Faculty of Engineering and Information Technology
Computer Science Department
COMP242 (Project No. 4)

Project Title: Movie Catalog Management System

Overview:

In this project, you will implement a **Movie Catalog Management System** that uses a **hash table** for storing movie information and resolves hash collisions using an **AVL tree**. The system will support reading movie data from a file, inserting new movies, updating existing movies, deleting movies, and saving the updated catalog to a file. The movies will be stored as key-value pairs, where the key is the movie title and the value is the movie details (such as description, release year, and rating).

Additionally, you will implement a **JavaFX GUI** that includes:

- A **TableView** for displaying the movie catalog.
- **Menus** for adding, updating, and deleting movies (See the Following Description).

Requirements:

1. Movie Class:

You need to implement a Movie class that represents a movie and contains the following attributes:

- **Title (String)**: The name of the movie.
- **Description (String)**: A brief description of the movie.
- **Release Year (int)**: The year the movie was released.
- **Rating (double)**: The rating of the movie will be between [0.0 and 10.0]. For example, 4.5 stars

2. MovieCatalog Class:

You will implement the MovieCatalog class, which manages the movie collection using a hash table. The hash table will store **movie titles (String)** as the keys and an **AVL tree** as the value for each movie entry.

The **MovieCatalog** class should have the following functionality:

- constructor for the size — allocate (size): Start with a table size that is a prime number. This minimizes the chances of hash collisions.
- insert to hash — put(Movie movie): Adds a new movie to the catalog or updates an existing movie.
- search in hash — get(String title): Retrieves the movie corresponding to the provided title.
- search and delete — erase(String title): Removes a movie with the specified title from the catalog.
- traverse the hash and save to file — saveMoviesToFile(): Saves the current catalog to a file (movies.txt).
- insert to hash from file — loadMoviesFromFile(): Loads movies from the file into the catalog.
- hash and collisions functions — hashFunction(String title): A simple hash function to determine the index for the movie title in the hash table. (**You are autonomous. Don't ask us about the function**) Linear brobing
- clear the hash — deallocate(): Frees all memory taken up by the hash table and the AVL trees.
- other functions

rehashing formula **Note: Adjust the hash table size dynamically when the average height of all AVL trees exceeds 3. For example, if the average height of the trees is greater than 3 due to frequent collisions, the hash table should be resized to reduce the height of the trees and improve performance.**

3. AVL Tree Class:

The **AVL tree** will be used to resolve hash collisions. Each node in the AVL tree should represent a movie, and the tree should be balanced for efficient search, insert, and delete operations.

The **AVL tree** should support:

- insert(Movie movie): Inserts a movie into the AVL tree.
- delete(String title): Deletes a movie based on the title.
- Other functions

4. File I/O:

Each movie should be represented by **4 lines in the file**, separated by blank lines.

```
Title: Toy Story
Description: A cowboy doll is profoundly challenged when his owner replaces....
Release Year: 1995
Rating: 8.3


Title: Finding Nemo
Description: After his son is captured in the Great Barrier Reef and taken to Sydney,......
Release Year: 2003
Rating: 8.1
...
```

