도서관리 프로그램
[인 메모리 / 파일 입출력]

2260341020 조재희

1. 프로젝트 소개

도서관리 프로그램을 자바 언어를 이용해서 프로그램이 실행되는 동안 정보가 추가되고 삭제될 수 있는 인 메모리 방식과 프로그램이 종료되더라도 해당 파일이 외부로 출력되어 저장되고 다시 프로그램을 실행하더라도 다시 정보를 사용할 수 있는 파일 입출력 방식으로 구현하였습니다.

인 메모리 방식은 HashMap과 ArrayList들을 적절히 사용해 그 내부에 정보들이 저장되고 프로그램이 종료되면 해당 자료구조에는 아무것도 남지 않게 됩니다. 파일 입출력 방식은 이미 자바에 구현되어 있는 입출력 API와 객체 직렬화, ArrayList들을 사용하여 구현할 수 있습니다.

인메모리 방식에 입출력 API를 사용해서 할 수도 있었지만 Map을 사용해볼 기회가 적어 인메모리 방식은 Map을 이용해 구현했고, 인메모리 방식은 List 방식으로 구현하였습니다.

2. 프로젝트 목적

a. 자바 언어에 대한 전반적인 이해를 하는 데에 목적이 있다.

b. 자바에서 제공하는 각종 API들을 사용할 수 있었고 배웠던 자료 구조들의 특성과 쓰임들에 대해 이해할 수 있다.

c. 수업 시간에 배웠던 클래스, 상속, 인터페이스, 메소드 구현 방식, 각종 반복문 등에 관해 언제 어떤 방식으로 언제 사용해야 하는지 이해할 수 있다.

3. 프로젝트 설명

가. 인 메모리 방식

```
System.out.println("========== 재희네도서관 ==========");
System.out.println("1.회원가입");
System.out.println("2.책 빌리기");
System.out.println("3.책 반납");
System.out.println("4.대출가능한 도서 목록");
System.out.println("5.비밀번호 확인");
System.out.println("6.비밀번호 수정");
System.out.println("6.(회원만)회원정보 보기");
System.out.println("7.(매니져만)회원정보 보기");
System.out.println("8.회원정보 탈퇴");
System.out.println("9.(매니져만)책 입고");
System.out.println("10.프로그램 종료");
```

a. 방식

인 메모리 방식에 구현된 기능은 다음과 같이 회원들의 회원가입을 받을 수 있는 회원가입과 탈퇴, 책 대출과 반납, 대출 가능한 도서목록의 확인, 비밀번호의 확인과 수정, 그리고 로그인을 하면 자기 자신의 대출 이력에 대한 정보를 볼 수 있고, 매니저의 경우 회원들이 연체를 하였을 경우 연락할 수 있도록 매니저 창을 따로 만들어 매니저로 로그인 할 경우 회원들의 정보를 볼 수 있게 했습니다. 그리고 매니저 로그인을 할 경우 새로운 책의 입고와 제거를 할 수 있도록 기능들을 구현했습니다.

b. 전체적인 구조

전체적인 구조는 다음과 같습니다. 메인창에서 원하는 기능을 선택할 수 있도록 switch 반복문을 사용했습니다. 그렇게 회원들과 매니저들은 자신들이 원하는 기능을 선택하고 사용할 수 있게 됩니다. while(true) 반복문을 통해서 계속 원하는 기능들을 선택할 수 있도록 하였고, 프로그램의 종료를 원할 경우 종료에 해당하는 버튼을 눌러 프로그램을 종료할 수 있도록 하였습니다. 그렇게 되면 계속해서 자신들이 회원 가입한 정보나 책을 입고한 정보들은 Map과 List들에 계속해서 저장이 되게 되고 프로그램이 종료되면 해당 정보들은 저장이 되지 않고 초기화됩니다.

```
start : while(true) {
System.out.println("원하시는 항목을 선택하주세요.");
switch(sc.nextInt()){
    case 1:
                    System.out.println("==================================회원가입==================================");
        Join2.join(joinlist,loginlist,booklist);
        //이거 자체도 예시적용변환 등. 여에서 이미 지정하보으므로.

        System.out.println("계속 하고 진행하고 싶으면 Y를 그만 끝나려면 N를 누르세요.");
        sc.nextLine();

        if(sc.nextLine().equalsIgnoreCase("Y")) {
                continue;
        }else if(!sc.nextLine().equalsIgnoreCase("Y")) {
                break start;
        }

    case 2:
                    System.out.println("==================================책 대여==================================");
        View.borrow(loginlist,booklist,bookmanage);

        System.out.println("계속 하고 진행하고 싶으면 Y를 그만 끝나려면 N를 누르세요.");
        sc.nextLine();
        if(sc.nextLine().equalsIgnoreCase("Y")) {
                continue;
        }else if(!sc.nextLine().equalsIgnoreCase("Y")) {
                break start;
        }

    case 3:
                    System.out.println("==================================책 반납==================================");
        View.turnin(loginlist,booklist,bookmanage);

        System.out.println("계속 하고 진행하고 싶으면 Y를 그만 끝나려면 N를 누르세요.");
        sc.nextLine();
        if(sc.nextLine().equalsIgnoreCase("Y")) {
                continue;
        }else if(!sc.nextLine().equalsIgnoreCase("Y")) {
                break start;
        }
```

```
public class LibraryMain {
    public static void main(String[] args){
        HashMap<Member02,Member> joinlist=new HashMap<>();
        HashMap<Member,Member> loginlist=new HashMap<>();
        HashMap<Member, Book> booklist=new HashMap<>();
                HashMap<Manager,String> managerlist=new HashMap<>();
                ArrayList<Book> bookmanage=new ArrayList<>();//책 이름, 지은이
        Scanner sc =new Scanner(System.in);
```

