TeamProject(Iteration#3)

3팀

|  |  |  |
| --- | --- | --- |
| 학번 | 성명 | 역할 |
| 2020315024 | 이용우 | 팀원 |
| 2021320033 | 황주원 | 팀원 |
| 2020210002 | 김무원 | **팀장** |
| 2022320029 | 이상민 | 팀원 |

목차

1. Use Case Diagram Iteration #1 ~ #3

(p. 3 ~ 7)

1. Use Case Description Iteration #1 ~ #3

(p. 8 ~ 28)

1. Sequence Diagram Iteration #1 ~ #3

(p. 29 ~ 49)

1. Class Diagram Iteration #1 ~ #3

(p. 50 ~ 54)

5. 버전 관리 및 Java Code

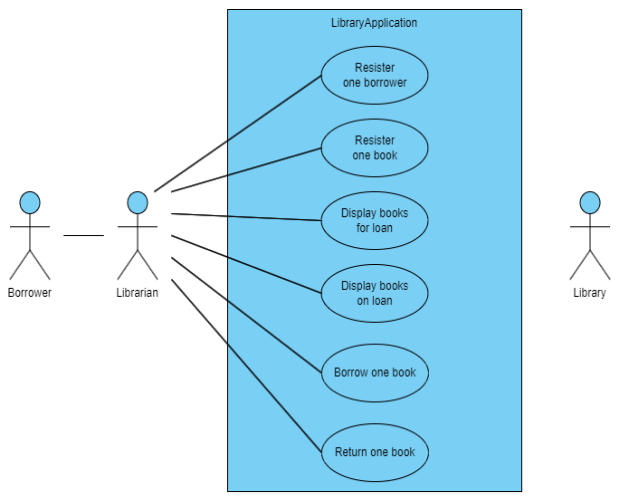
(p. 55)

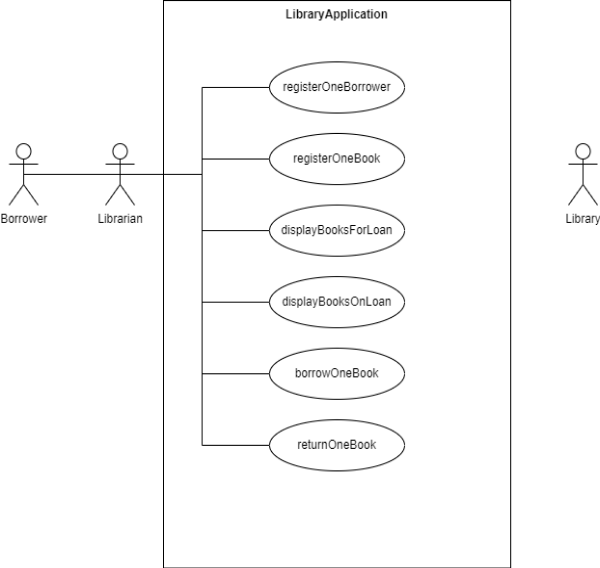
**Use Case Diagram Iteration #1 ~ #3**

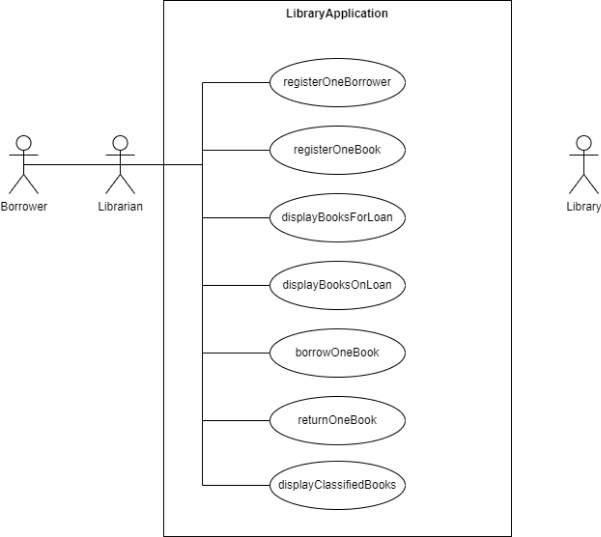
변경사항

1. 툴 변경 : Visiual Paradigm Online -> draw.io

변경 사유 : 실시간 작업이 가능하며, 버전관리가 용이하다.

**Use Case Diagram Iteration #1**

**Use Case Diagram Iteration #2**

**Use Case Diagram Iteration #3**

**Use Case Description Iteration #1 ~ #3**

변경사항

1. 더 명확한 표현을 위해 None -> N/A 로 수정함.

2. Iteration#2 displayBooksForLoan 의 System Response 2번 부분 숫자 표기누락 -> Iteration#3 displayBooksForLoan에 오류를 수정하기 위해 추가함.

3. c\_num, u\_name등의 간략한 변수명을 보는 사람이 알아보기 쉽게 하기 위해 낙타등 표기법에 따라 수정함.

**Use Case Description Iteration #1 <1> registerOneBorrower**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | registerOneBorrower | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower | | |
| 개요 | 등록되지 않은 Borrower가 Librarian에게 이용자 등록을 신청한다. | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1.등록되지 않은 Borrower가 이용자 등록을 위해 이용자 등록신청서(u\_name)를 작성하고 Librarian에게 제출한다. | | |  |
| 2.Librarian은 Borrower에게 받은 이용자 등록신청서 (u\_name)를 바탕으로 Borrower의 정보를 시스템에 등록한다. | | |  |
|  | | | 3.u\_name을 담은 Borrower 객체를 생성한다. |
|  | | | 4.Borrower 객체를 BorrowerCollection 객체에 저장 후 속성을 출력한다. |
| 5.Librarian은 Borrower에게 등록되었음을 고지하고 회원증을 발급해준다. | | |  |
| Alternative Scenario | |  | |

**Use Case Description Iteration #1 <2> registerOneBook**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | registerOneBook | | |
| Primary Actor | Librarian | | |
| Secondary Actor | None | | |
| 개요 | Librarian이 등록되지 않은 Book을 Library에 등록한다. | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1.Librarian이 등록되지 않은 Book의 title, author, c\_num를 확인한다. | | |  |
| 2.Librarian은 Book의 title, author, c\_num를 시스템에 등록한다. | | |  |
|  | | | 3.title, author, c\_num를 담은 Book 객체를 생성한다. |
|  | | | 4.Book 객체를 BookCollection 객체에 저장하고 속성을 출력한다. |
| Alternative Scenario | |  | |

**Use Case Description Iteration #1 <3> displayBooksForLoan**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | displayBooksForLoan | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower (Option) | | |
| 개요 | 시스템이 대출가능한 Book들을 화면에 출력한다. | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1.Librarian은 시스템에 대출가능한 Book의 목록을 요청한다. | | |  |
|  | | | 2.BookCollection 객체에서 Book객체들을 오름차순으로 정렬한다. |
|  | | | 3.정렬된 BookCollection 객체에서 Book 객체들 각각 isBookForLoan을  확인하고, true일 경우 속성을 출력한다. |
| Alternative Scenario | |  | |

**Use Case Description Iteration #1 <4> displayBooksOnLoan**

|  |  |  |
| --- | --- | --- |
| Use Case Name | displayBooksOnLoan | |
| Primary Actor | Librarian | |
| Secondary Actor | Borrower (Option) | |
| 개요 | 시스템이 대출중인 Book들을 화면에 출력한다. | |
| Main Scenario | | |
| Actor Action | | System Response |
| 1.Librarian은 시스템에 대출중인 Book의 목록을 요청한다. | |  |
|  | | 2.Book Collection 객체에서 Book 객체들을 오름차순으로 정렬한다. |
|  | | 3.정렬된 BookCollection 객체에서 Book 객체들 각각 isBookOnLoan을 확인하고, true일 경우 속성을 출력한다. |
| Alternative Scenario |  | |

**Use Case Description Iteration #1 <5> borrowOneBook**

|  |  |  |
| --- | --- | --- |
| Use Case Name | borrowOneBook | |
| Primary Actor | Librarian | |
| Secondary Actor | Borrower | |
| 개요 | Borrower가 Librarian을 통해 one book을 대출한다. | |
| Main Scenario | | |
| Actor Action | | System Response |
| 1.Borrower가 대출하려는 Book을 들고 대출창구에 도착한다. | |  |
| 2.Borrower가 대출하려는 Book과 회원증을 Librarian에게 제출한다. | |  |
| 3.Librarian이 대출할 Book과 회원증의 바코드(c\_num, u\_name)를 인식시킨다. | |  |
|  | | 4.BookCollection 객체에서 나온 대출할 Book 객체를 검색하고, BorrowerCollection 객체에서 Borrower 객체를 검색한다. |
|  | | 5.Book 객체의 속성과 Borrower 객체의 속성을 화면에 출력한다. |
|  | | 6.찾아낸 Book 객체와 Borrower 객체가 대출 가능한지 확인한다. |
|  | | 7.Loan 객체를 생성하고 Book 객체와 Borrower 객체를 양방향으로 링크한다. (Borrower-Loan-Book) |
|  | | 8.Loan 객체에서 대출일자를 현재시간으로 하고 반납기한일자를 계산하고 저장한다. |
|  | | 9.Loan 객체를 LoanCollection 객체에 저장한다. |
|  | | 10.Loan 객체의 속성을 화면에 출력한다. |
| 11. Librarian은 Borrower에게 Book과 회원증을 건네고, 반납일자를 고지한다. | |  |
| 12. Borrower는 Book과 회원증을 받고 대출창구를 떠난다. | |  |
| Alternative Scenario |  | |

