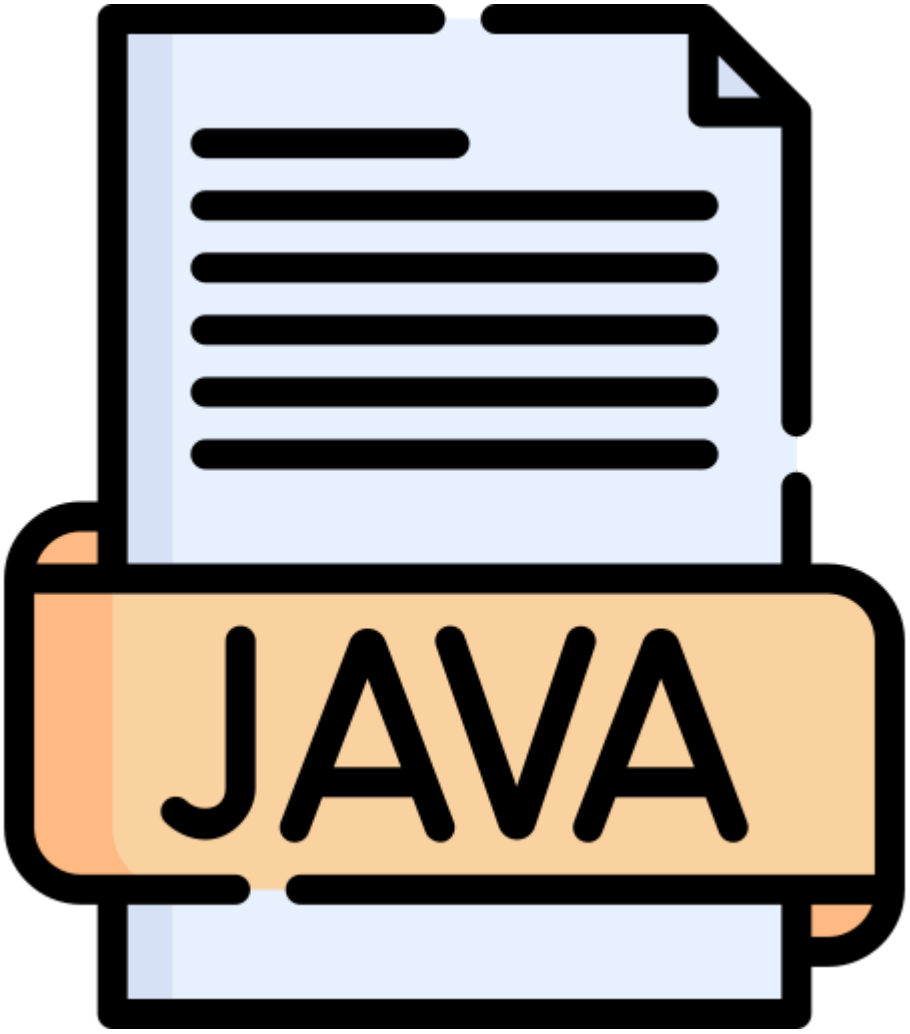
Rafael Martín Mayor

Actividad 1 UT 2 - Consultas JDBC



Índice

Explicación de las secciones del código **3**

Capturas de pantalla **4**

Código completo **5**

Explicación de las secciones del código

Clase Main.java

Esta clase contiene el punto de entrada del programa (main()), y actúa como interfaz de usuario por consola.

```
import java.util.*;
```

Importa utilidades de Java, como Scanner (para leer del teclado) y List.

Clase Main

```
public class Main {
    public static void main(String[] args) {
```

Aquí comienza el programa. Dentro del main, se crean los objetos y se controla el menú.

Creación del DAO (Data Access Object)

```
        PeliculaDAO dao = new PeliculaDAO();
```

Se crea una instancia de PeliculaDAO, que es la clase encargada de acceder a la base de datos SQLite (peliculas.db).

El constructor de PeliculaDAO automáticamente llama a crearTabla(), por si no existe la tabla.

