HACETTEPE UNIVERSITY

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

Name and Surname: Mustafa Aşkın Pınar

Identity Number: 20724964

Course: BBM104 Introduction to Programming Laboratory II

Experiment: Assignment 2

Subject: Inheritance, Access Modifiers

Due Date: 16.04.2021

Advisor: Merve Özdeş

E-mail: b20724964@cs.hacettepe.edu.tr

Main Program: Main.java

Problem

The aim of the program is to develop a simple Movie Database System similar to IMDB. The system will process several data input files and will generate results of commands which will be read from a command input file. All input files will be error free only syntactically.

There are six different recording samples in people file.

There are four different film type in films file.

All data input files will be processed according to the commands which will be given in a commands file. The command file contains 12 types of commands.

1. A user can rate a film.
2. It is possible to add a new feature film.
3. Be able to show the details of a movie
4. A user can list all films which he/she rated so far.
5. A user can edit a film which he/she rated so before.
6. A user can remove one of his/her rated films.
7. List all the TV Series in the system.
8. List all the films from a specified country.
9. List all the films released before a specified year.
10. List all the films released after a specified year.
11. List all the films in descending order and categorized according to film rating degrees.
12. List all the artists from a specified country and display in a categorized order.

Solution

The interface "reader" has been created. Classes were created to read each txt files. Object classes were created to save the information in the txt files which were read.

“Films” classic read the films.txt file. Then it assigned the information it read to the relevant classes under the movie class.

“Peoples” classic read the people.txt file. Then it assigned the information it read to the relevant classes under the people class.

“Commands” classic read the commands.txt file. It has read commands as a 2d string array (called *result*). This class contains some methods. These methods were created to execute commands given in commands.txt.

The operation was done by calling the above 3 classes in the “Main” class. Traveled over *result* with for loop. Then, the data on the *result* was compared with the methods and the operation was made.

Data Structure

I created nineteen different classes. Which are;

- Main.java
- Reader.java → Interface
  Has one method;
  - FileReader()
- Peoples.java → implements Reader.java
- Films.java → implements Reader.java
- Commands.java → implements Reader.java
  Has sixteen methods;
  - findFilm()
  - findPerson()
  - addFeatureFilm()
  - closeOutputFile()
  - createOutputFilr()
  - editRate()
  - listArtistByCountry()
  - listFilmsAfterYear()
  - listFilmsBeforeYear()
  - listFilmsByCountry()
  - listFilmsRate()
  - listTVSeries()
  - listUserRate()
  - rate()
  - removeRate()
  - viewFilm()
- Film.java
- ShortFilm.java → extends Film.java
- TVSeries.java → extends Film.java
- Documentary.java → extends Film.java
- FeatureFilm.java → extends Film.java
- People.java
- User.java → extends People.java
- Artist.java → extends People.java
- Director.java → extends Artist.java
- Writer.java → extends Artist.java
- Performer.java → extends Artist.java
- Actor.java → extends Performer.java
- ChildActor.java → extends Performer.java
- StuntPerformer.java → extends Performer.java

Explanation of my code

- findFilm() → Finds the category of the movie given the id.
- findPerson() → Finds the category of the person given the id.
- addFeatureFilm() → Adds movies according to the information provided.
- closeOutputFile() → Close the created output file.
- createOutputFilr() → Creates an output file with the given name.
- editRate() → It updates the previously given user score.
- listArtistByCountry() → Lists Artist whose country has been given.
- listFilmsAfterYear() → Lists movies after the given year.
- listFilmsBeforeYear() → Lists movies before the given year.
- listFilmsByCountry() → Lists movies whose country has been given.
- listFilmsRate() → Lists the movies according to their scores.
- listTVSeries() → Lists all TV Series.
- listUserRate() → Lists the ratings of the user given the id.
- rate() → Gives points to the movie by user.
- removeRate() → It removes the previously given user score.
- viewFilm() → Shows the information of the movie given the id.