**Use Case Description Iteration #1 <6> returnOneBook**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | | returnOneBook | |
| Primary Actor | | Librarian | |
| Secondary Actor | | Borrower | |
| 개요 | | Borrower가 borrow한 Book을 Librarian을 통해 Return한다. | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1.Borrower가 반납하려는 Book을 들고 반납창구에 도착한다. | | |  |
| 2.Borrower가 반납하려는 Book을 Librarian에게 제출한다. | | |  |
| 3.Librarian이 반납할 Book의 바코드(c\_num)를 인식시킨다. | | |  |
|  | | | 4.BookCollection 객체에서 반납할 Book 객체를 찾는다. |
|  | | | 5.Book 객체에 연결된 Loan 객체를 찾는다. |
|  | | | 6.Loan 객체가 링크를 해제하는 메시지를 받고, Loan객체와 연결된 Book 객체, Borrower 객체 각각의 양방향 링크를 해제한다. |
|  | | | 7.Loan 객체에서 반납된 일자를 현재시간으로 저장한다. |
|  | | | 8.Loan 객체에서 속성을 출력한다. |
| 9.Librarian은 Borrower에게 반납이 완료되었음을 고지한다. | | |  |
| 10.Borrower는 반납창구를 떠난다. | | |  |
| Alternative Scenario |  | | |

**Use Case Description Iteration #2 <1> registerOneBorrower**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | registerOneBorrower | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower | | |
| 개요 | 등록되지 않은 Borrower가 Librarian에게 이용자 등록을 신청한다. | | |
| 사전조건 | 1. Borrower 객체가 등록되지 않은 상태이다. | | |
| 사후조건 | 1. Borrower 객체가 borrowerCollection에 저장된 상태이다.  2. Borrower 객체에 Loan 객체가 없는 상태이다. | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 2.   등록되지 않은 Borrower가 이용자 등록을 위해 이용자 등록신청서(name)를 작성하고 Librarian에게 제출한다.(단, unique name은 name으로 대체한다.) | | |  |
| 3.   Librarian은 Borrower에게 받은 이용자 등록신청서(name)를 바탕으로 Borrower의 정보를 시스템에 등록한다. | | |  |
|  | | | 4.   name을 가진 Borrower 객체를 생성한다. |
|  | | | 4. Borrower 객체를 HashSet 클래스를 사용한 borrowerCollection에 추가하고, 속성을 출력한다. |
| 5. Librarian은 Borrower에게 등록되었음을 고지하고 회원증을 발급해준다. | | |  |
| Alternative Scenario | | N/A | |

**Use Case Description Iteration #2 <2> registerOneBook**

|  |  |  |
| --- | --- | --- |
| Use Case Name | registerOneBook | |
| Primary Actor | Librarian | |
| Secondary Actor | None | |
| 개요 | Librarian이 등록되지 않은 Book을 Library에 등록한다. | |
| 사전조건 | 1. Book 객체가 등록되지 않은 상태이다. | |
| 사후조건 | 1. Book 객체가 bookCollection에 저장된 상태이다.  2. Book 객체에서 Loan 객체의 참조변수가 없는 상태이다. | |
| Main Scenario | | |
| Actor Action | | System Response |
| 1. Librarian이 등록되지 않은 Book의 title, author, ISBN을 파악한다. (단, unique catalog number를 ISBN으로 대체한다.) | |  |
| 1. Librarian은 Book의 title, author, ISBN를 시스템에 등록한다. | |  |
|  | | 1. title, author, ISBN를 가진 Book 객체를 생성한다. |
|  | | 1. Book 객체를 TreeSet 클래스를 사용한 bookCollection에 추가하고, 속성을 출력한다. |
| Alternative Scenario | N/A | |

**Use Case Description Iteration #2 <3> displayBooksForLoan**

|  |  |  |
| --- | --- | --- |
| Use Case Name | displayBooksForLoan | |
| Primary Actor | Librarian | |
| Secondary Actor | Borrower (Option) | |
| 개요 | 시스템이 대출가능한 Book 객체들을 화면에 ISBN 오름차순으로 출력한다. | |
| 사전조건 | 1. bookCollection은 ISBN의 오름차순으로 정렬되어 있다. (BST) | |
| 사후조건 | N/A | |
| Main Scenario | | |
| Actor Action | | System Response |
| 1. Librarian은 시스템에 대출가능한 Book 객체의 목록을 요청한다. | |  |
|  | | 1. 2. bookCollection에서 Book 객체들을 반복해서 가져온다, isBookForLoan을 확인하고, true일 경우 속성을 출력한다. |
| Alternative Scenario | 2.1 bookCollection에 Book 객체가 한 권도 없을 경우에 “책이 한 권도 없습니다”라는 메시지를 출력한다.  2.2 bookColleciton에 모든 Book 객체가 대출 중인 경우 “모든 책이 대출 중입니다” 라는 메시지를 출력한다. | |

**Use Case Description Iteration #2 <4> displayBooksOnLoan**

|  |  |  |
| --- | --- | --- |
| Use Case Name | displayBooksOnLoan | |
| Primary Actor | Librarian | |
| Secondary Actor | Borrower (Option) | |
| 개요 | 시스템이 대출중인 Book 객체들을 화면에 ISBN의 오름차순으로 출력한다. | |
| 사전조건 | 1. bookCollection 객체는 ISBN의 오름차순으로 정렬되어 있다.(BST) | |
| 사후조건 | N/A | |
| Main Scenario | | |
| Actor Action | | System Response |
| 1. Librarian은 시스템에 대출중인 Book 객체의 목록을 요청한다. | |  |
|  | | 1. bookCollection에서 Book 객체들을 반복해서 가져온다, isBookOnLoan을 확인하고, true일 경우 속성을 출력한다. |
| Alternative Scenario | 2.1 bookCollection 객체에 Book 객체가 한 권도 없을 경우에 “책이 한 권도 없습니다”라는 메시지를 출력한다.  2.2 bookColleciton 객체에 모든 Book 객체가 대출 중이 아닌 경우 “대출 중인 책이 없습니다” 라는 메시지를 출력한다. | |

**Use Case Description Iteration #2 <5> borrowOneBook**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | borrowOneBook | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower | | |
| 개요 | Borrower 객체가 Librarian을 통해 책 한 권을 대출한다. | | |
| 사전조건 | 1. Book 객체가 등록되어 있는 상태이다.  2. Borrower 객체가 등록되어 있는 상태이다.  3. Book 객체와 Borrower 객체가 대출 가능한 상태이다. | | |
| 사후조건 | 1. Book 객체가 대출 중인 상태이다.  2. Borrower 객체가 대출 중인 상태이다.  3. Loan객체는 loanCollection에 저장된 상태이다. | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1. Borrower가 대출하려는 Book과 회원증을 들고 대출창구에 도착하여, Librarian에게 제출한다. | | |  |
| 1. Librarian이 대출할 Book과 회원증의 바코드(ISBN, name)를 인식시킨다. | | |  |
|  | | | 1. bookCollection에서 대출할 Book 객체를 찾고, borrowerCollection에서 Borrower 객체를 찾는다. |
|  | | | 1. Book 객체의 속성과 Borrower 객체의 속성을 화면에 출력한다. |
|  | | | 5. 찾아낸 Book 객체와 Borrower 객체가 대출 가능한지 확인한다. |
|  | | | 6. Loan 객체를 생성하고 Book 객체와 Borrower 객체를 양방향으로 링크한다. (Borrower-Loan-Book) |
|  | | | 7. Loan 객체에서 대출일자를 현재시간으로 저장하고, 반납기한일자를 계산하고 저장한다. |
|  | | | 8. Loan 객체를 loanCollection에 추가한다. |
|  | | | 9. Loan 객체의 속성을 화면에 출력한다. |
| 10. Librarian은 Borrower에게 Book과 회원증을 건네고, 반납일자를 고지한다. | | |  |
| 11. Borrower는 Book과 회원증을 받고 대출창구를 떠난다. | | |  |
| Alternative Scenario | | 3. 찾고자하는 Book 객체 또는 Borrower 객체가 없는 경우 “책을 찾을 수 없습니다.”, “이용자를 찾을 수 없습니다.” 라는 오류 메시지를 출력한다.  5.1 Book 객체가 대출 불가능할 경우 "해당 책은 대출 불가능합니다.” 라는 메시지를 출력한다.  5.2 Borrower 객체가 대출 불가능할 경우 “해당 이용자는 대출 불가능합니다.” 라는 메시지를 출력한다. | |

