도서 관리 프로그램 – 11/20발표

<전반적인 기능>

도서 관리 (조회, 삽입, 삭제 등)

도서 대여, 회원 등록, 로그인 (회원 구분, 학생, 대학원생, 교수 등)

도서 검색 (카테고리, 제목, 저자명)

검색하면 책 정보 간략하게 뜨기 (저자, 출판사, 장르, 페이지 수 등), 책 검색하고 선택하면 리뷰 보이게

연체상태 (회원별로 대여 가능시간 다르게)

읽었던 책 별점, 리뷰 등록 등

대여했던 도서 기록

책에 고유번호를 붙여서 검색 쉽도록

**<연체상태>**

대여 가능 기간 (학생 14일, 대학원생 21일, 교수 28일)

로그인 -> 메뉴에서 연체확인 선택 -> 대여한 날짜에서 대여가능기간을 더했을 때, 오늘 날짜를 지나쳤다면 연체 = TRUE, 아니라면 FALSE -> 책과 연체상태 출력

**Ex)** student인 name이 책의 연체여부를 판단하려고 할 때

->로그인

->메뉴에서 연체확인 선택

->대여가능기간과 함께 상태 출력

**사용된 함수**

checkOverdue() : 책이 연체되었는지 여부를 확인.

현재 날짜와 책의 대여 가능기간을 비교하여 연체 여부를 판단.

main함수에서 choice == 1일 때 호출된다.

**IOPE**

I : choice(사용자가 메뉴에서 선택한 옵션을 나타냄)

O : book.checkOverdue()(연체상태 확인), daylimit(대여 가능기한)

P : choice == 1 then

dueDate = ltm

dueDate.tm\_mDay += (getDaysLimit() – 1)

return diffrime(mktime(&dueDate), now) < 0;

E : 대여 가능기간 = 14, 오늘 날짜 = 11/10, 빌린 날짜 = 11/6

11/6 + 14 = 11/20

11/20 > 11/10 -> 연체 X

**시퀀스 다이어그램**

Book

Member

User

User

Main()

Check Login

연체상태 확인

checkOverdue()

OutPut

Result

**<읽었던 책 별점, 리뷰 등록>**

로그인 -> 메뉴에서 리뷰등록 선택 -> 대여 후 반납했던 책 기록을 가져옴 -> 책 선택 -> 별점과 리뷰작성 -> 객체배열에 저장

**Ex)** student인 name이 반납한 책의 리뷰를 등록하려고 할 때

->로그인

->메뉴에서 리뷰등록 선택

->반납한 책의 별점과 리뷰를 차례로 작성

->등록 완료

**사용된 함수**

addReview() : 사용자가 리뷰를 등록하고자 할 때 호출

책의 대여상태를 확인하고 책 정보 출력 후 별점과 리뷰 차례로 입력받음

**IOPE**

I : int choice(사용자가 메뉴에서 선택한 옵션을 나타냄)

O : float starPoint(사용자가 입력한 별점), string review(사용자가 입력한 리뷰)

P : choice == 2 then

print(책정보)

입력받기(starPoint)

입력받기(review)

addReview(starPoint, review) //리뷰 등록

E : starPoint = 4.5

Review = “좋아요”

addReview(starPoint, review)

**시퀀스 다이어그램**

Book

Member

User

User

addReview()

BookInfo()

setStarPoint()

Enter starPoint

Enter review

setReview()

Registered review

**사용된 객체지향의 특성**

추상화 - 클래스(도서정보, 회원정보 추상화)

상속 - 다양한 종류의 도서 모델링 (사용자 -> 교직원,학부생)

동적 바인딩 (가상함수) - 가상함수 사용하여 사용자 유형에 따라 메시지 출력

연산자 중복 – 두개의 도서를 병합할 때 (여러 권의 책 대여할 때)

예외처리 – 이미 대출중인 책이거나 대출 불가능 상태일 때 예외처리

포괄적 함수 (generic function) – 도서목록 처리

추상화 : 클래스 Book과 Member가 각각 책과 멤버를 추상화한다

동적 바인딩 (가상함수) : NotReturned()클래스의 what()함수가 가상함수로 선언된다. 예외가 발생했을 때 어떤 오류메세지를 반환할지 정한다.

예외처리 : addReview함수에서 예외처리가 이루어진다. NotReturned예외가 발생할 경우 오류메세지를 출력한다.

