```
Script started on Thu Oct 11 22:46:10 2018
[?1034hbash-3.2$ ls
Application.cpp         PlayList.h              Song.h
        main.cpp        typescript.pdf
Application.h           PlayListTester.cpp      SongTester.cpp
        testSongOutput.txt
Debug                   PlayListTester.h SongTester.h
testSongs.txt
PlayList.cpp            Song.cpp         hale.txt        typescript
bash-3.2$ cat Application.cpp PlayList.h Song.h main.cpp Application.h
PlayListTester.cpp SongTester.cpp testSongOutput.txt PlayListTester.h
 SongTester.h testSongs.txt PlayList.cpp Song.cpp
/* Application.cpp defines the Application methods.
    *Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

#include "Application.h"
#include "PlayListTester.h"
#include "PlayList.h"
#include <iostream>
using namespace std;

Application::Application() {
        PlayList pList("testSongs.txt");
        unsigned choice;
            while ( true ) {

                cout << "\n"
                                 << "Welcome to the PlayList
Manager!"<< "\n"
                                 << "Please enter: " << "\n"
                                         << "1 - to search the
PlayList for songs by a given artist" << "\n"
                                         << "2 - to search the
PlayList for songs from a given year" << "\n"
                                         << "3 - to search the
PlayList for songs with a given phrase in their title" << "\n"
                                         << "4 - to add a new song to
the PlayList" << "\n"
                                         << "5 - to remove a song to
the PlayList" << "\n"
                                         << "0 - to quit" << endl;

                cin >> choice;

                if (choice == 0) {
                                 cout << "Ending..." << endl;
```

```
                                break;
                                }
                else if ( choice == 1) {
                        string artist;
                        cout << "Please enter name of artist: " <<
endl;
                        cin >> artist;
                        vector<Song> v1 =
pList.searchByArtist(artist);
                        for(unsigned i = 0; i < v1.size(); i++) {
                                cout << v1[i].getTitle() << '\n' <<
v1[i].getYear() << '\n' << endl;
                        }
                }


                else if(choice == 2) {
                        unsigned year;
                        cout << "Please enter the year: " << endl;
                        cin >> year;
                        vector<Song> v1 = pList.searchByYear(year);
                        for(unsigned i = 0; i < v1.size(); i++) {
                                cout << v1[i].getTitle() << '\n' <<
v1[i].getArtist()<< '\n' << endl;
                        }
                }
                else if(choice == 3){
                        string phrase;
                        cout << "Please enter a phrase from the title
of the Song: " << endl;
                        cin >> phrase;
                        vector<Song> v1 =
pList.searchByTitlePhrase(phrase);
                        for(unsigned i = 0; i < v1.size(); i++) {
                                cout << v1[i].getTitle() << '\n' <<
v1[i].getYear()<< '\n' << endl;
                        }
                }
                else if (choice == 4){
                        string title;
                        unsigned year;
                        string artist;
                        cin.ignore(256, '\n');
                        cout << "Please enter title: " << endl;
                        cin >> title;
                        cout << "Please enter year: " << endl;
                        cin.ignore(256, '\n');
                        cin >> year;
                        cout << "Please enter name of the artist: "
<< endl;
```

```
                                cin >> artist;
                                cin.ignore(256, '\n');
                                pList.addSong(Song(title, artist, year));
                                unsigned save;
                                cout << "Do you want to Save? Enter 9." <<
endl;
                                cin >> save;
                                if (save == 9) {
                                        pList.save();
                                        cout << "saved" << endl;
                                }
                        }
                        else if (choice == 5) {
                                string title;
                                unsigned year;
                                string artist;
                                cout << "Please enter title to remove: "<<
endl;
                                cin.ignore(256, '\n');
                                cin >> title;
                                cout << "Please enter year to remove: " <<
endl;
                                cin.ignore(256, '\n');
                                cin >> year;
                                cout << "Please enter artist to remove: "<<
endl;
                                cin.ignore(256, '\n');
                                pList.removeSong(Song(title, artist, year));
                                unsigned save;
                                cout << "Do you want to Save? Enter 9." <<
endl;
                                cin >> save;
                                if (save == 9) {
                                        pList.save();
                                        cout << "saved" << endl;
                                }
                        }



                }
        }

/* PlayList.h declares class PlayList.
