

LAPORAN PJBL PRAKTIKUM PBO
BOOKER



DISUSUN OLEH :

David Andrey Nugroho	(2340506053)
Wahyu Dwi Yulianto	(2340506056)
Noufal Aji Prasetyo	(2340506059)
Achmad Madania H.M.	(2340506062)

PROGRAM STUDI S1 TEKNOLOGI INFORMASI
JURUSAN TEKNIK ELEKTRO FAKULTAS TEKNIK
UNIVERSITAS TIDAR

2024

I. Tujuan Praktikum

1. Mengimplementasikan konsep PBO dalam pengembangan aplikasi berbasis Java.
2. Membuat aplikasi berbasis PBO untuk mengelola data buku.

II. Dasar Teori

Pemrograman Berorientasi Objek (Object-Oriented Programming atau OOP) adalah paradigma pemrograman yang menggunakan konsep objek dan kelas untuk membangun sebuah program. Paradigma ini bertujuan untuk mempermudah pengelolaan dan pengembangan program dengan cara menyusun kode berdasarkan abstraksi dunia nyata.

Konsep dasar dari OOP mencakup beberapa elemen penting, yaitu:

1. Kelas (Class):

Kelas adalah blueprint atau tamplet untuk menciptakan objek. Kelas mendefinisikan attribute (data) dan metode (fungsi) yang dimiliki oleh objek.

2. Objek (Object):

Objek adalah instance dari kelas. Setiap objek memiliki identitas unik, data dalam bentuk attribute dan perilaku dalam bentuk metode.

3. Enakpsulasi(Encapsulation):

Enakpsulasi adalah mekanisme untuk menyembunyikan detail implementasi dari suatu objek dan hanya memperlihatkan antarmuka yang diperlukan. Hal ini dilakukan dengan menggunakan modifier akses seperti Private, Protected, dan Public

4. Abstraksi(Abstraction):

Abastaksi adlah proses menyederhanakan kompleksitas dengan hanya menampilkan attribute dan metode yang relevan dari suatu objek. Sementara detail lainnya disembuyikan.

5. Pewarisan(Inheritance):

Pewarisan memungkinkan sebuah kelas (kelas anak) untuk mewarisi attribute dan metode dari kelas lain (kelas induk). Hal ini memungkinkan kode dapat digunakan kembali dan mempermudah pengelolaan hierarki kelas.

6. Polimorfisme(Polymorphism):

Polimorfisme memungkinkan objek untuk memiliki banyak bentuk. Dalam konteks metode, polimorfisme memungkinkan metode yang sama memiliki implementasi yang berbeda di kelas yang berbeda

Manfaat Pemrograman Berorientasi Objek(OOP):

1. Reusability (Penggunaan Kembali): Kode dapat digunakan kembali melalui pewarisan sehingga mengurangi duplikasi kode
2. Modularitas: Program lebih modular karena dibagi menjadi kelas-kelas yang saling terpisah
3. Pemeliharaan Mudah: Mempermudah pemeliharaan dan pengembangan program, karena setiap bagian kode memiliki tanggung jawab yang jelas
4. Kemudahan Abstraksi: Memungkinkan pengembangan untuk fokus pada logika tingkat tinggi tanpa harus memikirkan detail implementasi.

III. Metode Praktikum

A. Alat dan bahan

1. Seperangkat komputer lengkap/Laptop dengan koneksi internet.
2. Web Browser (Chrome/Firefox/Opera/Edge/Safari/dll).
3. Aplikasi Kantor (Microsoft Office/Libre Office/WPS Office/etc).
4. JDK (<https://www.oracle.com/java/technologies/downloads/>).
5. Netbeans (<https://netbeans.apache.org/front/main/download/>).

B. Langkah kerja

- 1) Membuka Aplikasi Netbeans untuk membuat project.
- 2) Membuat project baru dengan nama Booker.
- 3) Membuat kelas baru dengan nama Book. Kelas ini digunakan sebagai kelas induk.

