2020-1학기 데이터베이스 시스템 01분반

**대용량 관계 DB 기반 응용 개발 Project**



중앙대학교 컴퓨터공학부 컴퓨터전공

20176758 박진영

**-목차-**

1. **개발환경 ……………………………………………………………………………………………………………………… 3**
2. **응용 분야 ……………………………………………………………………………………………………………………. 4**
3. 응용 분야 제목
4. 응용 분야 소개
5. 응용의 기능 리스트 및 설명
6. **관계 DB 스키마 및 설명 ……………………………………………………………………………………………….. 5**
7. DB 스키마
8. 설명
9. **트랜잭션 설계 결과 및 지원 기능 …………………………………………………………………………………… 7**
10. 동적 SQL문 리스트
11. 트랜잭션 리스트 및 지원 기능 설명
12. **소스 코드 설명 …………………………………………………………………………………………………………… 12**
13. Main.java
14. Transaction.java
15. 구현요건 충족에 관한 상세 설명
16. **테스트 레코드 생성 …………………………………………………………………………………………………...... 38**
17. 랜덤 데이터 생성
18. 데이터 로드
19. 로드 결과
20. **응용 실행 데모 …………………………………………………………………………………………………………… 47**
21. 도서 검색
22. 대출, 예약, 반납
23. 마이페이지
24. 관리
25. **소스 코드 실행 관련 필요사항 ……………………………………………………………………………………… 67**
26. **사용자 매뉴얼 …………………………………………………………………………………………………………….. 68**

**1. 개발 환경**

- System OS: window 10

- DBMS 제품명 및 버전: Mysql community server 8.0.20

- 대화식 SQL 도구: MySQL 8.0 command line client

- Java version: 14

- Console application으로 개발

**2. 응용 분야**

**1) 응용 분야 제목**: 도서관

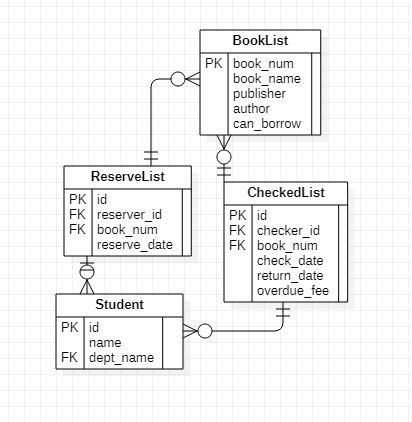
**2) 응용 분야 소개**: 대학 도서관의 도서 목록, 대출, 예약 기능 등을 제공한다.

**3) 응용의 기능 리스트**

1. 도서 검색: 도서관에 소장 중인 도서 목록을 도서명, 출판사, 저자, 도서번호로 검색할 수 있다.
2. 도서 대출: 학생은 도서 목록 중 대출 가능한 도서를 빌릴 수 있다.
3. 도서 반납: 대출한 도서를 반납한다.
4. 도서 예약: 대출을 원하는 도서를 타인이 빌리지 못하게 예약할 수 있다.
5. 내 정보: 내 대출, 예약, 연체료 정보를 확인할 수 있다.
6. 연체료 계산: 연체에 따라 발생한 요금을 계산한다. 해당 응용에선 반납 시에 결제하는 것으로 한정한다.
7. 도서 목록 수정: 존재하는 도서 목록에 대해 추가, 수정, 삭제를 할 수 있다.

**3. 관계 DB 스키마 및 설명**

**1) DB 스키마** (왼: DB 스키마, 오: Table 생성 결과) : 테이블 생성 구문은 별도 첨부 ( CreateTable.txt)



**2) 설명**

|  |  |  |
| --- | --- | --- |
| 테이블명 | 컬럼명 | 설명 |
| Student | id (Int) | 학생의 고유번호, Primary key |
| name (Char) | 학생의 이름 |
| BookList | book\_num (Int) | 도서의 고유 번호, Primary key |
| book\_name (Char) | 도서명 |
| publisher (Char) | 출판사 |
| author (Char) | 저자 |
| can\_borrow (Boolean) | 대출 가능한 지의 여부 |
| CheckedList | id (Int) | 대출 정보의 고유 번호, Primary key  Auto Increasement |
| checker\_id (Int) | 대출자 id  Student Table의 id를 외래키로 사용 |
| book\_num (Int) | 도서 번호  BookList Table의 book\_num을  외래키로 사용 |
| check\_date (Datetime) | 대출 일자 |
| return\_date (Datetime) | 반납 일자 |
| overdue\_fee (Int) | 연체료 |
| ReserveList | id (Int) | 예약 정보의 고유 번호, Primary key  Auto Increasement |
| reserve\_id (Int) | 예약자 id  Student Table의 id를 외래키로 사용 |
| book\_num (Int) | 도서 번호  BookList Table의 book\_num을  외래키로 사용 |
| reserve\_date (Datetime) | 예약 일자 |

**4. 트랜잭션 설계 결과 및 지원 기능**

**1) 동적 SQL 리스트**

1. S1

S1-1. 도서 목록에서 도서명으로 결과 검색

* "SELECT \* FROM BookList WHERE book\_name like ?"

S1-2. 도서 목록에서 출판사로 검색

* "SELECT \* FROM BookList WHERE publisher like ?"

S1-3. 도서 목록에서 저자로 검색

* "SELECT \* FROM BookList WHERE author like ?"

S1-4. 도서 목록에서 도서 고유 번호로 결과 검색

* "SELECT \* FROM BookList WHERE book\_num = ?"

S1-5. 도서 목록에서 도서명을 포함하는 개수 검색

* "SELECT count(\*) FROM BookList WHERE book\_name like ?"

S1-6. 도서 목록에서 출판사를 포함하는 개수 검색

* "SELECT count(\*) FROM BookList WHERE publisher like ?"

S1-7. 도서 목록에서 저자를 포함하는 개수 검색

* "SELECT count(\*) FROM BookList WHERE author like ?"

S1-8. 도서 목록에서 도서 고유 번호 개수 검색. 고유 번호가 리스트에 존재하는지 여부를 판단하기 위해 필요.

* "SELECT count(\*) FROM BookList WHERE book\_num = ?"

1. S2

S2-1. 도서 목록에 도서 정보 추가

* "INSERT INTO BookList VALUES (?,?,?,?,?)"

S2-2. 도서 정보 수정

* "UPDATE BookList SET book\_name=?, publisher=?, author=? WHERE book\_num=?"

S2-3. 도서 목록에서 도서 고유 번호에 해당하는 row 삭제

* "DELETE FROM BookList WHERE book\_num=?"

S2-4. 도서 고유 번호에 해당하는 row의 can\_borrow 컬럼값 수정.

* "UPDATE BookList SET can\_borrow=? WHERE book\_num=?"

1. S3

S3-1. 대출 리스트에 대출 정보 추가

* "INSERT INTO CheckedList (checker\_id,book\_num) VALUES (?,?)"

S3-2. 대출 리스트에서 return\_date 수정. 반납 기능에 필요.

* "UPDATE CheckedList SET return\_date=? WHERE checker\_id=? and book\_num=? and return\_date is NULL"

1. S4

S4-1. 예약 리스트에 예약 정보 추가

* "INSERT INTO ReserveList (reserver\_id, book\_num) VALUES (?,?)"

S4-2. 예약자 번호와 도서 번호로 예약 정보 조회

* "SELECT \* FROM ReserveList WHERE reserver\_id = ? and book\_num=?"

S4-3. 예약 정보 삭제

* "DELETE FROM ReserveList WHERE id = ?"

1. S5

S5-1. 대출자 번호로 대출 정보 조회

* "SELECT \* FROM CheckedList WHERE checker\_id = ?"

S5-2. 예약자 정보로 예약 정보 조회

* "SELECT \* FROM ReserveList WHERE reserver\_id = ?"

S5-3. 대출자 번호로 대출 정보 개수 검색

* "SELECT count(\*) FROM CheckedList WHERE checker\_id = ?"

S5-4. 예약자 번호로 예약 정보 개수 검색

* "SELECT count(\*) FROM ReserveList WHERE reserver\_id = ?"

1. S6

S6-1. 대출자 번호로 반납 일자가 지난 대출 정보 개수 검색

* SELECT count(\*) FROM CheckedList WHERE checker\_id = ? and return\_date is NULL and date\_format(check\_date, '%Y-%m-%d')<?"

S6-2. 대출자 번호로 반납 일자가 지난 대출 정보 조회

* "SELECT \* FROM CheckedList WHERE checker\_id = ? and return\_date is NULL and date\_format(check\_date, '%Y-%m-%d')<?"

S6-3. 대출 정보의 연체료 정보 수정

* "UPDATE CheckedList SET overdue\_fee=? WHERE id=?"

**2) 트랜잭션 리스트 및 지원 기능 설명**

1. T1 (검색 only)
2. T1-1: 도서 목록 검색

* S1-1 ~ S1-8의 동적 SQL 질의문을 사용해 도서명, 출판사, 저자, 도서번호로 원하는 목록을 조회하는 것과 Pagination 기능을 구현한다.

1. T1-2: 대출 정보 검색

* S5-1, S5-3의 동적 SQL 질의문을 사용해 대출자 번호에 따른 대출 정보 조회와 Pagination 기능을 구현한다.

1. T1-3: 예약 정보 검색

* S5-2, S5-4의 동적 SQL 질의문을 사용해 예약자 번호에 따른 예약 정보 조회와 Pagination 기능을 구현한다.

1. T2 (변경)
2. T2-1: 도서 정보 추가

* 도서번호, 도서명, 출판사, 저자를 입력 받아 S2-1의 동적 SQL 변경문을 사용해 도서 정보를 추가한다.

1. T2-2: 도서 정보 삭제

* 도서번호를 입력 받아 S2-3의 변경문을 사용해 도서 정보를 삭제한다.