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */
```

```
   #ifndef PLAYLIST_H_
   #define PLAYLIST_H_

   #include <string>
   #include "Song.h"
   #include <vector> // STL vector
   using namespace std;

   class PlayList {
   public:
      PlayList(const string& fileName);
      unsigned getNumSongs() const;
      vector<Song> searchByArtist(const string& artist) const;
      vector<Song> searchByYear(unsigned year);
      vector<Song> searchByTitlePhrase(const string& phrase);
      void addSong(const Song& newSong);
      void removeSong(const Song& aSong);
      void save() const;
   private:
      vector<Song> mySongs;
   };

   #endif /*PLAYLIST_H_*/
/* Song.h declares a class for storing song information.
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

   #ifndef SONG_H
   #define SONG_H

   #include <string>
   using namespace std;

   class Song {
   public:
           Song();
           Song(const string& title, const string& artist, unsigned
year);
           void readFrom(istream& in);
           void writeTo(ostream& out) const;
           string getTitle () const;
           string getArtist() const;
           unsigned getYear() const;
           bool operator==(const Song& song2) const;

   private:
      string   myTitle;
      string   myArtist;
      unsigned myYear;
   };

   #endif /*SONG_H_*/
/* main.cpp tests the classes in our project.
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date: Sept 11 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

   #include "SongTester.h"
   #include "PlayListTester.h"
   #include "SongTester.h"
   #include "PlayList.h"
   #include "Song.h"
   #include "Application.h"
   #include <iostream>
   using namespace std;

   int main() {
       SongTester sTester;
       sTester.runTests();
       PlayListTester plTester;
       plTester.runTests();
//       Application();



                 cout << "\nWelcome to the PlayList Manager!\n"<<
endl;
            while ( true ) {
                 cout << "\n"
                                  << "Please enter an option: " <<
"\n"
                                          << "1 - to search the
PlayList for songs by a given artist\n"
                                          << "2 - to search the
PlayList for songs from a given year\n"
                                          << "3 - to search the
PlayList for songs with a given phrase in their title\n"
                                          << "4 - to add a new song to
the PlayList" << "\n"
                                          << "5 - to remove a song to
the PlayList" << "\n"
                                          << "0 - to quit" << endl;
                 unsigned choice;
                 cin >> choice;
                 PlayList pList("testSongs.txt");
                 if (choice == 0) {
                                  cout << "\nEnding...";
                                  break;
                                  }
                 else if ( choice == 1) {
                          string artist;
                          cout << "Please enter name of artist: " <<
endl;
                          cin >> artist;
                          vector<Song> v1 =
pList.searchByArtist(artist);
                          for(unsigned i = 0; i < v1.size(); i++) {
                                  cout << v1[i].getTitle() << '\n' <<
v1[i].getYear() << '\n' << endl;
                          }
                 }


                 else if(choice == 2) {
                          unsigned year;
                          cout << "Please enter the year: " << endl;
                          cin >> year;
                          vector<Song> v1 = pList.searchByYear(year);
                          for(unsigned i = 0; i < v1.size(); i++) {
                                  cout << v1[i].getTitle() << '\n' <<
v1[i].getArtist()<< '\n' << endl;
                          }
                 }
                 else if(choice == 3){
                          string phrase;
                          cout << "Please enter a phrase from the title
of the Song: " << endl;
                          cin >> phrase;
                          vector<Song> v1 =
pList.searchByTitlePhrase(phrase);
                          for(unsigned i = 0; i < v1.size(); i++) {
                                  cout << v1[i].getTitle() << '\n' <<
v1[i].getYear()<< '\n' << endl;
                          }
                 }
                 else if (choice == 4){
                          string title;
                          unsigned year;
                          string artist;
                          cin.ignore(256, '\n');
                          cout << "Please enter title: " << endl;
                          cin >> title;
                          cout << "Please enter year: " << endl;
                          cin.ignore(256, '\n');
```

```
				cin >> year;
				cout << "Please enter name of the artist: "
<< endl;
				cin >> artist;
				cin.ignore(256, '\n');
				pList.addSong(Song(title, artist, year));
				unsigned save;
				cout << "Do you want to Save? Enter 9." <<
endl;
				cin >> save;
				if (save == 9) {
					pList.save();
					cout << "saved" << endl;
				}
			}
			else if (choice == 5) {
				string title;
				unsigned year;
				string artist;
				cout << "Please enter title to remove: "<<
endl;
				cin.ignore(256, '\n');
				cin >> title;
				cout << "Please enter year to remove: " <<
endl;
				cin.ignore(256, '\n');
				cin >> year;
				cout << "Please enter artist to remove: "<<
endl;
				cin.ignore(256, '\n');
				pList.removeSong(Song(title, artist, year));
				unsigned save;
				cout << "Do you want to Save? Enter 9." <<
endl;
				cin >> save;
				if (save == 9) {
					pList.save();
					cout << "saved" << endl;
				}
			}
		}
	}




/* Application.h declares class A.
