# LAPORAN
# PEMROGRAMAN BERORIENTASI OBJEK
# PERTEMUAN 14


Dosen Pengampu:

Bayu Adhi Nugroho, Ph.D
197905182014031001

Disusun oleh:

Riyana Isti Juwariyah 09010624014

**PROGRAM STUDI SISTEM INFORMASI**
**FAKULTAS SAINS DAN TEKNOLOGI**
**UNIVERSITAS ISLAM NEGERI SUNAN AMPEL**
**SURABAYA**
**2025**

# IMPLEMENTASI CRUD DENGAN UPLOAD DAN DOWNLOAD CSV BERBASIS PBO

A. Tujuan Praktikum

1. Mahasiswa mampu menerapkan konsep CRUD (Create, Read, Update, Delete) dalam aplikasi berbasis Pemrograman Berorientasi Objek (PBO).
2. Mahasiswa mampu mengimplementasikan fitur upload file CSV untuk membaca dan mengimpor data ke dalam aplikasi.
3. Mahasiswa mampu mengimplementasikan fitur download file CSV sebagai proses ekspor data dari aplikasi.
4. Mahasiswa mampu melatih penggunaan class, objek, dan enkapsulasi dalam pengelolaan data dan file berbasis PBO.
5. Mahasiswa mampu mengembangkan kemampuan menangani input/output file serta melakukan validasi data pada proses CSV

B. Petunjuk Praktikum

1. Membuat project baru di IDE Java (NetBeans/Eclipse/IntelliJ) dan menyiapkan struktur dasar program berbasis PBO (class, package, dan file utama).
2. Membuat struktur package seperti model, controller/dao, utils, dan view (opsional jika menggunakan GUI).
3. Membuat class Model (Entity) untuk merepresentasikan satu baris data yang akan disimpan ke dalam list.
4. Membuat class controller yang berisi fungsi CRUD.
5. Menambahkan data awal secara manual (opsional) pada list untuk kebutuhan pengujian fitur CRUD sebelum fitur CSV diimplementasikan.
6. Membuat fungsi Upload CSV menggunakan JFileChooser (memilih file CSV), BufferedReader (membaca isi file), split(",") (memisahkan kolom data dan menyimpannya ke ArrayList).
7. Membuat fitur Download CSV yang mengekspor seluruh list ke file .csv menggunakan FileWriter dengan format kolom yang rapi dan sesuai.
8. Membuat tampilan GUI.

9. Mengimplementasikan dan menguji alur CRUD serta Upload/Download CSV, memastikan data tersimpan, ditampilkan, diimpor, dan diekspor dengan benar tanpa eror.

## C. DASAR TEORI

### 1. File CSV (Comma-Separated Values)

CSV adalah format file teks sederhana yang digunakan untuk menyimpan data tabular (seperti spreadsheet) di mana setiap baris mewakili sebuah record, dan setiap record terdiri dari satu atau lebih field yang dipisahkan oleh koma (,) atau delimiter lainnya (seperti titik koma ;) tergantung pengaturan lokal.

Kelebihan Format CSV :

- Strukturnya mudah dipahami, dan file berukuran kecil karena hanya berisi data teks polos.
- Hampir semua aplikasi pengolah data (seperti Microsoft Excel, Google Sheets, database management systems, dan bahasa pemrograman) memiliki dukungan native atau library untuk membaca dan menulis file CSV.
- Tidak seperti file biner, data dalam CSV dapat dibuka dan diperiksa dengan text editor sederhana.
- Merupakan format *de facto* untuk proses impor/ekspor data antar sistem yang berbeda.

### 2. Upload CSV

Upload CSV adalah proses membaca data dari sebuah file CSV eksternal dan memasukkannya ke dalam aplikasi (biasanya dengan operasi Create). Aplikasi akan membaca file baris per baris, mem-parsing setiap baris menjadi atribut-atribut objek, dan kemudian menyimpannya ke dalam memori (koleksi) atau langsung ke database.

Alur kerja Upload CSV :

- Pengguna memilih file CSV sumber yang akan diunggah.
- Aplikasi membuka file CSV menggunakan kelas seperti File Reader dan BufferedReader.
- File dibaca baris demi baris.
- Setiap baris yang dibaca (kecuali header, jika ada) di-*split* atau dipisahkan berdasarkan delimiter yang ditentukan (misalnya koma) menjadi sebuah array of string.

