```java
package projects;

/**
 * The Class AppServer.
 */

/**
 * @author j.callado
 */
public abstract class AppServer implements Runnable {

        /* (non-Javadoc)
         * @see java.lang.Runnable#run()
         */
        @Override
        public abstract void run();

}
```

```java
package projects;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;


/**
 * The Class User.
 *
 * @author j.callado
 */
public class User implements Serializable{

        /** The user id. */
        private int userID;

        /** The user name. */
        private String userName;

        /** The password. */
        private String password;

        /** The login token. */
        private boolean loginToken;

        /** The checked out books. */
        private List<Book> checkedOutBooks = new ArrayList<Book>();

        /**
         * Instantiates a new user.
         *
         * @param userID the user id
         * @param userName the user name
         * @param password the password
         */
        public User(int userID, String userName,
                        String password){
                this.userID = userID;
                this.userName = userName;
                this.password = password;
                this.loginToken = true;
                checkedOutBooks.clear();
        }

        /**
         * Gets the book id.
         *
         * @return the book id
         */
        public int getUserID(){
                return userID;
        }

        /**
         * Gets the userName.
         *
         * @return the userName
         */
        public String getUserName(){
                return userName;
        }

        /**
         * Gets the title.
         *
         * @return the title
         */
        public String getPassword(){
                return password;
        }

        /**
         * Gets the language.
         *
         * @return the language
         */
        public boolean checkLoginToken(){
                return loginToken;
        }


        /**
         * Check out book.
         *
         * @param book the book
         */
        public void checkOutBook(Book book){
                this.checkedOutBooks.add(book);
        }

        /**
         * Check in book.
         *
         * @param book the book
         */
        public void checkInBook(Book book){
                this.checkedOutBooks.remove(book);
        }

        /**
         * Gets the checked out books.
         *
         * @return the checked out books
         */
        public List<Book> getCheckedOutBooks(){
                return checkedOutBooks;
        }

        /**
         * Sets the login token.
         *
         * @param action the new login token
         */
        public void setLoginToken(boolean action){
                this.loginToken = action;
        }


        /**
         * Prints the elements.
         */
        public void printElements(){

                System.out.println("userID: " + userID);
                System.out.println("userName: " + userName);
                System.out.println("loginToken: " + loginToken);


        }

}

```

```java
package projects;

import java.io.Serializable;

/**
 * @author j.callado
 */

/**
 * The Class Book.
 */
public class Book implements Serializable{

        /** The book id. */
        private String bookID;

        /** The author. */
        private String author;

        /** The title. */
        private String title;

        /** The language. */
        private String language;

        /** The date written. */
        private String dateWritten;

        /**
         * Instantiates a new book.
         *
         * @param bookIDString the book id string
         * @param tempBookAuthor the temp book author
         * @param tempBookTitle the temp book title
         * @param tempBookLanguage the temp book language
         * @param tempBookWritten the temp book written
         */
        public Book(String bookIDString, String tempBookAuthor,
                        String tempBookTitle, String tempBookLanguage,
                        String tempBookWritten){
                this.bookID = bookIDString;
                this.author = tempBookAuthor;
                this.title = tempBookTitle;
                this.language = tempBookLanguage;
                this.dateWritten = tempBookWritten.trim();
        }

        /**
         * Gets the book id.
         *
         * @return the book id
         */
        public String getBookID(){
                return bookID;
        }

        /**
         * Gets the author.
         *
         * @return the author
         */
        public String getAuthor(){
                return author;
        }

        /**
         * Gets the title.
         *
         * @return the title
         */
        public String getTitle(){
                return title;
        }

        /**
         * Gets the language.
         *
         * @return the language
         */
        public String getLanguage(){
                return language;
        }

        /**
         * Gets the date written.
         *
         * @return the date written
         */
        public String getDateWritten(){
                return dateWritten;
        }

        public void printElements(){

                System.out.println("bookID: " + bookID);
                System.out.println("Author: " + author);
                System.out.println("Title: " + title);
                System.out.println("Language: " + language);
                System.out.println("Date Written: " + dateWritten);

        }

}
```

```java
package projects;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;
import java.util.concurrent.atomic.AtomicInteger;

// TODO: Auto-generated Javadoc
/**
 * The Class LibraryServer.
 *
 * @author j.callado
 */

/**
 * The Class LibraryServer
 */


public class LibraryServer {

        /** The book mastercatalog. */
        private static List<Book> bookCatalog = new ArrayList<Book>();

        /** The user list. */
        public static ConcurrentMap<String, User> userList = new ConcurrentHashMap<String, User>();

        /** The total number of users with accounts on the server */
        public static AtomicInteger totalUsers = new AtomicInteger(0);

        /**
         * The main method.
