[프로그램 명세서]

PROJECT : 음원 관리 프로그램

학과 : 응용수학과

학번 : 2017103081

이름 : 박승혜

1. Database

class Database

{

private:

SortedList<SingerType> mSingerList; - 마스터모드의 가수 목록

SortedList<SongType> mSongList; - 마스터모드의 곡 목록

SortedList<AlbumType> mAlbumList; - 마스터모드의 앨범 목록

UserType mUser; - 유저모드를 위한 data

SortedList<PlayItemType> mChartList; - 유저모드에 띄워지는 전체 곡 차트

public:

Database();

~Database();

PlayItemType mPlaying; - 현재 재생되고 있는 파일을 나타낸다.

int GetSingerLength();

int AddSinger(SingerType inSinger);

SingerType GetSinger(int idx);

int GetSinger(SingerType &inSinger);

int DeleteSinger(SingerType inSinger);

int DeleteSinger(int idx);

int ReplaceSinger(SingerType inSinger);

void ResetSingerPointer();

int GetNextSinger(SingerType &inSinger);

int GetAlbumLength(SingerType inSinger);

int GetSongLength(SingerType inSinger);

int GetAlbumLength();

int AddAlbum(AlbumType inAlbum);

AlbumType GetAlbum(int idx);

int GetAlbum(AlbumType &inAlbum);

int DeleteAlbum(AlbumType inAlbum);

int DeleteAlbum(int idx);

int ReplaceAlbum(AlbumType inAlbum);

void ResetAlbumPointer();

int GetNextAlbum(AlbumType &inAlbum);

int GetSongLength(AlbumType inAlbum);

int GetSongLength();

int AddSong(SongType inSong);

SongType GetSong(int idx);

int GetSong(SongType &inSong);

int DeleteSong(SongType inSong);

int DeleteSong(int idx);

int ReplaceSong(SongType inSong);

void ResetSongPointer();

int GetNextSong(SongType &inSong);

void SetUser(UserType inUser);

UserType GetUser();

void SetUserId(CString inId);

void SetUserPw(CString inPw);

CString GetUserId();

CString GetUserPw();

void AddToPL(PlayItemType item);

void UpdateChart(int num = 5);

int GetChartLength();

int AddChart(PlayItemType inSong);

int DeleteChart(PlayItemType inSong);

int ReplaceChart(PlayItemType inSong);

void ResetChartPointer();

int GetNextChart(PlayItemType &inSong);

PlayItemType GetChartItem(int idx);

bool SongInLlist(CString inSong);

bool AlbumInLlist(CString inAlbum);

bool SingerInLlist(CString inSinger);

void AddSongtoL(CString inSong);

void AddAlbumtoL(CString inAlbum);

void AddSingertoL(CString inSinger);

void DelSongInL(CString inSong);

void DelAlbumInL(CString inAlbum);

void DelSingerInL(CString inSinger);

void UpdatePL(int num = 3);

int GetPLLength();

int DeletePL(PlayItemType inSong);

int ReplacePL(PlayItemType inSong);

void ResetPLPointer();

int GetNextPL(PlayItemType &inSong);

PlayItemType GetPLItem(int idx);

void GetBeforePLItem();

void GetNextPLItem();

};

2. UserType

class UserType

{

private:

CString u\_Id; -아이디

CString u\_Pw; -비밀번호

SortedList<PlayItemType> PlayList; -플레이리스트

UnSortedList<CString> LSongList; -좋아요를 누른 곡 리스트

UnSortedList<CString> LAlbumList; -좋아요를 누른 앨범 리스트

UnSortedList<CString> LSingerList; -좋아요를 누른 가수 리스트

public:

UserType();

~UserType();

CString getId();

CString getPw();

void setId(CString inId);

void setPw(CString inPw);

void AddToPL(PlayItemType item);

int getPLLength();

int DeletePL(PlayItemType inSong);

int ReplacePL(PlayItemType inSong);

void ResetPLPointer();

int GetNextPL(PlayItemType &inSong);

int GetPLItem(PlayItemType inSong);

PlayItemType GetPLItem(int idx);

void UserType::UpdatePL(int num);

bool SongInLlist(CString inSong);

bool AlbumInLlist(CString inAlbum);

bool SingerInLlist(CString inSinger);

void AddSongtoL(CString inSong);

void AddAlbumtoL(CString inAlbum);

void AddSingertoL(CString inSinger);

void DelSongInL(CString inSong);

void DelAlbumInL(CString inAlbum);

void DelSingerInL(CString inSinger);

bool operator == (const UserType &item);

};

3. ItemType들(song, album, singer) / 기본 접근 함수들 생략

class SongType

{

private:

CString s\_Name;

CString s\_Id;

CString s\_Singer;

CString s\_Album;

int s\_NumLikes;

class AlbumType

{

private:

CString s\_Name;

CString s\_Comeout;

CString s\_Type;

CString s\_Genre;

CString s\_Singer;

int s\_NumLikes;

class SingerType

{

private:

CString s\_Name;

CString s\_DebutYear;

CString s\_Type;

CString s\_Genre;

int s\_NumLikes;

4. PlayItemType -> 리스트를 원하는대로 정렬할 수 있는 타입

class PlayItemType

{

private:

CString m\_Id; 각각의 정렬기준에 해당하는 정보들을 갖고,

CString m\_Song;

CString m\_Singer;

CString m\_Album;

int m\_UsersNum;

int m\_InTime;

int m\_NumLikes;

CString m\_priority; priority라는 정렬시킬 정보를 하나 만들어 주었다.

public:

PlayItemType();

~PlayItemType();

CString getId();

CString getSong();

CString getSinger();

CString getAlbum();

int getUsersNum();

int getInTime();

int getNumLikes();

CString getPriority();

void setId(CString inId);

void setSong(CString inSong);

void setSinger(CString inSinger);

void setAlbum(CString inAlbum);

void setUsersNum(int inNum);

void setInTime();

void setNumLikes(int inNum);

void PlayItemType::setPriority(int num) { setPriority함수를 통해 정렬하고자 하는

switch (num)

{ 기준을 정수로 받으면 switch구문을

case 0:

m\_priority = m\_Id; 통해 정렬 기준 값을 각각의 변수로

break;

case 1: 변경해주었다.

m\_priority = m\_Song;

break;

case 2:

m\_priority = m\_Singer;

m\_priority += m\_Song;

break;

case 3:

m\_priority.Format(\_T("%d"), m\_UsersNum);

break;

case 4:

m\_priority.Format(\_T("%d"), m\_InTime);

break;

case 5:

m\_priority.Format(\_T("%d"),900000 - m\_NumLikes);

break;

default:

break;

}

}

void operator = (PlayItemType &item); 이후 비교연산자에서 priority를

bool operator == (const PlayItemType &item); 기준으로 비교하면 원하는 기준으로

bool operator < (const PlayItemType &item); 정렬이 가능하다.

bool operator > (const PlayItemType &item);

void operator = (SongType &item);

};