A Mini Project

As a part of sessional evaluation for Programming in Java

Autumn 2024

Submission of Proposal

Book Lending System

Submitted by:

Abhilash Bora (CSE21-01)

Anik Chiring Phukan (CSE21-03)

Anupal Baruah (CSE21-04)

Arindam Atraya (CSE21-05)

Submitted to:

Department of Computer Science and Engineering

DIBRUGARH UNIVERSITY INSTITUTE OF ENGINEERING AND

TECHNOLOGY

Dibrugarh, Assam

Table of Contents

Overview

The Book Lending System is a desktop application designed using Java Swing to manage the operations of a small lending library. The system enables users to interactively add books to the collection, lend them to borrowers, and manage payment confirmations for the associated lending fees. With a well-structured graphical user interface (GUI), the system presents a user-friendly environment that simplifies the process of tracking books and borrowers, making it easy for users to manage a personal or small-scale lending library.

This application focuses on key lending library operations, organized within two main classes: BookLendingSystem, which drives the interface and handles user interactions, and Book, which represents the data structure for each book in the collection. The GUI, created with Java Swing components, includes organized sections for input fields (such as title, price, and borrower information), a central book list, status messages, and actionable buttons for adding books, lending them, and confirming payments.

Upon adding a book, users can set the title, price, and the name of the person adding the book to the system. When lending a book, the system checks the availability status, prompts for borrower information, and calculates a lending fee based on a percentage of the book's price. This fee is displayed to the user, with a "Pay" button provided for confirmation. Users can also double-click on any book in the list to view detailed information, including title, price, lender, borrower, and status, in a separate dialog.

Designed to be extensible, the Book Lending System demonstrates core programming concepts such as event handling, GUI layout management, and object-oriented design. It provides a basis for developing further functionality, such as persistent data storage, user authentication, and more sophisticated lending policies, making it a solid foundation for a practical library management tool.

Abstract

The Book Lending System is a Java Swing-based application designed to facilitate the management of a small lending library. The system enables users to add books, lend them to borrowers, calculate lending fees, and track payment confirmations through an intuitive graphical user interface. The application comprises two main classes: BookLendingSystem, which handles the user interface and program logic, and Book, which models the properties and state of each book. Key features include book addition, borrower tracking, lending fee calculation based on book price, and real-time status updates. Users can easily interact with the system, view book details, and confirm payment of lending fees, making the process efficient and transparent. The application's design demonstrates fundamental principles of Java Swing programming, object-oriented design, and event-driven functionality, providing a foundation for future expansions such as data persistence, multi-user access, and advanced lending policies.

Code Structure

The application is split into two main classes:

1. BookLendingSystem (Main GUI Class): This class sets up the user interface, handles user interactions, and manages the logic for adding and lending books.
2. Book (Model Class): This class defines the structure and attributes of a book, including its title, price, lender, borrower, status, and who added the book.

1. BookLendingSystem Class - Detailed Analysis

Key Components:

Attributes:

ArrayList<Book> books: Stores the list of books in the system.

DefaultListModel<String> bookListModel: Manages the list model for displaying books in the GUI.

GUI Components: Includes JList for displaying books, JTextField for input fields, JLabel for status display, and buttons for actions like "Add Book," "Lend Book," and "Pay."

User Interface Layout:

Main Panel: Contains a BorderLayout with sub-panels for input, book listing, and status updates.

Input Panel: Uses GridBagLayout for organized input fields (Book Title, Price, Added By, and Borrower Name).

Buttons Panel: Contains buttons for adding books, lending books, and confirming payment. The "Pay" button is initially disabled and enabled only when lending a book.

Functionalities:

Add Book:

The addBook() method allows users to enter details for a new book (title, price, and addedBy). If any required field is empty or the price is invalid, it shows an error.

The book is added to the books list and displayed in the JList with details formatted for easy readability.

Lend Book:

The lendBook() method enables lending of a selected book.

Checks if the book is already borrowed; if not, it allows lending to a borrower and calculates a 5% lending fee, enabling the "Pay" button for fee confirmation.

Displays the calculated fee in a dialog box.

Confirm Payment:

The confirmPayment() method handles payment confirmation, disabling the "Pay" button once the fee is paid and updating the status label to reflect the payment.

Book Details Display:

A double-click listener on the book list enables viewing details of any selected book, showing its title, price, added by, status, and borrower information in a pop-up dialog.

