Recommendation Generator for Television Shows and Music (Songs)

Kaitlyn Forsberg

Spring 2020 Object-Oriented Software Design Final Project Report

PROJECT SUMMARY

This program can generate recommendations for television shows or music and it can also store recommendations for the user to view later. In the program, the user will enter whether they want recommendations for television shows or songs. For television shows, the user will answer questions such as the platform that they want to watch the show on (Netflix, Hulu, etc), the number of recommendations they want, the genre of television show that the user is looking to watch, and a series of other questions. Similar questions will be asked if the user is looking for song recommendations. Depending on the responses from the user, the program will print out the number of recommendations requested based on the input provided.

OVERVIEW OF CLASSES

Several classes were created for this project:

1. **Media**: This virtual class contains 4 data members and 5 member functions. It is the parent class to the Shows and Songs classes. Also, it includes a member function that can be used or overridden by either of the subclasses (polymorphism).
2. **Shows**: This class contains 8 data members and 6 member functions.This class includes member functions and exclusive variables for determining the recommendations for television shows.
3. **Songs**: This class contains 6 data members and 5 member functions.This class includes member functions and exclusive variables for determining song recommendations.
4. **Recommendation**: This class contains 3 data members and 6 member functions.This class is the subclass of the Shows and Songs classes. It accesses the protected members of the Shows and Songs classes to make recommendations for the user. Each Recommendation object will be composed of one television show or song recommendation. This additional class will allow for greater organization and reusability.

CLASS DESCRIPTIONS

MEDIA CLASS

Abstract class: Yes

Subclass of: N/A

Composed of: N/A

Alterations post proposal:

- I changed the name of the release_year variable to decade_option because the original name of the variable was not as clear. I also renamed the getter function for this variable for the same reason.
- Some of my getter functions (4 of them) were removed because they were not used. Since the variables in class MEDIA already exist in the main function they did not need to be returned there and the other classes could access the variables via inheritance if needed.

- I also changed my original idea for my friend function, that was originally going to use this class, to a more practical idea so the friend function is now used for the SHOWS and SONGS classes.
- The MEDIA class also became an abstract class, as its purpose is to minimize the reuse of variables and functions in its subclasses. Also since the MEDIA object did not need to be instantiated.

DATA MEMBERS - MEDIA			
Variable Name	**Data Type**	**Static**	**Description**
decade_option	string	no	Holds the option for the user's preference for the time period the show/song came out. Ex: The user could choose songs after 2010 to present.
num_recs	int	no	Holds the number of recommendations the user wants to be given.
media_type	string	no	Holds the type of media that the user wants recommendations for (shows or songs).
genre	string	no	Holds the chosen genre by the user for whichever type of Media.

MEMBER FUNCTIONS - MEDIA					
Signature	**Static**	**Virtual**	**Operator**	**Friend**	**Description**
void set_num_recs(int num_recs_param);	no	no	no	no	Sets value of the num_recs variable.
void set_media_type(string param);	no	no	no	no	Sets value of the media_type variable.
void set_decade_option(string option_param);	no	no	no	no	Sets value of the decade_option variable.
void set_genre(string genre_param;	no	no	no	no	Sets value of the genre variable.
string get_description() const;	no	yes	no	no	Pure virtual function that is overridden by MEDIA's subclasses, which are SHOWS and SONGS.

SHOWS CLASS
Abstract class: No
Subclass of: MEDIA
Composed of: N/A
Alterations post proposal:

- Three data members were added, genres, years, and chosen_platform. These members were added to store additional information about each television show and the chosen_platform variable was used to hold the value of the user's preferred platform.
- One data member, preferred_length, was removed because I chose to include the time period that each show was released instead of the preferred length for determining what recommendations the user would receive since most television shows are the same length (45 min-1 hr).
- I added the add_to_vectors() function to decrease the amount of code reuse in parsing the text file for data.
- I also added the friend function, determine_type, in this class as stated above so that it could be used in a friend/overloading operator function in the RECOMMENDATION.cpp file (description provided in RECOMMENDATION class).

Note: All of the vector<string> variables below are set by reading a tab-delimited text file of potential show recommendations. These variables only store the variables that correspond with the user's entered preferences (genre, platform, etc).