         *
         * @param args the arguments
         */
        @SuppressWarnings("resource")
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Scanner bookFile;
                try {
                        bookFile = new Scanner(new File("BookList.txt")).useDelimiter(",\\s*");


                        int tempBookID = 0;
                        String bookIDString;
                        String tempBookAuthor;
                        String tempBookTitle;
                        String tempBookLanguage;
                        String tempBookWritten;
                        while(bookFile.hasNextLine()){
                                if(tempBookID < 10){
                                        bookIDString = ("0" + tempBookID);
                                        tempBookTitle = bookFile.nextLine();
                                        //System.out.println("tempBookAuthor value: " +tempBookAuthor);
                                        tempBookAuthor = bookFile.nextLine();
                                        tempBookLanguage = bookFile.nextLine();
                                        tempBookWritten = bookFile.nextLine();
                                } else {
                                        bookIDString = new Integer(tempBookID).toString();
                                        tempBookTitle = bookFile.nextLine();
                                        tempBookAuthor = bookFile.nextLine();
                                        tempBookLanguage = bookFile.nextLine();
                                        tempBookWritten = bookFile.nextLine();
                                }
                                Book book = new Book(bookIDString, tempBookAuthor, tempBookTitle, tempBookLanguage, tempBookWritten);
                                bookCatalog.add(book);
                                tempBookID++;
                        }
                        int i = 1;
                        for(Book book : bookCatalog){
                                System.out.println("Book #" + i);
                                i++;
                                book.printElements();
                        }
                        bookFile.close();
                } catch (FileNotFoundException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                }
                try {


                        ServerSocket ss = new ServerSocket(45000);
                        while (true) {
                                System.out.println("Listening on...45000");
                                Socket cs = ss.accept();

                                System.out.println("Received connection from:"+cs.getInetAddress().getHostAddress()+":"+cs.getPort());

                                LibraryAppServer app = new LibraryAppServer(cs, bookCatalog);

                                // iterative
                                //app.run();

                                // concurrent
                                System.out.println("Server Log: Creating new thread");
                                Thread t = new Thread(app);
                                t.start();

                        }


                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }
}
```

```java
package projects;

import java.io.*;
import java.net.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.atomic.AtomicInteger;



/**
 * The Class LibraryAppServer.
 */

/**
 * @author j.callado
 */

public class LibraryAppServer extends AppServer {

        /** The login token. */
        private static User user = null;

        /** The book list. */
        private static List<Book> bookList = new ArrayList<Book>();

        /** The cs. */
        private Socket cs;

        /**
         * Instantiates a new library app server.
         *
         * @param s the s
         * @param bookCatalog the book catalog
         */
        public LibraryAppServer(Socket s, List<Book> bookCatalog) {
                cs = s;
                LibraryAppServer.bookList = bookCatalog;

        }

        /**
         * Send message.
         *
         * @param spr the spr
         * @param mess the mess
         */
        public static void sendMessage(PrintWriter spr, String mess) {
                int len = mess.length();
                spr.print(String.format("%03d",len)+":"+mess);
                spr.flush();
        }

        /**
         * Recv message.
         *
         * @param sbr the sbr
         * @return the string
         */
        public static String recvMessage(BufferedReader sbr) {
                try {
                        char[] slen = new char[4];
                        sbr.read(slen,0,4);
                        int len = Integer.parseInt(new String(slen,0,3));
                        char[] sdata = new char[len];
                        sbr.read(sdata,0,len);
                        return new String(sdata);
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        return null;
                }
        }

        /**
         * New user creation.
         *
         * @param cbr the cbr
         * @param cpr the cpr
         */
        public static void newUserCreation(BufferedReader cbr, PrintWriter cpr){

                int newUserID = LibraryServer.totalUsers.getAndIncrement();
                String newPassword = null;
                String newUserName = null;


                sendMessage(cpr, "OK");
                String req = recvMessage(cbr);//Receive new userName
                if(req != null){
                        //System.out.println(req);
                        String[] tokens = req.split(":");
                        newUserName = tokens[2];
                        sendMessage(cpr,"OK");
                } else {
                        sendMessage(cpr, "ERR");
                        return;
                }

                req = recvMessage(cbr);
                if(req != null){
                        //System.out.println(req);
                        String[] tokens2 = req.split(":");
                        //System.out.println(tokens2[2]);
                        newPassword = tokens2[2];
                        sendMessage(cpr,"OK");
                } else {
                        sendMessage(cpr, "ERR");
                        return;

                }
                user = new User(newUserID, newUserName,newPassword);
                synchronized (user) {
                        LibraryServer.userList.putIfAbsent(user.getUserName(), user);
                        user = LibraryServer.userList.get(user.getUserName());
                }


                return;
        }

        /**
         * Returning user login.
         *
         * @param cbr the cbr
         * @param cpr the cpr
         */
        public static void returningUserLogin(BufferedReader cbr, PrintWriter cpr){


                String nameField = null;
                String passwordField = null;
                User userLookup =null;

```

```



		sendMessage(cpr, "OK");
		String req = recvMessage(cbr);//Receive  Returning Username
		if(req != null){
			//System.out.println(req);
			String[] tokens = req.split(":");
			nameField = tokens[2];
			sendMessage(cpr,"OK");
				userLookup = LibraryServer.userList.get(nameField);
		} else {
			sendMessage(cpr, "ERR");
			return;
		}


		req = recvMessage(cbr); // recieve password
		if(req != null){
			//System.out.println(req);
			String[] tokens2 = req.split(":");
			System.out.println(tokens2[2]);
			passwordField = tokens2[2];
			if((passwordField.equals(userLookup.getPassword())) && (!userLookup.checkLoginToken())){
				synchronized (user) {
					user = userLookup;
				}
				user.setLoginToken(true);
				sendMessage(cpr,"OK");
			}
		} else {
			sendMessage(cpr, "ERR");
			return;
		}

		return;
	}

	/**
	 * Login server.
	 *
	 * @param cbr the cbr
	 * @param cpr the cpr
	 */
	public static void loginServer(BufferedReader cbr, PrintWriter cpr){

		System.out.println("SERVERLOG: In Login Method");
		String req = null;


			sendMessage(cpr, "OK");
			req = recvMessage(cbr);//Receive New User or Returning User
			if(req != null){
				//System.out.println(req);
				//String[] tokens = req.split(":");
				//String userID = tokens[2];
				sendMessage(cpr,"OK");
			} else {
				sendMessage(cpr, "ERR");
				return;
			}



		switch(req) {
		case "newUser":
			System.out.println("SERVERLOG: Entering newUserCreation method");
			newUserCreation( cbr, cpr);
			return;
		case "returningUser":
			returningUserLogin(cbr, cpr);
			return;
		default:
			break;
		}


		return;

		//System.out.println("SERVERLOG: DEBUG: loginServer Not Entered");

	}

	/**
	 * Checks if the user is logged in server.