<전체 코드>

|  |
| --- |
| #include <iostream>  #include <vector>  #include <ctime>  #include <iomanip>  #include <string>  using namespace std;  class Book {  private:  string bookname; //책이름  string writer; //저자  string publisher; //출판사  string genre; //분야  float numbering; //고유번호  float star\_point; //별점\*  string review; //리뷰\*  int page; //페이지 수  bool rent; //대여 여부\*  int daylimit; //대여 기한\*  public:  Book(const string& bookName, const string& writer, const string& publisher, const string& genre, float numbering, float starPoint, const string& review, int page, bool rent)  : bookname(bookName), writer(writer), publisher(publisher), genre(genre), numbering(numbering), star\_point(starPoint), review(review), page(page), rent(rent) {  };  string getBookName() const { return bookname; }  string getWriter() const { return writer; }  string getPublisher() const { return publisher; }  string getGenre() const { return genre; }  float getNumbering() const { return numbering; }  float getStarPoint() const { return star\_point; }  string getReview() const { return review; }  int getPage() const { return page; }  int getDaysLimit() const { return daylimit; }  bool isRented() const { return rent; }  void setBookName(const string& bookName) { bookname = bookName; }  void setWriter(const string& writer) { this->writer = writer; }  void setPublisher(const string& publisher) { this->publisher = publisher; }  void setGenre(const string& genre) { this->genre = genre; }  void setNumbering(float numbering) { this->numbering = numbering; }  void setStarPoint(float starPoint) { star\_point = starPoint; }  void setReview(const string& review) { this->review = review; }  void setPage(int page) { this->page = page; }  void setRentStatus(bool rent) { this->rent = rent; }  //책 정보 출력  void BookInfo() const {  cout << "Book Name : " << bookname << endl;  cout << "Writer : " << writer << endl;  cout << "Publisher : " << publisher << endl;  cout << "Genre : " << genre << endl;  cout << "Numbering : " << numbering << endl;  cout << "Star Point : " << star\_point << endl;  cout << "Review : " << review << endl;  cout << "Page : " << page << endl;  cout << "Rent Status: " << (rent ? "Rented" : "Available") << endl;  }    //연체 상태 확인  bool checkOverdue() const {  time\_t now = time(0);  tm ltm;  localtime\_s(&ltm, &now);  //대여한 날짜에서 대여 가능 기간을 더한 날짜  tm dueDate = ltm;  dueDate.tm\_mday += (getDaysLimit() - 1);  //현재 날짜가 연체 날짜를 지났는지 확인  return difftime(mktime(&dueDate), now) < 0;  }  };  class Member {  private:  string id; //사용자 id  string pw; //비밀번호  string name; //이름  string job; //직위(학생, 대학원생, 교수)  int booklimit; //빌릴 수 있는 권수  int daylimit; //대여 기한  public:  Member(const string& memberId, const string& password, const string& memberName, const string& memberJob, int bookLimit, int daysLimit)  : id(memberId), pw(password), name(memberName), job(memberJob), booklimit(bookLimit), daylimit(daysLimit) {  }  ;  string getId() const { return id; }  string getPassword() const { return pw; }  string getName() const { return name; }  string getJob() const { return job; }  int getBookLimit() const { return booklimit; }  int getDaysLimit() const { return daylimit; }  void setId(const string& memberId) { id = memberId; }  void setPassword(const string& password) { pw = password; }  void setName(const string& memberName) { name = memberName; }  void setJob(const string& memberJob) { job = memberJob; }  void setBookLimit(int bookLimit) { booklimit = bookLimit; }  void setDaysLimit(Member& member) {  if (member.getJob() == "student") {  daylimit = 14;  }  else if (member.getJob() == "graduate student") {  daylimit = 21;  }  else if (member.getJob() == "professor") {  daylimit = 28;  }  }    };  class NotReturned : public exception {  public:  virtual const char\* what() const throw() { //가상함수로  return "Error: 책이 반납되지 않았습니다. 반납 후 리뷰를 작성할 수 있습니다.";  }  };  void addReview(Member& member, Book& book, float starPoint, const string& review) {  try {  //책의 대여 상태 확인  if (book.isRented()) {  //책 정보 출력  cout << "Book Information:\n";  book.BookInfo();  //별점 및 리뷰 등록  book.setStarPoint(starPoint);  book.setReview(review);  cout << "리뷰가 성공적으로 등록되었습니다. " << book.getBookName() << "'.\n";  }  else {  //대여되지 않은 책에 대한 오류 출력  throw NotReturned();  }  }  catch (const exception& e) {  cout << e.what() << endl;  }  }  int main() {  //임시로 멤버와 책 생성  Member member("idid", "pwpw", "name", "student", 5, 14);  Book book("book1", "writer1", "publisher1", "genre1", 12345, 0.0, "", 300, true);  int choice;  //임시 로그인  if (member.getId() == "idid" && member.getPassword() == "pwpw") {  cout << "로그인 성공!\n";  //사용자의 메뉴 선택  cout << "1. 연체상태 확인\n";  cout << "2. 리뷰 등록\n";  cout << "메뉴를 선택하세요: ";  cin >> choice;  if (choice == 1) {  //연체 상태 확인  if (book.checkOverdue()) {  cout << "연체되었습니다.\n";  }  else {  cout << "대여 가능 기간입니다. 연체되지 않았습니다. "  << member.getName() << "님의 대여 가능기간은 " << member.getDaysLimit() << "입니다." << endl;  }  }  else if (choice == 2) {  //리뷰 등록  float starPoint;  string review;  //책 정보 출력  cout << "Book Information:\n";  book.BookInfo();  //사용자가 별점과 리뷰 입력  cout << "별점을 입력하세요(0.0 ~ 5.0): ";  cin >> starPoint;  cout << "리뷰를 입력하세요: ";  cin.ignore();  getline(cin, review);  //리뷰 등록  addReview(member, book, starPoint, review);  }  else {  cout << "잘못된 메뉴를 선택하셨습니다.\n";  }  }  else {  cout << "로그인 실패.\n";  }  return 0;  } |