STRUKTUR KODE :

1. Book.java

```
class Book {
    private String id;
    private String title;
    private String author;
```

- `class Book {` : Mendeklarasikan kelas `Book`.
- `private String id;` : Mendeklarasikan variabel `id` dengan akses `private`, hanya bisa diakses dalam kelas `Book`.
- `private String title;` : Mendeklarasikan variabel `title` dengan akses `private`, hanya bisa diakses dalam kelas `Book`.
- `private String author;` : Mendeklarasikan variabel `author` dengan akses `private`, hanya bisa diakses dalam kelas `Book`.

```
    public Book(String id, String title, String author) {
        this.id = id;
        this.title = title;
        this.author = author;
    }
```

- `public Book(String id, String title, String author) {` : Mendeklarasikan konstruktor kelas `Book` yang menerima tiga parameter (`id`, `title`, `author`).
- `this.id = id;` : Menginisialisasi variabel `id` kelas `Book` dengan nilai parameter `id`.
- `this.title = title;` : Menginisialisasi variabel `title` kelas `Book` dengan nilai parameter `title`.
- `this.author = author;` : Menginisialisasi variabel `author` kelas `Book` dengan nilai parameter `author`.

```
    public String getId() {
        return id;
    }
```

- `public String getId() {` : Mendeklarasikan metode `getId` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return id;` : Mengembalikan nilai `id` dari objek `Book`.

```
    public String getTitle() {
        return title;
    }
```

- `public String getTitle() {` : Mendeklarasikan metode `getTitle` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return title;` : Mengembalikan nilai `title` dari objek `Book`.

```
    public String getAuthor() {
        return author;
    }
```

- `public String getAuthor() {` : Mendeklarasikan metode `getAuthor` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return author;` : Mengembalikan nilai `author` dari objek `Book`.

```
    @Override
    public String toString() {
        return "ID: " + id + ", Title: " + title + ", Author: " + author;
    }
}
```

- `@Override` : Menandakan bahwa metode ini mengoverride (menimpa) metode `toString` dari kelas induk (`Object`).
- `public String toString() {` : Mendeklarasikan metode `toString` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return "ID: " + id + ", Title: " + title + ", Author: " + author;` : Mengembalikan representasi string dari objek `Book` yang berisi `id`, `title`, dan `author`.

2. Loan.java

```
class Loan {
    private String bookId;
    private String memberId;
    private String date;
    private String returnDate;
    private int period;
```

- `class Loan {` : Mendeklarasikan kelas `Loan`.
- `private String bookId;` : Mendeklarasikan variabel `bookId` dengan akses `private`, hanya bisa diakses dalam kelas `Loan`.
- `private String memberId;` : Mendeklarasikan variabel `memberId` dengan akses `private`, hanya bisa diakses dalam kelas `Loan`.
- `private String date;` : Mendeklarasikan variabel `date` dengan akses `private`, hanya bisa diakses dalam kelas `Loan`.
- `private String returnDate;` : Mendeklarasikan variabel `returnDate` dengan akses `private`, hanya bisa diakses dalam kelas `Loan`.
- `private int period;` : Mendeklarasikan variabel `period` dengan akses `private`, hanya bisa diakses dalam kelas `Loan`.

```
    public Loan(String bookId, String memberId, String date, String
returnDate, int period) {
        this.bookId = bookId;
        this.memberId = memberId;
        this.date = date;
        this.returnDate = returnDate;
        this.period = period;
    }
```

- `public Loan(String bookId, String memberId, String date, String returnDate, int period) {` : Mendeklarasikan konstruktor kelas `Loan` yang menerima lima parameter (`bookId`, `memberId`, `date`, `returnDate`, `period`).
- `this.bookId = bookId;` : Menginisialisasi variabel `bookId` kelas `Loan` dengan nilai parameter `bookId`.
- `this.memberId = memberId;` : Menginisialisasi variabel `memberId` kelas `Loan` dengan nilai parameter `memberId`.
- `this.date = date;` : Menginisialisasi variabel `date` kelas `Loan` dengan nilai parameter `date`.
- `this.returnDate = returnDate;` : Menginisialisasi variabel `returnDate` kelas `Loan` dengan nilai parameter `returnDate`.
- `this.period = period;` : Menginisialisasi variabel `period` kelas `Loan` dengan nilai parameter `period`.
-

```
    public String getBookId() {
        return bookId;
    }
```

- `public String getBookId() {` : Mendeklarasikan metode `getBookId` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return bookId;` : Mengembalikan nilai `bookId` dari objek `Loan`.

```
    public String getMemberId() {
        return memberId;
    }
```

