Academy Awards Database

Mike and Jacob

Michael Hoover

Jacob Hutchins

Due date: 12/11/19

**Table of contents**

1. Program description 3

2. Overall Software Architecture 4

a. Tree class 4

b. Actor/Picture tree 4

c. Menu 5

3. Input Requirements 6

4. Output Requirements 6

5. Problem Solution Discussion 7

6. Data Structures 7

7. User Interface Scheme 7

1. **Program Description**

This program uses a database system to store, manipulate, sort, and search through information about the Academy Awards. There are two .csv files of interest in this program. The first file contains data about the actors and actresses including their name, their award, whether or not they won the award, the name of the film, and the year of the film. The second file contains data about the films including their name, year, nominations, genre, etc. The program inputs these files and reads in the necessary information as sorted databases. For both databases, the program allows for the modification of individual records and fields, sorting by a field, searching through a field (either a full or partial search), and the outputting of a correctly formatted file containing either database after they have been modified.

1. **Overall Software Architecture**

**a. Actor tree**

Description: Stores actor entries in a tree data structure. Includes variables for each field. Each actor node contains int year, string award, bool winner, string name, and string film.

+printTreeInOrder: Outputs all the records of tree in easily readable format, sorted accordingly.

+buildTreeFromFile: Traverses csv file and assigns field values to corresponding variables. For example, if a line in the actor csv file is “1999, Oscar, 0, Tom, Good Movie”, the corresponding actor node’s fields will be assigned as follows: year = 1999, award = “Oscar”, winner = false, name = “Tom”, film = “Good Movie”.

Receives: File name

*Complete Searches*: Searches for and displays records with exact matches.

Receives: Search value

+compYear

+compAward

+compName

+compFilm

*Partial Searches*: Searches for and displays records with partial matches. For example, if the value being searched is “om” for the name field, a record with the name “Tom” will be found and displayed.

Receives: Search value, field to search

+partYear

+partAward

+partName

+partFilm

*Sort Functions*: Reorganizes tree based on a sortable field.

Receives: Field to search

+sortYear

+sortAward

+sortName

+sortFilm

+addToTree: Adds node, automatically puts node in correct spot.

Receives: Pointer to node to add

+addActNode: Adds node, automatically puts node in correct spot.

Receives: year, award, winner, name, film

+removeActNode: Removes node from tree, automatically resorts tree accordingly.

Receives: Pointer to node to remove

+findAndRemoveNode: Removes record based off user choice.

Receives: Number corresponding to actor record to remove.

+modActRecord: Searches for records, allows user to select which record to modify, allows user to change specific field.

Receives: Number corresponding to actor record to modify.

+writeToFile: Using current state of the tree, create a csv file that follows correct standard format for actors.

Receives: File name

+addToList: Adds node to search results.

+printList: Displays numbered list of search results.

**b. Picture tree**

Description: Stores picture entries in a tree data structure. Includes variables for each field. Each picture node contains int year, string award, bool winner, string name, and string film.

+displayTree: Outputs all the records of tree in easily readable format, sorted accordingly.

+buildTreeFromFile: Traverses csv file and assigns field values to corresponding variables. For example, if a line in the actor csv file is “1999, Oscar, 0, Tom, Good Movie”, the corresponding actor node’s fields will be assigned as follows: year = 1999, award = “Oscar”, winner = false, name = “Tom”, film = “Good Movie”.

Receives: File name

Returns: Boolean indicating success or failure

*Complete Searches*: Searches for and displays records with exact matches.

Receives: Search value

+completeNameSearch

+completeYearSearch

+completeNomSearch

+completeRatingSearch

+completeDurSearch

+completeG1Search

+completeG2Search

+completeRelSearch

+completeMetaSearch

*Partial Searches*: Searches for and displays records with partial matches. For example, if the value being searched is “om” for the name field, a record with the name “Tom” will be found and displayed.

Receives: Search value

+partialNameSearch

+partialYearSearch

+partialNomSearch

+partialRatingSearch

+partialDurSearch

+partialG1Search

+partialG2Search

+partialRelSearch

+partialMetaSearch

*Sorting Functions*: Reorganizes tree based on a sortable field.

+sortByName

+sortByYear

+sortByNom

+sortByRating

+sortByDur

+sortByG1

+sortByG2

+sortByRel

+sortByMeta

+addToTree: Adds node, automatically puts node in correct spot.

Receives: Pointer to node to add

+addToTree: Adds node, automatically puts node in correct spot.

Receives: name, year, nominations, rating, duration, genre 1, genre 2, release, meta

+removeRecord: Removes node from tree, automatically resorts tree accordingly.

Receives: Pointer to node to remove

+removeRecordSearch: Removes record based off user choice.

Receives: Number corresponding to actor record to remove.

