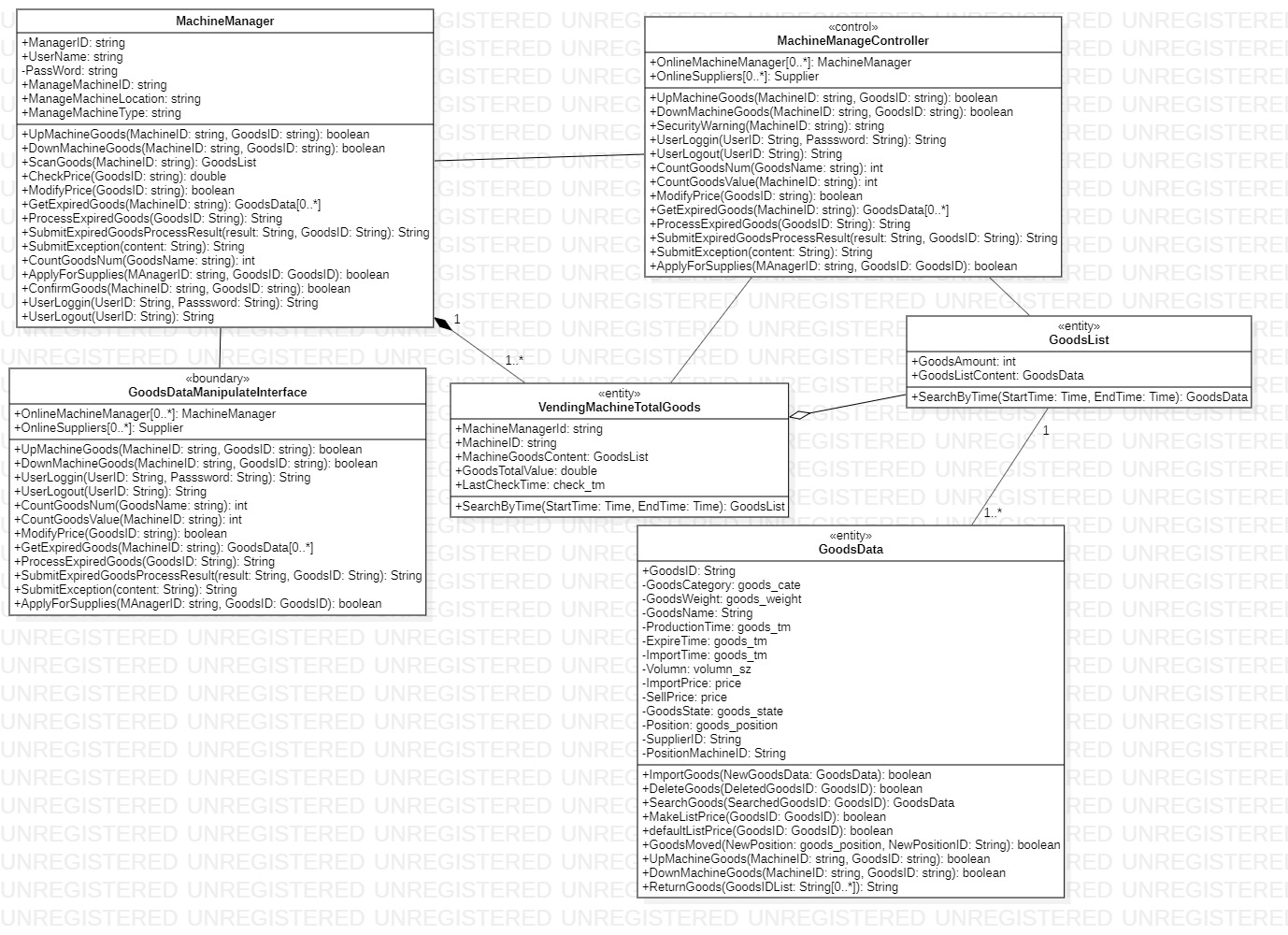
该图反映了宿舍设施报修用例，宿舍管理员可以使用该功能申请报修，维修人员可以通过此功能接受报修任务。在设施出现故障后，管理员可以发出维修请求，分派管理系统（体现为控制类TaskDistributor）将其转化为配送任务并分配到最合适的维修员的任务列表中。维修员可根据自身情况接受任务，并在完成维修后确认任务完成；此时系统提示管理员确认维修情况，在确认维修完成无误后，系统修改机械故障状态并同步到机器设施内部。

实体类DeliverTask描述的是一系列维修任务，实体类TaskList和控制类TaskDistributor是它的友元。DeliverTask类提供了较为全面的属性以记录描述配送任务需要的属性，但并不支持记录或转述Order类的所有信息，如OrderCreatorID；相似地，Order类也并不能访问DeliverTask类的大多数属性。

在管理员的一次报修请求发出后，TaskDistributor.GenerateTask()会将对应的Order类实例作为参数，生成一个DeliverTask类实例。

|  |  |
| --- | --- |
| 方法 | 说明 |
| Administrator.ModifyFaultInfo(…): boolean | 该函数封装了一些由控制类和边界类提供的相关函数，允许管理员在申请过程中对故障信息进行修改（包括新增、修改、撤回）。返回布尔类型表示操作失败或成功 |
| Administrator.ConfirmOrder(…): boolean | 该函数用于确认报修请求 |
| Administrator.QueryHistoryOrder(): boolean | 该函数用于查询设施历史报修请求 |
| Mender.ConfirmTaskFinished(…): boolean | 维修员实例确认维修任务已完成。该函数内部会调用UpdateTaskStatus以将指定ID（taskID）的任务的任务状态（taskStatus）属性设置为“finished”。 |
| Mender.AcceptTask(…): boolean | 维修员接受一个被放入维修任务列表的任务。该函数内部会调用UpdateTaskStatus以将指定ID（taskID）的任务的任务状态（taskStatus）属性设置为“accepted”。 |
| TaskList.ModifyAcceptTaskNum(): void | 根据当前实际接受任务的数量修改已接受任务数量(acceptTaskNum)的值。 |
| TaskList.ModifyAcceptValidity(): void | 当已接受任务数量>3时将接受许可(acceptValidity)设置为 false，否则置true。 |
| TaskDistributor.GenerateTask(…): DeliverTask | 根据报修请求生成维修任务。生成的维修任务保留请求的报修信息和任务号，并加入了维修任务的地理位置，维修时间，维修员ID，维修任务状态等信息 |
| TaskDistributor.EvaluateCourierStatus(…): double | 传入参数为维修人员 ID 及任务 ID，根据维修人员专业领域及维修任务列表与任务信息比对，利用系统算法得出打分并以浮点型返回，作为是否将该任务分配给某维修人员员的评判标准。 |
| TaskDistributor.DistributeTask(…): boolean | 将一个维修任务(deliverTask)加入一个维修员的任务列表，返回布尔型表示加入成功或失败。 |
| TaskDistributor.UpdateTaskStatus(…): boolean | 更新任务状态。可能传入的 status参数为  “initialized”, “distributed”, “accepted”和 “finished”，分别表示任务已生成、已被分派、已被接受和已被维修人员确认完成。 |
| TaskDistributor.RetrieveTask(…): boolean | 将一个任务从指定的任务列表中撤回。调用条件为任务状态(taskStatus)  ≠0，即任务被确认完成或被其他人员接受。 |
| OrderForm.AskForConfirmOrder(…): void | 向指定管理员弹出请求确认报修申请界面。 |
| OrderForm.AskForAcceptTask(…): void | 向指定的维修人员弹出请求接受维修任务界面。 |
| OrderForm.ShowTaskList(…): void | 显示任务列表。维修人员可以在此界面上选择接受任务。 |
| OrderForm.ShowFaultCondition(…): void | 显示设施情况。管理员和维修人员可以在此查看某设施目前故障情况。 |
| OrderController.UpdateOrderStatus(…):boolean | 修改指定 ID 的任务的状态。在本图中，该函数在DistributeTask()返回true时参数 status=”distributed”将被直接赋值给请求对象的orderStatus属性，表示报修任务已被分派给维修员；在管理员对象执行ConfirmOrder()时被调用时参数status=”finished”将被直接赋值给请求对象的orderStatus属性，表示任务已完成。 |