2. **Book Class - Detailed Analysis**
   Attributes:
   The Book class represents a book's essential properties and includes:

- **String title**: The book's title.
- **double price**: The book's price.
- **String lender and String borrower**: To track the lender and borrower.
- **String status**: Indicates if the book is "Available" or "Borrowed".
- **String addedBy**: Stores the name of the user who added the book.
  Methods:
- **Constructor**: Initializes a new book with a title, price, and "Available" status.
- **Getters and Setters**: Standard accessors and mutators for each attribute.
- **calculateLendingFee()**: Calculates a lending fee as 30% of the book's price. Note: The BookLendingSystem class overrides this value to 5% for lending operations.

Key Features

1. **Dynamic Book Management**:
    - The application dynamically manages books through the GUI, allowing users to add and lend books with ease.
2. **Lending Fee Calculation**:
    - Lending fees are automatically calculated as a percentage of the book price, promoting transparency in transactions.
3. **Payment Confirmation**:
    - A "Pay" button allows users to confirm the lending fee, which is acknowledged with a pop-up message, enhancing user experience.
4. **Double-Click Book Details**:
    - A double-click feature shows detailed book information, making it convenient for users to check book status without navigating through the entire list.

Improvements and Future Enhancements

1. **Error Handling Enhancements**:
    - Improve error handling by checking for invalid data types directly in input fields.
2. **Database Integration**:
    - For better scalability, integrate with a database to persist data, allowing the library to maintain its book records across sessions.
3. **User Login System**:
    - Implement a user authentication system, which would allow individual users to lend and return books under their own accounts.
4. **Enhanced Lending Policies**:
    - Add custom lending policies, such as adjustable lending fees or borrowing duration limits, for more flexibility.

Code

1st code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import javax.swing.border.TitledBorder;
public class BookLendingSystem extends JFrame {
    private ArrayList<Book> books;
    private DefaultListModel<String> bookListModel;
    private JList<String> bookList;
    private JTextField titleField, priceField, borrowerField, addedByField;
    private JLabel statusLabel;
    private JButton payButton;
    private double lendingFee;
    public BookLendingSystem() {
        books = new ArrayList<>();
        bookListModel = new DefaultListModel<>();
        setTitle("Book Lending System");
        setSize(800, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        // Main panel with padding
        JPanel mainPanel = new JPanel(new BorderLayout(10, 10));
        mainPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
        // Input Panel
        JPanel inputPanel = new JPanel(new GridBagLayout());
        inputPanel.setBorder(BorderFactory.createTitledBorder(
            BorderFactory.createEtchedBorder(), "Book Details",
```

```
    TitledBorder.LEFT, TitledBorder.TOP));
GridBagConstraints gbc = new GridBagConstraints();
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.insets = new Insets(5, 5, 5, 5);
// Add Book Section
gbc.gridx = 0; gbc.gridy = 0;
inputPanel.add(new JLabel("Book Title:"), gbc);
gbc.gridx = 1;
titleField = new JTextField(20);
inputPanel.add(titleField, gbc);
gbc.gridx = 0; gbc.gridy = 1;
inputPanel.add(new JLabel("Price (₹):"), gbc);
gbc.gridx = 1;
priceField = new JTextField(20);
inputPanel.add(priceField, gbc);
gbc.gridx = 0; gbc.gridy = 2;
inputPanel.add(new JLabel("Added By:"), gbc);
gbc.gridx = 1;
addedByField = new JTextField(20);
inputPanel.add(addedByField, gbc);
// Lending Section
gbc.gridx = 0; gbc.gridy = 3;
inputPanel.add(new JLabel("Borrower Name:"), gbc);
gbc.gridx = 1;
borrowerField = new JTextField(20);
inputPanel.add(borrowerField, gbc);
// Buttons Panel
JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 0));
JButton addButton = new JButton("Add Book");
```

