MINI PROJECT

MOVIE WATCHLIST CREATOR

- **STUDENT NAME: SIWALIK BHATTACHARYA**

  REG: 12104906
- **STUDENT NAME: SHASHWAT SINGH**

  REG: 12108648
- **STUDENT NAME: SHUBHRAJIT DAS**

  REG: 12205008

Submitted to

Dr. Om Prakash Yadav (26121)

Asst. Professor,

School of Computer Science and Engineering

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

March, 2023

ABSTRACT

The Movie Watchlist Creator is the project which will help users to maintain a library of their favourite movies. The library will be in the form of a list of movies. The user will be able to modify by adding or removing their favourite movies. This project is good example of how bigger system is divided into multiple smaller classes and all working together.

The Movie Watchlist Creator is developed in Java. It can handle one request at a time (either adding or removing or different given choices). The user will be required to enter the choices given to him then the program performs the action based on the requirement by the user. In this project the user will be able to keep the record of IMDB rating and PG rating which will help him to keep the record of movies in a better way.

The user will be given the choices to Display Watchlist, Choose Movie, Add Movie, Remove Movie, Exit (If don't want to perform any action).

The program then show the content based on requirement, like if user choose Display Watchlist

“choice: 1” it will show the Movie Name along with IMDB rating and PG rating, if the Movie is available, else will prompt a message referring to its unavailability.

INTRODUCTION

The program is to be developed for Movie Watchlist Creator. A Movie Watchlist Creator is a platform for the users which will provides them to keep the records of their favourite movies. It will be a friendly and easier to use to platform for the users for their record. The program consists of the following options:-

Option 1: Display Watchlist

Option 2: Choose Movie

Option 3: Add Movie

Option 4: Remove Movie

Option 5: Exit

This program consists of 5 classes which is a great example of how a bigger system is divided into multiple smaller classes. Each class has their own different functionality. Class Movie is class having three variables: name, IMDB Rating, PG-Rating. We have used constructor to initialize them. We have also created different classes like Display _Watchlist, Cinema, Add _Remove and Main class. When a user wants to add a movie he will choose

“*option 3”* (Add Movie) then the program will asks user to enter movie name then to enter IMDB rating and then to enter PG rating. Then the movie will be added to the list(if it was already not present). To see the list the user will select “*option 1”* (Display Watchlist) then the whole list of the movie having name, IMDB rating and PG-rating will be displayed on the screen. To see the details of specific movie user have to select “*option 2”* (Choose Movie) then the program will ask user to enter movie name following which all the details related to the movie will be displayed on the screen (if the movie name is present in the list). To remove or delete a specific movie user have to choose for “*option 4”* (Remove Movie) then the program will ask for the movie name after entering the movie name the movie will be deleted (if the movie name is present in the list). The final “*option 5”* (Exit) can be chosen to exit from the program.

EXISTING SYSTEM

1. The existing system is manual system.
2. The records that we keep in the registers manually are having the high chances of getting lost or getting damaged.
3. The system consists of a lot of manual entries.
4. Usage of papers and records in the process leads to less efficiently less productivity.
5. Increase lots of mistakes while writing in paper.
6. Writing manually is time taken process.
7. For these types of reasons, the new system is invented.

PROPOSED SYSTEM

1. The proposed system is automated system (details will be entered by user).
2. The automated system is less prone to error as compared to manual system.
3. There is no usage of papers and records in the process so it is high in productivity.
4. Decrease lots of mistakes as compared to manual system
5. Time taken will be decrease as compared to manual system.
6. These are the reasons the proposed systems are better as compared to manual system.

ADVANTAGES

1.Easier to use.

2.Chances of losing of records will be decreased.

3.It will reduce the usage the papers so it is also eco-friendly.

4.All the records will be at one place.

5. Getting records at one place will make it easier for the user to select, add, or remove his favourite movies.

MINIMUM SYSTEM REUIREMENTS

1. Operating System: - Windows XP or higher.

2.Architecture: - 32bit or 64bit.

3.Processor type: - Intel Core, AMD Ryzen (any of these).

4.Cores: - 8 Minimum

5.Threads: - 16 Minimum

6.RAM: - 2GB Minimum

7.Processor Speed: - 1.20 GHZ or higher

8.Hard disk: - 40GB or more.

