# 과제: 디지털 꾸러미 만들기

| | |
|---|---|
| 과목명 | 프로그래밍언어론 |
| 교수명 | 박재화 |
| 제출일 | 2022.12.21 |
| 학 번 | 20206147 |
| 학 과 | 응용통계학과 |
| 이 름 | 서은서 |

# <목차>

# Ⅰ. 서론

1) 배경

나는 이번 학기에 프로그래밍언어론과 객체지향프로그램을 동시에 수강하였다. 두 과목은 상당히 밀접한 연관성이 있었는데, 특히 꾸러미 만들기는 객지프에서 동일한 개념을 배웠기 때문에 이 과제는 더 친근하게 느껴졌다. 따라서 이 과제에서는 객체지향에서 배웠던 내용과 프로그래밍언어론에서 배웠던 개념을 동시에 사용하여 하나의 프로그램을 만들어 보고자 한다. 시험기간인만큼 지금 나에게 가장 친숙한 공간은 도서관이고 가장 친숙한 것은 책이었다. 이에 나는 도서관에서 관리자의 입장으로 도서 관리시스템과 관련하여 꾸러미를 만들어 보고자 한다.

2) 사용 데이터

사용한 데이터는 Kaggle(https://www.kaggle.com/datasets/sootersaalu/amazon-top-50-bestselling-books-2009-2019)에서 받아왔다. 이 데이터는 Amazon의 2009년부터 2019까지의 베스트셀러 Top50을 수집하였다. 총 550권의 책이 들어가 있으며, Fiction과 Non-Fiction으로 분류되었다. 나는 이 데이터를 내가 구현하고자 하는 프로그램의 기본정보로 활용하고자 했다. 이 데이터는 나의 프로그램내에서 가공되기에 너무 버겁다는 생각이 들었고 약간의 수정을 가미하여 최종으로 내가 사용할 데이터를 만들었다. 실제 데이터는 엑셀 형식으로 되어있지만 이를 txt파일로 변환하여 사용하였다. (이때 책 제목, 저자, 출고일, 가격, 분류는 " , (콤마) "를 이용하여 분리하였다.) 내가 최종으로 이용할 데이터는 위의 데이터에서 가장 위의 데이터 26개를 가져왔다. 그리고 실제 데이터에서는 책 제목이 띄어쓰기가 되었지만 내가 가공한 데이터 내에서는 띄어쓰기가 존재하지 않다. 저자명도 마찬가지로 띄어쓰기가 되어있지 않다. 자세한 이유는 본론에서 설명하고자 한다. 첨부파일에 처음 받아온 데이터와 최종으로 사용한 데이터를 포함하였다.

3) 구현하고자 하는 내용(범위)

내가 만들고자 하는 프로그램은 '도서 관리 시스템'이다. 흔히 말해 도서관에서 사용하는 도서 대출, 반납시스템이라고 생각하면 될 것 같다.  이 시스템의 주요 쟁점은 '도서를 대출하고 반납하려는 고객과 어떻게 상호작용 할 것인가'이다. 뿐만 아니라 도서관 내부에서도 손님과의 상호작용에 따라 재고 혹은 재정의 변동이 생길 것이다. 최근 도서관에서는 키오스크라는 기계를 도입하여 대면이 아니더라도 고객이 책을 간편하게 대출, 반납할 수 있다. 내가 이번 과제에서 만들 것은 가상의 기계이다. 우리는 이 기계를 이용하여 대출, 반납, 책의 재고 추가를 할 수 있다.