```
        JButton lendButton = new JButton("Lend Book");
        payButton = new JButton("Pay");  // New Pay button
        payButton.setEnabled(false);     // Initially disabled
        buttonPanel.add(addButton);
        buttonPanel.add(lendButton);
        buttonPanel.add(payButton);      // Add Pay button to panel
        gbc.gridx = 0; gbc.gridy = 4;
        gbc.gridwidth = 2;
        inputPanel.add(buttonPanel, gbc);
        // Book List with border
        bookList = new JList<>(bookListModel);
        bookList.setFont(new Font("Monospaced", Font.PLAIN, 12));
        JScrollPane scrollPane = new JScrollPane(bookList);
        scrollPane.setBorder(BorderFactory.createTitledBorder(
            BorderFactory.createEtchedBorder(), "Book List",
            TitledBorder.LEFT, TitledBorder.TOP));
        // Status Label
        statusLabel = new JLabel("Status: Ready");
        statusLabel.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10));
        statusLabel.setFont(new Font("SansSerif", Font.BOLD, 12));
        // Add components to main panel
        mainPanel.add(inputPanel, BorderLayout.NORTH);
        mainPanel.add(scrollPane, BorderLayout.CENTER);
        mainPanel.add(statusLabel, BorderLayout.SOUTH);
        // Add main panel to frame
        add(mainPanel);
        // Add Button Action
        addButton.addActionListener(e -> addBook());
    // Lend Button Action
```

```
 lendButton.addActionListener(e -> lendBook());
      // Pay Button Action
      payButton.addActionListener(e -> confirmPayment()); // Add payment confirmation
      // Double-click listener for book details
      bookList.addMouseListener(new MouseAdapter() {
         public void mouseClicked(MouseEvent e) {
            if (e.getClickCount() == 2) {
               showBookDetails();
            }
         }
      });
   }
   private void addBook() {
      try {
         String title = titleField.getText().trim();
         String priceText = priceField.getText().trim();
         String addedBy = addedByField.getText().trim();
         if (title.isEmpty() || priceText.isEmpty() || addedBy.isEmpty()) {
            showError("Please fill in all fields (Title, Price, and Added By)");
            return;
         }
         double price = Double.parseDouble(priceText);
         if (price <= 0) {
            showError("Price must be greater than 0");
            return;
         }
         Book book = new Book(title, price, addedBy);
         books.add(book);
         bookListModel.addElement(String.format("%-30s ₹%-10.2f Added by: %-20s
(Available)",
```

```
truncateString(title, 30),
                price,
                truncateString(addedBy, 20)));
            clearFields();
            statusLabel.setText("Status: Book added successfully");
        } catch (NumberFormatException ex) {
            showError("Please enter a valid price");
        }
    }
    private void lendBook() {
        int selectedIndex = bookList.getSelectedIndex();
        if (selectedIndex == -1) {
            showError("Please select a book to lend");
            return;
        }
        Book selectedBook = books.get(selectedIndex);
        if (selectedBook.getStatus().equals("Borrowed")) {
            showError("This book is already borrowed");
            return;
        }
        String borrower = borrowerField.getText().trim();
        if (borrower.isEmpty()) {
            showError("Please enter the borrower's name");
            return;
        }
        selectedBook.setBorrower(borrower);
        selectedBook.setStatus("Borrowed");
        // Calculate 30% of the book price as lending fee
        lendingFee = selectedBook.getPrice() * 0.05;
```

```
        bookListModel.setElementAt(
            String.format("%-5s ₹%-10.2f Borrowed by: %-15s",
                truncateString(selectedBook.getTitle(), 5),
                selectedBook.getPrice(),
                truncateString(borrower, 15)),
            selectedIndex
        );
        // Enable the Pay button after lending the book
        payButton.setEnabled(true);
        // Display the lending fee to the user
        JOptionPane.showMessageDialog(
            this,
            String.format("Lending fee (5%% of ₹%.2f): ₹%.2f", selectedBook.getPrice(),
lendingFee),
            "Lending Fee",
            JOptionPane.INFORMATION_MESSAGE
        );
        clearFields();
        statusLabel.setText("Status: Book lent successfully");
    }
    private void confirmPayment() {
        // Disable the Pay button after payment is confirmed
        payButton.setEnabled(false);
        // Show confirmation dialog for payment
        JOptionPane.showMessageDialog(
            this,
            String.format("Payment of ₹%.2f confirmed!", lendingFee),
            "Payment Confirmed",
            JOptionPane.INFORMATION_MESSAGE
        );
```

```
    // Update status to show that payment has been made
    statusLabel.setText("Status: Payment of ₹" + lendingFee + " confirmed.");
}
private void showBookDetails() {
    int selectedIndex = bookList.getSelectedIndex();
    if (selectedIndex != -1) {
        Book book = books.get(selectedIndex);
        String details = String.format(
            "Title: %s\n" +
            "Price: ₹%.2f\n" +
            "Added By: %s\n" +
            "Status: %s\n" +
            "Borrower: %s",
            book.getTitle(),
            book.getPrice(),
            book.getAddedBy(),
            book.getStatus(),
            book.getBorrower() != null ? book.getBorrower() : "N/A"
        );
        JOptionPane.showMessageDialog(
            this,
            details,
            "Book Details",
            JOptionPane.INFORMATION_MESSAGE
        );
    }
}
```