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

#ifndef APPLICATION_H_
#define APPLICATION_H_

class Application {
public:
         Application();
};

#endif /* APPLICATION_H_ */
 /* PlayListTester.cpp defines the PlayList test-methods.
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

   #include "PlayListTester.h"
   #include "PlayList.h"
   #include <iostream>
   #include <cassert>
   using namespace std;

   void PlayListTester::runTests() {
      cout << "\nTesting class PlayList..." << endl;
      testConstructors();
      testSearchByArtist();
      testSearchByTitlePhrase();
      testaddSongRemoveSong();
      testSave();

      cout << "All tests passed!" << endl;
   }


   void PlayListTester::testConstructors() {
         cout << "- constructors..." << flush;
         PlayList pList("testSongs.txt");
         assert( pList.getNumSongs() == 4 );
         cout << " 0 " << flush;

         cout << " Passed!" << endl;
      }


   void PlayListTester::testSearchByArtist() {
         cout << "- searchByArtist()... " << flush;
         // load a playlist with test songs
```

```
        PlayList pList("testSongs.txt");

        // empty case (0)
        vector<Song> searchResult = pList.searchByArtist("Cream");
        assert( searchResult.size() == 0 );
        cout << " 0 " << flush;

        // case of 1
        searchResult = pList.searchByArtist("Baez");
        assert( searchResult.size() == 1 );
        assert( searchResult[0].getTitle() == "Let It Be" );
        cout << " 1 " << flush;

        // case of 2
        searchResult = pList.searchByArtist("Beatles");
        assert( searchResult.size() == 2 );
        assert( searchResult[0].getTitle() == "Let It Be" );
        assert( searchResult[1].getTitle() == "Penny Lane" );
        cout << " 2 " << flush;

        cout << " Passed!" << endl;
    }

//Tester for searching by year
void PlayListTester::testSearchByYear() {
         cout << "- searchByYear()... " << flush;
         // load a playlist with test songs
         PlayList pList("testSongs.txt");

         // empty case (0)
         vector<Song> searchResult = pList.searchByYear(2015);
         assert( searchResult.size() == 0 );
         cout << " 0 " << flush;

         //case 1
         searchResult = pList.searchByYear(1967);
         assert( searchResult.size() == 1 );
         assert( searchResult[0].getTitle() == "Let It Be" );
         cout << " 1 " << flush;

         // case of 2
         searchResult = pList.searchByYear(2012);
         assert( searchResult.size() == 2 );
         assert( searchResult[0].getTitle() == "Let It Be" );
         assert( searchResult[1].getTitle() == "Call Me Maybe" );
         cout << " 2 " << flush;


         cout << " Passed!" << endl;
    }


    //Tester for searching by Title Phrase
    void PlayListTester::testSearchByTitlePhrase() {
            cout << "- searchByTitlePhrase()... " << flush;
            // load a playlist with test songs
            PlayList pList("testSongs.txt");

            // empty case (0)
            vector<Song> searchResult =
pList.searchByTitlePhrase("Cream");
            assert(searchResult.size() == 0 );
            cout << " 0 " << flush;

            // case of 1
            searchResult = pList.searchByTitlePhrase("Let It");
            assert(searchResult.size() == 2 );
            assert(searchResult[0].getTitle() == "Let It Be");
            cout << " 1 " << flush;

            // case of 2
            searchResult = pList.searchByTitlePhrase("Call Me");
            assert(searchResult.size() == 1 );
            assert(searchResult[0].getTitle() == "Call Me Maybe");
            cout << " 2 " << flush;

            cout << " Passed!" << endl;


    }

    //Tester for adding Song
    void PlayListTester::testaddSongRemoveSong() {
            cout << "- addSong()... " << flush;
            // load a playlist with test songs
            PlayList pList("testSongs.txt");

            //adding a song
            Song newSong("Sanufa", "Bas", 2018);
            pList.addSong(newSong);
            vector<Song> searchResult = pList.searchByArtist("Bas");
            assert( searchResult[0].getTitle() == "Sanufa" );
            cout << " 0 " << flush;

            //removing song
            Song Songtoremove("Sanufa", "Bas", 2018);
            pList.removeSong(Songtoremove);
            searchResult = pList.searchByArtist("Bas");
            assert(searchResult.size() == 0);
