图书馆管理系统实验报告

3230105688 李幸轩

2025 年 4 月 17 日

目录

1 系统架构描述

1.1 整体架构

本图书馆管理系统采用三层架构设计：

1. **表示层（UI 层）**：使用 Java Swing 构建图形用户界面，包括登录窗口、主窗口和各功能模块面板
2. **业务逻辑层**：包含数据访问对象（DAO）实现业务逻辑处理
3. **数据访问层**：通过 JDBC 连接 MySQL 数据库，实现数据持久化

系统各组件关系如下图所示：

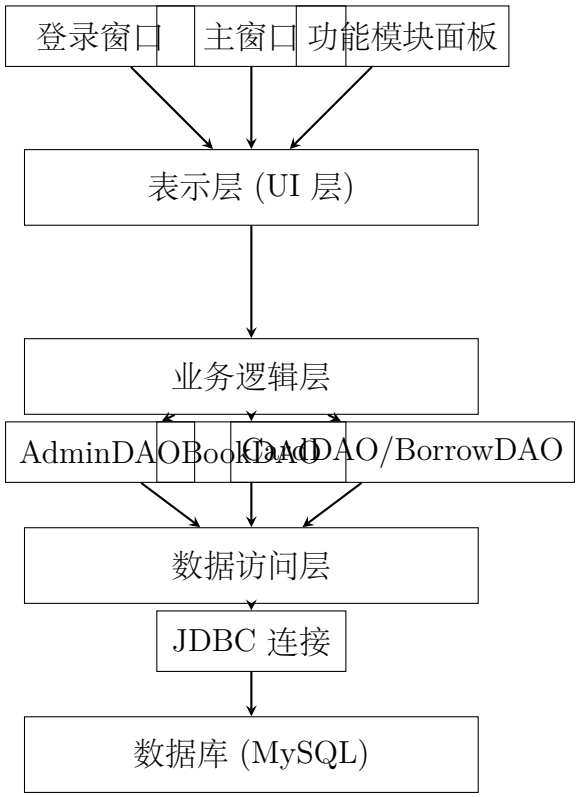

图 1: 系统架构图

1.2 模块划分

系统功能模块划分如下：

1. **管理员登录模块**：验证管理员身份
2. **图书入库模块**：支持单本和批量图书入库

3. **图书查询模块**：支持多条件组合查询
4. **借书模块**：处理借书操作
5. **还书模块**：处理还书操作
6. **借书证管理模块**：支持借书证的添加和删除

1.3 包结构

系统代码按功能划分为以下包：

- `com.library.main`: 系统入口类
- `com.library.model`: 实体类（Book, Admin, Card, BorrowRecord）
- `com.library.dao`: 数据访问对象类
- `com.library.ui`: 用户界面类
- `com.library.util`: 工具类（数据库连接）

下面是系统入口类的代码实现示例：

```
/**
 * 图书馆管理系统 - 主类
 * 系统入口点
 */
package com.library.main;

import com.library.ui.LoginFrame;

public class LibrarySystem {
    public static void main(String[] args) {
        // 创建并显示登录窗口
        // 使用EventQueue.invokeLater确保UI组件在事件调度线程中创建
        java.awt.EventQueue.invokeLater(() -> {
            new LoginFrame().setVisible(true);  // 实例化登录窗口并显示
        });
    }
}
```

Listing 1: 系统入口类实现

2　模块流程图

2.1　管理员登录模块

管理员登录流程如下图所示：

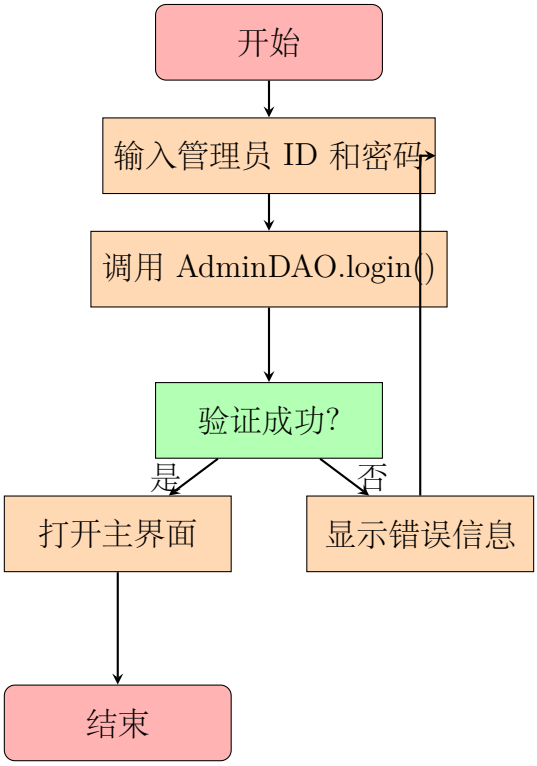

图 2: 管理员登录流程图

登录模块代码实现示例：

```
/**
 * 登录窗口界面
 */
package com.library.ui;

import com.library.dao.AdminDAO;
import com.library.model.Admin;
import com.library.util.DBUtil;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class LoginFrame extends JFrame {
    private JTextField txtAdminId;         // 管理员ID输入框
```

```
17      private JPasswordField txtPassword;    // 密码输入框
18      private JButton btnLogin;              // 登录按钮
19
20      public LoginFrame() {
21          // 初始化数据库
22          DBUtil.initDatabase();  // 确保数据库表结构已创建
23
24          // 设置窗口标题和大小
25          setTitle("图书馆管理系统 - 登录");
26          setSize(400, 250);
27          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
28          setLocationRelativeTo(null);  // 窗口居中显示
29
30          // 创建登录面板
31          JPanel panel = new JPanel();
32          panel.setLayout(new GridLayout(3, 2, 10, 20));
33          panel.setBorder(BorderFactory.createEmptyBorder(30, 30, 30, 30));
34
35          // 添加组件
36          panel.add(new JLabel("管理员ID: "));
37          txtAdminId = new JTextField(20);
38          panel.add(txtAdminId);
39
40          panel.add(new JLabel("密码: "));
41          txtPassword = new JPasswordField(20);
42          panel.add(txtPassword);
43
44          panel.add(new JLabel(""));
45          btnLogin = new JButton("登录");
46          panel.add(btnLogin);
47
48          // 添加登录按钮事件
49          btnLogin.addActionListener(new ActionListener() {
50              @Override
51              public void actionPerformed(ActionEvent e) {
52                  String adminId = txtAdminId.getText();
53                  String password = new String(txtPassword.getPassword());
54
55                  // 验证输入是否为空
56                  if (adminId.isEmpty() || password.isEmpty()) {
57                      JOptionPane.showMessageDialog(LoginFrame.this,
58                          "请输入管理员ID和密码", "错误", JOptionPane.
    ERROR_MESSAGE);
59                      return;
60                  }
```

```
61
62                  // 调用DAO进行登录验证
63                  AdminDAO adminDAO = new AdminDAO();
64                  Admin admin = adminDAO.login(adminId, password);
65
66                  if (admin != null) {
67                      // 登录成功，打开主界面
68                      MainFrame mainFrame = new MainFrame(admin);
69                      mainFrame.setVisible(true);
70                      dispose(); // 关闭登录窗口
71                  } else {
72                      JOptionPane.showMessageDialog(LoginFrame.this,
73                          "管理员ID或密码错误", "登录失败", JOptionPane.
    ERROR_MESSAGE);
74                  }
75              }
76          });
77
78          // 添加面板到窗口
79          add(panel);
80      }
81  }
```