Inserts de prueba (comentados)

```
        /*
        // Inserts de prueba
        dao.insertar(new Pelicula("Inception", "Christopher Nolan",
"Ciencia Ficcion", 2010, 148, 8.8, true));
        dao.insertar(new Pelicula("Interstellar", "Christopher Nolan",
"Ciencia Ficcion", 2014, 169, 8.6, true));
        dao.insertar(new Pelicula("Tenet", "Christopher Nolan",
"Accion", 2020, 150, 6.9, false));
        dao.insertar(new Pelicula("The Dark Knight", "Christopher
Nolan", "Accion", 2008, 152, 9.0, true));
```

```
        dao.insertar(new Pelicula("Avatar", "James Cameron", "Ciencia
Ficcion", 2009, 162, 7.8, true));
        dao.insertar(new Pelicula("Pulp Fiction", "Quentin Tarantino",
"Crimen", 1994, 154, 8.9, true));
        dao.insertar(new Pelicula("Whiplash", "Damien Chazelle",
"Drama", 2014, 107, 8.5, true));
        dao.insertar(new Pelicula("La La Land", "Damien Chazelle",
"Musical", 2016, 128, 8.0, true));
        dao.insertar(new Pelicula("The Irishman", "Martin Scorsese",
"Crimen", 2019, 209, 7.8, true));
        dao.insertar(new Pelicula("The Godfather", "Francis Ford
Coppola", "Crimen", 1972, 175, 9.2, false));
        */
```

Son ejemplos para poblar la base de datos manualmente.
Están comentados, pero si se descomentan, insertan películas de prueba en la BD.

Menú interactivo

```
        Scanner sc = new Scanner(System.in);
        int opcion;
```

Se prepara un lector (Scanner) para pedir al usuario opciones por teclado.

Luego, con un bucle do-while, se muestra un menú con distintas funcionalidades.

Cada número (1–10) ejecuta una consulta SQL distinta mediante métodos del DAO.

Estructura del menú

```
        do {
            System.out.println("\n--- CATÁLOGO DE PELÍCULAS ---");
            System.out.println("1. Listar todas las películas");
            System.out.println("2. Buscar por género");
            System.out.println("3. Buscar por director (rating ≥ 7)");
            System.out.println("4. Listar disponibles después de
2015");
            System.out.println("5. Top 5 por rating");
            System.out.println("6. Buscar por palabra clave en
título");
            System.out.println("7. Contar películas por género");
            System.out.println("8. Duración media por director");
            System.out.println("9. No disponibles con rating > 8");
```

```
            System.out.println("10. Más largas que la media");
            System.out.println("0. Salir");
            System.out.print("Opción: ");
            opcion = Integer.parseInt(sc.nextLine());
```

Muestra las opciones por pantalla.

Lee la opción del usuario y la convierte a número entero.

```
            switch (opcion) {
                case 1 ->
dao.listarAlfabeticamente().forEach(System.out::println);
                case 2 -> {
                    System.out.print("Género: ");

dao.buscarPorGenero(sc.nextLine()).forEach(System.out::println);
                }
                case 3 -> {
                    System.out.print("Director: ");

dao.buscarPorDirectorRating(sc.nextLine()).forEach(System.out::println)
;
                }
                case 4 ->
dao.disponiblesDesde2015().forEach(System.out::println);
                case 5 ->
dao.top5Rating().forEach(System.out::println);
                case 6 -> {
                    System.out.print("Palabra clave: ");

dao.buscarPorPalabraClave(sc.nextLine()).forEach(System.out::println);
                }
                case 7 -> dao.contarPorGenero();
                case 8 -> {
                    System.out.print("Director: ");
                    dao.duracionMediaPorDirector(sc.nextLine());
                }
                case 9 ->
dao.noDisponiblesRatingMayor8().forEach(System.out::println);
                case 10 ->
dao.masLargasQueMedia().forEach(System.out::println);
                case 0 -> System.out.println("Saliendo...");
```

```
                default -> System.out.println("Opción no válida.");
            }
        } while (opcion != 0);
    }
}
```

Dependiendo de la opción, ejecuta un método del DAO:

Por ejemplo:

```
case 1 -> dao.listarAlfabeticamente().forEach(System.out::println);
```

Llama a listarAlfabeticamente() y muestra cada película.