```

```
                cout << " 1 " << flush;

                cout << " Passed!" << endl;

        }


        //Tester for Saving Songs
        void PlayListTester::testSave() const {
                cout << "- Save()... " << flush;

                // load a playlist that is to be changed
                PlayList pList("testSongs.txt");

                //add a song to cause a change in the original playList
                Song s1("Sanufa", "Bas", 2018);
                pList.addSong(s1);
                cout << " 0 " << flush;

                // write the modified playList back into the original file
                pList.save();
                cout << " 1 " << flush;

                // load the saved file under new file name and test for the
added song
                //PlayList pList2("testSongs.txt");
                vector<Song> searchResult = pList.searchByArtist("Bas");
                assert(searchResult.size() == 1);
                cout << " 2 " << flush;

                //return the playList to its original
                pList.removeSong(s1);
                pList.save();
                cout << " 3 " << flush;

                // check that the song has been removed
                searchResult = pList.searchByArtist("Bas");
                assert(searchResult.size() == 0);
                cout << " 4 " << flush;

                cout << " Passed!" << endl;

        }

/* SongTester.cpp defines the test-methods for class SongTester.
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */
```

```
#include "SongTester.h"
#include "Song.h"
#include <iostream>
#include <cassert>
#include <fstream>
using namespace std;


void SongTester::runTests() {
   cout << "Testing class Song..." << endl;
   testConstructors();
   testReadFrom();
   testWriteTo();
   testOperator();

   cout << "All tests passed!" << endl;
}

void SongTester::testConstructors() {
       cout << "- constructors ... " << flush;
       // default constructor
       Song s;
       assert( s.getTitle() == "" );
       assert( s.getArtist() == "" );
       assert( s.getYear() == 0 );
       cout << " 0 " << flush;
       // explicit constructor
       Song s1("Badge", "Cream", 1969);
                assert( s1.getTitle() == "Badge" );
       assert( s1.getArtist() == "Cream" );
       assert( s1.getYear() == 1969 );
       cout << " 1 " << flush;

       cout << " Passed!" << endl;
   }

void SongTester::testReadFrom() {
      cout << "- readFrom()... " << flush;
      ifstream fin("testSongs.txt");
      assert( fin.is_open() );
      Song s;

      // read first song in test playlist
      s.readFrom(fin);
      assert( s.getTitle() == "Call Me Maybe" );
      assert( s.getArtist() == "Carly Rae Jepsen" );
      assert( s.getYear() == 2012 );
      cout << " 0 " << flush;
```

```
        // read second song in test playlist
        string separator;
        getline(fin, separator);
        s.readFrom(fin);
        assert( s.getTitle() == "Let It Be" );
        assert( s.getArtist() == "The Beatles" );
        assert( s.getYear() == 1967 );
        cout << " 1 " << flush;

        // read third song in test playlist
        getline(fin, separator);
        s.readFrom(fin);
        assert( s.getTitle() == "Let It Be" );
        assert( s.getArtist() == "Joan Baez" );
        assert( s.getYear() == 1971 );
        cout << " 2 " << flush;

        //reads fourth song in test playlist
        getline(fin, separator);
        s.readFrom(fin);
        assert( s.getTitle() == "Penny Lane" );
                 assert( s.getArtist() == "The Beatles" );
                 assert( s.getYear() == 1967 );
             cout << " 3 " << flush;

        fin.close();
        cout << "Passed!" << endl;
     }

  void SongTester::testWriteTo() {
        cout << "- writeTo()... " << flush;

        // declare three songs
        Song s1("Badge", "Cream", 1969);
        Song s2("Godzilla", "Blue Oyster Cult", 1977);
        Song s3("Behind Blue Eyes", "The Who", 1971);

        // write the three songs to an output file
        ofstream fout("testSongOutput.txt");
        assert( fout.is_open() );
        s1.writeTo(fout);
        s2.writeTo(fout);
        s3.writeTo(fout);
        fout.close();

        // use readFrom() to see if writeTo() worked
        ifstream fin("testSongOutput.txt");
        assert( fin.is_open() );
        Song s4, s5, s6;
        // read and check the first song
        s4.readFrom(fin);
