**ECE250-Project 0**

Bingjian Du, b23du

Jan 11th, 2020

1. Overview of Classes

Class: Song

Description: represents a “song”; provides related functions.

Member variables: name and artist.

Member functions:

Song(string name, string artist): build a new song object;

Song(): used to initial an empty “song” for future use;

Operator ==: overload the operator, make it possible to compare two identities of songs;

get\_artist(): return private variable artist;

get\_name(): return private variable name;

~Song():default destructor.

Class: Playlist

Description: represents a list of song

Member variables: “Songs” is a pointer of type Song. Empty is of type std::size\_t used to record the available space for the next Song. This programe use \*(Songs+empty) to add and remove songs on the list. List\_length stores the value of the maximum length of the list.

Member functions:

int add\_Song(Song a): add a new song to the list;

int remove\_Song(int n): remove a song from the list;

Playlist():default constructor;

~Playlist():default destructor.

Class diagrams

|  |  |
| --- | --- |
| Song | Playlist |
| string name;  string artist; | Song \*Songs;  size\_t empty;  size\_t list\_length; |
| string get\_name();  string get\_artist();  Song(string name,string artist);  Song();  ~Song()=default;  bool operator==(const Song &a); | int add\_Song(Song a);  int remove\_Song(int n);  Playlist()=default;  ~Playlist()=default; |

2. Constructors/Operator overloading

I wrote two constructors for class Song. Song(string name,string artist) create an actual object Song with name and artist. This is called when there is an input from terminal. Song() is called to initialize an array of type Song, it is more like an empty song with name= “ ”and artist= “ ”.

Due to the rule that each song on the list should be unique, the operator “==” is overloaded for the class Song, which is used to compare the equality of two songs.

From the project description, I do not feel the needs to make changes to destructors so I keep them as default.

3. Test Cases

There are 4 cases I tested in addition to the example tests.

Test 1: play a song outside of the length of list;

“can not play” is the expected output.

Test 2: remove a song from the end of list;

“can not erase” is the expected output

Test 3: add a song with the same name as the one on the list;

“can not insert” is the expected output

Test 4: remove a song from an “empty” space of the list;

“can not erase” is the expected output.

|  |  |
| --- | --- |
| Test | Output |
| m 5  i Mamma Mia;Abba  i We Will Rock You;Queen  i Daniel;Elton John  e 2  p 1  p 2  p 3  p 4  p 5  p 6  p 7  i Daniel;Elton John  i We Will Rock You;Queen  p 1  p 2  p 3  p 4  p 5  i 22;33  i 44;55  i 66;77  i 77;88  p 1  p 2  p 3  p 4  p 5  e 5  e 5  p 1  p 2  p 3  p 4  p 5 | success  success  success  success  success  played 1 Mamma Mia;Abba  played 2 Daniel;Elton John  can not play 3  can not play 4  can not play 5  can not play 6  can not play 7  can not insert Daniel;Elton John  success  played 1 Mamma Mia;Abba  played 2 Daniel;Elton John  played 3 We Will Rock You;Queen  can not play 4  can not play 5  success  success  can not insert 66;77  can not insert 77;88  played 1 Mamma Mia;Abba  played 2 Daniel;Elton John  played 3 We Will Rock You;Queen  played 4 22;33  played 5 44;55  success  can not erase 5  played 1 Mamma Mia;Abba  played 2 Daniel;Elton John  played 3 We Will Rock You;Queen  played 4 22;33  can not play 5 |