### 4.2.4宿舍自动贩卖机货物整理



该图包含了宿舍自动贩卖机货物整理用例，管理员通过该功能整理从供货商引进的自动贩卖机中的商品。机器管理员将执行以下几个操作：

商品上下架处理：在特定条件下对商品进行上架或下架操作。

扫描商品：检查贩卖机中剩余商品的种类情况。

标价错误处理：筛选出标价错误的商品进行价格修正。

过期商品处理：对过期商品进行下架操作。

商品整理：对货架中的商品进行整理操作。

补货请求：检查出缺货的商品并向供应商发出补货请求。

供货处理：对供货商提供的商品进行检查，录入信息操作，并进行上架处理。

**类图说明：**

MachineManager 贩卖机管理员

|  |  |
| --- | --- |
| 方法 | 说明 |
| UpMachineGoods(in MachineID:string, in GoodsID:string): boolean | 商品上架操作，返回布尔型表示操作成功或失败。 |
| DownMachineGoods(in MachineID:string, in GoodsID:string): boolean | 商品下架操作，返回布尔型表示操作成功或失败。 |
| ScanGoods(in MachineID:string): GoodsList | 扫描供应商品信息 |
| CheckPrice(in GoodsID:string): double | 检查供货商品价格是否正确，返回在贩卖机中的该商品此时价格 |
| ModifyPrice(in GoodsID:string): boolean | 修改贩卖机中某商品的价格，返回布尔型表示修改成功或失败。 |
| GetExpiredGoods(in MachineID:string): GoodsData[0..\*] | 获取过期商品信息 |
| SubmitExpiredGoodsProcessResult(in result:String, in GoodsID:String): String | 过期商品提交处理结果 |
| ApplyForSupplies(in ManagerID:string, in GoodsID:GoodsID): boolean | 向商品特定供应商提交补货请求，生成的补货请求包括商品种类，商品ID，补货需求数量，请求者（管理员）ID等信息。 |
| ConfirmGoods(in MachineID:string, in GoodsID:string): boolean | 管理员确认贩卖机中商品无误，机器没有出现故障。 |
| CountGoodsNum(in GoodsName:string): int | 统计贩卖机中各类商品实时总数量，返回总值 |
| SubmitException(in content:String): String | 提交异常报告 |

<<control>>MachineManageController 控制类

<<boundary>>GoodsDataManipulateInterface 边界类

<<entity>>VendingMachineTotalGoods 自动贩卖机货架信息

<<entity>>GoodsList 自动贩卖机特定商品列表

<<entity>>GoodsData 商品信息

### 4.2.5财务统计系统



该图反映了财务统计用例，财务人员通过此功能对宿舍开销费用进行统计，并制作报表。财务人员需要检索统计的开销信息包括了设施维护、日常事务、宿舍建设、员工薪水、突发事件处理、贫困补助。

该系统的主要参与者是财务人员（会计师），主要负责以下操作：

获取开销信息：包括设施维护、日常事务、宿舍建设、员工薪水、突发事件处理、贫困补助等开销的具体信息。在该用例中，把“开销数据”作为一个实体类，并作为其余六个具体开销信息的泛化，其余六个具体开销信息继承了“开销数据”的属性和操作，并且拥有自己不同的、独立的属性和操作。

获取报表信息，创建报表，制作报表，提交报表。

分析与统计：对开销情况进行数据统计并给出专业性的分析。

**类图说明：**

Accountant 会计师

|  |  |
| --- | --- |
| 方法 | 说明 |
| GetReportInfo(in StartTime:tm, in EndTime:tm): ReportData | 获取一段时间内的报表信息 |
| GetExpensesInfo(in StartTime:tm, in EndTime:tm): ExpensesData | 获取一段时间内的开销信息，返回一个“开销数据”类，包括了设施维护、日常事务、宿舍建设、员工薪水、突发事件处理、贫困补助等开销的具体信息。 |
| CreateNewReport(in FilePath:String): ReportData | 创建新报表 |
| SaveReport(in Report:ReportData): boolean | 保存报表 |
| UpdateReport(in Report:ReportData, in Content:String): boolean | 更新报表内容 |
| SubmitReport(in Report:ReportID): boolean | 提交报表 |
| SubmitException(): String | 提交异常报告 |
| AnalyzeAndStatistic(in Report:ReportData): void | 进行统计分析并制作报表 |

<<Boundary>>RegisterForStatisticsUserInterface 边界类

<<control>>StatisticsControler 控制类

<<entity>>ExpensesData 开销信息

<<entity>>ReportData 报表信息

<<entity>>EmployeeSalaryExpensesData员工薪水开销信息

<<entity>>PovertySubsidiesExpensesData贫困补助开销信息

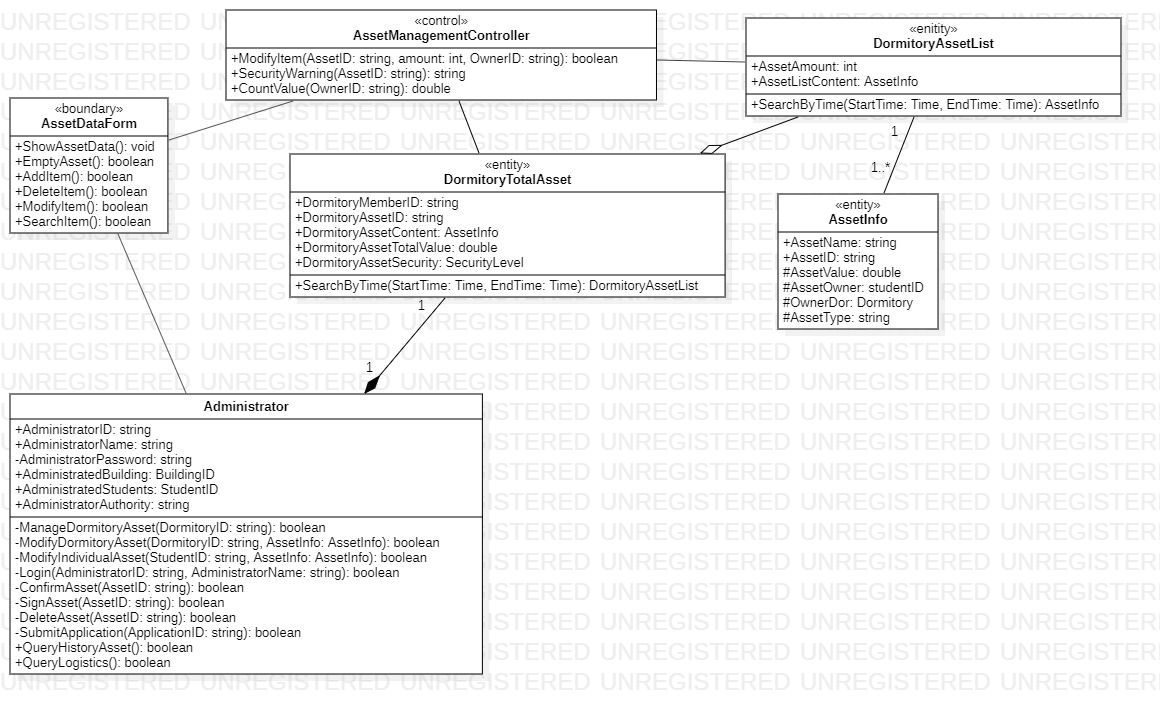
<<entity>>FacilityMaintenanceExpensesData设施维护开销信息

<<entity>>DailyTransactionExpensesData日常事务开销信息

<<entity>>DormitoryConstructionExpensesData宿舍建设开销信息

<<entity>>EmergencyExpensesData紧急事件处理开销信息

### 4.2.6宿舍资产管理



该图反映了宿舍资产管理用例，管理员和学生（主要参与者）可以对自己宿舍内部资产进行登记管理，包括添加、删除、修改资产。其中实体类DormitoryTotalAsset中的属性 DormitoryAssetContent 是包含了 DormitoryAssetList 类的实例，而DormitoryAssetList 类包含了属性资产数量（AssetAmount）和资产信息（AssetInfo类对象）。实体类AssetInfo类对象包含了一个资产具体的信息，包括资产类型、名称、价值、拥有者等等，其仅能被管理员调用函数生成、修改或删除。

其余还有：

<<control>>AssetManagementController 控制类

<<boundary>>AssetDataForm 边界类

**类图说明：**

Administrator 管理员

|  |  |
| --- | --- |
| 方法 | 说明 |
| ManageDormitoryAsset(in DormitoryID:string): boolean | 该函数封装了一些由控制类AssetManagementController和边界类AssetDataForm 提供的相关函数，允许管理者修改（包括增加、修改、删除）自己管理的资产，返回的布尔型表示操作失败或成功。 |
| ModifyDormitoryAsset(in DormitoryID:string, in AssetInfo:AssetInfo): boolean | 该函数允许管理者特定地以宿舍为单位对资产进行统一管理 |
| ModifyIndividualAsset(in StudentID:string, in AssetInfo:AssetInfo): boolean | 该函数允许管理者特定地以学生为单位对资产进行独立管理 |
| SubmitApplication(in ApplicationID:string): boolean | 提交资产使用申请 |
| QueryHistoryAsset(): boolean | 历史资产记录查询 |

