Music Player 명세서

프로그램 구조는 Application class에는 음악전체를 관리하는 m\_MasterList 그리고 나만의 음악리스트인 MusicList가 있다. 또한 MusicList에 음악을 추가할 때마다 추가되는 GenreList, AlbumList, ArtistList, SongList가 존재한다. 음악을 재생할 때 최근 플레이한 리스트인 PlayRecentList, 내 MusicList에 추가될 때 볼 수 있는 AddRecentlyMusicList , 음악을 재생한 시간에 따라서 정렬된 DeleteList가 있다. 그리고 로그인과 회원가입을 위한 MemberList가 존재한다.

MusicType 에는 음악에 대한 자세한 정보들이 담겨있고, Album, Genre, Song, Artist Type은 baseclass를 이용해서 상속을 시켰다.

PlayTime 에는 재생시간을 변수로 두어서 DeleteList를 구현할 때 이용하였다.

Login은 회원가입을 위한 변수들을 만들어서 MemberList를 구현할 때 이용하였다.

이용한 자료구조로는 BinarySearchTree, Queue, LinkedList를 이용하였다.

프로그램을 실행 할 때는 로그인, 회원가입으로 시작을 하고

관리자로 로그인 할 시에 관리자 모드로 넘어간다.

관리자 모드는

1 : 음악 리스트(Master List)에 추가

2 : 음악 리스트(Master List)에서 삭제 "\t

3 : 음악 리스트(Master List)에서 갱신"

4 : 파일에서 데이터를 읽어줍니다"

5 : 파일에서 데이터를 씁니다"

6 : 모든 음악을 보여줍니다";

7 : ID를 이용하여 자세한 음악 정보를 검색해합니다"

로 이루어져 있다.

유저모드로 로그인할 경우에는

1 : 음악 재생"

2 : 나만의 음악 리스트에 추가

3 : 나만의 음악 리스트에서 삭제"

4 : 나만의 음악 리스트를 보여줍니다"

5 : 최근에 추가한 나만의 음악 리스트"

6 : 최근에 재생한 나만의 음악 리스트"

7 : 삭제를 추천하는 음악 리스트"

8 : 노래제목으로 검색"

9 : 가수이름으로 검색"

10 : 앨범이름으로 검색"

11 : 장르이름으로 검색"

12 : 장르별로 출력"

13 : 앨범별로 출력"

14 : 노래제목별로 출력"

15 : 가수이름별로 출력"

로 이루어져 있다.

관리자 ID로 로그인 했을 경우 MasterList를 수정 삭제 추가 가능하다.

음원의 PrimaryKey는 음악의 첫 제목 + 앨범의 수록곡 수로 정의한다.

각 클래스의 ADT는 다음과 같다.

Application Class

\*/

class Application

{

public:

/\*\*

\* default constructor.

\*/

Application()

{

m\_Command = 0;

}

/\*\*

\* destructor.

\*/

~Application() {}

/\*\*

\* @brief Program driver.

\* @pre program is started.

\* @post program is finished.

\*/

void Run();

/\*\*

\* @brief Display command on screen and get a input from keyboard.

\* @pre none.

\* @post none.

\* @return user's command.

\*/

int GetCommand();

/\*\*

\* @brief Add new record into musiclist & gerelist

\* @pre list should be initialized.

\* @post new record is added into the list.

\* @return return 1 if this function works well, otherwise 0.

\*/

int AddmyMusic();

/\*\*

\* @brief Display all records in the list on screen.

\* @pre none.

\* @post none.

\*/

void DisplayAllMusic();

/\*\*

\* @brief Open a file by file descriptor as an input file.

\* @pre a file for reading is exist.

\* @post open the file for reading.

\* @param fileName a filename to open for reading.

\* @return return 1 if this function works well, otherwise 0.

\*/

int OpenInFile(char \*fileName);

/\*\*

\* @brief Open a file by file descriptor as an output file.

\* @pre list should be initialized.

\* @post open the file for writing.

\* @param fileName a filename to open for writing.

\* @return return 1 if this function works well, otherwise 0.

\*/

int OpenOutFile(char \*fileName);

/\*\*

\* @brief Open a file as a read mode, read all data on the file, and set list by the data.

\* @pre The file is not opened.

\* @post list holds all records from the file.

\* @return return 1 if this function works well, otherwise 0.

\*/

int ReadDataFromFile();

/\*\*

\* @brief Open a file as a write mode, and write all data into the file,

\* @pre The file is not opened.

\* @post the list is stored in the output file.

\* @return return 1 if this function works well, otherwise 0.

\*/

int WriteDataToFile();

/\*\*

\*@brief 사용자에게 노래제목을 입력받아 SortedList의 Get함수에 넘겨주는 역할을 한다.

\* @pre 없다.

