Pau Casesnoves CIFP

Elias Roig Alcon

Curso 2024-2025

FPGS Desarrollo Multiplataforma

Unidad de Trabajo 5: Estructuras de almacenamiento de la información.

A continuación te muestro la serie de ejercicios prácticos diseñados para reforzar conceptos fundamentales de programación en Java:

- Ejercicio 1:

  Programa que cuenta ocurrencias de una letra en una frase, utilizando métodos estáticos para la entrada de datos y el conteo.

- Ejercicio 2:

  Validación de códigos con formato específico, implementando métodos para verificar estructura y contar estadísticas.

- Ejercicio 3:

  Manejo de matrices y arrays, calculando máximos por fila y mostrando resultados en formato tabulado.

- Ejercicio 4:

  Sistema de VideoClub con array de objetos Película, implementando funcionalidades CRUD y estadísticas de valoración.

Ejercicio 1

```
package cat.paucasesnoves.prog.roig.Bloc04Ejer1;

import java.util.Scanner;

/**
 * Este programa Java solicita al usuario una frase y un carácter, y cuenta
 * cuántas veces aparece dicho carácter en la frase. Incluye validaciones
 * básicas de entrada y está organizado en métodos independientes para cada
 * función.
 *
 * @author Metku - Elias Roig
 */

public class Bloc04Ejer1 {

    private static final Scanner sc = new Scanner(System.in, "ISO-8859-1");
    private static String fraseUser;
    private static char buscarLetra;

    public static void main(String[] args) {

        pedirFrase();
        pedirLetra();
        analizarFrase(fraseUser, buscarLetra);
    }

    public static String pedirFrase() {

        while (true) {
            System.out.println("Introduce la frase que se va a analizar: ");
            fraseUser = sc.nextLine();

            if (fraseUser == null || fraseUser.trim().isEmpty()) {
                System.out.println("Debes escribir una frase completa.");
            } else if (fraseUser.length() > 50) {
                System.out.println("Escribe una frase más corta.");
            } else {
                System.out.println("Frase guardada.");
                break;
            }
        }
        return fraseUser;
    }

    private static char pedirLetra() {

        System.out.println("Introduce la letra que quieres analizar en la frase: (Número,
símbolo o letra)");
        buscarLetra = sc.next().charAt(0);
        System.out.println("Carácter guardado..." + "(" + buscarLetra + ")");
```

```
        return buscarLetra;
    }

    private static void analizarFrase(String frase, char caracter) {

        System.out.println("Buscando...");

        int contador = 0;
        for (int i = 0; i < frase.length(); i++) {
            if (frase.charAt(i) == caracter) {
                contador++;
            }
        }
        if (contador >= 1) {
            System.out.println("Ha sido encontrado " + contador + " veces!");
        } else {
            System.out.println("El carácter " + caracter + " no ha sido encontrado...");
        }
    }
}
```

Pruebas

```
Introduce la frase que se va a analizar:
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.
Escribe una frase m◆s corta. (m◆x 50)
Introduce la frase que se va a analizar:
Lorem ipsum is simply dummy text of...
Frase guardada.
Introduce la letra que quieres analizar en la frase: (N◆mero, s◆mbolo o letra)
x
Car◆cter guardado...(x)
Buscando...
Ha sido encontrado 1 veces!
```

```
Introduce la frase que se va a analizar:
123 123 123 123 123 123 @
Frase guardada.
Introduce la letra que quieres analizar en la frase: (N◆mero, s◆mbolo o letra)
@
Car◆cter guardado...(@)
Buscando...
Ha sido encontrado 1 veces!
```

Ejercicio 2

```
package cat.paucasesnoves.prog.roig.Bloc04Ejer2;

import java.util.Scanner;

/**
 * Este programa Java permite al usuario introducir 5 cadenas con formato
 * "123-LLLL", valida cada entrada, y determina si la parte alfabética empieza
 * por 'A'. Almacena los resultados en arrays y muestra un resumen con los
 * totales.
 *
 * @author Metku - Elias Roig
 */
public class Bloc04Ejer2 {

    private static Scanner sc = new Scanner(System.in, "ISO-8859-1");
    private static String[] arrayCadenas = new String[5];
    private static boolean[] arrayBooleanos = new boolean[5];
    private static int contadorTrue = 0;
    private static int contadorFalse = 0;

    private static boolean validarFormato(String cadena) {
        return cadena.matches("\\d{3}-[a-zA-Z]{4}");
    }

    private static void almacenarArrayCadena(String[] generarArrayCadenas) {
        System.out.println("Introduce 5 cadenas válidas: (123-LLLL)");
        int count = 0;
        while (count < 5) {
            String cadena = sc.nextLine();
            if (validarFormato(cadena)) {
                arrayCadenas[count] = cadena;
                count++;
            } else {
                System.out.println("Debes introducir un formato válido. Prueba otra vez.");
            }
        }
    }

    private static boolean[] generarArrayBooleanos(String[] arrayCadenaB) {
        boolean[] arrayBooleanos = new boolean[arrayCadenaB.length];

        for (int i = 0; i < arrayCadenaB.length; i++) {
            arrayBooleanos[i] = arrayCadenaB[i].split("-")[1].startsWith("A");
            if (arrayBooleanos[i]) {
                contadorTrue++;
            } else {
                contadorFalse++;
            }
        }
        return arrayBooleanos;
```

```
    }

    private static void mostrarArrays(String[] arrayCadenas, boolean[] arrayBoolean) {

        System.out.println("Resultado");
        for (int i = 0; i < arrayCadenas.length; i++) {
            System.out.println(arrayCadenas[i] + "\t\t" + arrayBoolean[i]);
        }
    }

    public static void main(String[] args) {

        almacenarArrayCadena(arrayCadenas);
        arrayBooleanos = generarArrayBooleanos(arrayCadenas);
        mostrarArrays(arrayCadenas, arrayBooleanos);
        System.out.println("Número de true: " + contadorTrue);
        System.out.println("Número de false: " + contadorFalse);
        System.out.println("Programa realizado por Elias Roig.");

    }
}
```