- Sebelum dibuat menjadi objek, data yang telah diparsing harus divalidasi. Ini mencegah error dan menjaga integritas data.
- Array of string yang telah divalidasi digunakan untuk menginstansiasi objek baru. Kemudian objek baru ditambahkan ke dalam koleksi (ArrayList).
- Setelah proses impor selesai, seringkali diperlukan untuk menulis ulang seluruh koleksi (yang sudah diperbarui) ke file CSV utama aplikasi untuk mempertahankan persistensi data.

3. Download CSV

   Download CSV adalah proses menulis data dari dalam aplikasi (dari koleksi objek) ke sebuah file CSV eksternal (biasanya dengan operasi Read). Aplikasi akan mengambil semua objek dari koleksi, mengonversi setiap objek menjadi sebuah baris string yang formatnya sesuai dengan CSV, dan menulisnya ke file.

   Alur kerja Download CSV:

   - Aplikasi membuka atau membuat file tujuan (misal: Data_Film.csv) menggunakan kelas seperti Filewritter dan BufferedWritter untuk efisiensi.
   - Aplikasi melakukan iterasi (perulangan) melalui setiap objek dalam koleksi.
   - String yang telah diformat untuk setiap objek ditulis ke dalam file sebagai baris baru.
   - Setelah semua objek berhasil ditulis, file ditutup untuk memastikan semua data telah disimpan dengan benar.

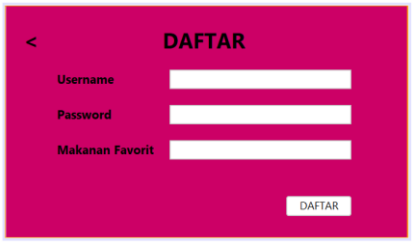
   Manfaat Download :

   - Backup data.
   - Menghasilkan file yang siap dicetak atau dianalisis lebih lanjut di aplikasi lain seprti Excel.

     Mengirim data ke sistem atau pihak lain yang membutuhkan.

D. PRAKTIKUM

1. Pada project PBOPertemuan13 rename menjadi PBOPertemuan14.
2. Package Akun
    a. Pada class Daftar tambahkan label dan textField untuk pertanyaan.




Kode ini berfungsi untuk proses pendaftaran akun baru. Sistem memeriksa apakah semua TextField termasuk TextField pertanyaan keamanan telah terisi. Pertanyaan keamanan (TfFav) digunakan sebagai data tambahan untuk verifikasi jika suatu saat pengguna perlu ubah password akun. Setelah validasi, sistem mengecek apakah username sudah ada di database. Jika belum, data username, password, dan pertanyaan keamanan disimpan, lalu pengguna dialihkan ke halaman login. Jika username sudah terdaftar, pendaftaran dibatalkan dan pengguna diminta mengganti username.

```
private void BtnDaftarActionPerformed(java.awt.event.ActionEvent evt) {
    if (TfUsnDaftar.getText().equals("") || TfPassDaftar.getText().equals("") || TfFav.getText().equals("")) {
        JOptionPane.showMessageDialog(null, "Isi semua data");
    } else {
        String user, pw, tanya;
        user = TfUsnDaftar.getText();
        pw = TfPassDaftar.getText();
        tanya = TfFav.getText();

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("PBO_Pertemuan12PU");
        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin();

        Login l = em.find(Login.class, user);
        if (l != null) {
            JOptionPane.showMessageDialog(null, "Username sudah ada, gunakan username lain");
            bersih();
            TfUsnDaftar.requestFocus();
        } else {
            Login o = new Login();
            o.setUsername(user);
            o.setPassword(pw);
            o.setPertanyaan(tanya);
            em.persist(o);
            em.getTransaction().commit();

            JOptionPane.showMessageDialog(null, "Sukses dibuat");
            bersih();

            Loginn n = new Loginn();
            n.setVisible(true);
            this.dispose();
        }
        em.close();
        emf.close();
    }
}
```

    b. Tambahkan Dialog baru untuk pertanyaannya.

Kode ini digunakan saat tombol Selesai diklik pada dialog input pertanyaan keamanan. Program memeriksa apakah TextField TfFav berisi teks atau tidak. Jika masih kosong, muncul pesan peringatan dan kursor diarahkan kembali ke TextField agar pengguna mengisinya. Jika sudah terisi, teks dari TfFav disimpan ke variabel jawaban, kemudian dialog ditutup dengan dispose() sehingga pengguna kembali ke tampilan utama.

```
private void jBtnSelesaiActionPerformed(java.awt.event.ActionEvent evt) {
    if (TfFav.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(parentComponent:this, message:"Pertanyaan wajib diisi!");
        TfFav.requestFocus();
        return;
    }

    jawaban = TfFav.getText().trim();
    dispose(); // Tutup dialog dan kembali ke frame utama
}
```