Otros casos:

buscarPorGenero(), buscarPorDirectorRating(), etc., piden datos por teclado.

contarPorGenero() y duracionMediaPorDirector() muestran resultados calculados.

case 0 sale del programa.

2. Clase Pelicula.java

```
public class Pelicula {
    private int idPelicula;
    private String titulo;
    private String director;
    private String genero;
    private int anio;
    private int duracion;
    private double rating;
    private boolean disponible;
```

Representa el modelo de datos de una película.

Cada película tiene esos campos, equivalentes a las columnas en la base de datos.

Constructores

```
public Pelicula() {}
```

Vacío: permite crear un objeto sin inicializar.

```
    public Pelicula(String titulo, String director, String genero, int
anio, int duracion, double rating, boolean disponible) {
        this.titulo = titulo;
        this.director = director;
        this.genero = genero;
        this.anio = anio;
        this.duracion = duracion;
        this.rating = rating;
        this.disponible = disponible;
    }
```

Con parámetros: inicializa todos los campos excepto idPelicula (cuando aún no se ha insertado en BD).

```
    public Pelicula(int idPelicula, String titulo, String director,
String genero, int anio, int duracion, double rating, boolean
disponible) {
        this(titulo, director, genero, anio, duracion, rating,
disponible);
        this.idPelicula = idPelicula;
    }
```

Con id: usado al leer desde la BD (ya tiene id_pelicula asignado por SQLite).

Getters y Setters

```
    // Getters y Setters
    public int getIdPelicula() { return idPelicula; }
    public void setIdPelicula(int idPelicula) { this.idPelicula =
idPelicula; }

    public String getTitulo() { return titulo; }
    public void setTitulo(String titulo) { this.titulo = titulo; }

    public String getDirector() { return director; }
    public void setDirector(String director) { this.director =
director; }
```

```
    public String getGenero() { return genero; }
    public void setGenero(String genero) { this.genero = genero; }

    public int getAnio() { return anio; }
    public void setAnio(int anio) { this.anio = anio; }

    public int getDuracion() { return duracion; }
    public void setDuracion(int duracion) { this.duracion = duracion; }

    public double getRating() { return rating; }
    public void setRating(double rating) { this.rating = rating; }

    public boolean isDisponible() { return disponible; }
    public void setDisponible(boolean disponible) { this.disponible =
disponible; }
```

Permiten acceder o modificar los atributos de forma controlada (encapsulación).

```
    @Override
    public String toString() {
        return String.format("%s (%d) - Director: %s | Género: %s |
Duración: %d min | Rating: %.1f | %s",
                titulo, anio, director, genero, duracion, rating,
(disponible ? "Disponible" : "No disponible"));
    }
}
```

Devuelve una cadena legible con la información completa de la película.

3. Clase PeliculaDAO.java

DAO = Data Access Object.
Se encarga de todas las operaciones con la base de datos SQLite.

Atributo url

```
private final String url = "jdbc:sqlite:peliculas.db";
```

Indica la ruta (URL JDBC) del archivo de base de datos.

```
    public PeliculaDAO() {
        crearTabla();
    }
```

Al crear el DAO, se asegura de que la tabla exista (si no, la crea).

```
    private void crearTabla() {
        String sql = """
        CREATE TABLE IF NOT EXISTS Peliculas (
            id_pelicula INTEGER PRIMARY KEY AUTOINCREMENT,
            titulo TEXT NOT NULL,
            director TEXT,
            genero TEXT,
            anio INTEGER,
            duracion INTEGER,
            rating REAL,
            disponible INTEGER
        );
        """;
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement()) {
            stmt.execute(sql);
        } catch (SQLException e) {
            System.out.println("Error al crear tabla: " +
e.getMessage());
        }
    }
```

Crea la tabla Peliculas si no existe.