DATA MEMBERS - SHOWS			
Variable Name	**Data Type**	**Static**	**Description**
shows	vector<string>	yes	Holds the values of all of the show names that apply to the preferences that the user has given.
years	vector<string>	yes	Holds the values of all of the years that each show was released, as long as these values correspond with the user's preferences.
ratings	vector<string>	yes	Holds the values of every show's IMDB rating, as long as these values correspond with the user's preferences.
platforms	vector<string>	yes	Holds the values of all of the show platforms, as long as these values correspond with the user's preferences.
genres	vector<string>	yes	Holds the values of all of the show, as long as these values correspond with the user's preferences.

lengths	vector<string>	yes	Holds the values of the approximate length of each show episode in hrs, as long as these values correspond with the user's preferences.
show_count	int	yes	Holds the number of shows that are in the shows variable.
chosen_platform	int	no	Holds the value of the platform chosen by the user (Netflix, Hulu, or either).

MEMBER FUNCTIONS - SHOWS					
Signature	**Static**	**Virtual**	**Operator**	**Friend**	**Description**
void set_shows();	no	no	no	no	Sets the values of the vector<string> variables above (shows, years, etc) based on the user's input.
string get_description const();	no	no	no	no	Overrides virtual function from MEDIA class. Returns a string representation of the user's preferences (genre, platform, etc).
int get_show_count();	yes	no	no	no	Returns the number of shows that were saved to the vector<string> variables above.
void set_chosen_platform(int platform_param);	no	no	no	no	Sets the chosen_platform variable.
string determine_type(Songs song, Shows show);	no	no	no	yes	Friend function used in the RECOMMENDATION.CPP file. It was used to determine the media type the user entered for the operator<< function below since this function was only friends to the RECOMMENDATION class (not MEDIA) so the value could not be accessed. This function is also a friend to the SONGS class.

void add_to_vectors(string, string, string, string, string, string);	no	no	no	no	General function used to reduce code reuse. It is used in set_shows() to add the components of each show into the vector<string> variables above.

SONGS CLASS
Abstract class: No
Subclass of: MEDIA
Composed of: N/A
Alterations post proposal:

- Two data members were added, release_year and genres, because I added additional information to be provided about the song recommendations.
- One data member, count_genre, was removed because it was repeating the same value as the song_count member.
- I added the add_to_vectors() function to decrease the amount of code reuse in parsing the text file for data.
- I removed some setter/getter functions (4 total) that were not needed as the other classes and the main function did not need to access the data if it could not already.
- I also added the friend function, determine_type, in this class as stated above so that it could be used in a friend/overloading operator function in the RECOMMENDATION.cpp file (description provided in RECOMMENDATION class).

Note: All of the vector<string> variables below are set by reading a tab-delimited text file of potential show recommendations. These variables only store the variables that correspond with the user's entered preferences (genre, release date preference, etc).

DATA MEMBERS - SONGS			
Variable Name	**Data Type**	**Static**	**Description**
songs	vector<string>	yes	Holds the values of all of the song names that apply to the preferences that the user has given.
albums	vector<string>	yes	Holds the values of each song's album title (only for the songs that apply to the preferences that the user has given).

artists	vector<string>	yes	Holds the values of each song's artist name (only for songs that apply to the preferences that the user has given).
release_year	vector<string>	yes	Holds the values of each song's release year (only for songs that apply to the preferences that the user has given).
genres	vector<string>	yes	Holds the values of each song's genre (only for songs that apply to the preferences that the user has given).
song_count	int	yes	Holds the number of songs stored in the "songs" data member.

MEMBER FUNCTIONS - SONGS					
Signature	**Static**	**Virtual**	**Operator**	**Friend**	**Description**
void set_songs();	no	no	no	no	Sets the values of the vector<string> variables above (songs, albums, etc) based on the user's input.
int get_song_count();	yes	no	no	no	Returns the value of variable song_count.
string get_description() const;	no	no	no	no	Overrides virtual function from MEDIA class. Returns a string representation of the user's preferences (genre, release date, etc).
string determine_type(Songs song, Shows show);	no	no	no	yes	Friend function used in the RECOMMENDATION.CPP file. It was used to determine the media type the user entered for the operator<< function below since this function was only friends to the RECOMMENDATION class (not MEDIA) so the value could not be accessed. It is also a friend to the SHOWS class.

void add_to_vectors(string, string, string, string, string);	no	no	no	no	General function used to reduce code reuse. It is used in set_songs() to add the components of each song into the vector<string> variables above.