**Use Case Description Iteration #2 <6> returnOneBook**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | | returnOneBook | |
| Primary Actor | | Librarian | |
| Secondary Actor | | Borrower | |
| 개요 | | Borrower가 대출한 Book을 Librarian을 통해 Return한다. | |
| 사전조건 | | 1. Book 객체가 등록된 상태이다.  2. Borrower 객체가 등록된 상태이다.  3. Book 객체와 Borrower 객체가 대출 중인 상태이다. | |
| 사후조건 | | 1. Book 객체가 반납된 상태이다.  2. Borrower 객체가 해당 Book을 반납한 상태이다.  3. Loan객체는 Book 객체, Borrower 객체와 링크가 없는 상태이다. | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1. Borrower가 반납하려는 Book을 들고 반납창구에 도착하여, Librarian에게 제출한다. | | |  |
| 1. Librarian이 반납할 Book의 바코드(ISBN)를 인식시킨다. | | |  |
|  | | | 1. bookCollection에서 반납할 Book 객체를 찾는다. |
|  | | | 1. Book 객체에 연결된 Loan 객체를 찾는다. |
|  | | | 1. Loan 객체가 링크를 해제하는 메시지를 받고, Loan객체와 연결된 Book 객체, Borrower 객체 각각의 양방향 링크를 해제한다. |
|  | | | 1. Loan 객체의 반납된 일자를 현재시간으로 저장한다. |
|  | | | 1. Loan 객체의 속성을 출력한다. |
| 1. Librarian은 Borrower에게 반납이 완료되었음을 고지한다. | | |  |
| 1. Borrower는 반납창구를 떠난다. | | |  |
| Alternative Scenario | 3. 찾고자하는 Book 객체가 없는 경우 오류 메시지를 출력한다. | | |

**Use Case Description Iteration #3 <1> registerOneBorrower**

|  |  |  |
| --- | --- | --- |
| Use Case Name | registerOneBorrower | |
| Primary Actor | Librarian | |
| Secondary Actor | Borrower | |
| 개요 | 등록되지 않은 Borrower가 Librarian에게 이용자 등록을 신청한다. | |
| 사전조건 | 1. Borrower 객체가 등록되지 않은 상태이다. | |
| 사후조건 | 1. Borrower 객체가 borrowerCollection에 저장된 상태이다.  2. Borrower 객체에 Loan 객체가 없는 상태이다. | |
| Main Scenario | | |
| Actor Action | | System Response |
| 1. 등록되지 않은 Borrower가 이용자 등록을 위해 이용자 등록신청서(name)를 작성하고 Librarian에게 제출한다.(단, unique name은 name으로 대체한다.) | |  |
| 2. Librarian은 Borrower에게 받은 이용자 등록신청서(name)를 바탕으로 Borrower의 정보를 시스템에 등록한다. | |  |
|  | | 1. name을 가진 Borrower 객체를 생성한다. |
|  | | 4. Borrower 객체를 HashSet 클래스를 사용한 borrowerCollection에 추가하고, 속성을 출력한다. |
| 5. Librarian은 Borrower에게 등록되었음을 고지하고 회원증을 발급해준다. | |  |
| Alternative Scenario | N/A | |

**Use Case Description Iteration #3 <2> registerOneBook**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | | registerOneBook | |
| Primary Actor | | Librarian | |
| Secondary Actor | | None | |
| 개요 | | Librarian이 등록되지 않은 Book을 Library에 등록한다. | |
| 사전조건 | | 1. Book 객체가 등록되지 않은 상태이다. | |
| 사후조건 | | 1. Book 객체가 bookCollection에 저장된 상태이다.  2. Book 객체에서 Loan 객체의 참조변수가 없는 상태이다. | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1. Librarian이 등록되지 않은 Book의 title, author, ISBN을 파악한다. (단, unique catalog number를 ISBN으로 대체한다.) | | |  |
| 1. Librarian은 Book의 title, author, ISBN를 시스템에 등록한다. | | |  |
|  | | | 1. title, author, ISBN를 가진 Book 객체를 생성한다. |
|  | | | 1. Book 객체를 TreeSet 클래스를 사용한 bookCollection에 추가하고, 속성을 출력한다. |
| Alternative Scenario | N/A | | |

**Use Case Description Iteration #3 <3> displayBooksForLoan**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | displayBooksForLoan | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower (Option) | | |
| 개요 | 시스템이 대출가능한 Book 객체들을 화면에 ISBN 오름차순으로 출력한다. | | |
| 사전조건 | 1. bookCollection은 ISBN의 오름차순으로 정렬되어 있다. (BST) | | |
| 사후조건 | N/A | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1. Librarian은 시스템에 대출가능한 Book 객체의 목록을 요청한다. | | |  |
|  | | | 2. bookCollection에서 Book 객체들을 반복해서 가져온다,  isBookForLoan을  확인하고, true일 경우 속성을 출력한다. |
| Alternative Scenario | | 2.1 bookCollection에 Book 객체가 한 권도 없을 경우에 “책이 한 권도 없습니다”라는 메시지를 출력한다.  2.2 bookColleciton에 모든 Book 객체가 대출 중인 경우 “모든 책이 대출 중입니다” 라는 메시지를 출력한다. | |

**Use Case Description Iteration #3 <4> displayBooksOnLoan**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | displayBooksOnLoan | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower (Option) | | |
| 개요 | 시스템이 대출중인 Book 객체들을 화면에 ISBN의 오름차순으로 출력한다. | | |
| 사전조건 | 1. bookCollection 객체는 ISBN의 오름차순으로 정렬되어 있다.(BST) | | |
| 사후조건 | N/A | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1. Librarian은 시스템에 대출중인 Book 객체의 목록을 요청한다. | | |  |
|  | | | 2. bookCollection에서 Book 객체들을 반복해서 가져온다,  isBookOnLoan을 확인하고, true일 경우 속성을 출력한다. |
| Alternative Scenario | | 2.1 bookCollection 객체에 Book 객체가 한 권도 없을 경우에 “책이 한 권도 없습니다”라는 메시지를 출력한다.  2.2 bookColleciton 객체에 모든 Book 객체가 대출 중이 아닌 경우 “대출 중인 책이 없습니다” 라는 메시지를 출력한다. | |

**Use Case Description Iteration #3 <5> borrowOneBook**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | borrowOneBook | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower | | |
| 개요 | Borrower 객체가 Librarian을 통해 책 한 권을 대출한다. | | |
| 사전조건 | 1. Book 객체가 등록되어 있는 상태이다.  2. Borrower 객체가 등록되어 있는 상태이다.  3. Book 객체와 Borrower 객체가 대출 가능한 상태이다. | | |
| 사후조건 | 1. Book 객체가 대출 중인 상태이다.  2. Borrower 객체가 대출 중인 상태이다.  3. Loan객체는 loanCollection에 저장된 상태이다. | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1. Borrower가 대출하려는 Book과 회원증을 들고 대출창구에 도착하여,  Librarian에게 제출한다. | | |  |
| 2. Librarian이 대출할 Book과 회원증의 바코드(ISBN, name)를 인식시킨다. | | |  |
|  | | | 3.bookCollection에서 대출할 Book 객체를 찾고, borrowerCollection에서 Borrower 객체를 찾는다. |
|  | | | 4. Book 객체의 속성과 Borrower 객체의 속성을 화면에 출력한다. |
|  | | | 5. 찾아낸 Book 객체와 Borrower 객체가 대출 가능한지 확인한다. |
|  | | | 6. Loan 객체를 생성하고 Book 객체와 Borrower 객체를 양방향으로 링크한다. (Borrower-Loan-Book) |
|  | | | 7. Loan 객체에서 대출일자를 현재시간으로 저장하고, 반납기한일자를 계산하고 저장한다. |
|  | | | 8. Loan 객체를 loanCollection에 추가한다. |
|  | | | 9. Loan 객체의 속성을 화면에 출력한다. |
| 10. Librarian은 Borrower에게 Book과 회원증을 건네고, 반납일자를 고지한다. | | |  |
| 11. Borrower는 Book과 회원증을 받고 대출창구를 떠난다. | | |  |
| Alternative Scenario | | 3. 찾고자하는 Book 객체 또는 Borrower 객체가 없는 경우 “책을 찾을 수 없습니다.”, “이용자를 찾을 수 없습니다.” 라는 오류 메시지를 출력한다.  5.1 Book 객체가 대출 불가능할 경우 "해당 책은 대출 불가능합니다.” 라는 메시지를 출력한다.  5.2 Borrower 객체가 대출 불가능할 경우 “해당 이용자는 대출 불가능합니다.” 라는 메시지를 출력한다. | |

**Use Case Description Iteration #3 <6> returnOneBook**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | returnOneBook | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower | | |
| 개요 | Borrower가 대출한 Book을 Librarian을 통해 Return한다. | | |
| 사전조건 | 1. Book 객체가 등록된 상태이다.  2. Borrower 객체가 등록된 상태이다.  3. Book 객체와 Borrower 객체가 대출 중인 상태이다. | | |
| 사후조건 | 1. Book 객체가 반납된 상태이다.  2. Borrower 객체가 해당 Book을 반납한 상태이다.  3. Loan객체는 Book 객체, Borrower 객체와 링크가 없는 상태이다. | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1. Borrower가 반납하려는 Book을 들고 반납창구에 도착하여, Librarian에게 제출한다. | | |  |
| 1. Librarian이 반납할 Book의 바코드(ISBN)를 인식시킨다. | | |  |
|  | | | 1. bookCollection에서 반납할 Book 객체를 찾는다. |
|  | | | 1. Book 객체에 연결된 Loan 객체를 찾는다. |
|  | | | 1. Loan 객체가 링크를 해제하는 메시지를 받고, Loan객체와 연결된 Book 객체, Borrower 객체 각각의 양방향 링크를 해제한다. |
|  | | | 1. Loan 객체의 반납된 일자를 현재시간으로 저장한다. |
|  | | | 1. Loan 객체의 속성을 출력한다. |
| 1. Librarian은 Borrower에게 반납이 완료되었음을 고지한다. | | |  |
| 1. Borrower는 반납창구를 떠난다. | | |  |
| Alternative Scenario | | 3. 찾고자하는 Book 객체가 없는 경우 오류 메시지를 출력한다. | |