Pruebas

```
Introduce 5 cadenas válidas: (123-LLLL)
123-LLLL
234-AAAA
345-aAaA
456-eAaA
567-@aga
Debes introducir un formato válido. Prueba otra vez.
567-2345
Debes introducir un formato válido. Prueba otra vez.
567-aAaA
Resultado
123-LLLL                false
234-AAAA                true
345-aAaA                false
456-eAaA                false
567-aAaA                false
Número de true: 1
Número de false: 4
Programa realizado por Elias Roig.
```

Ejercicio 3

```
package cat.paucasesnoves.prog.roig;

import java.util.Scanner;

/**
 * Este programa crea una matriz 3x2 de números decimales con valores
 * introducidos por el usuario. Muestra una tabla con los valores y el máximo de
 * cada fila.
 *
 * @author Metku
 */

public class Bloc04Ejer3 {

    public static void main(String[] args) {
        // Crear la matriz
        double[][] matriz = new double[3][2];
        Scanner sc = new Scanner(System.in);

        // Rellenar la matriz
        for (int i = 0; i < matriz.length; i++) {
            for (int j = 0; j < matriz[i].length; j++) {
                boolean valorValido = false;

                while (!valorValido) {
                    try {
                        System.out.println("Introduce los valores de la matriz [" + i + "][" + j + "]");
                        matriz[i][j] = sc.nextDouble();
                        valorValido = true;
                    } catch (Exception e) {
                        System.out.println("Error: Por favor, introduce un valor numérico válido.");
                        sc.nextLine(); // Limpiar el buffer
                    }
                }
            }
        }

        // Imprimir encabezado
        System.out.printf("%10s %10s %10s%n", "columna1", "columna2", "máximo");
        System.out.println();

        // Imprimir matriz con formato
        for (int i = 0; i < matriz.length; i++) {
            System.out.printf("fila%-5d", (i + 1));
            double maxFila = matriz[i][0];

            for (int j = 0; j < matriz[i].length; j++) {
                System.out.printf("%10.1f", matriz[i][j]);
                if (matriz[i][j] > maxFila) {
                    maxFila = matriz[i][j];
```

```
            }
        }
        System.out.printf("%15.0f%n", maxFila);
        System.out.println();
    }
  }
}
```

Pruebas

```
Introduce los valores de la matriz [0][0]
1
Introduce los valores de la matriz [0][1]
0
Introduce los valores de la matriz [1][0]
1.2
Introduce los valores de la matriz [1][1]
a
Error: Por favor, introduce un valor num�rico v�lido.
Introduce los valores de la matriz [1][1]
@
Error: Por favor, introduce un valor num�rico v�lido.
Introduce los valores de la matriz [1][1]
A
Error: Por favor, introduce un valor num�rico v�lido.
Introduce los valores de la matriz [1][1]
20
Introduce los valores de la matriz [2][0]
20.22
Introduce los valores de la matriz [2][1]
10
  columna1    columna2     m�ximo

fila1           1.0         0.0              1

fila2           1.2        20.0             20

fila3          20.2        10.0             20
```

Ejercicio 4 - VideoClub Mejorado

Clase VideoClub

```
package roig.videoclub;

import java.util.Scanner;
import roig.utilities.Utility;
import static roig.videoclub.controller.MenuController.createDirector;
import static roig.videoclub.controller.MenuController.createFilm;
import static roig.videoclub.controller.MenuController.increaseDirectorAwards;
import static roig.videoclub.controller.MenuController.modifyFilmSpectatorRating;
import static roig.videoclub.controller.MenuController.showBestRatingFilms;
import static roig.videoclub.controller.MenuController.showDirectorData;
import static roig.videoclub.controller.MenuController.showFilmClassification;
import static roig.videoclub.controller.MenuController.showFilmData;

/**
 * La clase VideoClub actúa como el punto de entrada de la aplicación,
 * gestionando un menú interactivo que permite a los usuarios realizar acciones
 * relacionadas con directores y películas, como crear registros, mostrar
 * información, modificar datos y clasificar películas. Utiliza un bucle
 * do-while para mantener la interacción hasta que el usuario decida salir.
 *
 * @author Metku - Elias Roig
 */
public class VideoClub {

    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        menu();
    }

    public static void menu() {
        int option = 0;

        do {
            System.out.println("=== Choose an option: ==="
                    + "\n1. Create a Director"
                    + "\n2. Show Director data"
                    + "\n3. Increase director's awards"
                    + "\n4. Create a film"
                    + "\n5. Show film data"
                    + "\n6. Modify film spectator rating"
                    + "\n7. Show film classification"
                    + "\n8. Show best films."
                    + "\n9. Exit");
            System.out.print("=== Option desired: ");
```

```
            option = Utility.validateInt();

            switch (option) {
                case 1 ->
                    createDirector();
                case 2 ->
                    showDirectorData();
                case 3 ->
                    increaseDirectorAwards();
                case 4 ->
                    createFilm();
                case 5 ->
                    showFilmData();
                case 6 ->
                    modifyFilmSpectatorRating();
//                      MenuController.testFilms();
                case 7 ->
                    showFilmClassification();
                case 8 ->
                    showBestRatingFilms();
                case 9 -> {
                    System.out.println("\n=====================================");
                    System.out.println("Thank you for using the app. Farewell!");
                    System.out.println("=====================================\n");
                    System.exit(0);
                }
                default ->
                    throw new AssertionError();
            }
        } while (option != 0);
    }

}
```

