CMP-5015Y Summative Coursework 2 - Music Database in C++

100166648 (ssq16shu)

Wed, 1 May 2019 09:25

Contents

Duration.h

```
/*****************************************************************************
 *
 *  Programming2 Coursework 2
 *
 *  Title:                Music Database in C++
 *
 *  File:                 Duration.h
 *
 *  Description:          Models the time duration information of a
 *                        track/album/collection of music
 *                        Paired with Duration.cpp
 *
 *  Author:               100166648 / ssq16shu
 *
 *  Version History:      v0.1     190307  initial skeleton
 *                        v0.2     190321  accessors, relational operators and
 *                                         constructor implemented.
 *                        v1.0     190328  IOstream and arithmetic operators added
 *                        v1.2     190412  addition operator overload returns obj.
 *                        v1.2.1   190415  + overload tweaked to remove memory leak
 *                        v2.0     190421  istream overload
 *                        v2.1     190424  normalised int constructor 'overflow'.
 *                                         changed '-' overload to return object
 *                        v2.2     190425  moved istream to cpp file
 *                                         removed failbit error messages
 *                        v3.0     190430  removed obsolete methods
 *
 *  Notes:           -    As commented in code: multiplication or division of
 *                        Durations are never needed so they are not implemented
 *                        in the arithmetic operator list for this class
 *
 ****************************************************************************/

#ifndef ALBUMDB_CL_DURATION_H
#define ALBUMDB_CL_DURATION_H

//include files
#include <stdio.h>
#include <iostream>
#include <iomanip>


class Duration {
    //instance variables:
    int hours;
    int minutes;
    int seconds;

public:
//-----Declarations
    //constructors
    Duration();
    Duration(int hours, int minutes, int seconds);

    //accessors
    int getHours() const;
    int getMinutes() const;
    int getSeconds() const;

    //cpp file operator overload(s)
    operator int() const;
    friend std::istream& operator>>(std::istream& is, Duration &d);

    //testing harness
    static void testHarnessDuration();
};


//-----Definitions
//--constructors:
//empty (may remove)
inline Duration::Duration(){
    this->hours=0;
    this->minutes=0;
    this->seconds=0;
}
//h:m:s supplied
inline Duration::Duration(int hours, int minutes, int seconds) {
    //adjust for potential overfull values (ie. max seconds: 59)
    if(seconds>59){
        //overflow into minutes
        minutes += seconds/60;
        //remove adjusted minutes
        seconds = seconds%60;
    }
    if(minutes>59){
        //overflow into hours
        hours += minutes/60;
        //remove adjusted hours
        minutes = minutes%60;
    }
    this->hours = hours;
    this->minutes = minutes;
    this->seconds = seconds;
}

//--accessors
inline int Duration::getHours() const {
    return hours;
}
inline int Duration::getMinutes() const {
    return minutes;
}
inline int Duration::getSeconds() const {
    return seconds;
}

//--(partial) arithmetic operator overloading (multiplication and division are
   never needed)
//returns summation of two duration objects as a duration object
inline Duration operator+(const Duration& d1, const Duration& d2) {
    int h = d1.getHours()+d2.getHours();
    int m =d1.getMinutes()+d2.getMinutes();
    int s=d1.getSeconds()+d2.getSeconds();
    Duration result = Duration(h,m,s);
    return result;
}

//returns a duration object of one duration subtracted from another (d1-d2)
inline int operator-(const Duration& d1, const Duration d2) {
    int h = d1.getHours() - d2.getHours();
    int m = d1.getMinutes() - d2.getMinutes();
    int s = d1.getSeconds() - d2.getSeconds();
    Duration result = Duration(h,m,s);
    return result;
}

//--relational operator overloading
//uses static cast to int (total seconds) to compare two duration objects
inline bool operator == (const Duration& d1, const Duration d2) {
    return static_cast<int> (d1) == static_cast<int> (d2);
}
inline bool operator != (const Duration& d1, const Duration d2) {
    return !(d1 == d2);
}
inline bool operator >  (const Duration& d1, const Duration d2) {
    return static_cast<int> (d1) > static_cast<int> (d2);
}
inline bool operator <  (const Duration& d1, const Duration d2) {
    return static_cast<int> (d1)<static_cast<int> (d2);
}
inline bool operator >= (const Duration& d1, const Duration d2) {
    return !(d1 < d2);
}
inline bool operator <= (const Duration& d1, const Duration d2) {
    return !(d1 > d2);
}

//-----IO stream operator overloading
//--output stream
//formats console output of object (as 'h:m:s')
inline std::ostream& operator<<(std::ostream& os, const Duration &d) {
    return os << std::setfill('0')
              << std::setw(2) << d.getHours()   << ":"
              << std::setw(2) << d.getMinutes() << ":"
              << std::setw(2) << d.getSeconds();
}

#endif //ALBUMDB_CL_DURATION_H
```

Duration.cpp

```
/*******************************************************************************
 *
 *  Programming2 Coursework 2
 *
 *  Title:              Music Database in C++
 *
 *  File:               Duration.cpp
 *
 *  Description:        Models the time duration information of a
 *                      track/album/collection of music
 *                      Paired with Duration.h
 *
 *  Author:             100166648 / ssq16shu
 *
 *  Version History:    v0.1    190307  initial skeleton
 *                      v1.0    190328  added cast override
 *                      v2.0    190422  testing harness
 *                      v3.0    190430  removed obsolete methods
 *
 *  Notes:
 *
 ******************************************************************************/

//-----h file inclusions
#include "Duration.h"


