

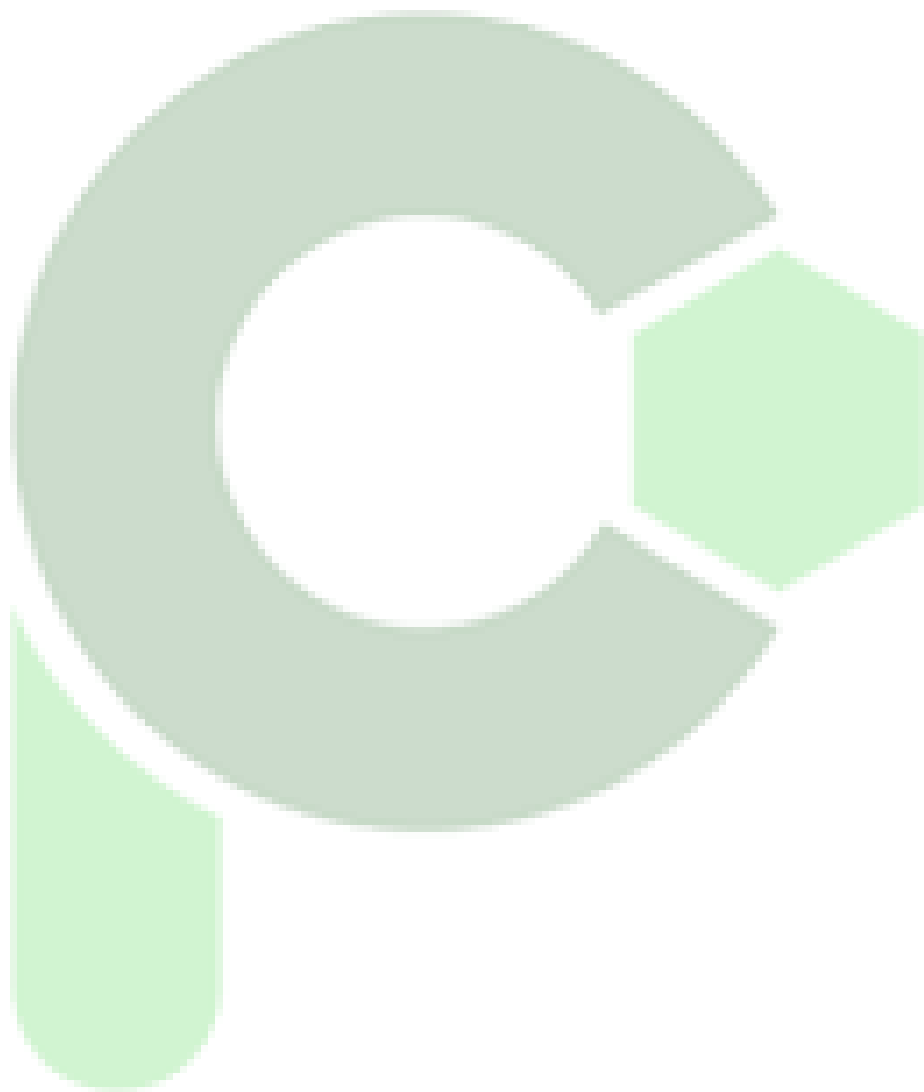


# Pau Casesnoves CFP

Elias Roig Alcon  
Curso 2024-2025

FPGS Desarrollo Multiplataforma

Unidad de Trabajo 5: Desarrollo de clases.





## Index

<b>Pau Casesnoves CIFP.....</b>	<b>1</b>
Unidad de Trabajo 5: Desarrollo de clases.....	1
<b>Index.....</b>	<b>2</b>
VideoClub (App).....	3
Clase: Main.....	3
Clases Objeto: (Film y Director).....	5
Film.....	5
Director.....	7
Clase Controller:.....	9
MenuController.....	9
Clase Utilidades:.....	12
Utility.....	12
Test Space.....	19



## 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 subtract awards? (y/n) : ");
    String subtract = sc.nextLine().trim().toLowerCase();

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

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

    if (director != null && subtract.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 && subtract.equals("y")) {
        increase = Utility.validateDirAwardsNum();
        director.setAwardsNum(director.getAwardsNum() - increase);
        System.out.println("Succes: [ Subtracting ] 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 subtractAwards() {
    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 valuation ===");