Kodenya sebagai berikut:

```
11 import java.io.Serializable;
12
13 public abstract class Book implements Serializable {
14     protected String title;
15     protected String genre;
16     protected String review;
17     protected int rating;
18
19     public Book(String title, String genre, String review, int rating) {
20         this.title = title;
21         this.genre = genre;
22         this.review = review;
23         this.rating = rating;
24     }
25
26     public String getTitle() {
27         return title;
28     }
29
30     public abstract String getDetails();
31 }
```

Gambar 1 Kelas Book

Penjelsan:

- a. public abstract class Book digunakan untuk mendeklarasikan kelas Book sebagai kelas abstrak, yang berarti tidak bisa dibuat objek langsung, dan kelas ini mengimplementasikan interface Serializable untuk mendukung proses serialisasi objek.
- b. protected String title; merupakan atribut untuk menyimpan judul buku. Modifier protected memungkinkan akses oleh kelas turunan
- c. protected String genre; merupakan atribut untuk menyimpan genre buku.

- d. `protected String review`; digunakan untuk menyimpan ulasan atau deskripsi tentang buku.
- e. `protected int rating`; digunakan untuk menyimpan nilai atau skor buku, misalnya dalam skala 1-5.
- f. `public Book(String title, String genre, String review, int rating)` merupakan konstruktor yang digunakan untuk menginisialisasi atribut `title`, `genre`, `review`, dan `rating` saat objek dari kelas turunan dibuat.
- g. `public String getTitle()` merupakan metode getter yang mengembalikan nilai atribut `title`. Ini memudahkan kelas lain untuk mendapatkan judul buku tanpa mengakses langsung atribut.
- h. `public abstract String getDetails()` merupakan metode abstrak yang harus diimplementasikan oleh kelas turunan. Fungsinya untuk menampilkan detail buku sesuai dengan jenis buku tersebut.

4) Membuat kela Cerpen sebagai kelas turunan dari kelas induk Book.

```

12  public class Cerpen extends Book {
13      private int pages;
14
15      public Cerpen(String title, String genre, String review, int rating, int pages) {
16          super(title, genre, review, rating);
17          this.pages = pages;
18      }
19
20      @Override
21      public String getDetails() {
22          return "Cerpen: " + title +
23              "\nGenre: " + genre +
24              "\nReview: " + review +
25              "\nRating: " + rating +
26              "\nJumlah Halaman: " + pages;
27      }
28  }
29

```

Gambar 2 Kelas Turunan Cerpen

Penjelasan:

- a. `public class Cerpen extends Book` digunakan untuk mendefinisikan kelas Cerpen sebagai turunan dari kelas Book. Kelas ini mewarisi semua atribut dan metode yang ada pada Book.
- b. `private int pages`; merupakan atribut `pages` digunakan untuk menyimpan jumlah halaman cerpen. Atribut ini bersifat privat agar hanya dapat diakses dalam kelas Cerpen.
- c. `public Cerpen(String title, String genre, String review, int rating, int pages)` merupakan konstruktor untuk kelas Cerpen. Menerima parameter untuk judul, genre, ulasan, rating, dan jumlah halaman, kemudian memanggil konstruktor dari kelas induk untuk menginisialisasi atribut yang diwarisi.

- d. `super(title, genre, review, rating);` digunakan untuk memanggil konstruktor kelas induk `Book` untuk mengatur nilai atribut `title`, `genre`, `review`, dan `rating`.
- e. `this.pages = pages;` digunakan untuk menginisialisasi atribut `pages` dengan nilai yang diterima melalui parameter.
- f. `@Override` menunjukkan bahwa metode berikutnya (`getDetails`) mengimplementasi ulang metode abstrak dari kelas induk.
- g. `public String getDetails()` merupakan metode ini memberikan implementasi dari metode abstrak `getDetails` di kelas `Book`, menghasilkan deskripsi lengkap cerpen.
- h. `return "Cerpen: " + title + "\nGenre: " + genre + "\nReview: " + review + "\nRating: " + rating + "\nJumlah Halaman: " + pages;` digunakan untuk mengembalikan string yang berisi detail lengkap cerpen, termasuk judul, genre, ulasan, rating, dan jumlah halaman dalam format yang mudah dibaca.

