[도서관리 프로젝트]

Ⅰ. 프로젝트 개요

1. **목적**: 도서관리 프로그램 인메모리/입출력(IO) 구축

2. 주요 기능

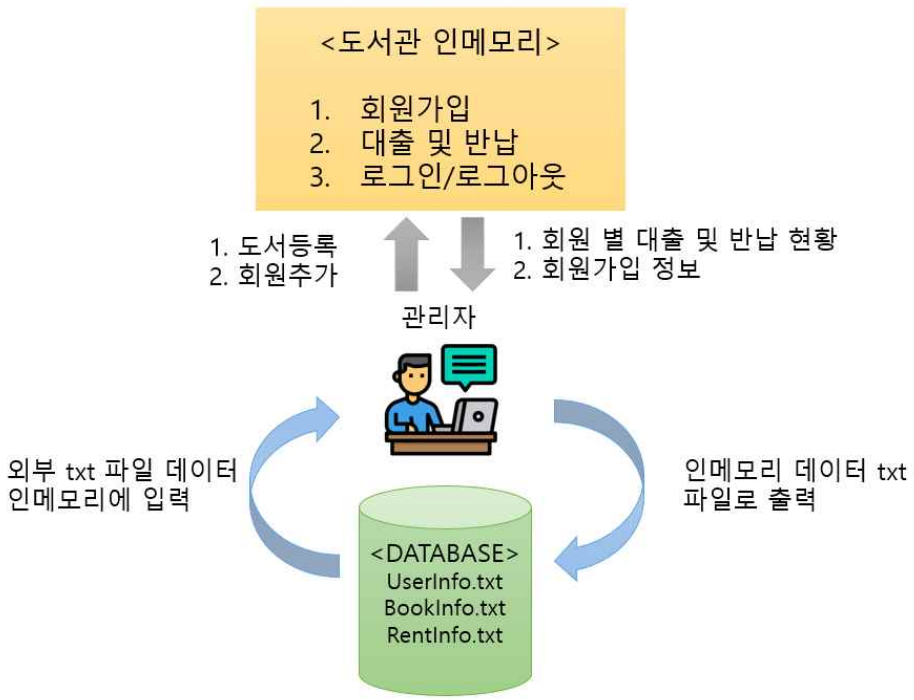
가. 관리자

1) 관리자 로그인 및 로그아웃 기능
2) 도서 등록 기능: 도서관리번호(중복금지), 도서명, 저자명, 출판사, 도서관 재고
3) 등록된 도서 수정 기능: 해당 도서관리번호에 맞는 항목 수정 기능
4) 회원정보 조회 기능: 회원추가 가능
5) 전체 회원의 도서 대출 및 반납 현황 확인 기능
6) (IO) 인메모리 파일 출력하기
7) (IO) 인메모리에 파일 입력하기

나. 일반회원

1) 회원가입: 아이디, 비밀번호, 이름, 생년월일, 이메일 입력 (단, 아이디 및 이메일 중복 금지)
2) 로그인: 아이디가 존재하지 않으면 알려줌, 아이디는 맞지만, 비밀번호가 틀리면 수정 가능
3) 마이페이지: 내 정보 확인 및 수정 가능, 대출 현황 조회 기능
4) 도서 검색: 관리자가 도서를 등록하면 도서명 일부만 입력해도 관련 정보 출력
5) 도서 대출: 원하는 도서의 도서관리 번호 입력 시 대출(대출 시 도서관 재고 줄어듦)
6) 도서 반납: 도서 관리 번호 입력 시 도서 반납 가능
7) 로그아웃

3. In-Memory와 입출력(IO)의 관계



Ⅱ. 프로젝트 Class와 Method

1. 인 메모리(In-Memory)

가. 도서관 메인 페이지

1) **LibraryMain**: 모든 클래스는 LibraryView로 모인 후 LibraryMain에서 메인 메소드를 통해 실행(launch)될 수 있습니다.