이런 식으로 HashMap으로 구현한 회원 정보들이 담긴 joinlist와 로그인을 할 때 사용하는 loginlist 해당 유저들의 대출 정보들을 담고 있는 booklist 들이 존재하고 매니저창에서 입고된 책에 대한 정보를 담고 있는 ArrayList로 구현한 managerlist들이 존재하게 됩니다.

```
joinlist.put(new Member02(name, identity), new Member(id,password,phonenumber));
loginlist.put(new Member(id,password),new Member(name,identity,phonenumber));
booklist.put(new Member(id,password),new Book("없음","없음","없음","없음"));
```

```
public Book(String bookname, String writer, String bookhistory, String date) {
    super();
    this.bookname = bookname;
    this.writer = writer;
    this.bookhistory = bookhistory;
    this.date=date;
}
public Book(String bookname, String writer) {
```

```
    }
    public Book(String bookname, String writer) {
        super();
        this.bookname = bookname;
        this.writer = writer;
    }
```

위와 같이 HashMap인 joinlist,loginlist 에는 Member들이 키와 value로 들어가 있습니다.

joinlist에는 Member(이름,주민등록번호)가 Key가 되며 Member(아이디,비밀번호,휴대폰번호)가 Value가 됩니다.
HashMap인 loginlist에는 Member(아이디,비밀번호)가 Key가 되며 Member(이름,주민등록번호,휴대폰번호)가 Value가 됩니다.
HashMap인 booklist 에는 Member가 키로 Book이 value로 들어가 있습니다. Member에는 아이디, 비밀번호가 Book에는 빌린 책, 빌린 책의 작가 이름, 대여 이력, 빌린 날짜들이 저장이 됩니다.
ArrayList인 managerlist에는 Book이 저장이 되며 Book(책 이름, 작가 이름)이 저장이 되게 됩니다.

c. 기능

-회원가입

```
public class Join2 {

    public static void join(Map<Member02,Member> joinlist,Map<Member,Member>loginlist, Map<Member,Book> booklist){

        Scanner sc =new Scanner(System.in);
        System.out.println("\n이름을 입력하세요\n");
        String name=sc.nextLine();
        System.out.println("\n주민번호를 -없이 입력하세요\n");
        String identity=sc.nextLine();
        System.out.println("\n아이디를 입력하세요\n");
        String id=sc.nextLine();
        System.out.println("\n휴대폰번호를 -없이 입력하세요\n");
        String phonenumber=sc.nextLine();
        System.out.println("\n비밀번호를 입력하세요\n");
        String password=sc.nextLine();
        if(!joinlist.containsKey(new Member(name,identity))) {
            joinlist.put(new Member02(name, identity), new Member(id,password,phonenumber));
            loginlist.put(new Member(id,password),new Member(name,identity,phonenumber));
            booklist.put(new Member(id,password),new Book("없음","없음","없음"));
            System.out.println("\n회원가입이 완료되었습니다");
        }else if(joinlist.containsKey(new Member(name,identity))){
            System.out.println("이미 회원입니다. 도서관을 이용해주세요");
        }
```

회원가입을 하게되면은 joinlist와 loginlist booklist에 해당 정보들이 위와 같이 저장이 됩니다. 그렇게 되면 joinlist는 나중에 매니저들이 회원들의 대여 정보나 휴대폰번호를 위에서 저장한 Key(이름, 주민등록번호)들을 이용해 찾을 수 있고 loginlist를 통해서는 Key(아이디, 비밀번호)를 이용해 loginlist가 해당 정보를 포함하고 있으면 로그인이 되고 정보를 포함하지 않으면 로그인이 되지 않게 됩니다.
booklist에도 Key(아이디, 비밀번호)가 되게 되며 대출과 반납 등에도 아이디와 비밀번호를 사용할 수 있고 추후에 대출 정보들을 확인할 수 있게 됩니다.