5) Membuat kelas Novel sebagai kelas turunan dari kelas induk book.

```

11 public class Novel extends Book {
12     private String author;
13     private String publisher;
14
15     public Novel(String title, String genre, String review, int rating, String author, String publisher) {
16         super(title, genre, review, rating);
17         this.author = author;
18         this.publisher = publisher;
19     }
20
21     @Override
22     public String getDetails() {
23         return "Novel: " + title +
24             "\nGenre: " + genre +
25             "\nReview: " + review +
26             "\nRating: " + rating +
27             "\nPenulis: " + author +
28             "\nPenerbit: " + publisher;
29     }
30 }

```

Gambar 3 Kelas Turunan Novel

Penjelasan:

- a. `public class Novel extends Book` merupakan kelas `Novel` yang turunan dari kelas `Book`. Kelas ini mewarisi atribut dan metode dari kelas induk `Book`.
- b. `private String author;` merupakan atribut `author` digunakan untuk menyimpan nama penulis novel. Atribut ini bersifat privat agar hanya dapat diakses dalam kelas `Novel`.
- c. `private String publisher;` merupakan atribut `publisher` digunakan untuk menyimpan nama penerbit novel. Sama seperti `author`, atribut ini juga bersifat privat.

- d. `public Novel(String title, String genre, String review, int rating, String author, String publisher)` merupakan konstruktor untuk kelas Novel. Menerima parameter untuk judul, genre, ulasan, rating, penulis, dan penerbit. Konstruktor ini memanggil konstruktor kelas induk untuk menginisialisasi atribut yang diwarisi.
 - e. `super(title, genre, review, rating);` digunakan untuk memanggil konstruktor kelas induk Book dan menginisialisasi atribut title, genre, review, dan rating.
 - f. `this.author = author;` digunakan untuk menginisialisasi atribut author dengan nilai yang diterima melalui parameter.
 - g. `this.publisher = publisher;` digunakan untuk menginisialisasi atribut publisher dengan nilai yang diterima melalui parameter.
 - h. `@Override` menunjukkan bahwa metode berikutnya (`getDetails`) mengimplementasi ulang metode abstrak dari kelas induk.
 - i. `public String getDetails()` merupakan metode yang memberikan implementasi dari metode abstrak `getDetails` di kelas Book. Metode ini menghasilkan deskripsi lengkap novel.
 - j. `return "Novel: " + title + "\nGenre: " + genre + "\nReview: " + review + "\nRating: " + rating + "\nPenulis: " + author + "\nPenerbit: " + publisher;` digunakan untuk mengembalikan string yang berisi detail lengkap novel, termasuk judul, genre, ulasan, rating, penulis, dan penerbit dalam format yang terstruktur.
- 6) Membuat kelas `BookManager`. Kelas ini memiliki hubungan agregasi dengan kelas Book.

```
11 import java.io.*;
12 import java.util.ArrayList;
13
14 public class BookManager {
15     private ArrayList<Book> books;
16     public BookManager() {
17         books = new ArrayList<>();
18         loadBooks();
19     }
20
21     public void addBook(Book book) {
22         books.add(book);
23         saveBooks();
24     }
25
26     public void removeBook(String title) {
27         books.removeIf(book -> book.getTitle().equalsIgnoreCase(title));
28         saveBooks();
29     }
30
31     public ArrayList<Book> getBooks() {
32         return books;
33     }
34     // Menyimpan buku ke file
35     private void saveBooks() {
36         try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("books.dat"))) {
37             out.writeObject(books);
38         } catch (IOException e) {
39             e.printStackTrace();
40         }
41     }
42 }
```

```

42 // Memuat buku dari file
43 private void loadBooks() {
44     try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("books.dat"))) {
45         books = (ArrayList<Book>) in.readObject();
46     } catch (FileNotFoundException e) {
47     } catch (IOException | ClassNotFoundException e) {
48         e.printStackTrace();
49     }
50 }
51 }
52
53
54

```

Gambar 4 Kelas BookManager

Penjelasan:

- a. `import java.io.*;` dan `import java.util.ArrayList;` digunakan untuk mengimpor kelas-kelas yang diperlukan. `java.io.*` untuk input/output (seperti serialisasi objek), dan `java.util.ArrayList` untuk menggunakan koleksi `ArrayList` yang menyimpan objek buku.
- b. `public class BookManager` adalah kelas yang mengelola daftar buku, dengan kemampuan untuk menambah, menghapus, dan memuat buku dari file.
- c. `private ArrayList<Book> books;` adalah atribut untuk menyimpan daftar buku dalam bentuk `ArrayList`. Tipe data yang disimpan adalah objek `Book`.
- d. `public BookManager()` adalah konstruktor yang digunakan untuk menginisialisasi objek `BookManager`. Dalam konstruktor ini, objek `ArrayList` baru dibuat dan kemudian memanggil metode `loadBooks()` untuk memuat data buku dari file.
- e. `public void addBook(Book book)` adalah metode untuk menambahkan buku baru ke dalam daftar `books`. Setelah menambah buku, metode `saveBooks()` dipanggil untuk menyimpan daftar buku yang diperbarui ke file.
- f. `public void removeBook(String title)` adalah metode untuk menghapus buku berdasarkan judul. Metode ini menggunakan `removeIf` untuk menghapus buku yang memiliki judul yang sama (dengan pengecekan case-insensitive). Setelah buku dihapus, `saveBooks()` dipanggil untuk menyimpan perubahan.
- g. `public ArrayList<Book> getBooks()` adalah metode untuk mengembalikan daftar buku yang ada dalam `books`.
- h. `private void saveBooks()` adalah metode untuk menyimpan daftar buku ke dalam file menggunakan `ObjectOutputStream`. File yang digunakan adalah `"books.dat"`. Metode ini menulis objek `books` ke file untuk disimpan.
- i. `private void loadBooks()` adalah metode untuk memuat daftar buku dari file `"books.dat"` menggunakan `ObjectInputStream`.

- 7) Membuat kelas BookerGUI yang digunakan sebagai kelas main untuk eksekusi program, yang nantinya akan menampilkan aplikasinya.

```
12 import javax.swing.*;
13 import java.awt.*;
14 import java.awt.event.ActionEvent;
15 import java.awt.event.ActionListener;
16
17 public class BookerGUI {
18     private JFrame frame;
19     private JTextField titleField, reviewField, ratingField, authorField, publisherField, pagesField;
20     private JComboBox<String> genreBox, bookTypeBox;
21     private JTextArea bookListArea;
22     private BookManager bookManager;
23
24     public BookerGUI() {
25         bookManager = new BookManager();
26         createGUI();
27     }
28
29     private void createGUI() {
30         // Frame Setup
31         frame = new JFrame("Manajemen Buku");
32         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
33         frame.setSize(600, 600);
34         frame.setLayout(new GridBagLayout());
35         GridBagConstraints gbc = new GridBagConstraints();
36
37         // Components
38         JLabel titleLabel = new JLabel("Judul:");
39         titleField = new JTextField(20);
40
41         JLabel genreLabel = new JLabel("Genre:");
42         genreBox = new JComboBox<>(new String[]{"Fiksi", "Fantasi", "Sejarah", "Non-Fiksi"});
43
44         JLabel reviewLabel = new JLabel("Review:");
45         reviewField = new JTextField(20);
46
47         JLabel ratingLabel = new JLabel("Rating (1-5):");
48         ratingField = new JTextField(5);
49
50         JLabel bookTypeLabel = new JLabel("Jenis Buku:");
51         bookTypeBox = new JComboBox<>(new String[]{"Novel", "Cerpen"});
52
53         JLabel authorLabel = new JLabel("Penulis (Novel):");
54         authorField = new JTextField(20);
55
56         JLabel publisherLabel = new JLabel("Penerbit (Novel):");
57         publisherField = new JTextField(20);
58
59         JLabel pagesLabel = new JLabel("Jumlah Halaman (Cerpen):");
60         pagesField = new JTextField(10);
61
62         JButton addButton = new JButton("Tambah Buku");
63         JButton deleteButton = new JButton("Hapus Buku");
64         JButton displayButton = new JButton("Tampilkan Buku");
65
66         JLabel bookListLabel = new JLabel("Daftar Buku:");
67         bookListArea = new JTextArea(15, 40);
68         bookListArea.setEditable(false);
69         JScrollPane scrollPane = new JScrollPane(bookListArea);
70
71         // Add components to the frame
72         gbc.insets = new Insets(5, 5, 5, 5);
73         gbc.gridx = 0; gbc.gridy = 0; frame.add(titleLabel, gbc);
74         gbc.gridx = 1; frame.add(titleField, gbc);
75
76         gbc.gridx = 0; gbc.gridy = 1; frame.add(genreLabel, gbc);
77         gbc.gridx = 1; frame.add(genreBox, gbc);
78
79         gbc.gridx = 0; gbc.gridy = 2; frame.add(reviewLabel, gbc);
80         gbc.gridx = 1; frame.add(reviewField, gbc);
81
82         gbc.gridx = 0; gbc.gridy = 3; frame.add(ratingLabel, gbc);
83         gbc.gridx = 1; frame.add(ratingField, gbc);
84
85         gbc.gridx = 0; gbc.gridy = 4; frame.add(bookTypeLabel, gbc);
86         gbc.gridx = 1; frame.add(bookTypeBox, gbc);
```

```

88 gbc.gridx = 0; gbc.gridy = 5; frame.add(authorLabel, gbc);
89 gbc.gridx = 1; frame.add(authorField, gbc);
90
91 gbc.gridx = 0; gbc.gridy = 6; frame.add(publisherLabel, gbc);
92 gbc.gridx = 1; frame.add(publisherField, gbc);
93
94 gbc.gridx = 0; gbc.gridy = 7; frame.add(pagesLabel, gbc);
95 gbc.gridx = 1; frame.add(pagesField, gbc);
96
97 gbc.gridx = 0; gbc.gridy = 8; frame.add(addButton, gbc);
98 gbc.gridx = 1; frame.add(deleteButton, gbc);
99
00 gbc.gridx = 0; gbc.gridy = 9; gbc.gridwidth = 2; frame.add(displayButton, gbc);
01
02 gbc.gridx = 0; gbc.gridy = 10; frame.add(bookListLabel, gbc);
03 gbc.gridy = 11; frame.add(scrollPane, gbc);
04
05 // Set initial visibility
06 pagesField.setEnabled(false);
07
08 // Listeners
09 bookTypeBox.addActionListener(new ActionListener() {
10     @Override
11     public void actionPerformed(ActionEvent e) {
12         String selectedType = (String) bookTypeBox.getSelectedItem();
13         if ("Novel".equals(selectedType)) {
14             authorField.setEnabled(true);
15             publisherField.setEnabled(true);
16             pagesField.setEnabled(false);
17             pagesField.setText("");
18         } else if ("Cerpen".equals(selectedType)) {
19             authorField.setEnabled(false);
20             publisherField.setEnabled(false);
21             authorField.setText("");
22             publisherField.setText("");
23             pagesField.setEnabled(true);
24         }