**Use Case Description Iteration #3 <7> displayClassifiedBooks**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | displayClassifiedBooks | | |
| Primary Actor | Librarian | | |
| Secondary Actor | Borrower(Option) | | |
| 개요 | 시스템이 Book 객체들을 분류하여 출력한다. | | |
| 사전조건 | 1. Book 객체가 등록된 상태이다.  2. 도서 데이터베이스에 ISBN과 해당 장르에 대한 매핑이 되어있다. | | |
| 사후조건 | 1. Book 객체가 분류된 상태이다. | | |
| Main Scenario | | | |
| Actor Action | | | System Response |
| 1. Librarian이 시스템에 찾고자 하는 장르의 내용분류번호를 입력한다. (단, 내용분류번호는 classificationNumber으로 대체한다.) | | |  |
|  | | | 2. classificationNumber의 목록들을 화면에 출력한다. |
| 3. Librarian이 찾으려는 장르의 classificationNumber를 선택한다. | | |  |
|  | | | 4. bookCollection에서 Book 객체들을 반복해서 검사한다. |
|  | | | 5. Book 객체의 classificationNumber의 isClassified를 확인하고, true일 경우 속성을 출력한다. |
| Alternative Scenario | | 5. 찾고자 하는 장르의 Book 객체가 없을 경우에 “해당 장르의 책이 한 권도 없습니다” 라는 메시지를 출력한다. | |

**Sequence Diagram Iteration #1 ~ #3**

변경사항

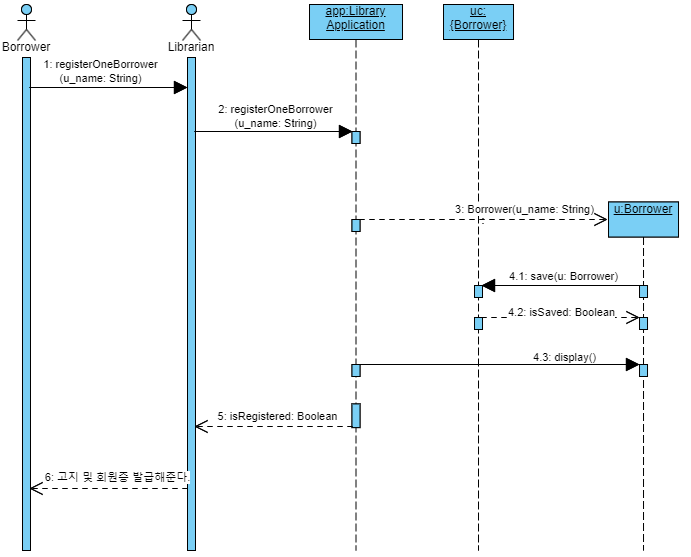
1. Iteration#2 displayBooksForLoan, displayBooksOnLoan 의 내용 중 atLeastOneOrMoreForLoan = true, atLeastOneOrMoreOnLoan = true 부분을 Iteration#3에서 삭제함.

변경 사유: 메소드를 보내는 부분에 변수값을 설정하는 오류를 수정함.

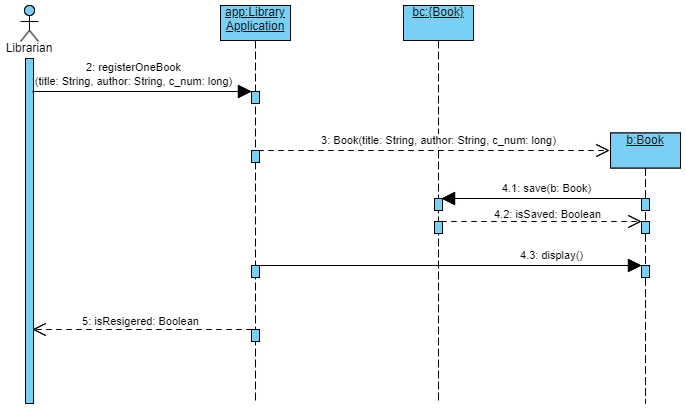
2. Iteration#2 borrowOneBook 의 1,2번째 alt 문 순서를 메시지의 올바른 사용을 위해 수정함.

3. Iteration#2의 Sequence Diagram 5번 6번에서 링크와 링크를 해제하는 부분을 Iteration#3에 구체적으로 표현하기 위해 추가함.

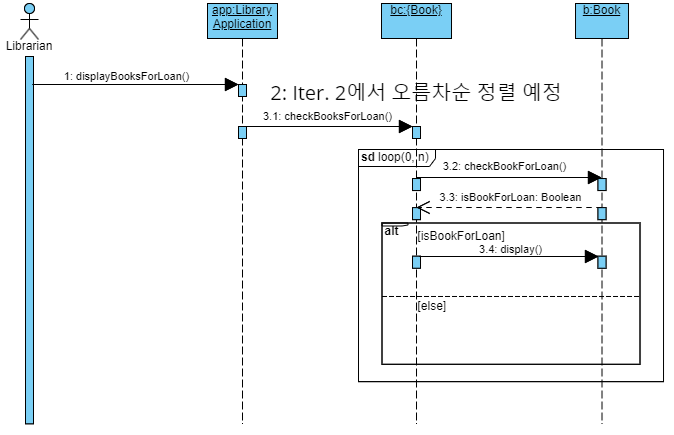
**Sequence Diagram Iteration #1 <1> registerOneBorrower**



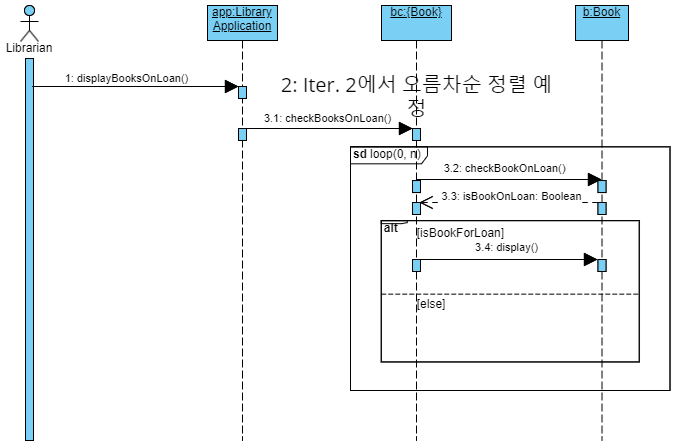
**Sequence Diagram Iteration #1 <2> registerOneBook**



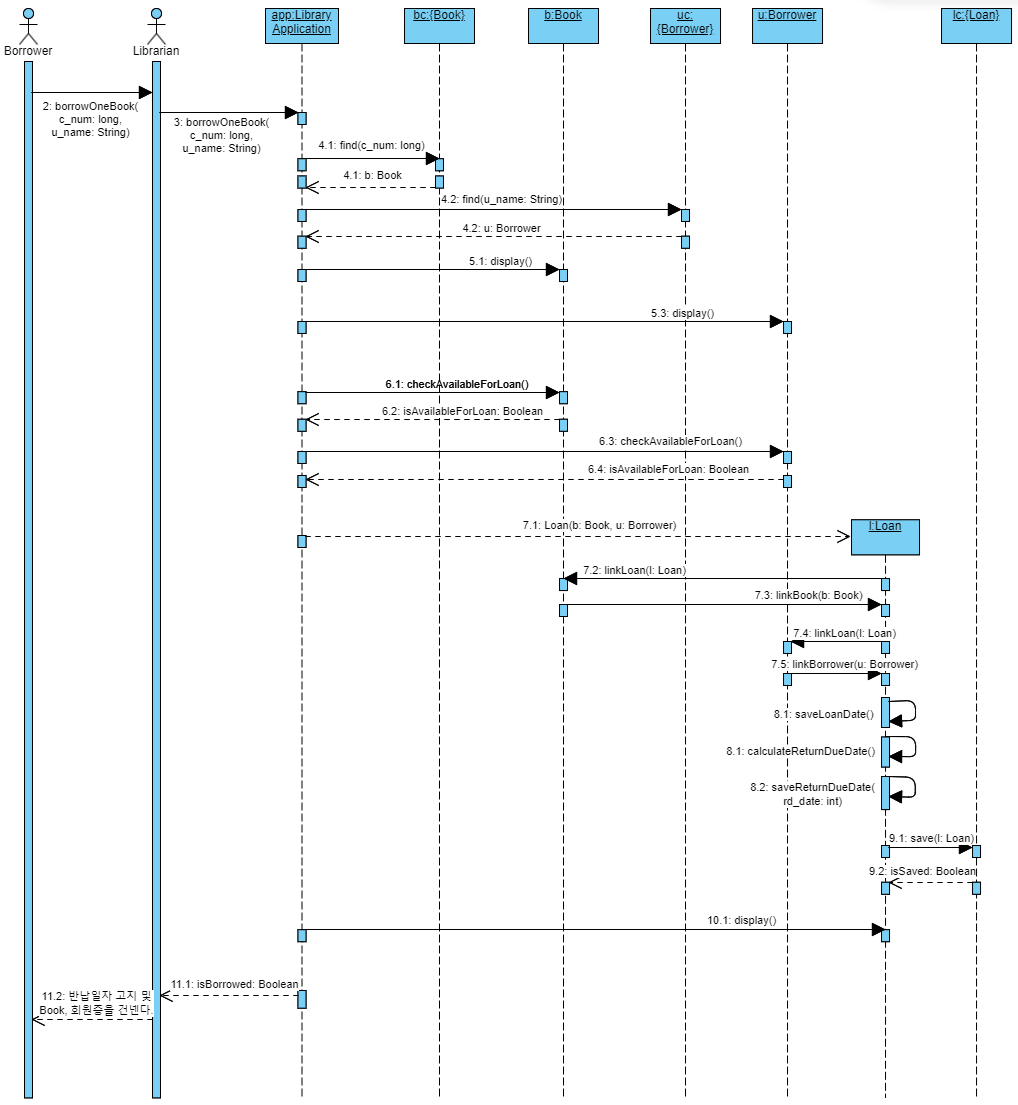
**Sequence Diagram Iteration #1 <3> displayBooksForLoan**