9.Software Required: Any text editor (like notepad) and command prompt.

10.Compiler Required: JDK 1.8 or higher.

CODE

```
import java.util.*;

class Movie {
    String name;
    double imdbRating;
    String pgRating;
```

```
    public Movie(String name, double imdbRating, String pgRating) {
        this.name = name;
        this.imdbRating = imdbRating;
        this.pgRating = pgRating;
    }
}
```

```
class Display_Watchlist {
    ArrayList<Movie> watchlist;
```

```
    public Display_Watchlist(ArrayList<Movie> watchlist) {
        this.watchlist = watchlist;
    }
```

```
    public void display() {
        if (watchlist.isEmpty()) {
            System.out.println("Watchlist is empty");
        } else {
            for (Movie movie : watchlist) {
                System.out.println(movie.name + " - IMDB: " + movie.imdbRating + ", PG: " +
movie.pgRating);
            }
        }
    }
}
```

```
class Cinema {
    ArrayList<Movie> watchlist;
```

```
    public Cinema(ArrayList<Movie> watchlist) {
        this.watchlist = watchlist;
    }
```

```
    public void chooseMovie(String name) {
        boolean found = false;
        for (Movie movie : watchlist) {
            if (movie.name.equalsIgnoreCase(name)) {
                System.out.println(movie.name + " - IMDB: " + movie.imdbRating + ", PG: " +
movie.pgRating);
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("Movie not found");
        }
    }
}
```

```
class Add_Remove {
    ArrayList<Movie> watchlist;

    public Add_Remove(ArrayList<Movie> watchlist) {
        this.watchlist = watchlist;
    }

    public void addMovie(Movie movie) {
     boolean found = false;
        for (Movie m : watchlist) {
            if (m.name.equalsIgnoreCase(movie.name)) {
                found = true;
                break;
            }
        }
        if (found) {
            System.out.println("Movie already exists");
        } else {
            watchlist.add(movie);
            System.out.println("++ADDED++");
        }
    }

    public void removeMovie(String name) {
        boolean found = false;
        for (Movie movie : watchlist) {
            if (movie.name.equalsIgnoreCase(name)) {
                watchlist.remove(movie);
                found = true;
                System.out.println("--REMOVED--");
                break;
            }
        }
        if (!found) {
            System.out.println("Movie not found");
        }
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        ArrayList<Movie> watchlist = new ArrayList<>();
        watchlist.add(new Movie("The Shawshank Redemption", 9.3, "R"));
        watchlist.add(new Movie("The Godfather", 9.2, "R"));
        watchlist.add(new Movie("The Dark Knight", 9.0, "PG-13"));
        watchlist.add(new Movie("The Last Of Us", 8.9, "TV-MA"));
        watchlist.add(new Movie("Top Gun: Maverick", 8.3, "M"));
        watchlist.add(new Movie("Joker", 8.4, "R"));
```

```
        Scanner scanner = new Scanner(System.in);
        String choice;
        Display_Watchlist displayWatchlist = new Display_Watchlist(watchlist);
        Cinema cinema = new Cinema(watchlist);
        Add_Remove addRemove = new Add_Remove(watchlist);
```

```
        do{
            System.out.println("1. Display Watchlist\n2. Choose Movie\n3. Add Movie\n4. Remove Movie\n5.
Exit");
          System.out.print("Enter your choice: ");
            choice = scanner.nextLine();
            //scanner.nextLine();
```

```
            switch (choice) {
                case "1":
                    System.out.println("\nThe watchlist: ");
                    displayWatchlist.display();
                    break;
                case "2":
                    System.out.print("\nEnter movie name: ");
                    String name = scanner.nextLine();
                    cinema.chooseMovie(name);
                    break;
                case "3":
                    System.out.print("\nEnter movie name: ");
                    String addName = scanner.nextLine();
                    System.out.print("\nEnter IMDB rating: ");
                    double imdbRating=0.0;
                    boolean b=true;
                    while(b){
                        String iR=scanner.nextLine();
                        try{
                            imdbRating = Double.parseDouble(iR);
                            b=false;
                        }
                        catch (NumberFormatException e) {
                            System.out.println("\nArgument must be numeric\nTry entering the IMDB
rating again: ");
                        }
                    }
                    System.out.print("\nEnter PG rating: ");
                    String pgRating = scanner.nextLine();
                    addRemove.addMovie(new Movie(addName, imdbRating, pgRating));
                    break;
                case "4":
                    System.out.print("\nEnter movie name: ");
                    String removeName = scanner.nextLine();
                    addRemove.removeMovie(removeName);
                    break;
                case "5":
                    System.out.println("\nExiting...");
                    break;
                default:
                    System.out.println("\nInvalid choice");
              break;
            }
          System.out.println();
           }while(!(choice.equals("5")));
       }
}
```