using namespace std;

//-----Overload(s)
//--istream
//checks input format and creates duration object if correct
 std::istream& operator>>(std::istream& is, Duration &d){
    //input storage
    int h =0, m = 0, s=0;
    char c1, c2;

    //check stream input successfully to ins and char
    if(is >> h >> c1 >> m >> c2 >> s){
        //confirm delimiting chars
        if (c1 == ':' && c2 == ':'){
            //create object using input values
            d = Duration(h,m,s);
        }else {
            //set failbit for stream to indicate error on input
            is.clear(std::ios_base::failbit);
            //message to user
            std::cout << "please enter Duration using correct placement of ':'"
               <<std::endl;
            //error message to console
            std::cerr << "Duration input stream failed. Did not match correct
               format" << std::endl;
        }
    }
    //return istream
    return is;
}

//--int assignment of duration object
//returns total duration in seconds when static_cast to int
Duration::operator int() const{
    return seconds + (minutes*60) + (hours*60*60);
}

//-----testing harness
void Duration::testHarnessDuration(){
    cout<<"----------------------------------------"<<endl;
    cout<<"     Duration class test harness:"<<endl;
    cout<<"----------------------------------------"<<endl;
    cout << "testing Duration parameter constructor using get methods:" << endl;
    Duration* testDur = new Duration(1,2,3);
    cout << "getHours \t(expected result: 1) : " << testDur->getHours() << endl;
    cout << "getMinutes \t(expected result: 2) : " << testDur->getMinutes() <<
       endl;
    cout << "getSeconds \t(expected result: 3) : " << testDur->getSeconds() <<
       endl;
    cout<<endl;

    cout<< "testing ostream overload:"<<endl;
    cout<<*testDur<<endl;
    cout<<endl;

    cout<<"testing static cast to int of duration :"<<*testDur<<endl;
    int cast = *testDur;
    cout<<"expected 3723 : " << cast << endl;
    cout<<endl;

    cout<< "testing addition of durations:"<<endl;
    Duration* testDur2 = new Duration(1,2,3);
    cout<< "duration 1 : "<<*testDur2<<endl;
    Duration* testDur3 = new Duration(4,5,6);
    cout<< "duration 2 : "<<*testDur3<<endl;
    Duration result =*testDur2 + *testDur3;
    cout<< "combined duration (expected: '05:07:09') : "<<result<<endl;
    cout<<endl;

    cout<< "testing subtraction of durations:"<<endl;
    Duration* testDur4 = new Duration(1,2,3);
    int d4 =*testDur4;
    cout<< "duration 1 : "<<*testDur4<<" in sec: "<<d4<<endl;
    Duration* testDur5 = new Duration(4,5,6);
    int d5 =*testDur5;
    cout<< "duration 2 : "<<*testDur5<<" in sec: "<<d5<<endl;
    int result2 = *testDur5 - *testDur4;
    cout<< "subtracted duration (2-1) in seconds(expected: 10983) : "<<result2<<
       endl;
    cout<<endl;

    cout<<"testing relational operator '==' :"<<endl;
    cout<<"expected result 'false' : "<< (testDur2==testDur3)<<endl;
    cout<<"expected result 'true' : "<< (testDur2==testDur2)<<endl;

    cout<<"testing relational operator '!=' :"<<endl;
    cout<<"expected result 'false' : "<< (testDur2!=testDur2)<<endl;
    cout<<"expected result 'true' : "<< (testDur2!=testDur3)<<endl;

    cout<<"testing relational operator '>' :"<<endl;
    cout<<"expected result 'false' : "<< (testDur2>testDur3)<<endl;
    cout<<"expected result 'true' : "<< (testDur3>testDur2)<<endl;

    cout<<"testing relational operator '>=' :"<<endl;
    cout<<"expected result 'false' : "<< (testDur2>=testDur3)<<endl;
    cout<<"expected result 'true' : "<< (testDur3>=testDur2)<<endl;
    cout<<"expected result 'true' : "<< (testDur2>=testDur2)<<endl;

    cout<<"testing relational operator '<' :"<<endl;
    cout<<"expected result 'false' : "<< (testDur3<testDur2)<<endl;
    cout<<"expected result 'true' : "<< (testDur2<testDur3)<<endl;

    cout<<"testing relational operator '<=' :"<<endl;
    cout<<"expected result 'false' : "<< (testDur3<=testDur2)<<endl;
    cout<<"expected result 'true' : "<< (testDur2<=testDur3)<<endl;
    cout<<"expected result 'true' : "<< (testDur2<=testDur2)<<endl;
    cout<<endl;

    cout<<"testing istream overload:"<<endl;
    Duration testD6;
    cout<<"please input test duration (format hh:mm:ss):"<<endl<<flush;

//    console-in code: commented-out after manual format testing complete
//    while(!(cin >> testD6)){
//        cin.clear();
//        cin.ignore(256,'\n');
//    }

    cin>>testD6;
    cout<<"entered duration:"<<endl;
    cout<<testD6;
    cout<<endl;

    //free all allocated test object(s) memory
    delete testDur;
    delete testDur2;
    delete testDur3;
    delete testDur4;
    delete testDur5;
}
```

Track.h

```
/******************************************************************************
 *
 *  Programming2 Coursework 2
 *
 *  Title:              Music Database in C++
 *
 *  File:               Track.cpp
 *
 *  Description:        Models a music track incl. titles and durations
 *                      Paired with Track.cpp
 *
 *  Author:             100166648 / ssq16shu
 *
 *  Version History:    v1.0    190412  instance, constructor and accessors
 *                      v1.1    190415  added ostream overload
 *                      v2.0    190422  istream overload
 *                      v2.1    190425  moved istream to cpp file
 *                                      removed failbit error messages
 *                      v2.2    190426  relational operator overloads
 *
 *  Notes:
 *
 ******************************************************************************/

#ifndef ALBUMDB_CL_TRACK_H
#define ALBUMDB_CL_TRACK_H

//include files
#include "Duration.h"

//additional types
using std::string;

class Track {
    //instance variables
    string title;
    Duration duration;

public:
    //-----Declarations
    //constructors
    Track();
    Track(string title, Duration duration);

    //accessors
    string getTitle() const;
    Duration getDuration() const;

    //--cpp file overload(s)
    //istream