\* @post 불완전한 data의 정보가 모두 채워지고 그 결과가 출력된다.

\* @return return 1 if this function works well, otherwise 0.

\*/

void SearchByName();

/\*\*

\* @brief 사용자에게 가수를 입력받아 SortedList의 Get함수에 넘겨주는 역할을 한다.

\* @pre 없다.

\* @post 불완전한 data의 정보가 모두 채워지고 그 결과가 출력된다.

\* @return return 1 if this function works well, otherwise 0.

\*/

void SearchByArtist();

/\*\*

\* @brief 사용자에게 장르를 입력받아 SortedList의 Get함수에 넘겨주는 역할을 한다.

\* @pre 없다.

\* @post 불완전한 data의 정보가 모두 채워지고 그 결과가 출력된다.

\* @return return 1 if this function works well, otherwise 0.

\*/

void SearchByGenre();

/\*\*

\* @brief 사용자에게 앨범을 입력받아 SortedList의 Get함수에 넘겨주는 역할을 한다.

\* @pre 없다.

\* @post 불완전한 data의 정보가 모두 채워지고 그 결과가 출력된다.

\* @return return 1 if this function works well, otherwise 0.

\*/

void SearchByAlbum();

/\*\*

\* @brief 사용자에게 노래 제목을 입력받아 Delete함수에 넘겨주는 역할을 한다.

\* @pre 없다.

\*@post 사용자가 입력한 노래 제목을 가지고 있는 MusicType이 m\_MusicList에서 사라진다.

\* @return 없다.

\*/

void DeletemyMusic();

/\*\*

\*@brief 사용자에게 노래 제목을 입력받아 SortedList의 Replace함수에 넘겨주는 역할을 한다.

\* @pre 없다.

\*@post 사용자가 입력한 노래 제목을 가지고 있는 MusicType을 다시 입력받아 갱신한다.

\* @return 없다.

\*/

void ReplaceMusic();

/\*\*

\*@brief 최근 재생목록(30개)를 보여준다.

\*@pre 리스트가 초기화 되어있어야한다.

\*@post AddRecentlyMusicList의 정보로 m\_GenreMusicList에서 찾아 모든 정보 출력

\*/

int DisplayNewMusic();

/\*\*

\*@brief 장르를 기준으로 음악 정보를 화면에 출력. 그리고 자세한 정보를 원한다면 출력

\*@pre 리스트가 초기화 되어있어야한다.

\*@post 장르를 기준으로 음악 정보가 화면에 출력된다.

\*/

void DisplayMusicByGenre();

/\*\*

\*@brief 장르가 일치하는 음악 정보를 찾아서 화면에 출력.(musiclist를 이용해서 비교)

\*@pre list should be initialized.

\*@post 직접 입력받은 장르가 일치하는 음악 정보를 찾아서 화면에 출력한다.

\*/

int AddMasterMusic();

/\*\*

\* @brief Display all my records(music list) in the list on screen.

\* @pre none.

\* @post none.

\*/

void DisplayAllmyMusic();

/\*\*

\* @brief 파라미터에서 가져온 데이터를 앨범리스트에 추가.

\*/

void AddMusictoAlbumList(MusicType item);

/\*\*

\* @brief 파라미터에서 가져온 데이터를 아트리스트리스트에 추가.

\*/

void AddAlbumToArtistList(AlbumType& item);

/\*\*

\* @brief 파라미터에서 가져온 데이터를 앨범리스트에 삭제.

\*/

void DeleteMusicToGenreList(MusicType selectItem);

/\*\*

\* @brief 파라미터에서 가져온 데이터를 앨범리스트에 삭제.

\*/

void DeleteMusicToAlbumList(MusicType selectItem);

/\*\*

\* @brief 파라미터에서 가져온 데이터를 뮤직리스트에 삭제.

\*/

void DeleteMusicToSongList(MusicType item);

/\*\*

\* @brief 파라미터에서 가져온 데이터를 장르리스트에 추가.

\*/

void AddMusictoGenreList(MusicType item);

/\*\*

\* @brief 파라미터에서 가져온 데이터를 송리스트에 추가.

\*/

void AddMusictoSongList(MusicType item);

/\*\*

\* @brief inArtist가 들어가있는 뮤직리스트의 자세한 정보를 출력

\*/

void PrintNFindDetailArtist(Node <ArtistType>\* root, string inArtist, ostream& out);

/\*\*

\* @brief inGenre가 들어가있는 뮤직리스트의 자세한 정보를 출력

\*/

void PrintNFindDetailGenre(Node<GenreType>\* root, string inGenre, ostream& out);

/\*\*

\* @brief inAlbum가 들어가있는 뮤직리스트의 자세한 정보를 출력

\*/

void PrintNFindDetailAlbum(Node <AlbumType>\* root, string inAlbum, ostream& out);

/\*\*

\* @brief inSong이 들어가있는 뮤직리스트의 자세한 정보를 출력

\*/

void PrintNFindDetailSongName(Node <SongType>\* root, string inSong, ostream& out);

/\*\*

\* @brief Display Login command on screen and get a input from keyboard.