```

```

125     }
126 });
127
128 addButton.addActionListener(e -> addBook());
129 deleteButton.addActionListener(e -> deleteBook());
130 displayButton.addActionListener(e -> displayBooks());
131
132 // Visibility
133 frame.setVisible(true);
134 }
135
136 private void addBook() {
137     try {
138         String title = titleField.getText();
139         String genre = (String) genreBox.getSelectedItem();
140         String review = reviewField.getText();
141         int rating = Integer.parseInt(ratingField.getText());
142         String bookType = (String) bookTypeBox.getSelectedItem();
143
144         if (bookType.equals("Novel")) {
145             String author = authorField.getText();
146             String publisher = publisherField.getText();
147             Novel novel = new Novel(title, genre, review, rating, author, publisher);
148             bookManager.addBook(novel);
149             JOptionPane.showMessageDialog(frame, "Novel berhasil ditambahkan!");
150         } else if (bookType.equals("Cerpen")) {
151             int pages = Integer.parseInt(pagesField.getText());
152             Cerpen cerpen = new Cerpen(title, genre, review, rating, pages);
153             bookManager.addBook(cerpen);
154             JOptionPane.showMessageDialog(frame, "Cerpen berhasil ditambahkan!");
155         }
156
157         clearFields();
158     } catch (Exception ex) {
159         JOptionPane.showMessageDialog(frame, "Error: " + ex.getMessage(), "Input Error", JOptionPane.ERROR_MESSAGE);
160     }
161 }

```

```

162
163 private void deleteBook() {
164     String title = JOptionPane.showInputDialog(frame, "Masukkan judul buku yang ingin dihapus:");
165     bookManager.removeBook(title);
166     JOptionPane.showMessageDialog(frame, "Buku berhasil dihapus (jika ada)!");
167 }
168
169 private void displayBooks() {
170     bookListArea.setText("");
171     for (Book book : bookManager.getBooks()) {
172         bookListArea.append(book.getDetails() + "\n\n");
173     }
174 }
175
176 private void clearFields() {
177     titleField.setText("");
178     reviewField.setText("");
179     ratingField.setText("");
180     authorField.setText("");
181     publisherField.setText("");
182     pagesField.setText("");
183 }
184
185 public static void main(String[] args) {
186     new BookerGUI();
187 }
188 }