    friend std::istream& operator>>(std::istream& is, Track &t);

    //testing harness
    static void testHarnessTrack();
};


//-----Definitions
//--constructors
//empty
inline Track::Track(){
    this->title = "noTitle";
    this->duration = Duration();
}
//title & duration supplied
inline Track::Track(string title, Duration duration) {
    this->title = title;
    this->duration = duration;
}

//--accessors
inline string Track::getTitle() const {
    return title;
}
inline Duration Track::getDuration() const {
    return duration;
}

//--relational operator overloading
//uses existing Duration relational operators to compare track objects
inline bool operator == (const Track& t1, const Track t2) {
    return t1.getDuration() == t2.getDuration();
}
inline bool operator != (const Track& t1, const Track t2) {
    return !(t1 == t2);
}
inline bool operator > (const Track& t1, const Track t2) {
    return t1.getDuration() > t2.getDuration();
}
inline bool operator < (const Track& t1, const Track t2) {
    return t1.getDuration() < t2.getDuration();
}
inline bool operator >= (const Track& t1, const Track t2) {
    return !(t1 < t2);
}
inline bool operator <= (const Track& t1, const Track t2) {
    return !(t1 > t2);
}

//-----IO stream operator overloading
//--output stream
//formats console output of object (as 'title - duration')
inline std::ostream& operator<<(std::ostream& os, const Track &t){
    return os << t.getTitle() << " - " << t.getDuration();
}


#endif //ALBUMDB_CL_TRACK_H
```

Track.cpp

```cpp
/*****************************************************************************
 *
 *  Programming2 Coursework 2
 *
 *  Title:              Music Database in C++
 *
 *  File:               Track.cpp
 *
 *  Description:        Models a music track incl. titles and durations
 *                      Paired with Track.h
 *
 *  Author:             100166648 / ssq16shu
 *
 *  Version History:    v0.1    190412  initial skeleton
 *                      v1.0    190423  test harness
 *                      v2.0    190424  moved istream overload to cpp file
 *
 *  Notes:          -
 *
 *****************************************************************************/

//-----inclusions
#include "Track.h"

using namespace std;

//-----Overload(s)
//--input stream
//checks input and creates Track object if valid

std::istream& operator>>(std::istream& is, Track &t) {
    //input storage
    std::string title;
    Duration time;
    char c;

    //check input success
    //(read to delimiter, remove initial whitespace, get rest of line as title)
    if (is >> time >> c >> std::ws &&
            std::getline(is, title, '\r')) {
        //confirm delimiting char
        if (c == '-') {
            //create object using input values
            t = Track(title, time);
        } else {
            //set failbit for stream to indicate error on input
            is.clear(std::ios_base::failbit);
            //message to user
            std::cout << "please enter Track details. Ensure use of ' - ' "
                    "between duration and title (ie. duration - title)"
                    << std::endl;
            //error message to console
            std::cerr << std::endl << "Track input stream failed. "
                    "Did not match correct format" << std::endl;
        }
    }
    //return istream
    return is;
}
```

```
//-----testing harness
void Track::testHarnessTrack() {
    cout << "----------------------------------------" << endl;
    cout << "     Track class test harness:" << endl;
    cout << "----------------------------------------" << endl;
    cout << "testing Track constuctors:" << endl;
    Track* testT = new Track();
    cout << "default : " << *testT << endl;
    *testT = Track("testingTitle", Duration(01, 01, 01));
    cout << "parameter constructor: " << *testT << endl;
    cout << "using accessor methods: " << endl;
    cout << testT->getTitle() << endl;
    cout << testT->getDuration() << endl;
    cout << endl;

    cout << "testing istream input" << endl;
    Track testT3;
    cout << "please enter test line to input to track" << endl;
    //    console-in code: commented-out after manual format testing complete
    //    while(!(cin >> testT3)){
    //        cin.clear();
    //        cin.ignore(256,'\n');
    //    }

    cout << "entered Track details:" << endl;
    cout << testT3 << endl;
    cout << endl;

    cout << "testing relational operators" << endl;
    Track t1("track1_title", Duration(1, 1, 1));
    Track t2("track2_title", Duration(2, 2, 2));
    cout << "track 1: " << t1 << endl;
    cout << "track 2: " << t2 << endl;
    cout << "t1 == t2: " << (t1 == t2) << endl;
    cout << "t1 != t2: " << (t1 != t2) << endl;
    cout << "t1 > t2: " << (t1 > t2) << endl;
    cout << "t2 > t1: " << (t2 > t1) << endl;
    cout << "t1 < t2: " << (t1 < t2) << endl;
    cout << "t2 < t1: " << (t2 < t1) << endl;
    cout << "t1 >= t2: " << (t1 >= t2) << endl;
    cout << "t2 >= t2: " << (t2 >= t2) << endl;
    cout << "t1 <= t2: " << (t1 <= t2) << endl;
    cout << "t1 <= t1: " << (t1 <= t1) << endl;

    //free any allocated memory for test object(s)
    delete testT;
}
```

Album.h

```
/***************************************************************************
 *
 *  Programming2 Coursework 2
 *
 *  Title:                 Music Database in C++
 *
 *  File:                  Album.h
 *
 *  Description:           Models a music album (a collection of tracks)
 *                         Paired with Album.cpp
 *
 *  Author:                100166648 / ssq16shu
 *
 *  Version History:       v0.1    190412  instance, constructor and accessors
 *                         v1.0    190415  mutators, ostream, extra constructors,
 *                                         add and print methods. completed testing
 *                         v2.0    190422  removed mutators added istream overload
 *                         v2.1    190425  moved istream to cpp file.
 *                                         removed failbit error messages.