Clases Model

Clase Director

```
package roig.videoclub.model;

import java.time.LocalDate;

/**
 * La clase Director define las propiedades y comportamientos de una director en el
 * sistema del videoclub.
 *
 * @author Metku
 */
public class Director {
```

```
    String directorName;
    LocalDate birthDate;
    int awardsNum;
    int lastFilmDirectedYear;

    public Director() {
    }

    public Director(String directorName, LocalDate birthDate, int awardsNum, int
lastFilmDirectedYear) {
        this.directorName = directorName;
        this.birthDate = birthDate;
        this.awardsNum = awardsNum;
        this.lastFilmDirectedYear = lastFilmDirectedYear;
    }

    public String getDirectorName() {
        return directorName;
    }

    public void setDirectorName(String directorName) {
        this.directorName = directorName;
    }

    public LocalDate getBirthDate() {
        return birthDate;
    }

    public void setBirthDate(LocalDate birthDate) {
        this.birthDate = birthDate;
    }

    public int getAwardsNum() {
        return awardsNum;
    }

    public void setAwardsNum(int awardsNum) {
        this.awardsNum = awardsNum;
    }

    public int getLastFilmDirected() {
        return lastFilmDirectedYear;
    }

    public void setLastFilmDirected(int lastFilmDirectedYear) {
        this.lastFilmDirectedYear = lastFilmDirectedYear;
    }

    @Override
    public String toString() {
        return "Director's data: \n"
                + "Name: " + this.directorName + ".\n"
```

```
                + "Birthdate: " + this.birthDate + ".\n"
                + "Awards recieved: " + this.awardsNum + ".\n"
                + "Last film directed year: " + this.lastFilmDirectedYear + ".\n";
    }
}
```

Clase Film

```
package roig.videoclub.model;

/**
 * La clase Film define las propiedades y comportamientos de una película en el
 * sistema del videoclub.
 *
 * @author Metku - Elias Roig
 */
public class Film {

    String title;
    int minutes;
    int spectatorsNum;
    double spectatorValoration;
    boolean allPublicFilm;
    String directorFilmName;
    private static int filmCounter = 0;
    String specRatingString;

    public Film() {

    }

    public Film(String title) {
        this.title = title;
        this.specRatingString = title;

    }

    public Film(String title, int minutes, int spectatorsNum, double spectatorValoration,
boolean allPublicFilm, String specRatingString, String directorFilmName) {
        this.title = title;
        this.minutes = minutes;
        this.spectatorsNum = spectatorsNum;
        this.spectatorValoration = spectatorValoration;
        this.allPublicFilm = allPublicFilm;
        this.directorFilmName = directorFilmName;
        this.specRatingString = specRatingString;
        filmCounter++;
    }

    public String getSpecRatingString() {
```

```
    return specRatingString;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public int getMinutes() {
    return minutes;
}

public void setMinutes(int minutes) {
    this.minutes = minutes;
}

public int getSpectatorsNum() {
    return spectatorsNum;
}

public void setSpectatorsNum(int spectatorsNum) {
    this.spectatorsNum = spectatorsNum;
}

public double getSpectatorValoration() {
    return spectatorValoration;
}

public void setSpectatorValoration(double spectatorValoration) {
    this.spectatorValoration = spectatorValoration;
}

public boolean isAllPublicFilm() {
    return allPublicFilm;
}

public void setAllPublicFilm(boolean allPublicFilm) {
    this.allPublicFilm = allPublicFilm;
}

public String getDirectorFilmName() {
    return directorFilmName;
}

public void setDirectorFilmName(String directorFilmName) {
    this.directorFilmName = directorFilmName;
}

public static int getFilmCounter() {
    return filmCounter;
```

```
    }

    public static void setFilmCounter(int filmCounter) {
        Film.filmCounter = filmCounter;
    }

    @Override
    public String toString() {
        return "\nTitle: " + this.getTitle()
                + "\nDuration: " + this.getMinutes()
                + "\nSpectator Number: " + this.getSpectatorsNum()
                + "\nSpectator Film Valoration: " + this.getSpectatorValoration()
                + "\nCalculated rating: " + this.specRatingString
                + "\nSuitable for all audience: " + this.isAllPublicFilm()
                + "\nDirector's name: " + this.getDirectorFilmName()
                + "\nTotal films in the Videoclub: " + filmCounter;
    }
}
```