c. Pada class “ResetPassword” kode ini berfungsi untuk mencari akun berdasarkan username lalu menampilkan dialog pertanyaan keamanan. Jika jawaban yang diberikan sesuai dengan data di database, maka pengguna diizinkan melanjutkan proses reset password. Jika tidak sesuai, proses dihentikan dan reset password tidak dapat dilakukan.

```
private void BtnCariActionPerformed(java.awt.event.ActionEvent evt) {
    if (TfUsnLogin.getText().equals(anObject: "")) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Isi Username Terlebih Dahulu");
    } else {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory(persistenceUnitName:"PBO_Pertemuan12PU");
        EntityManager em = emf.createEntityManager();

        em.getTransaction().begin();

        String user = TfUsnLogin.getText();
        Login l = em.find(type: Login.class, o: user);

        if (l == null) {
            JOptionPane.showMessageDialog(parentComponent: null, message: "Data tidak ditemukan");
            TfUsnLogin.requestFocus();
        } else {

            Pertanyaan dialog = new Pertanyaan(
                    (java.awt.Frame) SwingUtilities.getWindowAncestor(c: this),
                    modal: true
            );
            dialog.setLocationRelativeTo(c: this); // agar muncul di tengah
            dialog.setVisible(b: true);

            String input = dialog.getPertanyaan();
            // Cek jawaban dari database
            if (!input.equals(anObject: l.getPertanyaan())) {
                JOptionPane.showMessageDialog(parentComponent: null,
                        message: "Jawaban pertanyaan salah! Tidak bisa reset password.");

                TfUsnLogin.setEditable(b: true); // username bisa diubah lagi
                TfUsnLogin.requestFocus();
                return; // Stop proses
            }
            LblPassNew.setVisible(aFlag: true);
            BtnSimpan.setVisible(aFlag: true);
            TfPassLogin.setVisible(aFlag: true);
            TfUsnLogin.setEditable(b: false);
            TfPassLogin.requestFocus();
        }
        em.getTransaction().commit();
        em.close();
        emf.close();
    }
}
```

3. Package FilmBioskop
    a. Menambahkan entity class “studio”.
    b. Pada class “utama” tambahkan button insert, update, delete, download, upload, dan cetak untuk tabel Studio.
        - Button Insert untuk tabel studio. Kode ini berfungsi untuk menampilkan dialog input data studio ketika tombol ditekan. Dengan membuat dan membuka `InsertDlgStudio`, pengguna dapat menambahkan data studio baru melalui tampilan dialog yang muncul.

```
private void jBtnInsertStudioActionPerformed(java.awt.event.ActionEvent evt) {
    InsertDlgStudio dialog = new InsertDlgStudio(parent: this, modal: true, utama: this);
    dialog.setVisible(b: true);
}
```

        - Button Update untuk tabel studio. Kode ini berfungsi untuk memproses aksi saat tombol Update Studio ditekan. Jika pengguna telah memilih salah satu baris pada tabel studio, data studio yang dipilih akan diambil lalu dikirim ke dialog update (UpdateDlgStudio) untuk dilakukan pengubahan. Setelah dialog ditutup, tabel akan diperbarui jika ada perubahan data. Jika tidak ada baris yang dipilih, sistem akan menampilkan pesan agar pengguna memilih data terlebih dahulu.

```
private void jBtnUpdateStudioActionPerformed(java.awt.event.ActionEvent evt) {
    int row = jTblStudio.getSelectedRow();
    if (row != -1) {
        // Ambil data dari tabel
        String idStudio = jTblStudio.getValueAt(row, column: 0).toString();
        String studio = jTblStudio.getValueAt(row, column: 1).toString();

        // Buka dialog update
        UpdateDlgStudio dialog = new UpdateDlgStudio(parent: this, modal: true, utama: this, idStudio, studio);
        dialog.setVisible(b: true);

        // Reload tabel setelah update
        if (dialog.isUpdated()) {
            showTableStudio();
        }
    } else {
        JOptionPane.showMessageDialog(parentComponent:this, message:"Pilih data studio yang mau diupdate dulu!");
    }
}
```