RECOMMENDATION CLASS
Abstract class: No
Subclass of: SONGS AND SHOWS
Composed of: N/A
Alterations post proposal:

- Five data members were removed, because I was originally going to parse the text file for the potential recommendations in this class instead of the SONGS and SHOWS classes, but the process was much easier to do in those respective classes. Those classes were already reading the text file of all of the possible shows and songs so it was easier to put only the relevant shows/songs into a vector rather than put all of the shows/songs in a vector and try to eliminate the uneeded songs/shows afterwards.
- I added a function for the constructor so that object_count could be incremented.
- I switched out the overloading operator function for one that had a more practical use for my program because it allowed me to insert a recommendation into a text file to save it if the user chooses to do so.
- I added the following two member functions: void get_song_description(), void get_show_description() to print out part of the descriptions for the song/show recommmendations. These functions print out the information about each song/show like the name, release year, etc and the get_description() const function from the SHOWS and SONGS classes is different because it prints out information that the user asked for, such as the number of recommendations, chosen genre, chosen time period, etc.

DATA MEMBERS - RECOMMENDATION			
Variable Name	**Data Type**	**Static**	**Description**
object_count	int	yes	Holds the number of RECOMMENDATION objects created. Each object is the equivalent of one recommendation for the user.

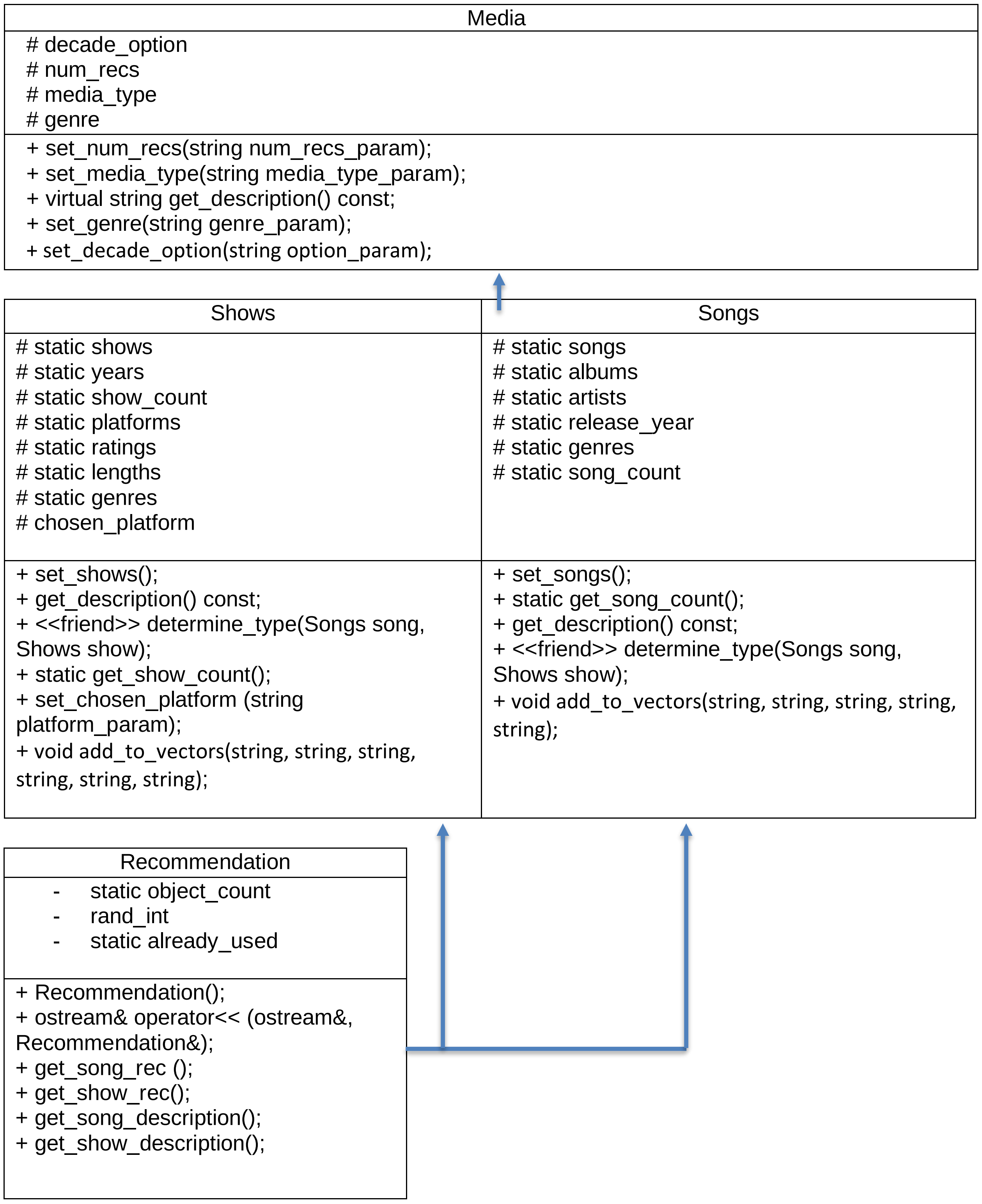
rand_int	int	no	Stores the randomly generated integer that is used to choose what recommendations from the static vector<string> data members of the SHOWS or SONGS classes.
already_used	vector<int>	yes	Stores the values of rand_int that have already been chosen so that the same recommendation would not be given to the user multiple times.