```
private String truncateString(String str, int length) {
    if (str == null) return "";
    return str.length() > length ? str.substring(0, length - 3) + "..." : str;
}
private void clearFields() {
    titleField.setText("");
    priceField.setText("");
    addedByField.setText("");
    borrowerField.setText("");
}
private void showError(String message) {
    JOptionPane.showMessageDialog(
        this,
        message,
        "Error",
        JOptionPane.ERROR_MESSAGE
    );
}
public static void main(String[] args) {
    try {
        // Set system look and feel
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (Exception e) {
        e.printStackTrace();
    }
    SwingUtilities.invokeLater(() -> {
        BookLendingSystem system = new BookLendingSystem();
        system.setVisible(true);
    });
```

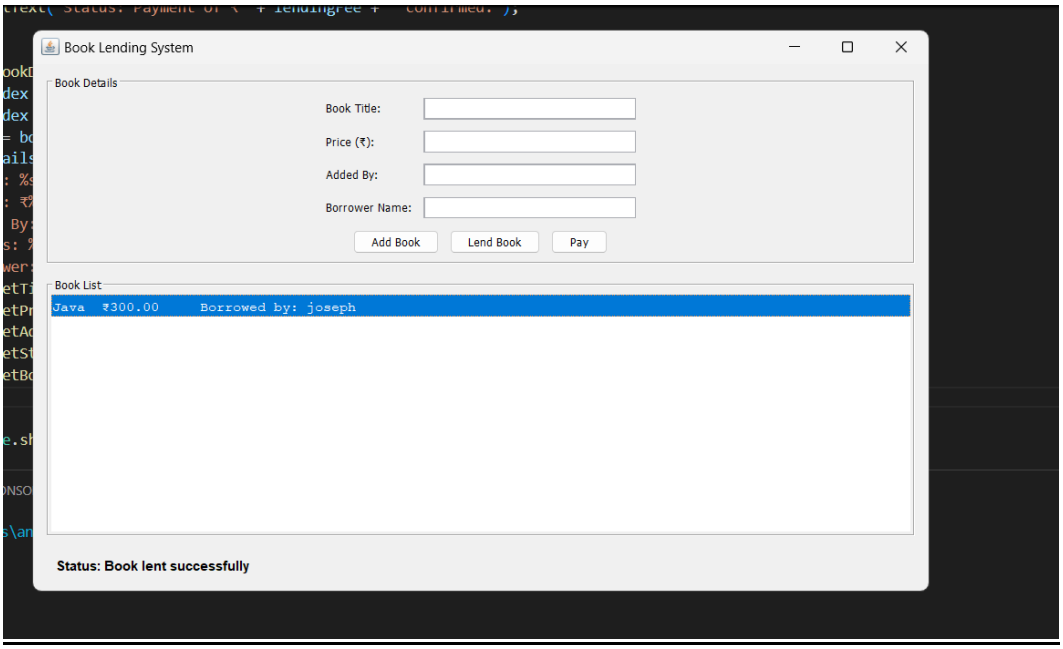
```
 }
}
```

2nd code

```
public class Book {
    private String title;
    private double price;
    private String lender;
    private String borrower;
    private String status; // "Available" or "Borrowed"
    private String addedBy; // Store who added the book
    public Book(String title, double price, String addedBy) {
        this.title = title;
        this.price = price;
        this.addedBy = addedBy;
        this.status = "Available";
    }
    // Getters and setters
    public String getTitle() { return title; }
    public double getPrice() { return price; }
    public String getLender() { return lender; }
    public String getBorrower() { return borrower; }
    public String getStatus() { return status; }
    public String getAddedBy() { return addedBy; }
    public void setLender(String lender) { this.lender = lender; }
    public void setBorrower(String borrower) { this.borrower = borrower; }
    public void setStatus(String status) { this.status = status; }
    public double calculateLendingFee() {
        return price * 0.30; // 30% of book price
    }
```

}

Output



Conclusion

The Book Lending System provides an intuitive and functional GUI for managing a lending library, covering essential operations such as adding, viewing, and lending books. With a strong foundation in Java Swing, the system can be expanded with database integration, authentication, and advanced features to meet the demands of a larger library environment.