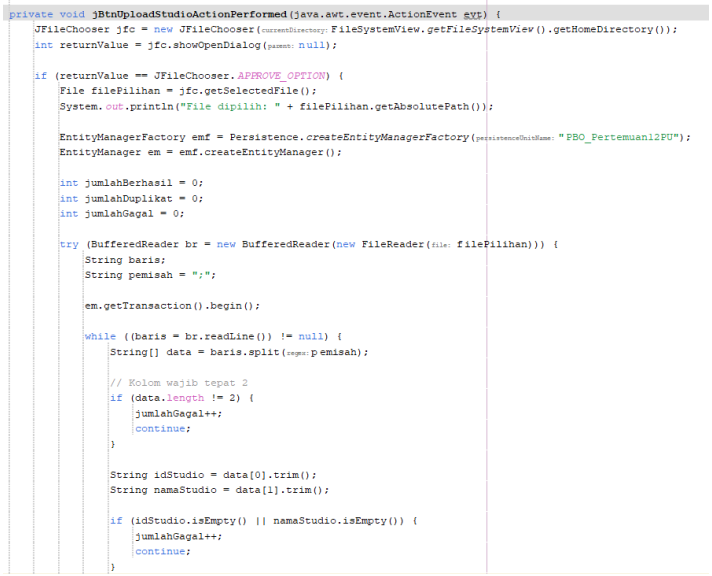
        - Button Delete untuk tabel studio. Kode ini berfungsi untuk menangani aksi tombol Delete Studio. Ketika tombol ditekan dan ada data studio yang dipilih di tabel, program mengambil ID dan nama studio dari baris tersebut lalu membuka dialog konfirmasi penghapusan (DeleteDlgStudio). Jika pengguna menyetujui dan data berhasil dihapus, tabel studio akan diperbarui. Jika tidak ada data yang dipilih, sistem memberi peringatan agar memilih data terlebih dahulu.

```
private void jBtnDeleteStudioActionPerformed(java.awt.event.ActionEvent evt) {
    int row = jTblStudio.getSelectedRow();
    if (row >= 0) {
        // Ambil data dari tabel
        String idStudio = jTblStudio.getValueAt(row, column: 0).toString();
        String Studio = jTblStudio.getValueAt(row, column: 1).toString();

        // Panggil dialog delete, kirim parent frame + data studio
        DeleteDlgStudio dialog = new DeleteDlgStudio(parent: this, modal: true, utama: this, idStudio, studio: Studio);
        dialog.setVisible(b: true);

        // Kalau user benar-benar delete, refresh tabel
        if (dialog.isDeleted()) {
            showTableStudio();
        }
    } else {
        JOptionPane.showMessageDialog(parentComponent:this, message:"Pilih data studio dulu dari tabel!");
    }
}
```

- Button Download untuk tabel studio. Kode ini berfungsi untuk menyimpan data Studio dari tabel ke file CSV. Saat tombol ditekan, pengguna memilih lokasi penyimpanan melalui *file chooser*. Jika nama file sudah ada, sistem meminta konfirmasi untuk menimpa. Setelah disetujui, program menulis seluruh isi tabel Studio ke dalam file CSV menggunakan ; sebagai pemisah data. Jika berhasil, muncul pesan sukses; jika terjadi kesalahan, tampil peringatan error.

```
private void jBtnDownloadStudioActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle(dialogTitle: "Simpan sebagai CSV");

    // Nama default
    fileChooser.setSelectedFile(new File(pathname: "Data_Studio.csv"));

    int userSelection = fileChooser.showSaveDialog(parent: this);

    if (userSelection == JFileChooser.APPROVE_OPTION) {

        File fileToSave = fileChooser.getSelectedFile();

        // Cek jika file sudah ada --> konfirmasi ke user
        if (fileToSave.exists()) {
            int confirm = JOptionPane.showConfirmDialog(
                    parentComponent: this,
                    message: "File sudah ada.\nApakah Anda ingin menimpanya?",
                    title: "Konfirmasi",
                    optionType: JOptionPane.YES_NO_OPTION,
                    messageType: JOptionPane.WARNING_MESSAGE
            );
            if (confirm != JOptionPane.YES_OPTION) {
                return; // batalkan
            }
        }

        try (FileWriter fw = new FileWriter(file: fileToSave)) {

            DefaultTableModel model = (DefaultTableModel) jTblStudio.getModel();
            int colCount = model.getColumnCount();
            int rowCount = model.getRowCount();

            // Tulis isi tabel
            for (int r = 0; r < rowCount; r++) {
                for (int c = 0; c < colCount; c++) {
                    Object value = model.getValueAt(row: r, column: c);
                    String cell = (value == null) ? "" : value.toString().replace(target: ";", replacement: ",");
                    fw.write(str: cell);
                    if (c < colCount - 1) {
                        fw.write(str: ";");
                    }
                }
                fw.write(str: "\n");
            }

            JOptionPane.showMessageDialog(
                    parentComponent: this,
                    "Data studio berhasil diekspor ke file CSV!\n" + fileToSave.getAbsolutePath(),
                    title: "Sukses",
                    messageType: JOptionPane.INFORMATION_MESSAGE
            );

        } catch (Exception e) {
            JOptionPane.showMessageDialog(
                    parentComponent: this,
                    "Terjadi kesalahan saat menyimpan: \n" + e.getMessage(),
                    title: "Error",
                    messageType: JOptionPane.ERROR_MESSAGE
            );
            e.printStackTrace();
        }
    }
}
```