-대출과 반납 기능

```
5
6
7
8       //책 빌린기 시스템 빌린 책, 빌린 날짜, 빌림으로 바꿔버림.
9     public static void borrow(HashMap<Member, Member> loginlist, HashMap<Member,Book> booklist, ArrayList<Book> bookmanage) {
10        int sum = 0;
11        Scanner sc = new Scanner(System.in);
12        System.out.println("ID를 입력하세요");
13        String id = sc.nextLine();
14        System.out.println("비밀번호를 입력하세요");
15        String password = sc.nextLine();
16
17        if (loginlist.containsKey(new Member(id, password)) && booklist.get(new Member(id, password)).getBookhistory().equals("완료")) {
18            System.out.println("빌리고 싶은 책을 입력하세요.");
19            String bookname = sc.nextLine();
20            System.out.println("빌리고 싶은 책의 작가를 입력하세요");
21            String writer = sc.nextLine();
22            for (int i = 0; i < bookmanage.size(); i++) {
23                if (bookmanage.get(i).getBookname().equals(bookname) && bookmanage.get(i).getWriter().equals(writer))
24                    sum++;
25            }
26
27            if (sum != 0) {
28                Date today = new Date();
29                String pattern = "yyyy-MM-dd hh:mm:ss(E)";
30                SimpleDateFormat sdf = new SimpleDateFormat(pattern);
31                String date = sdf.format(today);
32                String bookhistory = "빌림";
33                System.out.println(bookname + "를 대여하였습니다.");
34                System.out.println("대여날짜는" + date + "입니다." + "대여처리되었습니다.");
35                bookmanage.remove(new Book(bookname,writer));
36                booklist.put(new Member(id, password), new Book(bookname, writer, bookhistory, date));
37            } else {
38                System.out.println("도서 목록에 없습니다");
39            }
40        } else if (booklist.get(new Member(id, password)).getBookhistory().equals("빌림")) {
41            System.out.println("대여 이력이 있습니다. 먼저 반납해주세요");
42        } else if (!loginlist.containsKey(new Member(id, password))) {
43            System.out.println("회원이 아닙니다 회원가입을 해주세요");
44        }
```

아이디와 비밀번호로 로그인을 하고 대출 이력이 없다면 저장되어 있는 책 정보 중에서 책을 선택할 수 있게 됩니다. 대출을 하게 되면은 대여한 시간과 대여한 책, 대여한 책의 작가, 대출 이력 등이 저장이 되게 됩니다. 동시에 managerlist에 있던 책은 사라지게 됩니다.

```
//책반납하기 시스템 // 반납하면 책정보 싹다 바꿔버림. 반납완료, 반납시간, 안빌림
public static void turnin(HashMap<Member,Member> loginlist,HashMap<Member,Book> booklist,ArrayList<Book> bookmanage) {

    Scanner sc=new Scanner(System.in);
    System.out.println("ID를 입력하세요");
    String id=sc.nextLine();
    System.out.println("비밀번호를 입력하세요");
    String password=sc.nextLine();
    if(loginlist.containsKey(new Member(id,password))&&booklist.get(new Member(id,password)).getBookhistory().equals("빌림")) {
        System.out.println("반납하실 겁니까? Y or N");
        String answer=sc.nextLine();
        if(answer.equals("Y")) {
            String bookname="반납완료";
            Date today=new Date();
            String pattern = "yyyy-MM-dd hh:mm:ss(E)";
        SimpleDateFormat sdf=new SimpleDateFormat(pattern);
        String date=sdf.format(today);
        System.out.println("반납시간은"+date+"입니다.반납처리되었습니다");
            String bookhistory="완료";
        String booknamel=booklist.get(new Member(id,password)).getBookname();
        String writerl=booklist.get(new Member(id,password)).getWriter();
            booklist.put(new Member(id,password),new Book(booknamel,writerl,bookhistory,date));
        bookmanage.add(new Book(booknamel,writerl));
        }
    }else if(!loginlist.containsKey(new Member(id,password))){
         System.out.println("없는 정보입니다. 회원가입을 해주세요");
    }else if(booklist.get(new Member(id,password)).getBookhistory().equals("완료")){
        System.out.println("대여 이력이 없습니다. 반납할것이 없습니다.");
    }


}
 }
```

대출 이력이 있는 사람은 로그인을 통해 반납이 가능하게 되며, 반납을 하게 되면 반납을 한 책들은 managerlist에 다시 돌아오고, 반납한 날짜들이 기록이 되고 대여 전 상태로 돌아오게 됩니다.

managerlist는 ArrayList의 특성에 따라 책의 이름과 작가들이 동일한 책들이 여러 권 들어갈 수 있고, remove라는 메소드를 사용하게 되면 하나씩 사라지게 할 수 있습니다.

-대여 가능한 책 목록 보기

```
import java.util.*;

public class ShowBookList {
    public static void showBooklist(HashMap<Member,Member> loginlist, ArrayList<Book> bookmanage){
        Scanner sc=new Scanner(System.in);
        System.out.println("아이디를 입력하세요");
        String id=sc.nextLine();
        System.out.println("비밀번호를 입력하세요");
        String password=sc.nextLine();
        if(loginlist.containsKey(new Member(id,password))){
            System.out.println("==================대출 가능한 도서 목록입니다.=======================");
            for(int i=0;i<bookmanage.size();i++){
                System.out.println(bookmanage.get(i).getBookname()+bookmanage.get(i).getWriter());

            }
        }else{
            System.out.println("회원정보가 없습니다");
        }



    }

}
```

로그인을 하게 되면은 대여 가능한 책 목록을 managerlist를 통해서 볼 수 있게 됩니다. 고객은 해당 목록을 확인하고 대여를 할 수 있습니다.

