Pau Casesnoves CIFP

Elias Roig Alcon

Curso 2024-2025

FPGS Desarrollo Multiplataforma

Unidad de Trabajo 5: Desarrollo de clases.

Index

VideoClub (App)

Se trata de desarrollar una aplicación Java denominada VideoClub que permita gestionar la información de una película y su director. Mediante un menú que aparecerá en pantalla se podrán realizar determinadas operaciones:

Clase: Main

```
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";
    }

}
```

Clases Objeto: (Film y Director)

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;

    public Film() {

    }

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

    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()
                + "\nSuitable for all audience: " + this.isAllPublicFilm()
                + "\nDirector's name: " + this.getDirectorFilmName()
                + "\nFilms created: " + filmCounter;
    }
}
```

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 Controller:

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;

    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========================\n");
            System.out.println("There's no Director created yet.");
            System.out.println("========================\n");
        }
    }

    public static void increaseDirectorAwards() {
        int increase = 0;


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 directorFilmName = Utility.validateDirectorName();
```

```
            film = new Film(filmName, filmMinutes, spectatorNum, specValoration, suitableForAll,
directorFilmName);
            System.out.println("\n[[=== Film created succesfully ===]]\n");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void showFilmData() {
        if (film != null) {
            System.out.println("\n===== All Film data =====");
            System.out.println(film.toString() + ".");
            System.out.println("========================\n");
        } else {
            System.out.println("\n========================\n");
            System.out.println("There's no Film created yet.");
            System.out.println("========================\n");
        }
    }

    public static void modifyFilmSpectatorRating() {
        double modifyRating = 0;

        if (film != null) {
            modifyRating = Utility.validateSpecRating();
            film.setSpectatorValoration(modifyRating);
            System.out.println("Succes: New film rating: " + film.getSpectatorValoration() + "\n");
        }
    }

    public static void showFilmClassification() {
        Utility.generateReport(film, director);
    }

}
```

Clase Utilidades:

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 int totalAwards = 0;

    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 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 ===\n"
                + "----------------------------------------\n"
                + "[ 10 thousand spectators can't rate a film. ]\n"
                + "[ 500 thousand spectators can get an exellent rating. ]\n"
                + "----------------------------------------");
```

```
        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() {
        double filmRating;
        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
                }

                if (filmRating >= 8.0 && spectatorsNum >= 500000) {
                    specRating = "The film is excellent and recommended!";
                } else if (filmRating >= 5.5 && filmRating < 8.0 && spectatorsNum >= 10000 &&
spectatorsNum < 500000) {
                    specRating = "The film is good and recommended.";
                } else if (filmRating < 5.5 && spectatorsNum >= 10000 && spectatorsNum <
500000) {
                    specRating = "The film is not recommended.";
                } else if (spectatorsNum < 10000) {
                    specRating = "The film is unknown.";
                } else {
                    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;
}