- Button Upload untuk tabel studio. Kode ini digunakan untuk mengimpor data Studio dari file CSV ke database. Pengguna memilih file, lalu setiap baris data dibaca, dicek validitas dan duplikasinya, kemudian disimpan ke database jika memenuhi syarat. Setelah selesai, program menampilkan jumlah data yang berhasil diinput, duplikat, dan yang gagal.



```
private void jBtnUploadStudioActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser jfc = new JFileChooser(currentDirectory: FileSystemView.getFileSystemView().getHomeDirectory());
    int returnValue = jfc.showOpenDialog(parent: null);

    if (returnValue == JFileChooser.APPROVE_OPTION) {
        File filePilihan = jfc.getSelectedFile();
        System.out.println("File dipilih: " + filePilihan.getAbsolutePath());

        EntityManagerFactory emf = Persistence.createEntityManagerFactory(persistenceUnitName: "PBO_Pertemuan12PU");
        EntityManager em = emf.createEntityManager();

        int jumlahBerhasil = 0;
        int jumlahDuplikat = 0;
        int jumlahGagal = 0;

        try (BufferedReader br = new BufferedReader(new FileReader(file: filePilihan))) {
            String baris;
            String pemisah = ";";

            em.getTransaction().begin();

            while ((baris = br.readLine()) != null) {
                String[] data = baris.split(regex: pemisah);

                // Kolom wajib tepat 2
                if (data.length != 2) {
                    jumlahGagal++;
                    continue;
                }

                String idStudio = data[0].trim();
                String namaStudio = data[1].trim();

                if (idStudio.isEmpty() || namaStudio.isEmpty()) {
                    jumlahGagal++;
                    continue;
                }

                // Cek duplikat ID studio
                Studio existing = em.find(type: Studio.class, o: idStudio);
                if (existing != null) {
                    jumlahDuplikat++;
                    continue;
                }

                Studio s = new Studio();
                s.setIdStudio(idStudio);
                s.setNamaStudio(namaStudio);

                em.persist(o: s);
                jumlahBerhasil++;
            }

            em.getTransaction().commit();
            showTableStudio();

            JOptionPane.showMessageDialog(parentComponent: null,
                    "Proses Selesai!\n"
                    + "Berhasil: " + jumlahBerhasil + "\n"
                    + "Duplikat: " + jumlahDuplikat + "\n"
                    + "Data Error: " + jumlahGagal,
                    title: "Hasil Upload",
                    messageType: JOptionPane.INFORMATION_MESSAGE);

        } catch (Exception ex) {
            if (em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }
            Logger.getLogger(name: Utama.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
            JOptionPane.showMessageDialog(parentComponent: null, message: "Terjadi kesalahan saat upload data studio");
        } finally {
            em.close();
            emf.close();
        }
    }
}
```

- Button Cetak untuk tabel studio. Kode tersebut digunakan untuk mencetak atau menampilkan laporan Studio menggunakan JasperReport. Program membuka koneksi ke database, memuat file laporan .jasper, mengisi laporan dengan parameter dan data dari database, lalu menampilkan hasilnya melalui JasperViewer. Jika terjadi error, akan ditangani dan dicatat dalam log.

```
private void jBtnCetakStudioActionPerformed(java.awt.event.ActionEvent evt) {
    JasperReport reports;

    String path = ".\\src\\FilmBioskop\\reportStudio.jasper"; // file .jasper Studio
    try {
        try {
            Class.forName(className: driver); // driver JDBC
            conn = DriverManager.getConnection(url: koneksi, user, password);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(name: Utama.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        } catch (SQLException ex) {
            Logger.getLogger(name: Utama.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        }
        reports = (JasperReport) JRLoader.loadObjectFromFile(fileName: path);
        Map<String, Object> param = new HashMap<>();
        param.put(key: "username", value: username); // optional, jika ada parameter di report
        JasperPrint jprint = JasperFillManager.fillReport(sourceFileName: path, params: param, connection: conn);
        JasperViewer jviewer = new JasperViewer(jasperPrint: jprint, isExitOnClose: false);
        jviewer.setDefaultCloseOperation(operation: DISPOSE_ON_CLOSE);
        jviewer.setVisible(b: true);

    } catch (JRException ex) {
        Logger.getLogger(name: Utama.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
}
```

