

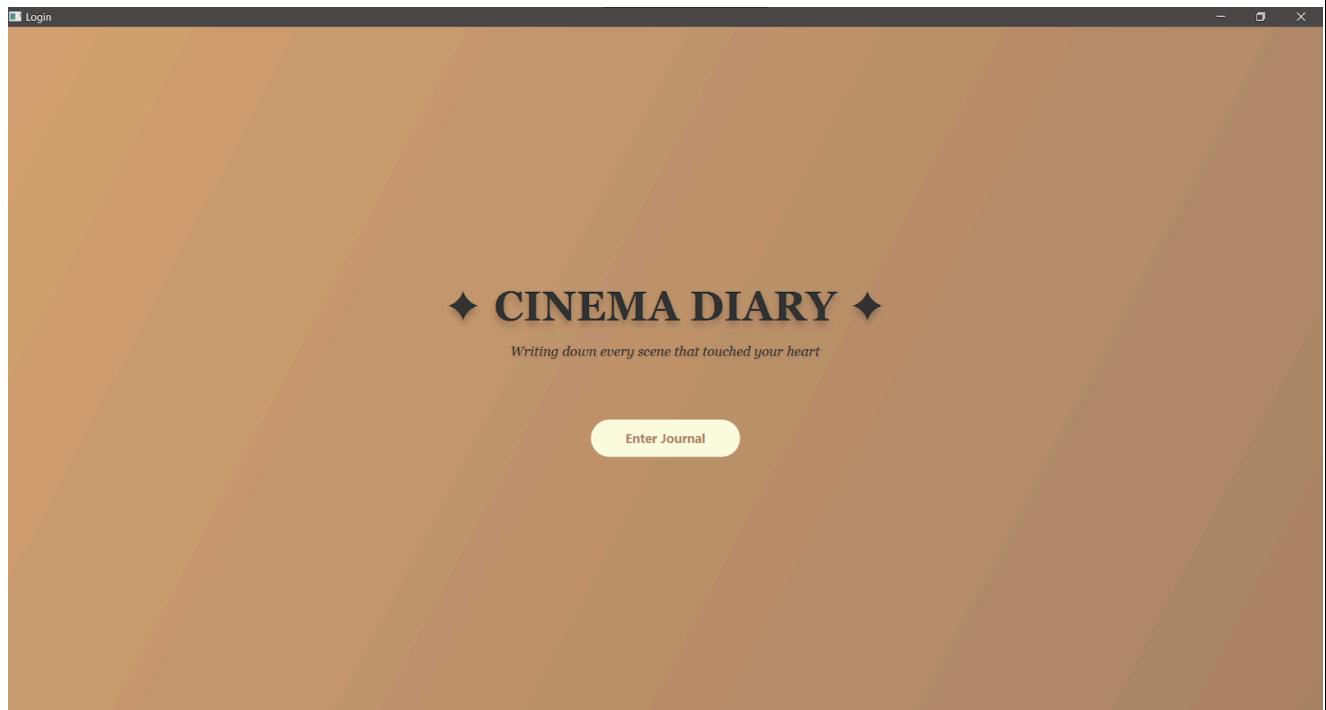
RESPONSI II
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK
2025/2026

IDENTITAS

Nama : Aprillia Wulandari
NIM : L0124040
Kelas : C
Judul Program : Cinema Diary
Deskripsi Program : Cinema Diary adalah aplikasi berbasis JavaFX yang digunakan untuk pencatatan daftar film dan serial yang telah ditonton. Aplikasi ini memungkinkan pengguna untuk menambahkan, mencari, memfilter, dan menghapus data tontonan, serta menyimpan data tersebut secara permanen ke dalam file teks (txt). Dengan demikian, aplikasi ini membantu pengguna dalam mengelola daftar film dan serial yang telah ditonton secara terorganisir.

DOKUMENTASI PROGRAM

Instruksi Pemakaian:



Ketika kode dieksekusi, maka akan muncul halaman *login* seperti gambar di atas. Kemudian klik tombol “Enter Journal” untuk masuk ke halaman utama aplikasi.



Gambar di atas merupakan halaman utama yang digunakan untuk mencatat dan mengelola jurnal tontonan movie dan series. Pengguna dapat menambahkan ulasan baru serta melihat daftar tontonan yang telah disimpan.



Misalkan pengguna ingin menambahkan film Harry Potter, pengguna terlebih dahulu mengisi judul film pada kolom *Title*, kemudian memilih tipe *Movie* dan genre *Fantasy*. Selanjutnya, pengguna mengisi kolom deskripsi. Setelah seluruh data terisi, pengguna dapat menekan tombol “+ Add”.

The screenshot shows the Cinema Diary application interface. On the left, there is a form titled "Write a Review" with fields for Title, Select Type (dropdown), Select Genre (dropdown), Rating (e.g. 4.5), Short description, and buttons for + Add and Delete. Below this is a search bar with placeholder "Search title..." and filters for All, Movie, and Series. On the right, there is a table titled "My Watch Journey" with columns: Title, Type, Genre, Rating, Status, and Description. A "Completed" button is at the top right of the table. The table lists various movies and series with their details and user notes.

Title	Type	Genre	Rating	Status	Description
The Little Rascal	Movie	Family	5.0	Done	you are so beautiful, for meeeee, can u seeee
Anne With An E	Series	Family	4.5	Done	carrot hair
Dead Poets Society	Movie	Family	4.8	Done	i dont like how neil's parents treat him
Evil Dead Rise	Movie	Horror	4.0	Done	aa...
Enola Holmes	Movie	Action	5.0	Undone	how to be a detective
Gilmore Girls	Series	Family	4.2	Undone	omg this series
Pride and Prejudice	Movie	Romance	4.8	Undone	omg omg i like them
Sherlock Holmes	Movie	Action	5.0	Undone	enola's brother
The History of Everything	Movie	Sci-Fi	5.0	Undone	the main char is ateis
The Medium	Movie	Horror	4.0	Undone	emmm begitu lah
The Texas Chainsaw	Movie	Thriller	4.2	Undone	omgg killed by chainsaw
Harry Potter	Movie	Fantasy	5.0	Undone	i'm not a muggle:)

Berdasarkan gambar di atas, dapat dilihat bahwa film Harry Potter berhasil ditambahkan dan ditampilkan pada tabel My Watch Journey.

The screenshot shows the Cinema Diary application interface. On the left, there is a form titled "Write a Review" with fields for Title, Select Type (dropdown), Select Genre (dropdown), Rating (e.g. 4.5), Short description, and buttons for + Add and Delete. Below this is a search bar with placeholder "Search title..." and filters for All, Movie, and Series. On the right, there is a table titled "My Watch Journey" with columns: Title, Type, Genre, Rating, Status, and Description. A "Completed" button is at the top right of the table. The table lists various movies and series with their details and user notes. The status for the Harry Potter row has changed from "Undone" to "Done".

Title	Type	Genre	Rating	Status	Description
The Little Rascal	Movie	Family	5.0	Done	you are so beautiful, for meeeee, can u seeee
Anne With An E	Series	Family	4.5	Done	carrot hair
Dead Poets Society	Movie	Family	4.8	Done	i dont like how neil's parents treat him
Evil Dead Rise	Movie	Horror	4.0	Done	aa...
Enola Holmes	Movie	Action	5.0	Undone	how to be a detective
Gilmore Girls	Series	Family	4.2	Undone	omg this series
Pride and Prejudice	Movie	Romance	4.8	Undone	omg omg i like them
Sherlock Holmes	Movie	Action	5.0	Undone	enola's brother
The History of Everything	Movie	Sci-Fi	5.0	Undone	the main char is ateis
The Medium	Movie	Horror	4.0	Undone	emmm begitu lah
The Texas Chainsaw	Movie	Thriller	4.2	Undone	omgg killed by chainsaw
Harry Potter	Movie	Fantasy	5.0	Done	i'm not a muggle:)

Pengguna dapat menandai film yang telah selesai ditonton dengan memilih baris film hingga berubah warna menjadi coklat, kemudian menekan tombol "Completed". Pada gambar di atas, terlihat bahwa status film Harry Potter berubah dari *undone* menjadi *done*.

Cinephile Scribbles

Write a Review

Title: interstellar
Select Type: Movie
Select Genre: Family
Rating (e.g. 4.5):
Short description:
+ Add Delete

Search title... All Movie Series

My Watch Journey

Title	Type	Genre	Rating	Status	Description
interstellar	Movie	Family	5.0	Done	about blackhole
The Little Rascal	Movie	Family	5.0	Done	you are so beautiful, for meeeee, can u seeee
Anne With An E	Series	Family	4.5	Done	carrot hair
Dead Poets Society	Movie	Family	4.8	Done	i dont like how neil's parents treat him
Evil Dead Rise	Movie	Horror	4.0	Done	aa...
Enola Holmes	Movie	Action	5.0	Undone	how to be a detective
Gilmore Girls	Series	Family	4.2	Undone	omg this series
Pride and Prejudice	Movie	Romance	4.8	Undone	omg omg i like them
Sherlock Holmes	Movie	Action	5.0	Undone	enola's brother
The History of Everything	Movie	Sci-Fi	5.0	Undone	the main char is ateis
The Medium	Movie	Horror	4.0	Undone	emmm begitu lah
The Texas Chainsaw	Movie	Thriller	4.2	Undone	omgg killed by chainsaw
Harry Potter	Movie	Fantasy	5.0	Done	i'm not a muggle:)

Pengguna dapat menghapus film yang telah selesai ditonton dengan memilih baris film hingga berubah warna menjadi coklat, kemudian menekan tombol “Delete”.

