

```

1: package projects;
2:
3: /**
4:  * The Class AppServer.
5:  */
6:
7: /**
8:  * @author j.callado
9:  */
10: public abstract class AppServer implements Runnable {
11:
12:     /* (non-Javadoc)
13:      * @see java.lang.Runnable#run()
14:      */
15:     @Override
16:     public abstract void run();
17:
18: }

```



```
1: package projects;
2:
3: import java.io.Serializable;
4: import java.util.ArrayList;
5: import java.util.List;
6:
7:
8: /**
9:  * The Class User.
10:  *
11:  * @author j.callado
12:  */
13: public class User implements Serializable{
14:
15:     /** The user id. */
16:     private int userID;
17:
18:     /** The user name. */
19:     private String userName;
20:
21:     /** The password. */
22:     private String password;
23:
24:     /** The login token. */
25:     private boolean loginToken;
26:
27:     /** The checked out books. */
28:     private List<Book> checkedOutBooks = new ArrayList<Book>();
29:
30:     /**
31:      * Instantiates a new user.
32:      *
33:      * @param userID the user id
34:      * @param userName the user name
35:      * @param password the password
36:      */
37:     public User(int userID, String userName,
38:                 String password){
39:         this.userID = userID;
40:         this.userName = userName;
41:         this.password = password;
42:         this.loginToken = true;
43:         checkedOutBooks.clear();
44:     }
45:
46:     /**
47:      * Gets the book id.
48:      *
49:      * @return the book id
50:      */
51:     public int getUserID(){
52:         return userID;
53:     }
54:
55:     /**
56:      * Gets the userName.
57:      *
58:      * @return the userName
59:      */
60:     public String getUserName(){
61:         return userName;
62:     }
63:
64:     /**
65:      * Gets the title.
66:      *
67:      * @return the title
68:      */
69:     public String getPassword(){
```

```
70:         return password;
71:     }
72:
73:     /**
74:      * Gets the language.
75:      *
76:      * @return the language
77:      */
78:     public boolean checkLoginToken(){
79:         return loginToken;
80:     }
81:
82:
83:     /**
84:      * Check out book.
85:      *
86:      * @param book the book
87:      */
88:     public void checkOutBook(Book book){
89:         this.checkedOutBooks.add(book);
90:     }
91:
92:     /**
93:      * Check in book.
94:      *
95:      * @param book the book
96:      */
97:     public void checkInBook(Book book){
98:         this.checkedOutBooks.remove(book);
99:     }
100:
101:     /**
102:      * Gets the checked out books.
103:      *
104:      * @return the checked out books
105:      */
106:     public List<Book> getCheckedOutBooks(){
107:         return checkedOutBooks;
108:     }
109:
110:     /**
111:      * Sets the login token.
112:      *
113:      * @param action the new login token
114:      */
115:     public void setLoginToken(boolean action){
116:         this.loginToken = action;
117:     }
118:
119:
120:     /**
121:      * Prints the elements.
122:      */
123:     public void printElements(){
124:
125:         System.out.println("userID: " + userID);
126:         System.out.println("userName: " + userName);
127:         System.out.println("loginToken: " + loginToken);
128:
129:     }
130:
131:
132: }
133:
```



```
1: package projects;
2:
3: import java.io.Serializable;
4:
5: /**
6:  * @author j.callado
7:  */
8:
9: /**
10:  * The Class Book.
11:  */
12: public class Book implements Serializable{
13:
14:     /** The book id. */
15:     private String bookID;
16:
17:     /** The author. */
18:     private String author;
19:
20:     /** The title. */
21:     private String title;
22:
23:     /** The language. */
24:     private String language;
25:
26:     /** The date written. */
27:     private String dateWritten;
28:
29:     /**
30:      * Instantiates a new book.
31:      *
32:      * @param bookIDString the book id string
33:      * @param tempBookAuthor the temp book author
34:      * @param tempBookTitle the temp book title
35:      * @param tempBookLanguage the temp book language
36:      * @param tempBookWritten the temp book written
37:      */
38:     public Book(String bookIDString, String tempBookAuthor,
39:                 String tempBookTitle, String tempBookLanguage,
40:                 String tempBookWritten){
41:         this.bookID = bookIDString;
42:         this.author = tempBookAuthor;
43:         this.title = tempBookTitle;
44:         this.language = tempBookLanguage;
45:         this.dateWritten = tempBookWritten.trim();
46:     }
47:
48:     /**
49:      * Gets the book id.