 *                         v3.0    190426  changed ostream formatting to mimic
 *                                         input formatting of istream overload
 *                         v4.0    190427  added '<' overload to be used for album
 *                                         sorting (alphabetically: artist, album)
 *
 *  Notes:                 No other relational operators (other than '<') have been
 *                         implementewd here as they are never needed. Less-than
 *                         has been utilised to override the default sorting of
 *                         album objects (removing the need for an additional
 *                         comparison method)
 *
 ***************************************************************************/

#ifndef ALBUMDB_CL_ALBUM_H
#define ALBUMDB_CL_ALBUM_H

//include file(s))
#include "Track.h"

#include <vector>

class Album {
    //instance variables
    string artist;
    string albumTitle;
    std::vector<Track> tracks;

public:
    //-----Declarations
    //constructors
    Album();
    Album(string artist, string title);
    Album(string artist, string title, std::vector<Track> tracks);

    //accessors
    string getArtist() const;
    string getTitle() const;
    std::vector<Track> getTracks() const;

    //cpp file methods
    void addTrack(Track track);
    void printTracks() const;
    //cpp file overload(s)
    friend std::istream& operator>>(std::istream& is, Album &a);

    //testing
    static void testHarnessAlbum();
};


//-----Definitions
//--constructors
//empty (for testing: may remove)
inline Album::Album(){
    this->artist = "noArtist";
    this->albumTitle = "noAlbumTitle";
}
//artist and title supplied
inline Album::Album(string artist, string title){
    this->artist = artist;
    this->albumTitle = title;
}
//create album from existing list(vector) of tracks
inline Album::Album(string artist, string title, std::vector<Track> tracks){
    this->artist = artist;
    this->albumTitle = title;
    this->tracks = tracks;
}

//--accessors
inline string Album::getArtist() const{
    return artist;
}
inline string Album::getTitle() const{
    return albumTitle;
}
inline std::vector<Track> Album::getTracks() const{
    return tracks;
}

//-----relational operator overload (used for sorting albums alphabetically)
inline bool operator < (const Album& a1, const Album  a2) {
    if (a1.getArtist() == a2.getArtist()){
        return (a1.getTitle() < a2.getTitle());
    }else{
        return (a1.getArtist() < a2.getArtist());
    }
}


//-----IO stream operator overloading
//--output stream
//formats console output of object (as 'artist : albumTitle' followed by tracks)
inline std::ostream& operator<<(std::ostream& os, const Album &a){
    std::cout << a.getArtist() << " : " << a.getTitle() << std::endl;
    a.printTracks();
    return os;

}

#endif //ALBUMDB_CL_ALBUM_H
```

Album.cpp

```
/*******************************************************************************
 *
 *   Programming2 Coursework 2
 *
 *   Title:               Music Database in C++
 *
 *   File:                Album.cpp
 *
 *   Description:         Models a music album (a collection of tracks)
 *                        Paired with Album.h
 *
 *   Author:              100166648 / ssq16shu
 *
 *   Version History:     v0.1    190412   initial skeleton
 *                        v1.0    190415   added test harness
 *                        v2.0    190422   testing of istream input
 *                        v3.0    190426   moved istream overload to cpp file
 *                        v3.1    190430   removed obsolete methods
 *
 *   Notes:
 *
 *******************************************************************************/


//-----inclusions
#include "Album.h"

#include <algorithm>

using namespace std;

//-----Overload(s)
//--input stream
//checks input format and creates album object if correct
std::istream& operator>>(std::istream& is, Album &a){

    //input storage
    string artist, albumTitle;
    char c = ':';
    //check stream input successfully to strings (':' delimited)
    if(std::getline(is,artist,c) && std::getline(is,albumTitle,'\r')){
        //remove trailing whitespace from artist
        artist.erase(artist.size()-1);
        //create object using substrings
        a = Album(artist,albumTitle);

        Track testTrack;
        while(is>>testTrack){
            a.addTrack(testTrack);
        }
        //clear failbit (to continue to next album)
        is.clear();
    }
    //return istream
    return is;
}

//adds track to album's track vector
void Album::addTrack(Track track){
    this->tracks.push_back(track);
}
```

```cpp
//prints track objects in album to console
void Album::printTracks() const {
    for(int i=0; i< this->tracks.size(); i++){
        cout<<"\t\t"<<this->tracks.at(i) <<endl;
    }
}


//-----testing
void Album::testHarnessAlbum(){
    cout<<"------------------------------------------"<<endl;
    cout<<"     Album class test harness:"<<endl;
    cout<<"------------------------------------------"<<endl;
    cout<<"testing Album constructors:"<<endl;
    Album* testA = new Album();
    cout<<"default : "<< *testA<<endl;
    *testA = Album("testingArtist", "testingTitle");
    cout<<"artist, title, using accessors: "<<endl;
    cout<<testA->getArtist()<<endl;
    cout<<testA->getTitle()<<endl;
    cout<<endl;

    cout<<"testing add to track vector..."<<endl;
    //create test Tracks and push to vector
    Track* testTa = new Track("testTrack1",Duration(1,2,3));
    Track* testTb = new Track("testTrack2",Duration(2,3,4));
    Track* testTc = new Track("testTrack1",Duration(3,4,5));
    testA->addTrack(*testTa);
    testA->addTrack(*testTb);
    testA->addTrack(*testTc);
    cout<<"printing tracklist:"<<endl;
    testA->printTracks();
    cout<<endl;


    cout<<"testing istream overload"<<endl;
    Album testAlb;
    cout<<"please enter album info to test"<<endl;
    cout<<"followed by either track listings or other alpha character\n"
          "(to break track-input loop)"<<endl;
    cin >>testAlb;
//    console-in code: commented-out after manual format testing complete
//        while(!(cin >> testAlb)){
//            cout<<"enter album"<<endl;
//        cin.clear();
//        cin.ignore(256,'\n');
//    }
//
//    cout<<"entered album details:"<<endl;
//    cout<<testAlb;
//    cout<<endl;
//    Track testTr;
//    cout<<"add track lines to album"<<endl;
//    while(cin>>testTr){
//        testAlb.addTrack(testTr);
//    }
//    cout<< "album with track listing:"<<endl;

    cout<<"testing track input to album"<<endl;
    cout<< testAlb <<endl;
    cout<<endl;
    cout<<"testing comparison function"<<endl;
    Album a1 = Album("testartist1","testAlbum1");
    Album a1_1 = Album("testartist1","testAlbum2");
    Album a2 = Album("testartist2","testAlbum1");
    Album a2_1 = Album("testartist2","testAlbum2");

    vector<Album> sortTest;
    sortTest.push_back(a2);
    sortTest.push_back(a1_1);
    sortTest.push_back(a2_1);
    sortTest.push_back(a1);
    cout<<"before:"<<endl;
    for (int i = 0; i < sortTest.size(); ++i) {
        cout<<sortTest.at(i)<<endl;
    }
    cout<<"attempting to sort"<<endl;
//obsolete comparison method removed by overloading '<' operator
//     std::sort(sortTest.begin(),sortTest.end(),albumCompare);
    cout<<"after:"<<endl;
    for (int i = 0; i < sortTest.size(); ++i) {
        cout<<sortTest.at(i)<<endl;
    }

    //delete test objects
    delete testA;
    delete testTa;
    delete testTb;
    delete testTc;
}
```

Collection.h

```
/******************************************************************************
 *
 *  Programming2 Coursework 2
 *
 *  Title:              Music Database in C++
 *
 *  File:               Collection.h
 *
 *  Description:        Models a collection of music albums
 *                      Paired with Collection.cpp
 *
 *  Author:             100166648 / ssq16shu
 *
 *  Version History:    v0.1    190412  initial skeleton
 *                      v1.0    190415  constructors, accessors
 *                      v1.1    190422  ostream overload
 *                      v2.0    190423  istream overload
 *                      v2.1    190425  moved istream to cpp file.