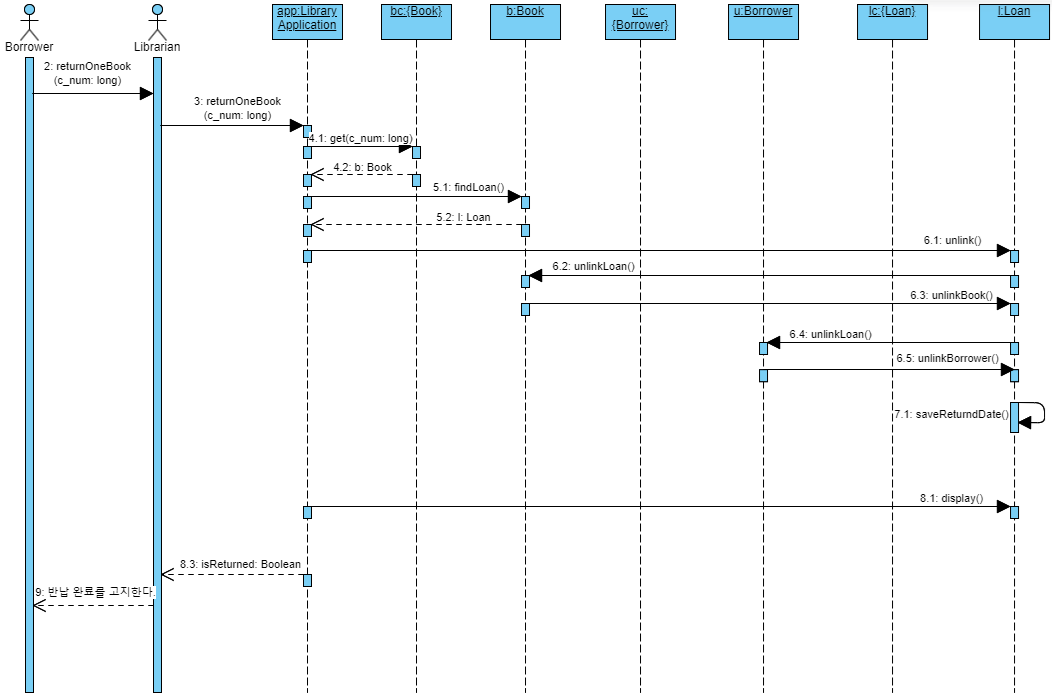
**Sequence Diagram Iteration #1 <4> displayBooksOnLoan**



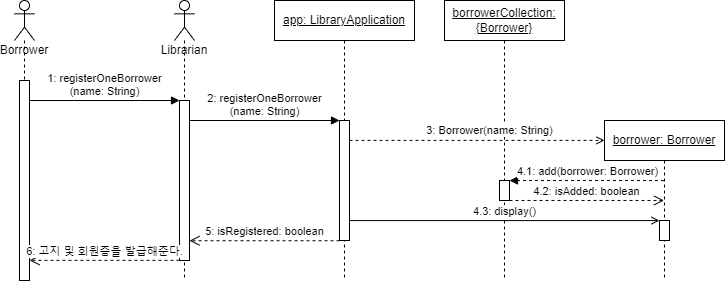
**Sequence Diagram Iteration #1 <5> borrowOneBook**



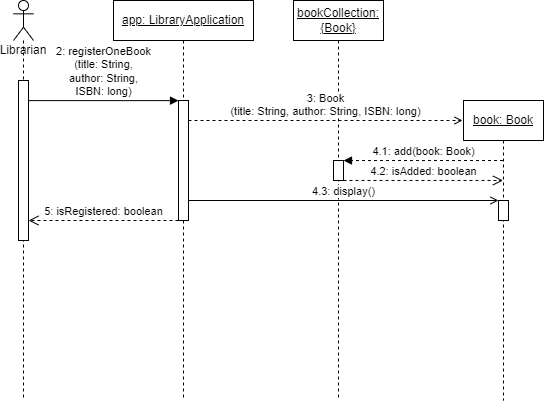
**Sequence Diagram Iteration #1 <6> returnOneBook**



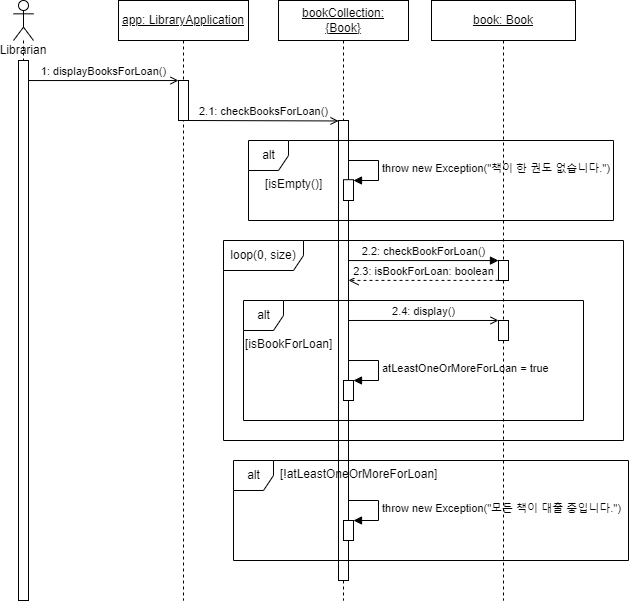
**Sequence Diagram Iteration #2 <1> registerOneBorrower**



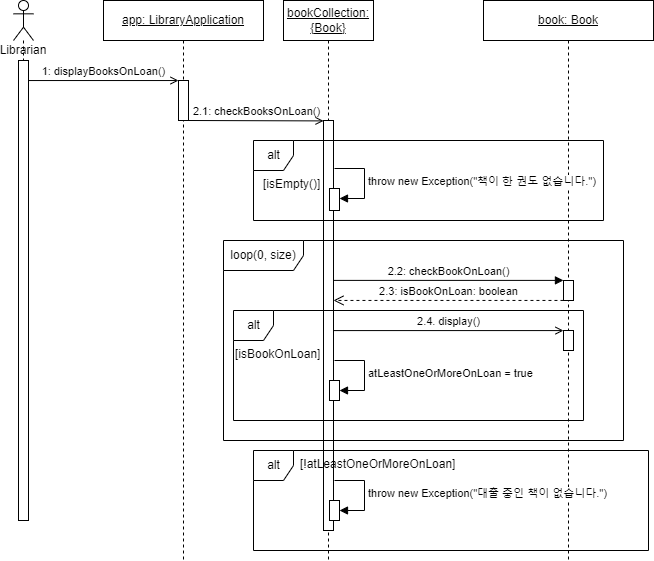
**Sequence Diagram Iteration #2 <2> registerOneBook**