```

Gambar 5 Kelas BookerGUI

Penjelasan:

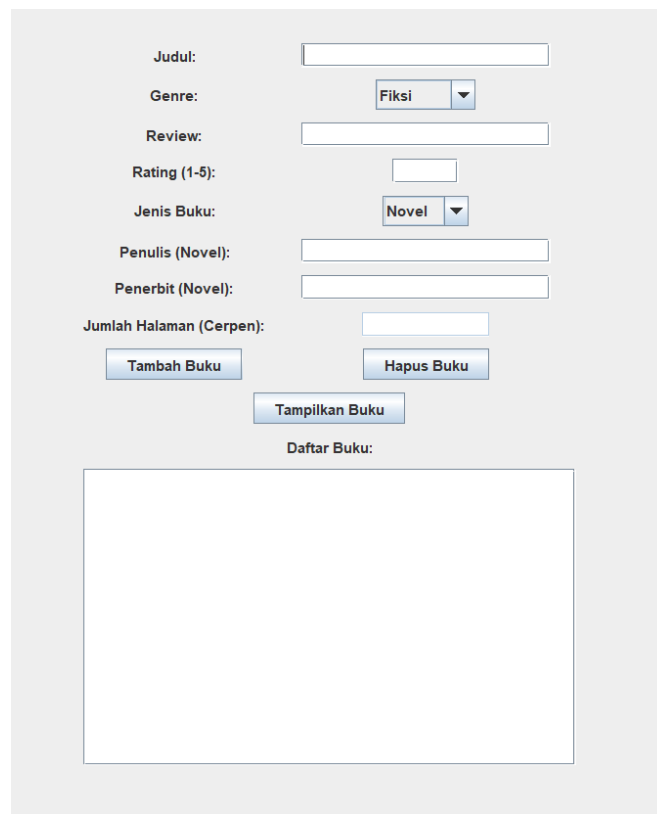
Kelas BookerGUI adalah implementasi antarmuka grafis (GUI) untuk mengelola data buku menggunakan Java Swing. Program ini menyediakan fitur untuk

menambahkan, menghapus, dan menampilkan daftar buku, baik berjenis Novel maupun Cerpen, yang dikelola melalui kelas BookManager. Komponen GUI seperti JTextField, JComboBox, dan JButton digunakan untuk menerima input pengguna, memilih jenis buku, serta mengelola data.

Saat jenis buku dipilih (Novel atau Cerpen), bidang input yang relevan akan diaktifkan atau dinonaktifkan secara dinamis menggunakan listener pada bookTypeBox. Metode addBook menambahkan buku ke daftar berdasarkan input pengguna, dengan validasi data. Metode deleteBook meminta judul buku untuk dihapus, sedangkan displayBooks menampilkan daftar buku yang ada. Semua operasi memanfaatkan fungsi dari BookManager untuk memanipulasi daftar buku. Program diinisialisasi melalui metode main, yang memanggil konstruktor BookerGUI untuk membuat dan menampilkan antarmuka aplikasi.

IV. Hasil dan Analisis

Output dari program yang telah dibuat adalah sebagai berikut:



The screenshot displays the user interface of the Booker application. It features several input fields for book details: 'Judul:' (Title), 'Genre:' (with a dropdown menu showing 'Fiksi'), 'Review:', 'Rating (1-5):', 'Jenis Buku:' (with a dropdown menu showing 'Novel'), 'Penulis (Novel):', 'Penerbit (Novel):', and 'Jumlah Halaman (Cerpen):'. Below these fields are three buttons: 'Tambah Buku' (Add Book), 'Hapus Buku' (Delete Book), and 'Tampilkan Buku' (Display Book). At the bottom, there is a label 'Daftar Buku:' (Book List) followed by a large empty rectangular box for displaying the list of books.

Gambar 6 Output Program

Output GUI pada gambar di atas menunjukkan aplikasi Booker telah berjalan dengan baik dan memenuhi fungsionalitas yang dirancang. Berikut adalah fitur yang berhasil diimplementasikan berdasarkan output:

1. Input Data Buku:

Terdapat field untuk memasukkan data seperti judul, genre, review, rating, jenis buku (Novel atau Cerpen), penulis, penerbit, dan jumlah halaman.