Cinephile Scribbles

Write a Review

Title: interstellar
Select Type: Movie
Select Genre: Family
Rating (e.g. 4.5):
Short description:
+ Add Delete

Search title... All Movie Series

My Watch Journey

Title	Type	Genre	Rating	Status	Description
interstellar	Movie	Family	5.0	Done	about blackhole
The Little Rascal	Movie	Family	5.0	Done	you are so beautiful, for meeeee, can u seeee
Anne With An E	Series	Family	4.5	Done	carrot hair
Dead Poets Society	Movie	Family	4.8	Done	i dont like how neil's parents treat him
Evil Dead Rise	Movie	Horror	4.0	Done	aa...
Enola Holmes	Movie	Action	5.0	Undone	how to be a detective
Gilmore Girls	Series	Family	4.2	Undone	omg this series
Pride and Prejudice	Movie	Romance	4.8	Undone	omg omg i like them
Sherlock Holmes	Movie	Action	5.0	Undone	enola's brother
The History of Everything	Movie	Sci-Fi	5.0	Undone	the main char is ateis
The Medium	Movie	Horror	4.0	Undone	emmm begitu lah
Harry Potter	Movie	Fantasy	5.0	Done	i'm not a muggle:)

Pada gambar di atas, terlihat bahwa pada awalnya terdapat judul film “The Texas Chainsaw”, tetapi setelah proses penghapusan dilakukan, film tersebut sudah tidak lagi muncul pada tabel My Watch Journey.



Apabila pengguna ingin melakukan pencarian film, pengguna dapat mengetikkan *keywords* pada kolom pencarian. Ketika pengguna mengetikkan kata “ha”, maka secara *real-time*, sistem akan menampilkan film-film yang judulnya mengandung *keywords* tersebut, seperti yang ditunjukkan pada gambar di atas.



Gambar di atas menunjukkan tampilan tabel ketika pengguna menekan tombol filter “All”, sehingga seluruh data movie dan series ditampilkan.



Gambar di atas menunjukkan tampilan tabel ketika pengguna menekan tombol filter “Movie”, di mana hanya data bertipe movie yang ditampilkan.



Gambar di atas menunjukkan tampilan tabel ketika pengguna menekan tombol filter “Series”, sehingga hanya data bertipe series yang ditampilkan.



Gambar di atas menunjukkan fitur *tooltip*, di mana ketika kursor diarahkan ke teks deskripsi pada tabel, sistem akan menampilkan seluruh isi deskripsi secara lengkap.



Gambar di atas menunjukkan fitur *sorting* default dari JavaFX berdasarkan judul secara alfabetis dengan klik "Title" pada kolom tabel.



Gambar di atas menunjukkan fitur *sorting* default dari JavaFX berdasarkan tipe secara alfabetis dengan klik “Type” pada kolom tabel.



Gambar di atas menunjukkan fitur *sorting* default dari JavaFX berdasarkan genre secara alfabetis dengan klik “Genre” pada kolom tabel.



Gambar di atas menunjukkan fitur *sorting* default dari JavaFX berdasarkan rating dengan klik “Rating” pada kolom tabel.



Gambar di atas menunjukkan fitur *sorting* default dari JavaFX berdasarkan status secara alfabetis dengan klik “Status” pada kolom tabel.



Gambar di atas menunjukkan fitur *sorting* default dari JavaFX berdasarkan deskripsi secara alfabetis dengan klik “Description” pada kolom tabel.

Fitur-Fitur:

1. Login aplikasi

Aplikasi ini menyediakan halaman *login* sebagai tampilan awal sebelum pengguna masuk ke halaman utama. Pengguna dapat masuk ke halaman utama dengan menekan tombol “Enter Journal”.

2. Menampilkan daftar movie atau series

Pada halaman utama, aplikasi menampilkan seluruh data movie dan series yang telah ditambahkan dalam bentuk tabel.

3. Menambahkan movie atau series

Pengguna dapat menambahkan data movie atau series dengan mengisi judul, tipe media (movie atau series), genre, rating, dan deskripsi singkat. Data yang ditambahkan akan langsung ditampilkan pada tabel.

4. Menghapus data movie atau series

Pengguna dapat menghapus data movie atau series dengan memilih baris pada tabel hingga berwarna coklat dan kemudian menekan tombol “Delete”.

5. Menandai film yang sudah ditonton

Aplikasi ini menyediakan fitur “Completed” untuk menandai movie atau series yang sudah ditonton dengan status *Done* atau *Undone*.

6. Pencarian film secara *real-time*

Pengguna dapat mencari film berdasarkan judul melalui kolom pencarian. Hasil pencarian akan ditampilkan secara *real-time* saat pengguna mengetik.

7. Filter berdasarkan tipe film

Aplikasi ini menyediakan fitur-fitur untuk menampilkan semua data “All”, hanya movie “Movie”, dan hanya series “Series”.

8. Tooltip deskripsi

Aplikasi ini menyediakan fitur *tooltip* deskripsi untuk menampilkan seluruh teks deskripsi ketika

kurSOR diarahkan ke kolom deskripsi pada tabel sehingga pengguna dapat membaca informasi secara lengkap.

9. Sorting data

Aplikasi ini menyediakan fitur *sorting* default dari *TableView* JavaFX yang memungkinkan pengguna mengurutkan data dengan mengklik judul kolom tabel. Data dapat diurutkan berdasarkan judul, tipe, genre, status, dan deskripsi secara alfabetis, serta berdasarkan *rating* secara numerik.

10. Penyimpanan data otomatis

Data movie atau series disimpan secara otomatis ke dalam *file* teks menggunakan proses *asynchronous* sehingga data tidak hilang saat aplikasi ditutup.

11. Pemuatan data otomatis

Saat aplikasi dijalankan kembali, data yang tersimpan akan dimuat secara otomatis dan ditampilkan kembali ke tabel.

12. Antarmuka Grafis Interaktif

Aplikasi ini menggunakan JavaFX dengan FXML dan CSS untuk memberikan tampilan antarmuka yang interaktif.

Penjelasan Source Code:

1. Controller

a. LoginController.java

```
1 package film.com.controller;
2
3 import javafx.event.ActionEvent;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Node;
6 import javafx.scene.Parent;
7 import javafx.scene.Scene;
8 import javafx.stage.Stage;
9
10 /**
11 * Controller untuk halaman login aplikasi Cinema Diary.
12 * <p>
13 * Kelas ini menangani proses perpindahan tampilan
14 * dari halaman login ke halaman utama aplikasi.
15 * </p>
16 *
17 * @author aprliwln
18 */
```

Package “[film.com.controller](#)” menandakan bahwa kelas ini berada di dalam folder “controller”. Beberapa *import* yang digunakan, yaitu “javafx.event.ActionEvent” berfungsi untuk menangani aksi dari pengguna, seperti klik tombol, “javafx.fxml FXMLLoader” berfungsi untuk memuat *file* tampilan berbasis FXML, “javafx.scene.Node” dana

“javafx.scene.Parent” berfungsi sebagai komponen antarmuka JavaFX, serta “javafx.scene.Scene” dan “javafx.stage.Stage” berfungsi untuk mengatur dan menampilkan jendela (window) aplikasi.

```
19 public class LoginController {  
20     /**  
21      * Dipanggil ketika user menekan tombol login.  
22      * <p>  
23      * Method ini akan memuat file {@code main.fxml}  
24      * dan mengganti scene login dengan scene utama.  
25      * </p>  
26      *  
27      * @param event aksi klik tombol login  
28      */  
29     public void login(ActionEvent event) {  
30         try {  
31             //load file main.fxml sbg tampilan setelah login  
32             Parent root = FXMLLoader.load(getClass().getResource("/film/com/view/main.fxml"));  
33  
34             //mengambil stage yg sedang aktif dr event tombol  
35             Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();  
36             stage.setScene(new Scene(root)); //mengganti scene login dgn scene utama  
37             stage.setTitle(value: "Cinema Diary");  
38         } catch (Exception e) {  
39             //jika terjadi kesalahan saat load fxml  
40             e.printStackTrace();  
41         }  
42     }  
43 }  
44 }
```

Kelas “LoginController” digunakan untuk menangani proses *login* pada aplikasi Cinema Diary. Di dalam kelas ini terdapat *method* “void login(ActionEvent event)” yang dipanggil ketika pengguna menekan tombol *login*. Pada *method* ini, “FXMLLoader.load()” digunakan untuk memuat *file* “main.fxml” sebagai tampilan halaman utama setelah *login*. Selanjutnya, objek *Stage* diambil dari *event* tombol yang sedang aktif dengan memanfaatkan *Node*, sehingga aplikasi mengetahui jendela mana yang sedang digunakan. Setelah itu *scene* pada *stage* digantikan dengan *scene* baru yang berisi tampilan utama menggunakan “stage.setScene(new Scene(root))” dan judul jendela menjadi “Cinema Diary”. Blok *try-catch* digunakan untuk menangani kemungkinan error saat proses pemuatan *file* FXML yang ditampilkan melalui “e.printStackTrace()”.

b. MainController.java

```
1 package film.com.controller;
2
3 import film.com.model.*;
4 import javafx.beans.property.SimpleStringProperty;
5 import javafx.fxml.FXML;
6 import javafx.scene.control.*;
7 import javafx.scene.control.cell.PropertyValueFactory;
8
9 import java.util.ArrayList;
10 import java.util.List;
11
12 /**
13 * Controller utama aplikasi Cinema Diary.
14 * <p>
15 * Mengelola input data film/series, filtering,
16 * pencarian, penandaan, serta interaksi TableView.
17 * </p>
18 *
19 * @author aprliwln
20 */
```