5. JavaFX GUI:

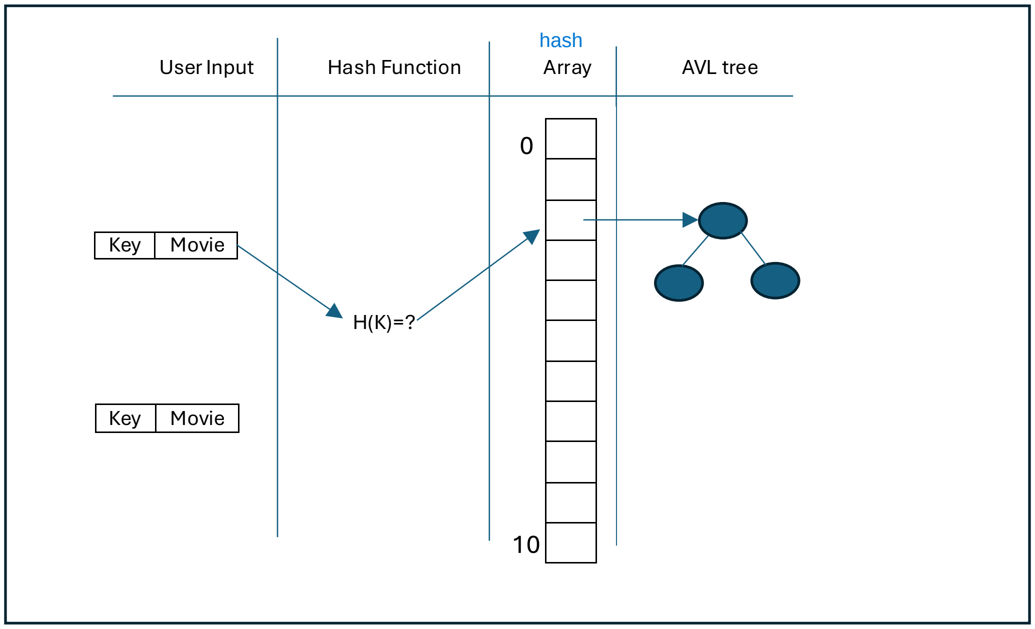
You need to implement a **JavaFX GUI** for interacting with Movie Catalog. The GUI should include the following components:

- **TableView**: Display the movie catalog in a table with columns for Title, Description, Release Year, and Rating.
  - The table should allow users to view, add, and delete movies.
  - The table should be updated automatically when movies are added, deleted, or updated.
- **Date picker:** for selecting the release year
- **Menus**: Implement a menu bar with the following options:
  - **File Menu**:
    - **Open**: Load movies from a file (movies.txt).
    - **Save**: Save the current movie catalog to a file (movies.txt).
    - **Exit**: Exit the application.

ask user if he wants to save to file before exiting

- **Movie Menu**:
  - **Add Movie**: Allows the user to add a new movie to the catalog.
  - **Update Movie**: Allows the user to update the details of an existing movie.
  - **Delete Movie**: Allows the user to remove a movie from the catalog.
  - when search change current movie
  - **Search Movie**: Allows the user to search for a movie by **title or release year**. If multiple results are found, the search operation will display the results in the table view.
  - we need to find a sorting formula
  - **Print sorted**: Retrieves movies sorted by their titles **in either ascending or descending order**. Use the **Next** and **Previous** buttons to navigate through the hash table. Pressing Next will display the movies' information and the corresponding AVL tree height for the next hash cell in the table view. Pressing Previous will display the movies' information and the tree height for the previous hash cell in the table view.

    Note: The sorting order (ascending or descending) is based on the user's selection from the combo box
  - print the AVL tree in order
    i think we need to sort the movies in the tree using the rating so they will be balanced based on it
  - **Print Top and Least Ranked Movies:** print the top and lowest ranked movie(s) in each AVL tree and their corresponding star ratings.



System Overview: Use of AVL Tree for Separate Chaining Instead of Linked List

The hash will be an array of references to AVL tree
THERE WILL BE NO DATA IN THE HASH MAP ONLY REFERENCE

Please note the Followings:

I. Your application should have all functionalities working properly.

II. There must be adequate documentation and comments in the code (e.g., functions, loops, etc.).

III. Your code should follow coding conventions (e.g., spacing, indentation, etc.) and guidelines (**Remember COMP2311**).

IV. This is an individual project. Disciplinary action will be taken against those who cheat. Additionally, **the use of AI tools for generating solutions or copying from websites is strictly prohibited**. Students found in violation of these policies will face severe consequences. It is crucial to ensure that all work submitted is your own and adheres to the guidelines provided for this assignment.

V. Please submit your Java files (java) and corresponding test text files (txt) via the ITC by Saturday, 4/ 1/2025, at 11:00 PM. **Late submissions will not be accepted under any circumstances.**