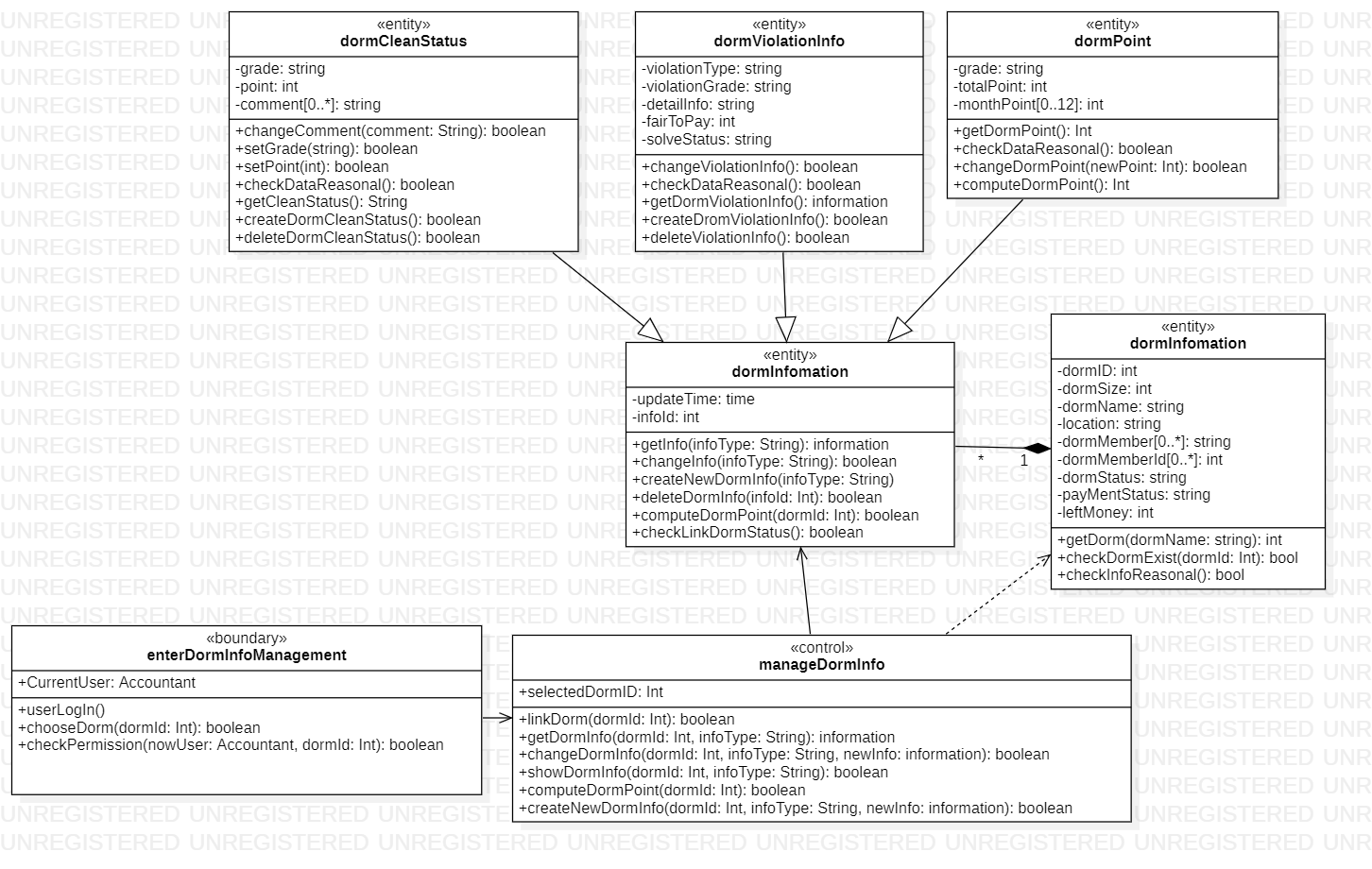
AssetDataForm 边界类

|  |  |
| --- | --- |
| 方法 | 说明 |
| ShowAssetData(): void | 显示当前资产具体信息。 |
| EmptyAsset(): boolean | 弹出清空所有资产界面，返回布尔值表示管理者选择清空或取消清空。 |
| AddItem(): boolean | 弹出添加某资产到总资产的界面，返回布尔值表示管理者成功添加或取消操作。 |
| DeleteItem(): boolean | 弹出从总资产中删除特定资产的界面，返回布尔值表示管理者成功操作或取中止操作。 |
| ModifyItem(): boolean | 弹出修改总资产中资产数量界面，返回布尔值表示管理者成功操作或取中止操作。 |
| SearchItem(): boolean | 弹出在总资产中查找特定资产的界面，返回布尔值表示系统查询成功或取查询失败。 |

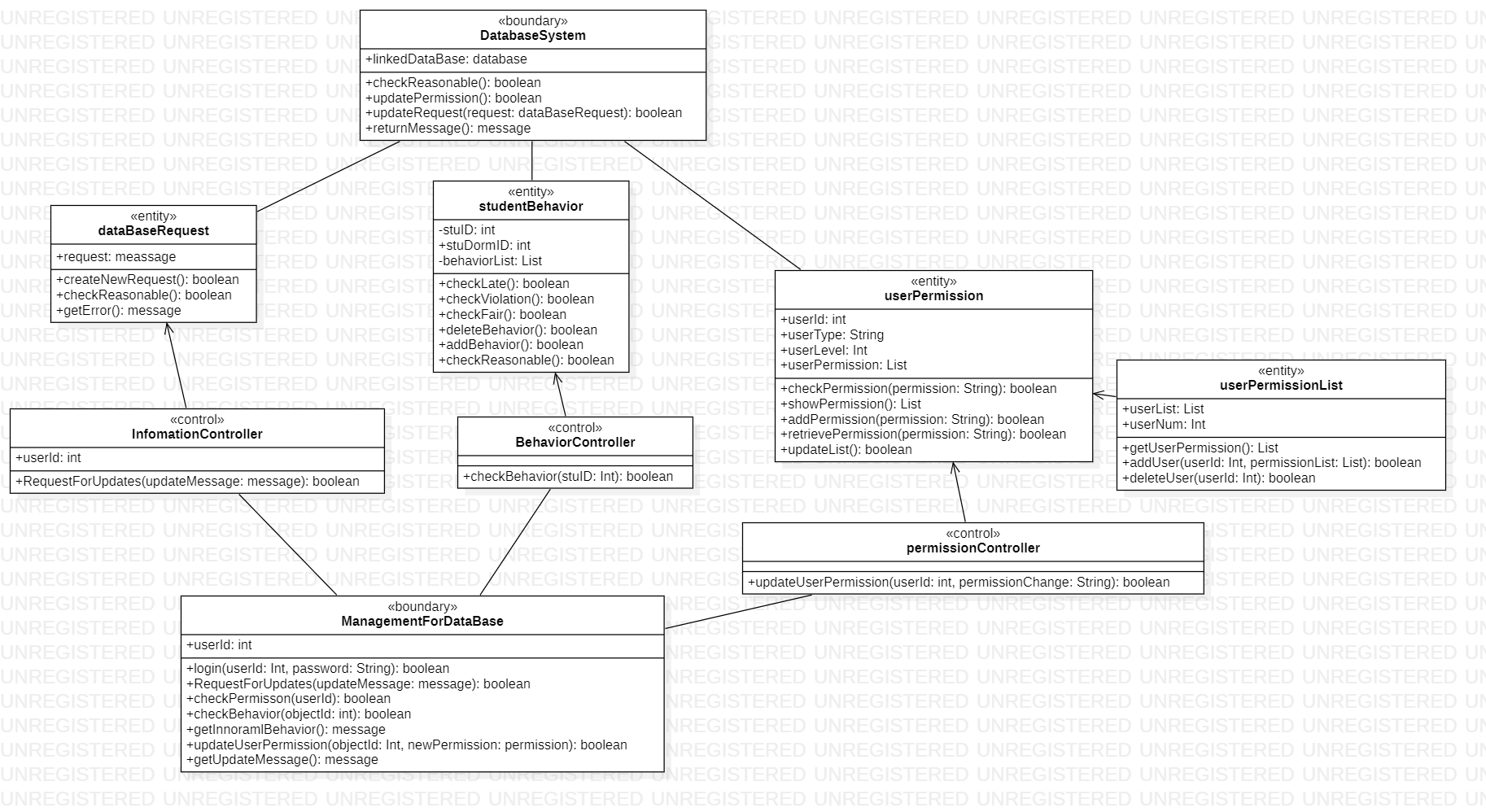
AssetManagementController 控制类

|  |  |
| --- | --- |
| 方法 | 说明 |
| ModifyItem(in AssetID:string, in amount:int, in OwnerID:string): boolean | 对于指定ID管理者管理的总资产修改资产。根据调用情况不同，党amount=-1时删除ID= AssetID 的资产，当amount≠-1 时将ID= AssetID 的DormitoryAssetList 中的AssetAmount修改为 amount。返回布尔值表示修改成功或失败。 |
| SecurityWarning(in AssetID:string): string | 资产安全警告。当系统检查到资产数量、价值、持有者等信息出现异常时，弹出资产数据异常警告，返回具体的警告内容。 |
| CountValue(in OwnerID:string): double | 传入一个资产持有者的ID作为参数，返回该持有者持有的所有资产价值总和。 |