2) **LibraryView**: 도서관 메인 홈페이지 메뉴를 모은 클래스 로그인 전과 후로 나누어짐

가) showLibraryMenu(): 로그인 전 메뉴로, 일반회원을 위해 ① 회원가입, ② 로그인, ③ 시스템 종료 기능을, 관리자를 위해 ④ 관리자 로그인 기능을 만들었습니다. 또한 메뉴 번호 잘못 입력 시 다시 입력받을 수 있도록 while 무한루프를 사용했습니다.

```
public void showLibraryMenu() {
    System.out.println("=============================================================");
    System.out.println("\t \t \t <도서관 홈페이지>");
    System.out.println("=============================================================");

    System.out.println("1. 회원가입");
    System.out.println("2. 로그인");
    System.out.println("3. 시스템 종료");
    System.out.println("-------------------------------------------------------------");
    System.out.println("4. 관리자 로그인");
    System.out.println();
    System.out.print("*해당 페이지를 열람하려면 메뉴 번호를 입력하세요(1~4): ");
    int menuNum;

    loop: while (true) {
        menuNum = Integer.parseInt(sc.nextLine());
        switch (menuNum) {
        case 1:
            RegistrationView rf = new RegistrationView();
            rf.showRegistrationMenu(); // 회원가입 메뉴로 이동
            break loop;
        case 2:
            lov.showLoginMenu(); // 일반회원 로그인 메뉴로 이동
            break loop;
        case 3:
            System.exit(0); // 시스템 종료
            break loop;
        case 4:
            AdminView adv = new AdminView();
            adv.adminShowLoginMenu(); // 관리자 로그인 메뉴로 이동
            break loop;
        }
        System.out.print("잘못 입력하셨습니다. 메뉴번호를 다시 입력하세요: ");
    }
}
```

3) showLibraryMenuLoginVer(): 일반회원의 로그인 후 화면으로 메뉴 번호 잘못 입력 시 while 무한루프를 통해 재입력이 가능하도록 설정했습니다. 또한 로그인 시 로그인한 회원의 아이디와 로그인 일시가 출력됩니다.

```java
public void showLibraryMenuLoginVer() {
    System.out.println("=====================================================================");
    System.out.println("\t \t \t <도서관 홈페이지>");
    System.out.println("=====================================================================");

    System.out.println("*" + LoginFunc.getId() + "님 환영합니다!");
    String pattern = "yyyy년도 MM월 dd일 hh시 mm분";
    SimpleDateFormat sdf = new SimpleDateFormat(pattern);
    System.out.println("*로그인 일시: " + sdf.format(new Date())); // 로그인 일시 출력

    System.out.println();
    System.out.println("1. 도서 검색");
    System.out.println("2. 도서 반납");
    System.out.println("3. 마이페이지");
    System.out.println("4. 로그아웃");
    System.out.print("*해당 페이지를 열람하려면 메뉴 번호를 입력하세요(1~4): ");
    System.out.println();
```

나. 관리자

1) **AdminView**: 관리자 메뉴를 담고 있는 클래스

가) adminShowLoginMenu(): 관리자로 로그인 페이지로, AdminFunc 클래스의 관리자 로그인 기능인 adminLogin() 메서드와 연결됩니다.

나) adminShowMenu(): 관리자 로그인 성공 후 메뉴로 ① 회원 목록 조회, ② 도서 등록, ③ 전체 회원 대출 및 반납현황 조회 및 ④ 로그아웃 기능이 있습니다

다) adminShowMemberList(): 관리자 메뉴에서 ① 회원목록 선택 시 출력되는 화면으로, AdminFunc 클래스의 memebrListPrint(), memberAdd() 메서드를 이용하고 있습니다.

2) **AdminFunc**: 관리자 로그인 기능과 회원목록 조회 및 등록 기능을 모아둔 클래스

가) adminLogin(): 관리자 로그인 기능으로, Map에 아이디와 비밀번호를 저장하고 입력된 값과 일치하면 로그인 성공, 실패하면 로그인 실패 메시지와 함께 프로그램이 종료됩니다.

```java
System.out.print("아이디: ");
String id = sc.nextLine();
System.out.print("비밀번호: ");
String password = sc.nextLine();
Map<String, String> adminLogin = new HashMap<>();
adminLogin.put("root", "root");

if (adminLogin.containsKey(id) && adminLogin.get(id).equals(password)) {
    System.out.println("로그인 성공!");
    AdminView adv = new AdminView();
    adv.adminShowMenu();
```