Package “[film.com.controller](#)” menandakan bahwa kelas ini berada di dalam folder “controller”. Beberapa import yang digunakan, yaitu “[film.com.model.*](#)” berfungsi untuk mengakses kelas-kelas model data movie atau series, “javafx.beans.property.SimpleStringProperty” berfungsi untuk mengelola dan menampilkan data properti dalam komponen JavaFX, serta “javafx.fxml.FXMLLoader” berfungsi untuk memuat file tampilan berbasis FXML. Selain itu, “javafx.scene.control.*” berfungsi untuk mengakses berbagai komponen kontrol antarmuka seperti Button, TextField, dan TableView, sedangkan “javafx.scene.control.cell.PropertyValueFactory” berfungsi untuk menghubungkan atribut data dengan kolom pada “TableView”. Import “java.util.ArrayList” dan “java.util.List” digunakan untuk mengelola kumpulan data movie atau series.

```
21 public class MainController {
22     /**Komponen Form Input */
23     @FXML private TextField titleField;
24     @FXML private ComboBox<String> typeBox;
25     @FXML private ComboBox<String> genreBox;
26     @FXML private TextField ratingField;
27     @FXML private TextField searchField;
28     @FXML private TextArea descArea;
29     /**Komponen Table View */
30     @FXML private TableView<CinemaItem> tableView;
31     @FXML private TableColumn<CinemaItem, String> titleCol;
32     @FXML private TableColumn<CinemaItem, String> typeCol;
33     @FXML private TableColumn<CinemaItem, String> genreCol;
34     @FXML private TableColumn<CinemaItem, Double> ratingCol;
35     @FXML private TableColumn<CinemaItem, String> descCol;
36     @FXML private TableColumn<CinemaItem, String> markedCol;
37     /**Menyimpan seluruh data media */
38     private List<CinemaItem> cinemaList = new ArrayList<>();
```

Kelas “MainController” berfungsi mengelola interaksi pengguna pada halaman utama aplikasi Cinema Diary. Atribut yang diberi notasi “@FXML” menandakan bahwa komponen tersebut terhubung langsung dengan elemen pada file FXML. Komponen *form input*, seperti “TextField”, “ComboBox”, dan “TextArea” digunakan untuk memasukkan dan mencari data movie atau series. Selain itu, “TableView” beserta “TableColumn” digunakan untuk menampilkan data media dalam bentuk tabel. Atribut “cinemaList” berfungsi untuk menyimpan seluruh data movie atau series yang ada di aplikasi.

```

40     /**
41      * Dipanggil otomatis saat FXML dimuat.
42      * <p>
43      * Method ini mengatur:
44      * </p>
45      * <ul>
46      *   <li>Isi ComboBox</li>
47      *   <li>Koneksi TableColumn</li>
48      *   <li>Tooltip deskripsi</li>
49      *   <li>Load data async</li>
50      *   <li>Pencarian realtime</li>
51      * </ul>
52      */
53  public void initialize() {
54      //pilihan tipe film
55      typeBox.getItems().addAll(...elements: "Movie", "Series");
56      //pilihan genre
57      genreBox.getItems().addAll(...elements: "Action", "Fantasy",
58      |           "Romance", "Sci-Fi", "Horror", "Family", "Thriller");
59
60      //menghubungkan kolom tabel dgn properti media item
61      titleCol.setCellValueFactory(new PropertyValueFactory<>(property: "title"));
62      typeCol.setCellValueFactory(new PropertyValueFactory<>(property: "type"));
63      genreCol.setCellValueFactory(new PropertyValueFactory<>(property: "genre"));
64      ratingCol.setCellValueFactory(new PropertyValueFactory<>(property: "rating"));
65      descCol.setCellValueFactory(new PropertyValueFactory<>(property: "description"));

```

Method “initialize()” digunakan untuk melakukan inisialisasi komponen saat tampilan utama pertama kali dimuat. Pada *method* ini, “typeBox” diisi dengan pilihan tipe media, yaitu movie dan series, sedangkan “genreBox” diisi dengan beberapa pilihan genre film. Selain itu, setiap kolom pada “TableView” dihubungkan dengan properti yang sesuai dari objek media menggunakan “PropertyValueFactory”.

```

67     //mengatur lebar kolom deskripsi
68     descCol.setMaxWidth(Double.MAX_VALUE); // Membuang batasan lebar maksimal
69     tableView.widthProperty().addListener((obs, oldVal, newVal) -> {
70         double totalWidth = newVal.doubleValue();
71         double otherColsWidth = titleCol.getWidth() + typeCol.getWidth() +
72             |   |   |   |   |   genreCol.getWidth() + ratingCol.getWidth() +
73             |   |   |   |   |   markedCol.getWidth();
74         //sisa ruang
75         double buffer = 25;
76         descCol.setPrefWidth(totalWidth - othercolsWidth - buffer);
77     });
78
79     //utk deskripsi yg panjang, ketika kursor diarahkan ke teks, maka seluruh teks akan terlihat
80     descCol.setCellFactory(column -> new TableCell<CinemaItem, String>() {
81         private final Tooltip tooltip = new Tooltip();
82         @Override
83         protected void updateItem(String item, boolean empty) {
84             super.updateItem(item, empty);
85
86             if (item == null || empty) {
87                 setText(value: null);
88                 setTooltip(value: null);
89             } else {
90                 setText(item);
91                 tooltip.setText(item);
92
93                 tooltip.setPrefWidth(value: 250);
94                 tooltip.setWrapText(value: true);
95                 setTooltip(tooltip);
96             }
97         }
98     });

```

Kode di atas berfungsi untuk mengatur tampilan kolom deskripsi pada “TableView”. Pertama, Kolom “descCol” diatur agar tidak memiliki batas lebar maksimum, kemudian lebarnya disesuaikan secara otomatis berdasarkan lebar “TableView” dengan menghitung sisa ruang setelah dikurangi lebar kolom lainnya. Selain itu dibuat *custom cell factory* agar teks deskripsi yang panjang tetap dapat ditampilkan. Jika teks terlalu panjang dan tidak muat di dalam sel, pengguna dapat mengarahkan kursor ke teks tersebut untuk menampilkan seluruh isi deskripsi melalui “Tooltip”.

```

100    //menampilkan status pada film
101    markedCol.setCellValueFactory(cellData -> {
102        CinemaItem item = cellData.getValue();
103        if (item instanceof Marked m) {
104            return new SimpleStringProperty(m.isMarked() ? "Done" : "Undone"); //markable
105        } else {
106            return new SimpleStringProperty(initialValue: "N/A"); //kalau tidak markable
107        }
108    });
109
110    //mengatur teks default tipe film
111    typeBox.setButtonCell(new ListCell<>() {
112        @Override
113        protected void updateItem(String item, boolean empty) {
114            super.updateItem(item, empty);
115            setText(empty || item == null ? "Select Type" : item);
116        }
117    });

```

Pada bagian “markedCol”, “setCellValueFactory” digunakan untuk menentukan nilai yang ditampilkan pada kolom status. Jika objek yang ditampilkan merupakan turunan dari “Marked”, maka status akan diperiksa melalui method “isMarked()”. Namun, jika data film tidak mendukung fitur penandaan, maka kolom status akan menampilkan teks “N/A”. Selain itu, pada komponen “typeBox”, “setButtonCell” digunakan untuk menampilkan teks “Select Type” ketika belum ada tipe film yang dipilih.

```

119    //mengatur teks default genre
120    genreBox.setButtonCell(new ListCell<>() {
121        @Override
122        protected void updateItem(String item, boolean empty) {
123            super.updateItem(item, empty);
124            setText(empty || item == null ? "Select Genre" : item);
125        }
126    });
127
128    //load data dari file scr asinkron (multithreading)
129    MediaStorage.loadAsync(data -> {
130        cinemaList = data; //simpan ke list utama
131        tableView.getItems().addAll(cinemaList); //tampilkan ke tabel
132    });
133
134    //untuk pencarian realtime
135    searchField.textProperty().addListener(obs, o, n) -> searchCinema());
136 }

```

Pada “genreBox”, “setButtonCell” digunakan untuk menampilkan teks “Select Genre” ketika belum ada genre yang dipilih. Selanjutnya, “MediaStorage.loadAsync” digunakan untuk memuat data movie atau series dari file secara asinkron menggunakan multithreading, di mana data yang diterima disimpan ke “cinemaList” dan langsung ditampilkan di “TableView”. Terakhir, *listener* pada “searchCinema()” dipanggil setiap kali pengguna mengetik yang memungkinkan pencarian data yang cepat.

```

138     /**Menampilkan semua data ke tabel*/
139     @FXML
140     public void showAll() {
141         tableView.getItems().setAll(cinemaList);
142     }
143     /**Filter hanya Movie */
144     @FXML
145     public void filterMovie() {
146         filterByType(type: "Movie");
147     }
148     /**Filter hanya series */
149     @FXML
150     public void filterSeries() {
151         filterByType(type: "Series");
152     }
153     /**Filter berdasarkan tipe */
154     private void filterByType(String type) {
155         tableView.getItems().setAll(
156             cinemaList.stream().filter(m -> m.getType().equals(type)).toList());
157     }

```

Method “shwoAll()” digunakan untuk menampilkan seluruh data movie atau series ke dalam tabel. Method “filterMovie()” digunakan untuk menampilkan hanya data dengan tipe movie, sedangkan method “filterSeries()” digunakan untuk menampilkan hanya data dengan tipe Series. Sementara itu, method “filterByType(String Type)” merupakan method pendukung yang digunakan untuk menyaring data berdasarkan tipe tertentu dan menampilkan hasilnya di “TableView”.

```

159     /**Pencarian berdasarkan judul */
160     @FXML
161     public void searchCinema() {
162         String keyword = searchField.getText().toLowerCase();
163
164         tableView.getItems().clear();
165
166         for (CinemaItem m : cinemaList) {
167             if (m.getTitle().toLowerCase().contains(keyword)) {
168                 tableView.getItems().add(m);
169             }
170         }
171     }
172
173     /**Menandai film yang sudah ditonton */
174     @FXML
175     public void marked() {
176         CinemaItem selected = tableView.getSelectionModel().getSelectedItem();
177         if (selected != null && selected instanceof Marked m) {
178             m.setMark(mark: true);
179             tableView.refresh();
180             MediaStorage.saveAsync(cinemaList);
181         }
182     }