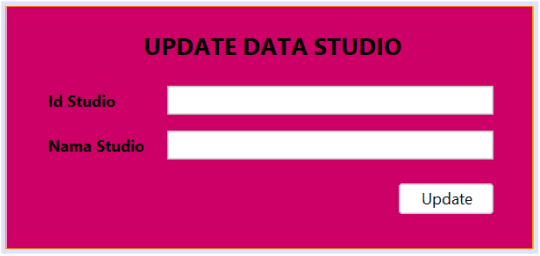
c. Menambahkan Dialog “InsertDlgStudio”.


Kode ini berfungsi untuk menyimpan data studio baru ke database dengan melakukan validasi terlebih dahulu. Sistem mengecek apakah input sudah terisi, memastikan ID dan nama studio belum pernah digunakan, lalu menyimpan data jika valid. Setelah berhasil, form dibersihkan dan tabel diperbarui, sementara jika terjadi error transaksi akan dibatalkan dan pesan kesalahan ditampilkan.

```
private void BtnSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    if (TfIdStudio.getText().trim().equals(anObject: "") || TfStudio.getText().trim().equals(anObject: "")) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Isi semua data terlebih dahulu!");
        return;
    }

    String idStudio = TfIdStudio.getText().trim();
    String namaStudio = TfStudio.getText().trim();

    EntityManagerFactory emf = Persistence.createEntityManagerFactory(persistenceUnitName: "PBO_Pertemuan12PU");
    EntityManager em = emf.createEntityManager();

    try {
        em.getTransaction().begin();

        // Cek apakah ID Studio sudah ada
        Studio existingStudio = em.find(type: Studio.class, o: idStudio);
        if (existingStudio != null) {
            JOptionPane.showMessageDialog(parentComponent: null, message: "ID Studio sudah digunakan! Gunakan ID lain.", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
            em.getTransaction().rollback();
            return;
        }

        // Cek apakah nama studio sudah ada
        Query q = em.createQuery(string: "SELECT COUNT(s) FROM Studio s WHERE s.namaStudio = :nama");
        q.setParameter(string: "nama", o: namaStudio);
        long count = (long) q.getSingleResult();

        if (count > 0) {
            JOptionPane.showMessageDialog(parentComponent: null, message: "Nama studio sudah ada!", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
            em.getTransaction().rollback();
            return;
        }
```

```
        // Buat object Studio baru
        Studio s = new Studio();
        s.setIdStudio(idStudio);
        s.setNamaStudio(namaStudio);

        em.persist(s);
        em.getTransaction().commit();

        JOptionPane.showMessageDialog(parentComponent:null, message:"Data studio berhasil disimpan!");
        bersih(); // method untuk membersihkan input studio
        utama.showTableStudio(); // refresh tabel studio

    } catch (Exception e) {
        if (em.getTransaction().isActive()) {
            em.getTransaction().rollback();
        }
        JOptionPane.showMessageDialog(parentComponent:null, "Gagal menyimpan data!\n" + e.getMessage(), title:"Error", messageType:JOptionPane.ERROR_MESSAGE);
        e.printStackTrace();
    } finally {
        em.close();
        emf.close();
    }
}
```

d. Menambahkan Dialog “UpdateDlgStudio”.