Listing 2: LoginFrame 类实现

2.2 图书入库模块

图书入库流程（包括单本和批量入库）如下图所示：

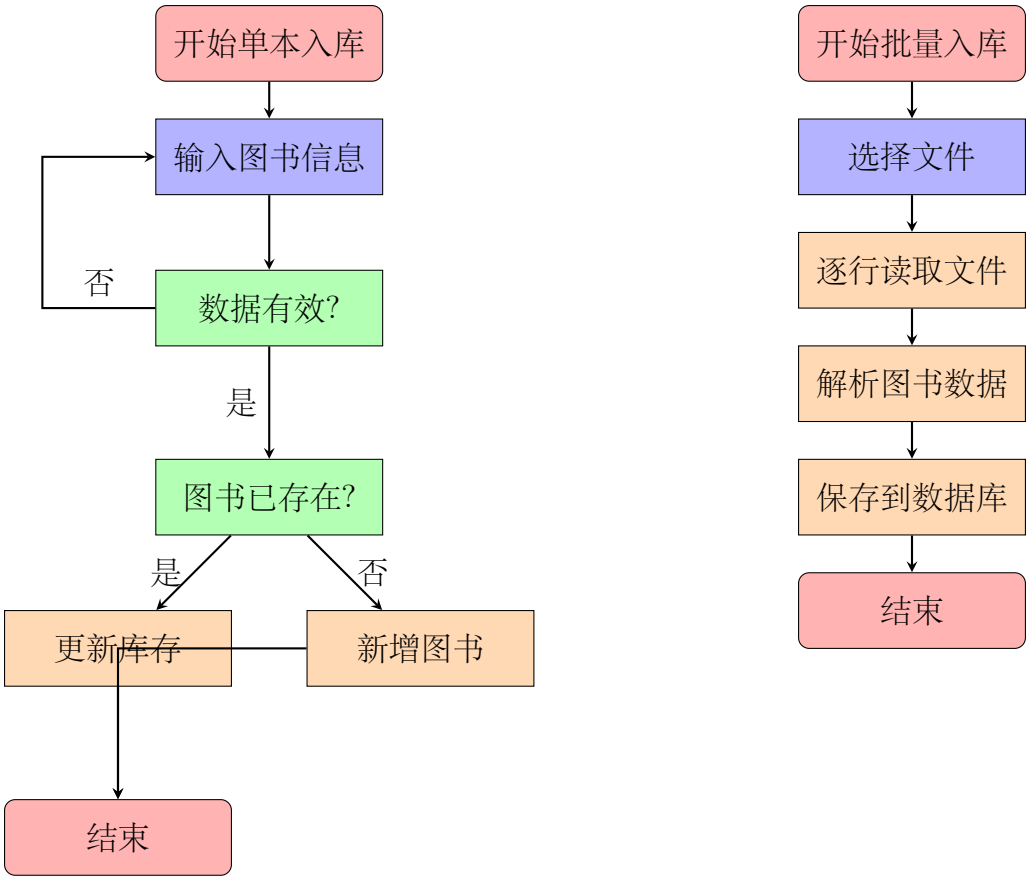

图 3: 图书入库流程图

BookDAO 类中的 addBook 方法代码实现示例:

```
/**
 * 添加图书
 * @param book 图书对象
 * @return 操作是否成功
 */
public boolean addBook(Book book) {
    Connection conn = null;
    PreparedStatement stmt = null;
    boolean success = false;

    try {
        conn = DBUtil.getConnection();

        // 检查书号是否已存在
        String checkSql = "SELECT * FROM Book WHERE book_id = ?";
        PreparedStatement checkStmt = conn.prepareStatement(checkSql);
        checkStmt.setString(1, book.getBookId());
        ResultSet rs = checkStmt.executeQuery();

```

```
20          if (rs.next()) {
21              // 图书已存在，更新库存和总量
22              String updateSql = "UPDATE Book SET total = total + ?, stock =
    stock + ? WHERE book_id = ?";
23              stmt = conn.prepareStatement(updateSql);
24              stmt.setInt(1, book.getTotal());       // 增加总藏书量
25              stmt.setInt(2, book.getTotal());       // 增加库存
26              stmt.setString(3, book.getBookId());   // 按书号查找
27          } else {
28              // 图书不存在，添加新记录
29              String insertSql = "INSERT INTO Book (book_id, category, title,
     publisher, " +
30                                 "publish_year, author, price, total, stock) "
     +
31                                 "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
32              stmt = conn.prepareStatement(insertSql);
33              stmt.setString(1, book.getBookId());
34              stmt.setString(2, book.getCategory());
35              stmt.setString(3, book.getTitle());
36              stmt.setString(4, book.getPublisher());
37              stmt.setInt(5, book.getPublishYear());
38              stmt.setString(6, book.getAuthor());
39              stmt.setBigDecimal(7, book.getPrice());
40              stmt.setInt(8, book.getTotal());
41              stmt.setInt(9, book.getTotal()); // 初始库存等于总量
42          }
43
44          rs.close();
45          checkStmt.close();
46
47          // 执行SQL语句
48          int rowsAffected = stmt.executeUpdate();
49          success = (rowsAffected > 0); // 影响的行数大于0表示操作成功
50      } catch (SQLException e) {
51          e.printStackTrace(); // 打印异常信息
52      } finally {
53          // 关闭资源
54          DBUtil.closeResources(conn, stmt, null);
55      }
56
57      return success;
58  }
```

Listing 3: BookDAO.addBook 方法实现

批量入库方法实现：

```java
/**
 * 批量入库方法
 * 从文件读取图书信息并保存到数据库
 */
private void addBatchBooks() {
    String filePath = txtFilePath.getText().trim();

    if (filePath.isEmpty()) {
        JOptionPane.showMessageDialog(this, "请选择文件", "错误", JOptionPane.ERROR_MESSAGE);
        return;
    }

    try {
        File file = new File(filePath);
        BufferedReader reader = new BufferedReader(new FileReader(file));
        List<Book> books = new ArrayList<>();
        String line;
        int lineCount = 0;

        txtResult.setText("开始导入图书...\n");

        while ((line = reader.readLine()) != null) {
            lineCount++;
            try {
                // 解析行数据格式: (book_id, category, title, publisher, year, author, price, quantity)
                line = line.trim();

                // 检查是否是空行
                if (line.isEmpty()) {
                    continue;
                }

                // 移除开头的"("和结尾的")"
                if (line.startsWith("(") && line.endsWith(")")) {
                    line = line.substring(1, line.length() - 1);
                }

                // 分割字段
                String[] fields = line.split(",");

                if (fields.length != 8) {
                    txtResult.append("第" + lineCount + "行格式错误, 应包含8个字段\n");
```

```
                    continue;
                }

                // 提取和转换字段
                String bookId = fields[0].trim();
                String category = fields[1].trim();
                String title = fields[2].trim();
                String publisher = fields[3].trim();
                int year = Integer.parseInt(fields[4].trim());
                String author = fields[5].trim();
                BigDecimal price = new BigDecimal(fields[6].trim());
                int quantity = Integer.parseInt(fields[7].trim());

                // 创建图书对象
                Book book = new Book(bookId, category, title, publisher,
    year, author, price, quantity, quantity);
                books.add(book);

                txtResult.append("解析第" + lineCount + "行: " + title + "\
    n");
            } catch (Exception e) {
                txtResult.append("解析第" + lineCount + "行出错: " + e.
    getMessage() + "\n");
            }
        }

        reader.close();

        if (books.isEmpty()) {
            txtResult.append("没有有效的图书数据\n");
            return;
        }

        // 保存到数据库
        BookDAO bookDAO = new BookDAO();
        int successCount = bookDAO.addBooks(books);

        txtResult.append("批量导入完成: 共" + books.size() + "条记录, 成功"
     + successCount + "条\n");
    } catch (Exception e) {
        txtResult.append("批量导入失败: " + e.getMessage() + "\n");
    }
}
```