        assert( s4.getTitle() == "Badge" );
        assert( s4.getArtist() == "Cream" );
        assert( s4.getYear() == 1969 );
        cout << " 0 " << flush;

        // read and check the second song
        s5.readFrom(fin);
        assert( s5.getTitle() == "Godzilla" );
        assert( s5.getArtist() == "Blue Oyster Cult" );
        assert( s5.getYear() == 1977 );
        cout << " 1 " << flush;

        // read and check the third song
        s6.readFrom(fin);
        assert( s6.getTitle() == "Behind Blue Eyes" );
        assert( s6.getArtist() == "The Who" );
        assert( s6.getYear() == 1971 );
        cout << " 2 " << flush;

        fin.close();
        cout << " Passed!" << endl;
    }

   void SongTester::testOperator() {
                cout << "- operator()..." << flush;
                // creating 3 song objects
                Song s1("Badge", "Cream", 1969);
                Song s2("Godzilla", "Blue Oyster Cult", 1977);
                Song s3("Behind Blue Eyes", "The Who", 1971);

                //compares s1 with s1 and asserts they are the same
                assert(s1.operator==(s1) == true);
                cout << " 0 " << flush;
                // asserts s1 and s2 are different
                assert(s1.operator==(s2) == false);
                cout << " 1 " << flush;

                cout << " Passed!" << endl;



   }
Badge
Cream
1969
Godzilla
Blue Oyster Cult
1977
```

```
Behind Blue Eyes
The Who
1971
/* PlayListTester.h tests the PlayList class.
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

   #ifndef PLAYLISTTESTER_
   #define PLAYLISTTESTER_

   class PlayListTester {
   public:
       void runTests();
       void testConstructors();
       void testSearchByArtist();
       void testSearchByYear();
       void testSearchByTitlePhrase();
       void testaddSongRemoveSong();
       void testSave() const;
   };

   #endif /*PLAYLISTTESTER_*/
/* SongTester.h declares a test-class for class Song.
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

   #ifndef SONGTESTER_H_
   #define SONGTESTER_H_

   class SongTester {
   public:
      void runTests();
      void testConstructors();
      void testReadFrom();
      void testWriteTo();
      void testOperator();
   };

   #endif /*SONGTESTER_H_*/
Call Me Maybe
Carly Rae Jepsen
2012

Let It Be
```

```
The Beatles
1967

Let It Be
Joan Baez
1971

Penny Lane
The Beatles
1967/* PlayList.cpp defines the PlayList methods.
    *Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

   #include "PlayList.h"
   #include <fstream>      // ifstream
   #include <cassert>      // assert()
   #include <vector>
   using namespace std;

/* PlayList constructor
        * @param: fileName, a string
    * Precondition: fileName contains the name of a playlist file.
    */
   PlayList::PlayList(const string& fileName) {
      // open a stream to the playlist file
      ifstream fin( fileName.c_str() );
      assert( fin.is_open() );

      // read each song and append it to mySongs
      Song s;
      string separator;
      while (true) {
          s.readFrom(fin);
          if ( !fin ) { break; }
          getline(fin, separator);
          mySongs.push_back(s);
      }



      // close the stream
      fin.close();
   }

/* Retrieve length of the playlist
    * Return: the number of songs in the playlist.
    */
   unsigned PlayList::getNumSongs() const {
            return mySongs.size();
   }


/* Search by artist
   * @param: artist, a string.
   * Return: a vector containing all the Songs in mySongs by artist.
   */
   vector<Song> PlayList::searchByArtist(const string& artist) const {
            vector<Song> v;
            for (unsigned i = 0; i < mySongs.size(); i++) {
                     if ( mySongs[i].getArtist().find(artist) !=
string::npos ) {
                              v.push_back( mySongs[i] );
                     }
            }
            return v;
   }



/* Search by year
  * @param: year, a positve integer.
  * Return: a vector containing all the Songs in mySongs by year.
  */

   vector<Song> PlayList::searchByYear(unsigned year) {
            vector<Song> v;
            for (unsigned i = 0; i < mySongs.size(); i++) {
                     if ( mySongs[i].getYear() == year) {
                              v.push_back( mySongs[i] );
                     }
            }
            return v;
   }



/* searchByTitlePhrase() searches the PlayList for Songs by a given
title phrase.
