Academy Awards

Trystan Kaes

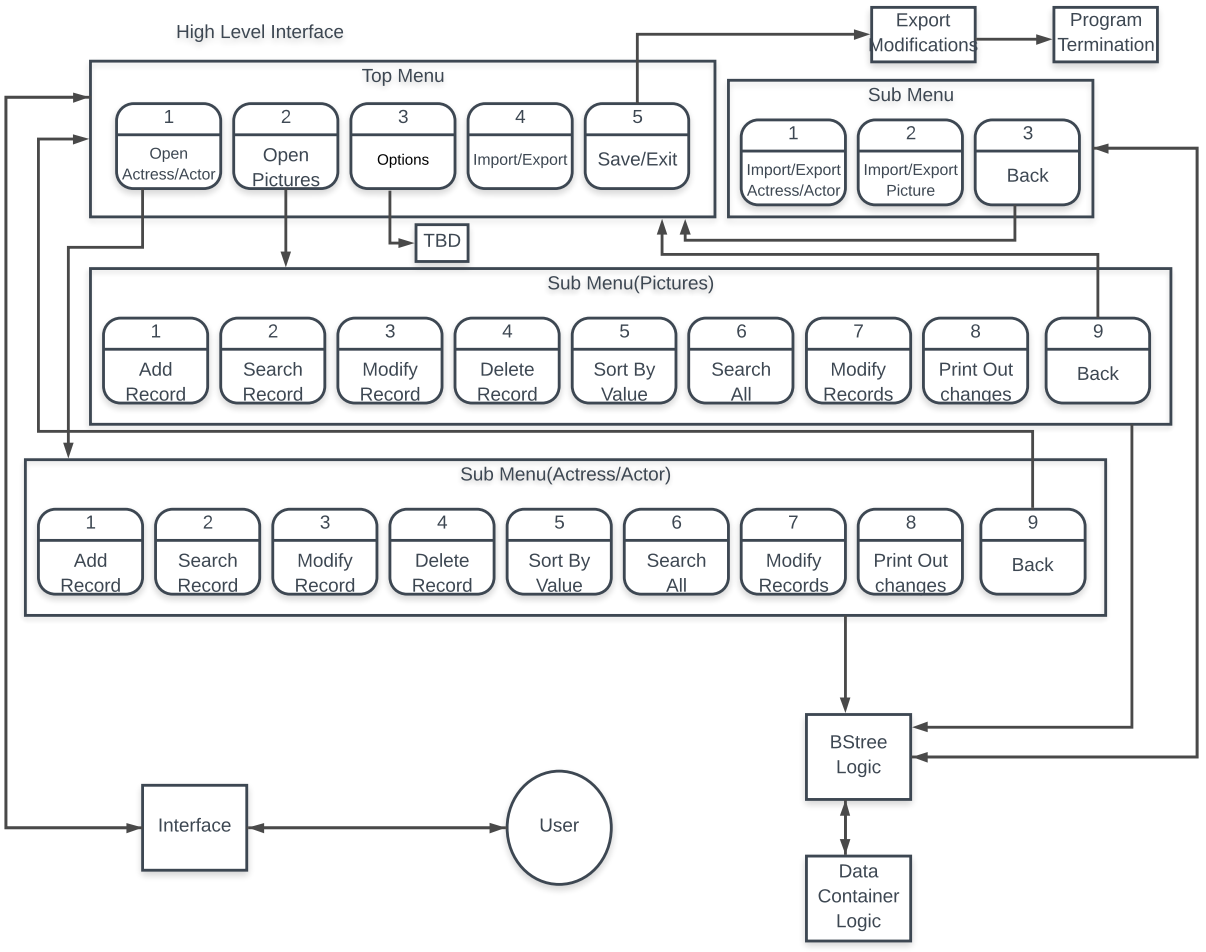
ID#108652414

April 14, 2018

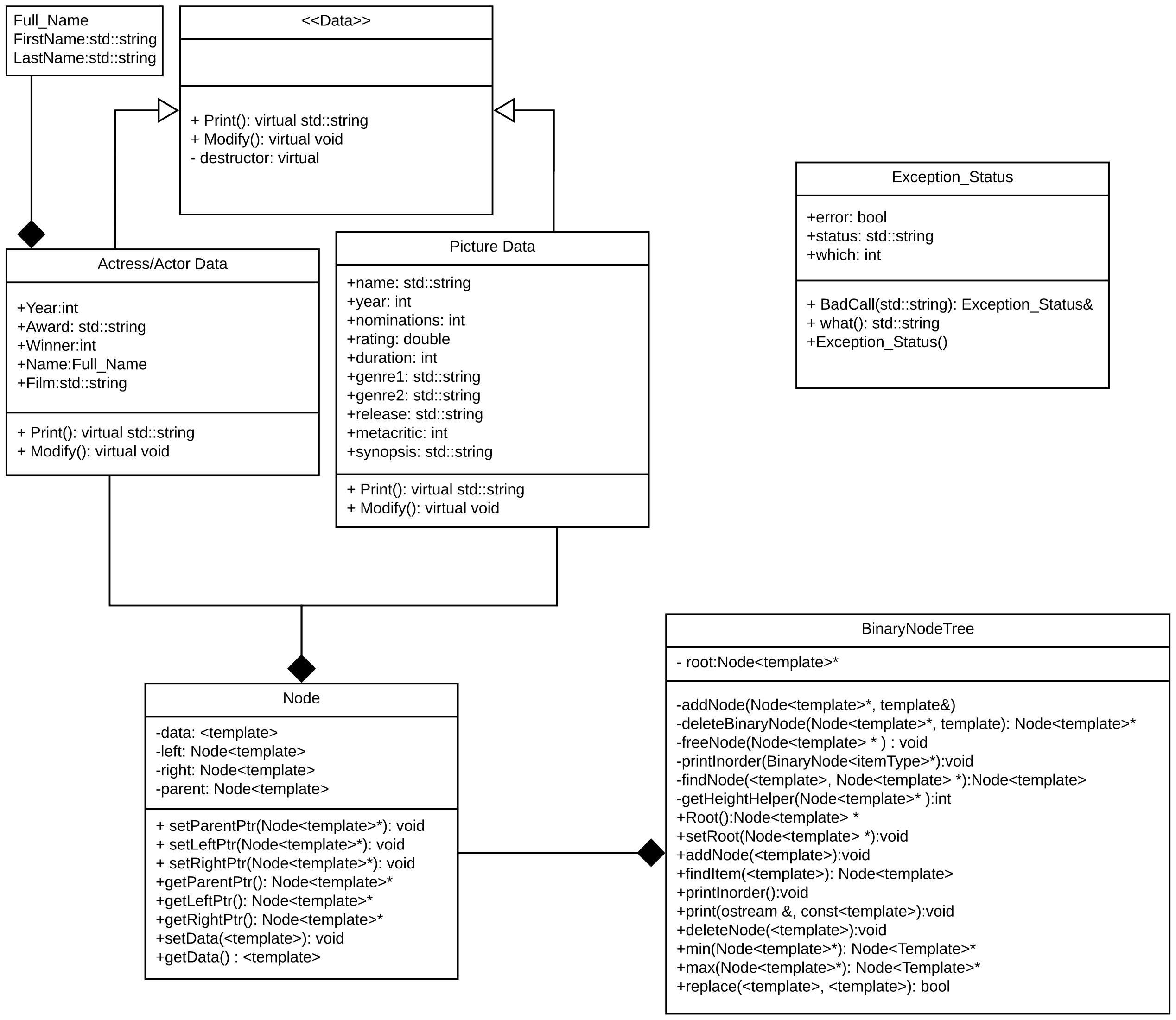
Academy Awards Final

This program is a navigable Academy Awards database to be run within a terminal. The user will be able to run the program, import the actress/actor file or the picture(film) file, and execute a selection of functions to modify the database.