1. T2-3: 반납

* 도서번호와 대출자 ID를 입력 받아 S2-4 변경문을 사용해 도서 정보의 can\_borrow 정보를 수정한다.
* S3-2 변경문으로 대출 정보에 반납 일자를 추가한다.

1. T3 (검색, 변경 혼합)
2. T3-1: 도서 정보 수정

* 도서번호를 입력 받아 S1-4 질의문을 사용해 해당되는 도서 정보를 보여준다.
* 수정하기 원하는 컬럼을 선택한 후 새로운 값을 입력 받는다.
* 새로 입력 받은 값 이외의 정보는 질의문을 사용했을 때 얻은 정보를 사용한다.
* S2-2 변경문을 사용해 도서 정보를 수정한다.

1. T3-2: 도서 대출

* 도서명을 입력 받고 도서명을 가진 정보가 1개 이상이라면 S1-1, S1-5 질의문을 사용해 도서 목록을 Pagination하여 보여준다.
* 빌리기 원하는 도서의 번호를 입력 받고 S1-4, S1-8 질의문을 사용해 해당 도서 정보를 가져온다.
* 도서명을 가진 정보가 1개라면 S1-1를 사용해 해당 도서 정보를 가져온다.
* 학생 ID를 입력 받고 S4-2 질의문을 사용해 같은 Id, 도서번호를 가진 예약 정보가 있는 지 조회한다.
* 예약 정보가 있다면 결과를 보여준 후 S4-3 변경문을 사용해 예약 정보를 삭제한다.
* S3-1 변경문으로 대출 정보를 추가한다.
* S2-4 변경문으로 도서의 대출 가능 값을 False로 수정한다.

1. T3-3: 도서 예약

* 도서명을 입력 받고 도서명을 가진 정보가 1개 이상이라면 S1-1, S1-5 질의문을 사용해 도서 목록을 Pagination하여 보여준다.
* 예약하길 원하는 도서의 번호를 입력 받고 S1-4, S1-8 질의문을 사용해 해당 도서 정보를 가져온다.
* 도서명을 가진 정보가 1개라면 S1-1를 사용해 해당 도서 정보를 가져온다.
* 학생 ID를 입력 받고 S4-2 질의문을 사용해 예약 정보를 조회한다.
* 예약 정보가 없다면 S4-1 변경문을 사용해 예약 정보를 추가한다.

1. T3-4: 연체료 업데이트 및 확인

* 학생 ID를 입력 받고 S6-1, S6-2 질의문을 사용해 반납 일자가 지난 대출 정보를 가져온다.
* 현재 날짜를 기준으로 연체료를 계산해 S6-3 변경문으로 각 대출 정보의 연체료 값을 수정한다.

1. T3-5: 연체료 정산:

* 학생 ID를 입력 받고 S6-1, S6-2 질의문을 사용해 반납 일자가 지난 대출 정보를 가져온 후 Pagination하여 연체료와 함께 보여준다.
* 계산하기 원하는 대출 ID를 입력 받고 S6-3 변경문을 사용해 연체료 정보를 0으로 수정한다.

**5. 소스 코드 설명**

**1) Main.java**

|  |  |
| --- | --- |
| **코드** | **설명** |
|  | Main() 시작  각 옵션 번호를 입력하면 switch문으로 해당되는 트랜잭션을 실행한다. |
|  | Case1: 도서  검색  Case2: 대출  Case3: 예약  Case4: 반납 |
|  | Case5: Mypage   * 0을 입력 받기 전까지 실행 * Option을 입력 받아 switch문에 사용한다. * Case1:   대출 리스트   * Case2:   예약 리스트   * Case3:   연체료 조회   * Case4:   연체료 계산 |
|  | Case 6:  Management   * 0을 입력 받기 전까지 실행한다.. * 도서목록을 관리할 수 있다. * Option을 입력 받아 switch문에 사용한다. * Case1:   도서 추가   * Case2:   도서 정보 수정   * Case3:   도서 삭제 |

**2) Transaction.java**

- 모든 트랜잭션 함수의 시작 부분에 con.setAutoCommit(**false**);을 실행하여 트랜잭션의 시작임을 알린다. 또한, pstmt = con.prepareStatement(s2\_2);, pstmt.setInt(4, book\_num);, pstmt.executeQuery();, pstmt.executeUpdate();를 사용해 동적 SQL문을 사용한다. 트랜잭션 마지막에 con.commit();을 실행해 변경 사항을 데이터베이스에 반영한다. 트랜잭션이 중단되어야 할 때 예외를 발생시켜 con.rollback();을 실행한다.

|  |  |
| --- | --- |
| **코드** | **설명** |
|  | Connection, PreparedStatement, ResultSet, 동적SQL문을 미리 생성해두었다. |
|  | 도서 목록 pagination |
|  | 대출 목록 Pagination |
|  | 예약 목록 Pagination |
|  | 연체정보 Pagination |
|  | T1-1. 도서검색  main에서 입력받은 option  option=1:  도서명 검색  option=2:  출판사 검색  option=3  저자 검색  option=4  도서 번호 검색  검색 정보 크기가  0이면 예외 발생 시킴  검색 후  Result Set을  PrintBookList() 사용해  입력받은  페이지에 따라 보여준다. |
|  | T1-2.  대출 목록 조회  입력받은 학생 ID를 가지는 대출 정보를 검색  개수 0이면  예외 발생 시킴  ResultSet을 PrintCheckedList()를 사용해  입력받은  Page에 따라  보여준다. |
|  | T1-3.  예약 목록 조회  입력 받은 학생 ID를 가지는 예약 정보 조회.  개수 0이면 예외 발생시킴  ResultSet을  PrintReserveList()를 사용해  원하는 Page에 따라 보여준다. |
|  | T2-1.  도서 추가  도서번호,  도서명,  출판사,  저자를  입력 받은 후  업데이트 실행 |
|  | T2-2  도서 삭제  도서 번호를  입력받아  해당되는  정보  삭제 실행.  영향 받은 row가 없다면 예외 발생 시킴 |
|  | T2-3.  도서 반납  도서번호, 학생ID를 입력 받고  도서정보의 대출가능 여부 수정  대출정보의 반납일자 수정  각 수정에 대해  영향 받은  row가 없다면 예외발생 |
|  | T3-1.  도서정보 수정  도서 번호를  입력 받아  도서 정보  검색 후 출력  수정하기  원하는 컬럼을 선택  새로운 값을  입력 받고  업데이트 수행. |
|  | T2-2. 대출  도서명 입력 받음  정보 개수가 0이면 예외 발생  정보가 1개보다 많다면 PrintBookList()를 사용해 원하는 페이지의 정보를 보여주고,  원하는 도서번호를 입력 받아 ResultSet 검색.  1개라면 도서명으로 ResultSet검색  학생 ID를  입력 받고  앞의 ResultSet에서 얻은  도서 번호로  예약정보 검색  예약 정보가  있다면  삭제  대출 정보  업데이트  도서 정보의 can\_borrow 값  False로  업데이트 |
|  | T3-3. 예약  도서명 입력 받음  정보 개수가 0이면 예외 발생  정보가 1개보다 많다면 PrintBookList()를 사용해 원하는 페이지의 정보를 보여주고,  원하는 도서번호를 입력 받아 ResultSet 검색.  1개라면 도서명으로 ResultSet검색  학생 ID를  입력 받고  앞의 ResultSet에서 얻은  도서 번호로  예약정보 검색  예약 정보가  이미  존재한다면  예외 발생  예약 정보 업데이트 |
|  | T3-4.  연체료 업데이트 및 조회  메인에서  입력 받은  학생ID로  반납 일자가  지난  대출 정보 조회  결과Set으로  현재 일자를  기준으로  연체료 계산,  업데이트. |
|  | T3-5  연체료 조회  및 정산  메인에서  입력 받은  학생 ID로  대출 정보 조회  ResultSet을  printOverdueList()를 통해  원하는 페이지를 보여줌.  계산을 원하는 대출번호를  입력 받고  해당 번호의  Overduefee=0으로 설정 |

**3) 구현요건 충족에 대한 상세 설명**

(1) 검색only 트랜잭션, 변경 트랜잭션, 혼합 트랜잭션은 위 4-2) 트랜잭션 리스트와 5-2) 소스코드 설명에서 확인할 수 있듯 2개 이상 포함되었다. (혼합 트랜잭션: T3-4, T3-5)

(2) (b), (e)를 만족하는 혼합 트랜잭션

1. T3-1

|  |  |  |
| --- | --- | --- |
| 동적SQL문 | 소스코드 | 설명 |
| S1-4  질의문 |  | 사용자로부터 도서번호를  입력받아  S1-4질의문의 값으로  할당하였다. |
| S2-2  변경문 |  | 새로 입력 받은 한 컬럼을 제외하고  위의 질의문에서 얻은 값들을 S2-2 변경문 값으로  할당하였다. |

1. T3-3

|  |  |  |
| --- | --- | --- |
| 동적SQL문 | 소스코드 | 설명 |
| S1-1  질의문 |  | 사용자로부터 도서명를 입력 받아  S1-1 질의문의 값으로  할당하였다. |
| S4-1  변경문 |  | 위에서 얻은 book\_num 값을 S4-1 변경문의 값으로 할당하였다.. |

(3) (a)~(e)를 모두 만족하는 혼합 트랜잭션: T3-2

|  |  |  |  |
| --- | --- | --- | --- |
|  | 동적SQL문 | 소스코드 | 설명 |
| (a) | S1-5 |  | 2개 이상의 질의문을  사용하였다.. |
| S1-1 |  |
| S1-8 |  |
| S1-4 |  |
| S4-2 |  |
| (b) | S1-5 |  | 사용자에게 입력받은  도서명을 S1-5질의문 값에  할당하였다. |
| (c) | S1-1  S4-2 |  | S1-1에서 얻은 book\_num 값을  S4-2에  할당해  사용하였다. |
| (d) | S1-1  S4-2 |  | S1-1에서  얻은  Publisher,  author값과  S4-2에서  얻은  Reserve\_id,  Reserve\_date값을  조합해  출력하였다. |
| (e) | S1-1  S2-3  S2-4 |  | S1-1에서  얻은  Book\_num  값을  S2-3,  S2-4  변경문  값에  할당하였다. |

