하위 설계서

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 1. 목적 |  * 고급객체지향 프로그래밍의 실습 과제이다. 클래스, 상속, 자료구조, 패턴 등을 사용한 프로그래밍을 통하여 실력을 향상시킨다. * 실전 프로그래밍에서의 예외 처리, 다양한 선택변수 등을 접하고 대비하는 능력을 향상시킨다.  |  | | --- | | 1. 사용자 Targeting |  * 디지털 음반 유통 프로그램을 이용하고자 하는 사용자 및 관리자를 대상으로 하며, 직관적인 인터페이스로 누구나 쉽게 디지털 음반 유통 프로그램을 이용할 수 있어야 한다.  |  | | --- | | 1. 설계 원칙 |   - User클래스는 유저의 목록(정수형 아이디, 문자열 비밀번호, 구매한 앨범의 아이디)을 관리한다.  - Composer 클래스는 음악가의 목록(아이디, 음악가의 이름)을 관리한다.  - MusicAlbum 클래스는 앨범 목록(앨범의 아이디, 앨범의 이름, 앨범의 음악가의 아이디, 앨범의 구매 횟수)을 관리한다. 여기서 앨범이란 일반적으로 알려진 하나의 음악을 의미한다.  - BaseManager는 파일 입출력을 담당하는 클래스이다. 문자열 fileName을 가지고 있으며, 자식클래스들은 이 fileName에 따라 해당되는 파일의 입출력을 수행한다. BaseManager 클래스를 상속하여 UserManager, ComposerManager AlbumManager 클래스들이 각각의 파일에 접근한다.  - UserManager는 사용자의 정보 관리 및 사용자 정보가 저장된 파일의 파일입출력을 담당한다.  - ComposerManager는 음악가의 정보가 저장된 파일의 파일입출력을 담당한다.  - AlbumManager는 앨범의 정보가 저장된 파일의 파일입출력을 담당한다.  - BaseRecoAlg 클래스는 사용자에게 앨범(음악)을 추천하는 클래스이다. 순수가상함수 recommendation( )을 가지고 있으며, 이 클래스를 상속받아 다양한 규칙의 추천 클래스를 만들어 추천을 수행할 수 있다. 즉, BaseRecoAlg 클래스는 전략패턴으로 설계된다. 기본적으로 사용자에게 구매 횟수가 높은 10개의 음악을 추천하는 알고리즘을 반드시 구현해야 한다. 이 추천 방식은 BaseRecoAlg 클래스를 상속받은 자식 클래스 RecommendTopTen 클래스에서 사용자에게 구매 횟수가 탑 10개의 추천을 recommend를 오버라이딩하여 구현한다.  - 메인에서는 오직 App의 run()만을 호출한다.   |  | | --- | | 1. 개발 도구 |   Microsoft Visual Studio C++ 2013   |  | | --- | | 1. 적용 범위 |  * 장소적 범위 : 디지털 음반 유통 프로그램이 사용될 수 있는 장소 * 프로그램적 범위 : 디지털 음반 유통 홈페이지 등  |  | | --- | | 1. 시나리오 흐름도 |   시나리오 흐름도     |  | | --- | | 1. 인터페이스 정의 |  * 문장은 사용자가 해당 기능에 대한 사용법을 명확히 알 수 있도록 명시한다. * 글자가 지저분하지 않도록 띄어쓰기와 줄 바꿈을 해준다. * 입력은 키보드로만 받는다. * 한국어와 영어 외의 언어는 사용하지 않는다.  |  | | --- | | 1. 헤더와 cpp, 클래스정의 |  * 클래스 선언은 헤더파일을 만들고 메인 cpp에 불러온다. * 헤더에서 선언한 클래스는 헤더와 같은 이름의 cpp에 구현한다.  |  |  | | --- | --- | | **헤더와 cpp** | | | 기능 | 이름 | | 클래스 선언 | AlbumManager.h  App.h  BaseManager.h  BaseRecAlg.h  Composer.h  ComposerManager.h  MusicAlbum.h  RecTopTen.h  User.h  UserManager.h | | 클래스 구현 | AlbumManager.cpp  App.cpp  BaseManager.cpp  BaseRecAlg.cpp  Composer.cpp  ComposerManager.cpp  MusicAlbum.cpp  RecTopTen.cpp  User.cpp  UserManager.cpp | | 메인 구현 | Main.cpp |  |  |  | | --- | --- | | **클래스** | | | 클래스 이름 | 기능 | | AlbumManager.h 앨범의 정보가 저장된 파일의 파일입출력을 담당  App.h 기능 총괄  BaseManager.h 파일 입출력을 담당  BaseRecAlg.h 사용자에게 앨범(음악)을 추천  Composer.h 음악가의 목록(아이디, 음악가의 이름)을 관리  ComposerManager.h 음악가의 정보가 저장된 파일의 파일입출력을 담당  MusicAlbum.h 앨범 목록(앨범의 아이디, 앨범의 이름, 앨범의 음악가의 아이디, 앨범의 구매 횟수)을 관리  RecTopTen.h 사용자에게 구매 횟수가 높은 10개의 음악을 추천하는 알고리즘  User.h 유저의 목록(정수형 아이디, 문자열 비밀번호, 구매한 앨범의 아이디)을 관리  UserManager.h 사용자의 정보 관리 및 사용자 정보가 저장된 파일의 파일입출력을 담당 | |   ▣ 클래스 별 요소와 기능 ▣   |  | | --- | | **App.h** | | #pragma once  //#include "BaseRecAlg.h"  #include "RecTopTen.h"  class App  {  public:  App();  ~App();  void run();    private:  void printUserMenu(); // 사용자 메뉴 출력  void printManagerMenu(); // 관리자 메뉴 출력  void userRegisteration(); // 유저 등록  void login(); // 로그인  void buyAlbum(); // 앨범 구매  void