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) {
                System.out.println("Error: Must be 0.0 or positive!");
            } 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("Calculation 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(specRating).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());
    }
}
```

Test Space

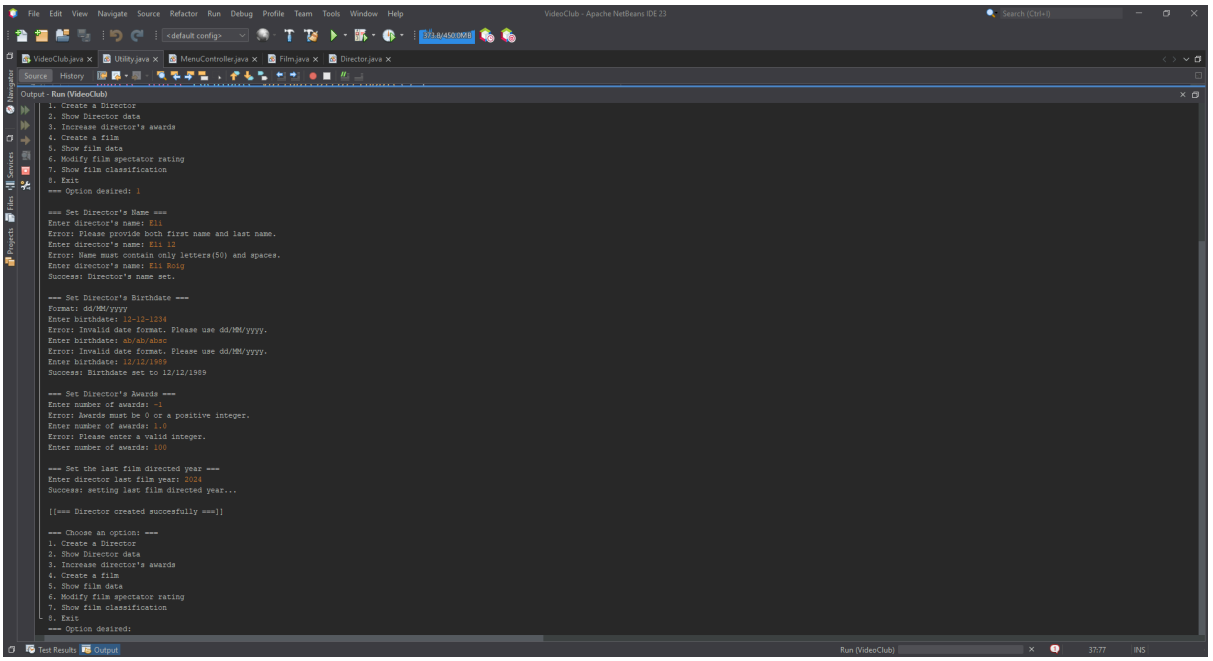

Figura 1: Opción 1

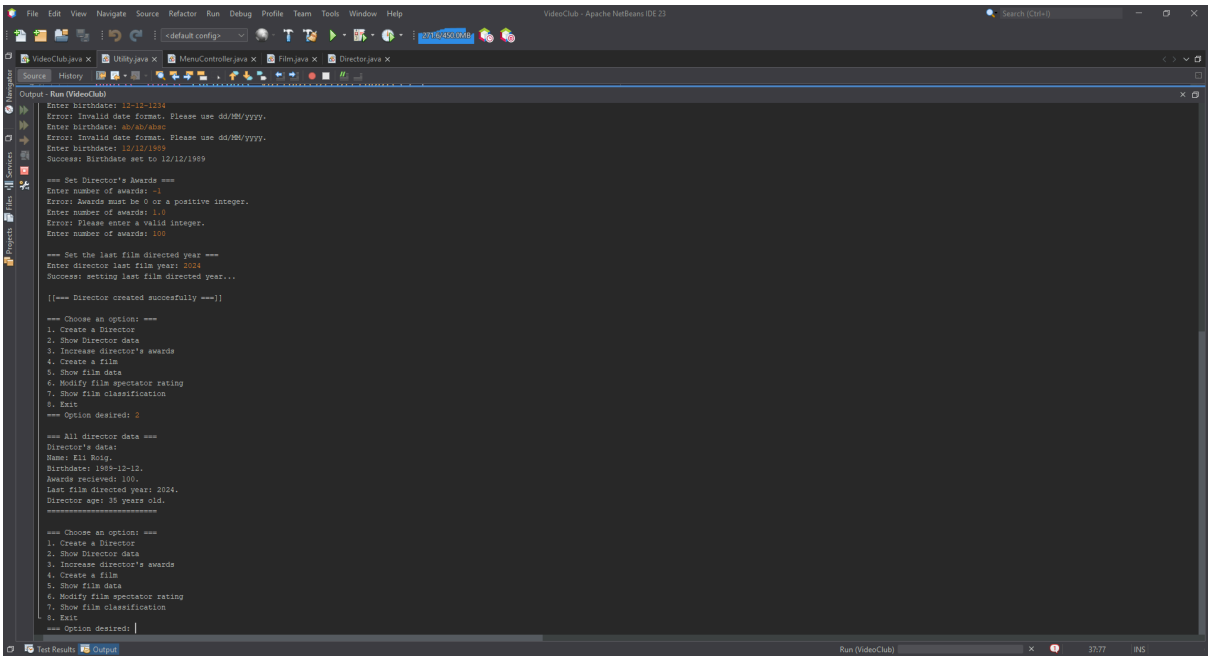

Figura 2: Opción 2

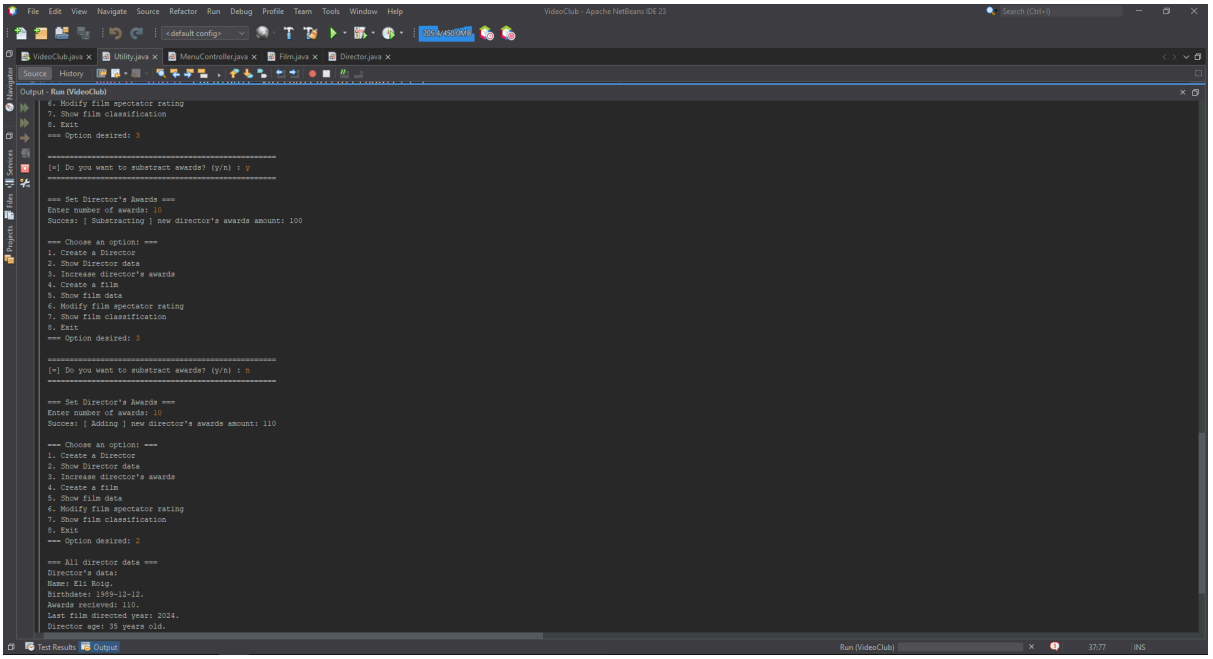

Figura 3: Opción 3