Usa try-with-resources para abrir conexión y ejecutar la sentencia de creación.

```
    public void insertar(Pelicula p) {
        String sql = "INSERT INTO Peliculas (titulo, director, genero,
anio, duracion, rating, disponible) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (Connection conn = DriverManager.getConnection(url);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, p.getTitulo());
            pstmt.setString(2, p.getDirector());
            pstmt.setString(3, p.getGenero());
            pstmt.setInt(4, p.getAnio());
            pstmt.setInt(5, p.getDuracion());
            pstmt.setDouble(6, p.getRating());
```

```
            pstmt.setInt(7, p.isDisponible() ? 1 : 0);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println("Error al insertar: " + e.getMessage());
        }
    }
```

Inserta una película nueva:

String sql = "INSERT INTO Peliculas (...) VALUES (?, ?, ?, ?, ?, ?, ?)";

Usa PreparedStatement (seguro contra inyecciones SQL).

Convierte boolean disponible a 1 o 0.

Métodos de consulta

Todos devuelven una lista de objetos Pelicula o imprimen información.

Por ejemplo:

```
    public List<Pelicula> listarAlfabeticamente() {
        return obtenerLista("SELECT * FROM Peliculas ORDER BY titulo
ASC");
    }
```

Obtiene todas las películas ordenadas por título.

buscarPorGenero(String genero)

Filtra por género (usa LIKE para coincidencias parciales).

buscarPorDirectorRating(String director)

Busca por director con rating >= 7.

disponiblesDesde2015()

Filtra las disponibles (disponible = 1) y con año > 2015.

top5Rating()

Ordena por rating descendente y limita a las 5 mejores.

buscarPorPalabraClave(String palabra)

Busca por título que contenga la palabra dada.

Consultas de agregación

```
    public void contarPorGenero() {
        String sql = "SELECT genero, COUNT(*) AS total FROM Peliculas
GROUP BY genero";
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                System.out.printf("%s -> %d películas%n",
rs.getString("genero"), rs.getInt("total"));
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
```

Cuenta cuántas películas hay por género y las imprime en consola.

```
    public void duracionMediaPorDirector(String director) {
        String sql = "SELECT AVG(duracion) AS media FROM Peliculas
WHERE director LIKE '%" + director + "%'";
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            if (rs.next()) {
                System.out.printf("Duración media de las películas de
%s: %.2f minutos%n", director, rs.getDouble("media"));
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
```

Calcula y muestra la duración media de las películas de ese director.

Otras consultas útiles
noDisponiblesRatingMayor8()

Películas no disponibles (disponible = 0) pero con rating > 8.

```
    public List<Pelicula> masLargasQueMedia() {
        String sql = """
            SELECT * FROM Peliculas
            WHERE duracion > (SELECT AVG(duracion) FROM Peliculas)
            ORDER BY duracion DESC
        """;
        return obtenerLista(sql);
    }
```

Devuelve películas más largas que la duración media general.

```
    private List<Pelicula> obtenerLista(String sql) {
        List<Pelicula> lista = new ArrayList<>();
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                lista.add(new Pelicula(
                        rs.getInt("id_pelicula"),
                        rs.getString("titulo"),
                        rs.getString("director"),
                        rs.getString("genero"),
                        rs.getInt("anio"),
                        rs.getInt("duracion"),
                        rs.getDouble("rating"),
                        rs.getInt("disponible") == 1
                ));
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
        return lista;
    }
}
```

Es el núcleo común para todas las consultas que devuelven películas:

Ejecuta el SQL recibido. Recorre el ResultSet. Crea objetos Pelicula con los datos de cada fila. Devuelve una lista de resultados. Así se evita repetir código en cada método de consulta.