    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("Success: 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

```
Output - Run (VideoClub)
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. Exit
--- Option desired: 1 ---

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

--- Set Director's Birthdate ---
Format: dd/MM/yyyy
Enter birthdate: 12-12-1994
Error: Invalid date format. Please use dd/MM/yyyy.
Enter birthdate: ab/ab/abab
Error: Invalid date format. Please use dd/MM/yyyy.
Enter birthdate: 12/12/1989
Success: Birthdate set to 12/12/1989

--- Set Director's Awards ---
Enter number of awards: -1
Error: Awards must be 0 or a positive integer.
Enter number of awards: 100
Error: Please enter a valid integer.
Enter number of awards: 100

--- Set the last film directed year ---
Enter director last film year: 2024
Success: setting last film directed year...

[==== Director created successfully ====]

--- 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. Exit
--- Option desired:
```

Figura 1: Opción 1

```
Output - Run (VideoClub)
Error birthdate: 12-12-1994
Error: Invalid date format. Please use dd/MM/yyyy.
Enter birthdate: ab/ab/abab
Error: Invalid date format. Please use dd/MM/yyyy.
Enter birthdate: 12/12/1989
Success: Birthdate set to 12/12/1989

=== Set Director's Awards ===
Enter number of awards: -1
Error: Awards must be 0 or a positive integer.
Enter number of awards: 100
Error: Please enter a valid integer.
Enter number of awards: 100

--- Set the last film directed year ---
Enter director last film year: 2024
Success: setting last film directed year...

[==== Director created successfully ====]

--- 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. Exit
--- Option desired: 2 ---

=== All director data ===
Director's data:
Name: Eli Roig.
Birthdate: 1989-12-12.
Awards received: 100.
Last film directed year: 2024.
Director age: 35 years old.

--- 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. Exit
--- Option desired:
```

Figura 2: Opción 2



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
VideoClub - Apache NetBeans IDE 21
Source
History
Output - Run (VideoClub)
6. Modify film spectator rating
7. Show film classification
8. Exit
=== Option desired: 3
=====
[+] Do you want to subtract awards? (y/n) : y
=====
=== Set Director's Awards ===
Enter number of awards: 10
Success: [ Subtracting ] new director's awards amount: 100

=== 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. Exit
=== Option desired: 3
=====
[+] Do you want to subtract awards? (y/n) : n
=====
=== Set Director's Awards ===
Enter number of awards: 10
Success: [ Adding ] new director's awards amount: 110

=== 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. Exit
=== Option desired: 3
=====
=== All director data ===
Director's data:
Name: Eli Roig
Birthdate: 1989-12-12
Awards received: 110
Last film directed year: 2024
Director age: 35 years old.

Test Results Output
Run (VideoClub) x 37.77 ms
```

Figura 3: Opción 3

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
VideoClub - Apache NetBeans IDE 21
Source
History
Output - Run (VideoClub)
1. Increase director's awards
4. Create a film
5. Show film data
6. Modify film spectator rating
7. Show film classification
8. Exit
=== Option desired: 3
=====
=== All director data ===
Director's data:
Name: Eli Roig
Birthdate: 1989-12-12
Awards received: 110
Last film directed year: 2024
Director age: 35 years old.

=== 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. Exit
=== Option desired: 3
=====
[+] Do you want to subtract awards? (y/n) : y
=====
=== Set Director's Awards ===
Enter number of awards: 10
Error: Awards must be 0 or a positive integer.
Enter number of awards: 1.1
Error: Please enter a valid integer.
Enter number of awards: 100
Success: [ Subtracting ] new director's awards amount: 10

=== 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. Exit
=== Option desired: 3
=====
[+] Do you want to subtract awards? (y/n) : y
=====
=== Set Director's Awards ===
Enter number of awards: 10
Error: Awards must be 0 or a positive integer.
Enter number of awards: 1.1
Error: Please enter a valid integer.
Enter number of awards: 100
Success: [ Subtracting ] new director's awards amount: 10

Test Results Output
Run (VideoClub) x 37.77 ms
```