```

Method “searchCinema()” digunakan untuk melakukan pencarian data berdasarkan judul, di

mana teks yang dimasukkan ke “searchField” akan dibandingkan dengan judul setiap item dan hanya item yang sesuai *keywords* akan ditampilkan. Sementara itu, *method* “marked()” digunakan untuk menandai movie atau series yang sudah ditonton. *Method* ini akan memeriksa item yang dipilih di tabel, kemudian jika item tersebut mendukung fitur *Marked*, statusnya diubah menjadi *marked*, tabel diperbarui menggunakan “tableView.refresh()” dan perubahan disimpan secara asinkron ke media penyimpanan melalui “MediaStorage.saveAsync(cinemaList)”.

```

184     /**Menambah film/series baru */
185     @FXML
186     public void addCinema() {
187         String title = titleField.getText();
188         String ratingText = ratingField.getText();
189         String desc = descArea.getText();
190         String type = typeBox.getValue();
191         String genre = genreBox.getValue();
192
193         //validasi jika judul kosong
194         if (title.isEmpty() || ratingText.isEmpty() || type == null) {
195             showAlert(title: "Error", message: "Title, Type, dan Rating wajib diisi!");
196             return;
197         }
198
199         //validasi rating
200         double rating;
201         try {
202             rating = Double.parseDouble(ratingText);
203         } catch (NumberFormatException e) {
204             showAlert(title: "Error", message: "Rating harus berupa angka (contoh: 4.5)");
205             return;
206         }
207         if (rating < 0 || rating > 5) {
208             showAlert(title: "Error", message: "Rating harus antara 0 sampai 5!");
209             return;
210         }

```

Method “addCinema()” digunakan untuk menambah movie atau series baru ke dalam aplikasi. Pertama, *method* ini mengambil *input* dari *form*, yaitu “titleField”, “ratingField”, “descArea”, “typeBox”, dan “genreBox”. Selanjutnya dilakukan validasi judul, tipe, dan rating yang wajib diisi, serta rating harus berupa angka antara 0 sampai 5.

```

212         //membuat objek Cinema item sesuai tipe
213         CinemaItem item = switch (type) {
214             case "Movie" -> new Movie(title, rating, desc, genre);
215             default -> new Series(title, rating, desc, genre);
216         };
217         //menambahkan ke list dan tabel
218         cinemaList.add(item);
219         tableView.getItems().add(item);
220         //simpan ke file scr asinkron
221         MediaStorage.saveAsync(cinemaList);
222         clearForm();
223     }

```

Setelah validasi berhasil, objek “CinemaItem” dibuat sesuai tipe yang dipilih, yaitu movie atau series. Item baru kemudian ditambahkan ke “cinemaList” dan ditampilkan di “TableView”. Terakhir, data disimpan secara asinkron ke file menggunakan “MediaStorage.saveAsync(cinemaList), dan form input dibersihkan melalui “clearForm()”.

```
225  /**Menghapus data */
226  @FXML
227  public void deleteCinema() {
228      CinemaItem selected = tableView.getSelectionModel().getSelectedItem();
229      if (selected != null) {
230          cinemaList.remove(selected);
231          tableView.getItems().remove(selected);
232          MediaStorage.saveAsync(cinemaList); //simpan perubahan
233      }
234  }
235
236  /**Menampilkan dialog error */
237  private void showAlert(String title, String message) {
238      Alert alert = new Alert(Alert.AlertType.ERROR);
239      alert.setTitle(title);
240      alert.setHeaderText(headerText: null);
241      alert.setContentText(message);
242      alert.showAndWait();
243  }
244
245  /**Membersihkan form */
246  private void clearForm() {
247      titleField.clear();
248      ratingField.clear();
249      descArea.clear();
250
251      typeBox.setValue(value: null);
252      genreBox.setValue(value: null);
253  }
254 }
```

Method “deleteCinema()” digunakan untuk menghapus movie atau series yang dipilih di “TableView”, di mana item dihapus dari “cinemaList” sekaligus dari tabel, kemudian perubahan disimpan secara asinkron menggunakan “MediaStorage.saveAsync()”. Method “showAlert(String title, String message)” digunakan untuk menampilkan dialog error kepada pengguna, dengan judul dan pesan yang ditentukan. Terakhir, method “clearForm()” digunakan untuk membersihkan semua input pada form, termasuk “TextField”, “TextArea”, dan “ComboBox”.

2. Model

a. CinemaItem.java

```

1 package film.com.model;
2
3 /**
4 * Kelas abstrak {@code CinemaItem} merepresentasikan
5 * media hiburan berupa movie atau series.
6 * <p>
7 * Kelas ini menjadi parent (superclass) bagi
8 * {@link Movie} dan {@link Series}.
9 * </p>
10 *
11 * @author aprliwln
12 */
13
14 public abstract class CinemaItem{
15     /**Judul media */
16     protected String title;
17     /**Rating media (0-5) */
18     protected double rating;
19     /**Deskripsi singkat media */
20     protected String description;
21     /**Genre media */
22     protected String genre;

```

Package “[film.com](#).model” menandakan bahwa kelas ini berada di dalam folder “model”. Kelas abstrak “CinemaItem” digunakan untuk merepresentasikan media berupa movie atau series. Beberapa atribut yang dimiliki antara lain “title” untuk menyimpan judul media, “rating” untuk menyimpan nilai rating media, “description” untuk deskripsi singkat, dan “genre” untuk genre media.

```

24 /**
25 * Konstruktor untuk membuat objek CinemaItem.
26 *
27 * @param title judul media
28 * @param rating rating media
29 * @param description deskripsi media
30 * @param genre genre media
31 */
32 public CinemaItem(String title, double rating, String description, String genre) {
33     this.title = title;
34     this.rating = rating;
35     this.description = description;
36     this.genre = genre;
37 }

```

“Konstruktor” “CinemaItem(String title, double rating, String description, String genre)” digunakan untuk membuat objek baru sekaligus menginisialisasi semua atribut tersebut.

```
39     /**@return judul media */
40     public String getTitle() { return title; }
41     /**@return rating media */
42     public double getRating() { return rating; }
43     /**@return deskripsi media */
44     public String getDescription() { return description; }
45     /**@return genre media */
46     public String getGenre() { return genre; }
47
48     /**
49      * Mengembalikan tipe media.
50      *
51      * @return "Movie" atau "Series"
52      */
53     public abstract String getType();
54 }
```

Beberapa *method* pada kelas “CinemaItem” digunakan untuk mengakses informasi dasar dari media. *Method* “getTitle()” mengembalikan judul media, “getRating()” mengembalikan nilai rating, “getDescription()” mengembalikan deskripsi singkat, dan “getGenre()” mengembalikan genre media. Selain itu, terdapat *method* abstrak “getType()” yang berfungsi untuk mengembalikan tipe media.

b. Marked.java

```

1 package film.com.model;
2
3 /**
4 * Interface {@code Marked} digunakan untuk menandai
5 * media yang sudah ditonton.
6 * <p>
7 * Interface ini diimplementasikan oleh
8 * {@link Movie} dan {@link Series}.
9 * </p>
10 *
11 * @author aprliwln
12 */
13 public interface Marked {
14     /**
15      * Mengubah status penandaan media.
16      *
17      * @param mark true jika sudah ditonton
18      */
19     void setMark(boolean mark);
20
21     /**
22      * Mengecek status penandaan media.
23      *
24      * @return true jika sudah ditonton
25      */
26     boolean isMarked();
27 }

```

Package “[film.com](#).model” menandakan bahwa kelas ini berada di dalam folder “model”. Interface “Marked” digunakan untuk menandai media yang sudah ditonton. Interface ini memiliki dua *method*, yaitu “setMark(boolean mark)” yang berfungsi untuk mengubah status penandaan media menjadi sudah ditonton atau belum, serta “isMarked()” yang digunakan untuk memeriksa apakah media tersebut sudah ditonton dan mengembalikan nilai *true* jika sudah ditandai.

c. MediaStorage.java

```
1 package film.com.model;
2
3 import java.io.*;
4 import java.util.ArrayList;
5 import java.util.List;
6 import java.util.function.Consumer;
7 import javafx.application.Platform;
8
9 /**
10 * Kelas {@code MediaStorage} bertugas
11 * untuk menyimpan dan memuat data media
12 * ke dalam file teks.
13 * <p>
14 * Proses I/O dilakukan secara asynchronous
15 * agar tidak mengganggu UI JavaFX.
16 * </p>
17 *
18 * @author aprliwln
19 */
```

Package “[film.com.model](#)” menandakan bahwa kelas ini berada di dalam folder “model”. Beberapa *import* yang digunakan antara lain “[java.io.*](#)” untuk operasi *input output file*, “[java.util.ArrayList](#)” dan “[java.util.List](#)” untuk menyimpan dan mengelola koleksi data media, “[java.util.function.Consumer](#)” untuk menerima hasil secara *callback*, serta “[javafx.application.Platform](#)” digunakan untuk mengeksekusi pembaruan UI JavaFX dari *thread* lain, terutama saat proses I/O dijalankan secara asinkron.

```

20 public class Mediastorage {
21     /**Nama file penyimpanan data */
22     private static final String FILE_NAME = "media_data.txt";
23
24     /**
25      * Menyimpan data media secara asynchronous.
26      *
27      * @param list daftar media
28      */
29     public static void saveAsync(List<CinemaItem> list) {
30         new Thread(() -> save(list)).start();
31     }
32
33     /**
34      * Memuat data media secara asynchronous.
35      *
36      * @param callback fungsi callback untuk update UI
37      */
38     public static void loadAsync(Consumer<List<CinemaItem>> callback) {
39         new Thread(() -> {
40             List<CinemaItem> data = load(); //load data scr sinkron
41             Platform.runLater(() -> callback.accept(data)); //update UI di JavaFx thread
42         }).start();
43     }

