图书管理系统详细设计文档

一、引言

本图书管理系统旨在满足图书馆对于图书信息管理的基本需求，提供高效、便捷的图书管理功能，包括图书的添加、查询和删除操作，提升图书馆管理效率和服务质量，实现图书资源的信息化管理。

二、系统架构

1. 分层架构

- 表现层：负责与用户进行交互，接收用户输入并展示系统输出。通过命令行界面实现，用户在此层输入操作指令及相关图书信息，系统反馈操作结果及提示信息。

- 业务逻辑层：包含图书管理的核心业务逻辑，如 addBook 、 searchBook 、 deleteBook 等方法。协调数据存储层与表现层之间的数据交互，根据用户操作调用相应的数据处理方法，对图书数据进行增删查等操作，并将处理结果返回给表现层。

- 数据存储层：目前使用 ArrayList<Book> 存储图书对象，实现简单的数据持久化。负责管理图书数据的存储、读取和更新，为业务逻辑层提供数据支持，未来可扩展为数据库存储，以增强数据管理能力和系统性能。

2. 模块关系

- 表现层依赖业务逻辑层，通过调用业务逻辑层的方法实现用户操作功能，并将业务逻辑层返回的结果展示给用户。

- 业务逻辑层依赖数据存储层，对数据存储层中的图书数据进行操作，同时将操作结果反馈给表现层。

三、功能模块设计

1. 添加图书模块

- 功能：实现向系统中添加新图书的功能，确保图书信息准确录入。

- 输入：用户在命令行输入图书名称和作者信息，由 Scanner 类获取。

- 处理流程：

- 创建 Scanner 对象，接收用户输入的图书名称和作者信息，分别存储为 bookName 和 author 。

- 利用接收到的信息创建 Book 对象 new Book(bookName, author) 。

- 将 Book 对象添加到 books 列表中，完成图书信息的存储。

- 向用户输出“图书添加成功！”提示信息。

2. 查询图书模块

- 功能：根据用户输入的图书名称，在系统中查找并返回该图书的详细信息，便于用户确认图书在库情况。

- 输入：用户在命令行输入要查询的图书名称，由 Scanner 类获取并存储为 searchKey 。

- 处理流程：

- 遍历 books 列表，对每个 Book 对象，判断其 name 属性是否与 searchKey 相等。

- 若找到匹配图书，输出“图书名称: [图书名称], 作者: [作者名称]”格式的图书信息，并结束查询流程。

- 若遍历完未找到匹配图书，输出“未找到该图书”提示信息。

3. 删除图书模块

- 功能：按照用户输入的图书名称，从系统中删除指定图书信息，保证系统数据与实际馆藏一致。

- 输入：用户在命令行输入要删除的图书名称，由 Scanner 类获取并存储为 delKey 。

- 处理流程：

- 遍历 books 列表，对每个 Book 对象，判断其 name 属性是否与 delKey 相等。

- 若找到匹配图书，使用 books.remove(i) 方法删除该图书对象（ i 为当前图书在列表中的索引），输出“图书删除成功”提示信息，并结束删除流程。

- 若遍历完未找到匹配图书，输出“未找到该图书，无法删除”提示信息。

四、数据设计

1. Book 类

- 属性：

-  name ：字符串类型，用于存储图书名称，作为图书的唯一标识，方便在查询和删除操作中进行匹配。

-  author ：字符串类型，用于存储图书作者信息，完善图书的基本描述。

- 方法：

- 构造函数 public Book(String name, String author) ：用于创建 Book 对象，初始化 name 和 author 属性，确保对象创建时具有完整的图书信息。

2. 数据存储结构

-  ArrayList<Book> books ：定义在 LibraryManagementSystem 类中，作为静态变量，用于存储所有图书对象。通过 ArrayList 的方法实现图书数据的添加、删除、查询等操作，满足系统对图书数据的管理需求。

五、界面设计

1. 主菜单界面：以命令行形式展示系统的操作选项，包括“1. 添加图书”“2. 查询图书”“3. 删除图书”“4. 退出”，提示用户输入操作选择。

2. 操作提示界面：在执行添加、查询、删除操作时，根据操作流程显示相应的提示信息，如添加图书时提示“请输入图书名称: ”“请输入图书作者: ”，查询图书时提示“请输入要查询的图书名称: ”，删除图书时提示“请输入要删除的图书名称: ”，以及操作成功或失败后的提示信息，如“图书添加成功！”“未找到该图书”“图书删除成功”“未找到该图书，无法删除”等。

六、系统安全设计

1. 输入验证：在获取用户输入的图书名称和作者信息时，对输入内容进行基本的验证，确保输入不为空字符串，避免因空值导致的系统异常。同时，可考虑对输入长度进行限制，防止过长输入影响系统性能和稳定性。

2. 异常处理：在系统运行过程中，对可能出现的异常情况进行捕获和处理，如 NullPointerException （当操作空对象时可能出现）、 IndexOutOfBoundsException （在 ArrayList 操作中索引越界时可能出现）等。通过合理的异常处理机制，确保系统在遇到异常情况时能够保持稳定运行，避免因异常导致系统崩溃，并向用户提供友好的错误提示信息，便于用户了解问题所在并采取相应措施。

七、系统部署设计

1. 运行环境：系统基于 Java 语言开发，需要在安装了 Java 运行时环境（JRE）的计算机上运行，支持 Windows、Linux、Mac OS 等主流操作系统。

2. 部署方式：将编译后的 Java 字节码文件打包成 jar 文件，通过命令行执行 java -jar [jar 文件名称].jar 命令启动图书管理系统，方便在不同环境中部署和运行。同时，可考虑将系统部署在服务器上，通过网络访问实现多用户并发操作，提高系统的可用性和实用性。

八、系统测试设计

1. 功能测试：

- 测试用例：

- 验证添加图书功能：输入不同的图书名称和作者信息，检查图书是否正确添加到 books 列表中，且添加后能通过查询功能找到。

- 验证查询图书功能：输入已添加图书的名称、部分名称、不存在的图书名称，检查查询结果是否符合预期，即准确找到或提示未找到。

- 验证删除图书功能：输入已添加图书的名称、不存在的图书名称，检查图书是否能正确删除或返回无法删除的提示，且删除后不能再通过查询功能找到。

- 预期结果：各项功能操作应准确无误，系统能按照设计要求正确执行添加、查询、删除操作，并给出相应的正确提示信息，保证系统功能的正确性和稳定性。

2. 性能测试：

- 测试场景：模拟大量图书的添加、查询和删除操作，如添加 1000 本图书后进行频繁的查询和删除操作，观察系统的响应时间和资源占用情况。

- 预期结果：系统在高负载情况下应保持稳定运行，响应时间在可接受范围内（例如，查询操作的平均响应时间不超过 2 秒），内存使用合理，不出现明显的卡顿、内存溢出或其他性能问题，确保系统在实际使用中能够满足一定规模的图书管理需求。

3. 安全测试：

- 测试用例：

- 输入非法数据，如空字符串、超长字符串（超过数据库字段限制长度）、特殊字符等作为图书名称和作者信息，检查系统的输入验证机制是否有效，能否正确处理非法输入，不导致系统崩溃或出现安全漏洞。

- 在系统运行过程中，模拟网络中断、突然断电等异常情况，恢复后检查数据的完整性和一致性，确保系统具有一定的容错能力和数据恢复能力。

- 预期结果：系统能够正确识别和处理非法输入，不出现异常错误或安全漏洞，在异常情况下能保证数据的完整性和可恢复性，确保系统的安全性和稳定性，保护图书数据的安全。