Listing 4: 批量图书入库方法实现

2.3 图书查询模块

图书查询流程如下图所示：

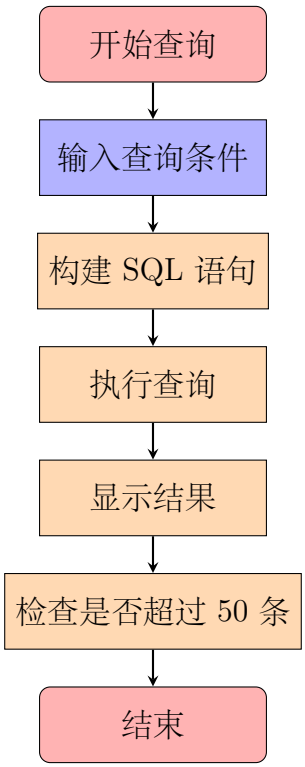

图 4: 图书查询流程图

图书查询代码实现示例：

```
/**
 * 查询图书
 * 支持多条件组合查询
 */
public List<Book> searchBooks(String category, String title, String publisher,
                              Integer minYear, Integer maxYear, String author,
                              BigDecimal minPrice, BigDecimal maxPrice, String sortBy) {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    List<Book> books = new ArrayList<>();

    try {
        conn = DBUtil.getConnection();

        // 使用StringBuilder动态构建SQL语句
```

```
17          StringBuilder sqlBuilder = new StringBuilder("SELECT * FROM Book
    WHERE 1=1");
18          List<Object> params = new ArrayList<>(); // 存储SQL参数
19
20          // 根据条件动态添加WHERE子句
21          if (category != null && !category.isEmpty()) {
22              sqlBuilder.append(" AND category LIKE ?");  // 使用LIKE实现模糊
    查询
23              params.add("%" + category + "%");
24          }
25
26          if (title != null && !title.isEmpty()) {
27              sqlBuilder.append(" AND title LIKE ?");
28              params.add("%" + title + "%");
29          }
30
31          if (publisher != null && !publisher.isEmpty()) {
32              sqlBuilder.append(" AND publisher LIKE ?");
33              params.add("%" + publisher + "%");
34          }
35
36          if (minYear != null) {
37              sqlBuilder.append(" AND publish_year >= ?");
38              params.add(minYear);
39          }
40
41          if (maxYear != null) {
42              sqlBuilder.append(" AND publish_year <= ?");
43              params.add(maxYear);
44          }
45
46          if (author != null && !author.isEmpty()) {
47              sqlBuilder.append(" AND author LIKE ?");
48              params.add("%" + author + "%");
49          }
50
51          if (minPrice != null) {
52              sqlBuilder.append(" AND price >= ?");
53              params.add(minPrice);
54          }
55
56          if (maxPrice != null) {
57              sqlBuilder.append(" AND price <= ?");
58              params.add(maxPrice);
59          }
```

```
60
61          // 添加排序子句
62          if (sortBy == null || sortBy.isEmpty()) {
63              sortBy = "title"; // 默认按书名排序
64          }
65          sqlBuilder.append(" ORDER BY ").append(sortBy);
66
67          // 限制返回最多50条记录
68          sqlBuilder.append(" LIMIT 50");
69
70          stmt = conn.prepareStatement(sqlBuilder.toString());
71
72          // 设置SQL参数
73          for (int i = 0; i < params.size(); i++) {
74              stmt.setObject(i + 1, params.get(i));
75          }
76
77          rs = stmt.executeQuery();
78
79          // 处理查询结果
80          while (rs.next()) {
81              Book book = new Book();
82              book.setBookId(rs.getString("book_id"));
83              book.setCategory(rs.getString("category"));
84              book.setTitle(rs.getString("title"));
85              book.setPublisher(rs.getString("publisher"));
86              book.setPublishYear(rs.getInt("publish_year"));
87              book.setAuthor(rs.getString("author"));
88              book.setPrice(rs.getBigDecimal("price"));
89              book.setTotal(rs.getInt("total"));
90              book.setStock(rs.getInt("stock"));
91              books.add(book);
92          }
93      } catch (SQLException e) {
94          e.printStackTrace();
95      } finally {
96          DBUtil.closeResources(conn, stmt, rs);
97      }
98
99      return books;
100 }
```

Listing 5: BookDAO.searchBooks 方法实现

2.4 借书模块

借书流程如下图所示：

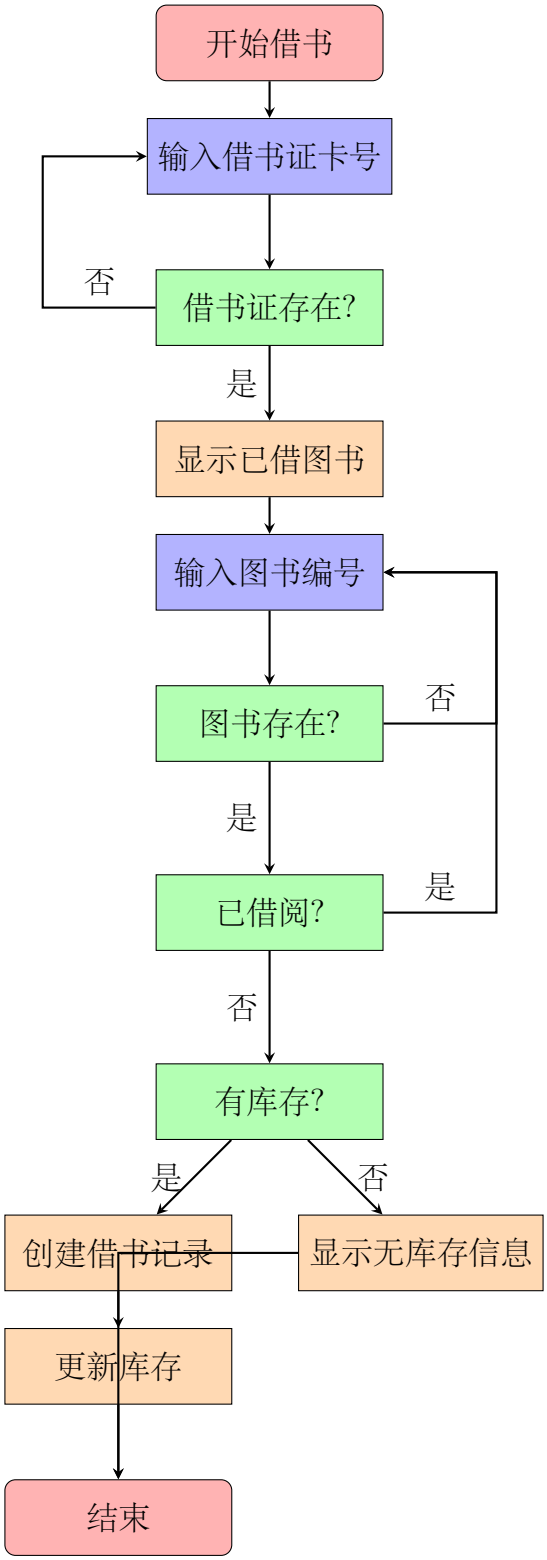

图 5: 借书流程图

借书模块代码实现示例：

```java
/**
 * 借书方法
 * 处理借书操作，包括验证借书证和图书，创建借书记录和更新库存
 */
private void borrowBook() {
    String cardId = txtCardId.getText().trim();
    String bookId = txtBookId.getText().trim();

    // 验证输入
    if (cardId.isEmpty() || bookId.isEmpty()) {
        JOptionPane.showMessageDialog(this, "请输入借书证卡号和图书编号", "错误", JOptionPane.ERROR_MESSAGE);
        return;
    }

    // 检查借书证是否存在
    CardDAO cardDAO = new CardDAO();
    Card card = cardDAO.getCardById(cardId);

    if (card == null) {
        JOptionPane.showMessageDialog(this, "借书证不存在", "错误", JOptionPane.ERROR_MESSAGE);
        return;
    }

    // 检查图书是否存在
    BookDAO bookDAO = new BookDAO();
    Book book = bookDAO.getBookById(bookId);

    if (book == null) {
        JOptionPane.showMessageDialog(this, "图书不存在", "错误", JOptionPane.ERROR_MESSAGE);
        return;
    }

    // 检查图书是否已被该借书证借出
    BorrowDAO borrowDAO = new BorrowDAO();
    if (borrowDAO.isBookBorrowed(bookId, cardId)) {
        JOptionPane.showMessageDialog(this, "该图书已被您借出", "错误", JOptionPane.ERROR_MESSAGE);
        return;
    }

    // 检查库存
    if (book.getStock() <= 0) {
```

```
42          // 查询最近归还时间
43          Date latestReturnDate = bookDAO.getLatestReturnDate(bookId);
44          String returnDateStr = "暂无归还记录";
45
46          if (latestReturnDate != null) {
47              SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
48              returnDateStr = sdf.format(latestReturnDate);
49          }
50
51          JOptionPane.showMessageDialog(this,
52              "该图书暂无库存，最近归还时间：" + returnDateStr, "提示",
    JOptionPane.INFORMATION_MESSAGE);
53          return;
54      }
55
56      // 创建借书记录
57      BorrowRecord record = new BorrowRecord();
58      record.setBookId(bookId);
59      record.setCardId(cardId);
60      record.setBorrowDate(new Date());  // 当前日期为借书日期
61      record.setAdminId(admin.getAdminId());  // 设置经手人
62
63      // 保存借书记录
64      boolean success1 = borrowDAO.addBorrowRecord(record);
65
66      // 更新库存（减1）
67      boolean success2 = bookDAO.updateBookStock(bookId, -1);
68
69      if (success1 && success2) {
70          JOptionPane.showMessageDialog(this, "借书成功", "提示", JOptionPane
    .INFORMATION_MESSAGE);
71          txtBookId.setText("");  // 清空图书编号输入框
72          checkBorrowedBooks();   // 刷新已借图书列表
73      } else {
74          JOptionPane.showMessageDialog(this, "借书失败", "错误", JOptionPane
    .ERROR_MESSAGE);
75      }
76  }
```