         * @param: phrase, a string within a title of the song
         * @return: a vector containing all Songs in the playList
         *            for whom title phrase is a substring of myTitle.
         */
   vector<Song> PlayList::searchByTitlePhrase(const string& phrase) {
            vector<Song> v;
            for (unsigned i = 0; i < mySongs.size(); i++) {
                     if ( mySongs[i].getTitle().find(phrase) !=
string::npos ) {
```

```
                                v.push_back( mySongs[i] );
                        }
                }
                return v;
        }



/* addSong() adds a new song in the playList for Songs by asking for
user input.
        *
        * @param: newSong, a string containing a separate Song object
created through user input.
        * @return: this does not return anything but appends the
newSong class in the memory.
        */
   void PlayList::addSong(const Song& newSong) {
            mySongs.push_back(newSong);

   }


/* removeSong() removes a song from the PlayList of songs through user
input information.
        * @param: aSong, a string containing an already Song object
found through user input information.
        * @return: this does not return a vector but removes a Song
object from the PlayList of songs.
        */

   void PlayList::removeSong(const Song& aSong) {
            vector<Song>::iterator i = mySongs.begin();
            while (i != mySongs.end()) {
                    if ( i -> getTitle() == aSong.getTitle() ) {
                    i = mySongs.erase(i);
            }
            else { ++i;}
            }
   }



/* save() saves and writes the Song object created through user input
(newSong) into the PlayList for Songs.
        *
        * @param: this method has no parameters
        * @return: this method does not return but writes the appended
Songs objected into the testSongs.txt file.
        */
   void PlayList::save() const {
           ofstream fileout("testSongs.txt");
           for ( unsigned i = 0; i < mySongs.size(); ++i) {
                   mySongs[i].writeTo(fileout);
                   fileout << "\n";
           }
           fileout.close();

   }
/* Song.cpp defines the methods for class Song (see Song.h).
    * Student Name:Nana Osei Asiedu Yirenkyi
    * Date:Sept 16 2018
    * project01
    * Begun by: Joel Adams, for CS 112 at Calvin College.
    */

    #include "Song.h"
    #include <cstdlib>
    #include <istream>

     /* Song default constructor
    * Postcondition: myTitle and myArtist are empty strings
    *            && myYear == 0.
    */
    Song::Song() {
        myTitle = myArtist = "";
        myYear = 0;
    }

    /* Explicit constructor
        * @param: title, a string
        * @param: artist, a string
        * @year: an unsigned int.
        * Postcondition: myTitle == title &&
        *                myArtist == artist &&
        *                myYear == year.
        */
    Song::Song(const string& title, const string& artist, unsigned
year) {
        myTitle = title;
        myArtist = artist;
        myYear = year;
   }


    /* Song input method...
                 * @param: in, an istream
            * Precondition: in contains the title, artist, and year
data for a Song.
            * Postcondition: the title, artist, and year data have
```

```
been read from in &&
            *                myTitle == title &&
            *                myArtist == artist &&
            *                myYear == year.
            */
    void Song::readFrom(istream& in) {
        getline(in, myTitle);
                getline(in, myArtist);
                string yearString;
                getline(in, yearString);
                myYear = atoi( yearString.c_str() );
    }

        /* Song output...
                * @param: out, an ostream
                * Postcondition: out contains myTitle, a newline,
                *                             myArtist, a newline,
                *                             myYear, and a newline.
                */
        void Song::writeTo(ostream& out) const {
                out << myTitle << '\n'
                                 << myArtist << '\n'
                                 << myYear  << '\n';
        }



    /* getter method for myTitle
        * Return: myTitle
        */
    string Song::getTitle() const {
        return myTitle;
    }



    /* getter method for myArtist
        * Return: myArtist
        */
    string Song::getArtist() const {
        return myArtist;
    }


    /* getter method for myYear
        * Return: myYear
        */
    unsigned Song::getYear() const {
        return myYear;
    }
    /* Operator
        * returns true if the Song to which this
        *  message is sent is the same as song2;
        *  and returns false otherwise.
