



# Project 1A

02/01/2021

---

Dalton Vining

Shane Callaway

Samuel Maynard

## How we designed the system:

Our main is within the Project\_1 page of our project. Within this page are the methods to read from and save to the txt file. We created our lists on this page, as well as an array to hold coming movies. Additionally our date/time format is created here to properly read from the file and save date data to our link lists properly. The menu, which supports user interaction to display or edit movie objects in various ways, is implemented on this page. The menu choices are:

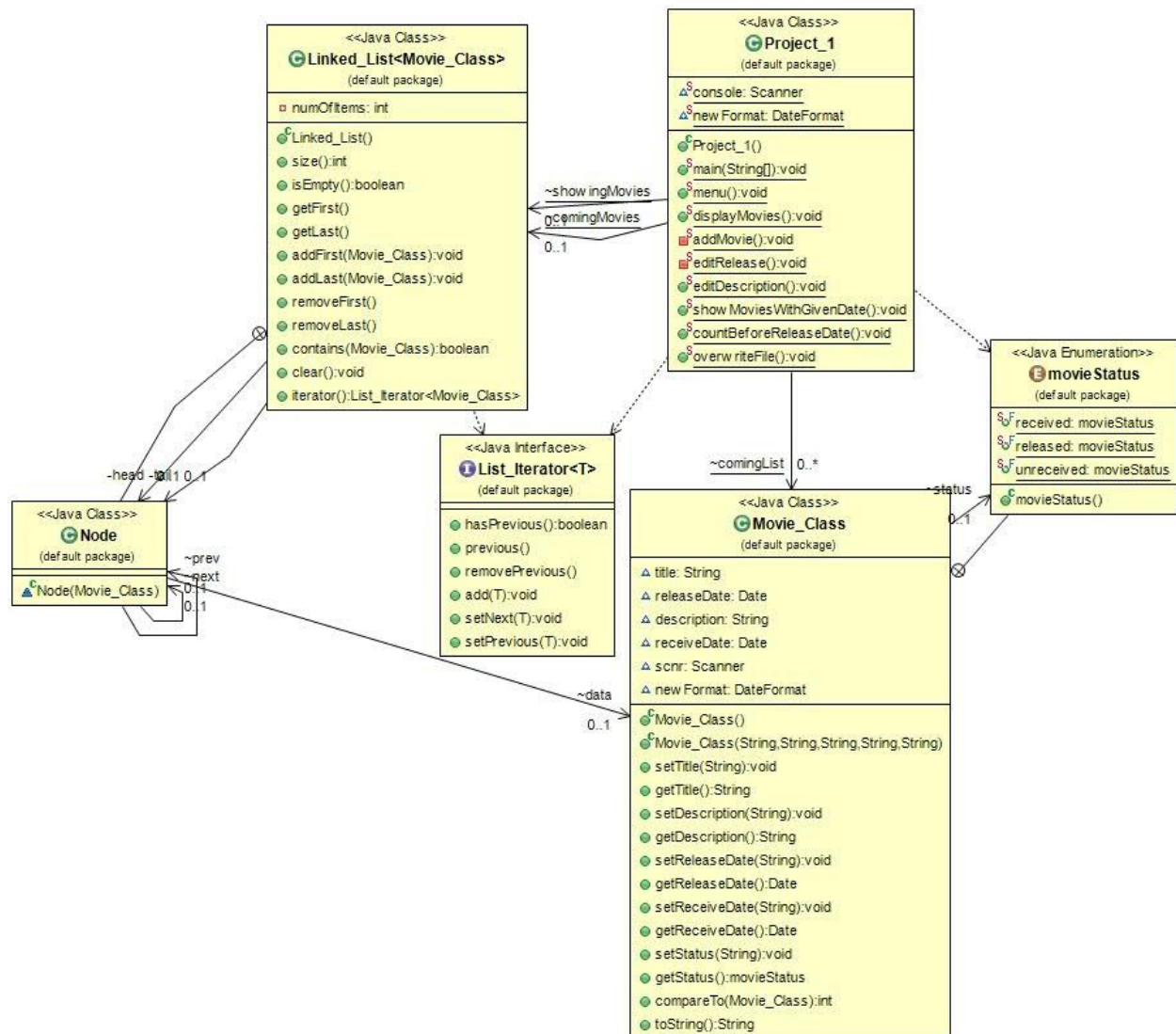
- displaying the movies stored in the linked lists, sorted by the enum status of received or released,
- adding a movie object to the list, the user must enter a release date, title, received date and description for the movie,
- editing the release date of a movie stored in either list
- editing the description of a movie in either list
- displaying all movies with a user specified release date
- counting the number of movies to release before a user specified date
- and finally saving all changes made to the txt file, overwriting what was previously stored on the file.

Outside of Project\_1 are our Iterator, Iterable, and List\_Iterator interfaces which allow us to properly go through each item in our linked list.

We implemented our linked\_list class outside of the Project\_1 page, this of course is responsible for defining the attributes of our linked lists we use in main.