OUTPUT

```
PS C:\Users\siwal\Desktop\Java Project> javac Main.java
PS C:\Users\siwal\Desktop\Java Project> java Main
1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 1

The watchlist:
The Shawshank Redemption - IMDB: 9.3, PG: R
The Godfather - IMDB: 9.2, PG: R
The Dark Knight - IMDB: 9.0, PG: PG-13
The Last Of Us - IMDB: 8.9, PG: TV-MA
Top Gun: Maverick - IMDB: 8.3, PG: M
Joker - IMDB: 8.4, PG: R

1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 2

Enter movie name: Joker
Joker - IMDB: 8.4, PG: R

1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 2

Enter movie name: abc
Movie not found
```

```
1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 3

Enter movie name: Beef

Enter IMDB rating: asd

Argument must be numeric
Try entering the IMDB rating again:
asd

Argument must be numeric
Try entering the IMDB rating again:
8.3

Enter PG rating: TV-MA
++ADDED++

1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 1

The watchlist:
The Shawshank Redemption - IMDB: 9.3, PG: R
The Godfather - IMDB: 9.2, PG: R
The Dark Knight - IMDB: 9.0, PG: PG-13
The Last Of Us - IMDB: 8.9, PG: TV-MA
Top Gun: Maverick - IMDB: 8.3, PG: M
Joker - IMDB: 8.4, PG: R
Beef - IMDB: 8.3, PG: TV-MA
```

```
1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 4

Enter movie name: abc
Movie not found

1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 4

Enter movie name: Beef
--REMOVED--

1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 1

The watchlist:
The Shawshank Redemption - IMDB: 9.3, PG: R
The Godfather - IMDB: 9.2, PG: R
The Dark Knight - IMDB: 9.0, PG: PG-13
The Last Of Us - IMDB: 8.9, PG: TV-MA
Top Gun: Maverick - IMDB: 8.3, PG: M
Joker - IMDB: 8.4, PG: R
```

```
1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 3

Enter movie name: Joker

Enter IMDB rating: 8.4

Enter PG rating: R
Movie already exists

1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 1

The watchlist:
The Shawshank Redemption - IMDB: 9.3, PG: R
The Godfather - IMDB: 9.2, PG: R
The Dark Knight - IMDB: 9.0, PG: PG-13
The Last Of Us - IMDB: 8.9, PG: TV-MA
Top Gun: Maverick - IMDB: 8.3, PG: M
Joker - IMDB: 8.4, PG: R

1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: 5

Exiting...
```

```
1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice: asd

Invalid choice

1. Display Watchlist
2. Choose Movie
3. Add Movie
4. Remove Movie
5. Exit
Enter your choice:
```

CONCLUSION

Creating a Movie Watchlist Creator Project helped us to gain experience on how multiple classes works together. Every member of the group contributed in making different classes. We hope this project will help user to save their time, and increase versatility of the system.

FUTURE ENHANCEMENTS

Future enhancements in the given project includes addition of "*updation function*". In the given project, in order to update the details of a existing Movie, the Movie has to be removed from the list and then added again (with the newer version of the data). This whole process takes a litter longer than running the two above mentioned processes concurrently. Other than that, this project is based on command line interface. Adding some graphical elements to it, will surely act as a promising future enhancement.

REFERENCES

- INTRODUCTION TO JAVA PROGRAMMING by Y. DANIEL LIANG, PEARSON.
- JAVA THE COMPLETE REFERENCE by HERBERT SCHILDT, MCGRAW HILL EDUCATION.