**6. 테스트 레코드 생성**

**1) 랜덤 데이터 생성**: 완전 랜덤한 값을 생성한 레포트3과 달리, 데이터 검색에 용이하게 값을 생성해주었다.

|  |  |
| --- | --- |
| **코드** | **설명** |
|  | 입력받은 String배열을 담은 CSV 파일을 생성하는 함수 |
|  | 입력받은 length+5 길이의 랜덤 영문 문자열 생성 함수 |
|  | 0~(range-1) 범위의 랜덤 정수를 생성하는 함수 |
|  | 입력받은 길이의 랜덤 정수를 생성하는 함수 |
|  | (Start)개월 전부터  (end) 개월 후까지의  “2020-\*\*-\*\* \*\*:\*\*:\*”  형식으로  랜덤 Datetime을 생성하는  함수 |

- Student.csv 생성

|  |  |
| --- | --- |
| Student  Table의  데이터  생성 |  |
| 설명 | 10만개 데이터 생성  ID: 길이 4인 랜덤 인트 + i값 5자리  ID 생성 후 다른 Table의 외래키로 사용하기 위해 배열에 저장  Name: 길이 15이하의 랜덤 문자열 |
| 데이터  생성  결과  (일부) | A: IDInt), 학생 번호  B: *name*(char), 학생 이름 |

- BookList.csv 생성

|  |  |
| --- | --- |
| BookList  Table의  데이터 생성 |  |
| 설명 | 10만개의 데이터 생성  Book\_num: 길이 4인 랜덤 정수 + i값을 길이 5로 채워서 생성  Book\_num은 다른 Table에서 외래키로 사용하므로 배열에 저장해둠  Book\_name, Publisher, Author: 위와 같이 문자열을 생성해둔 후 랜덤하게 사용  Can\_borrow: 0 또는 1로 생성 |
| 데이터 생성  결과  (일부) | A: *book\_num*(Int), 도서 고유번호  B: *bookname*(Varchar), 도서명  C: *publisher(*Varchar), 출판사  D: *author*(Varchar), 작가  E: *can\_borrow*(Boolean), 현재 대출 가능한지 여부 |

- CheckedList.csv 생성

|  |  |
| --- | --- |
| CheckedList  Table의  데이터 생성 |  |
| 설명 | 25만개의 데이터 생성  Checker\_id: Student 생성에 사용한 값을 사용  Book\_num: BookList 생성에 사용한 값을 사용  Check\_date: 6달 전부터 지금까지 랜덤한 날짜 생성  Retrun\_date: Check\_date 이후의 랜덤 날짜나 Null로 생성  Overdue\_fee: 0으로 초기화 |
| 데이터 생성  결과  (일부) | A: *id*(Int), PK  B: *checker\_id*(Int), 대출자 id  C: *book\_num*(Int), 대출도서 번호, 외래키  D: *check\_date*(Datetime), 대출 날짜  E: *return\_date*(Datetime), 반납 날짜  F: *overdue\_fee*(Int), 연체료 |

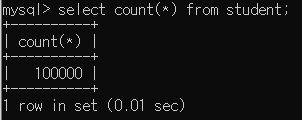
- ReserveList.csv 생성

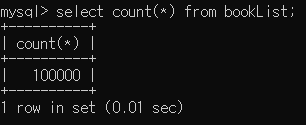
|  |  |
| --- | --- |
| ReserveList  Table의  데이터  생성 |  |
| 설명 | 25만개의 데이터 생성  reserver\_id: Student 생성에 사용한 값을 사용  Book\_num: BookList 생성에 사용한 값을 사용  Reserve\_date: 4달 전부터 지금까지 랜덤한 날짜 생성 |
| 데이터 생성  결과  (일부) | A: *id*(Int), PK  B: *reserver\_id*(Int), 예약자 id  C: *book\_num*(Int), 대출도서 번호, 외래키  D: *reserve\_date*(Datetime), 예약 날짜 |

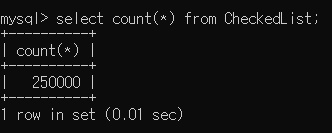
**2) 데이터 로드**: 아래와 같이 JDBC 코드를 통해 데이터를 로드했다.

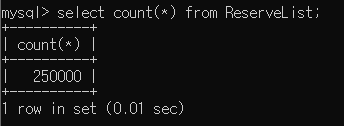
|  |
| --- |
|  |

**3) 로드 결과**: 아래와 같이 Select질의문을 통해 데이터가 로드된 것을 확인하였다.





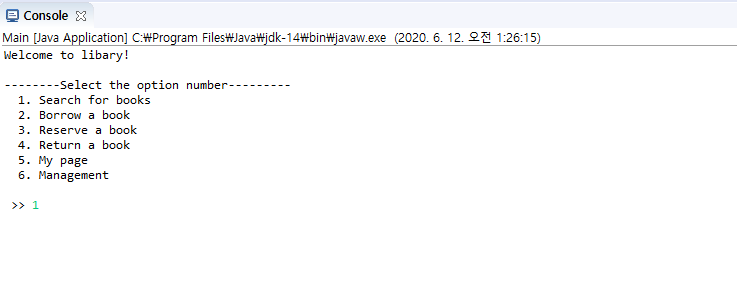




**7. 응용 실행 데모**

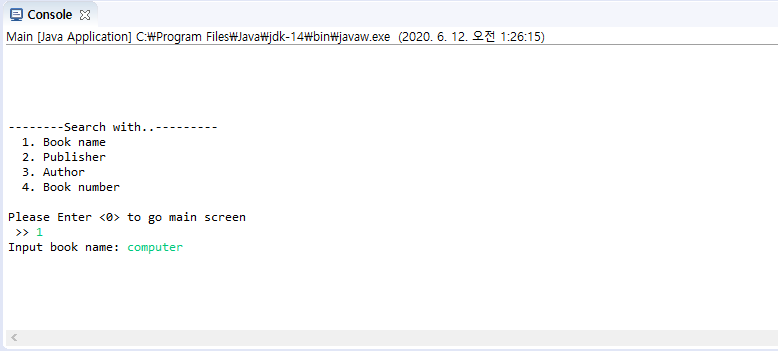
**1) 도서 검색**

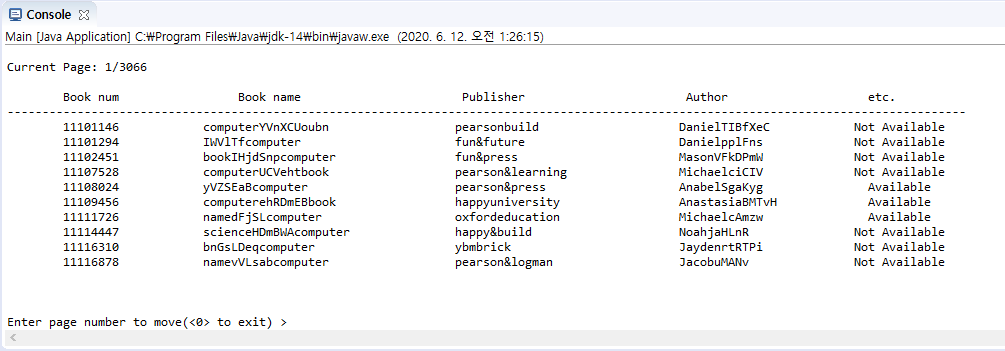
메인 화면에서 1을 입력하면 도서 검색이 가능하다.



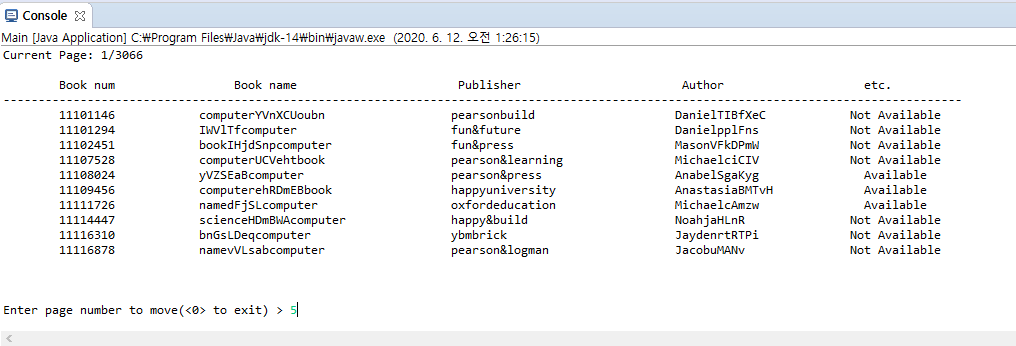
1. 도서명으로 검색

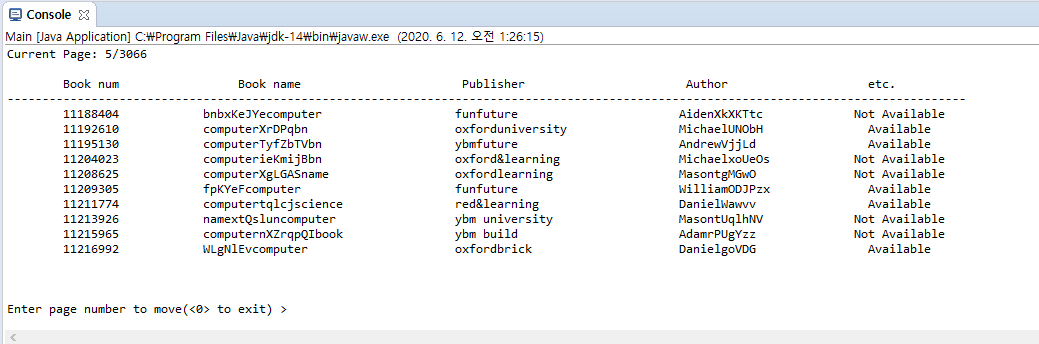
1번을 선택한 후 도서명을 입력하면 입력 받은 값을 포함하는 모든 레코드를 볼 수 있다.