\* @pre none.

\* @post none.

\* @return Login command.

\*/

int GetLogin();

/\*\*

\* @brief Login Program driver.

\* @pre Login program is started.

\* @post Login program is finished.

\*/

void Logina();

/\*\*

\* @brief Login Program driver.

\* @pre Login program is started.

\* @post Login program is finished.

\*/

int LoginToApp();

/\*\*

\* @brief Display Master mode command on screen and get a input from keyboard.

\* @pre none.

\* @post none.

\* @return Master mode command.

\*/

int MakeID();

/\*\*

\* @brief Master mode Program driver.

\* @pre Master mode program is started.

\* @post Maaster mode program is finished.

\*/

int GetMaster();

/\*\*

\* @brief Master mode Program driver.

\* @pre Master mode program is started.

\* @post Master mode program is finished.

\*/

void MasterRun();

/\*\*

\* @brief 앨범을 기준으로 음악 정보를 화면에 출력.

\* @pre 리스트가 초기화 되어있어야한다.

\* @post 앨범을 기준으로 음악 정보가 화면에 출력된다.

\*/

void DisplayMusicByAlbum();

/\*\*

\* @brief 노래제목을 기준으로 음악 정보를 화면에 출력.

\* @pre 리스트가 초기화 되어있어야한다.

\* @post 노래제목을 기준으로 음악 정보가 화면에 출력된다.

\*/

void DisplayMusicBySongName();

/\*\*

\* @brief 가수들을 기준으로 음악 정보를 화면에 출력.

\* @pre 리스트가 초기화 되어있어야한다.

\* @post 가수들을 기준으로 음악 정보가 화면에 출력된다.

\*/

void DisplayMusicByArtist();

/\*\*

\* @brief 사용자에게 ID을 입력받아 Delete함수에 넘겨주는 역할을 한다.

\* @pre 없다.

\* @post 사용자가 입력한 노래 제목을 가지고 있는 MusicType이 m\_MasterList에서 사라진다.

\* @return 없다.

\*/

void DeleteMasterMusic();

/\*\*

\* @brief param에서 가져온 item을 가수리스트에 삭제.

\*/

void DeleteAlbumToArtistList(AlbumType& Item);

/\*\*

\* @brief 음악을 Play

\*/

void PlayMusic();

/\*\*

\*@brief 최근 재생한 음악들을 음악제목을 이용해서 보여주는 함수

\*/

int DisplayPlayMusicList();

/\*\*

\*@brief 최근 재생한 음악들을 재생한 시간을 측정해서 시간이 짧은 것부터 보여주는 함수

\*/

int DisplayDeleteList();

protected:

ifstream m\_InFile; ///< input file descriptor.

ofstream m\_OutFile; ///< output file descriptor.

SortedLinkedList<MusicType> m\_MasterList; // < 음악전체 리스트

baseclass m\_MusicList; ///< 나만의 음악전체 리스트

BinarySearchTree<GenreType> m\_GenreList; // < 장르별 리스트

BinarySearchTree<AlbumType> m\_AlbumList; //<앨범별 리스트

BinarySearchTree<ArtistType> m\_ArtistList; // < 아티스트별 리스트

BinarySearchTree<SongType> m\_SongList; // < 노래리스트

SortedLinkedList <Login> m\_MemberList;

CircularQueueType<SongType> PlayRecentList; // < 최근 플레이한 리스트

CircularQueueType<SimpleMusicType> AddRecentlyMusicList; // < 최근 추가한 리스트

SortedLinkedList<PlayTime> DeleteList; // < 삭제할만할 리스트

int m\_Command; ///< current command number.

};

MusicType Class

class MusicType

{

public:

/\*\*

\* default constructor.

\*/

MusicType()

/\*\*

\* destructor.

\*/

~MusicType() {}

/\*\*

\* @brief Get Singer.

\* @pre Singer is set.

\* @post none.

\* @return Singer.

\*/

string GetSinger()

/\*\*

\* @brief Get Birth.

\* @pre Birth is set.

\* @post none.

\* @return Birth.

\*/

string GetBirth() {

/\*\*

\* @brief Get album

\* @pre album is set.

\* @post none.

\* @return album.

\*/

string GetAlbum()

/\*\*

\* @brief Get 회사 이름

\* @pre 회사 이름 is set.

\* @post none.

\* @return 회사 이름을 리턴.