 *                                      removed failbit error messages
 *                      v3.0    190428  changed sorting method to overloaded '<'
 *                      v3.1    190430  removed obsolete methods
 *
 *  Notes:
 *
 ******************************************************************************/

#ifndef ALBUMDB_CL_COLLECTION_H
#define ALBUMDB_CL_COLLECTION_H

//include files
#include "Album.h"

#include <vector>
#include <fstream>
#include <iostream>


class Collection {
    //instance variables
    std::vector<Album> albumCollection;

public:
    //-----Declarations
    //constructors
    Collection();
    Collection(std::vector<Album> albums);

    //accessors
    std::vector<Album> getCollection() const;

    //cpp file methods
    void addAlbum(Album album);
    void printCollection() const;
    void printAll() const;
    void longestTrack() const;
    void totalTime(std::string artist) const;
    void largestTrackcount() const;
    void printAlphabetical() const;

    //cpp file overload(s)
    friend std::istream &operator>>(std::istream &is, Collection &col);
```

```cpp
    //testing
    static void testHarnessCollection();
};

//-----Definitions
//--constructors
//empty
inline Collection::Collection() = default;

//passed an existing collection (used for testing: could be removed)
inline Collection::Collection(std::vector<Album> albums) {
    this->albumCollection = albums;
}

//--accessors
inline std::vector<Album> Collection::getCollection() const {
    return this->albumCollection;
}

//-----IO stream operator overloading
//--output stream
//formats console output of object (as)
inline std::ostream &operator<<(std::ostream &os, const Collection &c) {
    if (!c.getCollection().empty()) {
        c.printCollection();
    } else {
        std::cout << "collection is empty." << std::endl;
    }
    return os;
}

#endif //ALBUMDB_CL_COLLECTION_H
```

Collection.cpp

```cpp
/******************************************************************************
 *
 *  Programming2 Coursework 2
 *
 *  Title:              Music Database in C++
 *
 *  File:               Collection.cpp
 *
 *  Description:        Models a collection of music albums
 *                      Paired with Collection.h
 *
 *  Author:             100166648 / ssq16shu
 *
 *  Version History:    v0.1    190412  initial skeleton
 *                      v1.0    190415  add and print methods, completed testing
 *                      v1.1    190422  testing of ostream overload
 *                      v2.0    190423  album created. stops after first album
 *                      v2.1    190424  fixed. duplicate final album bug.
 *                      v2.2    190425  fixed. moved file opening to test/main
 *                      v3.0    190428  changed sorting method to copy vector
 *                      v3.1    190430  removed obsolete methods
 *
 *  Notes:
 *
 ******************************************************************************/

//-----h file inclusions
#include "Collection.h"

#include <sstream>
#include <algorithm>

using namespace std;

//-----Overload(s)
//--input stream
//checks input format and creates album object if correct
inline std::istream &operator>>(std::istream &is, Collection &col) {
    //input storage
    std::string fileName;
    std::string line;

    //album object
    Album album;
    while (is >> album) {
        col.addAlbum(album);
    }
    //return istream
    return is;
}


//-----Class methods
//adds album to collection's vector
void Collection::addAlbum(Album album) {
    this->albumCollection.push_back(album);
}

//prints album object details to console
void Collection::printCollection() const {
    cout << "Albums in Collection:" << endl;
    for (int i = 0; i< this->albumCollection.size(); i++) {
        cout << "\t" << this->albumCollection.at(i) << endl;
    }
}

//prints details of longest(duration) track in collection
void Collection::longestTrack() const {
    //current longest track storage
    Track longest;
    //current album info storage (print formatting: additional info)
    ostringstream album, title;

    //for all albums in collection
    for (int i = 0; i < this->albumCollection.size(); ++i) {
        //for all tracks in album
        for (int j =0; j<this->albumCollection.at(i).getTracks().size(); ++j) {
            //compare to current longest track (and replace)
            if (this->albumCollection.at(i).getTracks().at(j) > longest) {
                longest = this->albumCollection.at(i).getTracks().at(j);
                //clear and replace ostringstreams
                album.str("");
                title.str("");
                album << this->albumCollection.at(i).getTitle();
                title << this->albumCollection.at(i).getArtist();
            }
        }
    }
    //print details of longest track in collection
    std::cout << "Longest track in album collection:\n" << longest << std::endl;
    std::cout << "\t(" << title.str() << " : " << album.str() << ")" << endl;
}

//prints total run time of artist ("pink Floyd" specified in main() )
void Collection::totalTime(std::string artist) const {
    //total run-time storage
    Duration total;
    //for all albums
    for (int i = 0; i < this->albumCollection.size(); ++i) {
        //check for matching artist
        if (this->albumCollection.at(i).getArtist() == artist) {
            //for each track in album
            for(int j=0; j<this->albumCollection.at(i).getTracks().size(); ++j){
                //add to total runtime sum
                total = total + this->albumCollection.at(i)
                        .getTracks().at(j).getDuration();
            }
        }
    }
    std::cout << "Total runtime of all " << artist << " albums in collection:\n"
            << total << std::endl;
}

//prints details of album with largest number of tracks
void Collection::largestTrackcount() const {
    //album-index storage;
    int index = 0;
    //track counter
    int count = 0;