Clases Controller

Clase ArrayPeliculas

```
package roig.videoclub.controller;

import roig.videoclub.model.Film;

/**
 * Esta práctica de Java implementa un sistema de gestión de películas que
 * almacena películas en un array de tamaño fijo, muestra datos de películas,
 * busca por título o clasificación, y agrega nuevas películas. La clase
 * ArrayPeliculas maneja estas operaciones con validaciones básicas.
 *
 * @author Metku - Elias Roig
 */
public class ArrayPeliculas {

    private Film[] films;
    private int numFilms;

    // Constructor
    public ArrayPeliculas(int maxFilms) {
        films = new Film[maxFilms];
        numFilms = 0;
    }

    public int allFilmData() {
        if (numFilms == 0) {
            System.out.println("\n=====================================");
```

```
        System.out.println("Error: There's no film data.");
        System.out.println("=====================================\n");
    } else {
        for (int i = 0; i < numFilms; i++) {
            System.out.println("=====================================\n");
            System.out.println(films[i]);
            System.out.println("\n=====================================\n");
        }
    }
    return numFilms;
}

public Film[] getFilms(String title) {
    // Count matching films first
    int count = 0;
    for (int i = 0; i < numFilms; i++) {
        if (films[i].getTitle().toLowerCase().matches(title.toLowerCase())) {
            count++;
        }
    }

    // Create array with exact size
    Film[] matches = new Film[count];
    int index = 0;

    // Fill array with matching films doing a copy of it
    for (int i = 0; i < numFilms; i++) {
        if (films[i].getTitle().toLowerCase().matches(title.toLowerCase())) {
            matches[index++] = films[i];

        }
    }
    System.out.println("Films found: " + count);
    return matches;
}

public int getNumFilms() {
    return numFilms;
}

public boolean addPelicula(Film nuevo) {
    if (numFilms < films.length) {
        films[numFilms] = nuevo;
        numFilms++;
        return true;
    } else {
        System.out.println("Error: Film array is full.");
        return false;
    }
}

public Film[] getRating(String rating) {
    System.out.println("Searching for films with rating: " + rating);
```

```
        int count = 0;

        // Contar coincidencias
        for (int i = 0; i < numFilms; i++) {
            if (films[i].getSpecRatingString().equalsIgnoreCase(rating)) {
                System.out.println("=====================================");
                System.out.println("\nMatch found: " + films[i].toString()); // Depuración
                System.out.println("\n=====================================\n");
                count++;
            }
        }

        if (count == 0) {
            System.out.println("=====================================");
            System.out.println("\nNo matches found for rating: " + rating);
            System.out.println("=====================================\n");
        }

        // Crear un arreglo de coincidencias
        Film[] matches = new Film[count];
        int index = 0;

        for (int i = 0; i < numFilms; i++) {
            if (films[i].getSpecRatingString().equalsIgnoreCase(rating)) {
                matches[index++] = films[i];
            }
        }
        System.out.println("Films found with this title: " + count);
        System.out.println("=====================================\n");
        return matches;
    }

    public void printAllFilms() {
        if (numFilms == 0) {
            System.out.println("No films are currently stored.");
        } else {
            System.out.println("=== Films in the system ===");
            for (int i = 0; i < numFilms; i++) {
                System.out.println(films[i].toString());
            }
        }
    }

}
```

Clase MenuController

```
package roig.videoclub.controller;

import java.time.LocalDate;
```

```
import java.util.Scanner;
import roig.videoclub.model.Director;
import roig.utilities.Utility;
import roig.videoclub.model.Film;

/**
 * La clase MenuController gestiona la lógica de las opciones del menú principal
 * de la aplicación de videoclub. Permite crear y manipular objetos Director y
 * Film, mostrando sus datos, modificando atributos como premios o valoración de
 * espectadores, y generando reportes de clasificación. Utiliza validaciones de
 * entrada a través de métodos auxiliares y controla errores para garantizar
 * datos consistentes.
 *
 * @author Metku - Elias Roig
 */
public class MenuController {

    private static Scanner sc = new Scanner(System.in);
    private static Director director = null;
    private static Film film = null;

    private static final int maxFilms = 40;
    private static ArrayPeliculas arrayPeliculas = new ArrayPeliculas(maxFilms);

    public static void createDirector() {
        try {
            String directorName = Utility.validateDirectorName();
            LocalDate dirBirthDate = Utility.validateDirBirthDate();
            int dirAwardsNum = Utility.validateDirAwardsNum();
            int dirLastFilmDirected = Utility.validateDirLastFilmDirected();

            director = new Director(directorName, dirBirthDate, dirAwardsNum,
dirLastFilmDirected);
            System.out.println("[[=== Director created succesfully ===]]\n");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void showDirectorData() {
        if (director != null) {
            System.out.println("\n=== All director data ===");
            System.out.println(director.toString() + "Director age: " +
Utility.directorAge(director) + " years old.");
            System.out.println("========================\n");
        } else {
            System.out.println("\n================================");
            System.out.println("There's no Director created yet.");
            System.out.println("================================\n");
        }
    }

    public static void increaseDirectorAwards() {
```

```
        int increase = 0;

        if (director == null) {
            System.out.println("\n==================================");
            System.out.println("There's no Director created yet.");
            System.out.println("==================================\n");
            return;
        }


System.out.println("\n=====================================================
");
        System.out.print("[=] Do you want to substract awards? (y/n) : ");
        String substract = sc.nextLine().trim().toLowerCase();

System.out.println("=====================================================");

        while (!substract.equals("y") && !substract.equals("n")) {
            System.out.println("Error: Please, introduce 'y' or 'n' to continue.");
            substract = sc.nextLine().trim().toLowerCase();
        }