- `public String getMemberId() {` : Mendeklarasikan metode `getMemberId` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return memberId;` : Mengembalikan nilai `memberId` dari objek `Loan`.

```
    public String getDate() {
        return date;
    }
```

- `public String getDate() {` : Mendeklarasikan metode `getDate` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return date;` : Mengembalikan nilai `date` dari objek `Loan`.

```
    public String getReturnDate() {
        return returnDate;
    }
```

- `public String getReturnDate() {` : Mendeklarasikan metode `getReturnDate` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return returnDate;` : Mengembalikan nilai `returnDate` dari objek `Loan`.

```
    public int getPeriod() {
        return period;
    }
}
```

- `public int getPeriod() {` : Mendeklarasikan metode `getPeriod` dengan akses `public` yang mengembalikan nilai tipe data `int`.
- `return period;` : Mengembalikan nilai `period` dari objek `Loan`.

3. `Member.java`

```
class Member {
    private String id;
    private String name;
    private int loanCount;
```

- `class Member {` : Mendeklarasikan kelas `Member`.
- `private String id;` : Mendeklarasikan variabel `id` dengan akses `private`, hanya bisa diakses dalam kelas `Member`.
- `private String name;` : Mendeklarasikan variabel `name` dengan akses `private`, hanya bisa diakses dalam kelas `Member`.
- `private int loanCount;` : Mendeklarasikan variabel `loanCount` dengan akses `private`, hanya bisa diakses dalam kelas `Member`.

```
    public Member(String id, String name) {
        this.id = id;
        this.name = name;
        this.loanCount = 0; // Initialize loan count to 0
    }
```

- `public Member(String id, String name) {` : Mendeklarasikan konstruktor kelas `Member` yang menerima dua parameter (`id` dan `name`).
- `this.id = id;` : Menginisialisasi variabel `id` kelas `Member` dengan nilai parameter `id`.
- `this.name = name;` : Menginisialisasi variabel `name` kelas `Member` dengan nilai parameter `name`.
- `this.loanCount = 0;` : Menginisialisasi variabel `loanCount` dengan nilai awal `0`, menunjukkan bahwa anggota belum memiliki pinjaman buku.

```
    public String getId() {
        return id;
    }
```

- `public String getId() {` : Mendeklarasikan metode `getId` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return id;` : Mengembalikan nilai `id` dari objek `Member`.

```
public String getName() {
    return name;
}
```

- `public String getName() {` : Mendeklarasikan metode `getName` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return name;` : Mengembalikan nilai `name` dari objek `Member`.

```
public int getLoanCount() {
    return loanCount;
}
```

- `public int getLoanCount() {` : Mendeklarasikan metode `getLoanCount` dengan akses `public` yang mengembalikan nilai tipe data `int`.
- `return loanCount;` : Mengembalikan nilai `loanCount` dari objek `Member`.

```
public void incrementLoanCount() {
    loanCount++;
}
```

- `public void incrementLoanCount() {` : Mendeklarasikan metode `incrementLoanCount` dengan akses `public`.
- `loanCount++;` : Meningkatkan nilai `loanCount` sebesar 1 setiap kali metode ini dipanggil.

```
public void decrementLoanCount() {
    if (loanCount > 0) {
        loanCount--;
    }
}
```

- `public void decrementLoanCount() {` : Mendeklarasikan metode `decrementLoanCount` dengan akses `public`.
- `if (loanCount > 0) {` : Memeriksa apakah nilai `loanCount` lebih besar dari 0 sebelum menguranginya.
- `loanCount--;` : Mengurangi nilai `loanCount` sebesar 1 jika syarat di atas terpenuhi.

```
    public boolean hasActiveLoans() {
        return loanCount > 0;
    }
```

- `public boolean hasActiveLoans() {` : Mendeklarasikan metode `hasActiveLoans` dengan akses `public` yang mengembalikan nilai tipe data `boolean`.
- `return loanCount > 0;` : Mengembalikan `true` jika `loanCount` lebih besar dari 0, menandakan bahwa anggota masih memiliki pinjaman buku aktif.

```
    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Loan Count: " +
loanCount;
    }
}
```

- `@Override` : Menandakan bahwa metode ini mengoverride (menimpa) metode `toString` dari kelas induk (`Object`).
- `public String toString() {` : Mendeklarasikan metode `toString` dengan akses `public` yang mengembalikan nilai tipe data `String`.
- `return "ID: " + id + ", Name: " + name + ", Loan Count: " + loanCount;` : Mengembalikan representasi string dari objek `Member` yang berisi `id`, `name`, dan `loanCount`.