-비밀번호 찾아주기

```
//비밀번호 찾아주기 시스템 비밀번호 뿌려줌.
public static void checkout(HashMap<Member02,Member> joinlist) {
        Scanner sc =new Scanner(System.in);
        System.out.println("이름을 입력해주세요");
        String name=sc.nextLine();
        System.out.println("주민번호를 -없이 입력해주세요");
        String identity=sc.nextLine();
        if(joinlist.containsKey(new Member02(name,identity))){
                System.out.println(joinlist.get(new Member02(name,identity)).getPassword());
        }else {
                System.out.println("입력하신 정보가 올바르지 않습니다. 다시 확인해주세요");
        }

}
```

회원들은 비밀번호를 잊어 버릴 경우 자신의 이름과 주민등록번호를 통해 joinlist에 접근할 수 있고 비밀번호를 확인할 수 있습니다.

-비밀번호 변경하기

```
    //비밀번호 변경해주기 시스템
    public static void modify(HashMap<Member02,Member> joinlist,HashMap<Member,Member> loginlist,HashMap<Member,Book> booklist) {

            Scanner sc=new Scanner(System.in);
            System.out.println("\n아이디를 입력해주세요\n");
            String id=sc.nextLine();
            System.out.println("\n비밀번호를 입력해주세요\n");
            String password=sc.nextLine();
            if(loginlist.containsKey(new Member(id,password))){
                    String name=loginlist.get(new Member(id,password)).getName();
                    String identity=loginlist.get(new Member(id,password)).getIdentity();
                    String phonenumber=loginlist.get(new Member(id,password)).getPhonenumber();
                    String bookname=booklist.get(new Member(id,password)).getBookname();
                    String writer=booklist.get(new Member(id,password)).getWriter();
                    String bookhistory=booklist.get(new Member(id,password)).getBookhistory();
                    String date=booklist.get(new Member(id,password)).getDate();
                    System.out.println("새 비밀번호를 입력해주세요");
                    String newPassword=sc.nextLine();
                    joinlist.put(new Member02(name,identity),new Member(id,newPassword,phonenumber));
                    loginlist.put(new Member(id,newPassword),new Member(name,identity,phonenumber));
                    booklist.put(new Member(id,newPassword),new Book(bookname,writer,bookhistory,date));

                    joinlist.remove(new Member02(name,identity));
                    loginlist.remove(new Member(id,password));
                    booklist.remove(new Member(id,password));
            }else {
```

회원들은 로그인하여 자신의 비밀번호를 변경할 수 있습니다. 이 때 회원들의 정보가 담긴 joinlist, loginlist, booklist들이 동시에 변경을 해 새로운 정보들이 담길 수 있게 합니다.

기존에 있던 정보들은 삭제합니다.

-회원 정보 보기

```
public class Information {

    @Override
    public String toString() {
        return "Information{}";
    }

    public static void show(HashMap<Member,Member> loginlist, HashMap<Member,Book> booklist){
        Scanner sc=new Scanner(System.in);
        System.out.println("아이디를 입력하세요");
        String id=sc.nextLine();
        System.out.println("비밀번호를 입력하세요");
        String password=sc.nextLine();
        if(loginlist.containsKey(new Member(id,password))){
            System.out.println("빌린 책"+booklist.get(new Member(id,password)).getBookname());
            System.out.println("내역"+booklist.get(new Member(id,password)).getBookhistory());
            System.out.println("빌린 날짜"+booklist.get(new Member(id,password)).getDate());
            System.out.println("빌린 책의 작가"+booklist.get(new Member(id,password)).getWriter());

        }else {
                System.out.println("정보를 다시 입력하세요. 일치하는 정보가 없습니다.");
        }
```

회원들은 자신의 아이디와 비밀번호를 입력하게 되면은 자신의 대출 정보(빌린 책, 빌린 내역, 빌린 날짜, 빌린 책의 작가)를 확인할 수 있게 됩니다.

```
    //책 뭐 빌렸는지 보여줌.
public static void showAll(HashMap<Manager,String> managerlist, HashMap<Member,Book> booklist) {
    Scanner sc =new Scanner(System.in);
    System.out.println("매니저 id를 입력하세요");
    String id=sc.nextLine();
    System.out.println("매너지 비밀번호를 입력하세요");
    String password=sc.nextLine();
    if(managerlist.containsKey(new Manager(id))&&managerlist.get(new Manager(id)).equals(password)){
            Set<Map.Entry<Member, Book>> set = booklist.entrySet();
            for (Map.Entry<Member, Book> e : set) {
                    System.out.println(e.getKey().getId()+"\t"+e.getKey().getPassword() + "\t" + e.getValue().getBookname()+"\t"+e.getValue().getWriter()+"\t"+e.ge
            }
    }else {
            System.out.println("관리자가 아닙니다. 돌아가주세요");
    }

}
```

매니저 클래스에서는 매니저 아이디와 비밀번호를 입력하게 되면은 지금까지 회원들이 대출했던 정보들을 확인할 수 있게 됩니다. 이를 이용해 연체된 회원들에게 연락이 가능합니다.

