|  |
| --- |
| Project 1A  Erik Sklocic  Joseph Jennings  Sonal Sinha  Neha Metlapalli |

Design

**Data Structures Used:**

**ArrayList**: Two arrayLists are made in the menu.java file: movieListComing and movieListShowing. These are then manipulated in the main method based on the user’s input. For example, a user can add a movie (addM), display the movie list (dsply), edit a date of a movie (editDte), show the number of movies given before a date(numMDte), or save the list (save) by typing in an option from the menu. Depending on the chosen option, the needed arrayLists will be manipulated in the while loop of the main function. The menu class consists of functions that are defined in the Movie.java file where the tasks are carried out.

We chose to make two separate array lists in order to distinguish between “Showing” movies and “Coming” movies. This solution made it easier to change each arraylist on its own without affecting the other. For example, if a user chooses to “edit a date” in the “Coming” list, the program iterates through the “Coming” list and searches for the user’s input by movie name. If the movie name is not in the Coming list, the program outputs a message prompting the user to enter a different name. Keeping the arraylists separate helps organize which portion of data to sift through based on the user’s input.

We used Arraylist because we thought the advantages of Arraylist outweigh Linkedlist in our use case. Arraylist comes with built in indexing and an Iterator interface. If we ever had to fall back onto indexing we would lose the advantage of faster indexing using a Linkedlist due to the lack of support of proper indexing.

**Iterators**: An iterator is used often throughout the code in order to iterate through the given list and find a specific element. We used an iterator to optimize our code compared to using indexes which we tried to avoid where ever possible.

**Structure**

**menu Class:** The menu class, located in the menu.java file, contains the printed menu where the user can see the many options to choose from. These menu options presented to the user are: display menu (“dsply”), add a new movie (“addM”), edit a movie description(editMDesc), edit movie date (“editDte”), show movies(“showM”), and show number of movies before a given date(“numMDte”), and save(“save”). The program uses if statements to check the user’s input. If the user entered a proper input, then the appropriate method will be called to execute that command. Each option performs a different task and uses different functions to perform each task. Iterators are used many times in menu.java in order to iterate through a specific list.

**Movie Class:** The Movie class consists of the get and set methods of the program as well as the overridden toString() and displayMovies() methods. The toString() method returns the name, description, release date, receive date, and status of each of the movies to a string that is formatted to our needs. The displayMovies() method is used to display each movie object in the arraylist by its release date, description, receive date, and status (showing or coming).

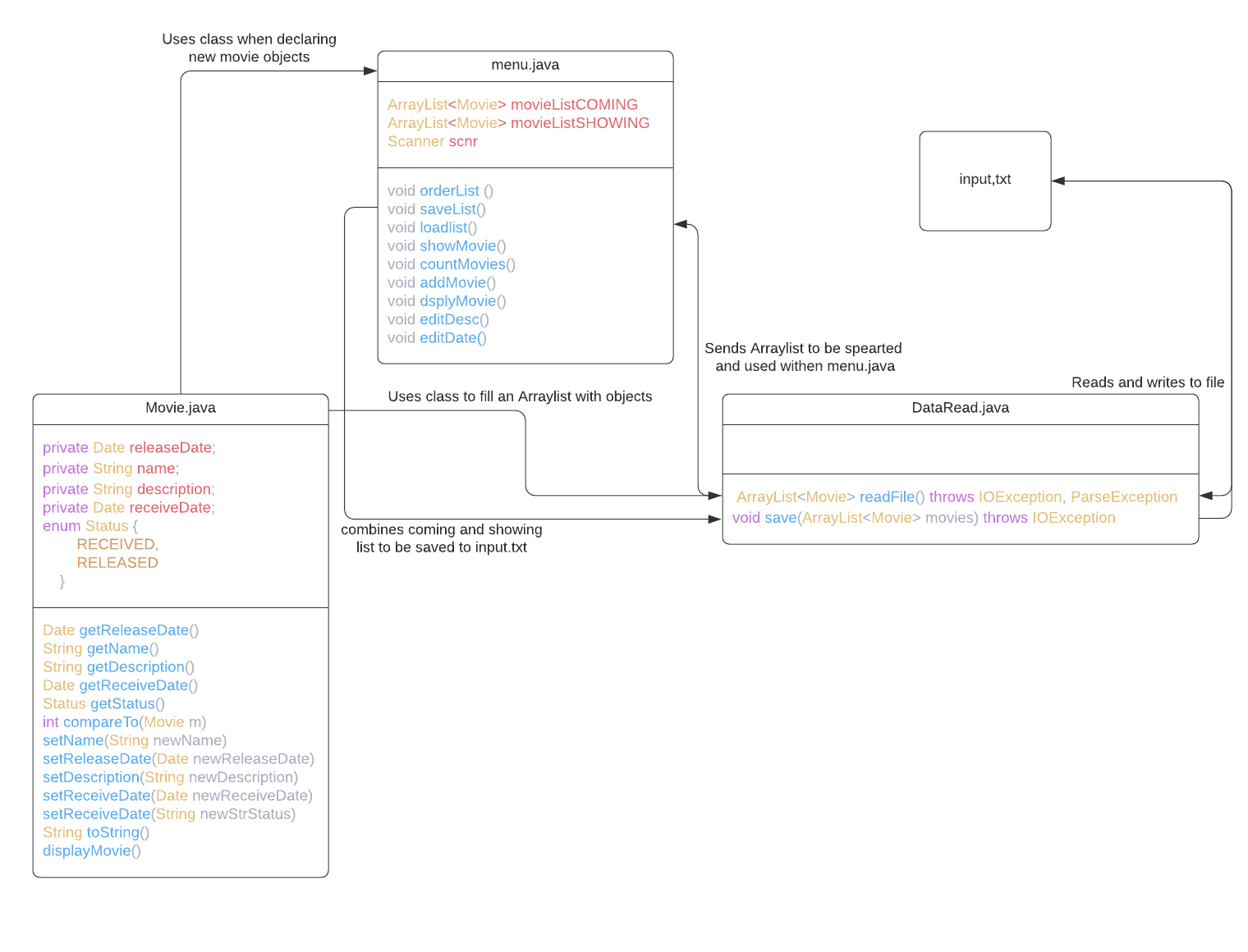
In addition, the movie class implements a “comparable” library interface which allows the program to compare different elements in movies.