	 *
	 * @param cbr the cbr
	 * @param cpr the cpr
	 * @return the boolean
	 */
	public static boolean isLoggedInServer( BufferedReader cbr, PrintWriter cpr){

		String resp = recvMessage(cbr);
		boolean tempToken;
		synchronized (user) {
			tempToken = user.checkLoginToken();
		}
		System.out.println(tempToken);
		if(resp != null){
			if(!tempToken){
				sendMessage(cpr, "ERR");
			} else if (tempToken){
				System.out.println("SERVERLOG: in isLoggedInServer Method");
				sendMessage(cpr, "OK");
			}
		}
		return tempToken;

	}

	/**
	 * Logout of the server.
	 *
	 * @param cbr the cbr
	 * @param cpr the cpr
	 */
	public void logoutServer(BufferedReader cbr, PrintWriter cpr){

		if(user != null){
		synchronized (user) {
			user.setLoginToken(false);
			LibraryServer.userList.replace(user.getUserName(), LibraryAppServer.user);
			}
			sendMessage(cpr, "OK");
			return;
		}else{
			sendMessage(cpr, "ERR");
			user.setLoginToken(false);
			return;
		}
```

```java

		//System.out.println("SERVERLOG: DEBUG: loginServer Not Entered");

	}

	/**
	 * Get Book id from search and then provide user book from server.
	 *
	 * @param targetBookIDString the target book id string
	 * @return the book
	 */
	public static Book bookIDSearchServer(String targetBookIDString){

		for(Book book : bookList){
			if(book.getBookID().equals(targetBookIDString)){
				return book;
			}
		}

		return null;
	}


	/**
	 * Author specific search of book list.
	 *
	 * @param cbr the cbr
	 * @param cpr the cpr
	 */
	public static void authorSearchServer(BufferedReader cbr, PrintWriter cpr){

		//System.out.println("SERVERLOG: In authorSearchServer Method");

		String foundID = "";
		String foundBookIDs = "";

		sendMessage(cpr,"OK");//comment this out to fix
		String req = recvMessage(cbr);
		String targetAuthorName = "";
		if(req != null){
			//System.out.println(req);
			String[] tokens = req.split(":");
			targetAuthorName = tokens[2];
			//System.out.println("SERVERLOG: In authorSearchServer Method, Received request to look for " + targetAuthorName );
			//sendMessage(cpr,"OK");
		} else {
			sendMessage(cpr, "ERR");
			return;
		}
		 req = recvMessage(cbr);
		//String token = "";
		if(req != null){
			//System.out.println(req);
			//String[] tokens = req.split(":");
			//token = tokens[2];
			//System.out.println("SERVERLOG: In authorSearchServer Method, Received request to look for " + targetAuthorName );
			//sendMessage(cpr,"OK");
		} else {
			sendMessage(cpr, "ERR");
			return;
		}


		int i = 0;
		for(Book book : bookList){
			String author = book.getAuthor();
			//System.out.println("SERVERLOG: in authorSearchServer Method, current author: "+ author);
			if(author.equals(targetAuthorName)){
				foundID = book.getBookID();
				System.out.println("SERVERLOG: In authorSearchServer Method, found bookID: " + foundID);
				i++;
				if(i <= 1){
					foundBookIDs = foundID;

				} else if (i >= 2){
					foundBookIDs+= "," + foundID;
				}
			}
		}

		System.out.println(foundBookIDs);
		if(foundBookIDs != null){
			System.out.println("Sending: " + foundBookIDs);
			sendMessage(cpr, foundBookIDs);
			return;
		} else {
			sendMessage(cpr, "ERR");
			return;
		}

	}

	/**
	 * Title specific search of the book list.
	 *
	 * @param cbr the cbr
	 * @param cpr the cpr
	 */
	public static void titleSearchServer(BufferedReader cbr, PrintWriter cpr){

		String foundID = "";
		String foundBookIDs = "";

		//System.out.println("SERVERLOG: In titleSearchServer Method");


		sendMessage(cpr, "OK");//TESTING THIS NOW
		String req = recvMessage(cbr);
		String targetTitle = "";
		if(req != null){
			System.out.println(req);
			String[] tokens = req.split(":");
			targetTitle = tokens[2];
			//sendMessage(cpr,"OK");
		} else {
			sendMessage(cpr, "ERR");
			return;
		}

		 /*req = recvMessage(cbr);
			if(req != null){
				System.out.println(req);
				sendMessage(cpr,"OK");
			} else {
				sendMessage(cpr, "ERR");
				return;
			} */

		int i = 0;
		for(Book book : bookList){
			String title = book.getTitle();
			//System.out.println("SERVERLOG: in authorSearchServer Method, current author: "+ author);
			if(title.equals(targetTitle)){
				foundID = book.getBookID();
				System.out.println("SERVERLOG: In titleSearchServer Method, found bookID: " + foundID);
				i++;
				if(i <= 1){
					foundBookIDs = foundID;
					System.out.println("SERVERLOG: In titleSearchServer Method, found 1 book");

				} else if (i >= 2){
					foundBookIDs+= "," + foundID;
				}
			}
		}
		System.out.println(foundBookIDs);
		if(foundBookIDs != null){
			sendMessage(cpr, foundBookIDs);
		} else {
			sendMessage(cpr, "ERR");
			return;
		}

	}

	/**
	 * Keyword specific search of the book list.