4. `Library.java`

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.plaf.ColorUIResource;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.io.*;
import java.time.Duration;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.stream.Collectors;
```

- Mengimpor berbagai pustaka yang diperlukan untuk GUI (`javax.swing.*`), pengelolaan file (`java.io.*`), manipulasi waktu (`java.time.*`), dan struktur data (`java.util.*`).

```
public class Library extends JFrame {
    private List<Book> books;
    private List<Member> members;
    private List<Loan> loans;
    private List<Loan> loanHistory;
    private JTextArea displayArea;
```

- Mendeklarasikan kelas `Library` yang memperluas `JFrame` untuk membuat jendela GUI.
- Mendeklarasikan variabel-variabel yang digunakan untuk menyimpan daftar buku, anggota, pinjaman, dan sejarah pinjaman, serta area teks untuk menampilkan informasi.

```
    public Library() {
        books = new ArrayList<>();
        members = new ArrayList<>();
        loans = new ArrayList<>();
        loanHistory = new ArrayList<>();

        loadBooks();
        loadMembers();
        loadLoans();
        loadLoanHistory();

        setTitle("Library Management System");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
```

- Konstruktor `Library` menginisialisasi daftar buku, anggota, pinjaman, dan sejarah pinjaman sebagai `ArrayList`.
- Memuat data dari file menggunakan metode `loadBooks`, `loadMembers`, `loadLoans`, dan `loadLoanHistory`.
- Mengatur judul jendela, ukuran, operasi default saat ditutup, dan menempatkan jendela di tengah layar.

```
        UIManager.put("Panel.background", new
ColorUIResource(Color.DARK_GRAY));
        UIManager.put("OptionPane.background", new
ColorUIResource(Color.DARK_GRAY));
        UIManager.put("Button.background", new ColorUIResource(new Color(45,
45, 45)));
        UIManager.put("Button.foreground", new ColorUIResource(new Color(0,
255, 255)));
        UIManager.put("Button.border", new EmptyBorder(10, 10, 10, 10));
        UIManager.put("TextArea.background", new
ColorUIResource(Color.BLACK));
        UIManager.put("TextArea.foreground", new ColorUIResource(new Color(0,
255, 255)));
        UIManager.put("TextArea.caretForeground", new
ColorUIResource(Color.WHITE));
        UIManager.put("ScrollPane.border", new EmptyBorder(10, 10, 10, 10));
        UIManager.put("Table.background", new ColorUIResource(Color.BLACK));
        UIManager.put("Table.foreground", new ColorUIResource(new Color(0,
255, 255)));
        UIManager.put("Table.gridColor", new ColorUIResource(Color.GRAY));
        UIManager.put("TableHeader.background", new ColorUIResource(new
Color(45, 45, 45)));
        UIManager.put("TableHeader.foreground", new ColorUIResource(new
Color(0, 255, 255)));
        UIManager.put("TableHeader.cellBorder", new EmptyBorder(5, 5, 5, 5));
```

- Menyesuaikan tampilan dan nuansa GUI dengan mengubah properti UI menggunakan `UIManager`.

```
        displayArea = new JTextArea();
        displayArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(displayArea);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(9, 1, 5, 5)); // 9 rows, 1
column, with gaps between buttons
```

- Menginisialisasi `JTextArea` untuk menampilkan informasi dan membungkusnya dalam `JScrollPane` agar dapat digulir.
- Menginisialisasi `JPanel` untuk menampung tombol-tombol dengan layout grid (9 baris, 1 kolom) dengan jarak antar tombol.

```
        JButton searchButton = createCustomButton("Search Book");
        searchButton.addActionListener(e -> searchBook());
        buttonPanel.add(searchButton);

        JButton borrowButton = createCustomButton("Borrow Book");
        borrowButton.addActionListener(e -> borrowBook());
        buttonPanel.add(borrowButton);

        JButton returnButton = createCustomButton("Return Book");
        returnButton.addActionListener(e -> returnBook());
        buttonPanel.add(returnButton);

        JButton viewHistoryButton = createCustomButton("View Loan History");
        viewHistoryButton.addActionListener(e -> viewLoanHistory());
        buttonPanel.add(viewHistoryButton);

        JButton viewBooksButton = createCustomButton("View Books");
        viewBooksButton.addActionListener(e -> viewBooks());
        buttonPanel.add(viewBooksButton);