\*/

string GetCompany()

/\*\*

\* @brief Get 노래의 발매 날짜

\* @pre 노래의 발매 날짜 is set.

\* @post none.

\* @return 노래의 발매 날짜를 리턴.

\*/

string GetAnnounce()

/\*\*

\* @brief Get 재생시간.

\* @pre 재생시간 is set.

\* @post none.

\* @return 재생시간을 리턴.

\*/

string GetRuntime()

/\*\*

\* @brief Get Genre

\* @pre Genre is set.

\* @post none.

\* @return Genre.

\*/

string GetGenre()

/\*\*

\* @brief Get 노래 가사

\* @pre 노래 가사 is set.

\* @post none.

\* @return 노래 가사.

\*/

string Gettext()

/\*\*

\* @brief Set Singer.

\* @pre none.

\* @post Singer is set.

\* @param inSinger Singer.

\*/

void SetSinger(string inSinger)

/\*\*

\* @brief Set Birth.

\* @pre none.

\* @post Birth is set.

\* @param inBirth Birth.

\*/

void SetBirth(string inBirth)

/\*\*

\* @brief Set album.

\* @pre none.

\* @post album is set.

\* @param inAlbum album.

\*/

void SetAlbum(string inAlbum)

/\*\*

\* @brief Set Genre.

\* @pre none.

\* @post Genre is set.

\* @param inGenre Genre.

\*/

void SetGenre(string inGenre)

/\*\*

\* @brief Set 노래 가사.

\* @pre none.

\* @post 노래 가사 is set.

\* @param intext text.

\*/

void Settext(string intext)

/\*\*

\* @brief Set Song record.

\* @pre none.

\* @post student Song is set.

\* @param inSN Songname.

\* @param inSinger Singer

\* @param inAlbum Album.

\* @param inGenre Genre.

\* @param intext text(노래가사)

\*/

void SetRecord(string inID, string inSN, string inSinger, string inAlbum, string inGenre, string inBirth, string intext)

/\*\*

\* @brief Display Singer on screen.

\* @pre Singer is set.

\* @post Singer is on screen.

\*/

void DisplaySingerOnScreen()

/\*\*

\* @brief Display Album on screen.

\* @pre Album is set.

\* @post Album is on screen.

\*/

void DisplayAlbumOnScreen()

/\*\*

\* @brief Display Birth on screen.

\* @pre Birth is set.

\* @post Birth is on screen.

\*/

void DisplayBirthOnScreen()

/\*\*

\* @brief Display Genre on screen.

\* @pre Genre is set.

\* @post Genre is on screen.

\*/

void DisplayGenreOnScreen()

/\*\*

\* @brief Display text(노래가사) on screen.

\* @pre text(노래가사) is set.

\* @post text(노래가사) is on screen.

\*/

void DisplaytextOnScreen()

/\*\*

\* @brief Set Singer from keyboard.

\* @pre none.

\* @post Singer is set.

\*/

void SetSingerFromKB();

/\*\*

\* @brief Set Album from keyboard.

\* @pre none.

\* @post Album is set.

\*/

void SetAlbumFromKB();

/\*\*

\* @brief Set Genre from keyboard.

\* @pre none.

\* @post Genre is set.

\*/

void SetGenreFromKB();

/\*\*

\* @brief Set Genre from keyboard.

\* @pre none.

\* @post Genre is set.

\*/

void SettextFromKB();

/\*\*

\* @brief Set Birth from keyboard.

\*/

void SetBirthFromKB();

/\*\*

\* @brief Set song record from keyboard.

\* @pre none.

\* @post song record is set.

\*/

void SetRecordFromKB();

/\*\*

\* @brief Read a record from file.

\* @pre the target file is opened.

\* @post student record is set.

\* @param fin file descriptor.

\* @return return 1 if this function works well, otherwise 0.

\*/

int ReadDataFromFile(ifstream& fin);

/\*\*

\* @brief Write a record into file.

\* @pre the target file is opened. And the list should be initialized.

\* @post the target file is included the new student record.

\* @param fout file descriptor.

\* @return return 1 if this function works well, otherwise 0.

\*/

int WriteDataToFile(ofstream& fout);

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator== (const MusicType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator> (const MusicType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator< (const MusicType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator>= (const MusicType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator<= (const MusicType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

void operator= (const MusicType& obj)

/\*\*

\* @brief Get 노래 제목

\* @pre 노래 제목 is set.

\* @post none.

\* @return 노래 제목.

\*/

string GetSongName()

/\*\*

\* @brief Get ID

\* @pre ID is set.

\* @post none.

\* @return ID.

\*/

string GetID()

/\*\*

\* @brief Set 노래 제목.

\* @pre none.

\* @post 노래 제목 is set.

\*/

void SetSongName(string inSN)

/\*\*

\* @brief Set ID.