```

Kelas “Mediastorage” berfungsi untuk menyimpan dan memuat data media ke dalam *file* teks. Atribut “FILE_NAME” menyimpan nama *file* tempat data disimpan, yaitu “media_data.txt”. Method “saveAsync(List<CinemaItem> list)” digunakan untuk menyimpan data secara asinkron dengan menjalankan proses penyimpanan di *thread* terpisah. Sementara itu, method “loadAsync(Consumer<List<CinemaItem>> callback)” digunakan untuk memuat data secara asinkron, di mana data akan dibaca secara sinkron, kemudian hasilnya dikirim ke *callback* dan dijalankan di JavaFX *Application Thread* menggunakan Platform.runLater(), sehingga UI dapat diperbarui dengan aman.

```

45     /**
46      * Menyimpan data media secara sinkron.
47      *
48      * @param list daftar media
49     */
50    public static void save(List<CinemaItem> list) {
51        try (PrintWriter pw = new PrintWriter(new FileWriter(FILE_NAME))) {
52            for (CinemaItem m : list) {
53                boolean marked = false;
54                if (m instanceof Marked mk) {
55                    marked = mk.isMarked(); //ambil status marked jika ada
56                }
57                pw.println(
58                    m.getType() + " | "
59                    m.getTitle() + " | "
60                    m.getRating() + " | "
61                    m.getDescription() + " | "
62                    m.getGenre() + " | "
63                    marked
64                );
65            }
66        } catch (IOException e) {
67            e.printStackTrace();
68        }
69    }

```

Method “`save(List<CinemaItem> list)`” digunakan untuk menyimpan data media secara sinkron ke dalam *file* teks. Setiap objek “`CinemaItem`” diiterasi dan ditulis ke *file*, mencakup tipe media, judul, rating, deskripsi, genre, serta status *marked* jika objek tersebut mengimplementasikan *interface* “`Marked`”. Jika terjadi kesalahan saat proses penulisan *file*, error akan ditangani dengan “`e.printStackTrace()`”.

```

71     /**
72      * Memuat data media dari file.
73      *
74      * @return list media
75     */
76    public static List<CinemaItem> load() {
77      List<CinemaItem> list = new ArrayList<>();
78      File file = new File(FILE_NAME);
79      //jika file tdk ada, kembalikan list kosong
80      if (!file.exists()) return list;
81
82      try (BufferedReader br = new BufferedReader(new FileReader(file))) {
83        String line;
84        while ((line = br.readLine()) != null) {
85          String[] data = line.split("\\|");
86          String type = data[0];
87          String title = data[1];
88          double rating = Double.parseDouble(data[2]);
89          String desc = data[3];
90          String genr = data[4];
91          boolean marked = data.length > 5 && Boolean.parseBoolean(data[5]);

```

Method “load()” digunakan untuk memuat data media dari *file* secara sinkron. Pertama, *method* ini membuat *ArrayList* kosong untuk menampung data dan memeriksa apakah *file* penyimpanan ada. Jika tidak ada, *method* mengembalikan *list* kosong. Selanjutnya, setiap baris *file* dibaca menggunakan “*BufferedReader*” dan di-*split* berdasarkan “|” untuk mendapatkan atribut media, seperti tipe, judul, rating, deskripsi, genre, dan status *marked*.

```

93           //memuat objek film berdasarkan tipe
94           CinemaItem item = switch (type) {
95             case "Movie" -> new Movie(title, rating, desc, genr);
96             case "Series" -> new Series(title, rating, desc, genr);
97             default -> null;
98           };
99
100          //set status marked jika implementasi Marked
101          if (item instanceof Marked m) {
102            m.setMark(marked);
103          }
104
105          if (item != null) list.add(item); //tambahkan ke list
106        }
107      } catch (IOException e) {
108        e.printStackTrace();
109      }
110    }
111  }
112 }

```

Berdasarkan tipe media, dibuat objek movie atau series. Jika objek tersebut mengimplementasikan *interface* “Marked”, status *marked* diatur sesuai data. Objek yang berhasil dibuat, kemudian ditambahkan ke *list*. Setelah seluruh baris diproses, *list* media

kemudian dikembalikan. Kesalahan I/O akan ditangani dengan “e.printStackTrace()”.

d. **Movie.java**

```
1 package film.com.model;
2
3 /**
4 * Kelas {@code Movie} merepresentasikan
5 * film layar lebar.
6 *
7 * @author aprliwlw
8 */
9 public class Movie extends CinemaItem implements Marked{
10     /**Status apakah film sudah ditonton */
11     private boolean marked = false;
12
13     /**
14     * Konstruktor Movie.
15     *
16     * @param title judul film
17     * @param rating rating film
18     * @param description deskripsi film
19     * @param genre genre film
20     */
21     public Movie(String title, double rating, String description, String genre) {
22         //panggil constructor induk (CinemaItem)
23         super(title, rating, description, genre);
24     }
}
```

Package “[film.com](#).model” menandakan bahwa kelas ini berada di dalam folder “model”. Kelas “Movie” digunakan untuk merepresentasikan movie dan merupakan turunan dari kelas “CinemaItem” sekaligus mengimplementasikan *interface* “Marked”. Kelas ini memiliki atribut “marked” untuk menyimpan status apakah movie sudah ditonton. *Constructor* “Movie(String title, double rating, String description, String genre)” digunakan untuk membuat objek movie barus sekaligus menginisialisasi atribut melalui pemanggilan *constructor* “CinemaItem”.

```
26  /**{@inheritDoc} */
27  @Override
28  public void setMark(boolean mark) {
29      this.marked = mark; //set status marked
30  }
31  /**{@inheritDoc} */
32  @Override
33  public boolean isMarked() {
34      return marked; //ambil status marked saat ini
35  }
36
37 /**
38 * Mengembalikan tipe media.
39 *
40 * @return "Movie"
41 */
42 @Override
43 public String getType() {
44     return "Movie";
45 }
46 }
```

Method “setMark(boolean mark)” digunakan untuk mengubah status movie apakah sudah ditonton, sedangkan method “isMarked()” digunakan untuk mengembalikan status penandaan saat ini. Selain itu, method “getType()” mengembalikan tipe media berupa “Movie” sesuai dengan kelas ini.

e. **Series.java**

```

1 package film.com.model;
2
3 /**
4 * Kelas {@code Series} merepresentasikan
5 * serial televisi atau web series.
6 *
7 * @author aprliwln
8 */
9 public class Series extends CinemaItem implements Marked{
10     /**Status apakah series sudah ditonton */
11     private boolean marked = false;
12
13     /**
14     * Konstruktor Series.
15     *
16     * @param title judul series
17     * @param rating rating series
18     * @param description deskripsi series
19     * @param genre genre series
20     */
21     public Series(String title, double rating, String description, String genre) {
22         //panggil constructor induk (CinemaItem)
23         super(title, rating, description, genre);
24     }

```

Package “[film.com](#).model” menandakan bahwa kelas ini berada di dalam folder “model”. Kelas “Series” digunakan untuk merepresentasikan series dan merupakan turunan dari kelas “CinemaItem” sekaligus mengimplementasikan *interface* “Marked”. Kelas ini memiliki atribut “marked” untuk menyimpan status apakah series sudah ditonton. *Constructor* “Series(String title, double rating, String description, String genre)” digunakan untuk membuat objek series barus sekaligus menginisialisasi atribut melalui pemanggilan *constructor* “CinemaItem”.

```

26  /**
27  * @Override
28  public void setMark(boolean mark) {
29     this.marked = mark; //set status marked
30 }
31 /**
32 * @Override
33 public boolean isMarked() {
34     return marked; //ambil status marked saat ini
35 }
36
37 /**
38 * Mengembalikan tipe media.
39 *
40 * @return "Series"
41 */
42 @Override
43 public String getType() {
44     return "Series";
45 }
46 }

```

Method “setMark(boolean mark)” digunakan untuk mengubah status series apakah sudah ditonton, sedangkan *method “isMarked()”* digunakan untuk mengembalikan status penandaan saat ini. Selain itu, *method “getType()”* mengembalikan tipe media berupa “Series” sesuai dengan kelas ini.