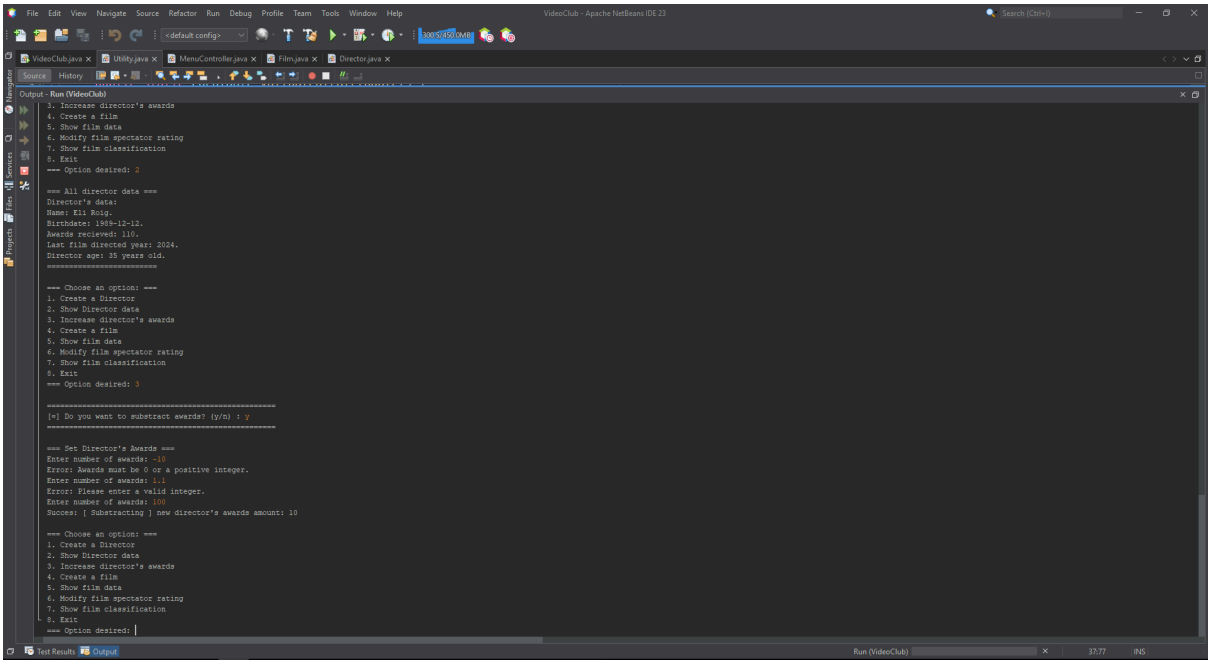

Figura 4: Validación opción 3

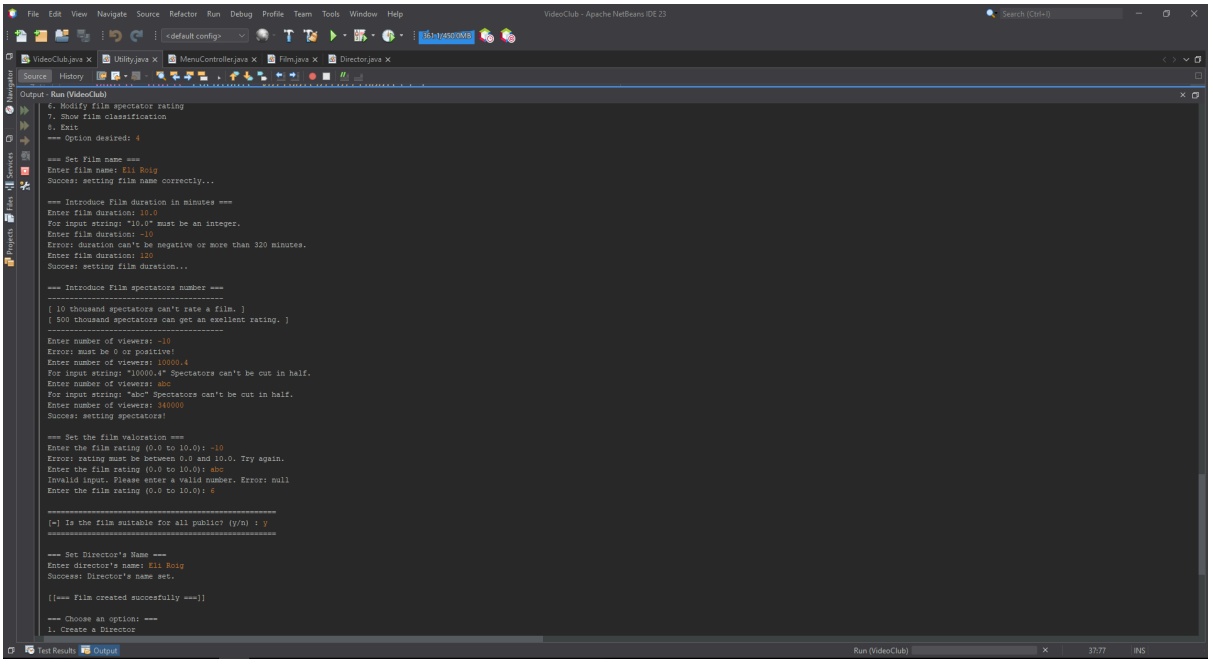

Figura 5: Opción 4

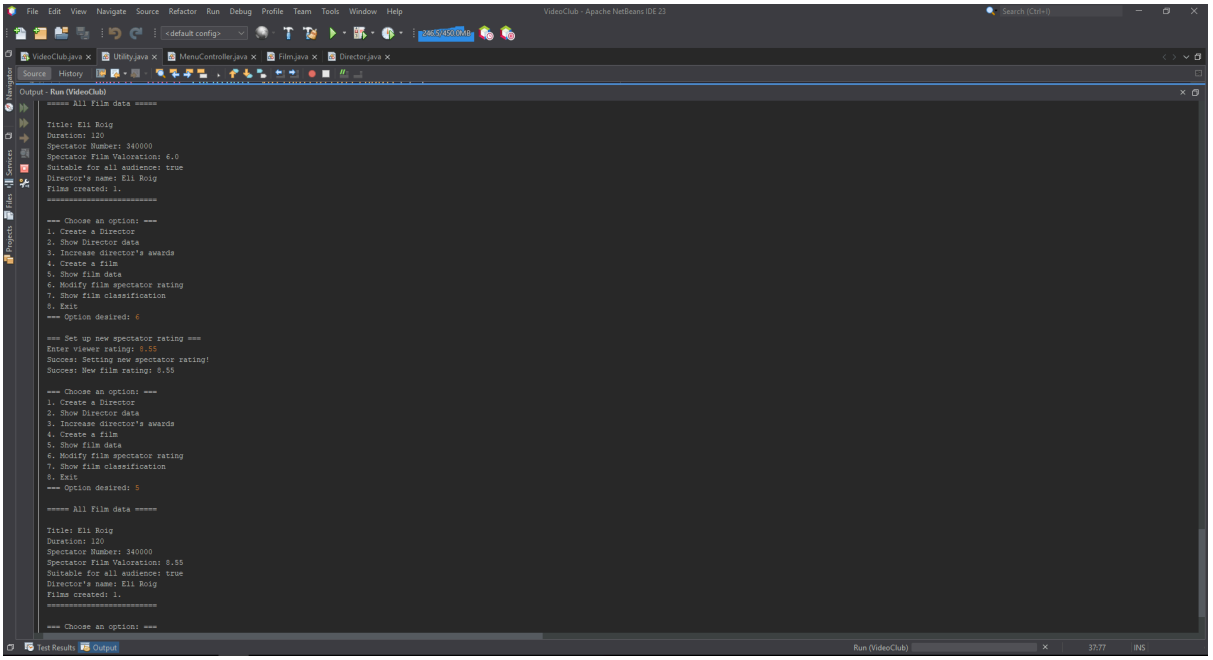

Figura 6: Opción 5 y 6