50:      *
51:      * @return the book id
52:      */
53:     public String getBookID(){
54:         return bookID;
55:     }
56:
57:     /**
58:      * Gets the author.
59:      *
60:      * @return the author
61:      */
62:     public String getAuthor(){
63:         return author;
64:     }
65:
66:     /**
67:      * Gets the title.
68:      *
69:      * @return the title
```

```
70:     */
71:     public String getTitle(){
72:         return title;
73:     }
74:
75:     /**
76:      * Gets the language.
77:      *
78:      * @return the language
79:      */
80:     public String getLanguage(){
81:         return language;
82:     }
83:
84:     /**
85:      * Gets the date written.
86:      *
87:      * @return the date written
88:      */
89:     public String getDateWritten(){
90:         return dateWritten;
91:     }
92:
93:     public void printElements(){
94:
95:         System.out.println("bookID: " + bookID);
96:         System.out.println("Author: " + author);
97:         System.out.println("Title: " + title);
98:         System.out.println("Language: " + language);
99:         System.out.println("Date Written: " + dateWritten);
100:
101:     }
102:
103: }
```



```

1: package projects;
2:
3: import java.io.File;
4: import java.io.FileNotFoundException;
5: import java.io.IOException;
6: import java.net.ServerSocket;
7: import java.net.Socket;
8: import java.util.ArrayList;
9: import java.util.List;
10: import java.util.Scanner;
11: import java.util.concurrent.ConcurrentHashMap;
12: import java.util.concurrent.ConcurrentMap;
13: import java.util.concurrent.atomic.AtomicInteger;
14:
15: // TODO: Auto-generated Javadoc
16: /**
17:  * The Class LibraryServer.
18:  *
19:  * @author j.callado
20:  */
21:
22: /**
23:  * The Class LibraryServer
24:  */
25:
26: public class LibraryServer {
27:
28:     /** The book mastercatalog. */
29:     private static List<Book> bookCatalog = new ArrayList<Book>();
30:
31:
32:     /** The user list. */
33:     public static ConcurrentMap<String, User> userList = new ConcurrentH
ashMap<String, User>();
34:
35:     /** The total number of users with accounts on the server */
36:     public static AtomicInteger totalUsers = new AtomicInteger(0);
37:
38:     /**
39:      * The main method.
40:      *
41:      * @param args the arguments
42:      */
43:     @SuppressWarnings("resource")
44:     public static void main(String[] args) {
45:         // TODO Auto-generated method stub
46:         Scanner bookFile;
47:         try {
48:             bookFile = new Scanner(new File("BookList.txt")).use
Delimiter(",\\s*");
49:
50:
51:             int tempBookID = 0;
52:             String bookIDString;
53:             String tempBookAuthor;
54:             String tempBookTitle;
55:             String tempBookLanguage;
56:             String tempBookWritten;
57:             while(bookFile.hasNextLine()){
58:                 if(tempBookID < 10){
59:                     bookIDString = ("0" + tempBookID);
60:                     tempBookTitle = bookFile.nextLine();
61:                     //System.out.println("tempBookAuthor
value: " +tempBookAuthor);
62:                     tempBookAuthor = bookFile.nextLine()
;
63:                     tempBookLanguage = bookFile.nextLine()
;
64:                     tempBookWritten = bookFile.nextLine(
);
65:
66:                 } else {
67:                     bookIDString = new Integer(tempBookID
).toString();
68:                     tempBookTitle = bookFile.nextLine();
69:                     tempBookAuthor = bookFile.nextLine()
;
70:                     tempBookLanguage = bookFile.nextLine()
;
71:                     tempBookWritten = bookFile.nextLine()
;
72:                     Book book = new Book(bookIDString, tempBookA
uthor, tempBookTitle, tempBookLanguage, tempBookWritten);
73:                     bookCatalog.add(book);
74:                     tempBookID++;
75:                 }
76:                 int i = 1;
77:                 for(Book book : bookCatalog){
78:                     System.out.println("Book #" + i);
79:                     i++;
80:                     book.printElements();
81:                 }
82:                 bookFile.close();
83:             } catch (FileNotFoundException e1) {
84:                 // TODO Auto-generated catch block
85:                 e1.printStackTrace();
86:             }
87:         } try {