\* @pre none.

\* @post ID is set.

\* @param inID ID.

\*/

void SetID(string inID)

/\*\*

\* @brief Set PlayTime.

\* @pre none.

\* @post PlayTime is set.

\* @param inPlaytime PlayTime.

\*/

void SetPlayTime(int inPlaytime)

/\*\*

\* @brief Display 노래 제목 on screen.

\* @pre 노래 제목 is set.

\* @post 노래 제목 is on screen.

\*/

void DisplaySNOnScreen()

/\*\*

\* @brief Display ID on screen.

\* @pre ID is set.

\* @post ID is on screen.

\*/

void DisplayIDOnScreen()

/\*\*

\* @brief Set 노래제목 from keyboard.

\* @pre none.

\* @post 노래제목 is set.

\*/

void SetSNFromKB();

/\*\*

\* @brief Set ID from keyboard.

\* @pre none.

\* @post ID is set.

\*/

void SetIDFromKB();

/\*\*

\* @brief Set Company from keyboard.

\* @pre none.

\* @post Company is set.

\*/

void SetCompanyFromKB();

/\*\*

\* @brief Set Announce from keyboard.

\* @pre none.

\* @post Announce is set.

\*/

void SetAnnounceFromKB();

/\*\*

\* @brief Set Runtime from keyboard.

\* @pre none.

\* @post Runtime is set.

\*/

void SetRunTimeFromKB();

/\*\*

\*@brief operator overloading

\*/

friend ostream& operator << (ostream& out, MusicType& item)

protected:

string m\_ID; // < ID (primary Key)

string m\_SongName; // < 노래제목

string m\_Singer; // < singer

string m\_Genre; // < 장르

string m\_Album; //< album

string m\_text; // < 가사

string m\_Birth; //< 가수의 생년월일

string m\_company; // < 소속사(레코드사)

string m\_announce; // < 발표년 월일

string m\_runtime; // < 재생시간

int m\_PlayTime; // < 유저가 재생한 시간

};

ArtistType Class

class ArtistType : public baseclass

{

private:

string m\_Artist; // < 가수의 이름

string m\_Birth; // < 가수의 생년 월일

string m\_Company; //< 소속사

BinarySearchTree <AlbumType> ArtistAlbumList; // < 가수의 앨범 리스트

public:

/\*\*

\* @brief : constructor

\*/

ArtistType()

/\*\*

\* @brief : desstructor

\*/

~ArtistType() {};

BinarySearchTree<AlbumType>\* GetList() { return &ArtistAlbumList; }

/\*\*

\* @brief : ArtistAlbumList에 앨범타입의 정보를 추가시켜주는 함수.

\* @post : ArtistAlbumList에 정보가 추가된다.

\* @return : 추가가 잘 되었다면 1을 리턴, 추가가 잘되지 않았다면 0을 리턴한다.

\*/

void AddAlbum(AlbumType& data)

int GetArtistAlbumListLength() const

/\*\*

\* @brief : 가수 이름과 가수의 생년월일을 보여주는 함수.

\*/

void DisplayAll()

/\*\*

\* @brief : 가수의 생년 월일을 키보드로 받아주는 함수.

\*/

void setBirthbyKB()

/\*\*

\* @brief : ArtistAlbumList에 앨범타입의 정보를 제거시켜주는 함수.

\* @post : ArtistAlbumList에 정보가 제거된다.

\* @return : 제거가 잘 되었다면 1을 리턴, 제거가 잘되지 않았다면 0을 리턴, 또한 제거했을때 ArtistAlbumList의 길이가 0이면 2를 리턴

\*/

void DeleteAlbum(AlbumType& data)

/\*\*

\* @brief : ArtistAlbumList에 앨범타입의 정보를 갱신시켜주는 함수.

\* @post : ArtistAlbumList에 정보가 갱신된다.

\* @return : 갱신이 잘 되었다면 1을 리턴, 갱신이 잘되지 않았다면 0을 리턴한다.

\*/

void ReplaceAlbum(AlbumType& data)

/\*\*

\* @brief : 가수 이름을 파라미터로 지정해주는 함수.

\*/

void SetArtist(string inArtist)

/\*\*

\* @brief : 가수의 생년월일을 파라미터로 지정해주는 함수.

\*/

void SetBirth(string inBirth)

/\*\*

\* @brief Get 가수 이름

\* @pre 가수 이름 is set.

\* @post none.

\* @return 가수 이름.

\*/

string GetArtist()

/\*\*

\* @brief : 가수의 소속사를 파라미터로 지정해주는 함수.

\*/

void SetCompany(string inCompany)

\* @brief Get 소속사 이름

\* @pre 소속사 이름 is set.

\* @post none.

\* @return 소속사 이름.