Capturas de pantalla

```
--- CATÁLOGO DE PELÍCULAS ---
1. Listar todas las películas
2. Buscar por género
3. Buscar por director (rating ? 7)
4. Listar disponibles después de 2015
5. Top 5 por rating
6. Buscar por palabra clave en título
7. Contar películas por género
8. Duración media por director
9. No disponibles con rating > 8
10. Más largas que la media
0. Salir
Opción: 1
Avatar (2009) - Director: James Cameron | Género: Ciencia Ficcion | Duración: 162 min | Rating: 7,8 | Disponible
Inception (2010) - Director: Christopher Nolan | Género: Ciencia Ficcion | Duración: 148 min | Rating: 8,8 | Disponible
Interstellar (2014) - Director: Christopher Nolan | Género: Ciencia Ficcion | Duración: 169 min | Rating: 8,6 | Disponible
La La Land (2016) - Director: Damien Chazelle | Género: Musical | Duración: 128 min | Rating: 8,0 | Disponible
Pulp Fiction (1994) - Director: Quentin Tarantino | Género: Crimen | Duración: 154 min | Rating: 8,9 | Disponible
Tenet (2020) - Director: Christopher Nolan | Género: Accion | Duración: 150 min | Rating: 6,9 | No disponible
The Dark Knight (2008) - Director: Christopher Nolan | Género: Accion | Duración: 152 min | Rating: 9,0 | Disponible
The Godfather (1972) - Director: Francis Ford Coppola | Género: Crimen | Duración: 175 min | Rating: 9,2 | No disponible
The Irishman (2019) - Director: Martin Scorsese | Género: Crimen | Duración: 209 min | Rating: 7,8 | Disponible
Whiplash (2014) - Director: Damien Chazelle | Género: Drama | Duración: 107 min | Rating: 8,5 | Disponible

--- CATÁLOGO DE PELÍCULAS ---
1. Listar todas las películas
2. Buscar por género
3. Buscar por director (rating ? 7)
4. Listar disponibles después de 2015
5. Top 5 por rating
6. Buscar por palabra clave en título
7. Contar películas por género
8. Duración media por director
9. No disponibles con rating > 8
10. Más largas que la media
0. Salir
Opción: 2
Género: Accion
Tenet (2020) - Director: Christopher Nolan | Género: Accion | Duración: 150 min | Rating: 6,9 | No disponible
The Dark Knight (2008) - Director: Christopher Nolan | Género: Accion | Duración: 152 min | Rating: 9,0 | Disponible

--- CATÁLOGO DE PELÍCULAS ---
1. Listar todas las películas
2. Buscar por género
3. Buscar por director (rating ? 7)
4. Listar disponibles después de 2015
5. Top 5 por rating
6. Buscar por palabra clave en título
7. Contar películas por género
8. Duración media por director
9. No disponibles con rating > 8
10. Más largas que la media
0. Salir
Opción: 3
Director: Christopher
Inception (2010) - Director: Christopher Nolan | Género: Ciencia Ficcion | Duración: 148 min | Rating: 8,8 | Disponible
Interstellar (2014) - Director: Christopher Nolan | Género: Ciencia Ficcion | Duración: 169 min | Rating: 8,6 | Disponible
The Dark Knight (2008) - Director: Christopher Nolan | Género: Accion | Duración: 152 min | Rating: 9,0 | Disponible
```

```
Opción: 4
La La Land (2016) - Director: Damien Chazelle | Género: Musical | Duración: 128 min | Rating: 8,0 | Disponible
The Irishman (2019) - Director: Martin Scorsese | Género: Crimen | Duración: 209 min | Rating: 7,8 | Disponible

--- CATÁLOGO DE PELÍCULAS ---
1. Listar todas las películas
2. Buscar por género
3. Buscar por director (rating ? 7)
4. Listar disponibles después de 2015
5. Top 5 por rating
6. Buscar por palabra clave en título
7. Contar películas por género
8. Duración media por director
9. No disponibles con rating > 8
10. Más largas que la media
0. Salir
Opción: 5
The Godfather (1972) - Director: Francis Ford Coppola | Género: Crimen | Duración: 175 min | Rating: 9,2 | No disponible
The Dark Knight (2008) - Director: Christopher Nolan | Género: Accion | Duración: 152 min | Rating: 9,0 | Disponible
Pulp Fiction (1994) - Director: Quentin Tarantino | Género: Crimen | Duración: 154 min | Rating: 8,9 | Disponible
Inception (2010) - Director: Christopher Nolan | Género: Ciencia Ficcion | Duración: 148 min | Rating: 8,8 | Disponible
Interstellar (2014) - Director: Christopher Nolan | Género: Ciencia Ficcion | Duración: 169 min | Rating: 8,6 | Disponible

--- CATÁLOGO DE PELÍCULAS ---
1. Listar todas las películas
2. Buscar por género
3. Buscar por director (rating ? 7)
4. Listar disponibles después de 2015
5. Top 5 por rating
6. Buscar por palabra clave en título
7. Contar películas por género
8. Duración media por director
9. No disponibles con rating > 8
10. Más largas que la media
0. Salir
Opción: 6
Palabra clave: iris
The Irishman (2019) - Director: Martin Scorsese | Género: Crimen | Duración: 209 min | Rating: 7,8 | Disponible

--- CATÁLOGO DE PELÍCULAS ---
1. Listar todas las películas
2. Buscar por género
3. Buscar por director (rating ? 7)
4. Listar disponibles después de 2015
5. Top 5 por rating
6. Buscar por palabra clave en título
7. Contar películas por género
8. Duración media por director
9. No disponibles con rating > 8
10. Más largas que la media
0. Salir
Opción: 7
Accion -> 2 películas
Ciencia Ficcion -> 3 películas
Crimen -> 3 películas
Drama -> 1 películas
Musical -> 1 películas
```

```
0. Salir
Opción: 8
Director: Christopher
Duración media de las películas de Christopher: 154,75 minutos

--- CATÁLOGO DE PELÍCULAS ---
1. Listar todas las películas
2. Buscar por género
3. Buscar por director (rating ? 7)
4. Listar disponibles después de 2015
5. Top 5 por rating
6. Buscar por palabra clave en título
7. Contar películas por género
8. Duración media por director
9. No disponibles con rating > 8
10. Más largas que la media
0. Salir
Opción: 9
The Godfather (1972) - Director: Francis Ford Coppola | Género: Crimen | Duración: 175 min | Rating: 9,2 | No disponible

--- CATÁLOGO DE PELÍCULAS ---
1. Listar todas las películas
2. Buscar por género
3. Buscar por director (rating ? 7)
4. Listar disponibles después de 2015
5. Top 5 por rating
6. Buscar por palabra clave en título
7. Contar películas por género
8. Duración media por director
9. No disponibles con rating > 8
10. Más largas que la media
0. Salir
Opción: 10
The Irishman (2019) - Director: Martin Scorsese | Género: Crimen | Duración: 209 min | Rating: 7,8 | Disponible
The Godfather (1972) - Director: Francis Ford Coppola | Género: Crimen | Duración: 175 min | Rating: 9,2 | No disponible
Interstellar (2014) - Director: Christopher Nolan | Género: Ciencia Ficcion | Duración: 169 min | Rating: 8,6 | Disponible
Avatar (2009) - Director: James Cameron | Género: Ciencia Ficcion | Duración: 162 min | Rating: 7,8 | Disponible
```