88:
89:
90:             ServerSocket ss = new ServerSocket(45000);
91:             while (true) {
92:                 System.out.println("Listening on...45000");
93:                 Socket cs = ss.accept();
94:
95:                 System.out.println("Received connection from
:"+cs.getInetAddress().getHostAddress()+":"+cs.getPort());
96:
97:                 LibraryAppServer app = new LibraryAppServer(
cs, bookCatalog);
98:
99:                 // iterative
100:                 //app.run();
101:
102:                 // concurrent
103:                 System.out.println("Server Log: Creating new
thread");
104:                 Thread t = new Thread(app);
105:                 t.start();
106:
107:             }
108:
109:
110:         } catch (IOException e) {
111:             // TODO Auto-generated catch block
112:             e.printStackTrace();
113:         }
114:
115:     }
116: }

```



```

1: package projects;
2:
3: import java.io.*;
4: import java.net.*;
5: import java.util.ArrayList;
6: import java.util.List;
7: import java.util.Scanner;
8: import java.util.concurrent.atomic.AtomicInteger;
9:
10:
11:
12: /**
13:  * The Class LibraryAppServer.
14:  */
15:
16: /**
17:  * @author j.callado
18:  */
19:
20: public class LibraryAppServer extends AppServer {
21:
22:     /** The login token. */
23:     private static User user = null;
24:
25:     /** The book list. */
26:     private static List<Book> bookList = new ArrayList<Book>();
27:
28:     /** The cs. */
29:     private Socket cs;
30:
31:     /**
32:      * Instantiates a new library app server.
33:      *
34:      * @param s the s
35:      * @param bookCatalog the book catalog
36:      */
37:     public LibraryAppServer(Socket s, List<Book> bookCatalog) {
38:         cs = s;
39:         LibraryAppServer.bookList = bookCatalog;
40:
41:     }
42:
43:     /**
44:      * Send message.
45:      *
46:      * @param spr the spr
47:      * @param mess the mess
48:      */
49:     public static void sendMessage(PrintWriter spr, String mess) {
50:         int len = mess.length();
51:         spr.print(String.format("%03d",len)+"."+mess);
52:         spr.flush();
53:     }
54:
55:     /**
56:      * Recv message.
57:      *
58:      * @param sbr the sbr
59:      * @return the string
60:      */
61:     public static String recvMessage(BufferedReader sbr) {
62:         try {
63:             char[] slen = new char[4];
64:             sbr.read(slen,0,4);
65:             int len = Integer.parseInt(new String(slen,0,3));
66:             char[] sdata = new char[len];
67:             sbr.read(sdata,0,len);
68:             return new String(sdata);
69:         } catch (IOException e) {

```

```

70:             // TODO Auto-generated catch block
71:             e.printStackTrace();
72:             return null;
73:         }
74:     }
75:
76:     /**
77:      * New user creation.
78:      *
79:      * @param cbr the cbr
80:      * @param cpr the cpr
81:      */
82:     public static void newUserCreation(BufferedReader cbr, PrintWriter c
pr){
83:
84:         int newUserID = LibraryServer.totalUsers.getAndIncrement();
85:         String newPassword = null;
86:         String newUserName = null;
87:
88:
89:         sendMessage(cpr, "OK");
90:         String req = recvMessage(cbr); //Receive new userName
91:         if(req != null){
92:             //System.out.println(req);
93:             String[] tokens = req.split(":");
94:             newUserName = tokens[2];
95:             sendMessage(cpr, "OK");
96:         } else {
97:             sendMessage(cpr, "ERR");
98:             return;
99:         }
100:
101:         req = recvMessage(cbr);
102:         if(req != null){
103:             //System.out.println(req);
104:             String[] tokens2 = req.split(":");
105:             //System.out.println(tokens2[2]);
106:             newPassword = tokens2[2];
107:             sendMessage(cpr, "OK");
108:         } else {
109:             sendMessage(cpr, "ERR");
110:             return;
111:         }
112:
113:         user = new User(newUserID, newUserName, newPassword);
114:         synchronized (user) {
115:             LibraryServer.userList.putIfAbsent(user.getUserName(
), user);
116:             user = LibraryServer.userList.get(user.getUserName()
);
117:
118:         }
119:
120:         return;
121:     }
122:
123:     /**
124:      * Returning user login.