\*/

string GetCompany()

/\*\*

\* @brief Get 가수의 생년 월일

\* @pre 가수의 생년 월일 is set.

\* @post none.

\* @return 가수의 생년 월일.

\*/

string GetBirth()

/\*\*

\* @brief 아티스트 타입으로 바꿔주는 함수.

\*/

void SetInfoA(string inArtist,string inCompany ,string inBirth)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator== (const ArtistType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator> (const ArtistType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator< (const ArtistType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator>= (const ArtistType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator<= (const ArtistType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

void operator=(ArtistType& obj)

/\*\*

\* @brief 앨범이름과 추가되어있는 곡수 출력

\*/

friend ostream& operator << (ostream& out, ArtistType& obj)

AlbumType Class

class AlbumType : public baseclass

{

private:

string m\_Albumname; // < 앨범명

string m\_Company; // < 소속사 이름

string m\_Artist; // < 가수 이름

string m\_Birth; // < 가수 생년 월일

int m\_AlbumsongNum; // < 앨범수록곡 수

BinarySearchTree<SimpleMusicType> AlbumMusic; // < 수록곡

public:

/\*\*

\* constructor

\*/

AlbumType()

/\*\*

\* @brief 앨범리스트 안에 있는 앨범 뮤지리스트에 데이터를 추가해준다. 또한 추가할때 마다 수록곡 수를 1씩 증가시킨다.

\* @return 리턴값이 1이면 add가 된 것이고, 0이면 add가 되지 않은 것이다.

\*/

void AddAlbumMusic(SimpleMusicType& data)

/\*\*

\* @brief 앨범리스트 안에 있는 앨범 뮤직리스트에서 데이터를 제거해준다.

\* @return 만약에 앨범뮤직리스트를 제거할 때 앨범리스트의 크기가 0이 되어버린다면 리턴값이 2, 0이 아니라면 리턴값이 1, 삭제가 되지 않는다면 리턴값을 0 으로 만든다.

\*/

void DeleteAlbumMusic(SimpleMusicType& data)

/\*\*

\* @brief 앨범명을 지정해주는 함수

\*/

void SetAlbum(string inAlbum)

BinarySearchTree<SimpleMusicType>\* GetList() { return &AlbumMusic; }

/\*\*

\* @brief 가수의 이름을 지정해주는 함수

\*/

void SetArtist(string inArtist)

/\*\*

\* @brief 앨범리스트 안에 있는 앨범 뮤지리스트에 데이터를 갱신해준다.

\* @return 리턴값이 1이면 data가 갱신된 것이고, 0이면 데이터가 갱신되지 않은 것이다.

\*/

void ReplaceAlbumMusic(SimpleMusicType& data)

/\*\*

\* @brief Set Album.

\* @pre none.

\* @post Album is set.

\* @param inAlbum album.

\*/

string GetAlbum()

/\*\*

\* @brief Set Birth.

\* @pre none.

\* @post Birth is set.

\* @param inBirth birth.

\*/

void SetBirth(string inBirth)

string GetBirth()

/\*\*

\* @brief Set Company.

\* @pre none.

\* @post Company is set.

\* @param inCompany company.

\*/

void SetCompany(string inCompany)

string GetCompany()

/\*\*

\* @brief Get Artist

\* @pre Artist is set.

\* @post none.

\* @return Artist.

\*/

string GetArtist()

/\*\*

\* @brief Get 앨범의 수록곡 수

\* @return m\_AlbumSongNumber.

\*/

int GetAlbumSongNumber()

/\*\*

\* @brief 앨범타입으로 지정해주는 함수이다.

\*/

void SetInfo(string inAlbum, string inArtist, string inBirth, string inCompany)

/\*\*

\* @brief 앨범타입 클래스에 있는 변수들을 출력해주는 함수이다.

\*/

void DisplayAll()

/\*\*

\* destructor

\*/

~AlbumType() {};

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator== (const AlbumType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator> (const AlbumType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator< (const AlbumType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator>= (const AlbumType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator<= (const AlbumType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

void operator=(AlbumType& obj)

int GetAlbumSong()

/\*\*

\* @brief 앨범이름과 추가되어있는 곡수 출력

\*/

friend ostream& operator << (ostream& out, AlbumType& obj)

GenreType Class

class GenreType : public baseclass

{

private:

BinarySearchTree<SimpleMusicType> m\_GenreMusicList; // < 장르뮤직리스트

string m\_Genre; //< 장르

public:

/\*\*

\* constructor.

\*/

GenreType()

/\*\*

\* destructor.

\*/

~GenreType() {}

BinarySearchTree<SimpleMusicType>\* GetList() { return &m\_GenreMusicList; }

/\*\*

\* @brief 장르 이름을 설정하는 함수

\*/

void setGenreName(string inGenre)

/\*\*

\* @brief Get m\_Genre.