	 *
	 * @param cbr the cbr
	 * @param cpr the cpr
	 */
	public static void keywordSearchServer( BufferedReader cbr, PrintWriter cpr){

		String foundID = "";
		String foundBookIDs = "";

		System.out.println("SERVERLOG: In keywordSearchServer Method");

		sendMessage(cpr, "OK");
		String req = recvMessage(cbr);
		String targetKeyword = "";
		if(req != null){
			System.out.println(req);
			String[] tokens = req.split(":");
			targetKeyword = tokens[2];
			//sendMessage(cpr,"OK");
		} else {
			sendMessage(cpr, "ERR");
			return;
		}
		/*
		 req = recvMessage(cbr);
			if(req != null){
				System.out.println(req);
				sendMessage(cpr,"OK");
			} else {
				sendMessage(cpr, "ERR");
				return;
			} */
		System.out.println("SERVERLOG: Searhing for " + targetKeyword);
		int i =0;
		for(Book book : bookList){
			System.out.println(book.getDateWritten());
			if(((book.getTitle()).toLowerCase()).contains(targetKeyword.toLowerCase()) ||
					((book.getAuthor()).toLowerCase()).contains(targetKeyword.toLowerCase()) ||
					((book.getLanguage()).toLowerCase()).contains(targetKeyword.toLowerCase()) ||
					((book.getDateWritten()).equals(targetKeyword))){
				foundID = book.getBookID();
				System.out.println("SERVERLOG: Middle of keywordSearchServer Method, found bookID: " + foundID);
				i++;
				if(i <= 1 ){
					foundBookIDs = foundID;

				} else if (i >= 2  ){
					foundBookIDs+= "," + foundID;
				}
			}
		}


		if(foundBookIDs != null){
			System.out.println("SERVERLOG: Sending " +foundBookIDs);
			sendMessage(cpr, foundBookIDs);
		} else {
			System.out.println("SERVERLOG: Did Not Find Anything");
			sendMessage(cpr, "ERR");
			return;
		}

	}

	/**
	 * Pick specific search for the book list.
	 *
	 * @param cbr the cbr
	 * @param cpr the cpr
	 */
	public static void searchServer(BufferedReader cbr, PrintWriter cpr){

		//System.out.println("SERVERLOG: In searchServer method");


		String req = recvMessage(cbr);
		sendMessage(cpr, "OK");
		req = recvMessage(cbr);
		String searchChoice = null;
		if(req != null) {
			//System.out.println(req);
			String[] tokens = req.split(":");
			searchChoice = tokens[2];
		} else {
			sendMessage(cpr, "ERR");
			return;
		}

		switch(searchChoice) {
		case "author":
			//System.out.println("SERVERLOG: Entering authorSearchServer method");
			authorSearchServer( cbr, cpr);
			return;
		case "title":
			titleSearchServer( cbr, cpr);
			return;
```

```
                case "keyword":
                        keywordSearchServer( cbr, cpr);
                        return;
                default:
                        sendMessage(cpr, "ERR");
                        return;
                }

                //System.out.println("SERVERLOG: DEBUG: loginServer Not Entered");


        }

        /**
         * Check out the user requested book.
         *
         * @param book the book
         */
        public static boolean checkOutBook(Book book){
                List<Book> tempUserBookList = new ArrayList<Book>();
                synchronized (user) {
                        tempUserBookList = user.getCheckedOutBooks();
                        if(tempUserBookList.size() <= 5){
                        user.checkOutBook(book);
                        return true;
                } else {
                        return false;
                }
                }
        }

        /**
         * Perform Borrow on the server.
         *
         * @param cbr the cbr
         * @param cpr the cpr
         */
        public static void borrowServer(BufferedReader cbr, PrintWriter cpr){

                boolean userListFull = false;

                String resp = recvMessage(cbr);
                System.out.println(resp);
                sendMessage(cpr, "OK");

                String req = recvMessage(cbr);
                if(req.equals("borrow")) {
                 sendMessage(cpr, "OK");
                 req = recvMessage(cbr);
                }
                String targetBookID = "";
                if(req != null) {
                        System.out.println(req);
                        String[] tokens = req.split(":");
                        if(tokens == null){
                                sendMessage(cpr, "ERR");
                                return;
                        }
                        targetBookID = tokens[2];
                        if(targetBookID == null){
                                sendMessage(cpr, "ERR");
                                return;
                        }
                        Book book = bookIDSearchServer(targetBookID);
                        userListFull = checkOutBook(book);
                        if(userListFull){
                        book.printElements();
                        String bookTitle = book.getTitle();
                        System.out.println(bookTitle);
                        sendMessage(cpr, bookTitle);
                        } else {
                                sendMessage(cpr, "ERR");
                                return;
                        }

                }
                }



        /**
         * Check in the user requested book.
         *
         * @param book the book
         */
        public static void checkInBook(Book book){

                synchronized (user) {
                        user.checkInBook(book);
                }
                System.out.println("SERVERLOG: Checking Back In: " + book.getBookID());
        }

        /**
         * Return the book to the server.
         *
         * @param cbr the cbr
         * @param cpr the cpr
         */
        public static void returnBookServer(BufferedReader cbr, PrintWriter cpr){

                String resp = recvMessage(cbr);
                sendMessage(cpr, "OK");
                String req = recvMessage(cbr);
                String targetBookID = "";
                if(req != null) {
                        System.out.println(req);
                        sendMessage(cpr, "OK");
                        String[] tokens = req.split(":");
                        targetBookID = tokens[2];
                        Book book = bookIDSearchServer(targetBookID);
                        checkInBook(book);
                        //sendMessage(cpr, "OK");
                } else{
                        sendMessage(cpr, "ERR");
                        return;
                }

        }

        /**
         * List the currently checked out books from the server.