Kode ini digunakan untuk memperbarui data studio yang sudah ada di database. Pertama, sistem memeriksa apakah field ID Studio dan Nama Studio sudah diisi. Setelah itu, dilakukan pencarian studio berdasarkan ID. Jika tidak ditemukan, proses dibatalkan dan pesan muncul. Jika ditemukan, nama studio diperbarui, lalu dicek apakah nama tersebut sudah dipakai oleh studio lain untuk mencegah duplikasi. Jika nama aman, perubahan disimpan dengan commit transaksi. Setelah berhasil, sistem menampilkan pesan sukses, membersihkan input, dan me-refresh tabel studio serta jadwal. Jika terjadi kesalahan, transaksi dibatalkan dan pesan error ditampilkan.

```
private void BtnUpdateActionPerformed(java.awt.event.ActionEvent evt) {
    if (TfIdStudio.getText().trim().equals(anObject: "") || TfStudio.getText().trim().equals(anObject: "")) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Isi semua data terlebih dahulu!");
        return;
    }

    String idStudio = TfIdStudio.getText().trim();
    String namaStudio = TfStudio.getText().trim();

    EntityManagerFactory emf = Persistence.createEntityManagerFactory(persistenceUnitName: "PBO_Pertemuan12PU");
    EntityManager em = emf.createEntityManager();

    try {
        em.getTransaction().begin();

        // Cari Studio berdasarkan ID
        Studio s = em.find(type: Studio.class, o: idStudio);
        if (s == null) {
            JOptionPane.showMessageDialog(parentComponent: null, message: "Data studio tidak ditemukan!");
            em.getTransaction().rollback();
        } else {
            s.setNamaStudio(namaStudio); // update nama studio

            // Cek apakah nama studio sudah dipakai oleh studio lain
            List<Studio> cekNama = em.createQuery(
                    string: "SELECT st FROM Studio st WHERE st.namaStudio = :nama AND st.idStudio <> :id", type: Studio.class)
                    .setParameter(string: "nama", o: namaStudio)
                    .setParameter(string: "id", o: idStudio)
                    .getResultList();

            if (!cekNama.isEmpty()) {
                JOptionPane.showMessageDialog(parentComponent: null, message: "Nama studio sudah digunakan! Gunakan nama lain.", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
                em.getTransaction().rollback();
                return;
            }
```

```
            em.getTransaction().commit();
            JOptionPane.showMessageDialog(parentComponent: null, message: "Data studio berhasil diupdate!");
            bersih(); // method untuk bersihkan input
        }
    } catch (Exception e) {
        if (em.getTransaction().isActive()) {
            em.getTransaction().rollback();
        }
        JOptionPane.showMessageDialog(parentComponent: null, "Gagal mengupdate data!\n" + e.getMessage(), title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
        e.printStackTrace();
    } finally {
        em.close();
        emf.close();
    }
    utama.showTableStudio();
    utama.showTableJadwal();
}
```

e. Menambahkan Dialog “DeleteDlgStudio”.


Kode ini digunakan untuk menghapus data studio dari database. Sistem memastikan data sudah dipilih, lalu meminta konfirmasi pengguna. Jika disetujui, data studio dicari dan dihapus, tabel diperbarui, serta input dibersihkan. Jika dibatalkan atau data tidak ditemukan, proses tidak dilanjutkan.

```
private void BtnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    if (TfIdStudio.getText().equals(anObject: "") || TfStudio.getText().equals(anObject: "")) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"Pilih data studio terlebih dahulu!");
    } else {
        String idStudio = TfIdStudio.getText();
        EntityManagerFactory emf = Persistence.createEntityManagerFactory(persistenceUnitName: "PBO_Pertemuan12PU");
        EntityManager em = emf.createEntityManager();

        int jawab = JOptionPane.showConfirmDialog(parentComponent:null, message:"Yakin ingin menghapus data studio ini?",
                title:"Konfirmasi Hapus", optionType:JOptionPane.YES_NO_OPTION);
        switch (jawab) {
            case JOptionPane.YES_OPTION -> {
                em.getTransaction().begin();

                Studio s = em.find(type: Studio.class, o: idStudio);
                if (s == null) {
                    JOptionPane.showMessageDialog(parentComponent:null, message:"Data studio tidak ditemukan");
                } else {
                    em.remove(o: s);

                    em.getTransaction().commit();
                    JOptionPane.showMessageDialog(parentComponent:null, message:"Data studio berhasil dihapus");
                    em.close();
                    bersih(); // method untuk bersihkan input
                }
            }
            case JOptionPane.NO_OPTION ->
                JOptionPane.showMessageDialog(parentComponent:this, message:"Hapus dibatalkan");
        }

        utama.showTableStudio(); // refresh tabel studio
        utama.showTableJadwal(); // refresh tabel jadwal, karena mungkin ada relasi
        dispose(); // tutup dialog/hapus
    }
}
```

f. Pada class utama tambahkan button download untuk tabel film. Kode ini digunakan untuk mengekspor data film dari tabel ke file CSV. Pengguna memilih lokasi penyimpanan menggunakan JFileChooser, lalu program menuliskan setiap data dalam tabel ke dalam file dengan pemisah titik koma.