    //for all albums
    for (int i = 0; i < this->albumCollection.size(); ++i) {
        if (this->albumCollection.at(i).getTracks().size() > count) {
            //update highest track count, and index of album
            count = this->albumCollection.at(i).getTracks().size();
            index = i;
        }
    }
    std::cout << "Album with the largest number of tracks (" << count << "):\n"
            << this->albumCollection.at(index) << std::endl;
}

//sorts and prints album collection alphabetically
void Collection::printAlphabetical() const {
    std::vector<Album> copy = this->getCollection();
    //sort collection using album comparison method
    std::sort(copy.begin(), copy.end());
    //print sorted collection
    std::cout << copy;
}

//-----testing
void Collection::testHarnessCollection() {
    cout << "----------------------------------------" << endl;
    cout << "     Collection class test harness:" << endl;
    cout << "----------------------------------------" << endl;
    Collection* testC = new Collection();
    cout << "creating albums to add to collection..." << endl;
    Album* testA1 = new Album("testartist1", "testTitle1");
    Album* testA2 = new Album("testartist2", "testTitle2");
    Album* testA3 = new Album("testartist3", "testTitle3");
    testC->addAlbum(*testA1);
    testC->addAlbum(*testA2);
    testC->addAlbum(*testA3);
    cout << "print collection:" << endl;
    testC->printCollection();
    cout << endl;

    cout << "testing ostream overload" << std::endl;
    cout << *testC;
    cout << endl;

    //    console-in code: commented-out after manual format testing complete
    //    cout<<"enter collection text file (including suffix)"<<endl;
    //    while(!(fileName >> tC)){
    //        cin.clear();
    //        cin.ignore(256,'\n');
    //    }

    cout << "testing istream overload" << endl;
    Collection tC;
    //string for input
    std::string fileName("albums.txt");
    //openfile
    std::ifstream inputFile;
    inputFile.open(fileName);
    //read-in to collection
    inputFile>>tC;
    //close file
    inputFile.close();
    std::cout << std::endl;

    std::cout << "testing all albums added to collection:" << std::endl;
    tC.printCollection();
    std::cout << std::endl;

    std::cout << "testing each album is populated:" << std::endl;
    //testing first album
    cout << "checking first album contents" << endl;

    //album details
    cout << tC.getCollection().at(0) << endl;
    cout << endl;

    //    cout<<"print all album contents in loop"<<endl;
    //    //for each album in collection
    //    for (int i =0; i < tC.getCollection().size(); i++){
    //        //print album details
    //        cout<<tC.getCollection().at(i)<<std::endl;
    //        //print track details
    //        tC.getCollection().at(i).printTracks();
    //        cout<<"--------------------------------"<<endl;
    //    }
    //    std::cout<<std::endl;

    std::cout << "testing printAll method (all albums and their track listings)"
            << std::endl;
    //note: printall method replaced with overload of collection ostream
    //tC.printAll();
    cout << tC << endl;
    cout << endl;

    std::cout << "testing longestTrack():" << std::endl;
    tC.longestTrack();
    std::cout << std::endl;

    std::cout << "testing total runtime" << std::endl;
    std::string artist = "Miles Davis";
    tC.totalTime(artist);
    std::cout << std::endl;

    std::cout << "testing largestTrackcount()" << std::endl;
    tC.largestTrackcount();
    std::cout << std::endl;

    std::cout << "testing alphabetical" << std::endl;
    tC.printAlphabetical();

    //free allocated memory for test object(s)
    delete testC;
    delete testA1;
    delete testA2;
    delete testA3;
}
```

main.cpp

```cpp
/*****************************************************************************
 *
 *  Programming2 Coursework 2
 *
 *  Title:              Music Database in C++
 *
 *  File:               main.cpp
 *
 *  Description:        class contains method call to populate music database
 *                      from text file.
 *                      Calls methods on album collection to complete tasks
 *                      required by coursework spec sheet.
 *                      Includes testharness method calls for all classes
 *
 *  Author:             100166648 / ssq16shu
 *
 *  Version History:    v0.1    190412  initial skeleton
 *                      v1.0    190428  testing complete and commented out.
 *                                      coursework task method calls added.
 *
 *  Notes:          -   testHarness calls are commented-out for clarity and to
 *                      allow for PASS submission (may require user input)
 *
 *****************************************************************************/

//-----include(s)
#include "Duration.h"
#include "Track.h"
#include "Album.h"
#include "Collection.h"

#include <iostream>

int main() {
//open text file
    std::string fileName = "albums.txt";
    std::ifstream inputFile;
    inputFile.open(fileName);
//check for errors opening file
    if (!inputFile) {
        std::cerr << "Unable to open input file." << std::endl;
        return 1;
    }
//collection object
    Collection collection;
//read file into collection
    inputFile>>collection;
//close file
    inputFile.close();

//display entire collection in alphabetical order
    std::cout<<"-----------------------------------------"<<std::endl;
    std::cout<<"Print entire collection in alphabetical order:"<<std::endl;
    std::cout<<"-----------------------------------------"<<std::endl;
    std::cout<<std::endl;

    collection.printAlphabetical();
    std::cout<<std::endl;

//display total play time of all pink floyd albums
    std::cout<<"-----------------------------------------"<<std::endl;
    std::cout<<"Display total play time for artist (Pink Floyd)"<<std::endl;
    std::cout<<"----------------------------------------"<<std::endl;