-책의 입고

```
public static void bookcare(HashMap<Manager, String> managerlist, ArrayList<Book> bookmanage) {
    Scanner sc = new Scanner(System.in);
    System.out.println("매니저 id를 입력하세요");
    String id = sc.nextLine();
    System.out.println("매너지 비밀번호를 입력하세요");
    String password = sc.nextLine();
    if (managerlist.containsKey(new Manager(id)) && managerlist.get(new Manager(id)).equals(password)) {
        System.out.println("1번을 누르면 새로 들어온 책을 새로 입고시킬 수 있습니다. 2번을 명단에서 책을 제거할 수 있습니다");
        int selection = sc.nextInt();
        if (selection == 1) {
            sc.nextLine();
            System.out.println("입고시킬 책을 입력하세요");
            String bookname = sc.nextLine();
            System.out.println("입고시킬 책의 작가를 입력하세요");
            String writer = sc.nextLine();
            bookmanage.add(new Book(bookname,writer));
        } else if (selection == 2) {
            sc.nextLine();
            System.out.println("도서명단에서 제외할 책을 입력하세요");
            String bookname1 = sc.nextLine();
            System.out.println("도서명단에서 제외할 책을 입력하세요");
            String writer1 = sc.nextLine();
            System.out.println("도서명단에서 제외할 책을 입력하세요");
            String specialCode1 = sc.nextLine();
            bookmanage.remove(new Book(bookname1,writer1));
        }
```

매니저 아이디와 비밀번호로 로그인하게 되면은 책을 추가할 수도 책을 없앨 수도 있습니다. 입고시킬 책의 이름과 작가를 입력하여 입고시키도록 합니다.

-회원 탈퇴

```java
public class Rm {

    public static void removeInformation(HashMap<Member02,Member> joinlist,HashMap<Member,Member> loginlist,HashMap<Member, Book> booklist){
        Scanner sc = new Scanner(System.in);
        System.out.println("아이디를 입력하세요");
        String id=sc.nextLine();
        System.out.println("비밀번호를 입력하세요");
        String password=sc.nextLine();

        if(loginlist.containsKey(new Member(id,password))){
            String name=loginlist.get(new Member(id,password)).getName();
            String identity=loginlist.get(new Member(id,password)).getIdentity();
            joinlist.remove(new Member02(name,identity));
            loginlist.remove(new Member(id,password));
            booklist.remove(new Member(id,password));
            System.out.println("회원정보를 모두 삭제하였습니다");
        }else{
            System.out.println("회원정보가 일치하지 않습니다. 다시 확인해주세요");
        }

    }

}
```

회원들은 아이디와 비밀번호를 입력하게 되면은 자신의 정보를 삭제할 수 있게 됩니다. 아이디와 비밀번호를 입력하게 되면은 회원들의 정보를 담았던 joinlist loginlist booklist에 있던 정보들이 전부 삭제가 되게 됩니다.

나. 파일 입출력 방식

a. 방식

인 메모리 방식과 다르게 전부 ArrayList로 구현하였습니다. 그렇게 되면은 맵에 비해 저장된 객체에서 getter 함수를 이용해서 쉽게 정보들을 가져올 수 있습니다. 대신 for 문을 통해서 일일이 고객정보를 확인해야 하는 번거로움이 있지만 비교적 Map에 비해 선언해야 할 리스트들이 적었습니다.

```java
public static void bookwrite(ArrayList<Book> book) {


    try {
        String fileName= "list/booklist.txt";

        FileOutputStream fos = new FileOutputStream(fileName,false);
        BufferedOutputStream bos = new BufferedOutputStream(fos);
        ObjectOutputStream out = new ObjectOutputStream(fos);


        out.writeObject(book);


        System.out.println("booklist.txt 저장완료...");
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
    }

}
public static void bookwrite() {
```

```
public class HelpTools {
    public static void joinwrite(ArrayList<Member> member) {


        try {
            String fileName= "list/joinlist.txt";

            FileOutputStream fos = new FileOutputStream(fileName,false);
            BufferedOutputStream bos = new BufferedOutputStream(fos);
            ObjectOutputStream out = new ObjectOutputStream(fos);
            ArrayList<Member> list = new ArrayList<>();


            out.writeObject(member);


            System.out.println("joinlist.txt 저장완료...");
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        }
    public static void joinwrite() {
```

위 방식처럼 파일 입출력 방식은 컴퓨터 내에 있는 정보들을 가져오고 다시 프로그램이 종료되면 작성했던 정보들이 저장되게 됩니다.

인메모리 방식과 다르게 Member(이름, 주민등록번호, 아이디, 비밀번호, 휴대폰번호)가 저장된 joinlist와 Book(아이디, 비밀번호, 책 이름, 책의 작가, 책의 고유번호, 빌린 날짜, 빌린 이력)이 저장된 Booklist, 입고된 책의 정보(책 이름, 책의 고유번호, 책의 작가)가 저장된 managerlist를 사용하게 됩니다. 각각에 접근할 때는 getter 함수를 따로 클래스에 정의해 접근을 하게 됩니다.

joinlist boolist managerlist 들은 객체 직렬화 개념을 통해 txt 파일로 컴퓨터 내에 저장되게 됩니다. 그리고 다시 불러 올때는 역직렬화 개념을 통해 다시 정보를 불러오게 됩니다.

