Academy Award Database Design Doc

# Objective

Design and implement a database system composed of two binary search trees: one for academy award winners listed in the actor\_actress.csv and one for movies in the pictures.csv files.

## Database files and fields

Each csv file holds the information that will be used for the database system. These database files have a series of records that are represented by comma separated values. Each record will need to have several fields.

The **actor-actress.csv** contains the fields Year, Award, Winner, Name, and Film:

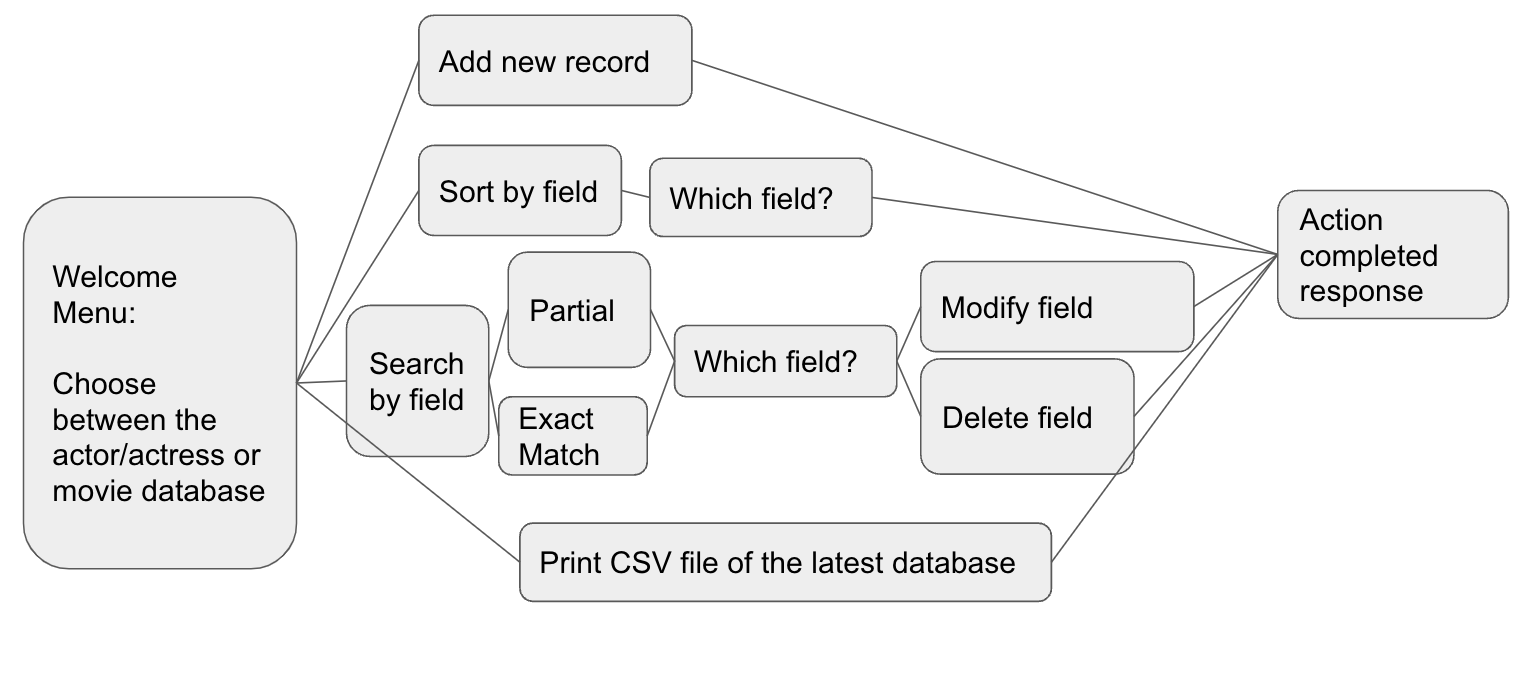
* **Year** – An integer value that represents a four-digit year. This is when the actor or actress won the award.
* **Award** – One of four strings:
  + Actress in a Leading Role
  + Actor in a Leading Role
  + Actress in a Supporting Role
  + Actor in a Supporting Role
* **Winner** – A boolean value. 0 represents a loss for the award and 1 represents a win for the award.
* **Name** – A string value that consists of the first and last name of the actor or actress.
* **Film** – A string value that consists of the name of the film.

The **pictures.csv** contains the fields name, year, nominations, rating, duration, genre1, genre2, release, metacritic, and synopsis.