Código completo

Main.java:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        PeliculaDAO dao = new PeliculaDAO();
        /*
        // Inserts de prueba
        dao.insertar(new Pelicula("Inception", "Christopher Nolan",
"Ciencia Ficcion", 2010, 148, 8.8, true));
        dao.insertar(new Pelicula("Interstellar", "Christopher Nolan",
"Ciencia Ficcion", 2014, 169, 8.6, true));
        dao.insertar(new Pelicula("Tenet", "Christopher Nolan",
"Accion", 2020, 150, 6.9, false));
        dao.insertar(new Pelicula("The Dark Knight", "Christopher
Nolan", "Accion", 2008, 152, 9.0, true));
        dao.insertar(new Pelicula("Avatar", "James Cameron", "Ciencia
Ficcion", 2009, 162, 7.8, true));
        dao.insertar(new Pelicula("Pulp Fiction", "Quentin Tarantino",
"Crimen", 1994, 154, 8.9, true));
        dao.insertar(new Pelicula("Whiplash", "Damien Chazelle",
"Drama", 2014, 107, 8.5, true));
        dao.insertar(new Pelicula("La La Land", "Damien Chazelle",
"Musical", 2016, 128, 8.0, true));
        dao.insertar(new Pelicula("The Irishman", "Martin Scorsese",
"Crimen", 2019, 209, 7.8, true));
        dao.insertar(new Pelicula("The Godfather", "Francis Ford
Coppola", "Crimen", 1972, 175, 9.2, false));
        */
        Scanner sc = new Scanner(System.in);
        int opcion;

        do {
            System.out.println("\n--- CATÁLOGO DE PELÍCULAS ---");
            System.out.println("1. Listar todas las películas");
            System.out.println("2. Buscar por género");
            System.out.println("3. Buscar por director (rating ≥ 7)");
            System.out.println("4. Listar disponibles después de
2015");
```