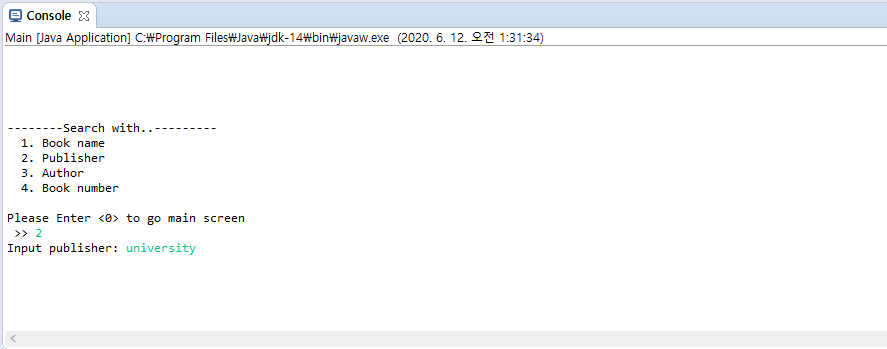
아래에 원하는 페이지 번호를 입력하면 다음과 같이 해당 페이지의 값들을 볼 수 있다.





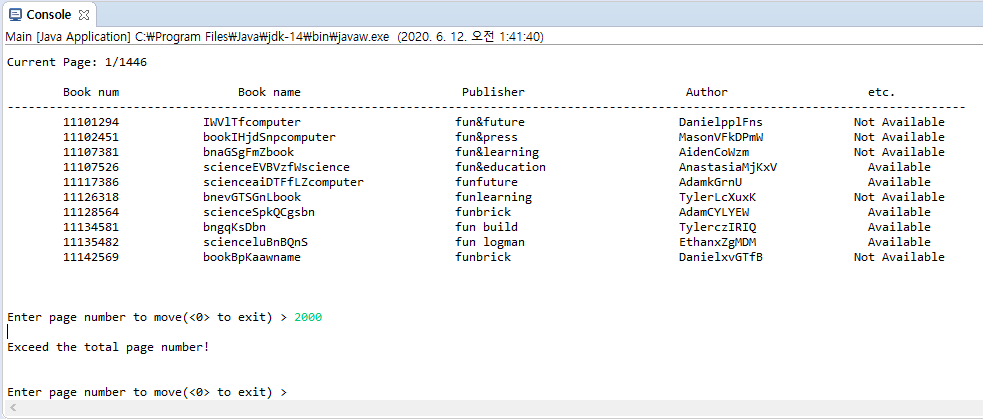
1. 출판사로 검색

2번을 선택한 후 검색할 값을 입력하면 아래와 같이 값을 포함하는 모든 레코드를 볼 수 있다.



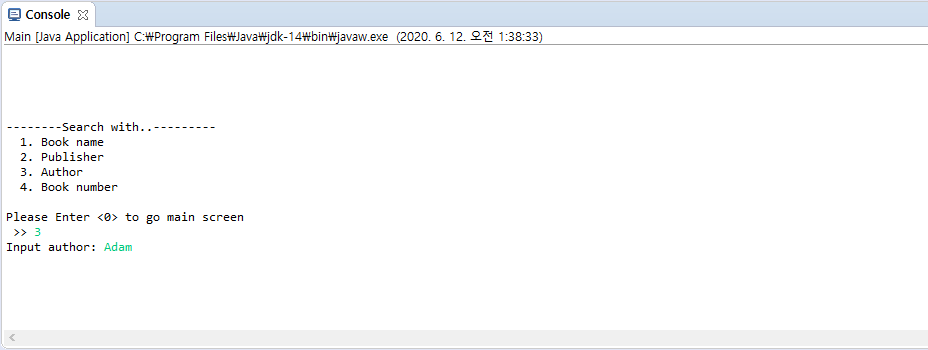


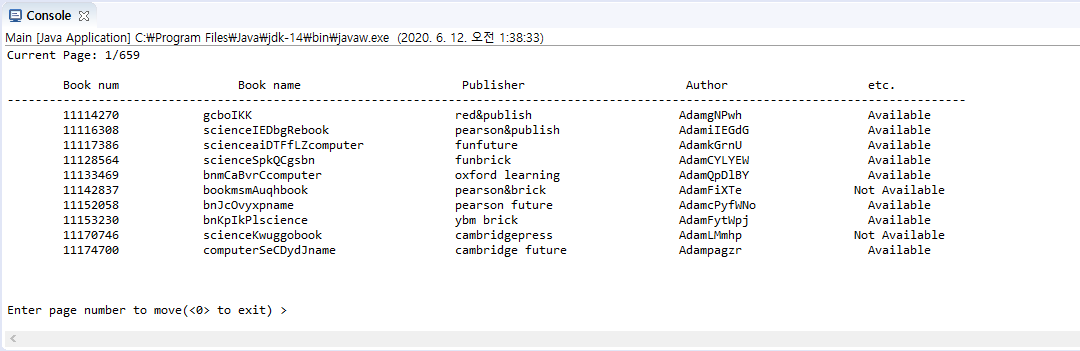
또한 pagination에서 전체 페이지 크기보다 큰 값을 입력하면 아래와 같이 문구가 출력된다.



1. 저자로 검색

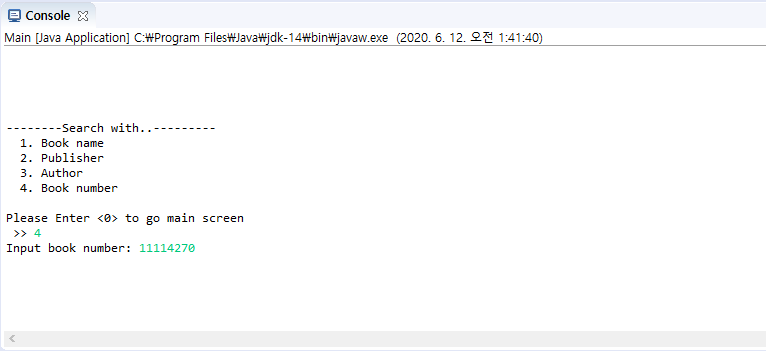
3번을 선택한 후 검색하기 원하는 값을 입력하면 해당 값을 포함하는 레코드들을 모두 볼 수 있다.

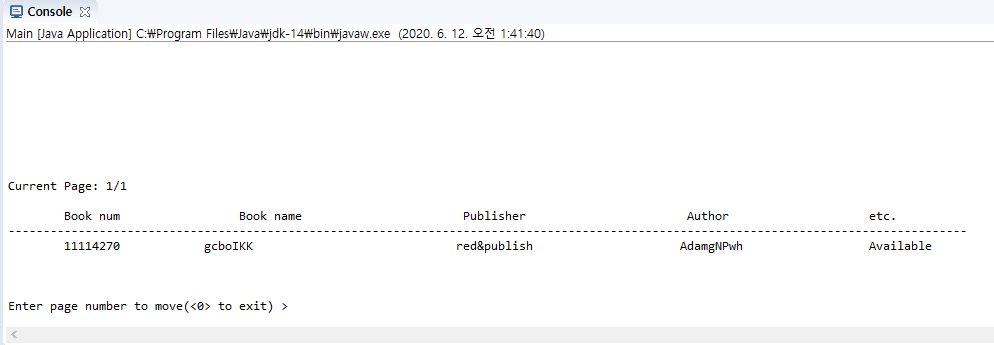




1. 도서 번호로 검색

4번을 선택한 후 도서 번호를 입력하면 해당 번호를 가진 도서 정보를 볼 수 있다.

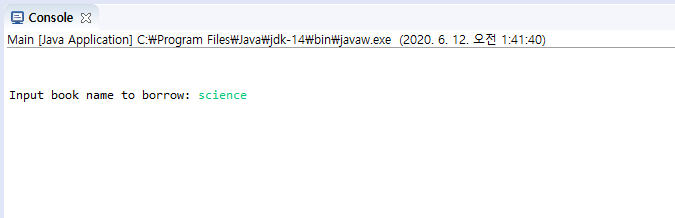
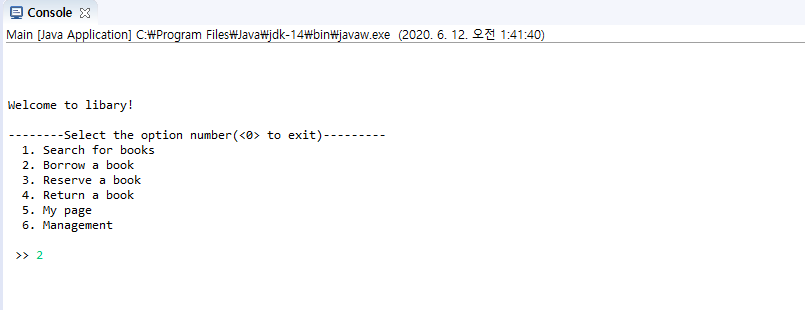