MEMBER FUNCTIONS - RECOMMENDATION					
Signature	**Static**	**Virtual**	**Operator**	**Friend**	**Description**
Recommendation();	no	no	no	no	Constructor for the class iterates the object_count data member every time an object is created.
void get_song_rec();	no	no	no	no	Stores each index of the songs to be used for the recommendations in the already_used variable.
void get_show_rec();	no	no	no	no	Stores each index of the shows to be used for the recommendations in the already_used variable. This function is separate from get_song_rec() to enhance organization.
void get_song_description();	no	no	no	no	Prints out a song recommendation with its name, album, artist, etc.
void get_show_description()	no	no	no	no	Prints out a show recommendation with its name, IMDB rating, etc.
ostream& operator<< (ostream&, Recommendation&);	no	no	yes	no	Overloading operator function used to store the information of a RECOMMENDATION object into a text file.

DEMONSTRATION OF OOP CONCEPTS

Demonstrated in...								
File name	Line #s	Encapsulation	Inheritance	Polymorphism	Static Members	Friend Functions	Overloaded Operators	Text files
Media.h Shows.cpp Songs.cpp main.cpp	36 106-121 69-71 329-336			X				
Media.h	14-17	X						
Shows.h Used in Shows.cpp	17-26	X			X			
Songs.h Used in Songs.cpp	18-23	X			X			
Recommendation.h Used in Recommendation.cpp	17-20	X			X			
Shows.h Songs.h Recommendation.h	15 16 16		X					
Shows.h Songs.h Recommendation.cpp	41 33 128-137					X		
main.cpp	289-326 338-359							X
Recommendation.cpp	99-125						X	

UML DIAGRAM



```
Media
# decade_option
# num_recs
# media_type
# genre
+ set_num_recs(string num_recs_param);
+ set_media_type(string media_type_param);
+ virtual string get_description() const;
+ set_genre(string genre_param);
+ set_decade_option(string option_param);
```

```
Shows
# static shows
# static years
# static show_count
# static platforms
# static ratings
# static lengths
# static genres
# chosen_platform
+ set_shows();
+ get_description() const;
+ <<friend>> determine_type(Songs song,
Shows show);
+ static get_show_count();
+ set_chosen_platform (string
platform_param);
+ void add_to_vectors(string, string, string,
string, string, string);
```

```
Songs
# static songs
# static albums
# static artists
# static release_year
# static genres
# static song_count
+ set_songs();
+ static get_song_count();
+ get_description() const;
+ <<friend>> determine_type(Songs song,
Shows show);
+ void add_to_vectors(string, string, string, string,
string);
```

```
Recommendation
- static object_count
- rand_int
- static already_used
+ Recommendation();
+ ostream& operator<< (ostream&,
Recommendation&);
+ get_song_rec ();
+ get_show_rec();
+ get_song_description();
+ get_show_description();
```