Listing 6: 借书方法实现

2.5 还书模块

还书流程如下图所示：

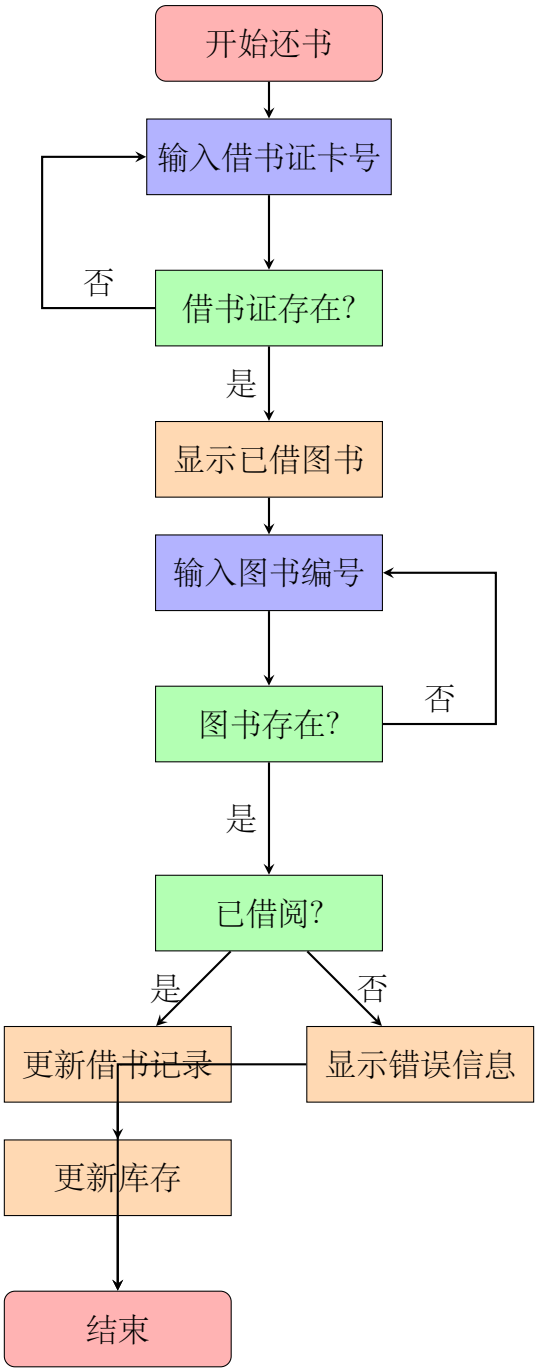

图 6: 还书流程图

还书模块代码实现示例:

```
/**
 * 还书方法
 * 处理还书操作，包括验证借书证和图书，更新借书记录和库存
 */
private void returnBook() {
    String cardId = txtCardId.getText().trim();
    String bookId = txtBookId.getText().trim();

```

```
9       // 验证输入
10      if (cardId.isEmpty() || bookId.isEmpty()) {
11          JOptionPane.showMessageDialog(this, "请输入借书证卡号和图书编号", "
    错误", JOptionPane.ERROR_MESSAGE);
12          return;
13      }
14
15      // 检查借书证是否存在
16      CardDAO cardDAO = new CardDAO();
17      Card card = cardDAO.getCardById(cardId);
18
19      if (card == null) {
20          JOptionPane.showMessageDialog(this, "借书证不存在", "错误",
    JOptionPane.ERROR_MESSAGE);
21          return;
22      }
23
24      // 检查图书是否存在
25      BookDAO bookDAO = new BookDAO();
26      Book book = bookDAO.getBookById(bookId);
27
28      if (book == null) {
29          JOptionPane.showMessageDialog(this, "图书不存在", "错误",
    JOptionPane.ERROR_MESSAGE);
30          return;
31      }
32
33      // 检查图书是否已被借出
34      BorrowDAO borrowDAO = new BorrowDAO();
35      if (!borrowDAO.isBookBorrowed(bookId, cardId)) {
36          JOptionPane.showMessageDialog(this, "该图书未被该借书证借出", "错误
    ", JOptionPane.ERROR_MESSAGE);
37          return;
38      }
39
40      // 更新借书记录（设置还书日期）
41      boolean success1 = borrowDAO.updateReturnRecord(bookId, cardId, new
    Date());
42
43      // 更新库存（加1）
44      boolean success2 = bookDAO.updateBookStock(bookId, 1);
45
46      if (success1 && success2) {
47          JOptionPane.showMessageDialog(this, "还书成功", "提示", JOptionPane
    .INFORMATION_MESSAGE);
```

```
48          txtBookId.setText("");  // 清空图书编号输入框
49          checkBorrowedBooks();   // 刷新已借图书列表
50      } else {
51          JOptionPane.showMessageDialog(this, "还书失败", "错误", JOptionPane
    .ERROR_MESSAGE);
52      }
53  }
```

Listing 7: 还书方法实现

更新还书记录的 DAO 方法实现:

```
1   /**
2    * 更新还书记录
3    * @param bookId 图书ID
4    * @param cardId 借书证ID
5    * @param returnDate 还书日期
6    * @return 操作是否成功
7    */
8   public boolean updateReturnRecord(String bookId, String cardId, Date
    returnDate) {
9       Connection conn = null;
10      PreparedStatement stmt = null;
11      boolean success = false;
12
13      try {
14          conn = DBUtil.getConnection();
15          // 更新尚未归还的借书记录
16          String sql = "UPDATE Borrow SET return_date = ? " +
17                       "WHERE book_id = ? AND card_id = ? AND return_date IS
    NULL";
18          stmt = conn.prepareStatement(sql);
19          stmt.setDate(1, new java.sql.Date(returnDate.getTime()));  // 设置
    还书日期
20          stmt.setString(2, bookId);  // 图书ID
21          stmt.setString(3, cardId);  // 借书证ID
22
23          int rowsAffected = stmt.executeUpdate();
24          success = (rowsAffected > 0);  // 影响的行数大于0表示操作成功
25      } catch (SQLException e) {
26          e.printStackTrace();
27      } finally {
28          DBUtil.closeResources(conn, stmt, null);
29      }
30
31      return success;
```

```
}
```