         *
         * @param cbr the cbr
         * @param cpr the cpr
         */
        public static void listCurrentBooksServer( BufferedReader cbr, PrintWriter cpr){

                String booksOut = "";
                List<Book> tempUserBookList;
                synchronized (user) {
                        tempUserBookList = user.getCheckedOutBooks();
                }
                if(!tempUserBookList.isEmpty()){
```

```java
			for(Book book : tempUserBookList){
				booksOut+= book.getTitle() + " - " + book.getBookID() + ",";
			}
			System.out.println("SERVERLOG: In listCurrentBook method");
			System.out.println(booksOut);
			sendMessage(cpr,booksOut);
		} else if(tempUserBookList.isEmpty()){
			sendMessage(cpr, "ERR");
			return;
		}


	}

	/* (non-Javadoc)
	 * @see AppServer#run()
	 */
	@Override
	public void run() {
		// TODO Auto-generated method stub

		boolean stop = false;
		boolean resetRecv = false;

		try {

			BufferedReader cbr = new BufferedReader(new InputStreamReader(cs.getInputStream()));
			PrintWriter cpr = new PrintWriter(cs.getOutputStream());


			while(!stop){
				String req = recvMessage(cbr);
				while(!resetRecv){
					if(req == null){
						sendMessage(cpr, "ERR");
					}
					if(req.equals("reset")){
					//System.out.println(req);
					sendMessage(cpr, "OK");
					resetRecv = true;
				}
					req = recvMessage(cbr);
				}
				//req = recvMessage(cbr);
				//System.out.println(req);
				//System.out.println("SERVERLOG: Checking Function Request");
				/*System.out.println("1) Login"
						+ ’\n’ + "2)Logout" + ’\n’ + "3) Search" + ’\n’ + "4) Borrow" + ’\n’ + "5) Return"
						+ ’\n’ + "6) List Checkedout Books" + ’\n’ + "7) Exit");*/


				switch(req) {
				case "login":
					System.out.println("SERVERLOG: Entering Login Method");
					loginServer(cbr, cpr);
					break;
				case "logout":
					if(isLoggedInServer(cbr, cpr)){
						logoutServer(cbr, cpr);
						cbr.close();
						cpr.close();
						cs.close();
						stop = true;
					}
					break;
				case "search":
					//sendMessage(cpr, "OK");
					//req = recvMessage(cbr);
					//sendMessage(cpr, "OK");
					searchServer(cbr, cpr);
					break;
				case "borrow":
					sendMessage(cpr, "OK");
					//req = recvMessage(cbr);
					searchServer(cbr, cpr);
					if(isLoggedInServer(cbr, cpr)){
						borrowServer(cbr, cpr);
					}
					break;
				case "return":
					sendMessage(cpr, "OK");
					req = recvMessage(cbr);
					listCurrentBooksServer(cbr, cpr);
					req = recvMessage(cbr);
					sendMessage(cpr, "OK");
					if(isLoggedInServer( cbr, cpr)){
						returnBookServer(cbr, cpr);
					}
					break;
				case "listCurrentBooks":
					listCurrentBooksServer( cbr, cpr);
					break;
				case "isLoggedIn"://Should never be called from this point of execution, will be checked before others
					isLoggedInServer( cbr, cpr);
				case "7":
					stop = true;
					cbr.close();
					cpr.close();
					cs.close();
				default:
					//System.out.println("Please Select A Valid Option(1-7)");
					break;
				}
				resetRecv = false;
			}

			//cbr.close();
			//cpr.close();
			//cs.close();


		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}
}
```

```java
package projects;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.logging.Logger;


/**
 * The Class LibraryClient.
 */

/**
 * @author j.callado
 */
public class LibraryClient {

	/**
	 * Send message.
	 *
	 * @param spr the spr
	 * @param mess the mess
	 */
	public static void sendMessage(PrintWriter spr, String mess) {
		int len = mess.length();
		spr.print(String.format("%03d",len)+":"+mess);
		spr.flush();
	}

	/**
	 * Recv message.
	 *
	 * @param sbr the sbr
	 * @return the string
	 */
	public static String recvMessage(BufferedReader sbr) {
		try {
			char[] slen = new char[4];
			sbr.read(slen,0,4);
			int len = Integer.parseInt(new String(slen,0,3));
			char[] sdata = new char[len];
			sbr.read(sdata,0,len);
			return new String(sdata);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			return null;
		}
	}

	public static void loginNewUser(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

		try{

			LibraryClient.sendMessage(spr, "newUser");
			String resp = LibraryClient.recvMessage(sbr);
			if (resp!= null) {
				System.out.println(resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received In loginNewUser Method");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			System.out.println("Welcome to the Online Digital Library Service!");
			System.out.println("Here, We Will Create Your Personalized Account So You Can Return to Your Selections!");
			System.out.println("Please Create Your userName(a-zA-Z0-9): ");
			String userName = br.readLine();

			LibraryClient.sendMessage(spr, "login:String:"+userName);
			resp = LibraryClient.recvMessage(sbr);
			if(resp!=null){
				System.out.println("Got:"+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received, Invalid userID Creation");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			System.out.println("Creat A Unique Password For Your Account: ");
			String password = br.readLine();

			LibraryClient.sendMessage(spr, "login:String:" + password);
			resp = LibraryClient.recvMessage(sbr);
			if(resp!=null){
				System.out.println("Got:"+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received, Invalid Password Creation");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			if(resp.equals("OK")){
				System.out.println("You Are Now Logged In!");
			}

		} catch (UnknownHostException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}

	public static void loginReturningUser(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

		try{

			LibraryClient.sendMessage(spr, "returningUser");
			String resp = LibraryClient.recvMessage(sbr);
			if (resp!= null) {
				System.out.println(resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received In ReturningUser Method");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			System.out.println("Welcome Back! Please Provide Your Exisiting Account Credentials Below");

			System.out.println("Enter Your Exisintg userName: ");
			String userName = br.readLine();

			LibraryClient.sendMessage(spr, "login:String:"+ userName);
			resp = LibraryClient.recvMessage(sbr);
			if(resp!=null){
				System.out.println("Got:"+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received, Invalid userID Provided");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			System.out.println("Enter your password: ");
			String password = br.readLine();

			LibraryClient.sendMessage(spr, "login:String:"+password);
			resp = LibraryClient.recvMessage(sbr);
			if(resp!=null){
				System.out.println("Got:"+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received, Invalid Password Provided");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			if(resp.equals("OK")){
				System.out.println("You Are Now Logged In!");
			}

		} catch (UnknownHostException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}


	/**
	 * User sends the login request.