```
            System.out.println("5. Top 5 por rating");
            System.out.println("6. Buscar por palabra clave en
título");
            System.out.println("7. Contar películas por género");
            System.out.println("8. Duración media por director");
            System.out.println("9. No disponibles con rating > 8");
            System.out.println("10. Más largas que la media");
            System.out.println("0. Salir");
            System.out.print("Opción: ");
            opcion = Integer.parseInt(sc.nextLine());

            switch (opcion) {
                case 1 ->
dao.listarAlfabeticamente().forEach(System.out::println);
                case 2 -> {
                    System.out.print("Género: ");

dao.buscarPorGenero(sc.nextLine()).forEach(System.out::println);
                }
                case 3 -> {
                    System.out.print("Director: ");

dao.buscarPorDirectorRating(sc.nextLine()).forEach(System.out::println)
;
                }
                case 4 ->
dao.disponiblesDesde2015().forEach(System.out::println);
                case 5 ->
dao.top5Rating().forEach(System.out::println);
                case 6 -> {
                    System.out.print("Palabra clave: ");

dao.buscarPorPalabraClave(sc.nextLine()).forEach(System.out::println);
                }
                case 7 -> dao.contarPorGenero();
                case 8 -> {
                    System.out.print("Director: ");
                    dao.duracionMediaPorDirector(sc.nextLine());
                }
                case 9 ->
dao.noDisponiblesRatingMayor8().forEach(System.out::println);
                case 10 ->
dao.masLargasQueMedia().forEach(System.out::println);
```

```
                case 0 -> System.out.println("Saliendo...");
                default -> System.out.println("Opción no válida.");
            }
        } while (opcion != 0);
    }
}
```

Pelicula.java:

```
public class Pelicula {
    private int idPelicula;
    private String titulo;
    private String director;
    private String genero;
    private int anio;
    private int duracion;
    private double rating;
    private boolean disponible;

    public Pelicula() {}

    public Pelicula(String titulo, String director, String genero, int
anio, int duracion, double rating, boolean disponible) {
        this.titulo = titulo;
        this.director = director;
        this.genero = genero;
        this.anio = anio;
        this.duracion = duracion;
        this.rating = rating;
        this.disponible = disponible;
    }

    public Pelicula(int idPelicula, String titulo, String director,
String genero, int anio, int duracion, double rating, boolean
disponible) {
        this(titulo, director, genero, anio, duracion, rating,
disponible);
        this.idPelicula = idPelicula;
    }

    // Getters y Setters
    public int getIdPelicula() { return idPelicula; }
```

```
    public void setIdPelicula(int idPelicula) { this.idPelicula =
idPelicula; }

    public String getTitulo() { return titulo; }
    public void setTitulo(String titulo) { this.titulo = titulo; }

    public String getDirector() { return director; }
    public void setDirector(String director) { this.director =
director; }

    public String getGenero() { return genero; }
    public void setGenero(String genero) { this.genero = genero; }

    public int getAnio() { return anio; }
    public void setAnio(int anio) { this.anio = anio; }

    public int getDuracion() { return duracion; }
    public void setDuracion(int duracion) { this.duracion = duracion; }

    public double getRating() { return rating; }
    public void setRating(double rating) { this.rating = rating; }

    public boolean isDisponible() { return disponible; }
    public void setDisponible(boolean disponible) { this.disponible =
disponible; }