3. File Lain

a. MediaTest.java

```

1 package film.com.model;
2
3 import static org.junit.jupiter.api.Assertions.*; //utk JUnit
4 import org.junit.jupiter.api.Test;
5
6 /**
7 * Kelas {@code MediaTest} berisi
8 * pengujian unit untuk kelas model.
9 *
10 * @author aprliwln
11 */

```

Package “film.com.model” menandakan bahwa kelas ini berada di dalam folder “model”. Beberapa *import* yang digunakan, yaitu “org.junit.jupiter.api.Test” untuk menandai *method* sebagai *test case* dalam JUnit 5, serta “static org.junit.jupiter.api.Assertions.*” yang

menyediakan berbagai *method assertion*, seperti “`assertEquals()`” untuk memeriksa kebenaran hasil pengujian.

```
12 public class MediaTest {  
13  
14     /**  
15      * Menguji tipe dan genre Movie.  
16      */  
17     @Test  
18     void testMovieTypeAndGenre() {  
19         CinemaItem movie = new Movie(title: "Interstellar", rating: 4.5,  
20             | | | | description: "Tentang Blackholes", genre: "Sci-Fi");  
21             //memastikan type movie benar  
22             assertEquals(expected: "Movie", movie.getType());  
23             //memastikan genre movie benar  
24             assertEquals(expected: "Sci-Fi", movie.getGenre());  
25     }  
26 }
```

Kelas “MediaTest” digunakan untuk melakukan pengujian unit pada kelas media menggunakan JUnit. *Method* “`testMovieTypeAndGenre()`” diberi anotasi “`@Test`” dan digunakan untuk menguji tipe dan genre dari objek “Movie”. Di dalam *method* ini, dibuat objek “Movie” baru dengan judul, rating, deskripsi, dan genre yang telah ditentukan, kemudian diuji menggunakan *assertion* “`assertEquals()`” untuk memastikan bahwa “`getType()`” mengembalikan “Movie” dan “`getGenre()`” mengembalikan genre yang sesuai.

```
27     /**  
28      * Menguji tipe dan genre Series.  
29      */  
30     @Test  
31     void testSeriesType() {  
32         CinemaItem series = new Series(title: "Stranger Things", rating: 4.8,  
33             | | | | description: "Misteri di Hawkins", genre: "Thriller");  
34             //memastikan type series benar  
35             assertEquals(expected: "Series", series.getType());  
36             //memastikan genre series benar  
37             assertEquals(expected: "Thriller", series.getGenre());  
38     }  
39 }
```

Method “`testSeriesType()`” diberi anotasi “`@Test`” dan digunakan untuk menguji tipe dan genre dari objek “Series”. Di dalam *method* ini, dibuat objek “Series” baru dengan judul, rating, deskripsi, dan genre yang telah ditentukan, kemudian diuji menggunakan *assertion* “`assertEquals()`” untuk memastikan bahwa “`getType()`” mengembalikan “Series” dan “`getGenre()`” mengembalikan genre yang sesuai.

b. module-info.java

```
1 module film.com {  
2     //utk komponen UI  
3     requires javafx.controls;  
4     //utk memuat file FXML  
5     requires javafx.fxml;  
6     //utk rendering grafis JavaFX  
7     requires transitive javafx.graphics;  
8  
9     //folder utk FXMLLoader dpt mengakses controller  
10    opens film.com.controller to javafx.fxml;  
11    //folder yg dibuka ke JavaFX base dan FXML  
12    opens film.com.model to javafx.base, javafx.fxml;  
13    //folder utama  
14    exports film.com;  
15 }
```

File “module-info.java” mendefinisikan modul “[film.com](#)” pada proyek JavaFX Cinema Diary. Beberapa *requires* digunakan untuk menyertakan dependensi, antara lain “javafx.controls” untuk komponen UI, “javafx.fxml” untuk memuat *file* FXML, serta “javafx.graphics” untuk *rendering* grafis JavaFX. Pernyataan “opens [film.com](#).controller to javafx.fxml” memungkinkan folder “controller” diakses oleh “FXMLLoader”, sedangkan “opens [film.com](#).model to javafx.base, javafx.fxml” membuka folder “model” ke modul JavaFX *base* dan FXML agar dapat digunakan di UI. Terakhir, “exports [film.com](#)” menandakan bahwa *package* utama “[film.com](#)” dapat diakses dari modul lain.

c. Responsi2_PPBO_L0124040.java

```
1 package film.com;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Scene;
6 import javafx.stage.Stage;
7
8 /**
9 * Kelas {@code Responsi2_PPBO_L0124040_SC} merupakan
10 * kelas utama (entry point) aplikasi Cinema Diary.
11 * <p>
12 * Kelas ini mewarisi {@link Application} dari JavaFX
13 * dan bertugas untuk:
14 * </p>
15 * <ul>
16 *   <li>Menjalankan lifecycle JavaFX</li>
17 *   <li>Memuat tampilan awal (login)</li>
18 *   <li>Menampilkan window utama aplikasi</li>
19 * </ul>
20 *
21 * @author aprliwln
22 */
```

Package “[film.com](#)” menandakan bahwa kelas ini berada di dalam folder “[film.com](#)”. Beberapa *import* yang digunakan, yaitu “javafx.application.Application” sebagai kelas dasar untuk membuat aplikasi JavaFX, “javafx.fxml.FXMLLoader” untuk memuat file FXML, “javafx.scene.Scene” untuk mengatur dan menampilkan konter antarmuka, serta “javafx.stage.Stage” untuk mengelola jendela (*window*) utama aplikasi.

```
23 public class Responsi2_PPBO_L0124040_SC extends Application {  
24     /**  
25      * Method {@code start} dipanggil secara otomatis  
26      * oleh JavaFX saat aplikasi dijalankan.  
27      * <p>  
28      * Method ini memuat file {@code login.fxml}  
29      * sebagai tampilan awal aplikasi.  
30      * </p>  
31      *  
32      * @param stage stage utama aplikasi  
33      * @throws Exception jika terjadi kesalahan  
34      *         saat memuat file FXML  
35      */  
36     @Override  
37     public void start(Stage stage) throws Exception {  
38         //memuat layout login dari file FXML  
39         FXMLLoader loader = new FXMLLoader(  
40             getClass().getResource("/film/com/view/login.fxml")  
41         );  
42         //set scene menggunakan layout yg dimuat  
43         stage.setScene(new Scene(loader.load()));  
44         //set judul aplikasi  
45         stage.setTitle("Login");  
46         //menampilkan jendela utama  
47         stage.show();  
48     }  
49 }
```

Kelas “Responsi2_PPBO_L0124040_SC” merupakan kelas utama aplikasi Cinema Diary yang mewarisi “Application” dari JavaFX. Method “start(Stage stage)” dipanggil secara otomatis oleh JavaFX saat aplikasi dijalankan, serta memuat tampilan awal dari file “login.fxml”, membuat Scene dari layout tersebut, mengatur judul jendela menjadi “Login”, dan menampilkan jendela utama dengan “stage.show()”.

```
51  /**
52  * Method {@code main} berfungsi sebagai
53  * titik awal eksekusi aplikasi Java.
54  * <p>
55  * Method ini akan memanggil lifecycle JavaFX
56  * melalui method {@code Application.launch()}.
57  * </p>
58  *
59  * @param args argumen command-line
60  */
Run | Debug
61  public static void main(String[] args) {
62  |   launch(); //memanggil lifecycle javafx
63  }
64 }
```

Selain itu, *method* “main(String[] args)” merupakan titik awal eksekusi aplikasi dengan memanggil *lifecycle* JavaFX melalui “launch()” sehingga seluruh proses akan inisialisasi.

4. GUI

a. login.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <?import javafx.scene.control.*?>
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.text.Font?>
5  <?import java.net.URL?>
6
7  <!--root layout-->
8  <StackPane xmlns="http://javafx.com/javafx/8"
9   |   |   xmlns:fx="http://javafx.com/fxml/1"
10  |   |   fx:controller="film.com.controller.LoginController"
11  |   |   styleClass="login-bg" <!--utk background-->
12
13  <!--menghubungkan stylesheet CSS-->
14  <stylesheets>
15  |   <URL value="@style.css" />
16  </stylesheets>
17
18  <!--utk menata elemen vertikal di tengah layar-->
19  <VBox alignment="CENTER" spacing="15">
20  |   <!--judul utama aplikasi-->
21  |   <Label text="♦ CINEMA DIARY ♦" textFill="WHITE">
22  |       <font>
23  |           <Font name="Georgia Bold" size="48"/>
24  |       </font>
25  |       <style>
26  |           -fx-effect: dropshadow(three-pass-box, □rgba(0,0,0,0.2), 10, 0, 0, 5);
27  |       </style>
28  |   </Label>

```

Kode di atas digunakan untuk mendefinsikan tampilan halaman *login*. *Root layout* menggunakan *StackPane* dengan “fx:controller=“[film.com](#).controller.LoginController””, sehingga setiap iterasi di UI akan dihubungkan dengan controller “LoginController”. *Layout* ini diberi kelas CSS “login-bg” untuk pengaturan *background* dan *stylesheet* “style.css” dihubungkan melalui elemen “*<stylesheets>*”. Di dalam *StackPane*, terdapat *VBox* yang menata elemen secara vertikal di tengah layar dengan jarak antar elemen 15 *pixel*. Label digunakan untuk menampilkan judul aplikasi dengan warna teks putih menggunakan *font* “Goergia Bold” ukuran 48, serta efek bayangan (*drop shadow*).

