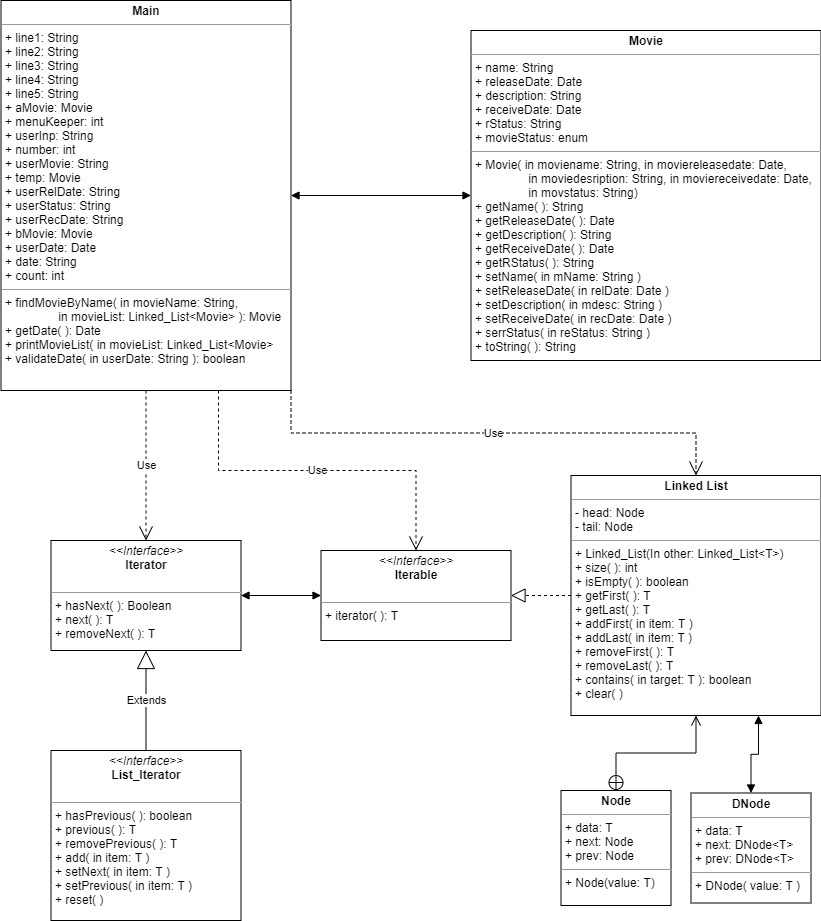
# Movie Management System

Adam Carmichael

Evan Colyer

Lindsey Erwin



The program starts by reading a list of movies from an input file and creating a Movie object for each movie. These objects include the title, release date, description, receive date, and current status (received or released). There are two separate doubly linked lists that the objects are then sorted into, the showingList, and the comingList. If the release date is prior to the current date the movie is added to the showingList, if the release date is after the current date it is added to the comingList. Once the initial lists are constructed the user can choose from a list of menu items and perform operations including listing the showing movies, listing the coming movies, adding a movie, editing a release date, editing a movie description, counting the number of movies with release dates before a given date, and saving your changes.

Doubly linked lists are used to store the objects so that they can be easily traversed, inserted, and deleted. There are several instances in which each list will undergo change. When a new movie is added to the coming list the movie must be inserted into the correct place so that the movies stay in order by release date. When the current date reaches the date of a movie in the comingList, that movie is removed from the comingList and added to the showingList.

An improvement that could be made to this program is putting the menu functions in a different class as methods. Because most of the code needed for each menu option is written directly in the Main class the class is quite long making it hard to read at times. We also have several instances where a movie object has to be created in order to perform a menu function. Creating a method for this may simplify the code and help with the naming of objects. Another improvement that could be made is including an option to return to the menu if you select the wrong menu item. At the moment you must enter a valid input for the selected menu item in order to return to the menu. An option to exit the selected item would aid in user experience.

Contributions

|  |  |
| --- | --- |
| Evan | Read input file and create linked lists of Movie objects |
| Evan | Menu options |
| Evan | Display showing movies |
| Evan | Display coming movies |
| Evan | Add a movie |
| Evan | Output lists to file |
| Evan | Exit program |
| Evan | Function to get today’s date |
| Evan | Movie class |
| Evan | DNode, Iterable, Iterator, Linked\_List, List\_Iterator |
| Adam | Add a movie |
| Adam | Edit release date of a movie |
| Adam | Edit description of a movie |
| Adam | findMovieByName( ) method |
| Adam | printMovieList( ) method |
| Adam | Order coming list by release date |
| Lindsey | Add a movie |
| Lindsey | Display and count all movies before given date |
| Lindsey | validateDate( ) method |
| Lindsey | UML diagram |
| Lindsey | Report |

Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | Description | Input Date | Expected Result | Actual Result | Pass/Fail |
| 1 | Check response when a movie title is entered that already exists | Menu selection: 3    Movie Title: The Godfather | Output: Movie already exists in list.  Enter new movie title | Output: Movie already exists in list.  Enter new movie title | Pass |
| 2 | Check result when a receive date is entered that is later than the release date | Menu selection: 3    Release Date: 10/09/2021    Receive Date: 10/20/2021 | Output: Invalid entry, receive date must be before release date.  Please make another entry    Movie is not added to coming list | Output: Invalid entry, receive date must be before release date.  Please make another entry    Movie is not added to coming list | Pass |
| 3 | Check result when an invalid date is entered | Menu selection: 4    Date: 10/3 | Output: Enter new date (“MM/dd/yyyy”)    Date is not changed | Output: Enter new date (“MM/dd/yyyy”)    Date is not changed | Pass |