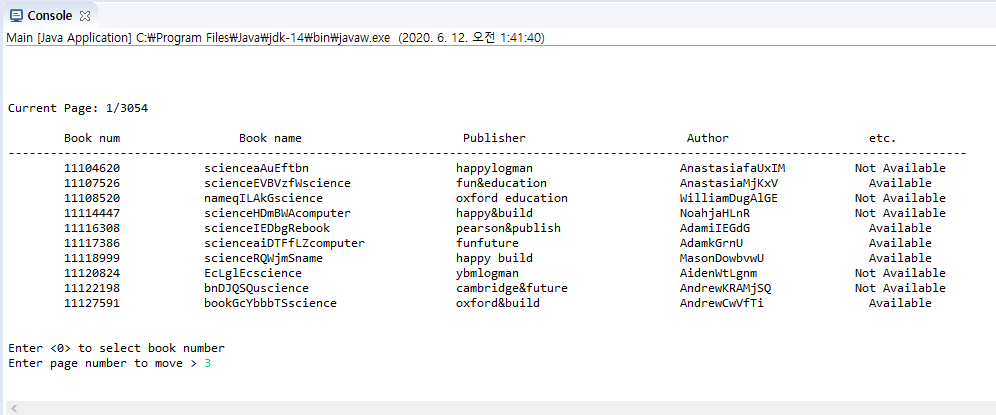
**2) 대출, 예약, 반납**

1. 대출

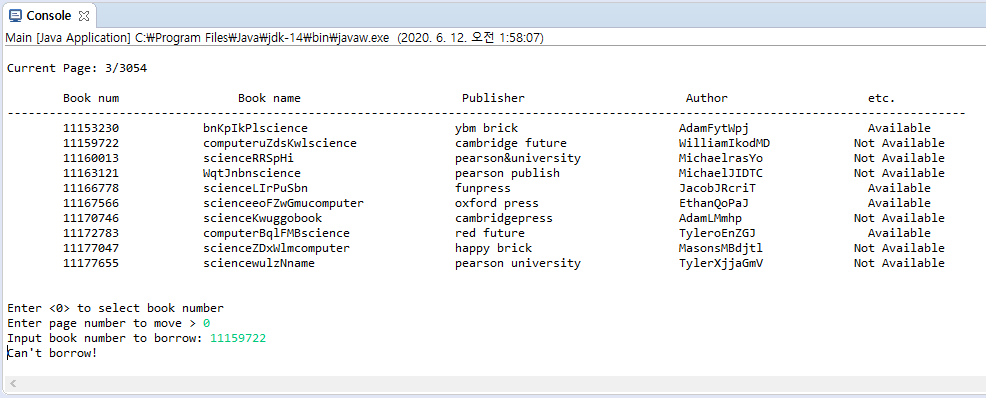
메인 화면에서 2번을 선택한 후 대출하길 원하는 도서명을 입력한다.



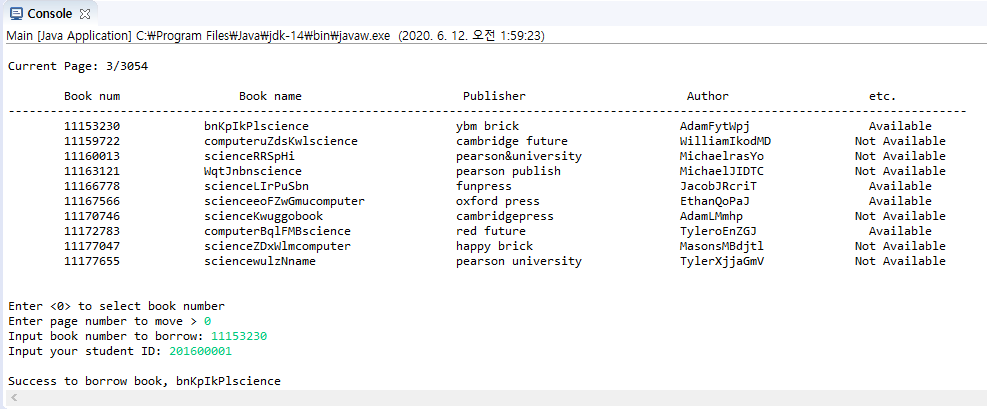
해당 값을 포함하는 도서 정보가 여러 개라면, 아래와 같이 레코드들을 페이지별로 확인할 수 있다.



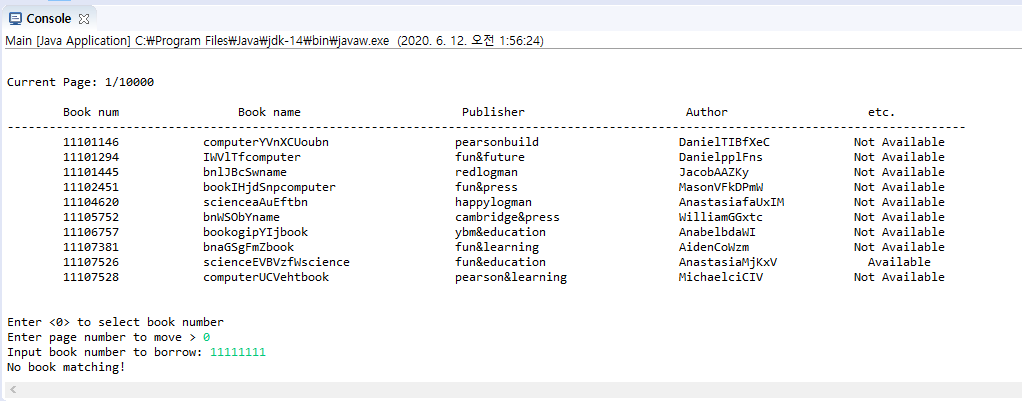
목록에서 도서 번호를 확인한 후 대출할 도서 번호를 아래와 같이 입력한다.



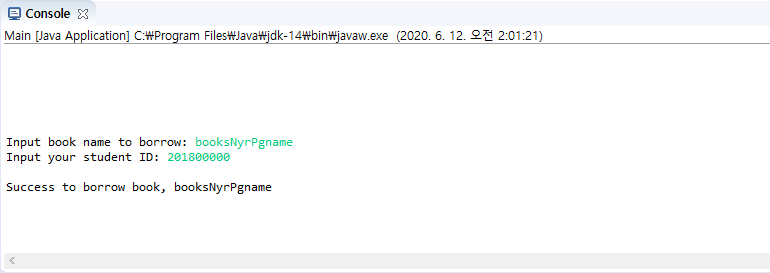
학생 ID를 입력하면 성공적으로 도서를 대출했다는 메시지가 출력된다.



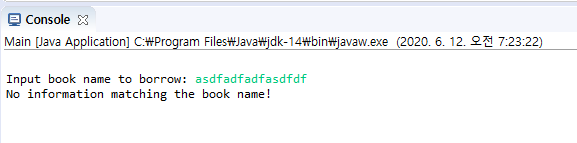
만약 도서 번호를 가진 도서가 존재하지 않는다면 아래와 같이 에러 메시지가 출력된다.



만약 입력 받은 도서명을 가진 정보의 개수가 1개라면 바로 학생 ID를 입력 받은 후 대출할 수 있다.

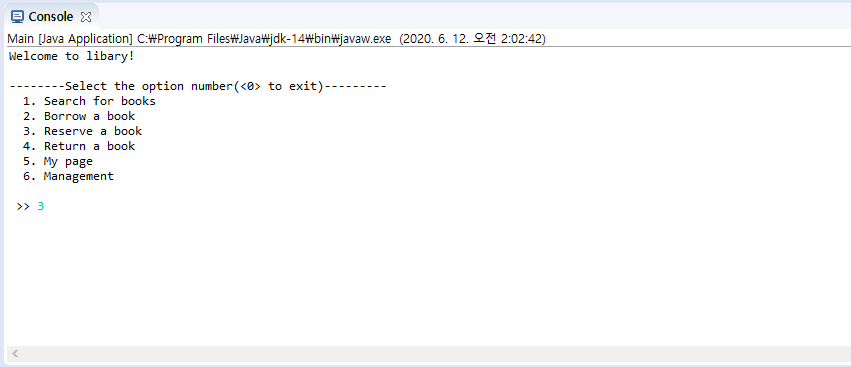


도서명을 가진 정보가 존재하지 않는다면 아래와 같이 에러 메시지가 출력된 후 종료된다.



1. 예약

메인 화면에서 3번을 선택한 후 예약을 원하는 도서명을 입력한다.





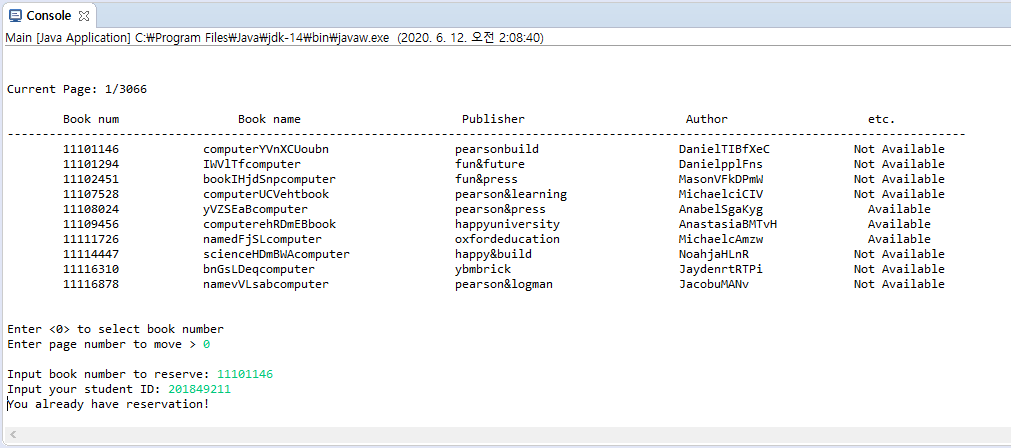
도서명을 가진 도서 정보의 개수가 1개보다 많다면 아래와 같이 해당 레코드들을 페이지로 볼 수 있다.



목록에서 도서 번호를 확인한 후 예약을 원하는 도서의 번호를 입력한다. 이어서 학생 ID를 입력한 후 예약을 할 수 있다.

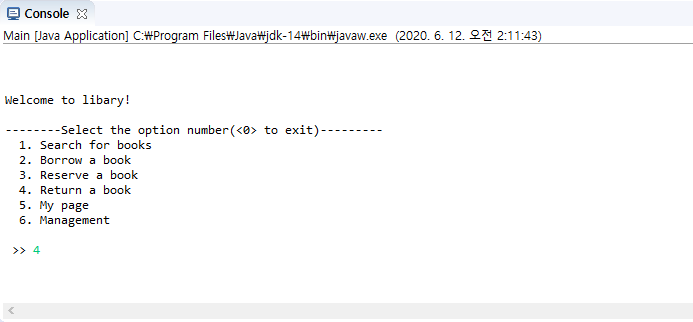


만약 해당 도서번호, 예약자 ID값을 갖는 예약 정보가 이미 있다면 아래와 같이 에러 메시지가 출력된다.