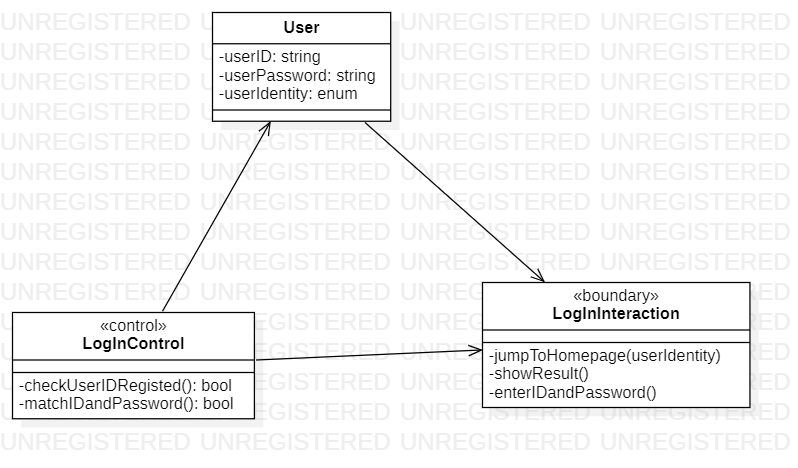
### 4.2.7 宿舍成绩系统



### 4.2.8 数据库管理系统



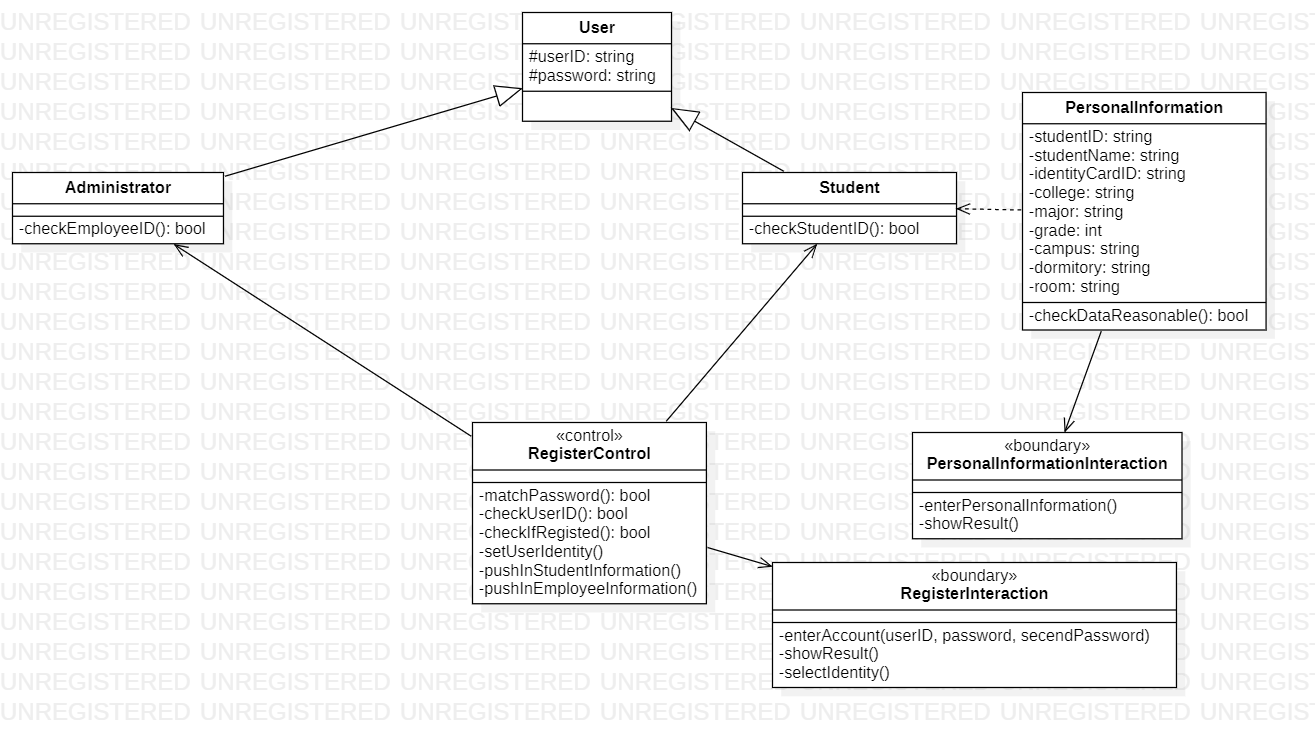
### 4.2.9 登录



控制类LogInControl管理整个登录过程，用户输入账号及密码后判断账号是否已经注册以及账号和密码是否匹配，登录成功则跳转到相应页面以让用户使用相应服务。

|  |  |
| --- | --- |
| 方法 | 说明 |
| LogInControl. checkUserIDRegisted(): bool | 检查用户输入的账号是否已注册在宿舍管理系统中 |
| LogInControl. matchIDandPassword(): bool | 检查账号及密码是否匹配 |
| LogInInteraction. jumpToHomepage(in userIdentity) | 用户登录成功后，根据身份跳转到相应的系统主页（管理员跳转至管理员页面，学生跳转至学生页面） |
| LogInInteraction. showResult() | 显示登录结果（成功或失败） |
| LogInInteraction. enterIDandPassword() | 用户输入账号及密码 |

### 4.2.10 注册

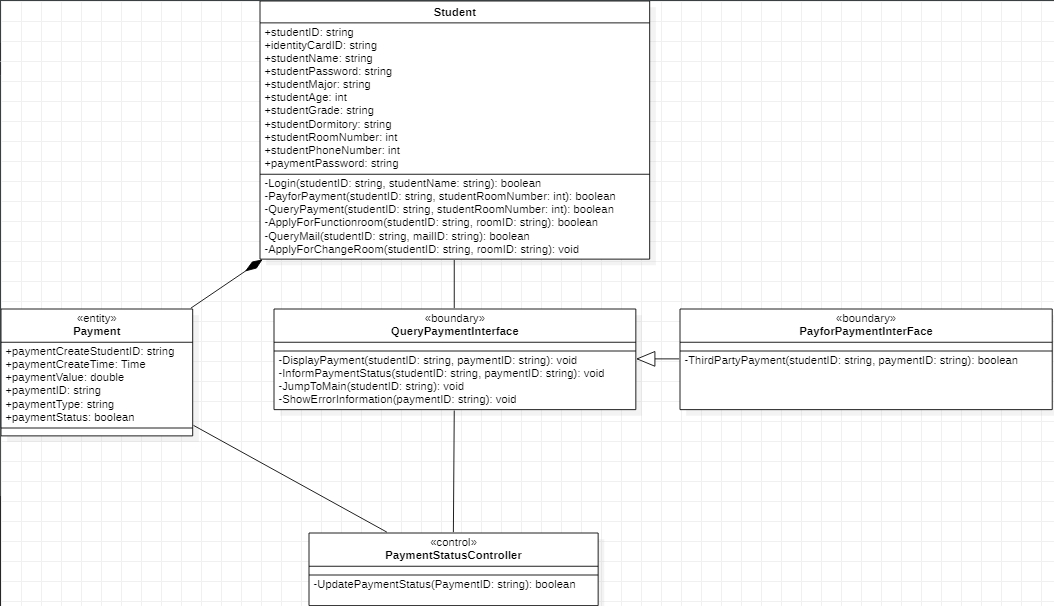


实体类Student类和Administrator类作为User类的子类，代表管理员及学生身份。学生在注册时需要输入个人身份信息等并存入数据库中。

控制类RegisterControl管理用户注册过程，并将注册后的账号及个人信息储存入数据库中。

|  |  |
| --- | --- |
| 方法 | 说明 |
| RegisterControl. setUserIdentity() | 设定注册用户的身份（管理员或学生） |
| RegisterControl. matchPassword(): bool | 比对两次输入的密码是否一致 |
| RegisterControl. checkUserID(): bool | 检查用户ID是否格式正确以及是否注册在学校数据库中 |
| RegisterControl. checkIfRegisted(): bool | 检查用户是否已经在宿舍管理系统中注册过账号 |
| RegisterControl. pushInEmployeeInformation() | 将管理员账号储存入数据库中 |
| RegisterControl. pushInStudentInformation() | 将学生账号及其个人信息储存入数据库中 |
| Student. checkStudentID(): bool | 检查学生ID是否在学校学生系统中注册 |
| Administrator. checkEmployeeID(): bool | 检查管理员ID是否在学校教职工系统中注册 |
| PersonalInformation. checkDataReasonable(): bool | 检查学生个人信息是否符合规范及正确 |
| PersonalInformationInteraction. enterPersonalInformation() | 要求学生输入个人信息 |
| PersonalInformationInteraction. showResult() | 显示学生个人信息结果 |
| RegisterInteraction. enterAccount(in userID, in password, in secendPassword) | 要求用户输入学号或工号及两次密码，用于注册账号 |
| RegisterInteraction. showResult() | 显示注册结果 |
| RegisterInteraction. selectIdentity() | 让用户选择身份 |

### 4.2.11 寝室费用系统



该图反映了学生查询寝室费用、缴纳寝室费用两个用例。学生通过此功能查询并缴纳与寝室有关的费用，包括住宿费、电费。其中边界类QueryPaymentInterface是学生查询寝室费用的界面，边界类PayforPaymentInterface是前者的子类，是学生查询完毕后缴纳寝室费用的界面。控制类PaymentStatusController会在学生缴纳费用完毕后，改变该笔支付账单的状态。实体类Payment是费用支付账单。

类图说明：

Student 学生

|  |  |
| --- | --- |
| 方法 | 说明 |
| QueryPayment(in studentID:string, in studentRoomNumber:int): boolean | 该函数允许学生查询与本人相关的一系列寝室费用，包括住宿费，电费。 |
| PayforPayment(in studentID:string, in studentRoomNumber:int): boolean | 该函数允许学生缴纳与本人相关的一系列寝室费用，包括住宿费，电费。 |