Jika file sudah ada, pengguna diminta konfirmasi untuk menimpa. Setelah berhasil, muncul pesan bahwa data berhasil disimpan. Jika gagal, ditampilkan pesan error.

```
private void jBtnDownloadFilmActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle(dialogTitle: "Simpan sebagai CSV");

    // Nama default
    fileChooser.setSelectedFile(new File(pathname: "Data_Film.csv"));

    int userSelection = fileChooser.showSaveDialog(parent: this);

    if (userSelection == JFileChooser.APPROVE_OPTION) {

        File fileToSave = fileChooser.getSelectedFile();

        // Cek jika file sudah ada --> konfirmasi ke user
        if (fileToSave.exists()) {
            int confirm = JOptionPane.showConfirmDialog(
                    parentComponent: this,
                    message: "File sudah ada.\nApakah Anda ingin menimpanya?",
                    title: "Konfirmasi",
                    optionType: JOptionPane.YES_NO_OPTION,
                    messageType: JOptionPane.WARNING_MESSAGE
            );
            if (confirm != JOptionPane.YES_OPTION) {
                return; // batalkan
            }
        }

        try (FileWriter fw = new FileWriter(file: fileToSave)) {

            DefaultTableModel model = (DefaultTableModel) jTblFilm.getModel();
            int colCount = model.getColumnCount();
            int rowCount = model.getRowCount();

            // Tulis isi tabel
            for (int r = 0; r < rowCount; r++) {
                for (int c = 0; c < colCount; c++) {
                    Object value = model.getValueAt(row: r, column: c);
                    String cell = (value == null) ? "" : value.toString().replace(target: ";", replacement: ",");
                    fw.write(str: cell);
                    if (c < colCount - 1) {
                        fw.write(str: ";");
                    }
                }
                fw.write(str: "\n");
            }

            JOptionPane.showMessageDialog(
                    parentComponent: this,
                    "Data berhasil diekspor ke file CSV!\n" + fileToSave.getAbsolutePath(),
                    title: "Sukses",
                    messageType: JOptionPane.INFORMATION_MESSAGE
            );

        } catch (Exception e) {
            JOptionPane.showMessageDialog(
                    parentComponent: this,
                    "Terjadi kesalahan saat menyimpan:\n" + e.getMessage(),
                    title: "Error",
                    messageType: JOptionPane.ERROR_MESSAGE
            );
            e.printStackTrace();
        }
    }
}
```

g. Pada class “utama” tambahkan button Download untuk tabel jadwal tayang. Kode ini digunakan untuk mengekspor data jadwal tayang dari tabel ke file CSV. Pengguna memilih lokasi penyimpanan melalui JFileChooser, kemudian program menulis setiap baris tabel ke file dengan pemisah titik koma. Jika file sudah ada, akan muncul konfirmasi untuk menimpanya.

Setelah proses berhasil, muncul pesan sukses, dan jika terjadi kesalahan akan ditampilkan pesan error.

```
private void jBtnDownloadJadwalActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle(dialogTitle: "Simpan sebagai CSV");

    // Nama default
    fileChooser.setSelectedFile(new File(pathname: "Data_JadwalTayang.csv"));

    int userSelection = fileChooser.showSaveDialog(parent: this);

    if (userSelection == JFileChooser.APPROVE_OPTION) {

        File fileToSave = fileChooser.getSelectedFile();

        // Cek jika file sudah ada --> konfirmasi ke user
        if (fileToSave.exists()) {
            int confirm = JOptionPane.showConfirmDialog(
                    parentComponent: this,
                    message: "File sudah ada.\nApakah Anda ingin menimpanya?",
                    title: "Konfirmasi",
                    optionType: JOptionPane.YES_NO_OPTION,
                    messageType: JOptionPane.WARNING_MESSAGE
            );
            if (confirm != JOptionPane.YES_OPTION) {
                return; // batalkan
            }
        }

        try (FileWriter fw = new FileWriter(file: fileToSave)) {

            DefaultTableModel model = (DefaultTableModel) jTblJadwal.getModel();
            int colCount = model.getColumnCount();
            int rowCount = model.getRowCount();

            // Tulis isi tabel
            for (int r = 0; r < rowCount; r++) {
                for (int c = 0; c < colCount; c++) {
                    Object value = model.getValueAt(row: r, column: c);
                    String cell = (value == null) ? "" : value.toString().replace(target: ";", replacement: ",");
                    fw.write(str: cell);
                    if (c < colCount - 1) {
                        fw.write(str: ";");
                    }
                }
                fw.write(str: "\n");
            }

            JOptionPane.showMessageDialog(
                    parentComponent: this,
                    "Data berhasil diekspor ke file CSV!\n" + fileToSave.getAbsolutePath(),
                    title: "Sukses",
                    messageType: JOptionPane.INFORMATION_MESSAGE
            );

        } catch (Exception e) {
            JOptionPane.showMessageDialog(
                    parentComponent: this,
                    "Terjadi kesalahan saat menyimpan:\n" + e.getMessage(),
                    title: "Error",
                    messageType: JOptionPane.ERROR_MESSAGE
            );
            e.printStackTrace();
        }
    }
}
```