나) memberListPrint(): 회원목록을 출력하는 기능으로 UserInfo 클래스의 인스턴스를 객체로 받는 List ‘userList’를 출력합니다.

```java
// 회원목록 출력 기능
public void memberListPrint() {
    System.out.println("아이디\t\t비밀번호\t\t이름\t\t생년월일\t\t이메일");
    System.out.println("-------------------------------------------------------------------------------------------");
    for (int i = 0; i < RegistrationFunc.userList.size(); i++) {
        System.out.println(RegistrationFunc.userList.get(i));
    }
}
```

다) memberAdd(): 회원등록 클래스인 RegistrationFunc의 회원입력기능 inputUser() 메서드를 이용하고 있습니다.

라) inputUserAgain(): 관리자가 회원을 여러번 등록할 수 있는 기능입니다.

3) **BookInfo**: 도서 정보에 관한 클래스로 ① 도서관리번호, ② 도서명, ③ 저자, ④ 출판사, ⑤ 도서재

고를 private 멤버변수로 가지고 있으며, toString 오버라이드를 통해 일정한 출력 형식을 부여했습니다.

```
@Override
public String toString() {
    return bookNumber + "\t\t" + bookName + "\t\t" + author + "\t\t" + publisher + "\t\t" + stock;
}
```

4) **AdminBookView**: 도서목록 조회 클래스

가) adminShowBookList(): 관리자에게 도서목록을 보여주는 기능으로 AdminBookFunc 클래스의 신착 도서 추가 메서드 inputBook()과 inputBookAgain()와 연결됩니다.

나) bookListPrint(): BookInfo의 인스턴스를 객체로 받는 List 'bookList'를 출력해서 보여주는 기능입니다.

5) **AdminBookFunc**: 도서목록 조회에 필요한 여러 기능을 담고 있는 클래스

가) bookNumberRegister(): 도서관리번호를 입력받는 메서드로, 효과적인 도서관리[1]를 위해 중복입력이 불가하도록 설정했습니다. 이 외에도 도서명, 저자명, 출판사, 재고에 대한 입력 메서드가 각각 존재합니다.

```
// 도서관리 번호 입력
public String bookNumberRegister() {
    System.out.print("도서관리번호(중복설정불가) : ");
    bookNumber = sc.nextLine().trim();
    for (int i = 0; i < bookList.size(); i++) {
        if (bookList.get(i).getBooknumber().equals(bookNumber)) {
            System.out.println("도서관리번호 " + bookNumber + "이/가 이미 존재합니다. 다시 입력해주세요.");
            bookNumberRegister();
        }
    }
    return bookNumber;
}
```

나) inputBook(): 신착도서 등록 메서드로 BookInfo를 객체로 받는 'bookList'에 항목을 추가합니다. 이와 더불어 도서를 계속 등록할 수 있도록 inputBookAgain() 메서드도 존재합니다.

```
static List<BookInfo> bookList = new ArrayList<>();

// 신착도서 등록
public void inputBook() {
    String bookNumber = bookNumberRegister();
    String bookName = bookNameRegister();
    String author = authorRegister();
    String publisher = publisherRegister();
    int stock = stockRegister();

    bookList.add(new BookInfo(bookNumber, bookName, author, publisher, stock));
}
```

다) changeBookList(): 도서 등록 시 오타를 수정할 수 있는 기능입니다. 우선 도서관리번호를 입력후 수정 항목을 선택하고, 바꾸고 싶은 내용을 입력하면 됩니다. 성공적으로 수정이 되었는지를 확인할 수 있도록 수정 후 정보를 출력해줍니다. for 문을 이용하여, bookList의 i번째 BookInfo의 도서 관리번호(getter를 통해 가져옴)가 앞서 입력한 도서관리번호와 일치한다면 원하는 수정 항목을 추출해 setter를 통해 변경합니다.