    std::cout<<std::endl;

    //string to hold artist requested (avoid user input for PASS)
    std::string artist = "Pink Floyd";
    collection.totalTime(artist);
    std::cout<<std::endl;

//display album with largest number of tracks
    std::cout<<"----------------------------------------"<<std::endl;
    std::cout<<"Display album with greatest number of tracks"<<std::endl;
    std::cout<<"----------------------------------------"<<std::endl;
    std::cout<<std::endl;

    collection.largestTrackcount();
    std::cout<<std::endl;

//display details of longest track in album collection
    std::cout<<"----------------------------------------"<<std::endl;
    std::cout<<"Display details of longest track in collection"<<std::endl;
    std::cout<<"----------------------------------------"<<std::endl;
    std::cout<<std::endl;

    collection.longestTrack();
    std::cout<<std::endl;


//Class testing harnesses
//(commented-out after testing)
//-----------------------------------------------------
//      Duration::testHarnessDuration();
//      Track::testHarnessTrack();
//      Album::testHarnessAlbum();
//      Collection::testHarnessCollection();

        return 0;
    }
```

output.txt

```
----------------------------------------
Print entire collection in alphabetical order:
----------------------------------------

Albums in Collection:
        Blondie :  Parallel Lines
                Hanging on the Telephone - 00:02:17
                One Way or Another - 00:03:31
                Picture This - 00:02:53
                Fade Away and Radiate - 00:03:57
                Pretty Baby - 00:03:16
                I Know But I Don't Know - 00:03:53
                11:59 - 00:03:19
                Will Anything Happen? - 00:02:55
                Sunday Girl - 00:03:01
                Heart of Glass - 00:03:54
                I'm Gonna Love You Too - 00:02:03
                Just Go Away - 00:03:21

        Goldfrapp :  Supernature
                Ooh La La - 00:03:24
                Lovely 2 C U - 00:03:25
                Ride a White Horse - 00:04:41
                You Never Know - 00:03:27
                Let It Take You - 00:04:30
                Fly Me Away - 00:04:25
                Slide In - 00:04:17
                Koko - 00:03:23
                Satin Chic - 00:03:28
                Time Out from the World - 00:04:47
                Number 1 - 00:03:25

        Jordi Savall & Hesperion XX :  Folias and Canarios
                Folias Pavana Con Su Glosa (Antonio De Cabezon) - 00:02:06
                Fantasia (Alonso Mudarra) - 00:02:49
                Tiento De Falsas, (Joan Cabanilles) - 00:04:05
                Jacaras (Gaspar Sanz) - 00:02:14
                Gaspar Sanz     - 00:02:09
                Paduana Del Re (Anonyme) - 00:02:17
                Anonyme - 00:01:14
                Arpegiatta (Girolamo Kapsberger) - 00:01:46
                Gallarda (Giacomo De Gorzanis) - 00:01:38
                Girolamo Capsberger - 00:02:12
                Si Ay Perdut Mon Saber (Ponc D'Ortafa/Jordi Savall) - 00:04:03
                La Mariagneta (Anon/Jordi Savall) - 00:01:53
                Con Que La Lavare (Anonyme) - 00:02:23
                El Pare I La Mare (Anonyme/Jordi Savall)   - 00:03:45
                Paradetas (Lucas Ruiz De Ribadayaz/Andrew Lawrence King) - 00:03:26
                Clarines Y Trompetas (Gaspar Sanz) - 00:05:34
                Fantasia (Joan Cabanilles) - 00:02:36
                Toccata & Chiaccona (Alessandro Piccinini) - 00:03:47
                Todo El Mundo En General (Francisco Correa De Arrauxo) - 00:04:00
                Canarios (Anonyme/Jordi Savall) - 00:02:18

        Kraftwerk :  The Man Machine
                The Robots - 00:06:11
                Spacelab - 00:05:51
                Metropolis - 00:05:59
                The Model - 00:03:38
                Neon Lights - 00:09:03
                The Man-Machine - 00:05:28
```

```
Kraftwerk :  Trans Europe Express
        Europe Endless ("Europa Endlos") - 00:09:42
        The Hall of Mirrors ("Spiegelsaal") - 00:07:57
        Showroom Dummies ("Schaufensterpuppen") - 00:06:15
        Trans-Europe Express ("Trans Europa Express") - 00:06:36
        Metal on Metal ("Metall auf Metall") - 00:02:12
        Abzug - 00:04:55
        Franz Schubert - 00:04:26
        Endless Endless ("Endlos Endlos") - 00:00:58

Led Zeppelin :  Led Zeppelin IV
        Black Dog - 00:04:54
        Rock and Roll - 00:03:40
        The Battle of Evermore - 00:05:51
        Stairway to Heaven - 00:08:02
        Misty Mountain Hop - 00:04:38
        Four Sticks - 00:04:44
        Going to California - 00:03:31
        When the Levee Breaks - 00:07:07

Marillion :  Script for a Jester's Tear
        Script for a Jester's Tear - 00:08:44
        He Knows You Know - 00:05:23
        The Web - 00:08:52
        Garden Party - 00:07:19
        Chelsea Monday - 00:08:17
        Forgotten Sons - 00:08:23

Miles Davis :  Kind of Blue
        So What - 00:09:22
        Freddie Freeloader - 00:09:46
        Blue in Green - 00:05:37
        All Blues - 00:11:33
        Flamenco Sketches - 00:09:26

Neil Pye :  Neil's Heavy Concept Album
        Hello Vegetables - 00:00:26
        Hole In My Shoe - 00:03:40
        Heavy Potato Encounter - 00:00:42
        My White Bicycle - 00:03:31
        Neil the Barbarian - 00:01:12
        Lentil Nightmare - 00:05:47
        Computer Alarm - 00:00:36
        Wayne - 00:01:36
        The Gnome - 00:02:29
        Cosmic Jam - 00:02:26
        Golf Girl - 00:04:40
        Bad Karma in the UK - 00:02:17
        Our Tune - 00:01:13
        Ken - 00:00:41
        The End of the World Cabaret - 00:01:09
        No Future (God Save the Queen) - 00:02:12
        Floating - 00:01:39
        Hurdy Gurdy Man - 00:03:46
        Paranoid Remix - 00:01:59
        The Amoeba Song - 00:01:19

Nimoy & Shatner :  Spaced Out
        King Henry The Fifth - 00:03:00
        Elegy For The Brave - 00:03:18