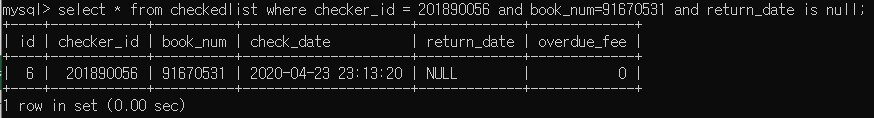


1. 반납

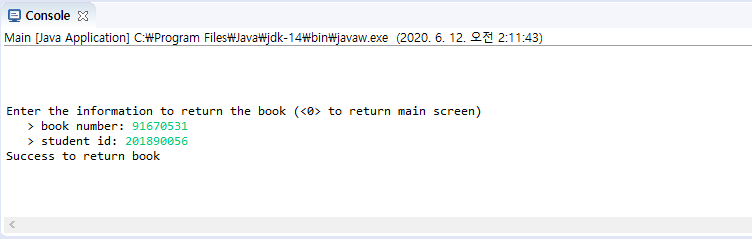
메인 화면에서 4번을 선택하면 반납을 진행할 수 있다.



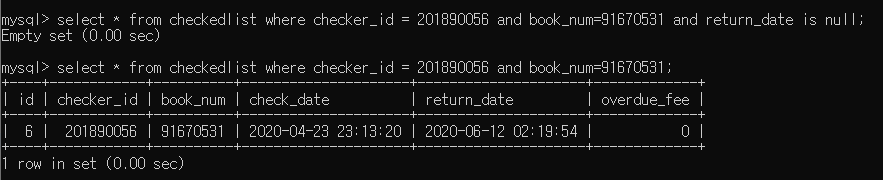
아래는 반납 기능을 실행하기 전에 대화식SQL 도구를 통해 확인한 값이다.



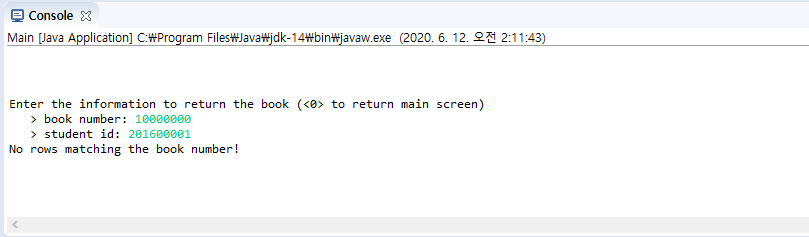
아래와 같이 도서 번호와 학생 ID를 입력하면 반납이 완료된다.



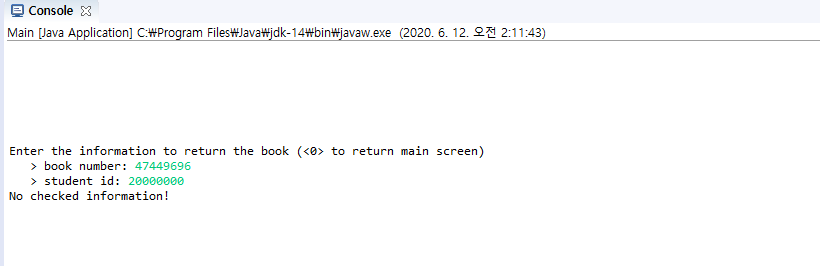
반납 기능을 실행한 후 return\_date에 실행 당시 시간이 추가된 것을 확인할 수 있다.



만약 도서 번호가 존재하지 않는다면 아래와 같이 메시지가 출력된 후 종료된다.

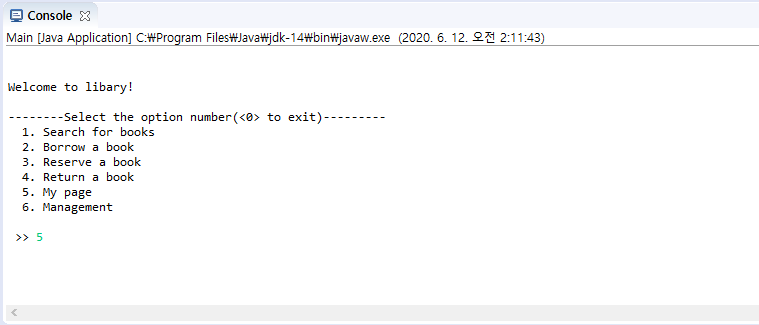


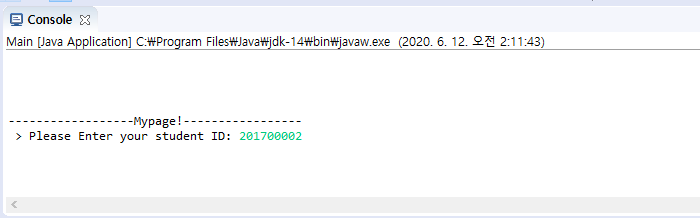
대출 정보가 존재하지 않는다면 아래와 같이 메시지가 출력된다.



**3) 마이페이지**

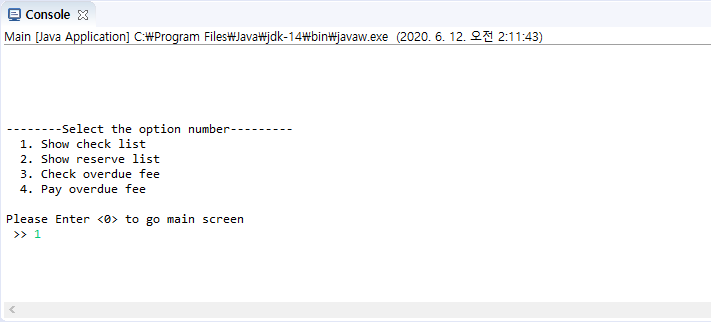
메인 화면에서 5번을 선택하고, 학생 ID를 입력하면 마이페이지 기능을 이용할 수 있다.

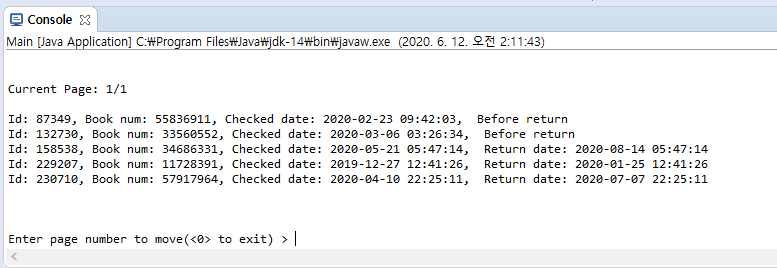




1. 대출 정보 조회

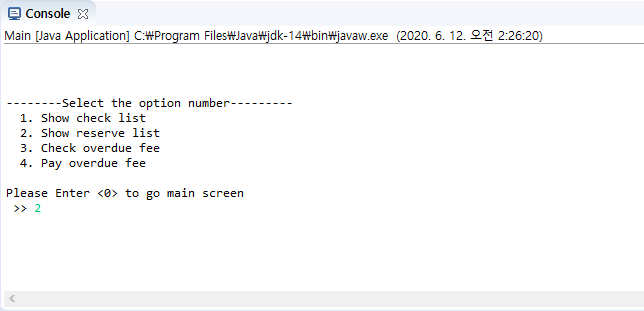
1번을 선택해 자신의 대출 정보를 조회할 수 있다.

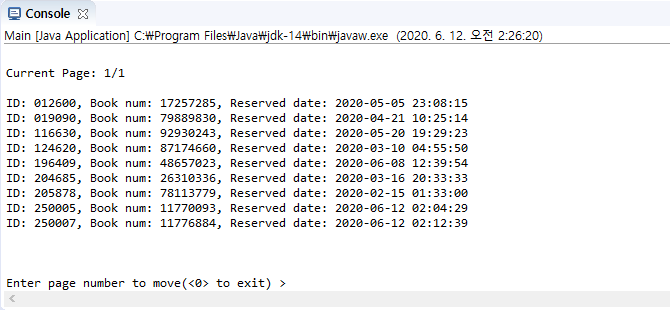




1. 예약 정보 조회

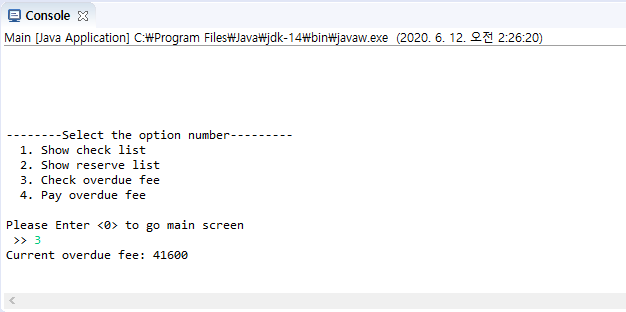
2번을 선택해 자신의 예약 정보를 조회할 수 있다.



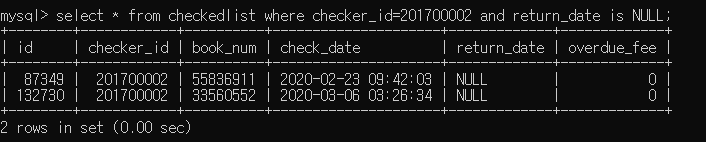


1. 연체료 업데이트 및 조회

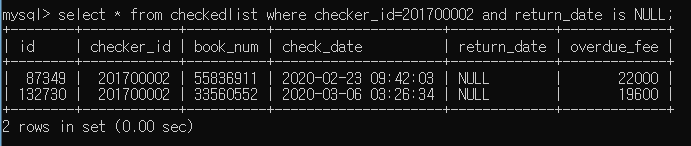
3번을 선택하면 아래와 같이 해당 시점까지의 자신의 전체 연체료를 확인할 수 있다.