Listing 8: BorrowDAO.updateReturnRecord 方法

2.6 借书证管理模块

借书证管理流程如下图所示：

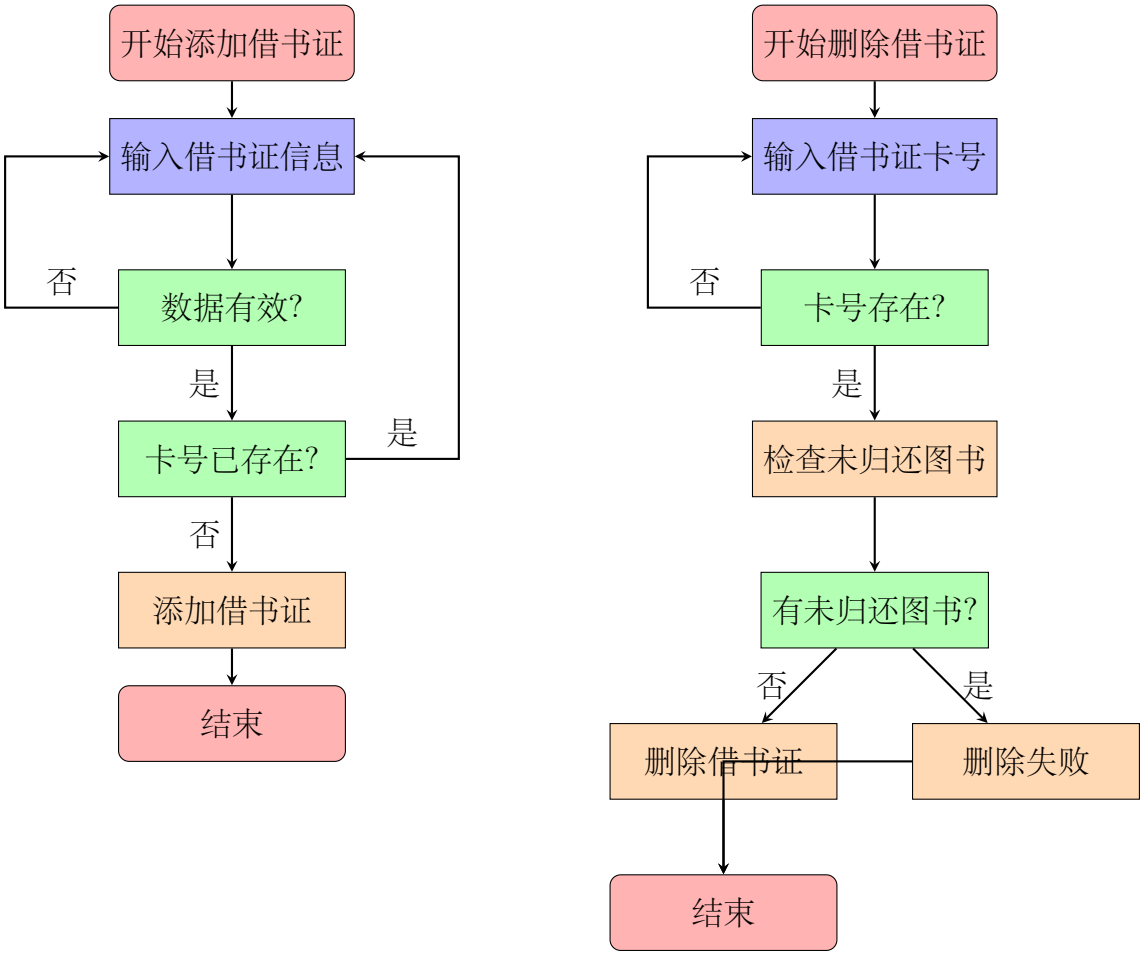

图 7: 借书证管理流程图

借书证管理代码实现示例：

```
/**
 * 添加借书证方法
 * 验证输入并将借书证信息保存到数据库
 */
private void addCard() {
    try {
        // 获取输入信息
        String cardId = txtAddCardId.getText().trim();
        String name = txtAddName.getText().trim();
        String department = txtAddDepartment.getText().trim();
```

```
        String type = (String) cboAddType.getSelectedItem();

        // 检查输入有效性
        if (cardId.isEmpty() || name.isEmpty() || department.isEmpty()) {
            JOptionPane.showMessageDialog(this, "所有字段都不能为空", "错误", JOptionPane.ERROR_MESSAGE);
            return;
        }

        // 检查卡号是否已存在
        CardDAO cardDAO = new CardDAO();
        Card existingCard = cardDAO.getCardById(cardId);

        if (existingCard != null) {
            JOptionPane.showMessageDialog(this, "该卡号已存在", "错误", JOptionPane.ERROR_MESSAGE);
            return;
        }

        // 创建借书证对象
        Card card = new Card(cardId, name, department, type);

        // 保存到数据库
        boolean success = cardDAO.addCard(card);

        if (success) {
            JOptionPane.showMessageDialog(this, "借书证添加成功", "提示", JOptionPane.INFORMATION_MESSAGE);
            clearAddInputs();  // 清空输入
            loadCards();       // 刷新借书证列表
        } else {
            JOptionPane.showMessageDialog(this, "借书证添加失败", "错误", JOptionPane.ERROR_MESSAGE);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "添加失败: " + e.getMessage(), "错误", JOptionPane.ERROR_MESSAGE);
    }
}
```

Listing 9: 添加借书证方法实现

删除借书证方法实现:

```
/**
 * 删除借书证方法
 * 验证借书证并执行删除操作
```

```
 */
private void deleteCard() {
    String cardId = txtDeleteCardId.getText().trim();

    if (cardId.isEmpty()) {
        JOptionPane.showMessageDialog(this, "请输入借书证卡号", "错误",
    JOptionPane.ERROR_MESSAGE);
        return;
    }

    // 检查借书证是否存在
    CardDAO cardDAO = new CardDAO();
    Card card = cardDAO.getCardById(cardId);

    if (card == null) {
        JOptionPane.showMessageDialog(this, "借书证不存在", "错误",
    JOptionPane.ERROR_MESSAGE);
        return;
    }

    // 确认删除
    int confirm = JOptionPane.showConfirmDialog(this,
        "确定要删除卡号为 " + cardId + " 的借书证吗？", "确认删除",
    JOptionPane.YES_NO_OPTION);

    if (confirm == JOptionPane.YES_OPTION) {
        // 删除借书证
        boolean success = cardDAO.deleteCard(cardId);

        if (success) {
            JOptionPane.showMessageDialog(this, "借书证删除成功", "提示",
    JOptionPane.INFORMATION_MESSAGE);
            txtDeleteCardId.setText("");
            loadCards(); // 刷新借书证列表
        } else {
            JOptionPane.showMessageDialog(this,
                "借书证删除失败，可能存在未归还的图书", "错误", JOptionPane
    .ERROR_MESSAGE);
        }
    }
}
```

Listing 10: 删除借书证方法实现

3 数据库表结构

系统使用 MySQL 数据库，包含以下四个表结构：

3.1 Admin 表（管理员）

字段名	数据类型	约束	说明
admin_id	VARCHAR(20)	主键	管理员 ID
password	VARCHAR(50)	非空	密码
name	VARCHAR(50)	非空	姓名
contact	VARCHAR(100)	非空	联系方式

表 1: Admin 表结构

3.2 Book 表（图书）

字段名	数据类型	约束	说明
book_id	VARCHAR(20)	主键	图书 ID
category	VARCHAR(50)	非空	类别
title	VARCHAR(100)	非空	书名
publisher	VARCHAR(100)	非空	出版社
publish_year	INT	非空	出版年份
author	VARCHAR(100)	非空	作者
price	DECIMAL(10,2)	非空	价格
total	INT	非空	总藏书量
stock	INT	非空	库存

表 2: Book 表结构

3.3 Card 表（借书证）

字段名	数据类型	约束	说明
card_id	VARCHAR(20)	主键	借书证 ID
name	VARCHAR(50)	非空	姓名
department	VARCHAR(100)	非空	单位
type	VARCHAR(20)	非空	类别（学生/教师等）

表 3: Card 表结构

3.4 Borrow 表（借书记录）

字段名	数据类型	约束	说明
borrow_id	INT	主键、自增	借书记录 ID
book_id	VARCHAR(20)	外键、非空	图书 ID
card_id	VARCHAR(20)	外键、非空	借书证 ID
borrow_date	DATE	非空	借书日期
return_date	DATE	可空	还书日期
admin_id	VARCHAR(20)	外键、非空	管理员 ID