QueryPaymentInterface 边界类

|  |  |
| --- | --- |
| 方法 | 说明 |
| DisplayPayment(in studentID:string, in paymentID:string): void | 显示当前与该学生相关的寝室费用支付信息。 |
| InformPaymentStatus(in studentID:string, in paymentID:string): void | 显示当前与该学生相关的寝室费用支付账单的状态，包括已支付与未支付。 |
| JumpToMain(in studentID:string): void | 操作完毕后，若无异常情况，则跳回主界面。 |
| ShowErrorInformation(in paymentID:string): void | 操作完毕后，若出现异常情况，则显示错误信息。 |

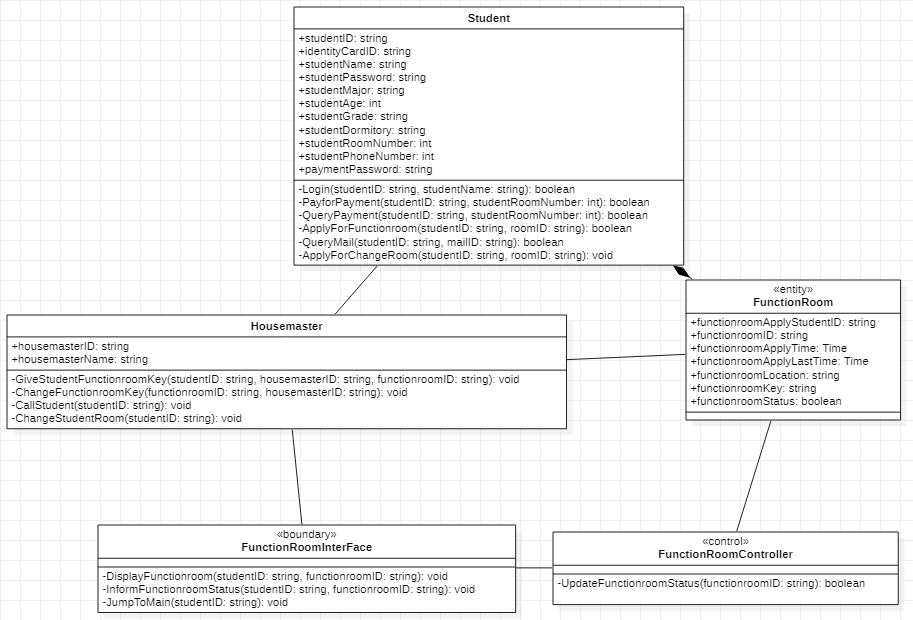
PayforPaymentInterface 边界类

|  |  |
| --- | --- |
| 方法 | 说明 |
| ThirdPartyPayment(in studentID:string, in paymentID:string): boolean | 唤起第三方支付平台，并根据支付成功与否返回父类边界。 |

PaymentStatusController 控制类

|  |  |
| --- | --- |
| 方法 | 说明 |
| UpdatePaymentStatus(in PaymentID:string): boolean | 当学生支付费用成功后，该函数会将账单状态更新为“已支付” |

### 4.2.12 功能室申请



该图反映了功能室申请这一用例。学生通过此功能向系统申请使用功能室的权限。在申请成功后，系统将通知宿管给予学生功能室的密码。学生拿到密码后，可以根据申请时间自行前往功能室。边界类FunctionRoomInterface是学生查看功能室使用情况并进行申请的界面，控制类FunctionRoomController当功能室被申请后，会将其状态更改为“不可申请”，当使用完毕后，会将其状态更改为“可申请”。

类图说明

Student 学生

|  |  |
| --- | --- |
| 方法 | 说明 |
| ApplyForFunctionroom(in studentID:string, in roomID:string): boolean | 该函数允许学生向系统提出功能室的申请。 |

Housemaster 宿管

|  |  |
| --- | --- |
| 方法 | 说明 |
| GiveStudentFunctionroomKey(in studentID:string, in housemasterID:string, in functionroomID:string): void | 该函数允许宿管给予该学生成功申请的功能室的钥匙。 |
| ChangeFunctionroomKey(in functionroomID:string, in housemasterID:string): void | 该函数允许宿管更改功能室的钥匙。 |
| CallStudent(in studentID:string): void | 该功能允许宿管呼叫学生。 |

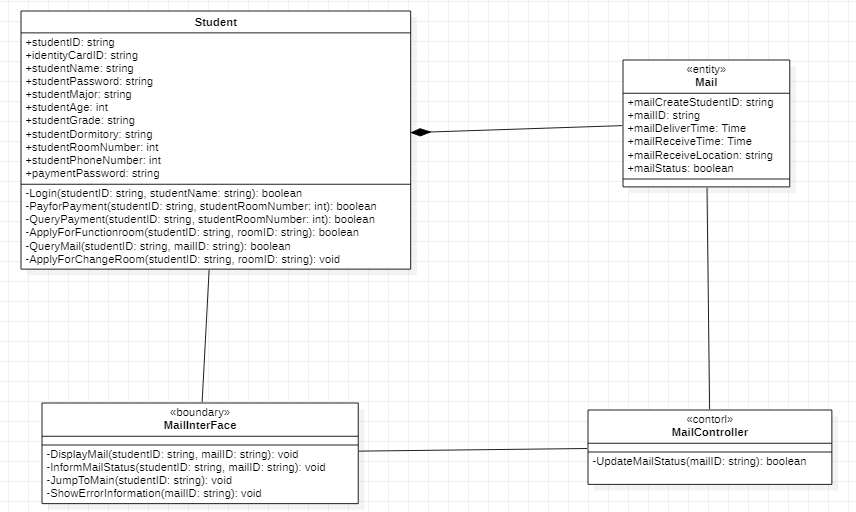
FunctionRoomInterface 边界类

|  |  |
| --- | --- |
| 方法 | 说明 |
| DisplayFunctionroom(in studentID:string, in functionroomID:string): void | 显示当前功能室信息。 |
| InformFunctionroomStatus(in studentID:string, in functionroomID:string): void | 显示当前功能室的使用情况，包括可申请和不可申请。 |
| JumpToMain(in studentID:string): void | 操作完毕后，若无异常情况，则跳回主界面。 |
| ShowErrorInformation(in mailID:string): void | 操作完毕后，若出现异常情况，则显示错误信息。 |

FunctionRoomController 控制类

|  |  |
| --- | --- |
| 方法 | 说明 |
| UpdateFunctionroomStatus(in functionroomID:string): boolean | 当学生申请功能室成功后，该函数会将功能室状态更改为“不可申请”；当学生使用完毕功能室后，该函数会将功能室状态更改为“可申请”。 |

### 4.2.13 快递、快件查询



该图反映了快递、快件查询这一用例。学生通过此功能查询自身的快递、快件相关信息。包括快递、快件的到达情况，签收情况等。其中边界类MailInterFace是学生查询快递、快件的界面。控制类MailController是在快递、快件签收完毕后改变快递、快件的签收状态。实体类Mail是快递、快件。

类图说明：

Student 学生

|  |  |
| --- | --- |
| 方法 | 说明 |
| QueryMail(in studentID:string, in mailID:string): boolean | 该函数允许学生查询当前快递、快件的签收情况 |

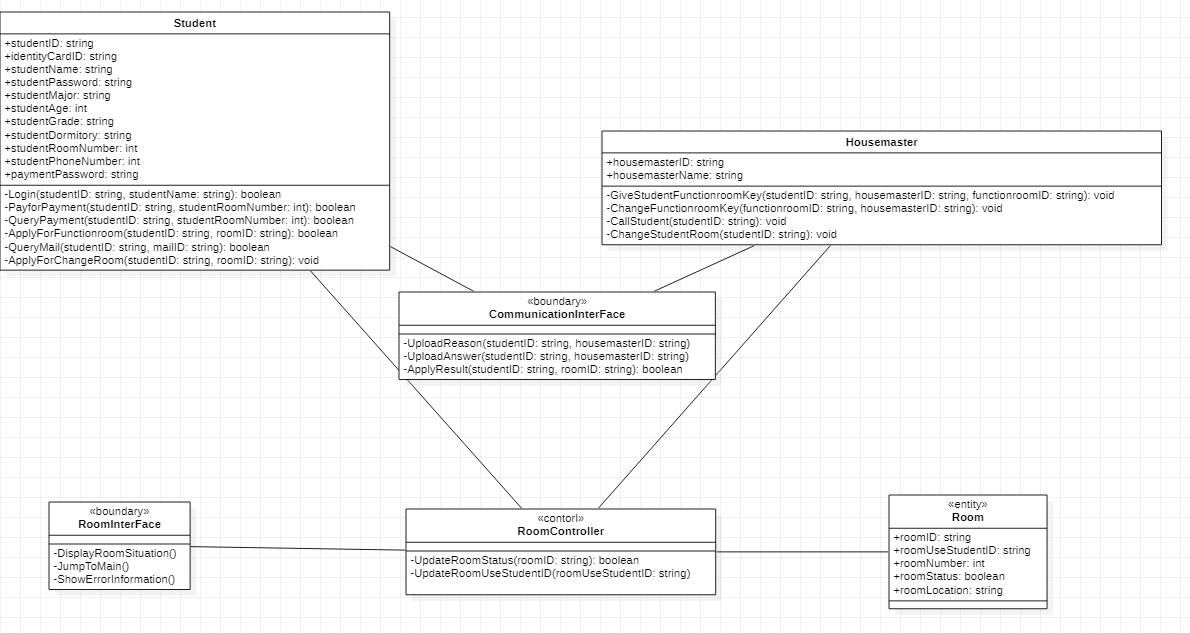
MailInterface 边界类

|  |  |
| --- | --- |
| 方法 | 说明 |
| DisplayMail(in studentID:string, in mailID:string): void | 显示当前与该学生相关的快递、快件信息。 |
| InformMailStatus(in studentID:string, in mailID:string): void | 显示当前与该学生相关的快递、快件状态，包括已签收与未签收。 |
| JumpToMain(in studentID:string): void | 操作完毕后，若无异常情况，则跳回主界面。 |
| ShowErrorInformation(in mailID:string): void | 操作完毕后，若出现异常情况，则显示错误信息。 |