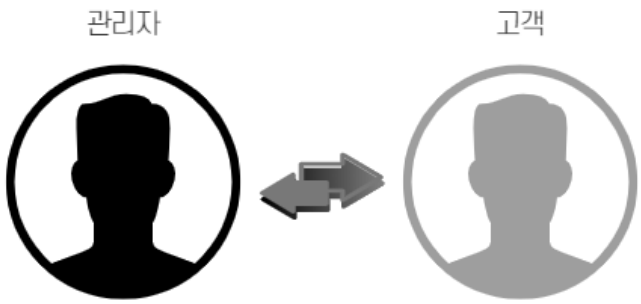
# Ⅱ. 본론

1) 실제 도서관리 시스템 구현방식

실제 도서관에서 어떤 상황들이 일어나는지에 관해 이야기해보자. 실제 도서관에서 발생할 수 있는 상황은 크게 두개로 나뉜다. 도서관과 고객과의 상호작용, 도서관 내부에서의 변화이다.

A. 도서관과 고객과의 상호작용

이는 모두가 너무나도 잘 알듯이 도서의 대출, 반납이 있다. 더 구체적으로 들어가면 대출한 날짜로부터 제한된 일자가 지났을 때까지 반납을 하지 않은 경우 연체가 되어 사용자에게 패널티를 부여할 수 있다. 예를 들어 일정기간 동안 책을 대출하지 못한다거나 바로 패널티를 풀고 싶으면 연체료를 부과하여 이를 없앨 수 있다. 이 두가지 기능을 제외하고도 본인이 원하는 도서가 도서관 내에 존재하는지 찾을 수 있을 뿐더러 만약 원하는 책이 없다면 도서관에 신청을 통해 책을 주문 할 수 있다. 만약 고객이 원하는 책이 도서관 재고에 존재하나 다른 사람이 대출 중이라면 고객은 다른 고객이 이 도서를 반납할 때 까지 기다려야한다. 이 때 고객은 책에 대한 예약을 걸어 둘 수 있다. 이 때 고객은 도서가 반납 될 시 바로 대여가 가능해 진다. 최근 도서관들은 인터넷도 활성화 되어있기 때문에 대면 뿐만 아니라 비대면으로도 고객과의 소통이 가능하다. 만약 고객이 도서관 민원게시판에 글을 작성했다면 도서관은 이를 해결할 의무를 가지고 있다.




B. 도서관 내부에서의 변화

도서관 내부에서의 변화는 고객과의 상호작용의 여파로써 작용한다. 만약 고객이 연체를 했을 때 연체료를 지불한다면 도서관 내부의 자본은 증가 할 것이다. 만약 고객이 책 주문을 신청했을 때 도서관은 이 불음에 응답하기 위해 자본을 이용하여 책을 사들일 것이다. 책에는 유통기한이 없지만 고객이 소비 할 수 없을 만큼 헤졌거나 손상이 있을 경우에는 책을 폐기 처분하고 새로운 책을 구입해야 한다. 이때도 자본의 변화와 재고의 변화가 일어난다. 만약 유명한 작가의 책이 발간되었거나 기존에 보유하고 있던 시리즈물의 새로운 편이 나왔다고 하자. 이런 경우 고객이 요구

하지 않아도 도서관에서 먼저 고객의 니즈를 예측하고 도서를 구입할 수 있다. 이 경우에도 자본의 변화와 도서 재고의 변화가 일어난다.