2. Pengelolaan Dinamis:

Penyesuaian Field Berdasarkan Jenis Buku:

- Jika jenis buku dipilih sebagai "Novel," hanya field "Penulis" dan "Penerbit" yang aktif.
- Jika jenis buku dipilih sebagai "Cerpen," hanya field "Jumlah Halaman" yang aktif.

3. Operasi Buku:

- Tambah Buku: Untuk menambahkan data buku ke dalam daftar.
- Hapus Buku: Untuk menghapus buku berdasarkan judul.

- Tampilkan Buku: Untuk menampilkan daftar buku yang sudah dimasukkan di area "Daftar Buku."

4. Tampilan Buku:

Area teks di bawah label "Daftar Buku" berfungsi untuk menampilkan detail buku yang telah ditambahkan.

Percobaan menambahkan buku:

Judul: Hanya Imajinasi

Genre: Fantasi

Review: Hmmm menarik

Rating (1-5): 4

Jenis Buku: Cerpen

Penulis (Novel):

Penerbit (Novel):

Jumlah Halaman (Cerpen): 108

Tambah Buku Hapus Buku

Tampilkan Buku

Daftar Buku:

Gambar 7 Percobaan menambahkan buku

Pada percobaan ini, mengisi data untuk menambahkan sebuah buku dengan jenis cerpen. Ketika memilih jenis bukunya adalah cerpen, maka pilihan penulis dan penerbit tidak bisa diisi. Begitu juga jika memilih buku jenis novel, maka kolom/pilihan penulis dan penerbit dapat diisi tetapi bagian jumlah halaman tidak bisa diisi karena mengimplementasikan konsep inheritance.

Berikut hasilnya setelah melakukan pengisian data:

Tampilkan Buku

Daftar Buku:

Cerpen: Hanya Imajinasi
Genre: Fantasi
Review: Hmmm menarik
Rating: 4
Jumlah Halaman: 108

Gambar 8 Hasil Percobaan

V. Kesimpulan

Kesimpulan dari pengembangan aplikasi Booker yaitu aplikasi ini adalah sistem manajemen buku berbasis antarmuka grafis yang memungkinkan pengguna untuk mengelola koleksi buku mereka dengan mudah. Pengguna dapat menambahkan buku jenis novel atau cerpen, dengan mengisi informasi seperti judul, genre, review, rating, serta atribut khusus sesuai dengan jenis buku yang dipilih. Untuk novel, pengguna perlu memasukkan nama penulis dan penerbit, sementara untuk cerpen, informasi yang diperlukan adalah jumlah halaman. Setelah buku ditambahkan, pengguna dapat melihat daftar buku lengkap dengan detailnya, atau menghapus buku tertentu berdasarkan judul yang dimasukkan.

Fungsi utama aplikasi ini mencakup penambahan buku, penghapusan buku, dan menampilkan daftar buku. Antarmuka grafis yang digunakan dirancang agar mudah dipahami, dengan tombol dan kolom input yang jelas untuk setiap fungsi. Secara keseluruhan, aplikasi ini menyediakan cara yang efisien dan ramah pengguna untuk mengelola berbagai jenis buku dalam koleksi pribadi.

VI. Referensi

Link GitHub Project Aplikasi: <https://github.com/Razhboi/Semester-3/tree/main/Semester%203/Praktikum%20PBO/Aplikasi%20Booker>

Khoir, M. (2021). Pemrograman Berorientasi Objek: Dasar dan Implementasi dengan Java. Graha Ilmu.

Marlinda, S. (2019). Pemrograman Berorientasi Objek dengan Java: Panduan Lengkap untuk Pemula.

Saputra, D., & Anggraini, H. (2020). Pemrograman Berorientasi Objek dengan Java untuk Pemula. Penerbit Elex Media Komputindo.

Siddik, M., & Sirait, A. (2018). Pengembangan Sistem Informasi Admisnistrasi Akademik dengan Rancangan Modul Program Menggunakan Bahasa Pemrograman Berorientasi Objek. *JOISIE Jurnal of Information System And Informatics Engineering*, 51-57.

Wahana Komputer. (2012). Pemrograman Berorientasi Objek dengan Java. Penerbit Andi.