+modActRecord: Searches for records, allows user to select which record to modify, allows user to change specific field.

Receives: Number corresponding to picture record to modify.

+saveTreeToFile: Using current state of the tree, create a csv file that follows correct standard format for actors.

Receives: File name

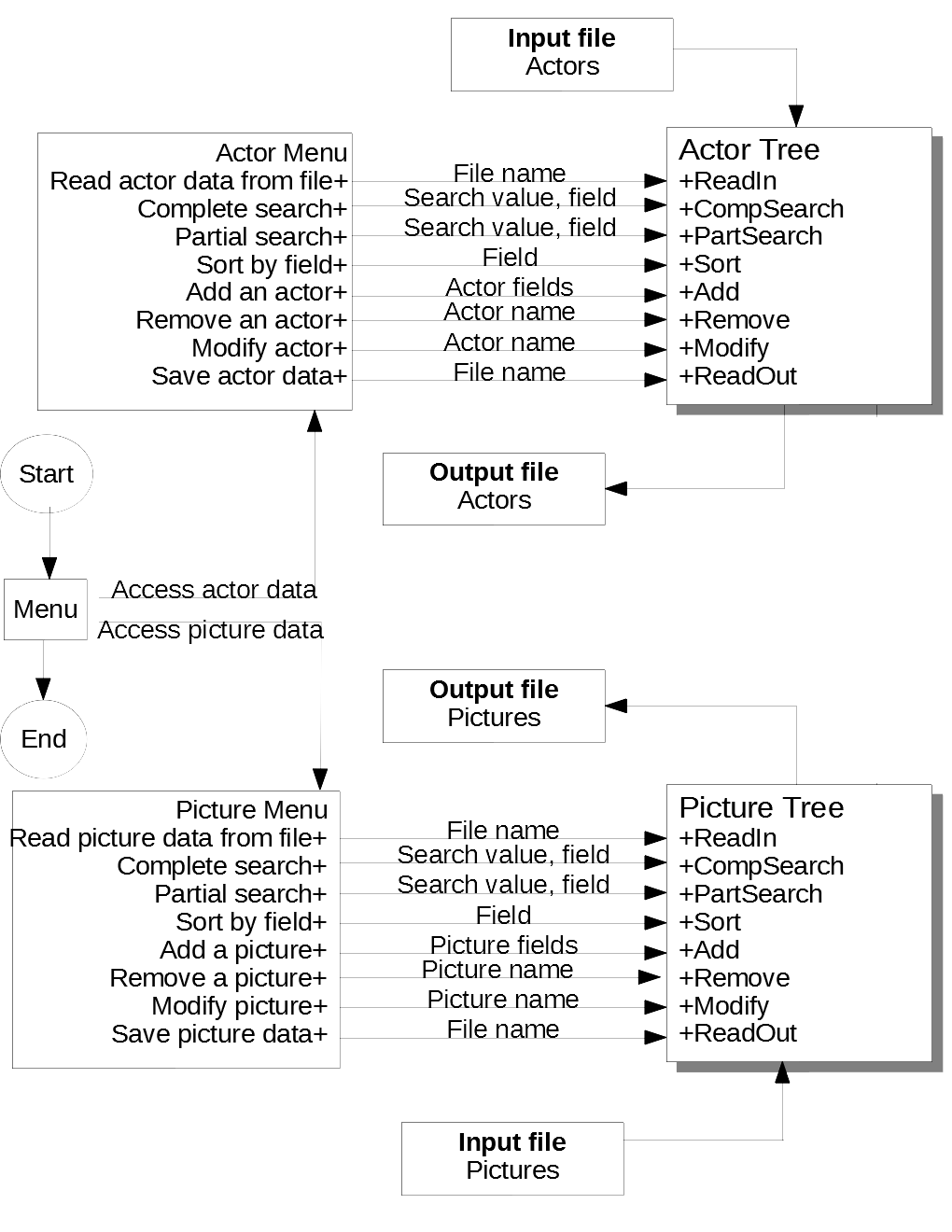
**c. Menu**

+mainMenu: Displays main menu. Allows user to access actor data, access picture data, or exit.

+actor/pictureMenu: Displays actor/picture menu. Allows user to read actor/picture data from a file, search (complete or partial), sort, add actor/picture, add, remove, modify, save actor/picture data to a file by using the ActorTree/PictureTree class.

+actor/picture search menus: Displays the appropriate search menu (complete or partial), allows the user to choose the field to search and input the value to search.

+actor/picture sort menu: Displays sort menu, allows user to select which field to sort.



**3. Input Requirements**

\*The user input for navigating the menu is defined in 7. User Interface Scheme

The buildTreeFromFile function for both the actor and picture tree takes in a file name and a csv file. The file name must be in the form {name}.csv. The csv file is formatted with each field value separated by a comma and each record separated by a new line. The type of each value in the file must match the type of the corresponding field.