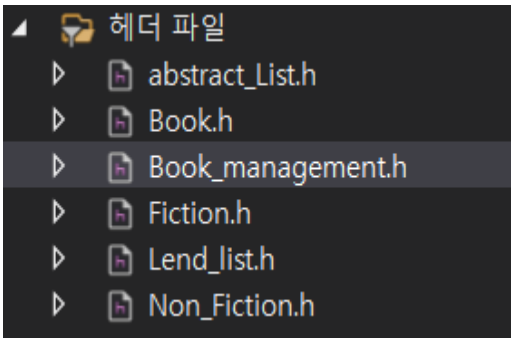

2) 내가 구현한 시스템의 설계

1)에서는 실제 도서관에서 발생하는 상황들을 고객과의 상호작용 그리고 도서관 내부에서의 변화로 나누어 설명하였다. 이번 2)에서는 내가 만든 프로그램에서 1)의 내용을 어떻게 녹여내어 구현하였는지에 대해 이야기할 것이다. 본격적으로 설명에 들어가기에 앞서 내가 실제 도서관에서 발생할 수 있는 상황에서 어떤 것을 구현하였고 어떤 것을 구현하지 않았는지에 대해 설명하고자 한다. 1)에서 말한 상황들을 대략적으로 말하자면 대출, 반납, 연체(연체료 부과, 일정기간 빌리지 못하게 함), 도서 예약 시스템, 도서 구입 신청, 신간 구입, 도서관 자본의 변화, 도서관 재고의 변화, 민원처리가 될 것이다. 나는 이것들 중 도서 대출, 반납, 도서 구입 신청 + 신간 구입 = 도서 재고 추가, 도서관 재고의 변화에 대해 구현 할 것이다. 이때 제한을 둘 것은 내 프로그램 도서관에 존재하는 모든 도서의 재고는 1이다. 또한 실제 도서관에서 책을 대출하거나 반납 할 때 본인 인증을 위하여 비밀번호 혹은 대출증의 바코드를 찍어 신분확인을 하는데 우리 프로그램은 오직 나(작성자)를 위해서만 존재하는 프로그램으로 사용자가 한 명이기 때문에 신분확인이 필요하지 않다. 따라서 비밀번호 입력 혹은 개인정보 입력(바코드를 찍는 것을 대체함)은 구현하지 않을 것이다. 이제 본격적으로 설명에 들어갈 것이다.

A. Kaggle의 Data 사용

위에서(서론-2)) 설명했듯이 Kaggle에서 받아온 데이터를 가공하여 사용할 것이다. 이때 책의 제목은 띄어쓰기 즉 공백이 제거 되어있다. 이유는 다음과 같다. 내가 구현할 프로그램에서의 대출, 반납기능에서는 도서관에서 보유하고 있는 도서 목록과 대출 된 도서들이 담겨있는 목록을 가지고 있다. 만약 대출, 반납을 시행하고 싶다면 먼저 두개의 목록에서 내가 원하는 책이 있는지를 찾아봐야 할 것이다. 이 때 우리는 책의 제목을 입력하게 되는데 이때 띄어쓰기를 입력하게 되면 제목에 들어가지 않고 다른 곳에 들어가는 오류가 발생한다. 다른 방안이 있을 수 있지만 이러한 오류 때문에 Kaggle에서 받아온 데이터에서 제목을 띄어쓰기 없이 수정하여 사용하겠다. 따라서 찾고자 하는 책을 입력 받을 때 또한 띄어쓰기 없는 제목을 입력받는다.

B. 헤더파일의 분할




내가 만든 프로그램에서는 총 6개의 헤더파일과 7개의 cpp파일이 존재한다.

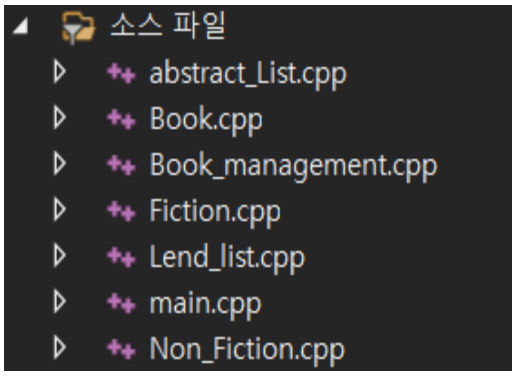
이렇게 헤더파일을 많이 만든 이유는 main() cpp 를 제외한 6개의 cpp 파일 즉, 만들어진 오브젝트 파일에 들어있는 함수들의 내용을 다른 소스 파일에서 사용할 수 있도록 하기 위해서 입니다. 간단하게 생각하면 library와 동일시 할 수 있다. 내 프로그램에서 예를 들자면 (클래스에 대한 자세한 설명은 다음 2)-C에서 다룰 것이다.) Book_management class는 Lend_list class에 있는 대출된 책의 목록과 도서관에서 보유하고 있는 도서 Fiction과 Non-Fiction의 도서 목록을 필요로 한다.