---

1) 도서명은 같지만, 저자명 및 출판사 등의 정보가 다른 도서들을 효과적으로 관리하기 위해 유일한 도서관리번호가 설정되어야 합니다.

```
System.out.print("수정사항: ");
String change = sc.nextLine().trim();

for (int i = 0; i < AdminBookFunc.bookList.size(); i++) {
    if (AdminBookFunc.bookList.get(i).getBooknumber().equals(bookNumber)) {
        if (choice == 0) {
            AdminBookFunc.bookList.get(i).setBookName(change);
        } else if (choice == 1) {
            AdminBookFunc.bookList.get(i).setAuthor(change);
        } else if (choice == 2) {
            AdminBookFunc.bookList.get(i).setPublisher(change);
        } else if (choice == 3) {
            AdminBookFunc.bookList.get(i).setStock(Integer.parseInt(change));
        }
```

6) **AdminRentView**: 전회원의 대출 및 반납현황 조회 클래스

가) showRentStatus(): 회원들의 도서대출 정보가 저장되어 있는 HashMap 'borroBook'에 있는 key와 Value를 출력합니다.

```
Set set = BorrowBookFunc.borrowBook.entrySet();
Iterator it = set.iterator();

while (it.hasNext()) {
    Map.Entry e = (Map.Entry) it.next();
    System.out.println("아이디: " + e.getKey());
    System.out.println(e.getValue());
    System.out.println("----------------------------------------------------------------------------------");
}
```

다. 일반회원

1) **UserInfo1**: 회원정보를 담은 클래스로 ① 아이디, ② 비밀번호, ③ 성명, ④ 생년월일, ⑤ 이메일을 private 멤버변수로 가지고 있습니다. 또한 오버라이드를 통해 toString()의 형식을 바꾸었습니다.

2) **RegistrationView**: 회원가입 페이지를 보여주는 기능을 담은 클래스로 RegistrationFunc 클래스의 inputUser() 메서드를 사용합니다.

가) showRegistrationFunc(): RegistrationFunc의 inputUser() 메서드를 이용합니다.

3) **RegistrationFunc**: 회원가입에 필요한 여러 기능을 담아둔 클래스로 AdminBookFunc과 동일한 원리로 만들었습니다.

가) inputUser(): UserInfo1의 멤버변수에 해당하는 값을 각각 입력받고, List 'userList'에 UserInfo1의 인스턴스들을 저장합니다.

```
static List<UserInfo1> userList = new ArrayList<>();

public void inputUser() {
    String id = idRegister();
    String password = passwordRegister();
    String name = nameRegister();
    String birthDay = birthDayRegister();
    String email = emailRegister();

    userList.add(new UserInfo1(id, password, name, birthDay, email));
}
```

4) **LoginView**: 일반회원 로그인 화면을 담은 클래스로, RegistrationView에서 회원가입이 완료되면 바로 연결되는 화면입니다.

가) showLoginMenu(): LoginFunc 클래스의 loginInput() 메서드와 연결됩니다.

5) **LoginFunc**: 일반회원 로그인에 필요한 여러 기능을 담은 클래스

가) inputId(): 아이디의 존재 여부를 확인하는 기능으로, 입력된 아이디가 List 'userList'에 포함된 그 어떠한 UserInfo1 인스턴스의 아이디(getter 이용)와도 일치하지 않는다면 아이디가 존재하지 않음을 알리고 계속 입력받습니다.

```
public void inputId() {
    System.out.print("아이디: ");
    id1 = sc.nextLine();
    setId(id1);
    for (int i = 0; i < RegistrationFunc.userList.size(); i++) {
        if (RegistrationFunc.userList.get(i).getId().equals(id1)) {
            inputPassword();
            loginValidation();
        }
    }
    System.out.println("아이디 [" + id1 + "]이/가 존재하지 않습니다. 다시 입력해주세요.");
    inputId();

}
```

나) loginValidation(): 입력한 아이디와 비밀번호가 일치한다면 로그인에 성공합니다.

```
for (int i = 0; i < RegistrationFunc.userList.size(); i++) {
    if (RegistrationFunc.userList.get(i).getId().equals(id1)
            && RegistrationFunc.userList.get(i).getPassword().equals(password)) {
        System.out.println("로그인 성공!");
        lbv.showLibraryMenuLoginVer();
    }
}
```