        if (director != null && substract.equals("n")) {
            increase = Utility.validateDirAwardsNum();
            director.setAwardsNum(director.getAwardsNum() + increase);
            System.out.println("Succes: [ Adding ] new director's awards amount: " +
director.getAwardsNum() + "\n");
        } else if (director != null && substract.equals("y")) {
            increase = Utility.validateDirAwardsNum();
            director.setAwardsNum(director.getAwardsNum() - increase);
            System.out.println("Succes: [ Substracting ] new director's awards amount: " +
director.getAwardsNum() + "\n");
        }
    }

    public static void createFilm() {
        try {
            String filmName = Utility.validateFilmName();
            int filmMinutes = Utility.duration();
            int spectatorNum = Utility.spectators();
            double specValoration = Utility.filmValoration();
            boolean suitableForAll = Utility.suitableForAll();
            String spectRatingString = Utility.specRatingString();
            String directorFilmName = Utility.validateDirectorName();

            film = new Film(filmName, filmMinutes, spectatorNum, specValoration,
suitableForAll, spectRatingString, directorFilmName);
            System.out.println("\n[[=== Film created successfully ===]]\n");

            boolean added = arrayPeliculas.addPelicula(film);
            if (!added) {
                System.out.println("Error: Could not add film, array might be full.");
            }
```

```
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }
}

public static void showFilmData() {

    arrayPeliculas.allFilmData();
}

public static void modifyFilmSpectatorRating() {
    if (film == null) {
        System.out.println("\n=====================================");
        System.out.println("Error: There's no film data.");
        System.out.println("=====================================\n");
        return;
    }
    System.out.println("Insert the title you are looking for: ");
    String input = sc.nextLine();

    Film[] matchingFilms = arrayPeliculas.getFilms(input);

    if (matchingFilms.length == 0) {
        System.out.println("\n=====================================");
        System.out.println("Error: No films found with that title");
        System.out.println("=====================================\n");
        return;
    }

    film = matchingFilms[0]; // Tomamos la primera coincidencia
    double modifyRating = Utility.validateSpecRating();
    film.setSpectatorValoration(modifyRating);

    System.out.println("Success: New film rating for '" + film.getTitle() + "': "
            + film.getSpectatorValoration() + "\n");
}

public static void showFilmClassification() {
    if (film == null) {
        System.out.println("\n=====================================");
        System.out.println("Error: There's no film data.");
        System.out.println("=====================================\n");
        return;
    }

    System.out.println("Insert the title you are looking for: ");
    String input = sc.nextLine();

    Film[] matchingFilms = arrayPeliculas.getFilms(input);

    if (matchingFilms.length == 0) {
        System.out.println("\n=====================================");
        System.out.println("Error: No films found with that title");
```

```
            System.out.println("=====================================\n");
            return;
        } else {
            Utility.generateReport(film, director);
        }

        film = matchingFilms[0];

    }

    public static void showBestRatingFilms() {
        if (film == null) {
            System.out.println("\n=====================================");
            System.out.println("Error: There's no film data.");
            System.out.println("=====================================\n");
            return;
        }

        System.out.println("\n=================="
                + "\nWhat rating are you looking for?"
                + "\n1. Excellent"
                + "\n2. Good"
                + "\n3. Not recommended"
                + "\n4. Unknown"
                + "\n==================");

        while (true) {
            System.out.println("Choose one option: (1-4)");

            String input = sc.nextLine(); // Leer la entrada del usuario como cadena

            try {
                int option = Integer.parseInt(input); // Intentar convertir la entrada a entero

                if (option < 1 || option > 4) {
                    System.out.println("Error: Input must be between 1 and 4. Please try again.");
                } else {
                    // Procesar la opción válida
                    String rating = switch (option) {
                        case 1 ->
                            "Excellent";
                        case 2 ->
                            "Good";
                        case 3 ->
                            "Not recommended";
                        case 4 ->
                            "Unknown";
                        default ->
                            null; // Esto nunca ocurrirá
                    };

                    Film[] matchingRatingFilms = arrayPeliculas.getRating(rating);
```

```
                if (matchingRatingFilms.length == 0) {
                    System.out.println("\n=====================================");
                    System.out.println("Error: No films found with that rating.");
                    System.out.println("=====================================\n");
                } else {
                    System.out.println("\n=====================================");
                    System.out.println("Films with rating '" + rating + "':");
                    for (Film f : matchingRatingFilms) {
                        System.out.println(f); // Asumimos que `Film` tiene un `toString()` bien
definido
                    }
                    System.out.println("=====================================\n");
                }
                break; // Salir del bucle tras procesar una entrada válida
            }
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid input. Please enter a number between 1 and
4.");
        }
    }
}

public static void testFilms() {
    arrayPeliculas.addPelicula(new Film("Film A", 120, 600000, 9.0, true, "Excellent",
"Director A"));
    arrayPeliculas.addPelicula(new Film("Film B", 100, 30000, 6.5, true, "Good",
"Director B"));
    arrayPeliculas.addPelicula(new Film("Film C", 90, 15000, 4.0, false, "Not
recommended", "Director C"));
    arrayPeliculas.addPelicula(new Film("Film D", 80, 5000, 7.0, true, "Unknown",
"Director D"));

    System.out.println("=== Running test ===");
    arrayPeliculas.printAllFilms();

    Film[] matches = arrayPeliculas.getRating("Excellent");
    System.out.println("Matches for 'Excellent': " + matches.length);
}

}
```