```

30      <!--tagline aplikasi-->
31      <Label text="Writing down every scene that touched your heart"
32          |   textFill="#FEFAE0">
33          |   <font>
34          |       <Font name="Georgia italic" size="16"/>
35          |   </font>
36          |   <style>
37          |       -fx-effect: dropshadow(gaussian, #rgba(0,0,0,0.1), 5, 0, 0, 1);
38          |   </style>
39      </Label>
40
41      <Region prefHeight="40"/>
42
43      <!--tombol utk masuk ke halaman utama-->
44      <Button text="Enter Journal"
45          |   onAction="#login"
46          |   styleClass="btn-start"
47          |   style="-fx-padding: 12 40; -fx-font-size: 15; -fx-cursor: hand;"/>
48      </VBox>
49  </StackPane>

```

Di bawahnya, terdapat Label *tagline* dengan *font* “Georgia Italic” ukuran 16 berwarna *cream* dan efek *drop shadow*. Selanjutnya, terdapat *Region* dengan tinggi 40 *pixel* sebagai spasi, lalu tombol *Button* yang menggunakan kelas CSS “*btn-start*” memiliki *padding* dan ukuran *font* khusus, serta *cursor* berubah menjadi tangan saat diarahkan. Tombol ini dihubungkan dengan *method* “*login()*” di *controller* untuk masuk ke halaman utama aplikasi.

b. main.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <?import javafx.scene.control.*?>
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.text.Font?>
5  <?import java.net.URL?>
6
7  <!--root layout utk menata elemen vertikal-->
8  <VBox xmlns="http://javafx.com/javafx/8"
9   |   xmlns:fx="http://javafx.com/fxml/1"
10  |   fx:controller="film.com.controller.MainController"
11  |   spacing="20"
12  |   styleClass="main-bg"
13  |   style="-fx-padding: 40;">
14
15  <!--menghubungkan stylesheet CSS-->
16  <stylesheets>
17  |   <URL value="@style.css" />
18  </stylesheets>
19
20  <!--judul aplikasi-->
21  <HBox alignment="CENTER">
22  |   <Label text="Cinephile Scribbles" textFill="#606C38">
23  |   |   <font>
24  |   |   |   <Font name="Georgia Bold" size="32"/>
25  |   |   </font>
26  |   </Label>
27  </HBox>

```

Kode di atas digunakan untuk mendefinsikan tampilan halaman utama aplikasi Cinema Diary. *Root layout* menggunakan *VBox* dengan “fx:controller=“[film.com](#).controller.MainController””, sehingga setiap iterasi di UI akan dihubungkan dengan *controller* “MainController”. *Layout* ini memiliki jarak antar elemen 20 *pixel*, *padding* 40 *pixel*, dan diberi kelas CSS “main-bg”. *Stylesheet* “style.css” dihubungkan melalui “<stylesheets>”. *HBox* digunakan untuk menampilkan judul aplikasi secara horizontal di tengah. Judul ditampilkan menggunakan Label dengan warna teks “#606C38”, font “Georgia Bold” ukuran 32.

```

29   <!--kontainer utama, horizontal-->
30   <HBox spacing="25" VBox.vgrow="ALWAYS">
31
32     <!--bagian form utk menambahakan film-->
33     <VBox spacing="12" prefWidth="350" styleClass="card" style="-fx-padding: 25;">
34       <Label text="Write a Review" textFill="#A98467"
35         style="-fx-font-weight: bold; -fx-font-size: 16;"/>
36
37       <TextField fx:id="titleField" promptText="Title"/>
38       <ComboBox fx:id="typeBox" promptText="Select Type" maxWidth="Infinity"/>
39       <ComboBox fx:id="genreBox" promptText="Select Genre" maxWidth="Infinity"/>
40       <TextField fx:id="ratingField" promptText="Rating (e.g. 4.5)"/>
41       <TextArea fx:id="descArea" promptText="Short description" prefRowCount="3" wrapText="true"/>
42
43     <!--tombol add & delete-->
44     <HBox spacing="10">
45       <Button text="+ Add" onAction="#addCinema" styleClass="btn-primary" HBox.hgrow="ALWAYS"
46         maxWidth="Infinity"/>
47       <Button text="Delete" onAction="#deleteCinema" style="-fx-background-color: #E3D5CA;
48         -fx-text-fill: #A98467; -fx-background-radius: 15; -fx-font-weight: bold;"/>
49     </HBox>
50
51     <Separator style="-fx-padding: 5 0;"/>
52
53     <!--search bar-->
54     <TextField fx:id="searchField" promptText="Search title..."/>
55
56     <!--tombol filter film-->
57     <HBox spacing="8" alignment="CENTER">
58       <Button text="All" onAction="#showAll" style="-fx-background-radius: 10;"/>
59       <Button text="Movie" onAction="#filterMovie" style="-fx-background-radius: 10;"/>
60       <Button text="Series" onAction="#filterSeries" style="-fx-background-radius: 10;"/>
61     </HBox>
62   </VBox>

```

Bagian di atas mendefinisikan kontainer utama halaman utama menggunakan *HBox* dengan jarak antar elemen 25 *pixel* dan tinggi mengikuti *VBox.vgrow= "ALWAYS"*. *VBox* berfungsi sebagai *form* untuk menambahkan movie atau series baru dengan jarak antar elemen 12 *pixel*, lebar 350 *pixel*, *padding* 25 *pixel*, dan diberi kelas CSS “card”. *Form* ini berisi *Label* untuk judul, “*TextField*” untuk judul dan rating pada masukan, “*ComboBox*” untuk tipe dan genre, serta “*TextArea*” untuk deskripsi singkat.

Lalu, terdapat *HBox* berisi tombol “+ Add” yang terhubung ke *method* “*addCinema()*” dan tombol “Delete” yang memanggil “*deleteCinema()*” yang diatur dengan lebar tombol menyesuaikan. Setelah itu, terdapat *Separator* sebagai pemisah visual, “*TextField*” untuk pencarian judul, dan *HBox* berisi tombol filter “All”, “Movie”, dan “Series” yang memanggil *method* “*showAll()*”, “*filterMovie()*”, dan “*filterSeries()*” di *controller*.

```

64      <!--bagian tabel utk menampilkan film yg ditambahkan-->
65      <VBox spacing="15" HBox.hgrow="ALWAYS" styleClass="card" style="-fx-padding: 25;">
66
67          <!--header tabel dan tombol completed-->
68          <HBox alignment="CENTER_LEFT">
69              <Label text="My Watch Journey" textFill="#A98467" style="-fx-font-weight: bold;
70                  -fx-font-size: 18;" HBox.hgrow="ALWAYS" maxWidth="Infinity"/>
71              <Button text="Completed" onAction="#marked" styleClass="btn-primary"/>
72          </HBox>
73
74          <!--table view utk menampilkan dftr film-->
75          <TableView fx:id="tableView" VBox.vgrow="ALWAYS">
76              <columns>
77                  <TableColumn fx:id="titleCol" text="Title" prefWidth="180"/>
78                  <TableColumn fx:id="typeCol" text="Type" prefWidth="100"/>
79                  <TableColumn fx:id="genreCol" text="Genre" prefWidth="120"/>
80                  <TableColumn fx:id="ratingCol" text="Rating" prefWidth="70"/>
81                  <TableColumn fx:id="markedCol" text="Status" prefWidth="100"/>
82                  <TableColumn fx:id="descCol" text="Description" minWidth="200" prefWidth="300"/>
83              </columns>
84
85              <!--resize kolom-->
86              <columnResizePolicy>
87                  <TableView fx:constant="UNCONSTRAINED_RESIZE_POLICY" />
88              </columnResizePolicy>
89          </TableView>
90      </VBox>
91  </HBox>
92 </VBox>

```

Bagian di atas mendefinisikan kontainer *VBox* di sisi kanan halaman utama untuk menampilkan daftar film yang telah ditambahkan. *VBox* ini memiliki jarak antar elemen 15 *pixel*, *padding* 25 *pixel*, kelas CSS “card”, dan lebar mengikuti sisa ruang. *HBox* yang berisi Label judul dengan warna teks “#A98467” dan *font* tebal ukuran 18, serta tombol “Completed” yang terhubung ke *method* “marked()” untuk menandai film yang sudah ditonton. Di bawahnya terdapat *TableView* untuk menampilkan daftar film dengan kolom “titleCol”, “typeCol”, “genreCol”m “ratingCol”, “markedCol”, dan “descCol”. Lebar setiap kolom diatur sesuai kebutuhan, sementara “descCol” memiliki lebar minimal 200 *pixel* dan preferensi 300 *pixel*. *TableView* menggunakan kebijakan *resize* “UNCONSTRAINED_RESIZE_POLICY” agar kolom dapat menyesuaikan ukuran secara fleksibel.

c. style.css

```

1  /*global style*/
2  .root {
3      -fx-font-family: "Segoe UI", sans-serif;
4      -fx-base: #F5EBE0;
5  }
6
7  /*backgrounds*/
8  .login-bg {
9      -fx-background-color: linear-gradient(to bottom right, #D4A373, #A98467);
10 }
11 .main-bg {
12     -fx-background-color: #F5EBE0;
13 }
14
15 /*cards*/
16 .card {
17     -fx-background-color: #FEFAE0;
18     -fx-background-radius: 20;
19     -fx-effect: dropshadow(three-pass-box, rgba(142, 151, 117, 0.2), 15, 0, 0, 5);
20 }

```

Selector “.root” mengatur *font* utama aplikasi serta warna dasar latar belakang. Kelas “.login-bg” dan “.main-bg” digunakan untuk mengatur *background* halaman *login* dan halaman utama. Selain itu, kelas “.card” digunakan untuk memberi gaya pada kontainer berupa latar belakang, sudut membulat, dan efek bayangan.

```

22 /*input fields*/
23 .text-field, .combo-box, .text-area {
24     -fx-background-color: #FAEDCD;
25     -fx-background-radius: 10;
26     -fx-border-color: transparent;
27     -fx-text-fill: #4A4A4A;
28 }
29 .text-field:focused {
30     -fx-border-color: #D4A373;
31     -fx-background-color: #ffffff;
32 }
33
34 /*buttons*/
35 .btn-primary {
36     -fx-background-color: #8E9775;
37     -fx-text-fill: white;
38     -fx-background-radius: 15;
39     -fx-font-weight: bold;
40     -fx-cursor: hand;
41 }
42 .btn-primary:hover {
43     -fx-background-color: #7A8463;
44 }
45
46 .btn-start {
47     -fx-background-color: #FEFAE0;
48     -fx-text-fill: #A98467;
49     -fx-background-radius: 30;
50     -fx-font-weight: bold;
51 }