다) changePassword(): 아이디는 제대로 입력받았지만, 비밀번호가 틀릴 때 변경할 수 있는 기회를 열어두었습니다. 앞서 회원가입 시 입력한 이메일과 아이디가 일치한다면 변경할 수 있도록 설정했습니다.

```
System.out.print(id + "님 회원가입시 등록한 이메일을 입력하세요: ");
String email = sc.nextLine().trim();
for (int i = 0; i < RegistrationFunc.userList.size(); i++) {
    if (RegistrationFunc.userList.get(i).getEmail().equals(email)) {
        System.out.print("이메일이 회원정보와 일치합니다. 새로운 비밀번호를 입력하세요: ");
        String newPassword = sc.nextLine().trim();
        RegistrationFunc.userList.get(i).setPassword(newPassword);
        System.out.println("비밀번호가 성공적으로 변경되었습니다!");
```

6) **SearchBookView**: 도서 검색 및 도서 대출 페이지를 보여주는 기능을 담은 클래스

가) showSearchBookMenu(): 도서 검색 페이지를 보여주는 메서드로 SearchBookFunc의 searchBookName() 메서드를 사용합니다.

나) showBorrowBookView(): 도서대출 페이지를 보여주는 메서드로 BorrowBookFunc 클래스의 borrowBook() 메서드를 이용하고 있습니다.

7) **SearchBookFunc**: 도서 검색 기능을 담고 있는 클래스

가) searchBookName(): for 문과 contains()를 이용해 입력받은 값을 포함하는 BookInfo 인스턴스를 출력하는 기능입니다.

```
public void searchBookName(String bookSearch) { // 도서명 검색 시 조회기능
    for (int i = 0; i < AdminBookFunc.bookList.size(); i++) {
        if (AdminBookFunc.bookList.get(i).getBookName().contains(bookSearch))
            System.out.println(AdminBookFunc.bookList.get(i));
    }
}
```

8) **BorrowBookFunc**: 도서대출에 필요한 기능들을 담은 클래스

가) borrowBook(): HashMap 'borrowBook'에 대출정보를 저장하는 기능으로 key값은 회원아이디, value값은 객체 BookInfo로 설정했습니다. 해당 도서가 대출되면 도서관 재고가 줄어들도록 설정했는데, 만약 대출할 도서가 없다면 도서 대출 불가능 메시지를 출력합니다.

```
public void borrowBook(String bookNumber) {
    LibraryView lv = new LibraryView();

    for (int i = 0; i < AdminBookFunc.bookList.size(); i++) {
        if (AdminBookFunc.bookList.get(i).getBooknumber().equals(bookNumber)) {

            int num = AdminBookFunc.bookList.get(i).getStock();
            if (num > 0) {// 재고가 있다면 대출 가능
                num--;
                AdminBookFunc.bookList.get(i).setStock(num);
                borrowBook.put(LoginFunc.getId(), AdminBookFunc.bookList.get(i));
                System.out.println("도서대출이 완료되었습니다!");
                printBorrowBook();
                System.out.println("메인메뉴로 이동합니다.");
                lv.showLibraryMenuLoginVer();
            } else { // 도서 재고가 없다면 대출 불가능
                System.out.println("대출 가능한 재고가 없습니다");
                System.out.println("메인메뉴로 이동합니다.");
                lv.showLibraryMenuLoginVer();
            }
```

나) printBorrowBook(): 대출 도서를 보여주는 기능입니다.

9) **ReturnBookView**: 도서 반납 페이지를 보여주는 클래스

가) showReturnBookMenu(): ReturnBookFunc 클래스의 showBorrowedBook(), returnBook() 메서드를 이용해 도서 반납 페이지를 보여줍니다.

10) **ReturnBookFunc**: 도서 반납에 필요한 메서드를 담고 있는 클래스

가) showBorrowBook: 해당 회원의 대출 도서를 보여줍니다.