        Highly Illogical - 00:02:20
        If I Had A Hammer [The Hammer Song] - 00:02:08
        Mr Tambourine Man - 00:02:50
        Where Is Love - 00:02:01
        Music To Watch Space Girls By - 00:02:22
        It Was A Very Good Year - 00:03:56
        Ruby Don't Take Your Love To Town - 00:02:47
        Hamlet - 00:03:50
        A Visit To A Sad Planet - 00:03:02
        Abraham, Martin and John - 00:03:20
        Lucy In The Sky With Diamonds - 00:02:58
        If I Was A Carpenter - 00:02:41
        How Insensitive - 00:03:32
        I'd Love Making Love To You - 00:02:53
        Put A Little Love In Your Heart - 00:02:30
        Sunny - 00:03:20
        Gentle On My Mind - 00:02:46
        I Walk The Line - 00:02:17
        Ballad Of Bilbo Baggins - 00:02:19
        Everybody's Talkin' - 00:02:58
        Both Sides Now - 00:02:53
        Spock Thoughts - 00:03:04

Pink Floyd :  Animals
        Pigs on the Wing (Part I) - 00:01:25
        Dogs - 00:17:03
        Pigs (Three Different Ones) - 00:11:25
        Sheep - 00:10:25
        Pigs on the Wing (Part II) - 00:01:23

Pink Floyd :  Dark Side of the Moon
        Speak to Me - 00:01:30
        Breathe - 00:02:43
        On the Run - 00:03:36
        Time - 00:07:01
        The Great Gig in the Sky - 00:04:36
        Money - 00:06:22
        Us and Them - 00:07:46
        Any Colour You Like - 00:03:25
        Brain Damage - 00:03:48
        Eclipse - 00:02:03

Pink Floyd :  Meddle
        One of These Days - 00:05:57
        A Pillow of Winds - 00:05:07
        Fearless - 00:06:05
        San Tropez - 00:03:40
        Seamus - 00:02:13
        Echoes - 00:23:31

Pink Floyd :  Momentary Lapse of Reason
        Signs of Life - 00:04:24
        Learning to Fly - 00:04:53
        The Dogs of War - 00:06:05
        One Slip - 00:05:10
        On The Turning Away - 00:05:42
        Yet Another Movies/Round and Around - 00:07:28
        A New Machine, Part 1 - 00:01:46
        Terminal Frost - 00:06:17
        A New Machine, Part 2 - 00:00:38
        Sorrow - 00:08:46

Pink Floyd :  Wish You Were Here
        Shine On You Crazy Diamond (Part One) - 00:13:33
        Welcome to The Machine - 00:07:26
        Have A Cigar - 00:05:07
        Wish You Were Here - 00:05:40
        Shine On You Crazy Diamond (Part Two) - 00:12:21

Pulp :  Different Class
        Mis-Shapes - 00:03:46
        Pencil Skirt - 00:03:11
        Common People - 00:05:50
        I Spy - 00:05:55
        Disco 2000 - 00:04:33
        Live Bed Show - 00:03:29
        Something Changed - 00:03:18
        Sorted for E's & Wizz - 00:03:47
        F.E.E.L.I.N.G.C.A.L.L.E.D.L.O.V.E - 00:06:01
        Underwear - 00:04:06
        Monday Morning - 00:04:16
        Bar Italia - 00:03:42

The Beatles :  Rubber Soul
        Drive My Car - 00:02:25
        Norwegian Wood (This Bird Has Flown) - 00:02:01
        You Won't See Me - 00:03:18
        Nowhere Man - 00:02:40
        Think for Yourself - 00:02:16
        The Word - 00:02:41
        Michelle - 00:02:40
        What Goes On - 00:02:47
        Girl - 00:02:30
        I'm Looking Through You - 00:02:23
        In My Life - 00:02:24
        Wait - 00:02:12
        If I Needed Someone - 00:02:20
        Run for Your Life - 00:02:18

The Dave Brubeck Quartet :  Take Five
        Blue Rondo a la Turk - 00:06:44
        Strange Meadow Lark - 00:07:22
        Take Five - 00:05:24
        Three to Get Ready - 00:05:24
        Kathy's Waltz - 00:04:48
        Everybody's Jumpin' - 00:04:23
        Pick Up Sticks - 00:04:16

The Jimi Hendrix Experience :  Are you Experienced?
        Foxy Lady - 00:03:22
        Manic Depression - 00:03:46
        Red House - 00:03:53
        Can You See Me - 00:02:35
        Love or Confusion - 00:03:17
        I Don't Live Today - 00:03:58
        May This Be Love - 00:03:14
        Fire - 00:02:47
        Third Stone from the Sun - 00:06:50
        Remember - 00:02:53
        Are You Experienced? - 00:04:17
        Hey Joe (Billy Roberts) - 00:03:30
        Stone Free - 00:03:36
        Purple Haze - 00:02:51
        51st Anniversary - 00:03:15
        The Wind Cries Mary - 00:03:20
        Highway Chile - 00:03:32
```

```
----------------------------------------
Display total play time for artist (Pink Floyd)
----------------------------------------

Total runtime of all Pink Floyd albums in collection:
03:46:20

----------------------------------------
Display album with greatest number of tracks
----------------------------------------

Album with the largest number of tracks (24):
Nimoy & Shatner :  Spaced Out
                King Henry The Fifth - 00:03:00
                Elegy For The Brave - 00:03:18
                Highly Illogical - 00:02:20
                If I Had A Hammer [The Hammer Song] - 00:02:08
                Mr Tambourine Man - 00:02:50
                Where Is Love - 00:02:01
                Music To Watch Space Girls By - 00:02:22
                It Was A Very Good Year - 00:03:56
                Ruby Don't Take Your Love To Town - 00:02:47
                Hamlet - 00:03:50
                A Visit To A Sad Planet - 00:03:02
                Abraham, Martin and John - 00:03:20
                Lucy In The Sky With Diamonds - 00:02:58
                If I Was A Carpenter - 00:02:41
                How Insensitive - 00:03:32
                I'd Love Making Love To You - 00:02:53
                Put A Little Love In Your Heart - 00:02:30
                Sunny - 00:03:20
                Gentle On My Mind - 00:02:46
                I Walk The Line - 00:02:17
                Ballad Of Bilbo Baggins - 00:02:19
                Everybody's Talkin' - 00:02:58
                Both Sides Now - 00:02:53
                Spock Thoughts - 00:03:04


----------------------------------------
Display details of longest track in collection
----------------------------------------

Longest track in album collection:
Echoes - 00:23:31
        (Pink Floyd :  Meddle)


RUN SUCCESSFUL (total time: 250ms)
```