125:      *
126:      * @param cbr the cbr
127:      * @param cpr the cpr
128:      */
129:     public static void returningUserLogin(BufferedReader cbr, PrintWrite
r cpr){
130:
131:         String nameField = null;
132:         String passwordField = null;
133:         User userLookup = null;
134:

```

```

135:
136:
137:
138:     sendMessage(cpr, "OK");
139:     String req = recvMessage(cbr); //Receive Returning Username
140:     if(req != null){
141:         //System.out.println(req);
142:         String[] tokens = req.split(":");
143:         nameField = tokens[2];
144:         sendMessage(cpr, "OK");
145:         userLookup = LibraryServer.userList.get(name
Field);
146:     } else {
147:         sendMessage(cpr, "ERR");
148:         return;
149:     }
150:
151:
152:     req = recvMessage(cbr); // recieve password
153:     if(req != null){
154:         //System.out.println(req);
155:         String[] tokens2 = req.split(":");
156:         System.out.println(tokens2[2]);
157:         passwordField = tokens2[2];
158:         if((passwordField.equals(userLookup.getPassword()))
&& (!userLookup.checkLoginToken())){
159:             synchronized (user) {
160:                 user = userLookup;
161:             }
162:             user.setLoginToken(true);
163:             sendMessage(cpr, "OK");
164:         }
165:     } else {
166:         sendMessage(cpr, "ERR");
167:         return;
168:     }
169:
170:     return;
171: }
172:
173: /**
174:  * Login server.
175:  *
176:  * @param cbr the cbr
177:  * @param cpr the cpr
178:  */
179: public static void loginServer(BufferedReader cbr, PrintWriter cpr){
180:
181:     System.out.println("SERVERLOG: In Login Method");
182:     String req = null;
183:
184:
185:     sendMessage(cpr, "OK");
186:     req = recvMessage(cbr); //Receive New User or Returni
ng User
187:
188:     if(req != null){
189:         //System.out.println(req);
190:         //String[] tokens = req.split(":");
191:         //String userID = tokens[2];
192:         sendMessage(cpr, "OK");
193:     } else {
194:         sendMessage(cpr, "ERR");
195:         return;
196:     }
197:
198:
199:     switch(req) {
200:         case "newUser":

```

```

201:         System.out.println("SERVERLOG: Entering newUserCreat
ion method");
202:         newUserCreation( cbr, cpr);
203:         return;
204:     case "returningUser":
205:         returningUserLogin(cbr, cpr);
206:         return;
207:     default:
208:         break;
209:     }
210:
211:
212:     return;
213:
214:     //System.out.println("SERVERLOG: DEBUG: loginServer Not Ente
red");
215:
216: }
217:
218: /**
219:  * Checks if the user is logged in server.
220:  *
221:  * @param cbr the cbr
222:  * @param cpr the cpr
223:  * @return the boolean
224:  */
225: public static boolean isLoggedInServer( BufferedReader cbr, PrintWri
ter cpr){
226:
227:     String resp = recvMessage(cbr);
228:     boolean tempToken;
229:     synchronized (user) {
230:         tempToken = user.checkLoginToken();
231:     }
232:     System.out.println(tempToken);
233:     if(resp != null){
234:         if(!tempToken){
235:             sendMessage(cpr, "ERR");
236:         } else if (tempToken){
237:             System.out.println("SERVERLOG: in isLoggedInIn
Server Method");
238:             sendMessage(cpr, "OK");
239:         }
240:     }
241:     return tempToken;
242: }
243:
244:
245: /**
246:  * Logout of the server.