Clase Utility

```
package roig.utilities;

import java.time.LocalDate;
```

```
import java.time.LocalDateTime;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.InputMismatchException;
import java.util.Scanner;
import roig.videoclub.model.Director;
import roig.videoclub.model.Film;

/**
 * La clase Utility proporciona métodos para validar y obtener datos
 * relacionados con directores y películas, como nombres, fechas, premios,
 * duraciones y espectadores, garantizando que los datos ingresados cumplan con
 * formatos y rangos adecuados antes de ser utilizados en la aplicación.
 *
 * @author Metku - Elias Roig
 */
public class Utility {

    private static final Scanner sc = new Scanner(System.in);
    private static String specRating;
    private static double filmRating;
    private static int totalAwards = 0;

    public static int validateInt() {

        while (true) {
            try {
                int option = Integer.parseInt(sc.nextLine());
                if (option < 1 || option > 9) {
                    System.out.println("""
                                    ====================================
                                       Option must be between 1 and 9
                                    ====================================""");
                    System.out.print("Try again: ");
                } else {
                    return option;
                }
            } catch (NumberFormatException e) {
                System.out.print("Error: Please enter a valid integer. Try again:");
            }
        }
    }

    public static String validateDirectorName() {
        System.out.println("\n=== Set Director's Name ===");

        while (true) {
            System.out.print("Enter director's name: ");
            String directorName = sc.nextLine().trim();

            if (directorName.isEmpty() || directorName.isBlank()) {
                System.out.println("Error: Name cannot be empty.");
```

```
        } else if (!directorName.matches("[a-zA-Z ]+")) {
            System.out.println("Error: Name must contain only letters(50) and spaces.");
        } else if (directorName.split("\\s+").length != 2) {
            System.out.println("Error: Please provide both first name and last name.");
        } else if (directorName.length() > 50) {
            System.out.println("Error: Name length must not exceed 50 characters.");
        } else {
            System.out.println("Success: Director's name set.");
            return directorName;
        }
    }
}

public static LocalDate validateDirBirthDate() {
    System.out.println("\n=== Set Director's Birthdate ===");
    System.out.println("Format: dd/MM/yyyy");

    LocalDate dirBirthDate = null;
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");

    while (true) {
        System.out.print("Enter birthdate: ");
        String input = sc.nextLine().trim();

        try {
            dirBirthDate = LocalDate.parse(input, formatter);
            System.out.println("Success: Birthdate set to " + dirBirthDate.format(formatter));
            break;
        } catch (DateTimeParseException e) {
            System.out.println("Error: Invalid date format. Please use dd/MM/yyyy.");
        }
    }
    return dirBirthDate;
}

public static int validateDirAwardsNum() {
    System.out.println("\n=== Set Director's Awards ===");

    while (true) {
        try {
            System.out.print("Enter number of awards: ");
            int dirAwardsNum = Integer.parseInt(sc.nextLine());
            if (dirAwardsNum < 0) {
                System.out.println("Error: Awards must be 0 or a positive integer.");
            } else {
                return dirAwardsNum;
            }
        } catch (NumberFormatException e) {
            System.out.println("Error: Please enter a valid integer.");
        }
    }
}
```

```
    public static int substractAwards() {
        System.out.println("\n=== Set Director's Awards ===");

        while (true) {
            try {
                System.out.print("Enter number of awards: ");
                int dirAwardsNum = Integer.parseInt(sc.nextLine());
                if (dirAwardsNum < 0) {
                    System.out.println("Error: Awards must be 0 or a positive integer.");
                } else {
                    System.out.println("Success: Awards set to " + dirAwardsNum + "\n");
                    return dirAwardsNum;
                }
            } catch (NumberFormatException e) {
                System.out.println("Error: Please enter a valid integer.");
            }
        }
    }

    public static int validateDirLastFilmDirected() {
        System.out.println("\n=== Set the last film directed year ===");
        int dirLastFilmDirected = 0;
        LocalDateTime currentYear = LocalDateTime.now();

        while (true) {
            try {
                System.out.print("Enter director last film year: ");
                dirLastFilmDirected = Integer.parseInt(sc.nextLine());
                if (dirLastFilmDirected >= 1895 && dirLastFilmDirected <=
currentYear.getYear()) {
                    System.out.println("Success: setting last film directed year...\n");
                    break;
                } else {
                    System.out.println("Error: Introduce a realistic year. Try again...");
                }
            } catch (Exception e) {
                System.out.println(e.getMessage() + " must be an integer.");
            }
        }
        return dirLastFilmDirected;
    }

    public static int directorAge(Director director) {
        if (director != null) {
            LocalDate actual = LocalDate.now();
            LocalDate BirthDate = director.getBirthDate();
            return Period.between(BirthDate, actual).getYears();
        } else {
            System.out.println("No Director created...");
            return -1;
        }
    }
```

```
    public static String validateFilmName() {

        System.out.println("\n=== Set Film name ===");

        while (true) {
            System.out.print("Enter film name: ");
            String filmName = sc.nextLine().trim();

            if (filmName.length() > 0 && filmName.length() <= 100) {
                System.out.println("Succes: setting film name correctly...\n");
                return filmName;
            } else if (filmName.trim().split(" ").length != 2) {
                System.out.println("Error: please enter both first name and last name.");
            } else {
                System.out.println("Error: the name should'tn be larger than 100 characters.");
            }
        }
    }

    public static int duration() {
        System.out.println("=== Introduce Film duration in minutes ===");

        while (true) {
            try {
                System.out.print("Enter film duration: ");
                int duration = Integer.parseInt(sc.nextLine());
                if (duration > 0 && duration < 320) {
                    System.out.println("Succes: setting film duration...\n");
                    return duration;
                } else {
                    System.out.println("Error: duration can't be negative or more than 320
minutes.");
                }
            } catch (Exception e) {
                System.out.println(e.getMessage() + " must be an integer.");
            }
        }
    }

    private static int spectatorsNum;

    public static int spectators() {
        System.out.println("""
                           === Introduce Film spectators number ===
                           ----------------------------------------
                           [ 10 thousand spectators can't rate a film. ]
                           [ 500 thousand spectators can get an exellent rating. ]
                           ----------------------------------------""");

        while (true) {
            try {
                System.out.print("Enter number of viewers: ");
                spectatorsNum = Integer.parseInt(sc.nextLine());
```

```
            if (spectatorsNum < 0) {
                System.out.println("Error: must be 0 or positive!");
            } else {
                System.out.println("Succes: setting spectators!\n");
                break;
            }
        } catch (NumberFormatException e) {
            System.out.println(e.getMessage() + " Spectators can't be cut in half.");
        }
    }
    return spectatorsNum;
}

public static double filmValoration() {
    System.out.println("=== Set the film valoration ===");

    while (true) {
        try {
            System.out.print("Enter the film rating (0.0 to 10.0): ");
            filmRating = sc.nextDouble();

            if (filmRating < 0.0 || filmRating > 10.0) {
                System.out.println("Error: rating must be between 0.0 and 10.0. Try again.");
                continue; // Volver al inicio del bucle
            }

            specRating = calculateRating(filmRating, spectatorsNum);

            System.out.println("This is the current rating of the film: " + filmRating
                    + ". \nReviews from moviegoers told: " + spectatorsNum + ".");

            break;

        } catch (InputMismatchException e) {
            System.out.println("Invalid input. Please enter a valid number. Error: " +
e.getMessage());
            sc.next(); // Limpiar la entrada no válida
        }
    }
    sc.nextLine();
    return filmRating;
}

private static String calculateRating(double filmRating, int spectatorsNum) {
    // Verificación inicial de número de espectadores
    if (spectatorsNum < 10000) {
        return "Unknown";
    }

    // Películas con alta audiencia y excelente valoración
    if (spectatorsNum >= 500000 && filmRating >= 8.0) {
        return "Excellent";
    }
```

```
        // Películas con audiencia media
        if (spectatorsNum >= 10000 && spectatorsNum < 500000) {
            if (filmRating >= 5.5) {
                return "Good";
            }
            return "Not recommended";
        }
        return "Unknown";
    }

    public static String specRatingString() {
        return specRating;
    }

    public static boolean suitableForAll() {


System.out.println("\n=====================================================
");
        System.out.print("[=] Is the film suitable for all public? (y/n) : ");
        String suitable = sc.nextLine().trim().toLowerCase();

System.out.println("=====================================================");

        while (!suitable.equals("y") && !suitable.equals("n")) {
            System.out.println("Error: Please, introduce 'y' or 'n' to continue.");
            suitable = sc.nextLine().trim().toLowerCase();
        }
        return suitable.equals("y");

    }

    public static double validateSpecRating() {

        System.out.println("\n=== Set up new spectator rating ===");
        double specRating = 0.0;

        while (true) {
            try {
                System.out.print("Enter viewer rating: ");
                specRating = Double.parseDouble(sc.nextLine());
                if (specRating < 0.0 || specRating > 10.0) {
                    System.out.println("Error: Must be positive or not higher than 10!");
                } else {
                    System.out.println("Succes: Setting new spectator rating!");
                    break;
                }
            } catch (Exception e) {
                System.out.println(e.getMessage() + " must be a double.");
            }
        }
        return specRating;
```

```
    }

    public static void generateReport(Film film, Director director) {
        System.out.println("\n=== Movie Classification Report ===");

        if (film == null) {
            System.out.println("Error: No movie has been created.");
            return;
        }
        if (director == null) {
            System.out.println("Error: No director has been created.");
            return;
        }

        if (!film.getDirectorFilmName().equals(director.getDirectorName())) {
            System.out.println("\nError: Director's name doesn't match in both objects.\n");
        } else {
            System.out.println("Succes: Director's name matches.");
        }

        String userName = "Elias Roig Alcon";
        String corporativeEmail = "eliasroig@paucasesnovescifp.cat";

        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
        String timestamp = now.format(formatter);

        int directorAge = Utility.directorAge(director);

        StringBuilder report = new StringBuilder();
        report.append("\nCalculation date and time: ").append(timestamp).append("\n");
        report.append("User: ").append(userName).append(" | Email:
").append(corporativeEmail).append("\n");
        report.append("\n--- Movie Details ---\n");
        report.append("Title: ").append(film.getTitle()).append("\n");
        report.append("Duration: ").append(film.getMinutes()).append(" minutes\n");
        report.append("Director: ").append(director.getDirectorName()).append(" (Age:
").append(directorAge).append(")\n");
        report.append("Classification: ").append(specRatingString()).append("\n");
        report.append("Viewers: ").append(film.getSpectatorsNum()).append("\n");
        report.append("Viewer rating: ").append(film.getSpectatorValoration()).append("\n");
        report.append("---------------------\n");

        System.out.println(report.toString());
    }
}
```