表 4: Borrow 表结构

数据库 ER 图如下：

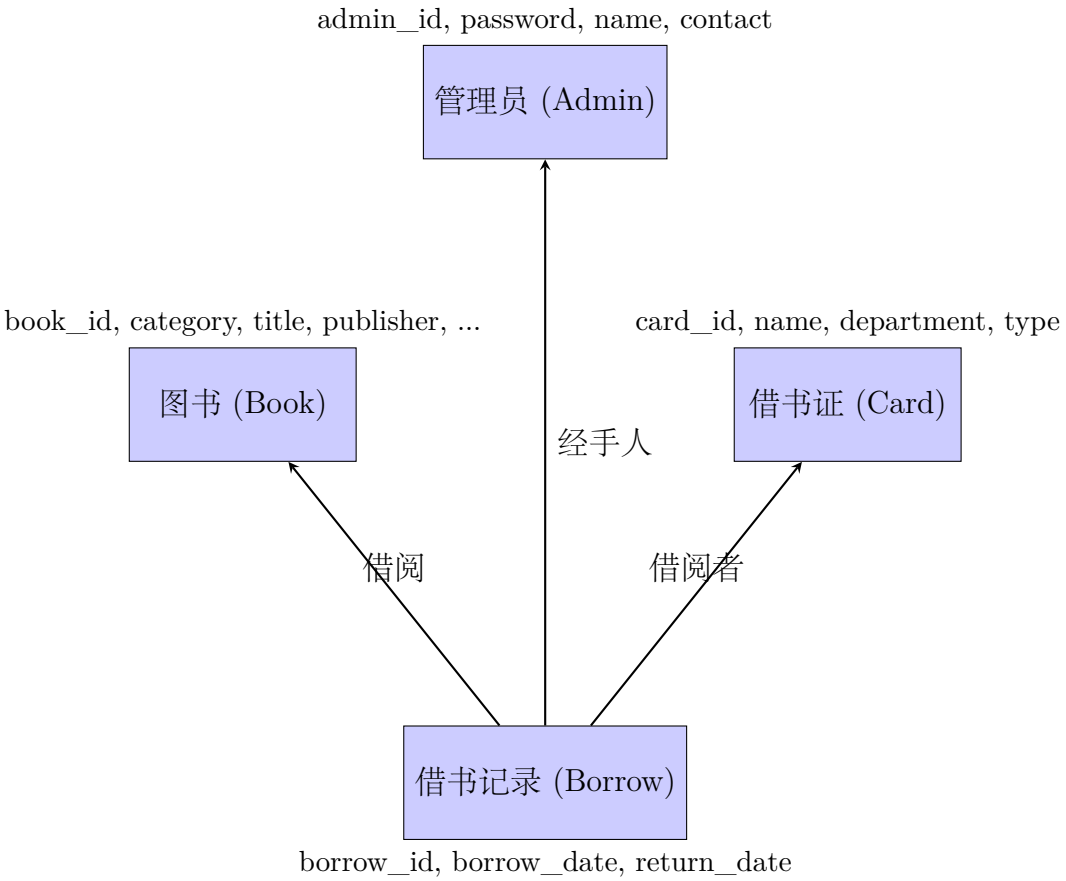

图 8: 数据库 ER 图

数据库表创建代码实现：

```
-- 创建Book表
CREATE TABLE IF NOT EXISTS Book (
    book_id VARCHAR(20) PRIMARY KEY,
```

```
    category VARCHAR(50) NOT NULL,
    title VARCHAR(100) NOT NULL,
    publisher VARCHAR(100) NOT NULL,
    publish_year INT NOT NULL,
    author VARCHAR(100) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    total INT NOT NULL,
    stock INT NOT NULL
);

-- 创建Card表
CREATE TABLE IF NOT EXISTS Card (
    card_id VARCHAR(20) PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    department VARCHAR(100) NOT NULL,
    type VARCHAR(20) NOT NULL
);

-- 创建Admin表
CREATE TABLE IF NOT EXISTS Admin (
    admin_id VARCHAR(20) PRIMARY KEY,
    password VARCHAR(50) NOT NULL,
    name VARCHAR(50) NOT NULL,
    contact VARCHAR(100) NOT NULL
);

-- 创建Borrow表
CREATE TABLE IF NOT EXISTS Borrow (
    borrow_id INT PRIMARY KEY AUTO_INCREMENT,
    book_id VARCHAR(20) NOT NULL,
    card_id VARCHAR(20) NOT NULL,
    borrow_date DATE NOT NULL,
    return_date DATE,
    admin_id VARCHAR(20) NOT NULL,
    FOREIGN KEY (book_id) REFERENCES Book(book_id),
    FOREIGN KEY (card_id) REFERENCES Card(card_id),
    FOREIGN KEY (admin_id) REFERENCES Admin(admin_id)
);
```

Listing 11: 数据库表创建代码

数据库初始化代码实现示例：

```
/**
 * 初始化数据库表
 * 创建系统必要的数据库表结构，并添加默认管理员账号
 */
```

```
public static void initDatabase() {
    Connection conn = null;
    Statement stmt = null;

    try {
        conn = getConnection();
        stmt = conn.createStatement();

        // 创建Book表
        String createBookTable = "CREATE TABLE IF NOT EXISTS Book (" +
                "book_id VARCHAR(20) PRIMARY KEY," +
                "category VARCHAR(50) NOT NULL," +
                "title VARCHAR(100) NOT NULL," +
                "publisher VARCHAR(100) NOT NULL," +
                "publish_year INT NOT NULL," +
                "author VARCHAR(100) NOT NULL," +
                "price DECIMAL(10,2) NOT NULL," +
                "total INT NOT NULL," +
                "stock INT NOT NULL" +
                ")";
        stmt.executeUpdate(createBookTable);

        // 创建Card表
        String createCardTable = "CREATE TABLE IF NOT EXISTS Card (" +
                "card_id VARCHAR(20) PRIMARY KEY," +
                "name VARCHAR(50) NOT NULL," +
                "department VARCHAR(100) NOT NULL," +
                "type VARCHAR(20) NOT NULL" +
                ")";
        stmt.executeUpdate(createCardTable);

        // 创建Admin表
        String createAdminTable = "CREATE TABLE IF NOT EXISTS Admin (" +
                "admin_id VARCHAR(20) PRIMARY KEY," +
                "password VARCHAR(50) NOT NULL," +
                "name VARCHAR(50) NOT NULL," +
                "contact VARCHAR(100) NOT NULL" +
                ")";
        stmt.executeUpdate(createAdminTable);

        // 创建Borrow表
        String createBorrowTable = "CREATE TABLE IF NOT EXISTS Borrow (" +
                "borrow_id INT PRIMARY KEY AUTO_INCREMENT," +
                "book_id VARCHAR(20) NOT NULL," +
                "card_id VARCHAR(20) NOT NULL," +
```

```
                "borrow_date DATE NOT NULL," +
                "return_date DATE," +
                "admin_id VARCHAR(20) NOT NULL," +
                "FOREIGN KEY (book_id) REFERENCES Book(book_id)," +
                "FOREIGN KEY (card_id) REFERENCES Card(card_id)," +
                "FOREIGN KEY (admin_id) REFERENCES Admin(admin_id)" +
                ")";
        stmt.executeUpdate(createBorrowTable);

        // 插入默认管理员账号（仅当Admin表为空时）
        String checkAdmin = "SELECT COUNT(*) FROM Admin";
        ResultSet rs = stmt.executeQuery(checkAdmin);
        rs.next();
        if (rs.getInt(1) == 0) {
            String insertAdmin = "INSERT INTO Admin (admin_id, password,
    name, contact) " +
                    "VALUES ('admin', 'admin123', '系统管理员',
    '13800138000')";
            stmt.executeUpdate(insertAdmin);
        }
        rs.close();

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        closeResources(conn, stmt, null);
    }
}
```

Listing 12: 数据库初始化方法实现

4 技术描述

本系统采用以下技术实现：

4.1 开发语言

- **Java**：使用 Java 语言开发整个系统，利用其面向对象特性实现模块化设计。Java 是一种广泛使用的编程语言，具有跨平台性，适合开发各类应用程序。

4.2 数据库技术

- **MySQL**: 作为后端数据库管理系统，存储所有应用数据。MySQL 是一个开源的关系型数据库管理系统，具有良好的性能和稳定性。
- **JDBC**: Java 数据库连接技术，通过 JDBC API 操作数据库。JDBC 提供了一种标准的方式来连接和操作各种关系型数据库。

JDBC 连接代码实现示例:

```
/**
 * 数据库连接工具类
 * 提供数据库连接和关闭功能
 */
package com.library.util;

import java.sql.*;

public class DBUtil {
    // 数据库连接信息
    private static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    private static final String DB_URL = "jdbc:mysql://localhost:3306/
    library?useSSL=false&serverTimezone=UTC";
    private static final String USER = "root";
    private static final String PASS = "password";

    /**
     * 获取数据库连接
     * @return 数据库连接对象
     */
    public static Connection getConnection() {
        Connection conn = null;
        try {
            // 加载JDBC驱动
            Class.forName(JDBC_DRIVER);
            // 建立连接
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return conn;
    }

    /**
     * 关闭数据库连接和相关资源
     * @param conn 数据库连接
```

```
     * @param stmt SQL语句对象
     * @param rs 结果集
     */
    public static void closeResources(Connection conn, Statement stmt,
    ResultSet rs) {
        try {
            if (rs != null) rs.close();
            if (stmt != null) stmt.close();
            if (conn != null) conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // ... 省略其他方法
}
```

Listing 13: JDBC 连接实现

4.3 图形界面技术

- **Java Swing**: 使用 Swing 组件库构建图形用户界面。Swing 是 Java 基础类库的一部分，提供了丰富的 UI 组件和易于使用的 API。
- **AWT**: 提供基础图形界面功能。AWT 是 Java 的抽象窗口工具包，是 Swing 的底层基础。

主窗口界面代码实现示例：

```
/**
 * 主窗口界面
 */
package com.library.ui;

import com.library.model.Admin;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MainFrame extends JFrame {
    private Admin admin;
    private JTabbedPane tabbedPane;