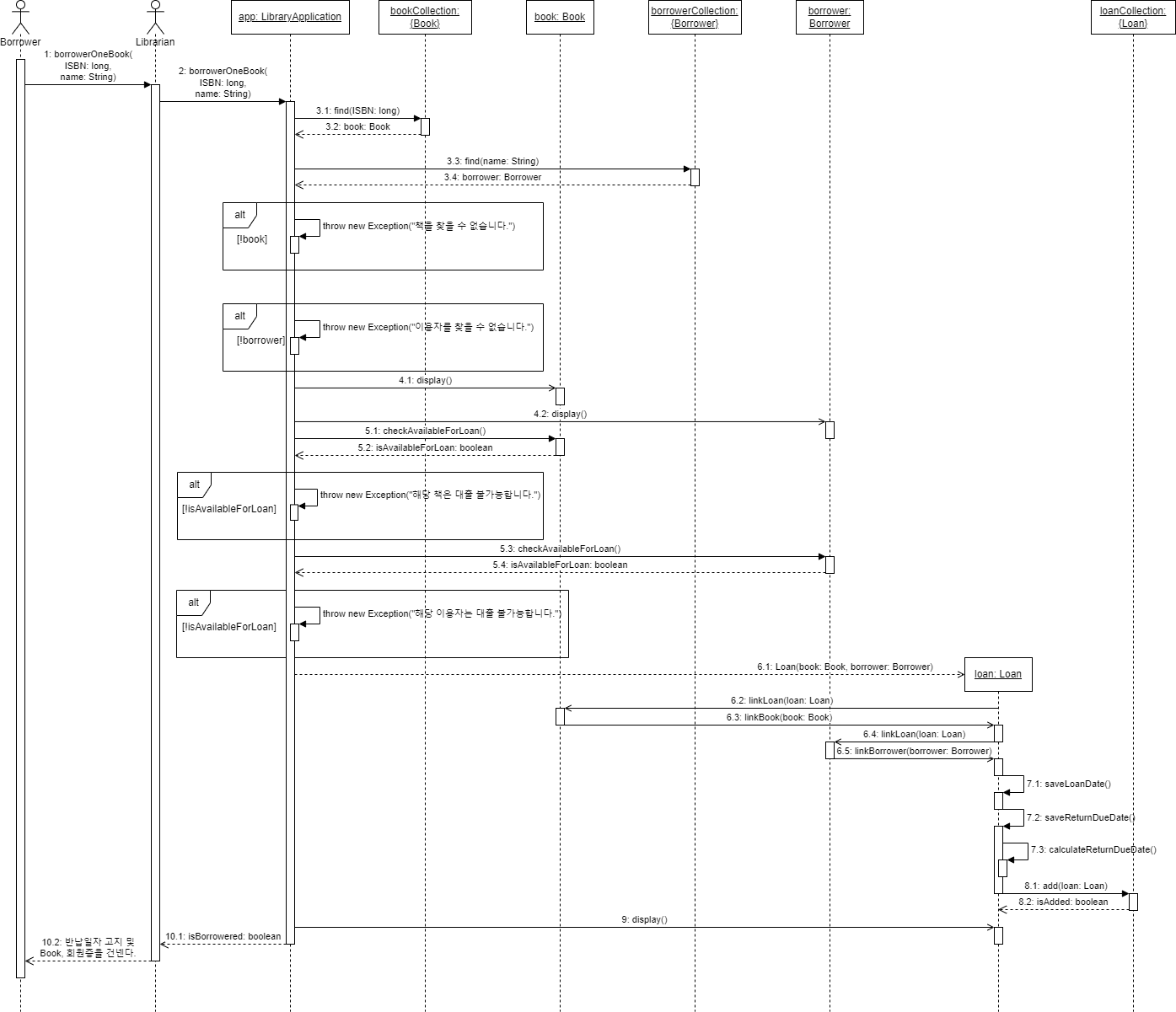


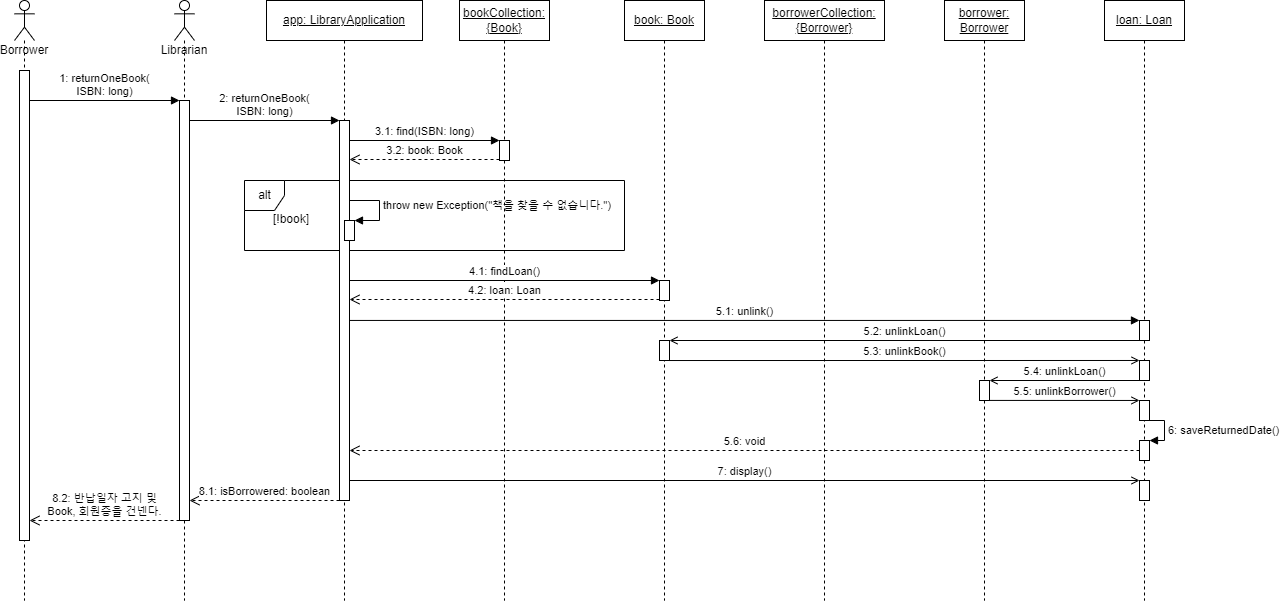
**Sequence Diagram Iteration #2 <3> displayBooksForLoan**



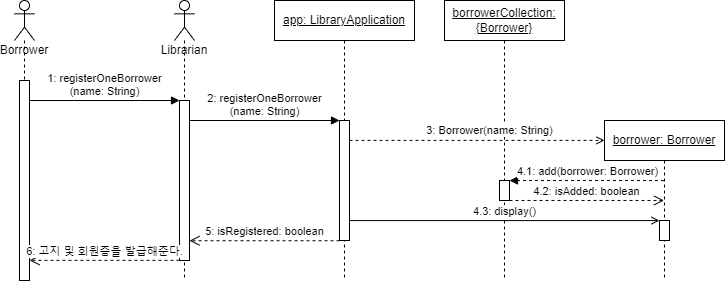
**Sequence Diagram Iteration #2 <4> displayBooksOnLoan**



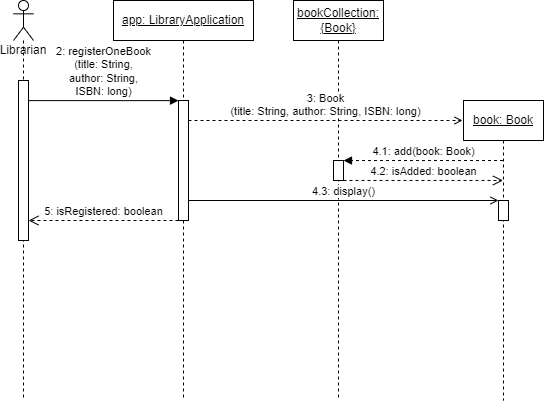
**Sequence Diagram Iteration #2 <5> borrowOneBook**

**Sequence Diagram Iteration #2 <6> returnOneBook**

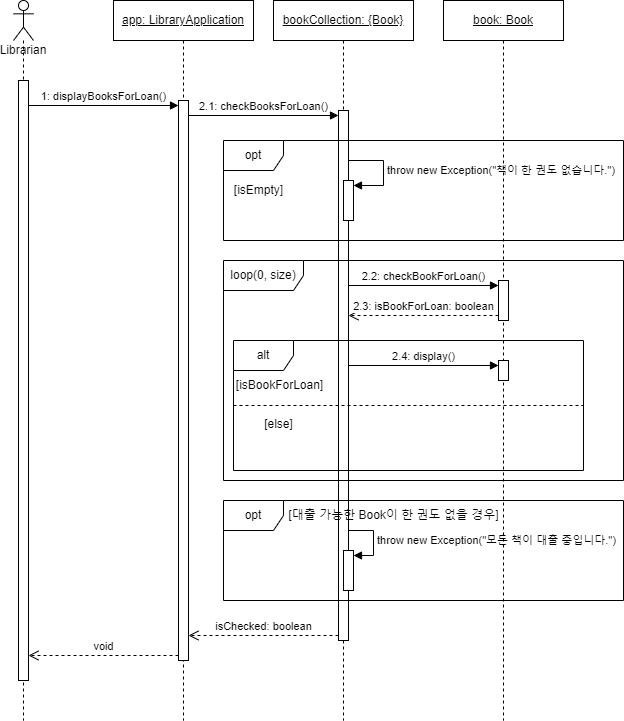
**Sequence Diagram Iteration #3 <1> registerOneBorrower**



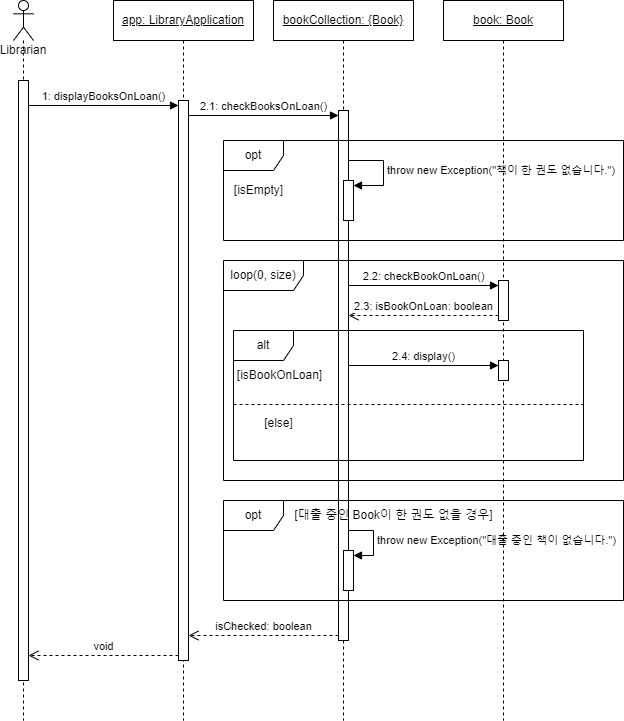
**Sequence Diagram Iteration #3 <2> registerOneBook**