        JButton viewMembersButton = createCustomButton("View Members");
        viewMembersButton.addActionListener(e -> viewMembers());
        buttonPanel.add(viewMembersButton);

        JButton addMemberButton = createCustomButton("Add Member");
        addMemberButton.addActionListener(e -> addMember());
        buttonPanel.add(addMemberButton);

        JButton addBookButton = createCustomButton("Add Book");
        addBookButton.addActionListener(e -> addBook());
        buttonPanel.add(addBookButton);

        JButton viewBorrowedBookButton = createCustomButton("View Borrowed
Books");
        viewBorrowedBookButton.addActionListener(e -> viewBorrowedBooks());
        buttonPanel.add(viewBorrowedBookButton);

        getContentPane().setLayout(new BorderLayout(10, 10));
        getContentPane().add(buttonPanel, BorderLayout.WEST);
        getContentPane().add(scrollPane, BorderLayout.CENTER);

        setVisible(true);
    }
```

- Membuat berbagai tombol menggunakan `createCustomButton` dan menambahkan `ActionListener` untuk setiap tombol agar memicu metode yang sesuai saat tombol diklik.
- Menambahkan tombol-tombol tersebut ke `buttonPanel`.
- Mengatur layout konten jendela menggunakan `BorderLayout`, menambahkan `buttonPanel` di sebelah kiri (`WEST`) dan `scrollPane` di tengah (`CENTER`).
- Menampilkan jendela dengan `setVisible(true)`.

```
    private JButton createCustomButton(String text) {
        JButton button = new JButton(text);
        button.setFocusPainted(false);
        button.setFont(new Font("Verdana", Font.BOLD, 14));
        button.setBackground(new Color(45, 45, 45));
        button.setForeground(new Color(0, 255, 255));
        button.setBorder(new EmptyBorder(10, 10, 10, 10));
        button.setCursor(new Cursor(Cursor.HAND_CURSOR));
        return button;
    }
```

- Metode untuk membuat tombol dengan gaya kustom, termasuk mengatur font, warna latar belakang dan teks, serta kursor.

```
    private void loadBooks() {
        try (BufferedReader reader = new BufferedReader(new
FileReader("books.csv"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] data = line.split(",");
                books.add(new Book(data[0], data[1], data[2]));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- Metode untuk memuat data buku dari file `books.csv`, membaca setiap baris, memisahkan data berdasarkan koma, dan menambahkan objek `Book` ke daftar `books`.

```
    private void loadMembers() {
        try (BufferedReader reader = new BufferedReader(new
FileReader("members.csv"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] data = line.split(",");
                members.add(new Member(data[0], data[1]));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- Metode untuk memuat data anggota dari file `members.csv` dengan cara yang sama seperti `loadBooks`.

```
    private void loadLoans() {
        try (BufferedReader reader = new BufferedReader(new
FileReader("loans.csv"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] data = line.split(",");
                loans.add(new Loan(data[0], data[1], data[2], data[3],
Integer.parseInt(data[4])));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- Metode untuk memuat data pinjaman dari file `loans.csv`, membaca setiap baris, memisahkan data, dan menambahkan objek `Loan` ke daftar `loans`.

```
    private void loadLoanHistory() {
        try (BufferedReader reader = new BufferedReader(new
FileReader("loan_history.csv"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] data = line.split(",");
                loanHistory.add(new Loan(data[0], data[1], data[2], data[3],
Integer.parseInt(data[4])));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- Metode untuk memuat sejarah pinjaman dari file `loan_history.csv` dengan cara yang sama seperti `loadLoans`.

```
    private void saveBooks() {
        try (PrintWriter writer = new PrintWriter(new
FileWriter("books.csv"))) {
            for (Book book : books) {
                writer.println(book.toCSV());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void saveMembers() {
        try (PrintWriter writer = new PrintWriter(new
FileWriter("members.csv"))) {
            for (Member member : members) {
                writer.println(member.toCSV());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void saveLoans() {
        try (PrintWriter writer = new PrintWriter(new
FileWriter("loans.csv"))) {
            for (Loan loan : loans) {
                writer.println(loan.toCSV());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void saveLoanHistory() {
        try (PrintWriter writer = new PrintWriter(new
FileWriter("loan_history.csv"))) {
            for (Loan loan : loanHistory) {
                writer.println(loan.toCSV());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- Metode untuk menyimpan data buku, anggota, pinjaman, dan sejarah pinjaman ke file yang sesuai dengan menulis data sebagai CSV.

```
    private void searchBook() {
        String bookTitle = JOptionPane.showInputDialog(this, "Enter the title
of the book:");
        List<Book> foundBooks = books.stream()
                .filter(book -> book.getTitle().equalsIgnoreCase(bookTitle))
                .collect(Collectors.toList());
        displayArea.setText("");
        if (foundBooks.isEmpty()) {
            displayArea.append("No books found with the title: " +
bookTitle);
        } else {
            foundBooks.forEach(book -> displayArea.append(book + "\n"));
        }
    }
```

