프로젝트 최종 보고서

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. 프로그램 개요  * **디지털 음반 유통 프로그램** * 사용자 기능 * (회원 가입, 로그인, 앨범 구매, 구매목록 출력, 음악 추천) * 관리자 기능 * (음악가 등록, 앨범 등록, 등록된 데이터 출력)   .  2. 전체 논리 구조도 및 모듈 별 기능   * 1. App      1. 기능 총괄, 입출력은 구현하지 않음      2. 메인에서 오직 run() 함수 만을 호출   2. User      1. 유저의 목록 관리   3. Composer      1. 음악가 목록 관리   4. MusicAlbum      1. 앨범 목록 관리   5. BaseManager      1. 파일 입출력 관리      2. 문자열 fileName을 가지고 있으며, 자식클래스들은 이 fileName에 따라 해당되는 파일의 입출력을 수행      3. BaseManager 클래스를 상속하여 UserManager, ComposerManager AlbumManager 클래스들이 각각의 파일에 접근   6. BaseRecommendationAlg      1. 사용자에게 앨범(음악)을 추천하는 클래스      2. 순수가상함수 recommendation( )을 가지고 있으며, 이 클래스를 상속받아 다양한 규칙의 추천 클래스를 만들어 추천을 수행      3. 전략패턴으로 설계      4. 기본적으로 사용자에게 구매 횟수가 높은 10개의 음악을 추천하는 알고리즘을 반드시 구현   7. UserManage      1. 사용자의 정보 관리 및 사용자 정보가 저장된 파일의 파일입출력을 담당.   8. ComposerManager      1. 음악가의 정보가 저장된 파일의 파일입출력을 담당.   9. AlbumManager      1. 앨범의 정보가 저장된 파일의 파일입출력을 담당.  1. 프로세스 처리도 & 시나리오 흐름도     5. 모듈의 함수     |  | | --- | | **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;  }; |   6. 설계 원칙  설계 요구 조건에 따라 충실히 설계한다.  - 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.  7. 개발도구  Microsoft Visual Studio 2013 C++  8. 사용 매뉴얼  **데이터 파일**  **로그인**  **아이디로 앨범 구입**  **구매 내역**  **데이터 추가**  **앨범 추천**  **이름으로 앨범 구입**  **정보 출력**   * 데이터 파일       MusicAlbum, Composer, User 데이터는 모두 바이너리 파일  따라서 데이터에 맞는 파일 입출력 적용이 필요  User 데이터는 vector에 담아 컨테이너에 저장  MusicAlbum과 Composer는 바이너리로 하나씩 읽어온 데이터를 이용해 비교  User, Composer, MusicAlbum 각 데이터의 정보 저장   * 로그인     사용자는 프로그램 이용 시 로그인 해야 함    회원 가입에서 아이디는 UserList의 데이터를 이용해 자동으로 할당해 계정 생성 가능    올바른 아이디와 비밀번호를 입력하면 현재 로그인 된 유저로 지정하고 사용자 메뉴 출력   * 아이디로 앨범 구입     아이디를 입력 받아 곡의 정보를 출력한 뒤 구매 여부를 물음   * 이름으로 앨범 구입     앨범 이름을 통해서도 구매 가능   * 구매 내역     구매 내역 확인에서는 현재까지 유저가 구매한 모든 음악을 확인 가능   * 음악 추천     음악 추천 알고리즘은 전략 패턴을 이용  App의 BaseRecAlg 포인터를이용해 사용자의 필요에 따라 적용    구매 횟수가 가장 높은 순서대로 추천하는 RecTopTen 알고리즘    최근 구매한 앨범의 장르와 유사한 앨범을 추천하는 RecMyAlg 알고리즘  이때 사용자가 구매하지 않은 앨범만 추천하도록 함   * 데이터 추가       관리자 모드에서 음악가와 앨범을 추가 가능   * 정보 출력     관리자는 사용자, 앨범, 음악가의 정보를 확인 가능 |