\* @return m\_Genre.

\*/

string getGenreName()

{

return m\_Genre;

}

/\*\*

\* @brief 장르타입으로 세팅해주는 함수.

\*

\*/

void setInfo(std::string inGenre)

{

m\_Genre = inGenre;

}

/\*\*

\* @brief 장르MusicList에 SimpleMusicType을 넣어줍니다

\* @return 넣었으면 1 넣지 못했으면 0을 반환.

\*/

void addGenre(SimpleMusicType& item)

/\*\*

\* @brief 장르MusicList에 데이터를 제거하는 함수.

\* @return 제거가 되는데 그 뮤직리스트의 크기가 0이면 2를 리턴, 크기가 0이 아니면 1을 리턴, 제거가 안되면 0을 리턴.

\*/

void deleteGenre(SimpleMusicType& item)

/\*\*

\* @brief 장르MusicList에 데이터를 바꿔주는 함수.

\* @return 제대로 바꾸었으면 1 바꾸지 못했으면 0을 반환.

\*/

void replaceGenre(SimpleMusicType& item)

/\*\*

int GetMusicLength() const

/\*\*

\* @brief 장르 아이디와 장르를 간단히 보여주는 함수,

\*/

void GenreType::DisplayAll()

\* @brief 오버로딩 연산자

\*/

void operator=(GenreType& obj)

/\*\*

\* @brief 오버로딩 연산자

\*

\*/

bool operator== (const GenreType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*

\*/

bool operator> (const GenreType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*

\*/

bool operator< (const GenreType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*

\*/

bool operator>= (const GenreType &obj)

/\*\*

\* @brief 오버로딩 연산자

\*

\*/

bool operator<= (const GenreType &obj)

/\*\*

\* @brief 오버로딩 << 연산자

\*/

friend ostream& operator << (ostream& out, GenreType& obj)

};

#endif //\_GENRETYPE\_H

SongType Class

class SongType : public baseclass

{

protected:

string m\_SongName; // < 노래제목

string m\_ID;

public:

/\*

\* constructor

\*/

SongType() {

/\*

\* destrutor

\*/

~SongType() {}

/\*\*

\* @brief Get 노래 제목

\* @pre 노래 제목 is set.

\* @post none.

\* @return 노래 제목을 리턴.

\*/

string GetSongName()

/\*\*

\* @brief Get ID

\*/

string GetID()

/\*\*

\* @brief Set ID

\*/

void SetID(string inID)

/\*\*

\* @brief Set Song Name.

\* @pre none.

\* @post SongName is set.

\* @param inSongName m\_SongName.

\*/

void SetSongName(string inSongName)

/\*\*

\* @brief Display 노래 제목 on screen.

\* @pre 노래 제목 is set.

\* @post 노래 제목 is on screen.

\*/

void DisplaySongName()

/\*\*

\* @brief Set 노래 제목 from keyboard.

\* @pre none.

\* @post 노래 제목 is set.

\*/

void SetSNFromKB()

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator== (const SongType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator> (const SongType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator< (const SongType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator>= (const SongType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator<= (const SongType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

void operator= (const SongType& obj)

/\*\*

\* @brief 연산자 << 오버로딩

\*/

friend ostream& operator << (ostream& out, SongType& obj)

/\*\*

\* @brief 송타입으로 세팅해주는 함수.

\*/

void setinfo(string inId, string inSN) {

Login Class

class Login

{

private :

string name; // < 회원 이름

string password; // < 회원 비밀번호

string ID; // < 회원 ID

string mail; //< 회원 이메일

public:

/\*\*

\* constructor.

\*/

Login()

/\*\*

\* destructor.

\*/

~Login() {};

/\*\*

\* @brief Get ID.

\* @pre ID is set.

\* @post none.

\* @return ID.

\*/

string GetID()

/\*\*

\* @brief Set ID.

\* @pre none.

\* @post ID is set.

\* @param inID ID.

\*/

void SetID(string inID)

/\*\*

\* @brief Set Password.

\* @pre none.

\* @post Password is set.

\* @param inPW password.

\*/

void SetPassWord(string inPW)

/\*\*

\* @brief Get PassWord.

\* @pre PassWord is set.

\* @post none.

\* @return Password.

\*/

string GetPassWord()

/\*\*

\* @brief Get Name.

\* @pre Name is set.

\* @post none.

\* @return Name.

\*/

string GetName()

/\*\*

\* @brief Get mail.

\* @pre mail is set.

\* @post none.

\* @return mail.