High Level Interface



Class Relationships


Full_Name
FirstName:std::string
LastName:std::string
<<Data>>
+ Print(): virtual std::string
+ Modify(): virtual void
- destructor: virtual
Exception_Status
+error: bool
+status: std::string
+which: int
+ BadCall(std::string): Exception_Status&
+ what(): std::string
+Exception_Status()
Actress/Actor Data
+Year:int
+Award: std::string
+Winner:int
+Name:Full_Name
+Film:std::string
+ Print(): virtual std::string
+ Modify(): virtual void
Picture Data
+name: std::string
+year: int
+nominations: int
+rating: double
+duration: int
+genre1: std::string
+genre2: std::string
+release: std::string
+metacritic: int
+synopsis: std::string
+ Print(): virtual std::string
+ Modify(): virtual void
Node
-data: <template>
-left: Node<template>
-right: Node<template>
-parent: Node<template>
+ setParentPtr(Node<template>*): void
+ setLeftPtr(Node<template>*): void
+ setRightPtr(Node<template>*): void
+getParentPtr(): Node<template>*
+getLeftPtr(): Node<template>*
+getRightPtr(): Node<template>*
+setData(<template>): void
+getData() : <template>
BinaryNodeTree
- root:Node<template>*
-addNode(Node<template>*, template&)
-deleteBinaryNode(Node<template>*, template): Node<template>*
-freeNode(Node<template> *) : void
-printInorder(BinaryNode<itemType>*):void
-findNode(<template>, Node<template> *):Node<template>
-getHeightHelper(Node<template>*):int
+Root():Node<template> *
+setRoot(Node<template> *):void
+addNode(<template>):void
+findItem(<template>): Node<template>
+printInorder():void
+print(ostream &, const<template>):void
+deleteNode(<template>):void
+min(Node<template>*): Node<Template>*
+max(Node<template>*): Node<Template>*
+replace(<template>, <template>): bool

Input Requirements

Inputs will come from both the keyboard and external files. There will be 2 files, one for data on actors and actress' and one for the Pictures(films). Proper error checking will be implemented to reject invalid input from both the keyboard and invalid external file. Each file will be a .csv (comma separated) file with each new object separated by a new line. For menus the user will be able to input only integers. In some cases such as modification and search terms the user will be requested to input full words. These will be received as std::string's and thoroughly error checked to assure fault tolerance. Confirmation fields during the course of the program may request confirmation in the form of the characters 'y' or 'n' to represent "Yes" or "No".

Output Requirements

This program will output formatted listings to the screen and data modifications to an external file. When exported to a file, they will be exported in .csv format as follows: for Actress/Actor — Year, Award, Winner, Name, Film; for Picture — Name, Year, Nominations, Rating, Duration, Genre1, Genre2, Release, Metacritic, Synopsis. The information will be exported as an ofstream. Each movie will be separated by a new line and each field will be separated by a comma.