* **name** – A string that represents the name of the film.
* **year** – An integer value that represents a four-digit year. This is when the award was won.
* **nominations** – An integer value that represents the number of nominations the actor or actress received.
* **rating** – A two-digit double value that represents the movie’s rating.
* **duration** – A three-digit integer that represents the running time of the film
* **genre1** – A string that represents the genre for the film.
* **genre2** – A string that represents a secondary genre for the film.
* **release** – A two-digit integer that represents the year the film was released.
* **metacrtic** – A two-digit integer that represents the movie’s metacritc rating.
* **synopsis** – A string that describes the film.

## Front-end design

This database system will include a text-based menu that users can interact with to add a record; modify, delete, sort, and search by field; search by field for a partial or exact match; and print a csv file of the latest database configuration for either database.



Users must select which database they would like to interact with before they can choose another option. Once they choose which database they would like to interact with, the message flow will proceed as follows:

“Please choose the database you would like to view:

1. Actor

2. Movie”  
  
“What would you like to do? Enter a number 1-4 to select one of the options below.

You can:

1. Add a new record

2. Sort records alphabetically by a certain field

3. Search for an entry by a field. This also allows you to modify or delete a particular field.

4. Print a CSV file of the latest database.”

* Add a new record
  + If the actor-actress.csv was selected:
    - What year was the award won? Enter a four-digit integer. For example, “1976”.
    - What was the name of the award?
    - Did the actor/actress win the award?
    - What was the name of the actor/actress?
    - What was the name of the film the actor/actress starred in?
      * The actor/actress has been added to the database
  + If the pictures.csv was selected:
    - What was the name of the film?
    - What year was the award won? Enter a four-digit integer. For example, “1976”.
    - How many nominations did the film receive?
    - What was the rating of the film? Enter a two-digit decimal. For example, “1.4”.
    - What was the duration of the film? Enter a whole number that represents the running time of the film in minutes. For example, “145”.
    - What was the main genre of the film?
    - What was the secondary genre of the film?
    - What year was the film released? Enter a two-digit integer to represent the year. For example, “89”.
    - What was the metacritic review score?
    - Please enter a short synopsis of the film.
      * The picture has been added to the database.
* Sort the database alphabetically by a field
  + Which field would you like to sort by?
    - If the actor-actress.csv was selected:
      * Year
      * Award
      * Winner
      * Name
      * Film
    - If the pictures.csv was selected:
      * name
      * year
      * nominations
      * rating
      * duration
      * genre1
      * genre2
      * release
      * metacritic
      * synopsis
        + Your database has been sorted by [field name].
* Search for a field
  + Would you like to complete a partial match or exact match sort?
    - Partial
    - Exact
    - Which field would you like to search for?  
      If the actor-actress.csv was selected:
      * Year
      * Award
      * Winner
      * Name
      * Film

If the pictures.csv was selected:

* + - * name
      * year
      * nominations
      * rating
      * duration
      * genre1
      * genre2
      * release
      * metacritic
      * synopsis
        + [Return field] What would you like to do?

Search within result

Modify field

Delete field

Main menu

* Print CSV file of the latest database
  + Here is your file: [first line of file, then lines with information from the field separated by a comma, and ending in a newline.

Back-end design

Classes

* **Pictures** – Serves as a blueprint for the Movie object. It contains all the fields from the pictures.csv as member variables:
  + Year
  + Award
  + Winner
  + Name
  + Film
* **ActorActress** – Serves as a blueprint for the ActorActress object. It contains all the fields from the actor-actress.csv as member variables:
  + name
  + year
  + nominations
  + rating
  + duration
  + genre1
  + genre2
  + release
  + metacritic
  + synopsis
* **BST** – The Binary Search Tree class, which contains logic to search and sort data.

## Backend design

The file is read in, split at the comma and each item in the file is assigned as an attribute of the associated class creating objects for every line in the file. All of the objects are added to the BST.

The setters from the objects are used to take the input from the user and add a new database entry.

The search functionality uses a getAllNodes() function from the BST that adds all nodes to a vector so a single node can be accessed. For partial search, it looks for a substring within the full name string and for complete search you just look for the exact user input.

The modify option allows you to set the winner or rating by using the getter function and passing in user input.

The delete function removes the node that was found in search using .Remove() from the BST class.

The sort function sorts from the leftmost sub tree to the right to get the output sorted by name. It uses the .PrintTree() function from the BST.

The saveCSV fucnction asks the user for a file name, creates a file with that file name and writes to the file the header and then all of the nodes that are in the BST, printing out all attributes for each object.