	 *
	 * @param s the s
	 * @param sbr the sbr
	 * @param spr the spr
	 * @param br the br
	 */
	public static void login(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

		try{

			boolean back = false;

			LibraryClient.sendMessage(spr, "login");
			String resp = LibraryClient.recvMessage(sbr);
			if (resp!= null) {
				System.out.println(resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received, Returning to Main Menu");
					return;
				}
			}

			while(!back){
				System.out.println("Enter 1-3 To Login into Your Account or Create a New One:" +’\n’ + "1) New User" + ’\n’
							+ "2) Returning User" + ’\n’ + "3) Back to Main Menu");
				String userChoice = br.readLine();

				switch(userChoice) {
				case "1":
					System.out.println("DEBUG: Entering loginNewUser Method");
					loginNewUser(s, sbr, spr, br);
					return;
				case "2":
					System.out.println("DEBUG: Entering loginReturningUser Method");
					loginReturningUser(s, sbr, spr, br);
					return;
				case "3":
					back = true;
					System.out.println("Returning to Main Menu");
					return;
				default:
					System.out.println("Please Select a Valid Option(1-3)");
					break;
				}
			}

			System.out.println("SYSTEM LOG: Exiting login Method, Returning to Main Menu");

		} catch (UnknownHostException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}

	/**
	 * Checks if it is logged in by asking the server to see if it has maintained the authentication token.
	 *
	 * @param sbr the sbr
	 * @param spr the spr
	 * @return true, if is logged in
	 */
	public static boolean isLoggedIn(BufferedReader sbr, PrintWriter spr
```

```java
){

		//try{

		boolean loggedIn = false;

		System.out.println("Checking Login Status...");

		LibraryClient.sendMessage(spr, "isLoggedIn");
		String resp = LibraryClient.recvMessage(sbr);
		if (resp != null) {
			System.out.println(resp);
			if(resp.equals("OK")){
				loggedIn = true;
				System.out.println("You Are Logged In!");
				//return loggedIn;
			} else if(resp.equals("ERR")){
				System.out.println("SYSTEM LOG: Error Received, You Must Login First!");
			}
		}

		//System.out.println("You Are Logged In!");
		//LibraryClient.sendMessage(spr, "OK");

		/*
		} catch (UnknownHostException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}*/
		System.out.println(loggedIn);
		return loggedIn;

	}

	/**
	 * Send logout request to the server.
	 * Note: Upon receiving an Error when logged in, the user will be logged out to ensure that his/her information cannot be modified and
	 * any other unwarranted behavior.
	 *
	 * @param s the s
	 * @param sbr the sbr
	 * @param spr the spr
	 * @param br the br
	 */
	public static void logout(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

		Boolean tokenActive = false;
		tokenActive = isLoggedIn(sbr, spr);
		System.out.println(tokenActive);
		if(tokenActive){

			LibraryClient.sendMessage(spr, "logout");
			String resp = LibraryClient.recvMessage(sbr);
			if (resp!= null) {
				System.out.println(resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received in logout Method");
					System.out.println("You Are Logged Out!");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			if(resp.equals("OK")){
				System.out.println("You Are Logged Out!");
			}

		System.out.println("Exiting Client");
		return;
		}
		return;
	}

	/**
	 * Send Author specific search request to the server.
	 *
	 * @param s the s
	 * @param sbr the sbr
	 * @param spr the spr
	 * @param br the br
	 */
	public static void authorSearch(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

		try{
			LibraryClient.sendMessage(spr, "search:String:author");
			String resp = LibraryClient.recvMessage(sbr);
			System.out.println("DEBUG: sent author Request");
			System.out.println("Got: "+resp);
			if(resp != null){
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received From author Request In authorSearch Method");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			System.out.println("Please Enter The Name Of The Author:");
			String authorName = br.readLine();

			LibraryClient.sendMessage(spr, "search:authorString:"+authorName);
			resp = LibraryClient.recvMessage(sbr);
			LibraryClient.sendMessage(spr, "search:authorString:sendResult");//
			resp = LibraryClient.recvMessage(sbr);//
			if(resp!=null){
				System.out.println("Got:"+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received From authorName Request In authorSearch Method");
					System.out.println("Returning to Main Menu");
					return;
				}
				String[] respArray = resp.trim().split(",");
				System.out.println("Displaying Top Author Results");
				for(int i = 0; i < respArray.length; i++){
					System.out.println("Result #"+i+": " + respArray[i]);
				}
				System.out.println("Please Note The BookID For Your Borrow");
			}

			return;
			//LibraryClient.sendMessage(spr, "OK");

		} catch (UnknownHostException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		System.out.println("DEBUG: authorSearch Not Entered!");
		return;

	}

	/**
	 * Send Title specific search request to the server.