이때 Book_management.h에서는 단 #include "Fiction.h" , #include "Non_Fiction.h" , #include "Lend_list.h" 세 줄을 통해 구현할 수 있다.

뿐만 아니라 book이라는 namespace를 새로 만들어 이 안에 클래스의 모든 메소드를 넣음으로써 main() 에서 using namespace book; 한 줄을 통해 모든 메소들을 사용할 수 있다.

C. 클래스의 분업

이 프로그램에서는 대출, 반납 기능을 주로 한다. 이때 모든 기능들을 분할하지 않고 하나의 클래스로 만든다면 변수간의 사용이 자유롭고 한 소스파일 내에서 작성가능하다는 장점이 있지만 이런 경우 하나의 클래스에 대한 의존성이 높아진다. 따라서 코드를 쉽게 변경할 수 없고, 만약 변경한다 하더라도 서로간의 의존성이 높기 때문에 프로그램실행이 잘 안될 수 있다.




우리는 이러한 오류발생의 가능성을 줄이고 코드의 수정이 비교적 쉽도록 cpp파일을 여러개로 분리하였다. 이는 SOLID 원칙 중 "Single Responsibility Prinicple"에 따른다. 이 원칙은 하나의 클래스는 오직 변화에 대한 한가지 이유만을 가진다. 즉 한가지 책임만 가짐을 의미한다. 어떻게 보면 본인 클래스의 책을 제외한 나머지는 다른 클래스에게 전과한다고도 볼 수 있다. 다음은 main을 제외한 총 6개의 cpp 파일들이다.

<cpp파일>

i. Abstract_list.cpp : 도서의 분류를 위한 클래스 Fiction, Non-Fiction가 상속하는

클래스로 Fiction class, Non-Fiction class의 공통된 메소드를 담고 있는 추상 클래스이다.

ii. Book.cpp : 실질적인 도서를 만들 수 있는 클래스로 Book class로 찍어낸 객체 하나는 도서 한권이 된다. 안에는 생성자와 소멸자가 들어있다.

iii. Book_management.cpp : 내 프로그램은 이 클래스을 이용하여 대출, 반납, 신간 수입, 도서 목록보기가 가능하다. 이 클래스 내에서 실질적인 구현이 일어나지는 않으나 이 클래스를 이용하여 위의 기능들을 모두 수행할 수 있다. 더 자세한 내용은 2)-D에서 다루도록 하겠다.

iv. Fiction.cpp : 분류가 Fiction인 도서를 다룬다.

v. Non_Fiction.cpp : 분류가 Non_Fiction인 도서를 다룬다.

vi. Lend_list.cpp : 도서를 대출할 때 대출된 책이 담기는 클래스이며 도서의 대출과 반납에서 이용된다.

vii. Main.cpp: Kaggle에서 가져와 가공한 txt파일을 한 줄 씩 읽어오면서 Book 객체를 생성한다. txt파일의 한 줄은 콤마를 기준으로 도서명, 저자, 출간일, 가격, 분류가 나누어지기 때문에 반복문을 통해 콤마를 기준으로 5덩어리를 내어 각각의 property에 넣어준다. 이 과정을 txt를 다 읽을 때까지 반복한다.

D. Book_management의 이용

위에서 언급했듯이 Book_management class를 통해 우리는 대출, 반납, 도서 추가, 도서 목록 조회가 가능하다. 즉 이 클래스를 통해서 나머지 class에 접근이 가능하다는 것이다. 이 클래스는 도서 추가를 제외한 다른 메소드들 (도서 대출, 도서 반납, 도서 목록 조회)의 실질적인 코드를 통한 구현은 하지 않고 오직 다른 클래스의 메소드를 이용하여 구현하고 있다. 예를 들어 도서 대출의 경우 Book_management에 있는 lend_book()를 이용하기만 하면 저절로 lend_list class에 있는 메소드를 이용해 도서목록에 추가된다. 즉 이는 고객과 도서관이 소통을 할 때 이 Book_management 만을 이용하면 된다는 것을 의미한다.