Algorithm

Reader.java

1. Declare the method named FileReader()

Main.java

1. Create a Peoples object.
2. Create a Films object.
3. Create a Commands object.
4. Call the method createOutputFile()
5. For loop for execute some operations
6. Call the method closeOutputFile()

Peoples.java

1. Only read people.txt and has one for loop
    a. If the data is meaningful, it is assigned to the appropriate objects.
    b. If not print "undefined person category" on the terminal

Films.java

1. Only read films.txt and has one for loop
    a. If the data is meaningful, it is assigned to the appropriate objects.
    b. If not print "undefined person category" on the terminal

Commands.java

1. Read commands.txt
    a. Adds all read data to the 2d string array named result.
2. crateOutputFile()
    a. Creates an output file with the given name.
3. closeOutputFile()
    a. Close the created output file.
4. findFilm()
    a. This method is private
    b. Finds the category of the movie given the id.
    c. If id is in this list return ids category.
    d. If not return “”.
5. findPerson()
    a. This method is private
    b. Finds the category of the person given the id.
    c. If id is in this list return ids category.
    d. If not return “”.
6. rate()
    a. Check the film id and user id.
    b. If the film was not rated before and rating point is between 1 and 10 add a rate point.
    c. If not
        i. If user id or film id is wrong write “Command Failed” in output file.
        ii. If not print “This film was earlier rated.”
    d. If rating point is not between 1 and 10 print” rate should be between 1 and 10."
7. editRate()
    a. Check the film id and user id.
    b. If the film was rated before that user and new rating point is between 1 and 10, update user’s rating point
    c. If not
        i. If user id or film id is wrong write “Command Failed” in output file.
        ii. If not print “rate should be between 1 and 10.”
8. removeRate()
    a. Check the film id and user id.
    b. If the film was rated before that user, removes user’s rating point
    c. If not
        i. If user id or film id is wrong write “Command Failed” in output file.
9. addFeatureFilm()
    a. Check the film id
    b. Check the all-people ids in this movie are in the people.txt
        i. If a and b are true create a new feature film than add values
        ii. If not print “Command Failed”

10. viewFilm()
    a. Check the film id
    b. Find its category
        i. If the film is existing write its information
        ii. If not print “Command Failed”
11. listUserRates()
    a. Check the user id
    b. If the user is existing write its ratings.
    c. If not
        i. If user hasn’t rated any film write “There is not any ratings so far”
        ii. If not print “Command Failed”
12. listTVSeries()
    a. Check the film id
    b. If the film is existing in TVSeries write its information.
    c. If not write “No result”
13. listFilmByCountry()
    a. It browses all the movies by their category and writes the ones with the specified country.
    b. If not write “No resul”
14. listFilmBeforeYear()
    a. It browses all the movies by their category and writes the ones with the specified year.
    b. If not write “No resul”
15. listFilmAfterYear()
    a. It browses all the movies by their category and writes the ones with the specified year.
    b. If not write “No resul”
16. listFilmsRate()
    a. It browses all the movies by their category
    b. If movie has a rating point, then write by order
    c. If not write “No result”
17. listArtistByCountry()
    a. It browses all the people by their category and writes the ones with the specified country.
    b. If not write “No resul”

Film.java

This class is parent class for all films.

1. getRatingScore()
    a. Returns the average of the film by adding all the rating scores of the film.
2. getRatingScoreSize()
    a. Returns how many users rated the movie.

3. addRatingScore()
    a. add user id and user's rating point
4. editRatingScore()
    a. to change movie's rate point
5. removeRatingScore()
    a. to remove movie's rate point
6. other methods are only getter and setter methods for its attributes

People.java

This class only has getter and setter methods for its attributes.
This class is parent class for all people

User.java

1. addRate()
    a. add films id and user's rating point
2. editRate()
    a. to change movie's rate point
3. removeRate()
    a. to remove movie's rate point
4. getRateList()
    a. Returns all movies rated by the user.

*.java

1. All other classes have only getter and setter methods for its attributes

Advantages

In general, the fact that the object classes are in one place and the method classes are in one place increases the readability of the cod. The variable and method names I use do not require much explanation. I tried not to import the library as much as possible. All operations were made using simple methods. You can understand what operations have been done just by looking at the Main class.