b. 전체적인 구조

```
System.out.println("=============================================================================================================================================");
System.out.println("================================================================== 재희네도서관 ==================================================================");
System.out.println("1.회원가입");
System.out.println("2.책 빌리기");
System.out.println("3.책 반납");
System.out.println("4.대출 가능한 책 목록 확인");
System.out.println("5.비밀번호 찾기");
System.out.println("6.(매니저만)책 입고");
System.out.println("7.(매니저만)책 제거");
System.out.println("8.자기 정보 보기");
System.out.println("9.회원탈퇴");
System.out.println("10.프로그램 종료");
System.out.println("11.(절대 누르지 마시오.)모든 정보 초기화");
System.out.println("12.프로그램 종료");
System.out.println("=============================================================================================================================================");
System.out.println("=============================================================================================================================================");

ArrayList<BookManager> list=BookManager.bookinfo();
start:
while (true) {
    System.out.println("원하시는 항목을 클릭해주세요.");
```

```
public static void bookwrite(ArrayList<Book> book) {


    try {
        String fileName= "list/booklist.txt";

        FileOutputStream fos = new FileOutputStream(fileName,false);
        BufferedOutputStream bos = new BufferedOutputStream(fos);
        ObjectOutputStream out = new ObjectOutputStream(fos);


        out.writeObject(book);


        System.out.println("booklist.txt 저장완료...");
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
    }

}
public static void bookwrite() {
```

인 메모리와 마찬가지로 자신이 원하는 기능을 숫자를 눌러서 선택할 수 있고, 작성한 정보들은 ArrayList에 저장이 되고, ArrayList들은 파일 입출력 API에 의해 파일 형태로 컴퓨터에 저장이 되게 됩니다.

c.기능

-회원가입

```
int sum=0;
Scanner sc = new Scanner(System.in);
System.out.println("이름을 입력해주세요");
String name = sc.nextLine();
System.out.println("주민번호를 입력해주세요");
String identity = sc.nextLine();
System.out.println("휴대폰번호를 -없이 입력해주세요");
String phonenumber = sc.nextLine();
System.out.println("아이디를 입력해주세요");
String id = sc.nextLine();
System.out.println("비밀번호를 입력해주세요");
String password = sc.nextLine();
String bookhistory = "대여 이력 없음";
String bookname = "없음";
String date = "없음";
System.out.println(memberlist.get(0).getId().equals(id));
    for(int i=0;i<memberlist.size();i++){
        if((memberlist.get(i).getId()).equals(id)&&(memberlist.get(i).getPassword().equals(password))){
            sum++;
        }
    }
    if(sum==0){
        memberlist.add(new Member(name,identity,id,password,phonenumber));
        booklist.add(new Book(id,password,"없음","없음","없음","없음","대여 이력 없음"));
        HelpTools.joinwrite(memberlist);
        HelpTools.bookwrite(booklist);
        System.out.println("회원가입이 완료되었습니다");;
    }else{
        System.out.println("이미 회원입니다");
    }
```

회원가입시 회원들은 이름, 주민등록번호, 휴대폰번호, 아이디, 비밀번호를 입력하게 되며 입력을 통해 동일한 회원이 있는지 확인하고 없을 경우 회원가입이 진행되어 리스트(joinlist, booklist)에 저장되고 리스트들은 때 미리 만들어둔 입출력 API를 통해 파일에 출력이 되게 됩니다.

이 때 serializable 인터페이스를 상속받아 직렬화와 역직렬화가 사용되게 됩니다.

-책 대출 . 반납

```
int login1 = 0;
int bookborrow = 0;
ArrayList<BookManager> managerlist = BookManager.bookinfo();
ArrayList<Book> booklist = HelpTools.bookread();
Scanner sc = new Scanner(System.in);
System.out.println("아이디를 입력해주세요");
String id = sc.nextLine();
System.out.println("비밀번호를 입력해주세요");
String password = sc.nextLine();
Date today = new Date();
String pattern = "yyyy-MM-dd hh:mm:ss(E)";
SimpleDateFormat sdf = new SimpleDateFormat(pattern);
String date = sdf.format(today);
System.out.println(booklist.size());
System.out.println(booklist.get(0).getId().equals(id));
for (int i = 0; i < booklist.size(); i++) {
    System.out.println("for문 작동중");
    if (booklist.get(i).getId().equals(id) && booklist.get(i).getPassword().equals(password)) {
        login1++;
        bookborrow = i;
    }
}
System.out.println(login1);
if (login1 == 0) {
    System.out.println("회원 정보가 일치하지 않습니다.");
} else {
    System.out.println("회원입니다.");
    if (login1 != 0 && booklist.get(bookborrow).getBookhistory().equals("대여 이력 없음")) {
        System.out.println("빌리고 싶은 책을 입력하세요.");
        String bookname = sc.nextLine();
```

```
        System.out.println("회원입니다.");
        if (login1 != 0 && booklist.get(bookborrow).getBookhistory().equals("대여 이력 없음")) {
            System.out.println("빌리고 싶은 책을 입력하세요.");
            String bookname = sc.nextLine();
            for (int j = 0; j < managerlist.size(); j++) {
                if (managerlist.get(j).getBookname().equals(bookname)) {
                    booklist.add(new Book(booklist.get(bookborrow).getId(), booklist.get(bookborrow).getPassword(), bookname, managerlist.get(j).getWriter(), managerlist.g
                    booklist.remove(bookborrow);
                    managerlist.remove(j); // 리스트는 동일한 책이 있어도 맨 앞줄부터 하나씩 제거되는 걸 이용함.
                    System.out.println(bookname + "를" + date + "에 대여처리하였습니다");
                    HelpTools.bookwrite(booklist);
                    BookManager.bookadd(managerlist);
                    break;
                }
            }
        } else if (login1 != 0 && booklist.get(bookborrow).getBookhistory().equals("대여")) {
            System.out.println("이미 대여한 이력이 있습니다.");
        }
    }
}

public static void turnin() {
```