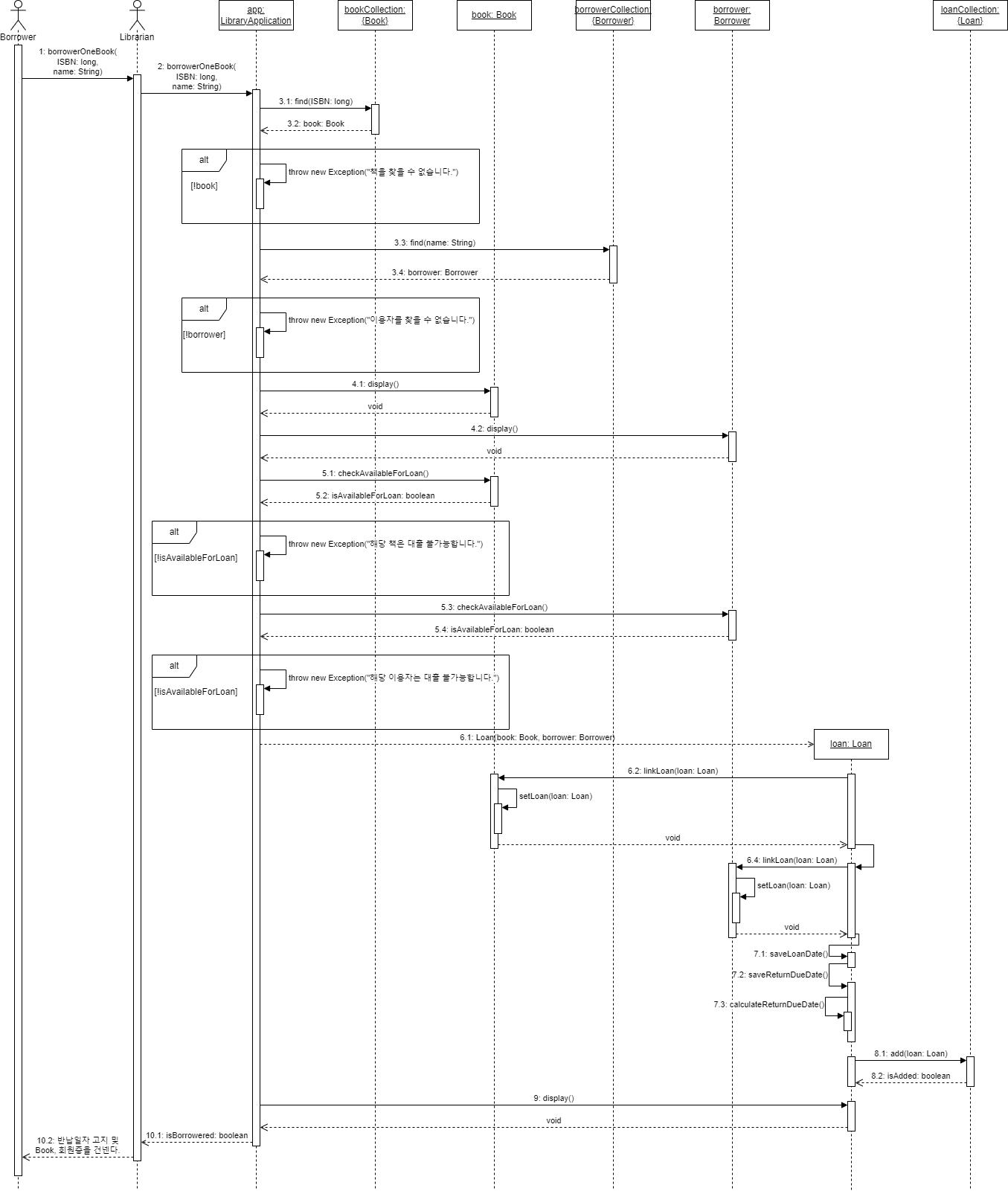
**Sequence Diagram Iteration #3 <3> displayBooksForLoan**

****

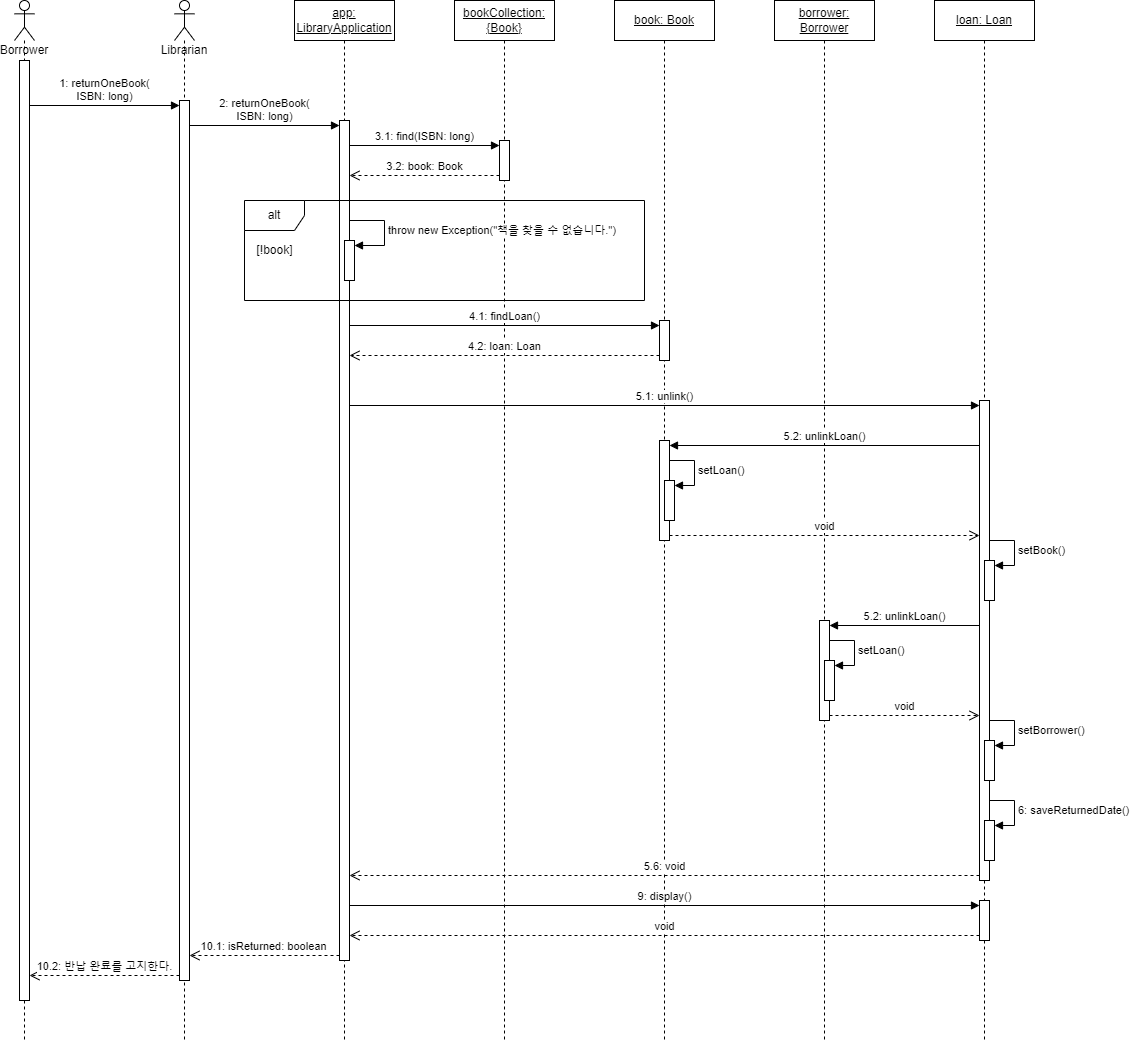
**Sequence Diagram Iteration #3 <4> displayBooksOnLoan**

****

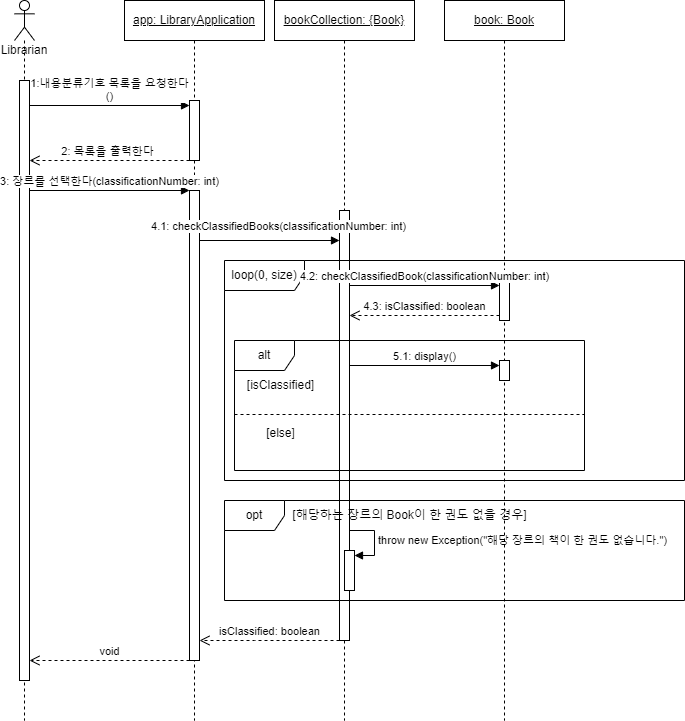
**Sequence Diagram Iteration #3 <5> borrowOneBook**