- Metode untuk mencari buku berdasarkan judul, menampilkan buku-buku yang ditemukan di `displayArea`.

```
    private void borrowBook() {
        String bookTitle = JOptionPane.showInputDialog(this, "Enter the title
of the book to borrow:");
        String memberId = JOptionPane.showInputDialog(this, "Enter your
member ID:");
        Book book = books.stream()
                .filter(b -> b.getTitle().equalsIgnoreCase(bookTitle))
                .findFirst()
                .orElse(null);
        if (book == null) {
            JOptionPane.showMessageDialog(this, "Book not found!");
            return;
        }
        Member member = members.stream()
                .filter(m -> m.getId().equals(memberId))
                .findFirst()
                .orElse(null);
        if (member == null) {
            JOptionPane.showMessageDialog(this, "Member not found!");
            return;
        }
        Random random = new Random();
        int loanId = random.nextInt(100000);
        Loan loan = new Loan(String.valueOf(loanId), member.getId(),
book.getId(),
                LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-
MM-dd HH:mm:ss")), 0);
        loans.add(loan);
        saveLoans();
        displayArea.setText("Book borrowed successfully!");
    }
```

- Metode untuk meminjam buku, meminta judul buku dan ID anggota, mencari buku dan anggota yang sesuai, membuat objek `Loan` baru, menambahkannya ke daftar `loans`, dan menyimpan data pinjaman.

```
    private void returnBook() {
        String bookTitle = JOptionPane.showInputDialog(this, "Enter the title
of the book to return:");
        String memberId = JOptionPane.showInputDialog(this, "Enter your
member ID:");
        Loan loan = loans.stream()
                .filter(l -> l.getMemberId().equals(memberId) &&
books.stream().anyMatch(b -> b.getId().equals(l.getBookId()) &&
b.getTitle().equalsIgnoreCase(bookTitle)))
                .findFirst()
                .orElse(null);
        if (loan == null) {
            JOptionPane.showMessageDialog(this, "Loan not found!");
            return;
        }
        loans.remove(loan);
        loanHistory.add(loan);
        saveLoans();
        saveLoanHistory();
        displayArea.setText("Book returned successfully!");
    }
```

- Metode untuk mengembalikan buku, meminta judul buku dan ID anggota, mencari pinjaman yang sesuai, menghapus pinjaman dari daftar `loans`, menambahkannya ke daftar `loanHistory`, dan menyimpan data.

```
    private void viewLoanHistory() {
        displayArea.setText("");
        loanHistory.forEach(loan -> displayArea.append(loan + "\n"));
    }

    private void viewBooks() {
        displayArea.setText("");
        books.forEach(book -> displayArea.append(book + "\n"));
    }

    private void viewMembers() {
        displayArea.setText("");
        members.forEach(member -> displayArea.append(member + "\n"));
    }

    private void addMember() {
        String memberId = JOptionPane.showInputDialog(this, "Enter new member
ID:");
        String memberName = JOptionPane.showInputDialog(this, "Enter new
member name:");
        members.add(new Member(memberId, memberName));
        saveMembers();
        displayArea.setText("Member added successfully!");
    }

    private void addBook() {
        String bookId = JOptionPane.showInputDialog(this, "Enter new book
ID:");
        String bookTitle = JOptionPane.showInputDialog(this, "Enter new book
title:");
        String bookAuthor = JOptionPane.showInputDialog(this, "Enter new book
author:");
        books.add(new Book(bookId, bookTitle, bookAuthor));
        saveBooks();
        displayArea.setText("Book added successfully!");
    }

    private void viewBorrowedBooks() {
        displayArea.setText("");
        loans.forEach(loan -> displayArea.append(loan + "\n"));
    }

    public static void main(String[] args) {
        new Library();
    }
}
```

- Metode tambahan untuk melihat sejarah pinjaman, daftar buku, daftar anggota, menambah anggota baru, menambah buku baru, dan melihat daftar buku yang dipinjam saat ini.
- Metode `main` untuk menjalankan aplikasi.