Figura 4: Validación opción 3



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
VideoClub - Apache NetBeans IDE 21
VideoClub.java X Utility.java X MenuController.java X Film.java X Director.java X
Source History
Output - Run (VideoClub)
4. Modify film spectator rating
7. Show film classification
8. Exit
--- Option desired: 4
--- Set Film name ---
Enter film name: Eli Roig
Success: setting film name correctly...
--- Introduce film duration in minutes ---
Enter film duration: 120
For input string: "120" must be an Integer.
Enter film duration: -10
Error: duration can't be negative or more than 320 minutes.
Enter film duration: 120
Success: setting film duration...
--- Introduce film spectators number ---
[ 10 thousand spectators can't rate a film. ]
[ 500 thousand spectators can get an excellent rating. ]
Enter number of viewers: -10
Error: must be 0 or positive!
Enter number of viewers: 10000.4
For input string: "10000.4" Spectators can't be out in half.
Enter number of viewers: abc
For input string: "abc" Spectators can't be out in half.
Enter number of viewers: 340000
Success: setting spectators!
--- Set the film valuation ---
Enter the film rating (0.0 to 10.0): -10
Error: rating must be between 0.0 and 10.0. Try again.
Enter the film rating (0.0 to 10.0): abc
Invalid input. Please enter a valid number. Error: null
Enter the film rating (0.0 to 10.0): 6
=====
[?] Is the film suitable for all public? (y/n) : y
--- Set Director's Name ---
Enter director's name: Eli Roig
Success: Director's name set.
[?] Film created successfully [?]
--- Choose an option: ---
1. Create a Director
```

Figura 5: Opción 4

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
VideoClub - Apache NetBeans IDE 21
VideoClub.java X Utility.java X MenuController.java X Film.java X Director.java X
Source History
Output - Run (VideoClub)
===== All Film data =====
Title: Eli Roig
Duration: 120
Spectator Number: 340000
Spectator Film Valuations: 6.0
Suitable for all audience: true
Director's name: Eli Roig
Films created: 1.
=====
--- 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. Exit
--- Option desired: 5
--- Set up new spectator rating ---
Enter viewer rating: 6.55
Success: Setting new spectator rating!
Success: New film rating: 6.55
--- 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. Exit
--- Option desired: 5
===== All Film data =====
Title: Eli Roig
Duration: 120
Spectator Number: 340000
Spectator Film Valuations: 6.55
Suitable for all audience: true
Director's name: Eli Roig
Films created: 1.
=====
--- Choose an option: ---
```

Figura 6: Opción 5 y 6



```
7. Show film classification
8. Exit
--- Option desired: 5

===== All Film data =====
Title: Eli Roig
Duration: 120
Spectator Number: 340000
Spectator Film Valorization: 9.55
Suitable for all audience: true
Director's name: Eli Roig
Film created: 1.
=====

--- 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. Exit
--- Option desired: 7

--- Movie Classification Report ---
Success: Director's name matches.
Calculation date and time: 2024-12-30 13:13:09
User: Elias Roig Alcon | Email: eliasroig@paucasesnovescifp.cat

--- Movie Details ---
Title: Eli Roig
Duration: 120 minutes
Director: Eli Roig (Age: 35)
Classification: The film is good and recommended.
Viewers: 340000
Viewer rating: 9.55

--- 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. Exit
--- Option desired: |
```

Figura 6: Opción 7; Nombre del director coincide.

```
[+] Is the film suitable for all public? (y/n) : n

--- Set Director's Name ---
Enter director's name: Error: Name cannot be empty.
Enter director's name: Eliandro Gonzalez
Success: Director's name set.

[+] Film created successfully [+]

--- 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. Exit
--- Option desired: 7

--- Movie Classification Report ---
Error: Director's name doesn't match in both objects.
Calculation date and time: 2024-12-30 13:45:03
User: Elias Roig Alcon | Email: eliasroig@paucasesnovescifp.cat

--- Movie Details ---
Title: Eli in Java
Duration: 120 Minutes
Director: eli li (Age: 37)
Classification: The film is excellent and recommended!
Viewers: 560000
Viewer rating: 10.0

--- 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. Exit
--- Option desired: |
```

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