Problem Solution Discussion

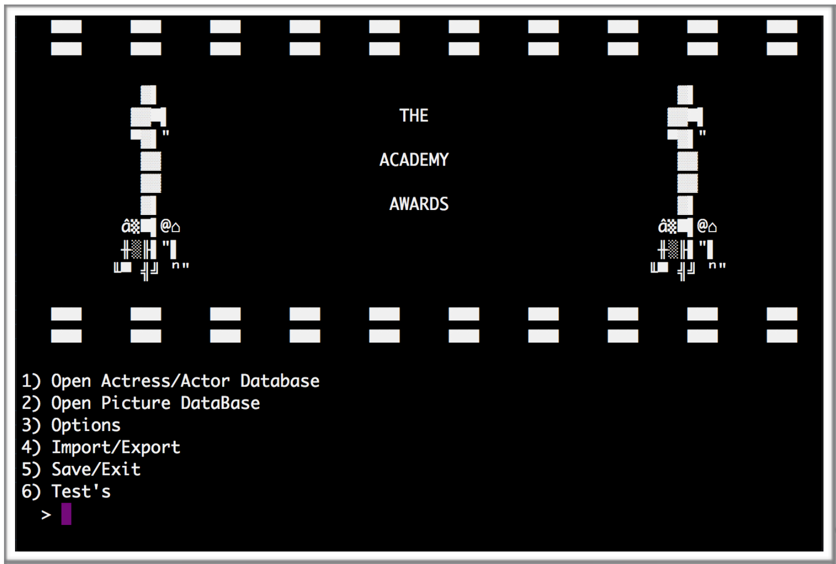
The majority of this database's logic will be provided by functionality built into Binary Search Trees. This functionality will be used to sort, list, search, and delete entries. The nodes within this binary tree will contain class's to hold data. Each of these two possible datatypes (Actress/Actor or Picture) will contain within them functions to get, print, and modify records. Initial data will be passed to them as a constructor. If the user chooses to modify these records, this interface will be executed in the form of an enum(used as a command to specify which element to change), and the new data to replace the current information. In practice, this will look like searching for a specific node within the tree, accessing the data from the node, and calling the modify function with the necessary parameters.

Data Structures

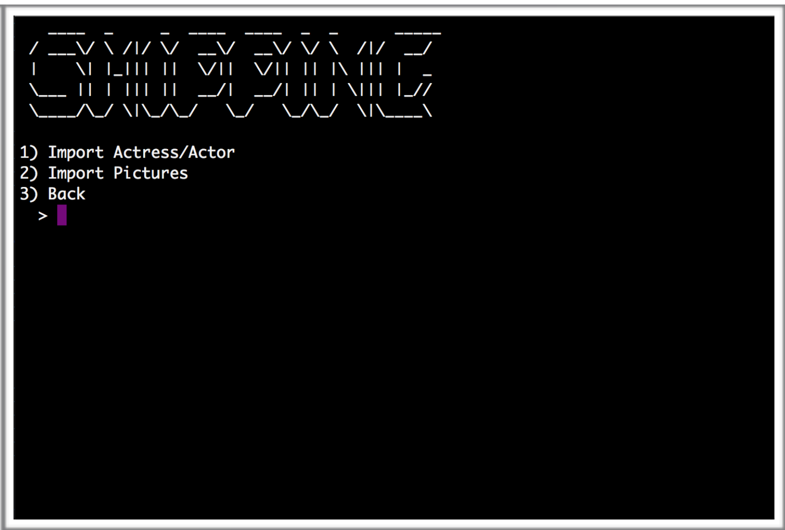
The data structure to be used within this program is a Binary Search Tree. The choice behind using this is less based in efficiency (as given the contents of this program I do not foresee a large need to plan for exceptionally heavily loads) and more based in its simplicity of execution and functionality. The majority of this programs difficulty will be found in searching amidst many containers for specific data. For this, I believe Binary Search Tree's to be perfect.

User Interface Scheme

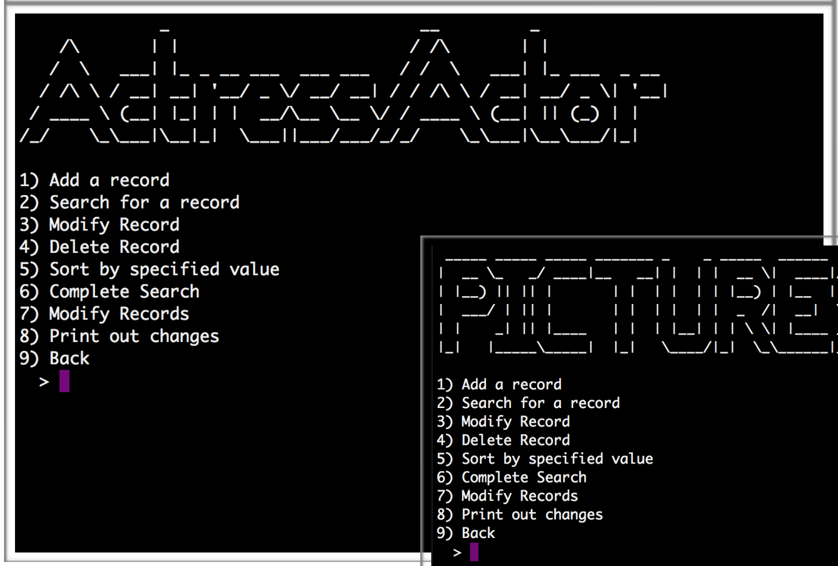
Included below is a sample from the first iteration and mockup of this project. The main menu will include the options below. The option menu, the import/export menu, and both database submenus are as shown below. Editing of fields will be implemented as a series of prompts followed by the necessary confirm or deny to validate user choices.