```

```
17      public MainFrame(Admin admin) {
18          this.admin = admin;
19
20          // 设置窗口标题和大小
21          setTitle("图书馆管理系统 - " + admin.getName());
22          setSize(800, 600);
23          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24          setLocationRelativeTo(null);  // 窗口居中显示
25
26          // 创建选项卡面板
27          tabbedPane = new JTabbedPane();
28
29          // 添加各功能模块面板
30          tabbedPane.addTab("图书入库", new BookAddPanel(admin));
31          tabbedPane.addTab("图书查询", new BookSearchPanel());
32          tabbedPane.addTab("借书", new BorrowPanel(admin));
33          tabbedPane.addTab("还书", new ReturnPanel(admin));
34          tabbedPane.addTab("借书证管理", new CardManagementPanel());
35
36          // 添加选项卡面板到窗口
37          add(tabbedPane);
38
39          // 创建菜单栏
40          JMenuBar menuBar = new JMenuBar();
41          JMenu menuSystem = new JMenu("系统");
42          JMenuItem menuItemLogout = new JMenuItem("注销");
43          JMenuItem menuItemExit = new JMenuItem("退出");
44
45          menuSystem.add(menuItemLogout);
46          menuSystem.add(menuItemExit);
47          menuBar.add(menuSystem);
48
49          // 添加菜单事件
50          menuItemLogout.addActionListener(new ActionListener() {
51              @Override
52              public void actionPerformed(ActionEvent e) {
53                  LoginFrame loginFrame = new LoginFrame();
54                  loginFrame.setVisible(true);
55                  dispose(); // 关闭主窗口
56              }
57          });
58
59          menuItemExit.addActionListener(new ActionListener() {
60              @Override
61              public void actionPerformed(ActionEvent e) {
```

```
                System.exit(0); // 退出程序
            }
        });

        // 设置菜单栏
        setJMenuBar(menuBar);
    }
}
```

Listing 14: 主窗口界面实现

4.4 系统架构模式

- **MVC 模式**: 将系统分为模型 (Model)、视图 (View) 和控制器 (Controller) 三部分，实现关注点分离。
  - **模型 (Model)**: 包括实体类 (Book, Admin, Card, BorrowRecord)
  - **视图 (View)**: 各 UI 界面类 (LoginFrame, MainFrame 等)
  - **控制器 (Controller)**: 数据访问对象类 (BookDAO, AdminDAO 等)
- **DAO 模式**: 使用数据访问对象隔离业务逻辑和数据访问逻辑，提高代码的可维护性和可测试性。

4.5 开发工具

- **IDE**: Eclipse/IntelliJ IDEA，提供代码编辑、调试、运行等功能
- **构建工具**: Maven（管理依赖）
- **数据库工具**: MySQL Workbench（数据库设计与管理）

5 功能实现

5.1 管理员登录

实现了基本的管理员登录验证功能：

- 用户输入管理员 ID 和密码
- 系统验证账号密码是否正确
- 登录成功后进入主界面，失败则提示错误

系统初始化时会自动创建默认管理员账号（admin/admin123）。

AdminDAO 类的 login 方法代码实现：

```
/**
 * 验证管理员登录
 * @param adminId 管理员ID
 * @param password 密码
 * @return 管理员对象，如果验证失败则返回null
 */
public Admin login(String adminId, String password) {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    Admin admin = null;

    try {
        conn = DBUtil.getConnection();
        String sql = "SELECT * FROM Admin WHERE admin_id = ? AND password =
    ?";
        stmt = conn.prepareStatement(sql);
        stmt.setString(1, adminId);  // 设置管理员ID参数
        stmt.setString(2, password); // 设置密码参数
        rs = stmt.executeQuery();    // 执行查询

        if (rs.next()) {
            // 查询结果存在，创建管理员对象
            admin = new Admin();
            admin.setAdminId(rs.getString("admin_id"));
            admin.setPassword(rs.getString("password"));
            admin.setName(rs.getString("name"));
            admin.setContact(rs.getString("contact"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        DBUtil.closeResources(conn, stmt, rs);
    }

    return admin;
}
```

Listing 15: AdminDAO.login 方法实现

5.2 图书入库

实现了两种图书入库方式：

- **单本入库**：通过表单输入图书信息，支持添加新书或增加已有图书的库存
- **批量入库**：从文本文件导入图书信息，支持按指定格式批量添加图书

BookDAO 类中的 addBooks 方法代码实现：

```
/**
 * 批量添加图书
 * @param books 图书对象列表
 * @return 成功添加的图书数量
 */
public int addBooks(List<Book> books) {
    int successCount = 0;

    // 遍历图书列表，逐一添加
    for (Book book : books) {
        if (addBook(book)) {
            successCount++;  // 添加成功计数加1
        }
    }

    return successCount;
}
```

Listing 16: BookDAO.addBooks 方法实现

5.3 图书查询

实现了多条件组合查询功能：

- 支持按类别、书名、出版社、年份区间、作者、价格区间进行查询
- 支持按用户指定字段（书名、类别、出版社、年份、作者、价格）排序
- 查询结果显示在表格中，限制最多显示 50 条记录

查询界面代码实现：

```
/**
 * 图书查询面板
 */
package com.library.ui;

import com.library.dao.BookDAO;
```

```java
import com.library.model.Book;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.math.BigDecimal;
import java.util.List;

public class BookSearchPanel extends JPanel {
    private JTextField txtCategory;
    private JTextField txtTitle;
    private JTextField txtPublisher;
    private JTextField txtMinYear;
    private JTextField txtMaxYear;
    private JTextField txtAuthor;
    private JTextField txtMinPrice;
    private JTextField txtMaxPrice;
    private JComboBox<String> cboSortBy;
    private JButton btnSearch;
    private JTable tblBooks;
    private DefaultTableModel tableModel;

    public BookSearchPanel() {
        setLayout(new BorderLayout(10, 10));
        setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        // 创建查询条件面板
        JPanel searchPanel = new JPanel();
        searchPanel.setLayout(new GridLayout(5, 4, 10, 10));

        // 添加查询条件输入框
        searchPanel.add(new JLabel("类别: "));
        txtCategory = new JTextField();
        searchPanel.add(txtCategory);

        searchPanel.add(new JLabel("书名: "));
        txtTitle = new JTextField();
        searchPanel.add(txtTitle);

        searchPanel.add(new JLabel("出版社: "));
        txtPublisher = new JTextField();
        searchPanel.add(txtPublisher);

```

```
52          searchPanel.add(new JLabel("作者: "));
53          txtAuthor = new JTextField();
54          searchPanel.add(txtAuthor);
55
56          searchPanel.add(new JLabel("最小年份: "));
57          txtMinYear = new JTextField();
58          searchPanel.add(txtMinYear);
59
60          searchPanel.add(new JLabel("最大年份: "));
61          txtMaxYear = new JTextField();
62          searchPanel.add(txtMaxYear);
63
64          searchPanel.add(new JLabel("最小价格: "));
65          txtMinPrice = new JTextField();
66          searchPanel.add(txtMinPrice);
67
68          searchPanel.add(new JLabel("最大价格: "));
69          txtMaxPrice = new JTextField();
70          searchPanel.add(txtMaxPrice);
71
72          searchPanel.add(new JLabel("排序字段: "));
73          String[] sortOptions = {"书名", "类别", "出版社", "年份", "作者", "
    价格"};
74          cboSortBy = new JComboBox<>(sortOptions);
75          searchPanel.add(cboSortBy);
76
77          searchPanel.add(new JLabel(""));
78          btnSearch = new JButton("查询");
79          searchPanel.add(btnSearch);
80
81          // 创建表格
82          String[] columnNames = {"书号", "类别", "书名", "出版社", "年份", "
    作者", "价格", "总藏书量", "库存"};
83          tableModel = new DefaultTableModel(columnNames, 0) {
84              @Override
85              public boolean isCellEditable(int row, int column) {
86                  return false; // 设置表格不可编辑
87              }
88          };
89          tblBooks = new JTable(tableModel);
90          JScrollPane scrollPane = new JScrollPane(tblBooks);
91
92          // 添加查询按钮事件
93          btnSearch.addActionListener(new ActionListener() {
94              @Override
```

```
            public void actionPerformed(ActionEvent e) {
                searchBooks();
            }
        });

        // 添加组件到面板
        add(searchPanel, BorderLayout.NORTH);
        add(scrollPane, BorderLayout.CENTER);
    }

    // 图书查询方法代码部分已在上文提供
}
```