나) returnBook: 대출된 도서를 반납하는 기능인데, 이때 반납된 도서의 재고를 다시 1만큼 증가시킵니다.

```
// 대출도서 반납
public void returnBook(String bookNumber) {
    for (int i = 0; i < AdminBookFunc.bookList.size(); i++) {
        if (AdminBookFunc.bookList.get(i).getBooknumber().equals(bookNumber)) {
            int num = AdminBookFunc.bookList.get(i).getStock();
            num++;
            AdminBookFunc.bookList.get(i).setStock(num);
            BorrowBookFunc.borrowBook.remove(LoginFunc.getId());
        }
```

11) **MyPageView**: 내 정보를 보여주는 기능을 담고 있는 클래스

가) showMyPage(): 마이페이지를 보여주는 메서드로 ① 내 회원 정보와 ② 대출 현황을 보여주고, 회원 정보를 수정할 수 있는 MypageFunc의 메서드들과 연결되어 있습니다.

12) **MyPageFunc**: 마이페이지를 구축하는데 필요한 메서드를 담고 있는 클래스

가) printMyInfo(): 내 아이디와 일치하는 회원 정보를 출력해줍니다.

```
// 내 회원정보를 보여주는 기능
public void printMyInfo() {
    for (int i = 0; i < RegistrationFunc.userList.size(); i++) {
        if (RegistrationFunc.userList.get(i).getId().equals(LoginFunc.getId())) {
            System.out.println(RegistrationFunc.userList.get(i));
        }
    }
}
```

나) changeUserList(): 관리자가 도서 정보를 변경하는 원리를 그대로 적용했습니다.

다) returnDateCalculation(): 도서 반납 일자를 계산해주는 메서드입니다.

2. 파일 입출력(IO)

파일 입출력은 In-Memory와 거의 같지만 다음과 같은 차이를 보입니다.

가. AdminView에서 ① 정보 불러오기와 ② 파일 저장하기 항목이 신설

```
public void adminShowMenu() {
    System.out.println("=====================================================================");
    System.out.println("\t \t \t <관리자 메뉴>");
    System.out.println("=====================================================================");
    System.out.printf("*관리자님 환영합니다!%n");

    System.out.println("1. 정보 불러오기");
    System.out.println("2. 회원 목록");
    System.out.println("3. 도서 등록");
    System.out.println("4. 대출 및 반납 현황");
    System.out.println("5. 파일 저장하기");
    System.out.println("---------------------------------------------------------------------");
    System.out.println("6. 관리자 로그아웃");
```

나. AdminFileSaveFunc 클래스 신설

1) printSaveUserList(): FileWriter, BufferedFileWriter, PrintWriter를 이용하여 인메모리에 있는 회원정보를 DATA 폴더 아래에 있는 UserInfo.txt에 텍스트 파일 형식으로 저장합니다. 향후 split을 통해 데이터를 불러올 예정이므로 ","를 통해 데이터를 구분해 저장했습니다.

```
public void printSaveUserList() {
    FileWriter fw = null;
    BufferedWriter bw = null;
    PrintWriter pw = null;

    try {
        fw = new FileWriter("DATA/UserInfo.txt");
        bw = new BufferedWriter(fw);
        pw = new PrintWriter(bw);

        for (int i = 0; i < RegistrationFunc.userList.size(); i++) {
            System.out.println(RegistrationFunc.userList.get(i));
            pw.printf("%s,%s,%s,%s,%s%n", RegistrationFunc.userList.get(i).getId(),
                    RegistrationFunc.userList.get(i).getPassword(), RegistrationFunc.userList.get(i).getName(),
                    RegistrationFunc.userList.get(i).getBirthDay(), RegistrationFunc.userList.get(i).getEmail());
        }
```

2) printSaveBookList(): 인메모리에 있는 도서정보를 DATA 폴더 아래에 있는 BookInfo.txt에 텍스트 파일 형식으로 저장합니다.

3) printSaveBookList(): 인메모리에 있는 전체 회원의 도서 대출 정보를 DATA 폴더 아래에 있는 RentInfo.txt에 텍스트 파일 형식으로 저장합니다.

```
for (int i = 0; i < RegistrationFunc.userList.size(); i++) {
    pw.printf("%s,%s%n", RegistrationFunc.userList.get(i).getId(),
            BorrowBookFunc.borrowBook.get(RegistrationFunc.userList.get(i).getId()));
}
```

다. AdminFileView 클래스 신설

1) showFileSave(): 모든 회원의 활동이 끝나면 관리자가 프로그램을 종료하기 전에 관리자 메인페이

지의 ⑤ 파일 저장하기를 선택하면, 인메모리의 정보를 저장되었음을 시각적으로 보여주는 기능입니다.