	 *
	 * @param s the s
	 * @param sbr the sbr
	 * @param spr the spr
	 * @param br the br
	 */
	public static void titleSearch(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

		try{

			LibraryClient.sendMessage(spr, "search:String:title");
			String resp = LibraryClient.recvMessage(sbr);
			if(resp!=null){
				System.out.println("Got: "+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received From title Request In titleSearch Method");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			System.out.println("Please Enter The Title Of The Book:");
			String titleName = br.readLine();

			LibraryClient.sendMessage(spr, "search:String:"+titleName);
			//resp = LibraryClient.recvMessage(sbr);
			//LibraryClient.sendMessage(spr, "search:authorString:sendResult");//
			resp = LibraryClient.recvMessage(sbr);//
			if(resp!=null){
				System.out.println("Got:"+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received From titleName Request In titleSearch Method");
					System.out.println("Returning to Main Menu");
					return;
				}
				if(resp.contains(",")){
					String[] respArray = resp.trim().split(",");
					System.out.println("Displaying Top Title Results");
					for(int i = 0; i <= respArray.length; i++){
						System.out.println("Result #"+i+": "+ respArray[i]);
					}
				} else {
					System.out.println("Displaying Top Title Results");
					System.out.println("Result #0: "+ resp);
				}
				System.out.println("Please Note The BookID For Your Borrow");
			}

			return;


		} catch (UnknownHostException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		System.out.println("DEBUG: titleSearch Not Entered!");
		return;

	}

	/**
	 * Send Keyword specific search request to the server.
	 *
	 * @param s the s
	 * @param sbr the sbr
	 * @param spr the spr
	 * @param br the br
	 */
	public static void keywordSearch(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){
		try{

			LibraryClient.sendMessage(spr, "search:String:keyword");
			String resp = LibraryClient.recvMessage(sbr);
			if(resp!=null){
				System.out.println("Got: "+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received From title Request In titleSearch Method");
					System.out.println("Returning to Main Menu");
					return;
				}
			}

			System.out.println("Please Enter A Keyword:");
			String keywordString = br.readLine();

			LibraryClient.sendMessage(spr, "search:String:"+keywordString);
			//resp = LibraryClient.recvMessage(sbr);
			//LibraryClient.sendMessage(spr, "search:authorString:sendResult");//
			resp = LibraryClient.recvMessage(sbr);//
			if(resp!=null){
				System.out.println("Got:"+resp);
				if(resp.equals("ERR")){
					System.out.println("SYSTEM LOG: Error Received From keywordString Request In keywordSearch Method");
					System.out.println("Returning to Mai
```

```java
n Menu");
                    return;
                }
                resp = resp.replace(":", "");
                String[] respArray = resp.trim().split(",");
                System.out.println("Displaying Top Keyword Results");
                for(int i = 0; i < respArray.length; i++){
                    System.out.println("Result #"+ i +": "+ respArray[i]);
                }
                System.out.println("Please Note The BookID For Your Borrow");
            }

            return;


        } catch (UnknownHostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println("DEBUG: keywordSearch Not Entered!");
        return;

    }

    /**
     * Prompt user to select specific Search function.
     *
     * @param s the s
     * @param sbr the sbr
     * @param spr the spr
     * @param br the br
     */
    public static void search(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

        try{
            boolean back = false;

            LibraryClient.sendMessage(spr, "search");
            String resp = LibraryClient.recvMessage(sbr);
            if (resp!= null) {
                System.out.println("Got: "+resp);
                if(resp.equals("ERR")){
                    System.out.println("SYSTEM LOG: Error Received From search Request In search Method");
                    System.out.println("Returning to Main Menu");

                    return;
                }
            }
            //resp = LibraryClient.recvMessage(sbr);//new
            while(!back){
                System.out.println("Enter 1-4 To Select Your Search (Case-Sensitive):" +’\n’ + "1) Author" + ’\n’
                        + "2) Title" + ’\n’ + "3) Keyword" + ’\n’ + "4) Back to Main Menu");
                String userChoice = br.readLine();

                switch(userChoice) {
                case "1":
                    System.out.println("DEBUG: Entering authorSearch Method");
                    authorSearch( s, sbr, spr, br);
                    return;
                case "2":
                    System.out.println("DEBUG: Entering titleSearch Method");
                    titleSearch(s, sbr, spr, br);
                    return;
                case "3":
                    System.out.println("DEBUG: Entering keywordSearch Method");
                    keywordSearch(s, sbr, spr, br);
                    return;
                case "4":
                    back = true;
                    resp = recvMessage(sbr);
                    return;
                default:
                    System.out.println("Please Select a Valid Option(1-4). Note: Search is Case-Sensitive");
                    break;
                }
            }


        } catch (UnknownHostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


        return;
    }

    /**
     * Send Borrow request to the server.
     *
     * @param s the s
     * @param sbr the sbr
     * @param spr the spr
     * @param br the br
     */
    public static void borrow(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

        Boolean tokenActive = false;
        tokenActive = isLoggedIn(sbr, spr);
        System.out.println(tokenActive);
        if(tokenActive){

            try{

                LibraryClient.sendMessage(spr, "borrow");
                String resp = LibraryClient.recvMessage(sbr);

                if(resp!=null){
                    System.out.println("Got: "+resp);
                    if(resp.equals("ERR")){
                        System.out.println("SYSTEM LOG: Error Received From borrow Request In borrow Method");
                        System.out.println("Returning to Main Menu");

                        return;
                    }
                }


                System.out.println("Please Enter The BookID
Of Your Book Request(Only Five Books Can Be Out At Once!) :");
                String bookID = br.readLine();

                LibraryClient.sendMessage(spr, "borrow:String:"+bookID);
                resp = LibraryClient.recvMessage(sbr);
                if(resp!=null){
                    System.out.println("Got: "+resp);
                    if(resp.equals("ERR")){
                        System.out.println("SYSTEM LOG: Error Received From bookID Request In borrow Method");
                        System.out.println("Your bookList might be full. Please Limit to Five Borrows");
                        System.out.println("Returning to Main Menu");
                        return;
                    }
                    System.out.println("Here Is Your Request: "+resp);
                    System.out.println("Please Enjoy Your Selection!");
                }
                return;

            } catch (UnknownHostException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        } else {
            System.out.println("Returning to the Main Menu");
            return;
        }
    }

    /**
     * Send Return book request to the server.