Actor csv file:

Format:

*Year, Award, Winner, Name, Film*

Example:

*1999, Oscar, 1, Tom, Movie*

*2002, Best Lead Actor, 0, Bob, Another Movie*

*…*

Picture csv file:

Format:

*Name, Year, Nominations, Rating, Duration, Genre1, Genre2, Release, Metacritic, Synopsis*

Example:

*Movie, 1999, 12, 8.5, 120, Main Genre, Sub Genre, November, 85, Description of film…*

*…*

**4. Output Requirements**

\*Screen outputs in the menu are defined in 7. User Interface Scheme

The displayTree function displays an appropriately sorted list of each record in the tree. For example:

*Year Award Winner Name Film*

*1999 Oscar Won award Tom Movie*

*2002 Best Lead Actor Did not win Bob Another Movie*

*…*

The complete and partial search functions display a numbered list of each record that matches the search. For example:

*Year Award Winner Name Film*

*1. 1999 Oscar Won award Tom Movie*

*2. 2002 Best Lead Actor Did not win Bob Another Movie*

*…*

The Sort function displays the newly sorted data with the DisplayTree function.

The Add function displays the newly added record.

The Modify and Remove functions display a menu to select which record to manipulate with one of the search functions.

The saveTreeToFile function writes to a csv file that is formatted in the same way described in 3. Input Requirements.

**5. Problem Solution Discussion**

Our approach involves an abstract database class that allows inherited classes (i.e. the actor or picture class) to define a sufficiently operational binary search tree. The menu function implements an actor and picture tree to be able to use each database as needed. Other functions can traverse the tree in order or execute a binary search on the data, depending on the requirements of the specific function. When executing a partial search the program builds and stores a linked list of the results to be accessed at a later time. This technique allows the user to search the data and then select an element from the results to modify or delete.

**6. Data Structures**

We chose to use the binary search tree as our data structure for two reasons: 1) it allows efficient searching of data; 2) ner elements can be added without having to move the storage location of existing elements. We considered two alternatives, a linked list and an array. The drawback of a linked list is it only allows sequential searches of its elements, a slower technique than a binary search. We decided that due to the large size of the databases in this project, an array would be less than ideal due to the exorbitant number of operations needed to add and delete elements. Overall, we felt a binary search tree was the best option.

**7. User Interface Scheme**

The following lists all the options for the main menu and each submenu. The menu is navigated by entering an integer to choose which option to access.

1 Access actor/actress data

* 1. Read actor data from a file

* 1. Complete search

Choose a field to search:

* + 1. *Year*
    2. *Award*
    3. *Name*
    4. *Film*
    5. *Back*
    6. *Exit program*

Enter your choice (1-6)

* 1. Partial search

Choose a field to search:

* + 1. *Year*
    2. *Award*
    3. *Name*
    4. *Film*
    5. *Back*
    6. *Exit program*

Enter your choice (1-6)

1.4 Sort by field

Choose a field to sort by:

* + 1. *Year*
    2. *Award*
    3. *Winner*
    4. *Name*
    5. *Film*
    6. *Back*
    7. *Exit program*

Enter your choice (1-6)

1.5 Add an actor

1.6 Remove an actor

1.7 Modify actor

1.8 Save actor data

1.9 Back

1.10 Exit program

Enter your choice (1-10)

1. Access picture data
   1. Read picture data from a file

2.2 Complete search

Choose a field to search:

* + 1. *Name*
    2. *Year*
    3. *Nominations*
    4. *Rating*
    5. *Duration*
    6. *Genre 1*
    7. *Genre 2*
    8. *Release*
    9. *Metacritic*
    10. *Back*
    11. *Exit program*

Enter your choice (1-11)

2.3 Partial search

Choose a field to search:

* + 1. *Name*
    2. *Year*
    3. *Nominations*
    4. *Rating*
    5. *Duration*
    6. *Genre 1*
    7. *Genre 2*
    8. *Release*
    9. *Metacritic*
    10. *Synopsis*
    11. *Back*
    12. *Exit program*

Enter your choice (1-12)

* 1. Sort by field

Choose a field to sort by:

* + 1. *Name*
    2. *Year*
    3. *Nominations*
    4. *Rating*
    5. *Duration*
    6. *Genre 1*
    7. *Genre 2*
    8. *Release*
    9. *Metacritic*
    10. *Back*
    11. *Exit program*

Enter your choice (1-11)

2.5 Add a picture

2.6 Remove a picture

2.7 Modify picture

2.8 Save picture data

2.9 Display picture data

2.10 Back

2.11 Exit program

Enter your choice (1-10)

3 Exit program

Enter your choice (1-3)