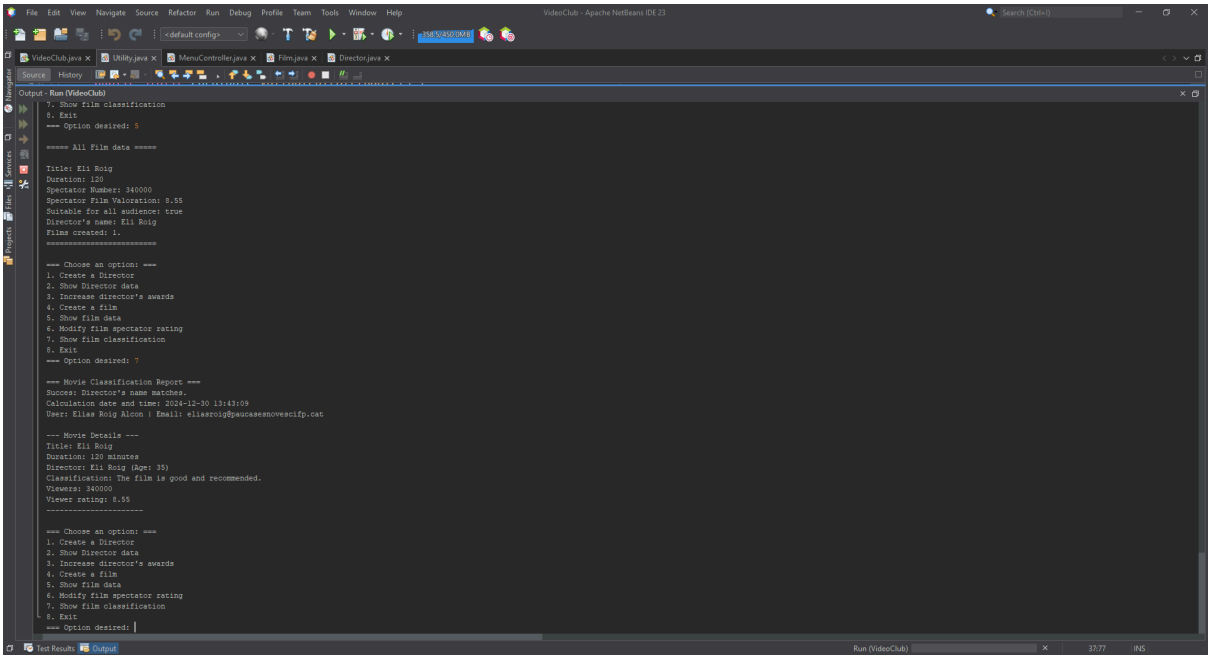

Figura 6: Opción 7; Nombre del director coincide.

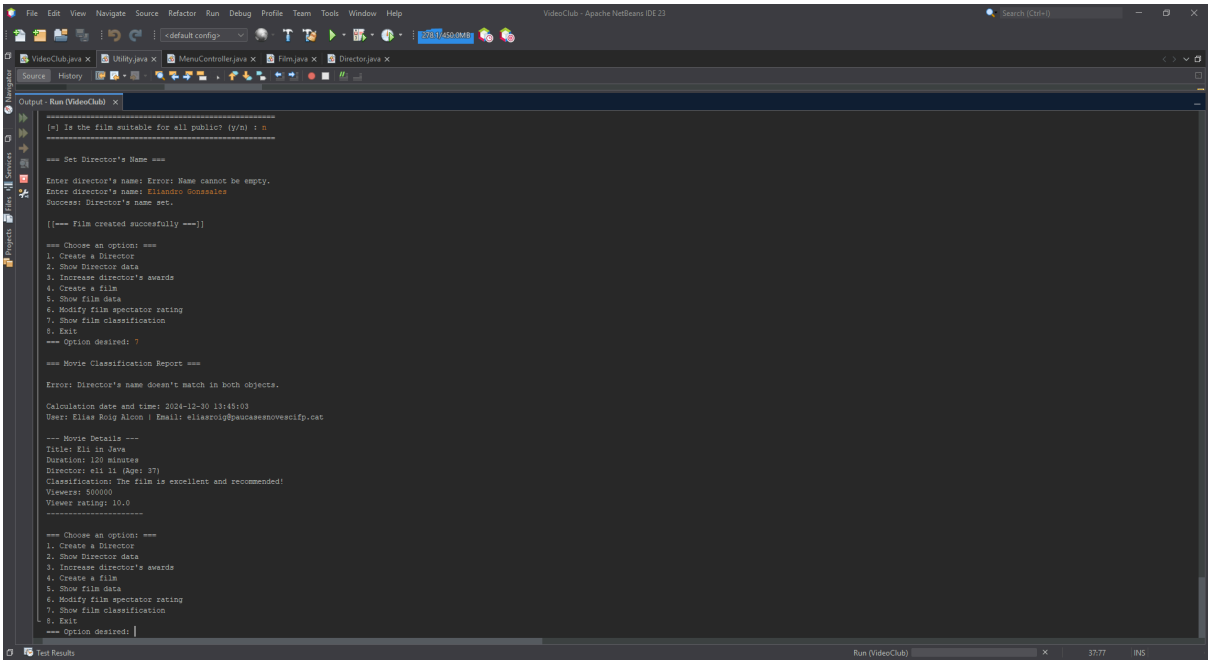

Figura 7: Opción 7; Nombre del director no coincide.