Pruebas

1. Primero las pruebas a los elementos que están vacíos para comprobar las excepciones. (2, 3, 5, 6, 7, 8)

```
=== Choose an option: ===
1. Create a Director
2. Show Director data
3. Increase director's awards
4. Create a film
5. Show film data
6. Modify film spectator rating
7. Show film classification
8. Show best films.
9. Exit
=== Option desired: 2

================================
There's no Director created yet.
================================
```

```
=== Choose an option: ===
1. Create a Director
2. Show Director data
3. Increase director's awards
4. Create a film
5. Show film data
6. Modify film spectator rating
7. Show film classification
8. Show best films.
9. Exit
=== Option desired: 3

================================
There's no Director created yet.
================================
```

```
=== Choose an option: ===
1. Create a Director
2. Show Director data
3. Increase director's awards
4. Create a film
5. Show film data
6. Modify film spectator rating
7. Show film classification
8. Show best films.
9. Exit
=== Option desired: 5

====================================
Error: There's no film data.
====================================
```

```
=== Choose an option: ===
1. Create a Director
2. Show Director data
3. Increase director's awards
4. Create a film
5. Show film data
6. Modify film spectator rating
7. Show film classification
8. Show best films.
9. Exit
=== Option desired: 6

====================================
Error: There's no film data.
====================================
```

```
=== Choose an option: ===
1. Create a Director
2. Show Director data
3. Increase director's awards
4. Create a film
5. Show film data
6. Modify film spectator rating
7. Show film classification
8. Show best films.
9. Exit
=== Option desired: 7

====================================
Error: There's no film data.
====================================
```

```
=== Choose an option: ===
1. Create a Director
2. Show Director data
3. Increase director's awards
4. Create a film
5. Show film data
6. Modify film spectator rating
7. Show film classification
8. Show best films.
9. Exit
=== Option desired: 8

====================================
Error: There's no film data.
====================================
```

2. Comprobación de que el menú funcione y solo se puedan ingresar números.

```
=== Choose an option: ===
1. Create a Director
2. Show Director data
3. Increase director's awards
4. Create a film
5. Show film data
6. Modify film spectator rating
7. Show film classification
8. Show best films.
9. Exit
=== Option desired: 11
=======================================
    Option must be between 1 and 9
=======================================
Try again: a
Error: Please enter a valid integer. Try again:1
```