E. 왜 C++을 사용하였는가?

i. Encapsulation

"Encapsulation"란 변경이 가능하거나 변경이 쉬운 부분을 잘 변하지 않는 부분 뒤로 숨기는 것을 의미한다. 대표적인 예시로는 위에서 설명했던 Book_management class가 있다. 나의 프로그램에서 고객에서 실질적으로 보여

지는 부분 즉 main()에서 사용하고 있는 클래스는 오직 Book_management class 하나이다. 우리는 이 하나의 클래스만을 사용하여 모든 메소드에 접근이 가능하다. 우리가 실질적으로 사용되는 메소드들(lend_book, return_book() 등...)은 쉽게 변경을 할 수 있다. 만약 겉으로 노출이 되어있다면 말이다. 나의 프로그램에서는 Book_management class 뒤에 다른 클래스들을 숨김으로써 쉽게 접근을 할 수 없게 했다. 또 하나의 좋은 점은 이를 통해 클래스 간의 복잡도가 줄어든다.

ii. Abstraction

Abstraction란 핵심이 되는 요소만을 뽑아서 모아둔 것을 의미하며 확대를 전제로 한다. 즉 개념적인 상을 만든다고 할 수 있다. 나의 프로그램에서는 분류가 해당된다. 책이라는 객체는 분류가 Fiction 혹은 Non_Fiction으로 나눠지는 것 외에는 동일한 행동을 하고 있다. 따라서 abstract_list라는 추상클래스를 만들어 Fiction class와 Non_Fiction class가 상속하여 동일한 메소드을 이용하지만 가지고 있는 도서 목록은 다르게 만들 수있다. 만약에 내가 분류에 두가지 외에 에세이라는 분류를 추가하고 싶다고 하자 이때 나는 별로의 복잡한 수정 없이 에세이라는 클래스를 하나 만들고 abstract_list를 상속받아 사용하면 된다. 이 처럼 추상클래스를 사용하면 확장이 쉽다.

iii. Inheritance

Inheritance이란 부모클래스의 메소드를 자식클래스가 물려받아 사용할 수 있음을 의미한다. 예시는 (ii)에서 설명하였다.

iv. Polymorphison

Polymorphison이란 여러 개의 서로 다른 객체가 동일한 기능을 서로 다른 방법으로 처리할 수 있음을 의미하여 상속을 통해 구현될 수 있다. 다형성의 구현방법은 여러가지인데 대표적으로 오버로딩과 오버라이딩이있다. 오버로딩이란 동일한 클래스 내에서 동일한 반환타입, 동일한 메소드 이름을 가졌지만 받아오는 인자에 따라 구현되는 내용이 다른 것을 말한다. 오버라이딩이란 상속을 통해 부모클래스의 메소드를 자식클래스가 받아오지만 자신의 메소드로 재사용할 수 있음을 의미한다. 내 프로그램에서는 Polymorphison이 들어가지 않았으나 추가되면 한층 완벽한 프로그램이 될 것이다.

F. 기타

프로그램내에서는 list와 map을 사용하고 있다.

i. List

list는 도서관에서 보유한 도서를 보관할 때 사용한다. 왜 array대신 list를 사용했을까? Array를 사용할 시 우리는 초기에 크기를 지정해주어야 한다. 만약 우리가 넣고 싶은 값이 array의 크기를 초과한다면 우리는 array의 크기를 늘리는 작업

을 수행 한 뒤 집어 넣어야 한다. 그러나 list는 초기에 크기를 지정해주지 않아도 되며 추가와 삭제가 편리하다. 도서목록의 경우 수정이 불가피하게 빈번하게 일어남으로 list를 사용하는 것이 좋을 것이라 판단하였다.