아래는 기능을 사용하기 전이고,

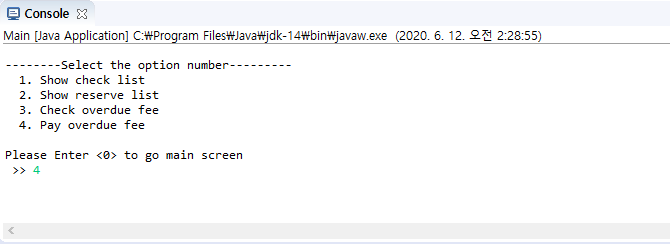


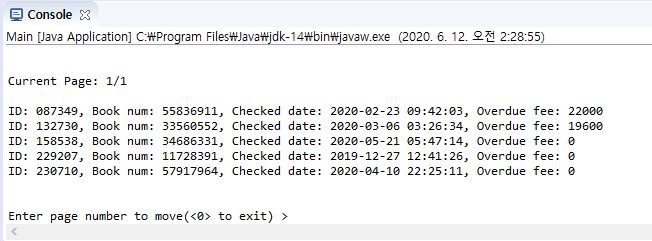
기능을 사용한 후 연체료 정보가 업데이트된 것을 확인할 수 있다.



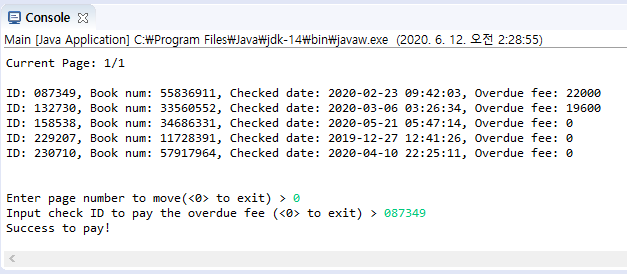
1. 연체 정보 조회 및 정산

4번을 선택하면 자신의 대출 정보와 함께 연체 정보를 확인할 수 있다.

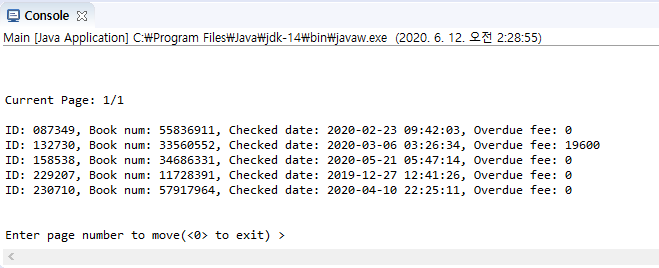




아래와 같이 정산하길 원하는 대출 번호를 입력하면 해당 연체료를 정산할 수 있다.

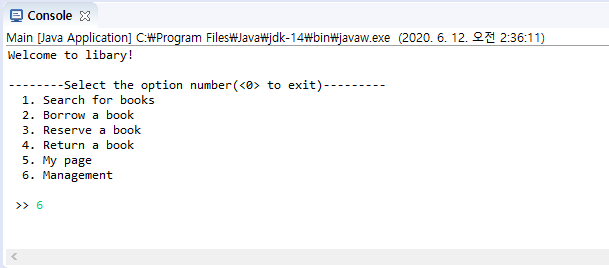


다시 4번을 선택해 정보를 조회해보면 값이 0으로 바뀐 것을 확인할 수 있다.



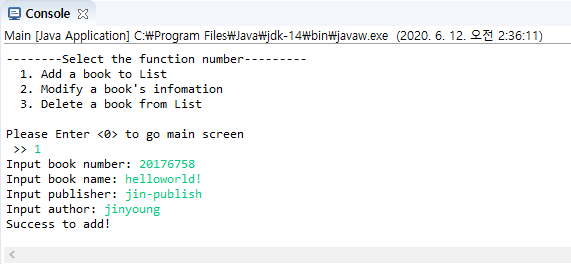
**4) 관리**

메인 화면에서 6번을 선택하면 관리 메뉴를 사용할 수 있다.

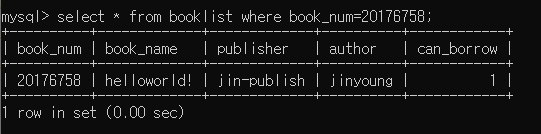


1. 도서 정보 추가

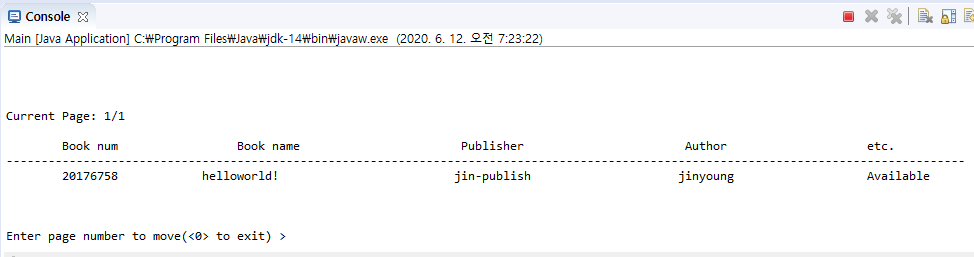
1번을 선택하고, 도서 번호, 도서명, 출판사, 저자를 입력하면 도서 정보를 추가할 수 있다.



아래는 추가한 정보를 대화식SQL도구에서 확인한 결과이다.

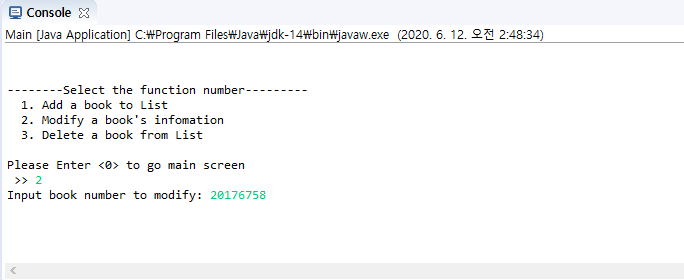


다음과 같이 응용 기능을 통해서도 확인할 수 있다.

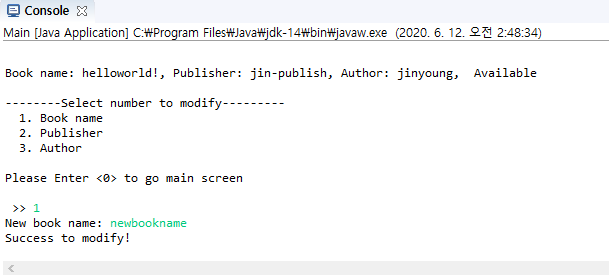


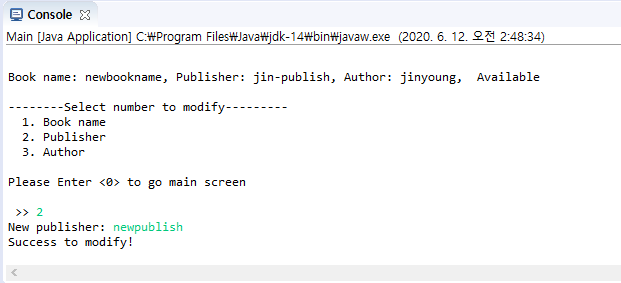
1. 도서 정보 수정

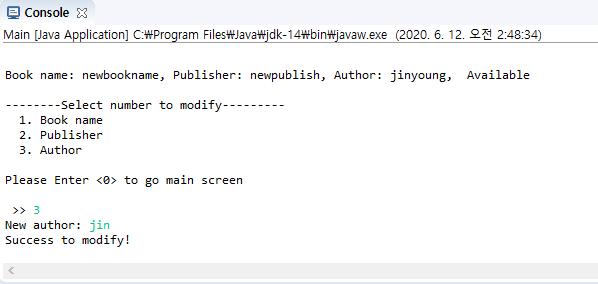
2번을 선택하고 수정하길 원하는 도서 번호를 입력한다.



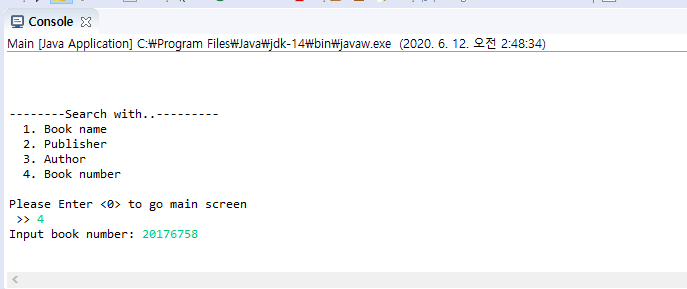
다음과 같이 도서 번호에 해당되는 도서 정보를 확인할 수 있다. 수정하길 원하는 옵션 번호를 입력한 후 새로운 값을 입력하면 수정이 완료된다.

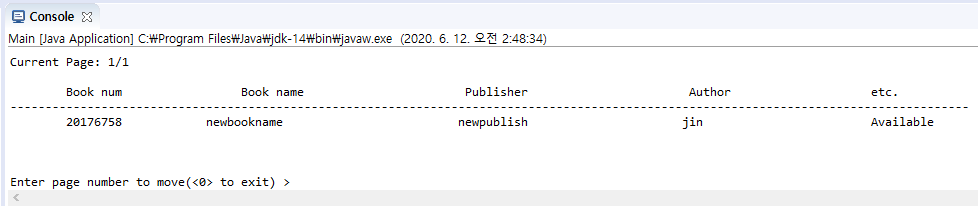






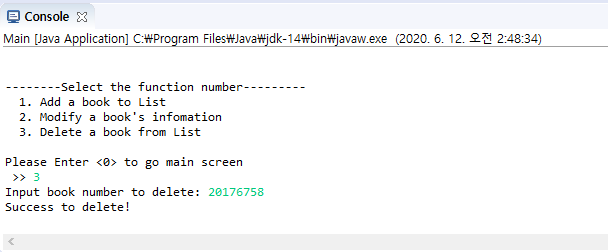
아래와 같이 응용의 메인 1번 기능을 통해 수정된 값을 확인할 수 있다.