Disadvantages

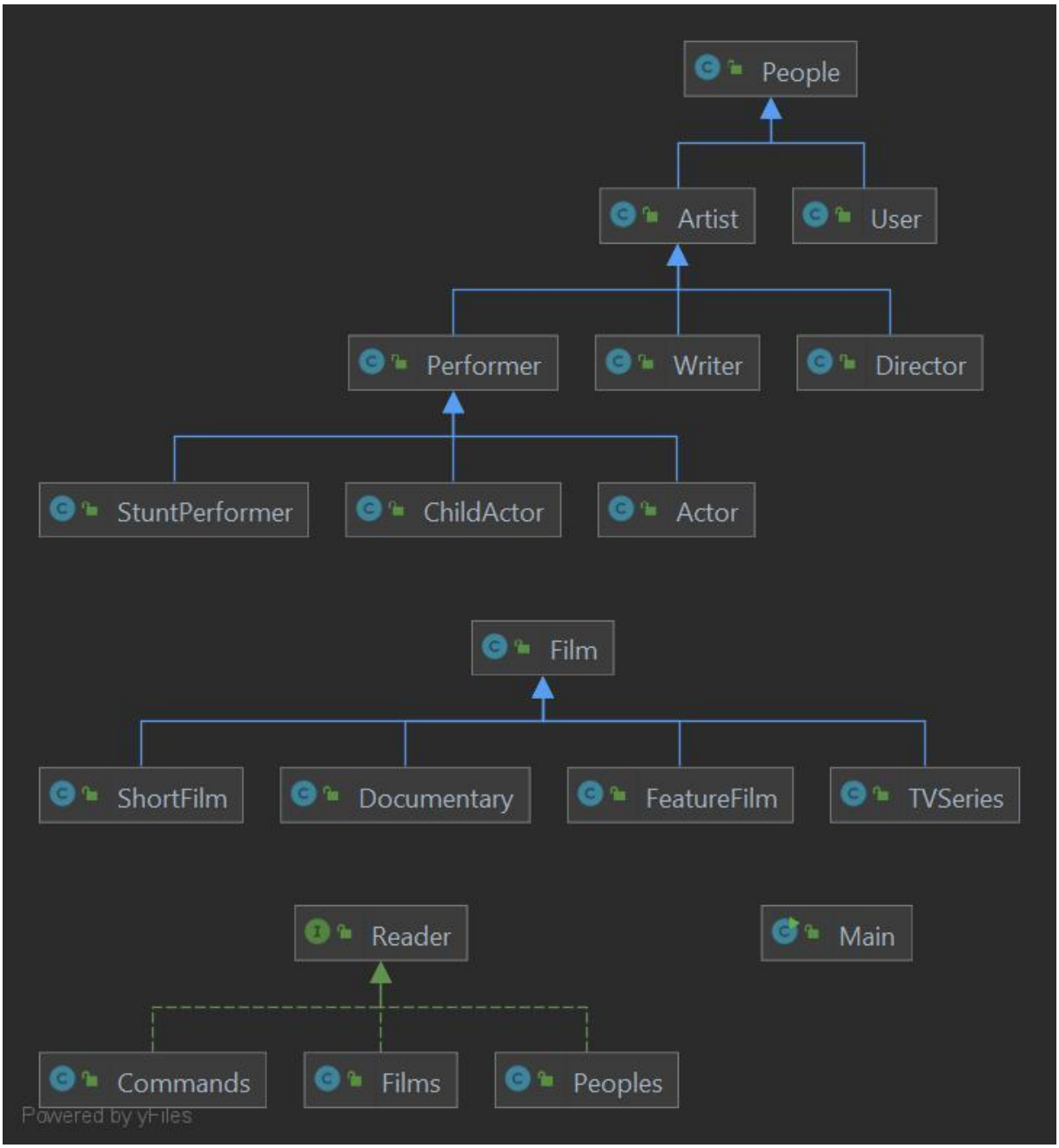
Not using libraries too much caused some loops to take too long. If I had used hashmap, some places would be shortened.

References

1. https://www.stacoverflow.com
2. https://www.geeksforgeeks.org

Diagrams

Class Diagrams



Note: I added the other diagrams to the file as jpg. because if I added it here the resolution would be lower.