UM00 UserManager模块开发报告

UM01 通用功能

1. 项目概述

在本项目中，`UserManager` 类负责管理宿舍管理系统中的所有用户（如学生、管理员等）的相关操作，提供了一些基础的用户管理功能。为了确保系统的灵活性和扩展性，通用功能模块被设计为独立的功能区块，负责处理与数据库交互、验证用户身份、查询数据等常见操作。本章节将详细介绍 `UserManager` 类中的通用功能，包括如何与数据库进行交互、用户操作的验证、以及如何保障用户输入的有效性等。

2. 主要功能及实现步骤

2.1 数据库查询与执行

- **查询并打印 (`query`)**
  该方法用于执行数据库查询，并直接在控制台输出结果。通过传递SQL查询语句，它与数据库交互并显示查询结果。
  实现步骤:
  - 输入SQL查询语句
  - 调用 `db.query(SQL)` 来执行查询
  - 返回并打印查询结果
- **执行 (`execute`)**
  执行不需要返回结果的数据库操作（如插入、更新、删除）。
  实现步骤:
  - 输入SQL操作语句
  - 调用 `db.execute(SQL)` 来执行该操作
- **查询但不打印 (`queryExists`)**
  该方法用于执行查询操作，但不输出结果，仅用于验证某条记录是否存在。
  实现步骤:
  - 输入SQL查询语句
  - 使用 `db.queryExists(SQL)` 判断是否存在相关数据
  - 返回布尔值结果（`true` 或 `false`）

2.2 用户验证与注册

- **一键获取所有用户信息 (`getAllUsers`)**
  查询并显示所有用户的信息。
  实现步骤:
  - 使用 `db.query("SELECT * FROM users;")` 查询所有用户信息
  - 在控制台中打印所有查询结果
- **获取并返回ID (`get_ID`)**
  该方法用于获取学生的学号，首先进行学号输入，并验证学号是否存在。
  实现步骤:
  - 循环要求用户输入学号
  - 检查该学号是否存在（通过调用 `queryExists`）
  - 如果学号存在且有效，返回该学号
- **用户ID存在性检查 (`IDExists`)**
  该方法用于检查指定的学生ID是否存在于数据库中。
  实现步骤:
  - 使用SQL查询验证学生ID是否存在
  - 返回布尔值，表示ID是否有效
- **用户注册 (`registerUser`)**
  用于注册新的用户（学生或管理员），该方法会将用户的基本信息（如学号、姓名、性别等）插入到数据库中。
  实现步骤:
  - 接受用户输入的学号、密码、姓名、性别等信息
  - 使用SQL插入语句，将用户信息插入到 `users` 表
  - 执行插入操作，并根据结果反馈注册成功或失败
- **修改密码 (`userPasswordChange`)**
  该方法允许用户修改其密码。
  实现步骤:
  - 提示用户输入新密码
  - 更新数据库中对应用户的密码字段
  - 根据执行结果输出修改成功或失败的提示

2.3 宿舍管理与查询

- **宿舍是否存在 (`dormitoryExistsByName`)**
  该方法用于检查指定的宿舍名称是否已经存在于数据库中。
  实现步骤:
  - 输入宿舍名称
  - 使用SQL查询检查宿舍楼是否存在
  - 返回布尔值，表示该宿舍楼是否已存在

- **获取宿舍ID（`getDormitoryIDByName`）**
  用于根据宿舍名称获取宿舍的唯一ID。
  实现步骤：
  - 根据宿舍名称查询对应的宿舍ID
  - 如果宿舍存在，返回宿舍ID，否则返回失败信息

2.4 学生入住与选择宿舍

- **是否入住（`isStudentCheckedIn`）**
  该方法用于检查某个学生是否已经入住宿舍。
  实现步骤：
  - 输入学生ID
  - 查询该学生是否已在 `users` 表中标记为已入住
  - 返回布尔值，表示学生是否已入住
- **选择有效的宿舍（`selectValidRoom`）**
  用于学生选择合适的宿舍楼及房间。
  实现步骤：
  - 首先查询所有符合性别要求并且有空房间的宿舍楼
  - 提示用户输入宿舍楼名称，验证该名称是否合法并且有空房
  - 根据宿舍楼名称查询空房间，并允许学生选择房间号
  - 验证房间号的有效性，确保该房间为空且符合要求

2.5 用户登录

- **用户登录（`loginUser`）**
  用于用户登录验证，学生或管理员通过输入学号和密码进行登录。
  实现步骤：
  - 输入用户ID和密码
  - 查询数据库验证用户的ID、密码和身份（管理员或普通用户）
  - 如果验证成功，返回登录成功信息；否则返回失败提示

3. 遇到的问题与解决方案

3.1 问题：用户输入的验证与错误处理

在实现学生注册、登录、修改密码等功能时，涉及用户输入数据的验证（如学号是否已存在、密码格式是否正确等）。
解决方案：

- 为每个功能添加了循环和条件判断，确保用户输入的数据有效。

- 通过数据库查询操作验证学号、用户名等是否已存在，防止重复注册。

3.2 问题：数据库查询的性能与异常处理

在处理查询时，由于需要对数据库进行多次查询，可能会影响系统的性能，尤其在大量数据操作时可能会出现延迟或错误。
解决方案：

- 优化了SQL查询语句，避免多余的查询和重复操作。
- 加强了异常处理机制，避免数据库查询失败时程序崩溃。

3.3 问题：用户选择宿舍时的数据一致性

在宿舍选择功能中，学生可以选择宿舍楼及房间。然而，如果多个学生同时选择相同的房间，可能会造成数据不一致。
解决方案：

- 在选择宿舍时，增加了房间的实时查询，确保学生选择的房间在其操作时是空闲的。
- 使用了锁机制保证房间选择的同步性。

4. 收获总结

- **数据库操作的熟练度提高**：通过频繁的数据库查询、插入和更新操作，增强了对SQL语句的掌握和优化能力。
- **用户体验的优化**：通过增强错误处理、输入验证等措施，提升了用户操作的流畅性和系统稳定性。
- **系统的扩展性设计**：在开发过程中，考虑了未来可能需要扩展的功能，尤其是在用户管理和宿舍管理的设计上，确保了系统具有较好的扩展性。

通过这部分功能的开发，系统的基础功能得到了完善，提供了一个稳定的框架，支持学生和管理员的日常操作，同时也为后续功能的扩展提供了良好的基础。