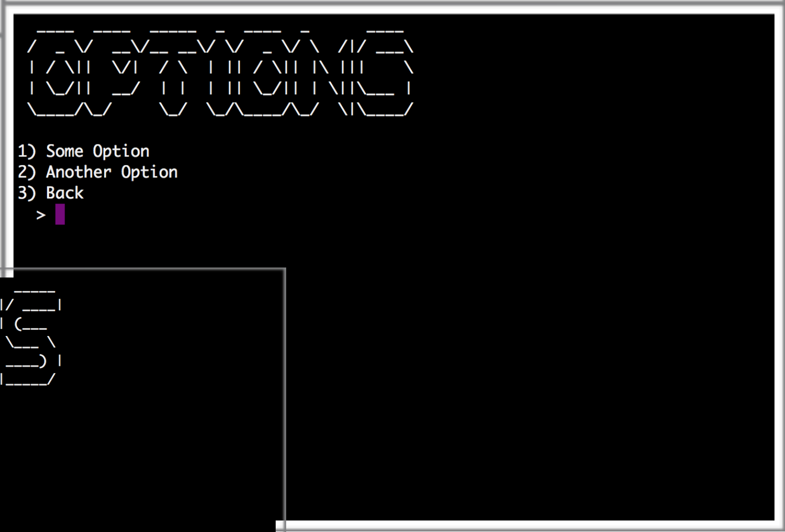
```
1) Open Actress/Actor Database
2) Open Picture DataBase
3) Options
4) Import/Export
5) Save/Exit
6) Test's
  >
```



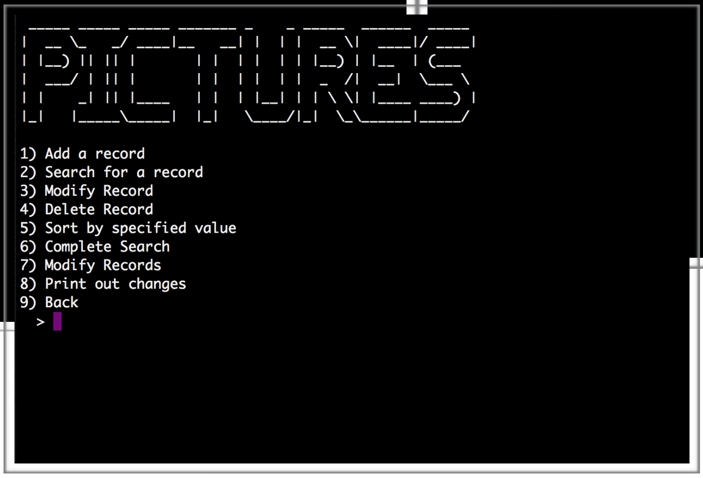
```
1) Import Actress/Actor
2) Import Pictures
3) Back
  >
```



```
1) Add a record
2) Search for a record
3) Modify Record
4) Delete Record
5) Sort by specified value
6) Complete Search
7) Modify Records
8) Print out changes
9) Back
  >
```



```
1) Some Option
2) Another Option
3) Back
  >
```



```
1) Add a record
2) Search for a record
3) Modify Record
4) Delete Record
5) Sort by specified value
6) Complete Search
7) Modify Records
8) Print out changes
9) Back
  >
```