        */
    bool Song::operator==(const Song& song2) const {
        if (myTitle != song2.getTitle()) {
                return false;
        }
        else if (myArtist != song2.getArtist()) {
                return false;
        }
        else if (myYear != song2.getYear()) {
                return false;
        }
        return true;
    }
bash-3.2$ cd Debug
bash-3.2$ ls
Application.d            PlayListTester.d SongTester.d
makefile        subdir.mk
Application.o            PlayListTester.o SongTester.o
objects.mk
PlayList.d               Song.d                   main.d
        project01
PlayList.o               Song.o                   main.o
        sources.mk
bash-3.2$ make all
make: Nothing to be done for `all'.
bash-3.2$ cd ..
bash-3.2$ ./Debug/project01
Testing class Song...
- constructors ...  0  1  Passed!
- readFrom()...  0  1  2  3 Passed!
- writeTo()...  0  1  2  Passed!
- operator()... 0  1  Passed!
All tests passed!

Testing class PlayList...
- constructors... 0  Passed!
- searchByArtist()...  0  1  2  Passed!
- searchByTitlePhrase()...  0  1  2  Passed!
- addSong()...  0  1  Passed!
- Save()...  0  1  2  3  4  Passed!
All tests passed!

Welcome to the PlayList Manager!
```

```
Please enter an option:
1 - to search the PlayList for songs by a given artist
2 - to search the PlayList for songs from a given year
3 - to search the PlayList for songs with a given phrase in their
title
4 - to add a new song to the PlayList
5 - to remove a song to the PlayList
0 - to quit
1
Please enter name of artist:
Carly
Call Me Maybe
2012


Please enter an option:
1 - to search the PlayList for songs by a given artist
2 - to search the PlayList for songs from a given year
3 - to search the PlayList for songs with a given phrase in their
title
4 - to add a new song to the PlayList
5 - to remove a song to the PlayList
0 - to quit
2
Please enter the year:
2012
Call Me Maybe
Carly Rae Jepsen


Please enter an option:
1 - to search the PlayList for songs by a given artist
2 - to search the PlayList for songs from a given year
3 - to search the PlayList for songs with a given phrase in their
title
4 - to add a new song to the PlayList
5 - to remove a song to the PlayList
0 - to quit
3
Please enter a phrase from the title of the Song:
Call
Call Me Maybe
2012


Please enter an option:
1 - to search the PlayList for songs by a given artist
2 - to search the PlayList for songs from a given year
3 - to search the PlayList for songs with a given phrase in their
```

```
title
4 – to add a new song to the PlayList
5 – to remove a song to the PlayList
0 – to quit
4 1
Please enter name of artist:
Joan
Let It Be
1971


Please enter an option:
1 – to search the PlayList for songs by a given artist
2 – to search the PlayList for songs from a given year
3 – to search the PlayList for songs with a given phrase in their
title
4 – to add a new song to the PlayList
5 – to remove a song to the PlayList
0 – to quit
2
Please enter the year:
1971
Let It Be
Joan Baez


Please enter an option:
1 – to search the PlayList for songs by a given artist
2 – to search the PlayList for songs from a given year
3 – to search the PlayList for songs with a given phrase in their
title
4 – to add a new song to the PlayList
5 – to remove a song to the PlayList
0 – to quit
3 2
Please enter the year:
1967
Let It Be
The Beatles

Penny Lane
The Beatles


Please enter an option:
1 – to search the PlayList for songs by a given artist
2 – to search the PlayList for songs from a given year
3 – to search the PlayList for songs with a given phrase in their
title
4 – to add a new song to the PlayList
```

```
5 - to remove a song to the PlayList
0 - to quit
4
Please enter title:
Tribe
Please enter year:
2018
Please enter name of the artist:
Bas
Do you want to Save? Enter 9.
9
saved

Please enter an option:
1 - to search the PlayList for songs by a given artist
2 - to search the PlayList for songs from a given year
3 - to search the PlayList for songs with a given phrase in their
title
4 - to add a new song to the PlayList
5 - to remove a song to the PlayList
0 - to quit
5
Please enter title to remove:
Tribe
Please enter year to remove:
2018
Please enter artist to remove:
Do you want to Save? Enter 9.
9
saved

Please enter an option:
1 - to search the PlayList for songs by a given artist
2 - to search the PlayList for songs from a given year
3 - to search the PlayList for songs with a given phrase in their
title
4 - to add a new song to the PlayList
5 - to remove a song to the PlayList
0 - to quit
0

Ending...bash-3.2$ ∂[Kexit

Script done on Thu Oct 11 22:52:28 2018
```