****

**Sequence Diagram Iteration #3 <6> returnOneBook**

****

**Sequence Diagram Iteration #3 <7> displayClassifiedBooks**

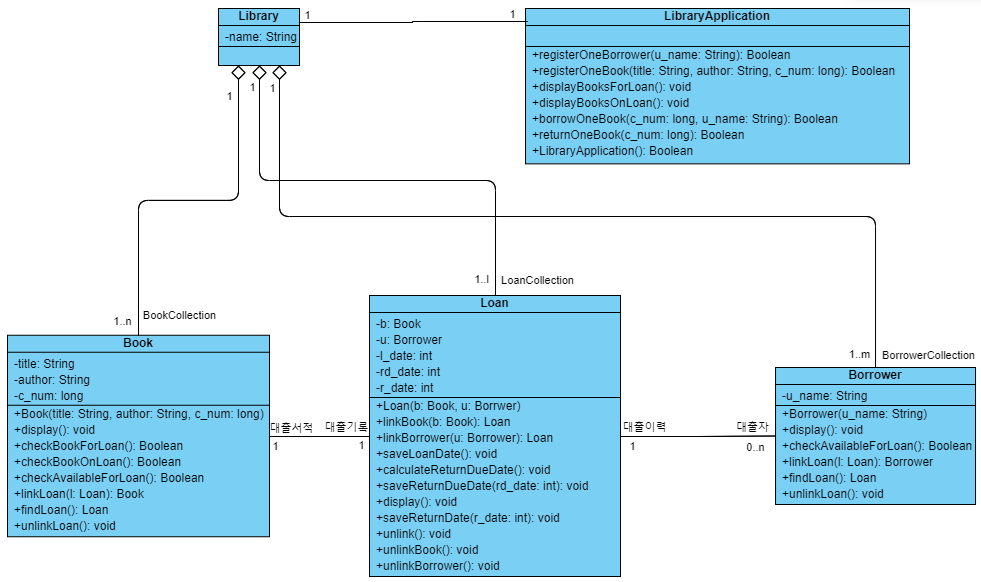
****

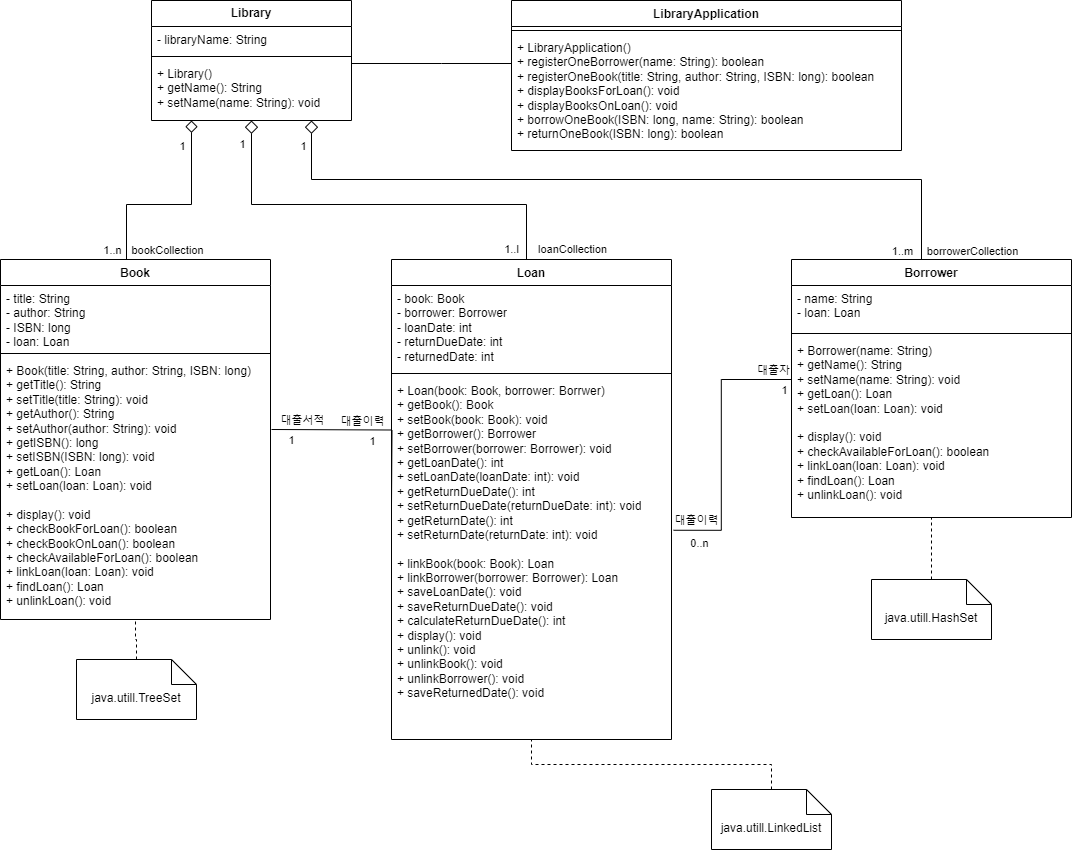
**Class Diagram Iteration #1 ~ #3**

**변경 사항**

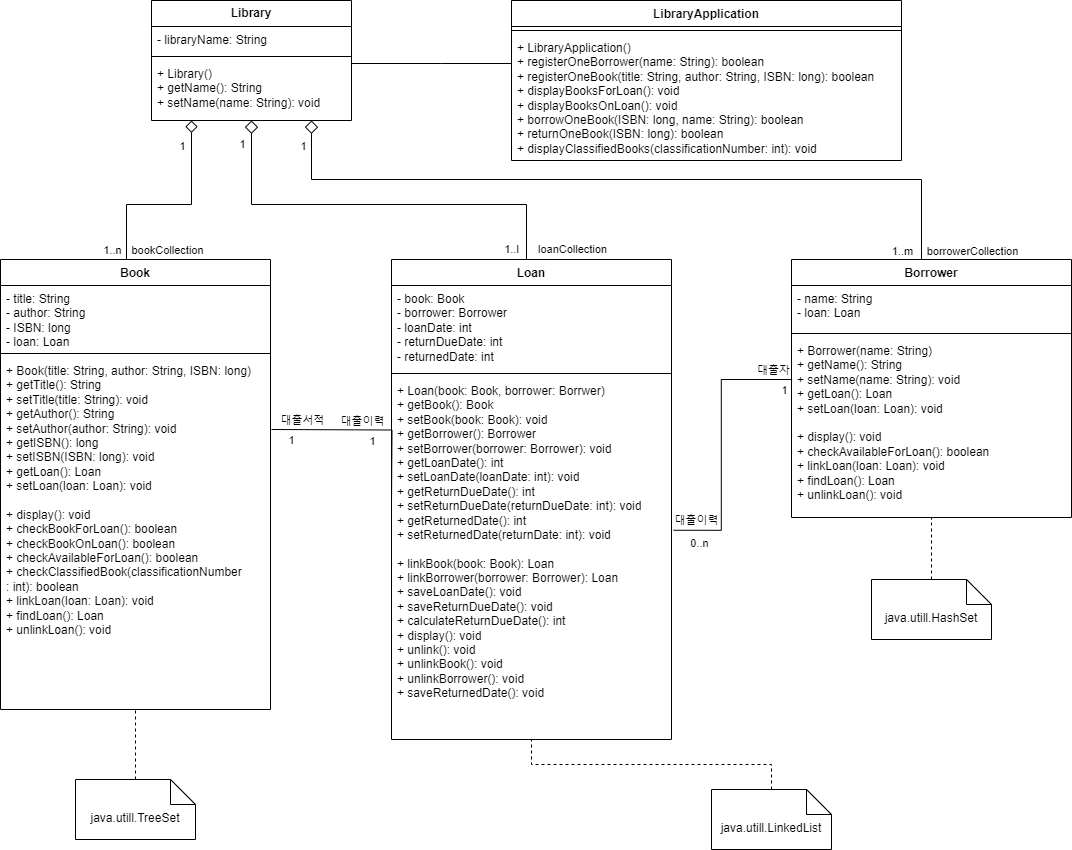
Sequence Diagram의 수정사항에 따라 Class Digram을 수정함.

**Class Diagram Iteration #1**



**Class Diagram Iteration #2**

**Class Diagram Iteration #3**

****

**버전 관리 및 Java Code**

버전 관리: Github

Github 주소: <https://github.com/Halozhan/2023-2-LibraryApplication.git>

Java Code : 압축 파일로 별도 첨부함.