ii. Map

map은 대출된 도서를 저장할 때 사용한다. 왜 list가 아니라 map을 사용했을까? List는 데이터의 수정과 삭제가 용의하지만 그것은 데이터가 리스트의 끝자락에 존재했을 때의 이야기이다. 리스트의 중간 값을 제거하기 위해서는 iterator를 이용하여 데이터의 위치를 파악한 후 제거를 해야 한다. 대출된 도서의 목록에서는 고객이 반납하고자 하는 책의 위치를 찾는 행위가 빈번하게 일어나기 때문에 list를 사용하는 것은 올바르지 않다고 판단하였다. Map의 경우 원하는 도서의 이름을 입력하면 그 와 짝을 이루는 도서의 객체를 반환해줌으로 적합하다. 그러나 이를 위해서는 lend_list에 저장할 때 <도서명, 도서객체>가 한 쌍을 이루도록 저장해주어야 하는 번거로움이 있다.

# Ⅲ. 결론

1) 아쉬운 점 및 보완 할 점

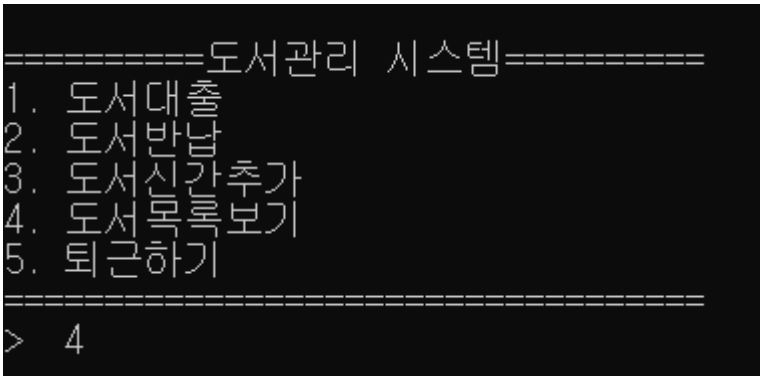
문자입력 시 띄어쓰기를 포함하지 못한다는 것이 굉장한 큰 오점이라 생각한다. 이를 보완하여 만약 띄어쓰기를 포함하여 받을 수 있다면 도서의 이름을 명확하게 구분하고 보다 정확한 프로그램이 될 수 있을 것이다. 내 프로그램에서 여러 권의 도서를 대출 할 시 한번에 여러 개의 도서를 대출할 수 있는 것이 아니라 한 질문 당 한권의 책을 대출, 반납 할 수 있다. 따라서 이를 반복문을 추가한면 질문을 여러 번 반복해야 하는 수고를 덜 수 있다. 또한 오늘의 날짜를 설정하고 도서를 빌리고 lend_list에 추가할 때 날짜 또한 함께 저장하여 +n일 후에 (반납일) < (빌린 날짜+n) 일때 연체에 대한 패널티를 부가하는 메소도 추가되면 좋을 것 같다.

2) 기대되는 점

만약 이 프로그램에 재정적인 부분이 추가된다면? 즉 자산을 관리하는 클래스를 추가한다면 도서관에서 확장하여 서점에서도 이 프로그램을 사용할 수 있을 것으로 기대한다.

3) 구현결과

A. 시작화면




B. 도서목록보기

i. Fiction

ii. Non_Fiction

```
    Non_Fiction
=========================
<1>
제목: 10-DayGreenSmoothieCleanse
저자: JJSmith
장르: non_Fiction

--------------------------------
<2>
제목: 12RulesforLife:AnAntidotetoChaos
저자: JordanB.Peterson
장르: non_Fiction

--------------------------------
<3>
제목: "5
저자: 000AwesomeFacts(AboutEverything!)(NationalGeographicKids)"
장르: non_Fiction

--------------------------------
<4>
제목: "AHigherLoyalty:Truth
저자: Lies
장르: non_Fiction

--------------------------------
<5>
제목: APatriot'sHistoryoftheUnitedStates:FromColumbus'sGreatDiscoverytotheWaronTerror
저자: LarrySchweikart
장르: non_Fiction
```