247:  *
248:  * @param cbr the cbr
249:  * @param cpr the cpr
250:  */
251: public void logoutServer(BufferedReader cbr, PrintWriter cpr){
252:
253:     if(user != null){
254:         synchronized (user) {
255:             user.setLoginToken(false);
256:             LibraryServer.userList.replace(user.getUserName(), L
ibraryAppServer.user);
257:         }
258:         sendMessage(cpr, "OK");
259:         return;
260:     }else{
261:         sendMessage(cpr, "ERR");
262:         user.setLoginToken(false);
263:         return;
264:     }

```

```

265:
266:         //System.out.println("SERVERLOG: DEBUG: loginServer Not Ente
red");
267:     }
268: }
269:
270: /**
271:  * Get Book id from search and then provide user book from server.
272:  *
273:  * @param targetBookIDString the target book id string
274:  * @return the book
275:  */
276: public static Book bookIDSearchServer(String targetBookIDString){
277:
278:     for(Book book : bookList){
279:         if(book.getBookID().equals(targetBookIDString)){
280:             return book;
281:         }
282:     }
283:
284:     return null;
285: }
286:
287:
288: /**
289:  * Author specific search of book list.
290:  *
291:  * @param cbr the cbr
292:  * @param cpr the cpr
293:  */
294: public static void authorSearchServer(BufferedReader cbr, PrintWrite
r cpr){
295:
296:         //System.out.println("SERVERLOG: In authorSearchServer Metho
d");
297:
298:         String foundID = "";
299:         String foundBookIDs = "";
300:
301:         sendMessage(cpr,"OK");//comment this out to fix
302:         String req = rcvMessage(cbr);
303:         String targetAuthorName = "";
304:         if(req != null){
305:             //System.out.println(req);
306:             String[] tokens = req.split(":");
307:             targetAuthorName = tokens[2];
308:             //System.out.println("SERVERLOG: In authorSearchServ
er Method, Received request to look for " + targetAuthorName );
309:             //sendMessage(cpr,"OK");
310:         } else {
311:             sendMessage(cpr, "ERR");
312:             return;
313:         }
314:         req = rcvMessage(cbr);
315:         //String token = "";
316:         if(req != null){
317:             //System.out.println(req);
318:             //String[] tokens = req.split(":");
319:             //token = tokens[2];
320:             //System.out.println("SERVERLOG: In authorSearchServ
er Method, Received request to look for " + targetAuthorName );
321:             //sendMessage(cpr,"OK");
322:         } else {
323:             sendMessage(cpr, "ERR");
324:             return;
325:         }
326:
327:
328:         int i = 0;

```

```

329:         for(Book book : bookList){
330:             String author = book.getAuthor();
331:             //System.out.println("SERVERLOG: in authorSearchServ
er Method, current author: "+ author);
332:             if(author.equals(targetAuthorName)){
333:                 foundID = book.getBookID();
334:                 System.out.println("SERVERLOG: In authorSear
chServer Method, found bookID: " + foundID);
335:                 i++;
336:                 if(i <= 1){
337:                     foundBookIDs = foundID;
338:
339:                 } else if (i >= 2){
340:                     foundBookIDs+= "," + foundID;
341:                 }
342:             }
343:         }
344:
345:         System.out.println(foundBookIDs);
346:         if(foundBookIDs != null){
347:             System.out.println("Sending: " + foundBookIDs);
348:             sendMessage(cpr, foundBookIDs);
349:             return;
350:         } else {
351:             sendMessage(cpr, "ERR");
352:             return;
353:         }
354:     }
355: }
356:
357: /**
358:  * Title specific search of the book list.