**Input.txt File:** The input.txt file is a text file that consists of information about each movie that will be read by the readFile() method in the DataRead Class. The information in this file is separated by a comma and a space, which helps differentiate between the different categories. The categories are: movie name, description, release date, receive date, and status (showing or coming).

**DataRead:** DataRead Class consists of the readFile() method and the save() method. The readFile() method reads the input.txt file that exists outside the class line by line. It uses a while loop to check if there's another line to be read. It stores the information in a string called "line." Then, it splits line by the commas and spaces and stores that in an array called lineList. Each index in lineList corresponds to a different category of information. For example, lineList[0] corresponds to the movie name and lineList[4] corresponds to the movie status.

The save() method in the DataRead class finalizes the changes made by the user into the final arraylist by overwriting the original input file.

UML



Assignments

**Erik Sklocic:** Assigned to managing the GitHub and making the functions of addMovie, showMovie, and countMovies.

**Joseph Jennings:** Assigned to initializing and designing the menu.java, making the DataRead.java class along with its functions of readfile and save.

**Sonal Sinha:** Assigned to making the functions of dsplyMovie, editDesc, and editDate.

**Neha Metlapalli:** Assigned to making the movie class along with its functions and variables, and tasked with misc help around the project.

Test Cases

**Test Case 1:**

Expected output:

Dsply = displays the coming movies and showing movies in a large list

addM = Adds movie “Show Down” with the desc of “Action” with the release date of 4/5/2000 and a arrival date of 4/1/2000

editDte = Edits the date of movie “Show Down” release date to 4/6/2000

editMDesc = Edits the movie description of “Show Down” to “Action/Drama”

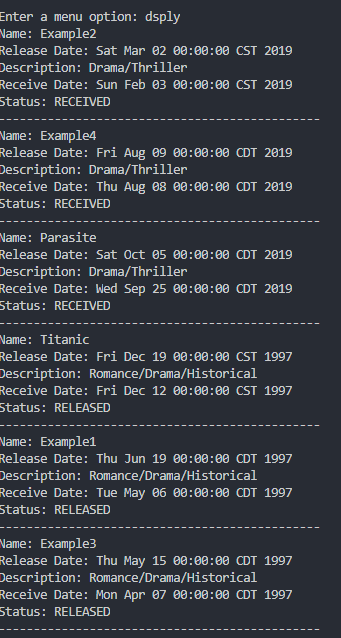
showM = Sets movies with a release date of 4/5/2000 to the showing list

numMDte= Shows movies before the given date of 4/6/2000 in the COMING LIST which is none

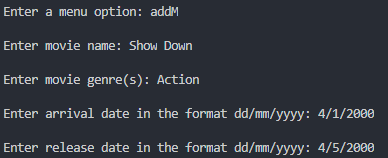
Save = saves the current state of the file which should have in order of Parasite, Example2, Example 4, Titanic, Example1, Example3 and Show Down

Actual Output:

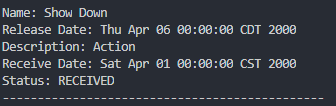
Dsply:



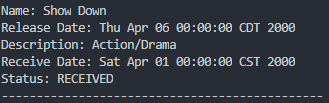
addM:



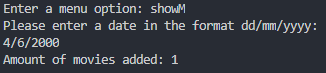
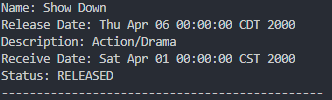
editDte:



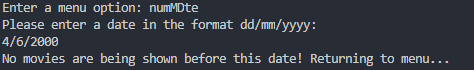
editMDesc:



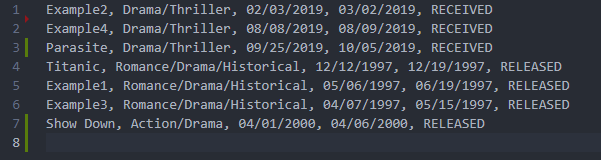
showM:



numMDte:



Save:



**Test Case 2:**

**This test case will go off the same file from the last test!**

Expected output:

Dsply = displays the coming movies and showing movies in a large list

addM = Adds movie “Kick out” with the desc of “Action” with the release date of 3/5/2000 and a arrival date of 4/1/2000 which will error out and request a new set of dates which will be 3/5/2000 and 3/1/2000 and work

editDte = Edits the date of movie “Kick out” release date to 2/5/2000 which will error out and kick the user back to menu, retry with 3/6/2000 and work

editMDesc = Edits the movie description of “Kick out” to “Action/Adventure”

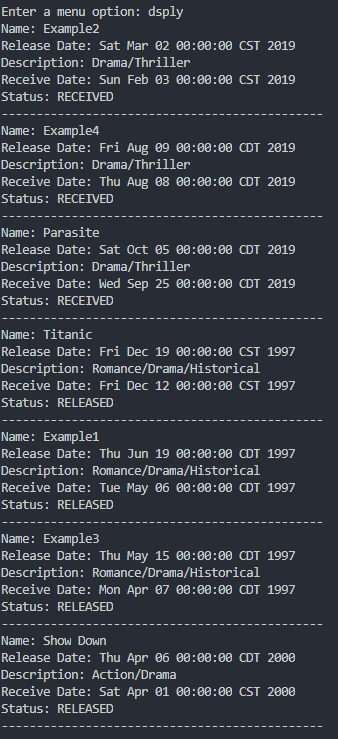
showM = Sets movies with a release date of 4/5/2020 to the showing list which won't add any

numMDte= Shows movies before the given date of 4/6/2020 in the COMING LIST which is ALL of the coming movies

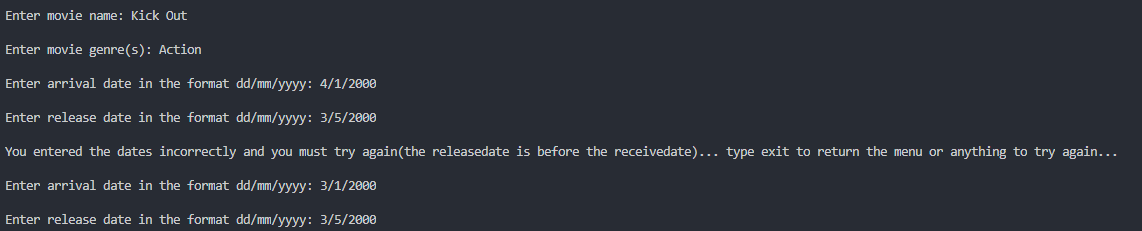
Save = saves the current state of the file which should have in order of Parasite, Example2, Example 4, Kick Out, Titanic, Example1, Example3 and Show Down

Actual Output:

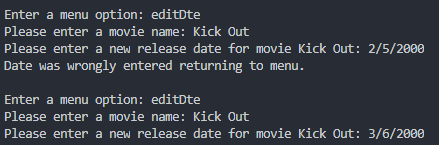
Dsply:



addM:



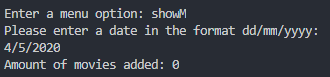
editDte:



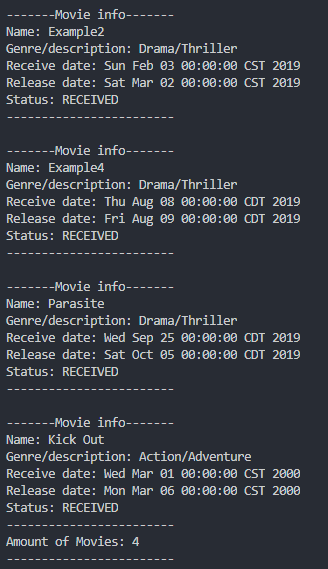
editMDesc:



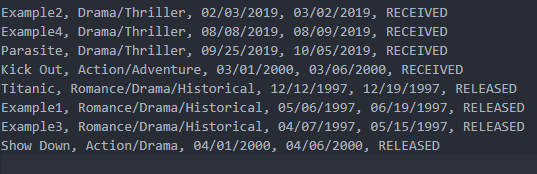
showM:



numMDte:



Save:



Improvements

The improvements that could have been made in this project can vary within the code itself. An example of this is the orderList() method which uses Indexing. The method of sorting we used could have been replaced with an Iterator one instead, making the overall time complexity more efficient. The addMovie method for the menu is quite long for a method that adds a movie to the list. Maybe if more time was spent on thinking on how to add movies to the movielistCOMING list, our sorting algorithm could have been improved in both speed and ease of understanding. Another improvement that could be made is in the DataRead file, it has a few cases where the method uses a for loop where an iterator while loop could have replaced them for improved speed. The dsplyMovie in menu.java also suffers from the same fate, in which an iterator while loop could have been used for faster iterations of the list.