Lastly is the Movie\_Class class, which creates our movie objects which get stored in the correct linked list based off of the enum type attribute. Each movie class object instance is created using a set of strings, which are then converted into the proper variable types via methods within the Movie\_Class class. Dates are formatted so as to display in dd/MM/yyyy format. The class has methods that can be called with an instance of the class to alter or get the data of any of the main attributes of a Movie object including; the title, the release date, the received date, the status, and a brief description.

## UML Class diagram:



## Two test cases:

We have a user-input text file we read from:

*Twilight, 2010-10-20, Drama, 2010-09-20, released;*  
*Twilight 2, 2011-10-20, Drama, 2011-09-20, released;*  
*Twilight 3, 2012-10-20, Fantasy, 2012-09-20, received;*  
*Twilight 4, 2013-10-20, Comedy, 2013-09-20, received;*  
*Twilight 5, 2014-10-20, Sci-Fi, 2014-09-20, received;*

When we implement our function `Display_Movies`, it produces the exact output:

***Showing Movies***

***Twilight, 2010-10-20, Drama, 2010-09-20, released***

***Twilight 2, 2011-10-20, Drama, 2011-09-20, released***

***Coming Movies***

***Twilight 3, 2012-10-20, Fantasy, 2012-09-20, received***

***Twilight 4, 2013-10-20, Comedy, 2013-09-20, received***

***Twilight 5, 2014-10-20, Sci-Fi, 2014-09-20, received***

We edit the description and release date of `Twilight 4` to “Horror” and “2013-11-20” respectively.

Expected/Exact Output:

***Twilight, 2010-10-20, Drama, 2010-09-20, released;***

***Twilight 2, 2011-10-20, Drama, 2011-09-20, released;***

***Twilight 3, 2012-10-20, Fantasy, 2012-09-20, received;***

***Twilight 4, 2013-11-20, Horror, 2013-09-20, received;***

***Twilight 5, 2014-10-20, Sci-Fi, 2014-09-20, received;***

For our second case we have the same list:

***Twilight, 2010-10-20, Drama, 2010-09-20, released;***

***Twilight 2, 2011-10-20, Drama, 2011-09-20, released;***

***Twilight 3, 2012-10-20, Fantasy, 2012-09-20, received;***

***Twilight 4, 2013-10-20, Comedy, 2013-09-20, received;***

***Twilight 5, 2014-10-20, Sci-Fi, 2014-09-20, received;***



Now we will add a movie to our ComingMovie list with the following attributes:

***Name: Test Movie***

***Release Date: 2021-08-26***

***Description: Test***

***Receive Date: 2021-07-26***

Next we will begin showing the movie that releases on ***2012-10-20***,

And now we will count how many movies that are left in the coming list come out before ***2015-01-01***, which produces the expected output:

***The amount of movies coming before 2015-01-01 is 2.***

Expected/Exact Output:

***Twilight, 2010-10-20, Drama, 2010-09-20, released;***

***Twilight 2, 2011-10-20, Drama, 2011-09-20, released;***

***Twilight 3, 2012-10-20, Fantasy, 2012-09-20, **released**;***


***Twilight 4, 2013-10-20, Comedy, 2013-09-20, received;***

***Twilight 5, 2014-10-20, Sci-Fi, 2014-09-20, received;***

*****Test Movie, 2021-08-26,Test, 2021-07-26, received;*****

### **How we contributed:**

Samuel focused mainly on the aspects of the project involved with reading and writing from the txt file. He designed the format for the file to be written in, and how it would be split by the program when reading from the file. The second part of file usage required for this program was saving changes made to any movie object to the file. Sam wrote the method `overwriteFile` to iterate over every data point saved in each linked list and write them to the txt file, overwriting the information previously stored there. On top of the file parts of the program, Sam supported, to a minor degree, Shane and Dalton with their parts, and vice versa, in order to make sure our



parts of the project properly interacted with each other's parts. Finally, we all pitched in to write this project report, with Dalton taking care of formatting, and Shane creating the UML diagram.

Shane worked primarily on the `Movie_Class` class, dealing with the research and implementation of the date type and enum type for variables. Included in this class was the need to convert the taken strings from the constructor into their proper variable types, the ability to call and obtain the release dates and status of each `Movie_Class` object, and the ability to edit descriptions and the status of each `Movie_Class` object. Shane also assisted in making small changes to the other portions of the project as needed, to a minor degree to ensure all portions worked together as needed.

Dalton worked primarily on the menu method and all methods the menu supported, aside from overwriting the text file. Using the `Linked_list` class, `List_iterator`, and `Iterable` interfaces, he traversed, added to, and removed from both linked lists according to what the method needs. A method also needed to print the nodes of both lists in a `toString` method also designed by Dalton. A few methods required simple editing of the nodes data instead of adding or removing.

### **Improvements:**

Our page `Project_1` that contains the `main()` method may be a bit over crowded. Creating classes separately would have been better for code readability, i.e. making the parts of the `Project_1` that read and write from the file into separate classes so that all that happens on the `Project_1` page is calling those classes.

There could also be methods added in to change other aspects of each movie and reorganize lists based on things such as title, or description. A GUI could also be implemented so that a user would have a more clear view of what options and actions they could take.