Listing 17: 图书查询界面实现

5.4 借书功能

实现了完整的借书流程：

- 输入借书证卡号，显示该借书证已借图书
- 输入图书编号，检查库存状态
- 如有库存，生成借书记录并减少库存
- 如无库存，提示最近归还日期

借书记录 DAO 方法实现：

```
/**
 * 添加借书记录
 * @param record 借书记录对象
 * @return 操作是否成功
 */
public boolean addBorrowRecord(BorrowRecord record) {
    Connection conn = null;
    PreparedStatement stmt = null;
    boolean success = false;

    try {
        conn = DBUtil.getConnection();
        String sql = "INSERT INTO Borrow (book_id, card_id, borrow_date,
    admin_id) " +
                     "VALUES (?, ?, ?, ?)";
        stmt = conn.prepareStatement(sql);
        stmt.setString(1, record.getBookId());  // 图书ID
```

```
17          stmt.setString(2, record.getCardId());  // 借书证ID
18          // 将Java Date转换为SQL Date
19          stmt.setDate(3, new java.sql.Date(record.getBorrowDate().getTime())
    );  // 借书日期
20          stmt.setString(4, record.getAdminId()); // 管理员ID
21
22          int rowsAffected = stmt.executeUpdate();
23          success = (rowsAffected > 0);  // 影响的行数大于0表示操作成功
24      } catch (SQLException e) {
25          e.printStackTrace();
26      } finally {
27          DBUtil.closeResources(conn, stmt, null);
28      }
29
30      return success;
31  }
```

Listing 18: BorrowDAO.addBorrowRecord 方法

5.5 还书功能

实现了完整的还书流程：

- 输入借书证卡号，显示该借书证已借图书
- 输入归还的图书编号
- 验证该图书是否被该借书证借出
- 成功还书后更新借书记录和图书库存

更新图书库存的方法实现：

```
1   /**
2    * 更新图书库存
3    * @param bookId 图书ID
4    * @param stockChange 库存变化量(正数为增加, 负数为减少)
5    * @return 操作是否成功
6    */
7   public boolean updateBookStock(String bookId, int stockChange) {
8       Connection conn = null;
9       PreparedStatement stmt = null;
10      boolean success = false;
11
12      try {
13          conn = DBUtil.getConnection();
```

```
        String sql = "UPDATE Book SET stock = stock + ? WHERE book_id = ?";
        stmt = conn.prepareStatement(sql);
        stmt.setInt(1, stockChange);    // 库存变化量
        stmt.setString(2, bookId);      // 图书ID

        int rowsAffected = stmt.executeUpdate();
        success = (rowsAffected > 0);  // 影响的行数大于0表示操作成功
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        DBUtil.closeResources(conn, stmt, null);
    }

    return success;
}
```

Listing 19: BookDAO.updateBookStock 方法

5.6 借书证管理

实现了借书证的添加和删除功能：

- 添加借书证：录入卡号、姓名、单位和类别信息
- 删除借书证：根据卡号删除借书证，检查是否有未归还图书

删除借书证 DAO 方法实现：

```
/**
 * 删除借书证
 * @param cardId 借书证ID
 * @return 操作是否成功
 */
public boolean deleteCard(String cardId) {
    Connection conn = null;
    PreparedStatement stmt = null;
    boolean success = false;

    try {
        conn = DBUtil.getConnection();
        // 先检查是否有未归还的书
        String checkSql = "SELECT COUNT(*) FROM Borrow WHERE card_id = ?
   AND return_date IS NULL";
        PreparedStatement checkStmt = conn.prepareStatement(checkSql);
        checkStmt.setString(1, cardId);
        ResultSet rs = checkStmt.executeQuery();
```

```
            rs.next();
            int borrowCount = rs.getInt(1);  // 获取未还书数量
            rs.close();
            checkStmt.close();

            if (borrowCount > 0) {
                // 还有未归还的书，不能删除
                return false;
            }

            // 执行删除操作
            String sql = "DELETE FROM Card WHERE card_id = ?";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, cardId);

            int rowsAffected = stmt.executeUpdate();
            success = (rowsAffected > 0);  // 影响的行数大于0表示操作成功
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DBUtil.closeResources(conn, stmt, null);
        }

        return success;
    }
```

Listing 20: CardDAO.deleteCard 方法

6 系统特色与创新点

6.1 图形用户界面

使用 Java Swing 构建了完整的图形用户界面，提高了系统易用性：

- 使用选项卡（JTabbedPane）组织不同功能模块
- 使用表格（JTable）展示查询结果
- 提供直观的表单输入和按钮操作

下面是使用 JTabbedPane 的代码实现：

```
// 创建选项卡面板
tabbedPane = new JTabbedPane();

```

```
// 添加各功能模块面板
tabbedPane.addTab("图书入库", new BookAddPanel(admin));
tabbedPane.addTab("图书查询", new BookSearchPanel());
tabbedPane.addTab("借书", new BorrowPanel(admin));
tabbedPane.addTab("还书", new ReturnPanel(admin));
tabbedPane.addTab("借书证管理", new CardManagementPanel());

// 添加选项卡面板到窗口
add(tabbedPane);
```

Listing 21: 使用 JTabbedPane 组织界面

6.2 数据验证与错误处理

系统实现了全面的数据验证和错误处理机制：

- 表单输入验证（非空检查、数据类型检查等）
- 数据库操作异常捕获与处理
- 用户友好的错误提示

数据验证代码示例：

```
// 检查输入有效性
if (bookId.isEmpty() || category.isEmpty() || title.isEmpty() ||
    publisher.isEmpty() || author.isEmpty()) {
    JOptionPane.showMessageDialog(this, "所有字段都不能为空", "错误",
    JOptionPane.ERROR_MESSAGE);
    return;
}

if (year <= 0 || price.compareTo(BigDecimal.ZERO) <= 0 || quantity <= 0) {
    JOptionPane.showMessageDialog(this, "年份、价格和数量必须为正数", "错误
    ", JOptionPane.ERROR_MESSAGE);
    return;
}
```

Listing 22: 输入数据验证示例

6.3 批量导入功能

实现了图书批量导入功能，提高了数据录入效率：

- 支持从文本文件批量导入图书信息
- 提供详细的导入过程日志和结果统计

- 自动跳过格式错误的记录

文件解析代码示例：

```
while ((line = reader.readLine()) != null) {
    lineCount++;
    try {
        // 解析行数据格式：(book_id, category, title, publisher, year, author, price, quantity)
        line = line.trim();

        // 检查是否是空行
        if (line.isEmpty()) {
            continue;
        }

        // 移除开头的"("和结尾的")"
        if (line.startsWith("(") && line.endsWith(")")) {
            line = line.substring(1, line.length() - 1);
        }

        // 分割字段
        String[] fields = line.split(",");

        if (fields.length != 8) {
            txtResult.append("第" + lineCount + "行格式错误，应包含8个字段\n");
            continue;
        }

        // 提取和转换字段
        String bookId = fields[0].trim();
        String category = fields[1].trim();
        String title = fields[2].trim();
        String publisher = fields[3].trim();
        int year = Integer.parseInt(fields[4].trim());
        String author = fields[5].trim();
        BigDecimal price = new BigDecimal(fields[6].trim());
        int quantity = Integer.parseInt(fields[7].trim());

        // 创建图书对象
        Book book = new Book(bookId, category, title, publisher, year, author, price, quantity, quantity);
        books.add(book);

        txtResult.append("解析第" + lineCount + "行：" + title + "\n");
```

```
40      } catch (Exception e) {
41          txtResult.append("解析第" + lineCount + "行出错: " + e.getMessage()
        + "\n");
42      }
43  }
```

Listing 23: 批量导入文件解析示例

7 总结

本图书馆管理系统基于 Java 和 MySQL 实现，采用了三层架构设计，成功完成了实验指导书中的所有要求功能。系统具有以下特点：

1. 完整的功能模块：实现了图书入库、查询、借书、还书、借书证管理等功能。
2. 友好的用户界面：使用 Java Swing 构建图形界面，提高了系统易用性。
3. 可靠的数据存储：使用 MySQL 数据库存储系统数据，确保数据的持久性和一致性。
4. 良好的架构设计：采用 MVC 和 DAO 设计模式，实现了代码的模块化和可维护性。

通过本次实验，加深了对数据库原理及应用开发的理解，提高了系统设计与实现的能力。系统虽然实现了基本功能，但在安全性、性能优化和用户体验等方面还有提升空间，这些将作为未来改进的方向。