\*/

string GetMail()

/\*\*

\* @brief 회원가입

\*/

void MakeID(string inId, string inpassword, string inName, string inMail)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator== (const Login &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator> (const Login &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator< (const Login &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator>= (const Login &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator<= (const Login &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

void operator= (const Login& obj)

};

PlayTime Class

#pragma once

{

private :

double playtime; // < 유저가 음악을 듣는 시간

string Songname; // < 음악이름

public:

/\*\*

\* @brief Set PlayTime

\* @pre none.

\* @post Playtime is set.

\* @param inPlaytime playtime.

\*/

void SetPlayTime(int inplaytime)

/\*\*

\* @brief Get PlayTime

\* @pre Playtime is set.

\* @post none.

\* @return PlayTime을 리턴.

\*/

double GetPlayTime()

/\*\*

\* @brief Set Song Name.

\* @pre none.

\* @post SongName is set.

\* @param inSongName SongName.

\*/

void SetSongName(string inSongName)

/\*\*

\* @brief Get 노래 제목

\* @pre 노래 제목 is set.

\* @post none.

\* @return 노래 제목을 리턴.

\*/

string GetSongName()

/\*\*

\* @brief 노래제목과 재생한시간을 보여주는 함수

\*/

void Display()

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator== (const PlayTime &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator> (const PlayTime &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator< (const PlayTime &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator>= (const PlayTime &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

bool operator<= (const PlayTime &obj)

/\*\*

\* @brief 오버로딩 연산자

\*/

void operator= (const PlayTime& obj)

Baseclass Class

class baseclass {

protected:

static BinarySearchTree <MusicType> mlist;

public:

/\*\*

\* @brief 현재 리스트의 갯수 return

\*/

int GetLength()

/\*\*

\* @brief 현재 리스트의 추가

\*/

void Add(MusicType& data)

/\*\*

\* @brief 현재 리스트를 get

\*/

int Get(MusicType& data)

/\*\*

\* @brief 현재 리스트를 수정

\*/

void Replace(MusicType& data)

/\*\*

\* @brief 현재 리스트의 삭제

\*/

void DeleteItem(MusicType data)

/\*\*

\* @brief Tree의 node를 스크린에 출력한다.

\*/

void PrintTree(ostream &out)const

\* @brief mlist의 RetrieveItem을 return.

\*/

void RetrieveItem(MusicType& item, bool &found)

/\*\*

\* @brief 자식클래스에서 마스터리스트의 정보를 가져오기 위한 함수

\*/

void Print(Node <SimpleMusicType>\* root, ostream& out)

SimpleMusic Type Class

class SimpleMusicType

{

public:

/\*\*

\* constructor.

\*/

SimpleMusicType()

/\*\*

\* destructor.

\*/

~SimpleMusicType() {}

/\*\*

\* @brief Get Song name

\* @pre Song name is set

\* @post none.

\* @return Song name.

\*/

string GetSongName()

/\*\*

\* @brief Get Song ID

\* @pre Song ID is set

\* @post none.

\* @return Song ID.

\*/

string GetID()

/\*\*

\* @brief Set song name.

\* @pre none.

\* @post Song name is set.

\* @param inSN song name.

\*/

void SetSongName(string inSN)

/\*\*

\* @brief Set song ID.

\* @pre none.

\* @post Song ID is set.

\* @param inID song ID.

\*/

void SetID(string inID)

{

m\_ID = inID;

}

/\*\*

\* @brief Display Song name on screen.

\* @pre Song name is set.

\* @post Song name is on screen.

\*/

void DisplaySNOnScreen()

/\*\*

\* @brief Display Song ID on screen.

\* @pre Song ID is set.

\* @post Song ID is on screen.

\*/

void DisplayIDOnScreen()

/\*\*

\* @brief Display Song name&ID on screen.

\* @pre Song name&ID are set.

\* @post Song name&ID are on screen.

\*/

void DisplaySimpleOnScreen()

\* @brief Set song name from keyboard.

\* @pre none.

\* @post Song name is set.

\*/

void SetSNFromKB();

/\*\*

\* @brief Set song ID from keyboard.

\* @pre none.

\* @post Song ID is set.

\*/

void SetIDFromKB();

/\*\*

\* @brief Set song ID&song name from keyboard.

\* @pre none.

\* @post Song ID& song name are set.

\*/

void SetSimpleFromKB();

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator== (const SimpleMusicType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator> (const SimpleMusicType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator< (const SimpleMusicType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator>= (const SimpleMusicType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

bool operator<= (const SimpleMusicType &obj)

/\*\*

\* @brief 연산자 오버로딩

\*/

void operator=(const SimpleMusicType& obj)

/\*\*

\*@brief 심플뮤직타입으로 타입을 세팅해주는 함수.

\*/

void setinfo(string inId, string inSN) {

protected:

string m\_ID; // < ID(primary Key)

string m\_SongName; // < song name

};