```

Bagian di atas digunakan untuk mengatur tampilan *input field* dan tombol pada aplikasi Cinema Diary. Selector “.text-field”, “.combo-box”, dan “.text-area” mengatur warna latar, sudut membulat, serta warna teks dengan tampilan berbeda saat *field* sedang aktif. Selain itu, kelas “.btn-primary” digunakan untuk tombol utama dengan warna hijau, teks putih, sudut membulat, dan efek hover. Sementara itu, kelas “.btn-start” digunakan untuk tombol awal pada halaman *login* dengan warna latar terang, teks cokelat, dan tampilan tebal.

```

53 /*table*/
54 .table-view {
55     -fx-background-color: transparent;
56     -fx-background-radius: 15;
57     -fx-border-color: #E3D5CA;
58     -fx-border-radius: 15;
59     -fx-padding: 5;
60 }
61
62 /*judul tabel*/
63 .table-view .column-header-background {
64     -fx-background-color: #E3D5CA;
65     -fx-background-radius: 15 15 0 0;
66 }
67 .table-view .column-header {
68     -fx-background-color: transparent;
69     -fx-size: 40;
70 }
71 .table-view .column-header .label {
72     -fx-text-fill: #606C38;
73     -fx-font-weight: bold;
74     -fx-alignment: CENTER;
75 }
76
77 /*sel tabel*/
78 .table-view .table-cell {
79     -fx-border-color: transparent;
80     -fx-padding: 10;
81     -fx-text-fill: #4A4A4A;
82     -fx-font-size: 13px;
83 }

```

Bagian di atas digunakan untuk mengatur tampilan *TableView* pada aplikasi Cinema Diary. Style “.table-view” mengatur latar transparan, sudut membulat, *border*, dan *padding* tabel. Bagian *header* tabel diatur agar memiliki latar warna khusus, sudut membulat di bagian atas, serta teks judul kolom yang rata tengah, tebal, dan berwarna hijau. Selain itu, *style* pada “.table-cell” digunakan untuk mengatur *padding*, warna teks, dan ukuran *font* pada tabel.

```

85  /*area sisa scroll bar*/
86  .table-view .column-header-background .filler {
87  |   -fx-background-color: ■#E3D5CA;
88  |   -fx-background-radius: 0 15 0 0;
89  }
90  /*baris tabel*/
91  .table-row-cell {
92  |   -fx-background-color: ■#FEFAE0;
93  |   -fx-border-color: ■#F5EBE0;
94  |   -fx-border-width: 0 0 1 0;
95  }
96  /*baris ganjil berbeda warna*/
97  .table-row-cell:odd {
98  |   -fx-background-color: ■#FAEDCD;
99  }
100 /*baris yg dipilih*/
101 .table-row-cell:selected {
102 |   -fx-background-color: ■#D4A373;
103 }
104 .table-row-cell:selected .table-cell {
105 |   -fx-text-fill: ■white;
106 }
107 /*hapus garis vertikal*/
108 .table-view .virtual-flow .scroll-bar:vertical,
109 .table-view .virtual-flow .scroll-bar:horizontal {
110 |   -fx-background-color: transparent;
111 }

```

Bagian di atas digunakan untuk menyempurnakan tampilan *TableView*. *Style* pada “.filler” mengatur area sisa *header* tabel agar warnanya selaras dengan judul kolom. Baris tabel diatur memiliki warna latar tertentu dengan perbedaan warna untuk baris ganjil. Saat baris dipilih, latar berubah warna dan teks menjadi putih untuk menonjolkan seleksi. Selain itu, *scrollbar* vertikal dan horizontal dibuat transparan.

```

113  /*scroll bar*/
114  .table-view .scroll-bar:horizontal,
115  .table-view .scroll-bar:vertical {
116      -fx-background-color: transparent;
117      -fx-padding: 2;
118  }
119
120  .table-view .scroll-bar .thumb {
121      -fx-background-color: #A98467;
122      -fx-background-radius: 10;
123      -fx-opacity: 0.5;
124  }
125
126  .table-view .scroll-bar .thumb:hover {
127      -fx-opacity: 1.0;
128      -fx-background-color: #8E9775;
129  }
130  /*tombol <> di scroll bar*/
131  .table-view .scroll-bar .increment-button,
132  .table-view .scroll-bar .decrement-button {
133      -fx-background-color: transparent;
134      -fx-shape: "";
135  }
136
137  .table-view .scroll-bar .track {
138      -fx-background-color: #F5EBE0;
139      -fx-background-radius: 10;
140      -fx-border-color: #E3D5CA;
141      -fx-border-radius: 10;
142  }

```

Bagian di atas digunakan untuk mengatur tampilan *scroll bar* pada *TableView*. *Scroll bar* horizontal dan vertikal dibuat transparan. Bagian *thumb* diatur dengan warna coklat, sudut membulat, dan tingkat transparansi tertentu, lalu berubah warna dan menjadi lebih jelas saat diarahkan kursor. Tombol *increment* dan *decrement* pada *scroll bar* disembunyikan dengan membuat latarnya transparan, sementara bagian *track* diberi warna latar, *border*, dan sudut membulat.

```

144 /*kotak tooltip saat kursor diarahkan ke teks deskripsi*/
145 .tooltip {
146     -fx-background-color: #FEFAE0;
147     -fx-text-fill: #4A4A4A;
148     -fx-border-color: #D4A373;
149     -fx-border-radius: 8;
150     -fx-background-radius: 8;
151     -fx-font-size: 13px;
152     -fx-padding: 10px;
153     -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.1), 10, 0, 0, 5);
154     -fx-pref-width: 250;
155 }
156
157 /*wrap textnya jika panjang*/
158 .tooltip > .label {
159     -fx-text-alignment: left;
160     -fx-wrap-text: true;
161 }

```

Bagian di atas digunakan untuk mengatur tampilan *tooltip* yang muncul saat kursor diarahkan ke teks deskripsi. *Tooltip* diberi latar warna terang, warna teks gelap, *border*, dan sudut membulat, serta efek bayangan. Selain itu, teks di dalam *tooltip* diatur agar *wrap text* dan rata kiri, sehingga deskripsi yang panjang tetap dapat ditampilkan.

Tabel penggunaan materi:

NO	MATERI	PERAN
1.	Enkapsulasi	Enkapsulasi digunakan untuk melindungi data agar tidak bisa diakses langsung dari luar kelas. Pada program ini, atribut seperti “title”, “rating”, “description”, “genre”, dan “marked” pada kelas “CinemaItem”, “Movie”, dan “Series” dibuat dengan akses <i>protected</i> dan <i>private</i> . Akses terhadap data tersebut dilakukan melalui <i>getter</i> dan <i>setter</i> .
2.	Inheritance	<i>Inheritance</i> digunakan untuk mewariskan atribut dan <i>method</i> dari kelas induk ke kelas turunan. Pada program ini, kelas “Movie” dan “Series” merupakan turunan dari kelas “CinemaItem”.
3.	Polimorfisme	<i>Polimorfisme</i> digunakan agar objek turunan dapat diperlakukan sebagai objek kelas induk. Pada program ini, objek “Movie” dan “Series” disimpan dalam satu <i>list</i> bertipe “List<CinemaItem>”. Selain itu, penggunaan “instanceof Marked” memungkinkan program memeriksa apakah suatu objek memiliki kemampuan untuk ditandai meskipun direferensikan sebagai “CinemaItem”.
4.	Abstract Class	<i>Abstract class</i> digunakan sebagai kerangka dasar yang tidak dapat diinstansiasi langsung. Pada program ini, “CinemaItem” berperan sebagai <i>abstract class</i> yang mendefinisikan atribut

		dan <i>method</i> umum untuk semua jenis film. Kelas ini memastikan bahwa setiap turunan (“Movie” dan “Series”) memiliki struktur data dan perilaku dasar yang sama.
5.	Interface	<i>Interface</i> digunakan untuk menyediakan kemampuan tambahan tanpa terikat pada pewarisan kelas. Pada program ini, <i>interface</i> “Marked” digunakan untuk menandai media yang sudah ditonton (status <i>done/undone</i>) dengan mendefinisikan <i>method</i> “setMark(boolean)” dan “isMarked()” yang kemudian diimplementasikan oleh kelas “Movie” dan “Series”.
6.	GUI-Programming	GUI <i>Programming</i> digunakan untuk membangun antarmuka grafis menggunakan JavaFX. Pada program ini, komponen seperti “TextField”, “ComboBox”, “TextArea”, “TableView”, dan “ TableColumn” digunakan untuk menerima <i>input user</i> dan menampilkan data. <i>Class</i> “LoginController” dan “MainController” berfungsi sebagai penghubung antara tampilan (FXML) dan logika program.
7.	Documentation (Javadoc)	Javadoc digunakan untuk memberikan dokumentasi pada <i>class</i> dan <i>method</i> agar mudah dipahami oleh pengembang lain. Pada program ini, Javadoc digunakan untuk mendokumentasi <i>class</i> , <i>method</i> , dan parameter agar memudahkan pemahaman serta pemeliharaan kode.
8.	Testing	<i>Testing</i> digunakan untuk memastikan bahwa <i>method</i> dan <i>class</i> berjalan sesuai dengan yang diharapkan. Pada program ini, pengujian dilakukan menggunakan JUnit 5 melalui <i>class</i> “MediaTest” dengan cara memverifikasi tipe media dan genre pada objek “Movie” dan “Series” menggunakan <i>assertion</i> “assertEquals()”
9.	Multithreading	<i>Multithreading</i> digunakan untuk menjalankan proses secara <i>asynchronous</i> agar GUI tidak <i>freeze</i> . Pada program ini, penyimpanan data menggunakan “saveAsync()” dan “loadAsync()” dijalankan di <i>thread</i> terpisah agar proses penyimpanan <i>file</i> tidak mengganggu responsivitas antarmuka <i>user</i> .