회원들은 아이디와 비밀번호를 확인 받게 되며 회원가입을 했고, 대여 이력이 없을 경우에도 서 목록에 있는 책을 대여 받게 됩니다. 해당 책은 대여함과 동시에 사라지고 booklist가 새로 갱신되어 해당 회원의 대출 목록에 들어가게 됩니다. ArrayList의 특성을 이용하면 책의 권수가 유한하게 정해지고, 동일한 책이어도 중복이 가능하므로 쉽게 해결할 수 있습니다.

-대여 가능한 책 보기

```
public static void showBook() {
    int sum = 0;
    Scanner sc = new Scanner(System.in);
    System.out.println("아이디를 입력해주세요");
    String id = sc.nextLine();
    System.out.println("비밀번호를 입력해주세요");
    String password = sc.nextLine();
    ArrayList<Member> list1 = HelpTools.joinread();
    ArrayList<BookManager> list2 = BookManager.bookinfo();
    for (int i = 0; i < list1.size(); i++) {
        if (list1.get(i).getId().equals(id) && list1.get(i).getPassword().equals(password)) {
            sum++;
        }
    }
    if (sum == 1) {
        for (int i = 0; i < list2.size(); i++) {
            System.out.println("=======================대출가능한 책목록==============================");
            System.out.println(list2.get(i).getBookname() + list2.get(i).getUniqueNumber() + list2.get(i).getWriter());
            System.out.println("=====================================================================");
        }
    } else {
        System.out.println("회원 정보가 일치하지 않습니다. 다시 확인해주세요.");
    }


}
```

회원들은 아이디와 비밀번호를 입력하여 로그인하게 되면 대출이 가능한 도서 목록을 확인할 수 있게 됩니다. 다른 회원이 대출해 간 도서 목록은 대여할 수 없습니다.

-비밀번호 찾기

```
import java.util.ArrayList;
import java.util.Scanner;

public class ShowPassword{


    public static void showPassword() {

        Scanner sc = new Scanner(System.in);
        System.out.println("이름을 입력해주세요");
        String name = sc.nextLine();
        System.out.println("주민번호를 입력해주세요");
        String identity = sc.nextLine();
        ArrayList<Member> list=HelpTools.joinread();
        System.out.println(list.size());
        for(int i=0;i<list.size();i++){
            if((list.get(i).getName().equals(name))&&list.get(i).getIdentity().equals(identity)){
                System.out.println("비밀번호는" + list.get(i).getPassword());
                break;
            }
        }
```

회원들은 자신의 이름과 주민등록번호를 통해서 비밀번호를 찾을 수 있습니다.

-회원정보 보기

```
public static void showinformation() {

    Scanner sc = new Scanner(System.in);
    System.out.println("아이디를 입력해주세요");
    String name = sc.nextLine();
    System.out.println("비밀번호를 입력해주세요");
    String password = sc.nextLine();
    ArrayList<Book> list=HelpTools.bookread();
    for(int i=0;i<list.size();i++){
        if((list.get(i).getId().equals(name))&&list.get(i).getPassword().equals(password)){
            System.out.println(list.get(i).getBookname()+list.get(i).getBookhistory()+list.get(i).getUniqueCode()+list.get(i).getDate()+list.get(i).getWriter());
        }
    }
```

회원은 자신의 아이디와 비밀번호를 입력하면 자신의 대출 상황(책 이름, 빌린 책의 작가, 빌린 날짜, 책의 고유번호, 대여 상태)에 대해서 알 수 있습니다.

-매니저 책 입고 책 제거

```
            System.out.println("bookmanager.txt 저장완료...");
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void bookremove(ArrayList<BookManager> list) {

        try {



            String fileName = "list/bookmanager.txt";

            FileOutputStream fos = new FileOutputStream(fileName, false);
            BufferedOutputStream bos = new BufferedOutputStream(fos);
            ObjectOutputStream out = new ObjectOutputStream(fos);
            out.writeObject(list);
            System.out.println("bookmanager.txt 저장완료...");
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }


    }

//-------------------------------------------------------------------------------------------------------------

public static void bookadd(BookManager bookmanager) {


    try {
        String fileName = "list/bookmanager.txt";
        FileOutputStream fos = new FileOutputStream(fileName,false);
        BufferedOutputStream bos = new BufferedOutputStream(fos);
        ObjectOutputStream out = new ObjectOutputStream(fos);

        ArrayList<BookManager> list = bookinfo();

        list.add(bookmanager);

        out.writeObject(list);

        System.out.println("bookmanager.txt 저장완료...");
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void bookremove(ArrayList<BookManager> list) {

    try {
```

매니저 아이디와 비밀번호로 접근하면 각 매니저들은 책을 입고하거나 제거할 수 있습니다. 입고할 때는 책의 이름과 책의 작가 그리고 책의 고유번호를 입력하게 됩니다.