printPurchasedList(); // 구매 목록 출력  void recommend(); // 추천  void addComposer(); // 음악가 등록  void addAlbum();// 앨범 등록  private:  BaseRecAlg\* rcdAlg; // 추천 알고리즘 클래스 포인터  UserManager\* um; // 사용자 매니저 포인터  ComposerManager\* cm; // 음악가 매니저 포인터  AlbumManager\* am; // 앨범 매니저 포인터  int currentUserId; // 현재 로그인 된 사용자 아이디  }; |  |  |  | | --- | --- | | **AlbumManager.h** | | | #pragma once  #include "MusicAlbum.h"  #include "BaseManager.h"  class AlbumManager :  public BaseManager  {  public:  AlbumManager();  ~AlbumManager();  public:  bool addMusicAlbumToFile(MusicAlbum addedMusicAlbum); // 새로운 앨범의 정보를 파일에 저장  MusicAlbum searchMusicAlbumByName(string \_name); // 주어진 검색어와 일치하는 이름을 가진 앨범을 반환  MusicAlbum searchMusicAlbumById(int \_id); // 주어진 아이디와 일치하는 아이디를 가진 앨범을 반환  bool updateMusicAlbumSellCount(int albumId); // 앨범의 아이디를 받아 앨범의 구매 횟수를 갱신 및 파일에 재작성  vector<MusicAlbum> getAllAlbumList(); // 모든 앨범의 정보를 읽어 목록을 반환  }; |  |  | | --- | | **UserManager.h** | | #pragma once  #include "User.h"  #include "BaseManager.h"  #include <fstream>  class UserManager :  public BaseManager  {  public:  UserManager();  ~UserManager();  public:  bool addUserToFile(User addedUser); // 새로운 사용자의 정보를 파일에 저장  bool authenticateUser(int \_id, string \_pw); // 사용자가 로그인 시 아이디와 비밀번호가 맞는지 확인  vector<int> getPurchasedListByUserId(int \_id); // 아이디를 통해 구매한 앨범 목록을 반환  vector<User> getAllUserList(); // 모든 사용자의 목록을 반환  bool addAlbumIdToPurchaseList(int \_userId, int \_albumId); // 주어진 아이디의 사용자의 구매목록에 앨범 아이디를 추가하고\_파일을 재작성  private:  vector<User> userList;  }; |  |  | | --- | | **ComposerManager.h** | | #pragma once  #include "Composer.h"  #include "BaseManager.h"  #include <fstream>  class ComposerManager :  public BaseManager  {  public:  ComposerManager();  ~ComposerManager();  public:  bool addComposerToFile(Composer addedComposer); // 새로운 음악가의 정보를 파일에 저장  Composer searchComposerByName(string \_name); // 주어진 검색어와 일치하는 이름을 가진 음악가를 반환  Composer searchComposerById(int \_id); // 주어진 아이디와 일치하는 아이디를 가진 음악가를 반환  vector<Composer> getAllComposerList(); // 모든 음악가 정보를 읽어 목록을 반환  }; |  |  | | --- | | **BaseManager.h** | | #pragma once  #include <fstream>  #include <string>  #include <vector>  #include <iostream>  using namespace std;  class BaseManager  {  public:  BaseManager();  ~BaseManager();  void setFileName(string \_name);  private:  string fileName;  }; | |  |  |  | | --- | | 1. Main의 변수와 cpp정의 |   App : App 클래스 선언   |  | | --- | | 1. 프로젝트 추진 일정 |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | **추진일정** | | | | | | | | | | | | **1주** | | **2주** | | **3주** | | **4주** | | **5주** | | | | **요구사항 분석** |  |  |  |  |  |  |  |  |  |  |  | | **헤더,  cpp작성** |  |  |  |  |  |  |  |  |  |  |  | | **main작성** |  |  |  |  |  |  |  |  |  |  |  | | **1차 완성** |  |  |  |  |  |  |  |  |  |  |  | | **옵션 추가** |  |  |  |  |  |  |  |  |  |  |  | | **버그 수정** |  |  |  |  |  |  |  |  |  |  |  | | **검토 및  제출** |  |  |  |  |  |  |  |  |  |  |  | |