    @Override
    public String toString() {
        return String.format("%s (%d) - Director: %s | Género: %s |
Duración: %d min | Rating: %.1f | %s",
                titulo, anio, director, genero, duracion, rating,
(disponible ? "Disponible" : "No disponible"));
    }
}
```

PeliculaDAO.java:

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PeliculaDAO {
```

```
    private final String url = "jdbc:sqlite:peliculas.db";

    public PeliculaDAO() {
        crearTabla();
    }

    private void crearTabla() {
        String sql = """
        CREATE TABLE IF NOT EXISTS Peliculas (
            id_pelicula INTEGER PRIMARY KEY AUTOINCREMENT,
            titulo TEXT NOT NULL,
            director TEXT,
            genero TEXT,
            anio INTEGER,
            duracion INTEGER,
            rating REAL,
            disponible INTEGER
        );
        """;
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement()) {
            stmt.execute(sql);
        } catch (SQLException e) {
            System.out.println("Error al crear tabla: " +
e.getMessage());
        }
    }


    public void insertar(Pelicula p) {
        String sql = "INSERT INTO Peliculas (titulo, director, genero,
anio, duracion, rating, disponible) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (Connection conn = DriverManager.getConnection(url);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, p.getTitulo());
            pstmt.setString(2, p.getDirector());
            pstmt.setString(3, p.getGenero());
            pstmt.setInt(4, p.getAnio());
            pstmt.setInt(5, p.getDuracion());
            pstmt.setDouble(6, p.getRating());
            pstmt.setInt(7, p.isDisponible() ? 1 : 0);
            pstmt.executeUpdate();
        } catch (SQLException e) {
```

```
            System.out.println("Error al insertar: " + e.getMessage());
        }
    }

    // Consultas específicas
    public List<Pelicula> listarAlfabeticamente() {
        return obtenerLista("SELECT * FROM Peliculas ORDER BY titulo
ASC");
    }

    public List<Pelicula> buscarPorGenero(String genero) {
        return obtenerLista("SELECT * FROM Peliculas WHERE genero LIKE
'%" + genero + "%'");
    }

    public List<Pelicula> buscarPorDirectorRating(String director) {
        return obtenerLista("SELECT * FROM Peliculas WHERE director
LIKE '%" + director + "%' AND rating >= 7");
    }

    public List<Pelicula> disponiblesDesde2015() {
        return obtenerLista("SELECT * FROM Peliculas WHERE disponible =
1 AND anio > 2015");
    }

    public List<Pelicula> top5Rating() {
        return obtenerLista("SELECT * FROM Peliculas ORDER BY rating
DESC LIMIT 5");
    }

    public List<Pelicula> buscarPorPalabraClave(String palabra) {
        return obtenerLista("SELECT * FROM Peliculas WHERE titulo LIKE
'%" + palabra + "%'");
    }

    public void contarPorGenero() {
        String sql = "SELECT genero, COUNT(*) AS total FROM Peliculas
GROUP BY genero";
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
```

```
                System.out.printf("%s -> %d películas%n",
rs.getString("genero"), rs.getInt("total"));
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public void duracionMediaPorDirector(String director) {
        String sql = "SELECT AVG(duracion) AS media FROM Peliculas
WHERE director LIKE '%" + director + "%'";
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            if (rs.next()) {
                System.out.printf("Duración media de las películas de
%s: %.2f minutos%n", director, rs.getDouble("media"));
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public List<Pelicula> noDisponiblesRatingMayor8() {
        return obtenerLista("SELECT * FROM Peliculas WHERE disponible =
0 AND rating > 8");
    }

    public List<Pelicula> masLargasQueMedia() {
        String sql = """
            SELECT * FROM Peliculas
            WHERE duracion > (SELECT AVG(duracion) FROM Peliculas)
            ORDER BY duracion DESC
        """;
        return obtenerLista(sql);
    }


    private List<Pelicula> obtenerLista(String sql) {
        List<Pelicula> lista = new ArrayList<>();
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
```

```
            while (rs.next()) {
                lista.add(new Pelicula(
                        rs.getInt("id_pelicula"),
                        rs.getString("titulo"),
                        rs.getString("director"),
                        rs.getString("genero"),
                        rs.getInt("anio"),
                        rs.getInt("duracion"),
                        rs.getDouble("rating"),
                        rs.getInt("disponible") == 1
                ));
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
        return lista;
    }
}
```