     *
     * @param s the s
     * @param sbr the sbr
     * @param spr the spr
     * @param br the br
     */
    public static void returnBook(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){

        boolean isLoggedin = isLoggedIn(sbr, spr);

        if(isLoggedin){

            try{

                LibraryClient.sendMessage(spr, "returnBook");
                String resp = LibraryClient.recvMessage(sbr);
                if(resp!=null){
                    System.out.println("Got: "+resp);
                    if(resp.equals("ERR")){
                        System.out.println("SYSTEM LOG: Error Received From returnBook Request In returnBook Method");
                        System.out.println("Returning to Main Menu");
                        return;
                    }
                }

                System.out.println("Please Enter The BookID Of\nThe Book You Wish to Return :");
                String bookID = br.readLine();

                LibraryClient.sendMessage(spr, "returnBook:String:"+bookID);
                resp = LibraryClient.recvMessage(sbr);
                if(resp!=null){
                    System.out.println(resp);
                    if(resp.equals("ERR")){
                        System.out.println("SYSTEM LOG: Error Received From bookID Request In returnBook Method");
                        System.out.println("Returning to Main Menu");
                        return;
                    }
                    if(resp.equals("OK")){
                        System.out.println("Your Book Has Been Returned!\nPlease Search For Your Next Selection!");
                    }
                    //Just in case something else comes over the wire besides OK or ERR
                    //System.out.println("Error Received, Exiting Program");
                    //System.exit(0);
                }
                return;

            } catch (UnknownHostException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            //System.out.println("DEBUG: returnBook Not Entered!");


        } else if(!isLoggedin){
            System.out.println("Returning to the Main Menu");
            return;
        } else {
            System.out.println("SYSTEM LOG: Unknown Error Received, Exiting Program");
            System.exit(0);
        }
    }


    /**
     * List currently checked out books from the server.
     *
     * @param s the s
     * @param sbr the sbr
     * @param spr the spr
     * @param br the br
     */
    public static void listCurrentBooks(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){


        //try{

        LibraryClient.sendMessage(spr, "listCurrentBooks");
        String resp = LibraryClient.recvMessage(sbr);
        if(resp!=null){
```

```java
                        System.out.println(resp);
                        if(resp.equals("ERR")){
                                System.out.println("You Do Not Currently Have Any Books Checked Out");
                                System.out.println("Please Search For A Selection Or Borrow Once You Have Found One");
                                System.out.println("Returning to Main Menu");
                                return;
                        }
                        String[] respArray = resp.trim().split(",");
                        System.out.println("Displaying Your Currently Checked Out Books:");
                        for(int i = 0; i < respArray.length; i++){
                                System.out.println("Result #"+i+": "+ respArray[i]);
                        }
                        System.out.println("Please Note The BookID For Your Borrow or Return");
                }


                return;

                //System.out.println("DEBUG: listCurrentBooks Not Entered!");

        }

        /**
         * The main method.
         *
         * @param args the arguments
         */
        public static void main(String[] args) {
                // TODO Auto-generated method stub

                Boolean exit = false;


                try {
                        Socket s = new Socket("localhost", 45000);

                        BufferedReader sbr = new BufferedReader(new InputStreamReader(s.getInputStream()));
                        PrintWriter spr = new PrintWriter(s.getOutputStream());

                        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

                        while(!exit){
                                sendMessage(spr, "reset");
                                String resp = recvMessage(sbr);
                                System.out.println("Please Select 1-7 As An Option:");
                                System.out.println("1) Login"
                                                + ’\n’ + "2) Logout/exit" + ’\n’ + "3) Search (Case-Sensitive)" + ’\n’ + "4) Borrow" + ’\n’ + "5) Return"
                                                + ’\n’ + "6) List Checked Out Books\nEnter a Number: ");
                                String userChoice = br.readLine();
                                //String resp;
                                switch(userChoice) {
                                case "1":
                                        login( s, sbr, spr, br);
                                        break;
                                case "2":
                                        sendMessage(spr, "logout");
                                        logout(s, sbr, spr, br);
                                        exit = true;
                                        sbr.close();
                                        spr.close();
                                        br.close();
                                        s.close();
                                        System.exit(0);
                                case "3":
                                        search(s, sbr, spr, br);

                                        break;
                                case "4":
                                        sendMessage(spr, "borrow");
                                        resp = recvMessage(sbr);
                                        if(resp.equals("OK")){
                                                search(s, sbr, spr, br);
                                                borrow(s, sbr, spr, br);
                                        } else if(resp.equals("ERR")){
                                                System.out.println("SYSTEM LOG: ERROR RECEIVED");
                                        }
                                        break;
                                case "5":
                                        sendMessage( spr, "return");
                                        resp = recvMessage(sbr);
                                        listCurrentBooks(s, sbr, spr, br);
                                        sendMessage(spr, "return");
                                        resp = recvMessage(sbr);
                                        System.out.println(resp);
                                        if(resp.equals("OK")){
                                                returnBook(s, sbr, spr, br);
                                        }  else if(resp.equals("ERR")){
                                                System.out.println("SYSTEM LOG: ERROR RECEIVED");
                                        }
                                        break;
                                case "6":
                                        listCurrentBooks(s, sbr, spr, br);
                                        break;
                                default:
                                        System.out.println("Please Select A Valid Option(1-7)");
                                        break;
                                }
                        }


                } catch (UnknownHostException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

}
```