359:  *
360:  * @param cbr the cbr
361:  * @param cpr the cpr
362:  */
363: public static void titleSearchServer(BufferedReader cbr, PrintWriter
cpr){
364:
365:         String foundID = "";
366:         String foundBookIDs = "";
367:
368:         //System.out.println("SERVERLOG: In titleSearchServer Method
");
369:
370:         sendMessage(cpr, "OK");//TESTING THIS NOW
371:         String req = rcvMessage(cbr);
372:         String targetTitle = "";
373:         if(req != null){
374:             System.out.println(req);
375:             String[] tokens = req.split(":");
376:             targetTitle = tokens[2];
377:             //sendMessage(cpr,"OK");
378:         } else {
379:             sendMessage(cpr, "ERR");
380:             return;
381:         }
382:
383:         /*req = rcvMessage(cbr);
384:         if(req != null){
385:             System.out.println(req);
386:             sendMessage(cpr,"OK");
387:         } else {
388:             sendMessage(cpr, "ERR");
389:             return;
390:         } */
391:
392:         int i = 0;
393:         for(Book book : bookList){

```

```

394:         String title = book.getTitle();
395:         //System.out.println("SERVERLOG: in authorSearchServ
er Method, current author: " + author);
396:         if(title.equals(targetTitle)){
397:             foundID = book.getBookID();
398:             System.out.println("SERVERLOG: In titleSearc
hServer Method, found bookID: " + foundID);
399:             i++;
400:             if(i <= 1){
401:                 foundBookIDs = foundID;
402:                 System.out.println("SERVERLOG: In ti
tleSearchServer Method, found 1 book");
403:
404:             } else if (i >= 2){
405:                 foundBookIDs+= "," + foundID;
406:             }
407:         }
408:     }
409:     System.out.println(foundBookIDs);
410:     if(foundBookIDs != null){
411:         sendMessage(cpr, foundBookIDs);
412:     } else {
413:         sendMessage(cpr, "ERR");
414:         return;
415:     }
416: }
417:
418: /**
419:  * Keyword specific search of the book list.
420:  *
421:  * @param cbr the cbr
422:  * @param cpr the cpr
423:  */
424: public static void keywordSearchServer( BufferedReader cbr, PrintWri
ter cpr){
425:
426:     String foundID = "";
427:     String foundBookIDs = "";
428:
429:     System.out.println("SERVERLOG: In keywordSearchServer Method
");
430:
431:     sendMessage(cpr, "OK");
432:     String req = recvMessage(cbr);
433:     String targetKeyword = "";
434:     if(req != null){
435:         System.out.println(req);
436:         String[] tokens = req.split(":");
437:         targetKeyword = tokens[2];
438:         //sendMessage(cpr, "OK");
439:     } else {
440:         sendMessage(cpr, "ERR");
441:         return;
442:     }
443: }
444: /**
445:  * req = recvMessage(cbr);
446:  * if(req != null){
447:  *     System.out.println(req);
448:  *     sendMessage(cpr, "OK");
449:  * } else {
450:  *     sendMessage(cpr, "ERR");
451:  *     return;
452:  * } */
453: System.out.println("SERVERLOG: Searching for " + targetKeywor
d);
454: int i = 0;
455: for(Book book : bookList){
456:     System.out.println(book.getDateWritten());

```

```

457:         if(((book.getTitle()).toLowerCase()).contains(target
Keyword.toLowerCase()) ||
458:            ((book.getAuthor()).toLowerCase()).c
ontains(targetKeyword.toLowerCase()) ||
459:            ((book.getLanguage()).toLowerCase())
.contains(targetKeyword.toLowerCase()) ||
460:            ((book.getDateWritten()).equals(targ
etKeyword))){
461:             foundID = book.getBookID();
462:             System.out.println("SERVERLOG: Middle of key
wordSearchServer Method, found bookID: " + foundID);
463:             i++;
464:             if(i <= 1 ){
465:                 foundBookIDs = foundID;
466:
467:             } else if (i >= 2 ){
468:                 foundBookIDs+= "," + foundID;
469:             }
470:         }
471:     }
472:
473:     if(foundBookIDs != null){
474:         System.out.println("SERVERLOG: Sending " +foundBookI
Ds);
475:         sendMessage(cpr, foundBookIDs);
476:     } else {
477:         System.out.println("SERVERLOG: Did Not Find Anything
");
478:         sendMessage(cpr, "ERR");
479:         return;
480:     }
481: }
482:
483: }
484:
485: /**
486:  * Pick specific search for the book list.