2) showFileRead(): 지난 저장 시점부터 그대로 이어받아 인메모리를 운영하기 위해 UserInfo.txt, BookInfo.txt, RentInfo.txt를 불러오고, 제대로 파일을 입력받았는지 확인하기 위해 그 내용을 출력해줍니다.

라. AdminFileReadFunc 클래스 신설

1) readUserFile(): DATA/UserInfo.txt에 있는 정보를 읽어온 후 split(“,”)을 통해 데이터를 arr 배열에 저장하고 그 값을 다시 List ‘userList’에 저장합니다. 그리고 잘 입력되었는지 확인하기 위해 인메모리의 ‘userList’를 출력해줍니다.

```
public void readUserFile() {
    try {
        List<String> lines = Files.readAllLines(Paths.get("DATA/UserInfo.txt"));
        for (int j = 0; j < lines.size(); j++) {
            String[] arr = lines.get(j).split(",");
            RegistrationFunc.userList.add(new UserInfo1(arr[0], arr[1], arr[2], arr[3], arr[4]));

        }
        for (int i = 0; i < RegistrationFunc.userList.size(); i++) {
            System.out.println(RegistrationFunc.userList.get(i));
        }

    } catch (Exception e) {
        e.printStackTrace();
```

2) readBookFile(): DATA/BookInfo.txt에 있는 정보를 읽어온 후 split(“,”)을 통해 데이터를 arr 배열에 저장하고 그 값을 다시 List ‘bookList’에 저장합니다. 그리고 잘 입력되었는지 확인하기 위해 인메모리의 ‘bookList’를 출력해줍니다.

```
public void readBookFile() {
    try {
        List<String> lines = Files.readAllLines(Paths.get("DATA/BookInfo.txt"));
        for (int j = 0; j < lines.size(); j++) {
            String[] arr = lines.get(j).split(",");
            int num = Integer.parseInt(arr[4]);
            AdminBookFunc.bookList.add(new BookInfo(arr[0], arr[1], arr[2], arr[3], num));

        }
        for (int i = 0; i < AdminBookFunc.bookList.size(); i++) {
            System.out.println(AdminBookFunc.bookList.get(i));
        }

    } catch (Exception e) {
        e.printStackTrace();
```

3) readRentFile(): DATA/RentInfo.txt에 있는 정보를 읽어온 후 split(“,”)을 통해 데이터를 arr 배열에 저장하고 그 값을 다시 HashMap ‘borrowBook’에 저장합니다. 그리고 잘 입력되었는지 확인하기 위해 인메모리의 ‘borrowBook’를 출력해줍니다.

```
public void readRentFile() {
    try {
        List<String> lines = Files.readAllLines(Paths.get("DATA/RentInfo.txt"));
        for (int i = 0; i < lines.size(); i++) {
            String[] arr = lines.get(i).split(",");
            if(arr[1].startsWith(AdminBookFunc.bookList.get(i).getBooknumber())) {
                BorrowBookFunc.borrowBook.put(arr[0], AdminBookFunc.bookList.get(i));
            }
        }

        Set set = BorrowBookFunc.borrowBook.entrySet();
        Iterator it = set.iterator();

        while (it.hasNext()) {
            Map.Entry e = (Map.Entry) it.next();
            System.out.println("아이디: " + e.getKey());
            System.out.println(e.getValue());
        }

    } catch (Exception e) {
        e.printStackTrace();
```

Ⅲ. 느낀점 및 향후 보완점

1. 문제를 하나씩 해결해 나가는 과정에서 조금씩 성장하고 있는 나를 발견할 수 있었습니다.
2. Serialize, 상속, 인터페이스 및 설계에 대해 좀 더 배우고 싶습니다.
3. try-catch문을 통한 예외처리를 통해 여러 버그들을 더 효과적으로 관리해나가고 싶습니다.