C. 도서 대출(Fiction)

```
==========도서관리 시스템==========
1. 도서대출
2. 도서반납
3. 도서신간추가
4. 도서목록보기
5. 퇴근하기
====================================
> 1

도서대출을 시작합니다.

책의 장르를 입력해주세요(Fiction or Non_Fiction): fiction

<1>
제목: 11/22/63:ANovel
저자: StephenKing
장르: fiction

--------------------------------
<2>
제목: 1984(SignetClassics)
저자: GeorgeOrwell
장르: fiction

--------------------------------
<3>
제목: ADancewithDragons(ASongofIceandFire)
저자: GeorgeR.R.Martin
장르: fiction

--------------------------------
```

```
어떤 책을 빌리시겠습니까?(띄어쓰기 없이 입력): Allegiant
  "Allegiant"을 대출하였습니다.
```

D. 도서 대출(Non_Fiction)

```
==========도서관리 시스템==========
1. 도서대출
2. 도서반납
3. 도서신간추가
4. 도서목록보기
5. 퇴근하기
====================================
>  1

도서대출을 시작합니다.

책의 장르를 입력해주세요(Fiction or Non_Fiction): nonfiction

<1>
제목: 10-DayGreenSmoothieCleanse
저자: JJSmith
장르: non_Fiction

--------------------------------
<2>
제목: 12RulesforLife:AnAntidotetoChaos
저자: JordanB.Peterson
장르: non_Fiction

--------------------------------
<3>
제목: "5
저자: OOOAwesomeFacts(AboutEverything!)(NationalGeographicKids)"
장르: non_Fiction
```

```
  "Allegiant"을 대출하였습니다.
어떤 책을 빌리시겠습니까?(띄어쓰기 없이 입력): Allegiant
```

E. 도서 반납

```
==========도서관리 시스템==========
1. 도서대출
2. 도서반납
3. 도서신간추가
4. 도서목록보기
5. 퇴근하기
====================================
> 2

도서반납을 시작합니다.

 <대출리스트>
<1>
제목: AStolenLife:AMemoir
저자: JayceeDugard
장르: non_Fiction

--------------------------------
<2>
제목: Allegiant
저자: VeronicaRoth
장르: fiction

--------------------------------
어떤 책을 반납하시겠습니까?: AStolenLife:AMemoir
도서 "AStolenLife:AMemoir"가 반납되었습니다.
```

F. 도서 추가

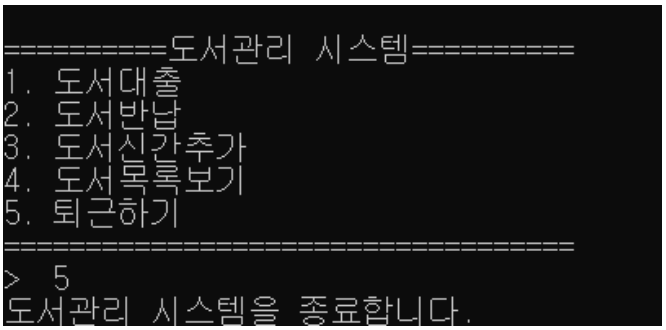
```
==========도서관리 시스템==========
1. 도서대출
2. 도서반납
3. 도서신간추가
4. 도서목록보기
5. 퇴근하기
====================================
> 3

신간 추가를 시작합니다.
책의 제목을 입력해주세요: Hello
책의 저자를 입력해주세요: 서은서
책의 장르를 입력해주세요(Fiction or Non_Fiction): nonfiction
책의 가격을 입력해주세요: 10
책의 출간년도를 입력해주세요: 2022
```

-> 생성 결과


---
<14>
제목: Hello
저자: 서은서
장르: non_Fiction
---

G. 퇴근하기


==========도서관리 시스템==========
1. 도서대출
2. 도서반납
3. 도서신간추가
4. 도서목록보기
5. 퇴근하기
==========
> 5
도서관리 시스템을 종료합니다.