487:  *
488:  * @param cbr the cbr
489:  * @param cpr the cpr
490:  */
491: public static void searchServer(BufferedReader cbr, PrintWriter cpr)
{
492:
493:     //System.out.println("SERVERLOG: In searchServer method");
494:
495:     String req = recvMessage(cbr);
496:     sendMessage(cpr, "OK");
497:     req = recvMessage(cbr);
498:     String searchChoice = null;
499:     if(req != null) {
500:         //System.out.println(req);
501:         String[] tokens = req.split(":");
502:         searchChoice = tokens[2];
503:     } else {
504:         sendMessage(cpr, "ERR");
505:         return;
506:     }
507:
508:     switch(searchChoice) {
509:     case "author":
510:         //System.out.println("SERVERLOG: Entering authorSear
chServer method");
511:         authorSearchServer( cbr, cpr);
512:         return;
513:     case "title":
514:         titleSearchServer( cbr, cpr);
515:         return;
516:

```

```

517:         case "keyword":
518:             keywordSearchServer( cbr, cpr);
519:             return;
520:         default:
521:             sendMessage(cpr, "ERR");
522:             return;
523:     }
524:
525:     //System.out.println("SERVERLOG: DEBUG: loginServer Not Ente
red");
526: }
527:
528: /**
529:  * Check out the user requested book.
530:  *
531:  * @param book the book
532:  */
533: public static boolean checkOutBook(Book book){
534:     List<Book> tempUserBookList = new ArrayList<Book>();
535:     synchronized (user) {
536:         tempUserBookList = user.getCheckedOutBooks();
537:         if(tempUserBookList.size() <= 5){
538:             user.checkOutBook(book);
539:             return true;
540:         } else {
541:             return false;
542:         }
543:     }
544: }
545:
546: /**
547:  * Perform Borrow on the server.
548:  *
549:  * @param cbr the cbr
550:  * @param cpr the cpr
551:  */
552: public static void borrowServer(BufferedReader cbr, PrintWriter cpr)
{
553:
554:     boolean userListFull = false;
555:
556:     String resp = recvMessage(cbr);
557:     System.out.println(resp);
558:     sendMessage(cpr, "OK");
559:
560:     String req = recvMessage(cbr);
561:     if(req.equals("borrow")) {
562:         sendMessage(cpr, "OK");
563:         req = recvMessage(cbr);
564:     }
565:     String targetBookID = "";
566:     if(req != null) {
567:         System.out.println(req);
568:         String[] tokens = req.split(":");
569:         if(tokens == null){
570:             sendMessage(cpr, "ERR");
571:             return;
572:         }
573:         targetBookID = tokens[2];
574:         if(targetBookID == null){
575:             sendMessage(cpr, "ERR");
576:             return;
577:         }
578:         Book book = bookIDSearchServer(targetBookID);
579:         userListFull = checkOutBook(book);
580:         if(userListFull){
581:             book.printElements();
582:             String bookTitle = book.getTitle();
583:

```

```

584:         System.out.println(bookTitle);
585:         sendMessage(cpr, bookTitle);
586:     } else {
587:         sendMessage(cpr, "ERR");
588:         return;
589:     }
590: }
591: }
592: }
593:
594:
595: /**
596:  * Check in the user requested book.
597:  *
598:  * @param book the book
599:  */
600: public static void checkInBook(Book book){
601:
602:     synchronized (user) {
603:         user.checkInBook(book);
604:     }
605:     System.out.println("SERVERLOG: Checking Back In: " + book.ge
tBookID());
606: }
607:
608: /**
609:  * Return the book to the server.
610:  *
611:  * @param cbr the cbr
612:  * @param cpr the cpr
613:  */
614: public static void returnBookServer(BufferedReader cbr, PrintWriter
cpr){
615:
616:     String resp = recvMessage(cbr);
617:     sendMessage(cpr, "OK");
618:     String req = recvMessage(cbr);
619:     String targetBookID = "";
620:     if(req != null) {
621:         System.out.println(req);
622:         sendMessage(cpr, "OK");
623:         String[] tokens = req.split(":");
624:         targetBookID = tokens[2];
625:         Book book = bookIDSearchServer(targetBookID);
626:         checkInBook(book);
627:         //sendMessage(cpr, "OK");
628:     } else{
629:         sendMessage(cpr, "ERR");
630:         return;
631:     }
632: }
633:
634: }
635:
636: /**
637:  * List the currently checked out books from the server.