-회원 탈퇴

```java
import java.util.ArrayList;
import java.util.Scanner;

public class Withdrawal {

    public static void withdraw(){
        Scanner sc =new Scanner(System.in);
        System.out.println("아이디를 입력하세요");
        String id=sc.nextLine();
        System.out.println("비밀번호를 입력하세요");
        String password=sc.nextLine();
        System.out.println("휴대폰번호를 입력하세요");
        String phonenumber=sc.nextLine();
        ArrayList<Book> list1=HelpTools.bookread();
        ArrayList<Member> list2=HelpTools.joinread();
        for(int i=0;i<list2.size();i++){
            if((list2.get(i).getId().equals(id))&&(list2.get(i).getPassword().equals(password))&&list2.get(i).getPhonenumber().equals(phonenumber)){
                System.out.println("안전하게 탈퇴처리 되었습니다.");
                list2.remove(i);
                HelpTools.joinwrite(list2);
            }
        }
        for(int i=0;i<list1.size();i++){
            if(list1.get(i).getId().equals(id)&&list1.get(i).getPassword().equals(password)){
                list1.remove(i);
                HelpTools.bookwrite(list1);
            }
        }
```

회원들은 자신의 아이디와 비밀번호 휴대폰번호를 입력하면 회원 탈퇴가 가능합니다. 고객의 정보가 담겨있던 joinlist, booklist는 삭제 처리됩니다.

-회원 정보 및 모든 시스템 초기화

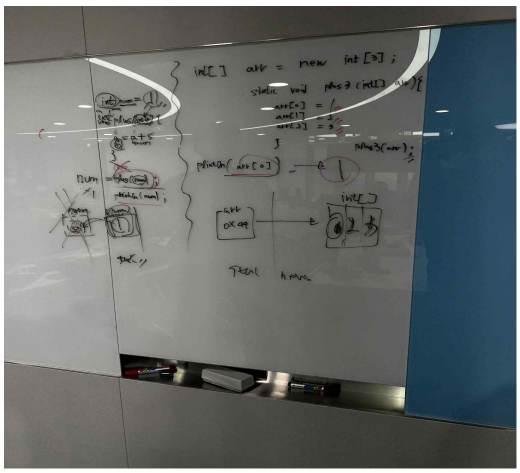
```java
case 11:
    System.out.println("====================================절대 주의====================================");
    System.out.println("====================================정보 초기화 창====================================")
    HelpTools.bookwrite();
    HelpTools.joinwrite();
    BookManager.bookadd();

    System.out.println("계속 하고 진행하고 싶으면 Y를 그만 끝내려면 N을 누르십시오.");
    sc.nextLine();
    if (sc.nextLine().equalsIgnoreCase("Y")) {
        continue;
    } else if (!sc.nextLine().equalsIgnoreCase("Y")) {
        break start;
    }
```

입출력 API에는 append에 대한 설정을 지정하는 부분이 있습니다. 덮어쓸지 말지를 결정하

는 부분이고, default로 false가 되어 있는데 이 점을 이용하면 지금까지 저장했던 부분들을 날리고 새롭게 초기화시킬 수 있습니다.

4. 프로젝트를 하며 배운 내용과 느낀 점

파일 입출력 API에서 자주 에러가 발생했다. try-catch 문에서 나는 에러들을 하나 하나 검색하고 동기들에게도 물어보며 학습했지만 아직도 의문이 생기는 부분들이 많았습니다. 제 논리 구조상에는 분명히 에러가 나면 안되는 지점인데 컴퓨터는 그렇게 생각을 하지 못 하는 부분인 듯 하여 좀 더 개발 공부를 하며 배워나가야 할 부분이라고 생각합니다.



코드를 작성하다 보니 일일이 작성하지 않아도 되는 코드를 작성하는 문제들이 있었습니다. return문을 작성하지 않아도 저장이 되는데도 return값을 굳이 써주는 그런 문제였는데 교수님과 동기들을 통해 해결할 수 있었습니다. java 내부적으로는 메소드에 int num값을 집어넣을 때 num의 값을 복사해 a에 집어넣어 a값이 바뀌기 때문에 return을 하지 않으면 num값이 바뀌지 않는 것이었고, 배열이나 Map, List 같은 경우 주소값을 복사해 넣기 때문에 굳이 return을 하지 않아도 저장이 된다는 사실이었습니다.

개념적으로는 분명히 상속, 인터페이스 등의 개념을 이해하고 있었지만 막상 프로젝트를 하면서 적용하려니 쉽지 않았습니다. 많이 사용해보고 경험을 쌓는 것이 중요하다는 생각이 들었고 다음 프로젝트에는 좀 더 효율적이고 객체지향적인 언어처럼 짜볼 수 있도록 노력해야겠다고 생각했습니다.

5. 깃헙주소와 도서관리시스템 구현 동영상

깃헙

인메모리: https://github.com/chopilyeon/LibraryFinal.git
파일 입출력: https://github.com/chopilyeon/LibraryFile.git

동영상

인메모리:https://youtu.be/qSjuKNe8xIg
파일 입출력:https://www.youtube.com/watch?v=YS1G5wik_eo