MailController 控制类

|  |  |
| --- | --- |
| 方法 | 说明 |
| UpdateMailStatus(in mailID:string): boolean | 当学生签收快递成功后，该函数会将快递状态更新为“已签收  ” |

### 4.2.14 申请更换宿舍



该图反映了学生申请更换宿舍这一用例。学生通过此功能向宿管申请更换宿舍。在申请更换宿舍时，学生需要向宿管提出更换宿舍的原因，并等待宿管答复。在得到允许后，即可成功更换宿舍，否则更换宿舍失败。边界类RoomInterface是在申请更换宿舍时查看当前宿舍情况的界面，边界类CommunicationInterface是学生与宿管进行直接交流时的界面。控制类RoomController会在学生更换宿舍成功后，将相关宿舍的人员状况更新。

类图说明

Student 学生

|  |  |
| --- | --- |
| 方法 | 说明 |
| ApplyForChangeRoom(in studentID:string, in roomID:string): void | 该用例允许学生提出更换宿舍的申请。 |

Housemaster 宿管

|  |  |
| --- | --- |
| 方法 | 说明 |
| ChangeStudentRoom(in studentID:string): void | 该用例允许宿管在学生申请更换宿舍成功后，对学生的宿舍进行调整。 |

RoomInterface 边界类

|  |  |
| --- | --- |
| 方法 |  |
| DisplayRoomSituation() | 显示当前宿舍人员分配情况 |
| JumpToMain() | 操作完毕后，若无异常情况，则跳回主界面。 |
| ShowErrorInformation(in mailID:string): void | 操作完毕后，若出现异常情况，则显示错误信息。 |

CommunicationInterface 边界类

|  |  |
| --- | --- |
| 方法 | 说明 |
| UploadReason(in studentID:string, in housemasterID:string) | 该函数允许学生向宿管发送申请更换宿舍的原因 |
| UploadAnswer(in studentID:string, in housemasterID:string) | 该函数允许宿管向学生发送对其更换宿舍原因的答复 |
| ApplyResult(in studentID:string, in roomID:string): boolean | 该函数向学生告知最终申请结果 |

Roomcontroller

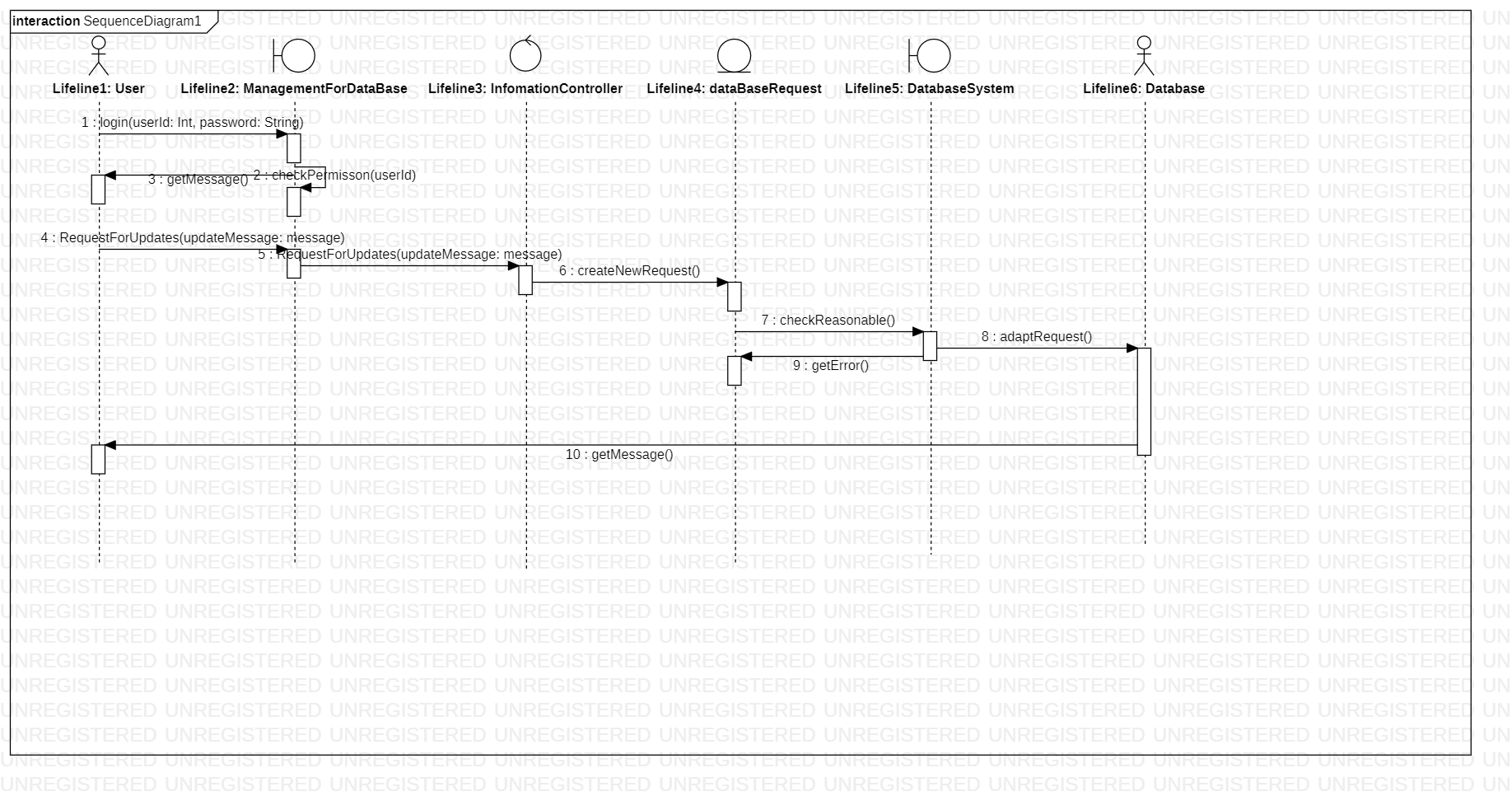
|  |  |
| --- | --- |
| 方法 | 说明 |
| UpdateRoomStatus(in roomID:string): boolean | 该函数将更新宿舍状态，包括有“已满员”和“未满员” |
| UpdateRoomUseStudentID(in roomUseStudentID:string) | 该函数将更新宿舍人员信息 |

## 4.3 序列图

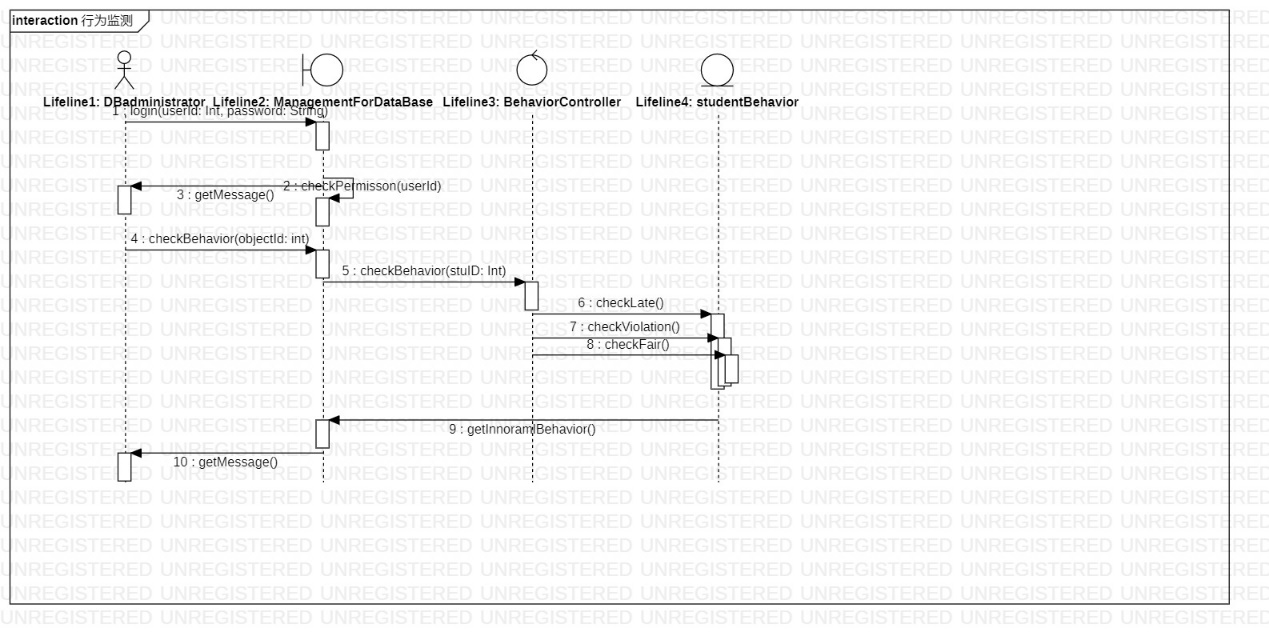
本部分将给出一些序列图（又称时序图，Sequence Diagram）及部分描述。