1. 도서 정보 삭제

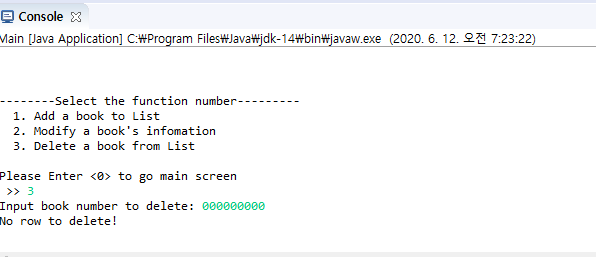
3번을 선택한 후 삭제할 도서의 번호를 입력하면 성공적으로 삭제가 완료된다.



다음은 대화식SQL도구에서 삭제된 결과를 확인한 캡처 화면이다.



아래와 같이 존재하지 않는 도서번호를 입력하면 에러 메시지가 출력된 후 종료된다.



**8. 소스 코드 실행 관련 필요사항**

- Mysql driver 파일(.jar)을 빌드 패스 라이브러리에 추가한다.

- Createtable.java를 실행해 테스트 데이터를 생성한다.(경로 수정 필요)

-> 사용한 테스트 데이터는 폴더에 첨부 (TestData 폴더)

- MysqlLoadDataInfile.java를 실행해 데이터를 테이블에 로드한다.(경로 수정 필요)

- Transaction.java에서 url, id, pw를 변경해준다.

- Main.java의 main()함수로 실행한다.

\*제출 폴더\*

Java 파일들은 DBProject 폴더 안에 존재.

Createtable.java, MysqlLoadDateInfile.java는 LoadData 폴더(패키지) 안에 존재

Main.java, Transaction.java는 Library 폴더(패키지) 안에 존재

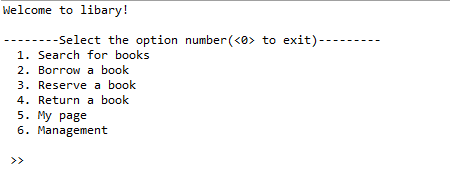
TestData 폴더 안 프로젝트에 사용한 테스트 데이터 파일(.csv) 존재

CreateTable.txt : 테이블 생성 구문

레포트는 word파일과 pdf파일 두 버전.

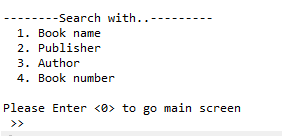
**9. 사용자 매뉴얼**

**0)**

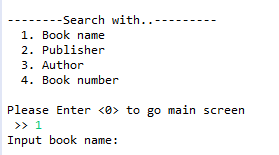
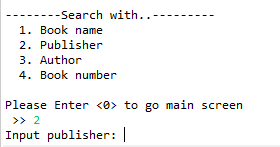


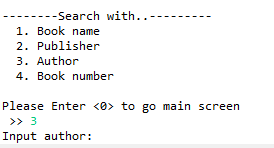
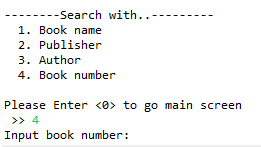
* **메인 화면. 원하는 옵션의 번호를 입력한다. 0 입력 시 종료**

**1)**

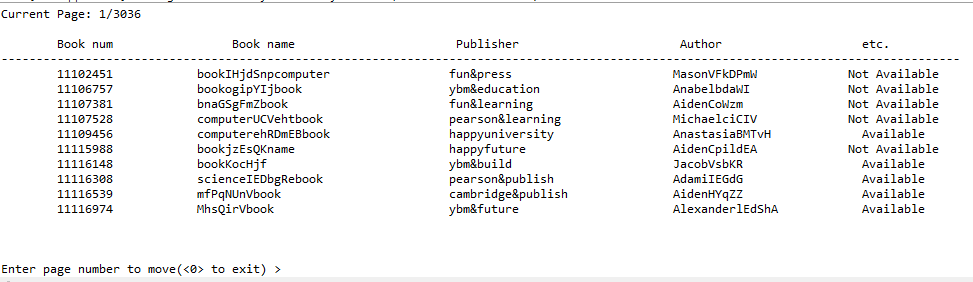


* **메인 1번. 검색을 원하는 옵션 번호를 입력한다. 0 입력 시 메인 화면으로 돌아간다.**

* **메인 1번. 옵션 선택 시 검색 값을 입력할 수 있다.**

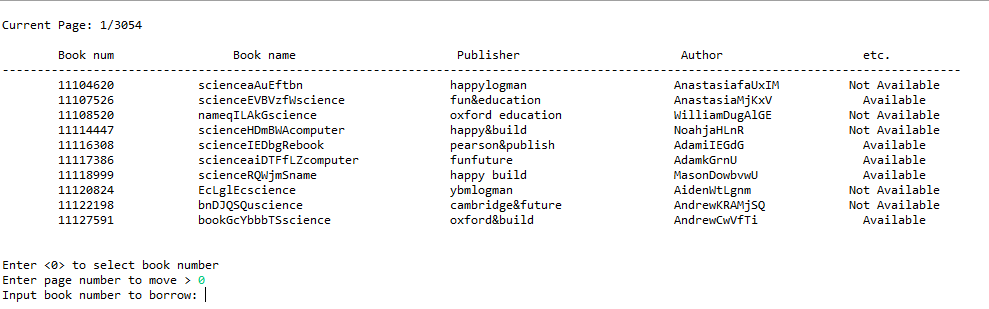


* **메인 1번. 검색 값 입력 시 다음과 같은 목록을 볼 수 있고, 페이지를 입력해 원하는 페이지로 이동할 수 있다. 0 입력 시 메인 1번으로 돌아간다.**

**2)**



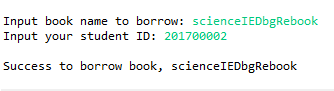
* **메인 2번. 빌리기 원하는 책 이름을 입력한다.**



* **메인 2번. 입력한 책 이름이 여러 개일 경우 목록으로 확인할 수 있다. 0을 입력해 빌릴 책의 번호를 입력할 수 있다.**



* **메인 2번. 책의 번호와 학생 ID를 입력하면 대출할 수 있다.**



* **메인 2번. 입력한 책 이름이 1개일 경우 바로 학생 ID를 입력해 대출할 수 있다.**

**3)**

* **메인 3번. 메인 2번과 동일하게 사용한다.**

**4)**

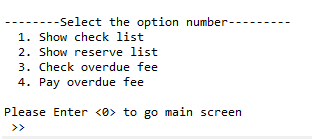


* **메인 4번. 반납할 책의 번호와 학생 ID를 입력해 책을 반납한다. 0을 입력하면 메인 화면으로 돌아간다.**

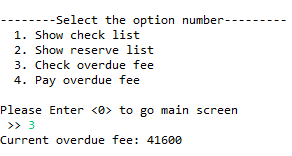
**5)**



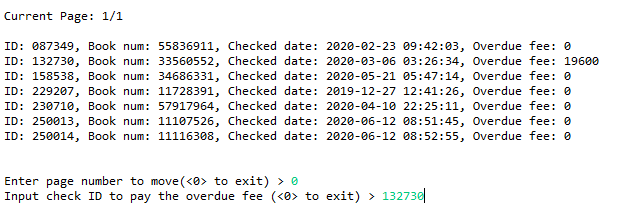
* **메인 5번. 학생 ID를 입력해 마이 페이지에 들어갈 수 있다.**



* **메인 5번. 옵션을 선택해 해당 기능을 이용할 수 있다. 0을 선택하면 메인 화면으로 돌아간다.**
* **메인 5번, 옵션 1,2는 메인 1번과 같이 각 목록을 페이지로 조회할 수 있다.**

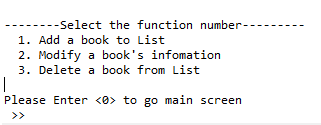


* **메인 5번. 옵션 3을 선택하면 연체료를 확인할 수 있다.**

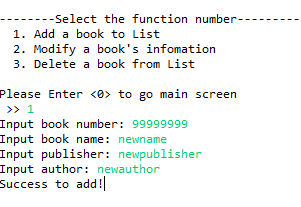


* **메인 5번. 옵션 4를 선택하면 목록을 확인할 수 있고 원하는 ID를 입력하여 정산한다.**

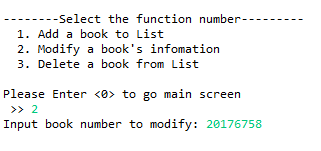
**6)**



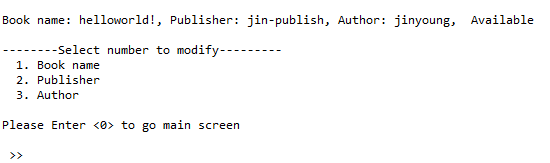
* **메인 6번. 옵션 번호를 선택해 도서 목록 관리 기능을 사용할 수 있다. 0을 선택하면 메인 화면으로 돌아간다.**



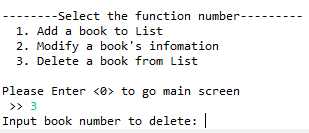
* **메인 6번. 옵션 1번을 선택해 정보를 입력하여 도서를 추가할 수 있다.**



* **메인 6번. 옵션 2번을 선택해 수정하길 원하는 도서 번호를 입력한다.**



* **메인 6번. 옵션 2번(이어서) 정보를 확인하고 수정하길 원하는 옵션을 선택해 수정할 수 있다.**



* **메인 6번. 옵션 3번을 선택, 도서 번호를 입력한 후 삭제할 수 있다.**