3. Pruebas unitarias de la creación de las clases modelos. (1, 4)

```
=== Set Director's Name ===
Enter director's name: Eli4s ROig
Error: Name must contain only letters(50) and spaces.
Enter director's name: Elias
Error: Please provide both first name and last name.
Enter director's name: Elias Roig
Success: Director's name set.
```

```
=== Set Director's Birthdate ===
Format: dd/MM/yyyy
Enter birthdate: 24-10-1995
Error: Invalid date format. Please use dd/MM/yyyy.
Enter birthdate: ab/ab/abcd
Error: Invalid date format. Please use dd/MM/yyyy.
Enter birthdate: 24/10/1995
Success: Birthdate set to 24/10/1995
```

```
=== Set Director's Awards ===
Enter number of awards: ab
Error: Please enter a valid integer.
Enter number of awards: 2@
Error: Please enter a valid integer.
Enter number of awards: 20.2
Error: Please enter a valid integer.
Enter number of awards: 20
```

```
=== Set the last film directed year ===
Enter director last film year: abcd
For input string: "abcd" must be an integer.
Enter director last film year: 1000
Error: Introduce a realistic year. Try again...
Enter director last film year: 3000
Error: Introduce a realistic year. Try again...
Enter director last film year: 2004
Success: setting last film directed year...

[[=== Director created succesfully ===]]
```

```
=== Set Film name ===
Enter film name: Video Club de Elias
Succes: setting film name correctly...

=== Introduce Film duration in minutes ===
Enter film duration: ab
For input string: "ab" must be an integer.
Enter film duration: 120.2
For input string: "120.2" must be an integer.
Enter film duration: 120
Succes: setting film duration...
```

```
=== Introduce Film spectators number ===
---------------------------------------
[ 10 thousand spectators can't rate a film. ]
[ 500 thousand spectators can get an exellent rating. ]
---------------------------------------
Enter number of viewers: abcde
For input string: "abcde" Spectators can't be cut in half.
Enter number of viewers: 120000
Succes: setting spectators!
```

```
=== Set the film valoration ===
Enter the film rating (0.0 to 10.0): n
Invalid input. Please enter a valid number. Error: null
Enter the film rating (0.0 to 10.0): @
Invalid input. Please enter a valid number. Error: null
Enter the film rating (0.0 to 10.0): 10
This is the current rating of the film: 10.0.
Reviews from moviegoers told: 120000.
```

```
================================================
[=] Is the film suitable for all public? (y/n) : l
================================================
Error: Please, introduce 'y' or 'n' to continue.
n
```

4. Procedemos con las pruebas unitarias de los métodos 2, 3, 5, 6, 7, 8.

```
=== Option desired: 2

=== All director data ===
Director's data:
Name: Elias Roig.
Birthdate: 1995-10-24.
Awards recieved: 20.
Last film directed year: 2004.
Director age: 29 years old.
=========================
```

```
=== Option desired: 3

================================================
[=] Do you want to substract awards? (y/n) : n
================================================

=== Set Director's Awards ===
Enter number of awards: 120
Succes: [ Adding ] new director's awards amount: 140
```

```
=== Option desired: 5
====================================


Title: VideoClub de Elias
Duration: 120
Spectator Number: 10
Spectator Film Valoration: 10.0
Suitable for all audience: false
Director's name: eli li
Total films in the Videoclub: 3

====================================

====================================


Title: Video Club de Elias
Duration: 120
Spectator Number: 120000
Spectator Film Valoration: 10.0
Suitable for all audience: false
Director's name: Elias Roig
Total films in the Videoclub: 3

====================================

====================================


Title: ELOL
Duration: 10
Spectator Number: 10
Spectator Film Valoration: 10.0
Suitable for all audience: false
Director's name: Elias Roig
Total films in the Videoclub: 3

====================================
```

```
=== Option desired: 6
Insert the title you are looking for:
video club de elias
Films found: 1

=== Set up new spectator rating ===
Enter viewer rating: abc
For input string: "abc" must be a double.
Enter viewer rating: 204000
Error: Must be positive or not higher than 10!
Enter viewer rating: 9
Succes: Setting new spectator rating!
Success: New film rating for 'Video Club de Elias': 9.0
```

```
=== Option desired: 7
Insert the title you are looking for:
video 123
Films found: 0


====================================
Error: No films found with that title
====================================
```

```
=== Option desired: 7
Insert the title you are looking for:
video club de elias
Films found: 1

=== Movie Classification Report ===
Succes: Director's name matches.

Calculation date and time: 2025-01-26 17:56:42
User: Elias Roig Alcon | Email: eliasroig@paucasesnovescifp.cat

--- Movie Details ---
Title: Video Club de Elias
Duration: 120 minutes
Director: Elias Roig (Age: 29)
Classification: Unknown
Viewers: 120000
Viewer rating: 9.0
----------------------
```

```
=== Option desired: 8

=================
What rating are you looking for?
1. Excellent
2. Good
3. Not recommended
4. Unknown
=================
Choose one option: (1-4)
1

=====================================
Success: processing films with the selected rating...
Searching for films with rating: Excellent
=====================================
No matches found for rating: Excellent
Films found with this title: 0
=====================================
```

```
4. Unknown
================
Choose one option: (1-4)
0
Error: Input must be between 1 and 4. Please try again.
Choose one option: (1-4)
a
Error: Please enter a valid number between 1 and 4.
Choose one option: (1-4)
@
Error: Please enter a valid number between 1 and 4.
Choose one option: (1-4)
```

```
Choose one option: (1-4)
2

=====================================
Success: processing films with the selected rating...
Searching for films with rating: Good
=====================================

Match found:
Title: Video Club de elias
Duration: 120
Spectator Number: 204000
Spectator Film Valoration: 9.0
Calculated rating: Good
Suitable for all audience: false
Director's name: Elias Roig
Total films in the Videoclub: 1

=====================================
Films found with this title: 1
=====================================
```

Y hasta aquí las pruebas de esta entrega.
Un saludo.