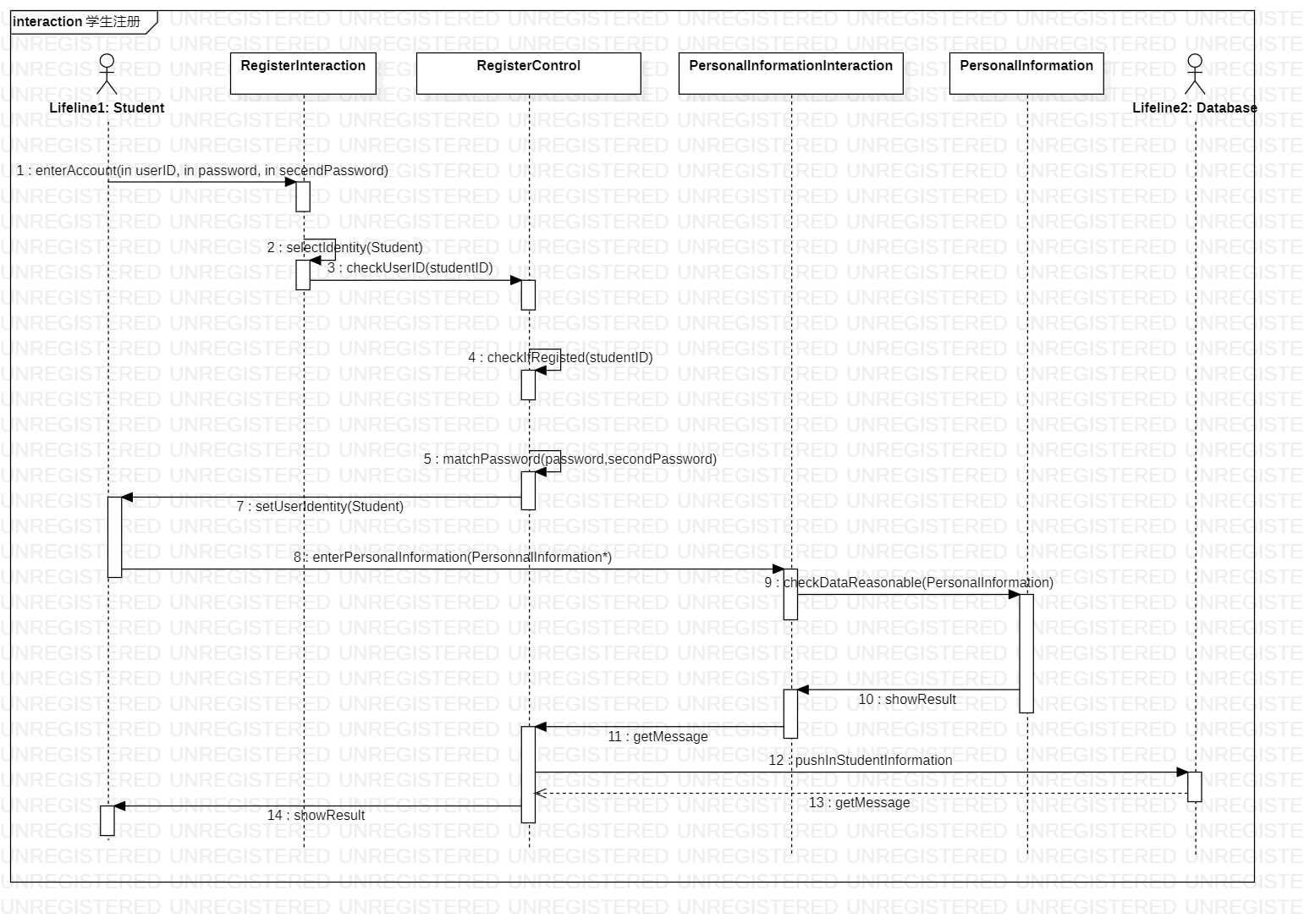
### 4.3.1 信息更新与检索



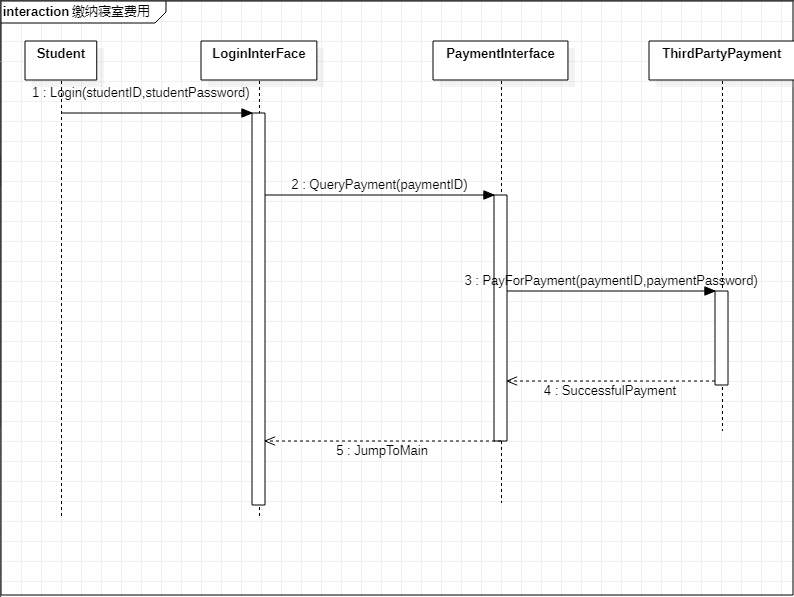
### 4.3.2 行为检测



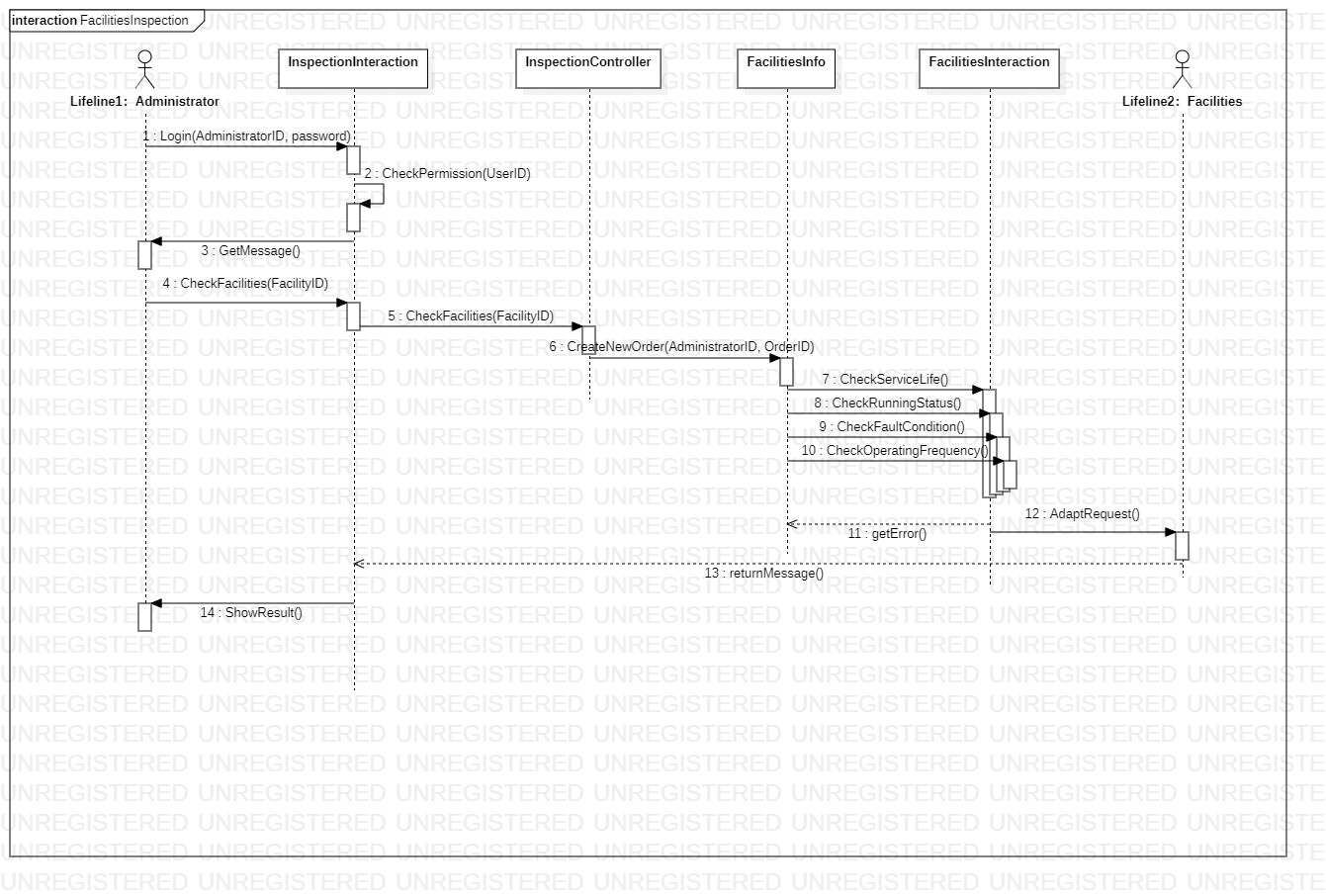
### 4.3.3 学生注册



### 4.3.4 缴纳寝室费用

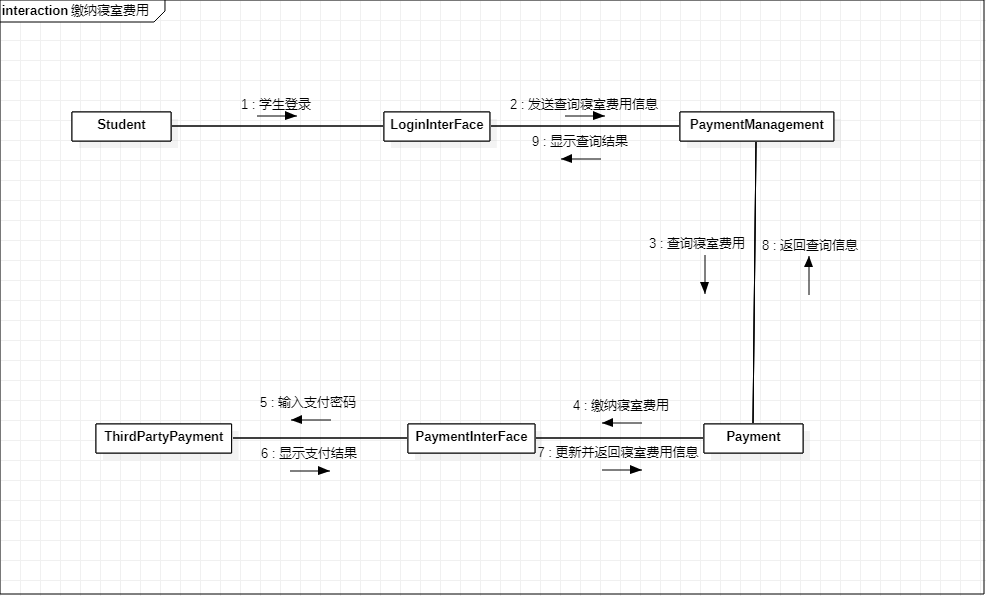


### 4.3.5 宿舍设施检查

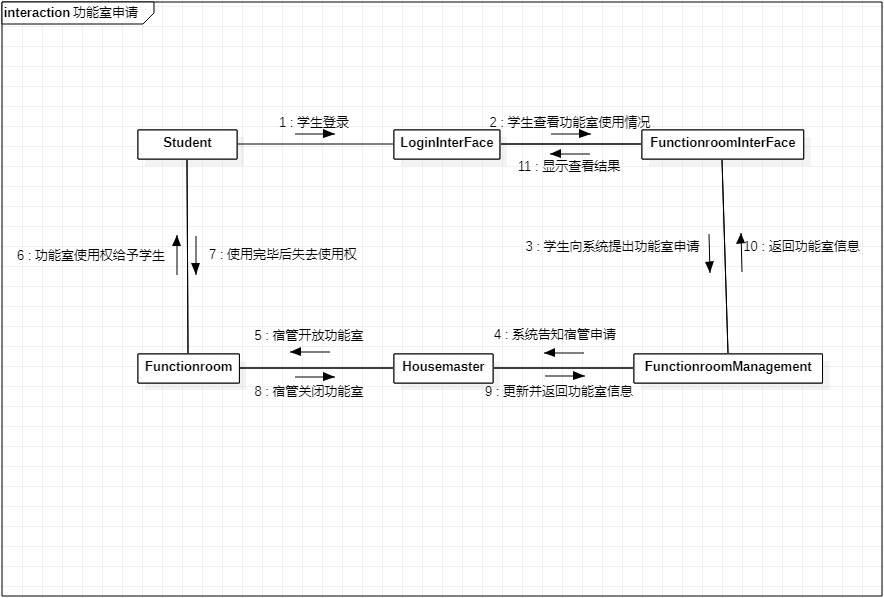
****

## 4.4 通信图

### 4.4.1 缴纳寝室费用



### 4.4.2 功能室申请



# 五、参考资料与说明

## 5.1参考书目

**UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition). Martin Fowler.**

UML Distilled作为我们团队在模型分析过程中制作UML图的主要参考书目，对我们的模型分析过程起到了指导作用，主要参考方面为类图、时序图和通信图的制作。本书的第三章及第五章描述了类的相关属性及类图中附加关键符号的概念，这些概念和使用方法在我们团队的模型分析、构建类图的过程中至关重要。本书第四章介绍了时序图及其构建方法，阐释了单个场景的时序图分析构建过程——通过描述各个对象在用例间的消息传递构建时序图。第十二章详尽地从多个分析角度介绍通信图，包括何时采用通信图描述及两种编号方法。此外，在本书第七章中还介绍了包的概念和使用方法，在我们架构分析的过程中也被一定程度上应用。一个包可以视为一个分组结构，在UML构造中将相关的类和元素打包组合到更高级的单元中，有利于接下来构建过程的条理性和清晰度。

## 5.2 参考文章及期刊

**[1] Layered Application Guidelines in Microsoft Application Architecture Guide**

本文讨论应用程序的整体结构，将组件的逻辑分组分为不同的层，这些层可以相互通信并与其他客户端和应用程序进行通信，有助于我们更好地理解分层体系结构。层关心的是组件和功能的逻辑划分，并不去考虑组件的物理实现。图可以位于不同的层，也可以位于同一层。阅读本文后，我们学习了如何将应用程序划分为不同的逻辑部分，如何为应用程序选择适当的功能布局以及应用程序如何支持多种客户端类型。层次结构图包含这些层的高级表示方法以及它们与用户之间的联系，调用在应用程序业务层中实现的服务的其他应用程序，提供数据访问的关系数据库或 Web 服务等数据源以及提供访问数据的外部或远程服务。

**[2] Domain Driven Design/领域驱动设计（来源于网站）**

Eric Evans的“Domain-Driven Design领域驱动设计”简称DDD，Evans DDD是一套综合软件系统分析和设计的面向对象建模方法，主要应用于比较复杂的需求场景。根据 Eric 的理论，业务层将细分为两个层次：应用层和领域层。应用层：定义软件可以完成的工作，并且指挥具有丰富含义的领域对象来解决问题，保持精练；不包括业务规则或知识，无业务情况的状态；领域层：负责表示业务概念、业务状态的信息和业务规则，是业务软件核心。层次之间必须清晰分离，每个层都是内聚的，并且只依赖它的下层。

**[3] 舒攀,陈金刚.数字化校园建设中宿舍管理系统的设计与实现[J].武汉工程大学学报,2008(04):108-111.**

这篇文章以比较全局的思想，介绍了一款基于web的学生宿舍管理系统，帮助我们系统性地了解学生宿舍管理系统应具备地功能及结构，此外还拓展了宿舍数据共享问题的解决方法。学生宿舍管理系统开发目的是对全校宿舍资源进行统一的分配和管理，实现管理的信息化、网络化、规范化和科学化，提高管理效率，也使得管理工作更加人性化。系统架构包括：安全授权管理、宿舍基础数据管理、宿舍资源分配管理、数据接口等。随后介绍了原型实现和评估工作，展示了解决方案的可行性和效率。我们主要将文章中的人性化思想加入到系统设计中，创新性地开拓了更多便利于学生的服务功能。

**[4]Elena Planas,Jordi Cabot. How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus[J]. Computer Standards &amp; Interfaces,2020,67.**

这篇文章介绍了两种UML建模工具，分别是Magicdraw和Papyrus，虽然我们团队在构建UML模型中使用了starUML工具，但在团队决策使用的UML工具时，这篇文章给予了指导作用。工具使用性的分析方面包括：构建模型所遵循的策略、构建模型所需的工作以及在整个建模过程中出现的障碍。每种工具有其不同的优缺点，我们选择工具的重点放在工具的可用性上，而不是研究每个参与者定义的模型的质量和创造性的变化。经过分析抉择，我们选择上手较易、图形分类清晰、功能齐全、符号标识度高的starUML来完成此次模型分析的任务。

# 六、组员分工

|  |  |
| --- | --- |
| 雷泓  1851202 | 项目简介、参考文献、文档排版、类图、时序图 |
| 王文政  1853928 | 分析机制、类图、时序图、类图整理、文档排版 |
| 肖博阳  1852672 | 类图、时序图、通信图、文档排版 |
| 张明哲  1852452 | 架构分析、类图、时序图、类图整理、类与用例实现 |