638:  *
639:  * @param cbr the cbr
640:  * @param cpr the cpr
641:  */
642: public static void listCurrentBooksServer( BufferedReader cbr, Print
Writer cpr){
643:
644:     String booksOut = "";
645:     List<Book> tempUserBookList;
646:     synchronized (user) {
647:         tempUserBookList = user.getCheckedOutBooks();
648:     }
649:     if(!tempUserBookList.isEmpty()){

```

```

650:         for(Book book : tempUserBookList){
651:             booksOut+= book.getTitle() + " - " + book.ge
tBookID() + ",";
652:         }
653:         System.out.println("SERVERLOG: In listCurrentBook me
thod");
654:         System.out.println(booksOut);
655:         sendMessage(cpr,booksOut);
656:     } else if(tempUserBookList.isEmpty()){
657:         sendMessage(cpr, "ERR");
658:         return;
659:     }
660:
661: }
662:
663:
664: /* (non-Javadoc)
665:  * @see AppServer#run()
666:  */
667: @Override
668: public void run() {
669:     // TODO Auto-generated method stub
670:
671:     boolean stop = false;
672:     boolean resetRecv = false;
673:
674:     try {
675:
676:         BufferedReader cbr = new BufferedReader(new InputStr
eamReader(cs.getInputStream()));
677:         PrintWriter cpr = new PrintWriter(cs.getOutputStream
());
678:
679:         while(!stop){
680:             String req = recvMessage(cbr);
681:             while(!resetRecv){
682:                 if(req == null){
683:                     sendMessage(cpr, "ERR");
684:                 }
685:                 if(req.equals("reset")){
686:                     //System.out.println(req);
687:                     sendMessage(cpr, "OK");
688:                     resetRecv = true;
689:                 }
690:                 req = recvMessage(cbr);
691:             }
692:             //req = recvMessage(cbr);
693:             //System.out.println(req);
694:             //System.out.println("SERVERLOG: Checking Fu
nction Request");
695:             /*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");*/
696:
697:             switch(req) {
698:                 case "login":
699:                     System.out.println("SERVERLOG: Enter
ing Login Method");
700:
701:                     loginServer(cbr, cpr);
702:                     break;
703:                 case "logout":
704:                     if(isLoggedInServer(cbr, cpr)){
705:                         logoutServer(cbr, cpr);
706:                         cbr.close();
707:                         cpr.close();

```

```

711:             cs.close();
712:             stop = true;
713:         }
714:         break;
715:     case "search":
716:         //sendMessage(cpr, "OK");
717:         //req = recvMessage(cbr);
718:         //sendMessage(cpr, "OK");
719:         searchServer(cbr, cpr);
720:         break;
721:     case "borrow":
722:         sendMessage(cpr, "OK");
723:         //req = recvMessage(cbr);
724:         searchServer(cbr, cpr);
725:         if(isLoggedInServer(cbr, cpr)){
726:             borrowServer(cbr, cpr);
727:         }
728:         break;
729:     case "return":
730:         sendMessage(cpr, "OK");
731:         req = recvMessage(cbr);
732:         listCurrentBooksServer(cbr, cpr);
733:         req = recvMessage(cbr);
734:         sendMessage(cpr, "OK");
735:         if(isLoggedInServer(cbr, cpr)){
736:             returnBookServer(cbr, cpr);
737:         }
738:         break;
739:     case "listCurrentBooks":
740:         listCurrentBooksServer(cbr, cpr);
741:         break;
742:     case "isLoggedIn"://Should never be called f
rom this point of execution, will be checked before others
743:         isLoggedInServer(cbr, cpr);
744:     case "7":
745:         stop = true;
746:         cbr.close();
747:         cpr.close();
748:         cs.close();
749:     default:
750:         //System.out.println("Please Select
A Valid Option(1-7)");
751:         break;
752:     }
753:     resetRecv = false;
754: }
755:
756: //cbr.close();
757: //cpr.close();
758: //cs.close();
759:
760: } catch (IOException e) {
761:     // TODO Auto-generated catch block
762:     e.printStackTrace();
763: }
764:
765: }
766:
767: }
768:

```

```

1: package projects;
2:
3: import java.io.BufferedReader;
4: import java.io.IOException;
5: import java.io.InputStreamReader;
6: import java.io.PrintWriter;
7: import java.net.Socket;
8: import java.net.UnknownHostException;
9: import java.util.logging.Logger;
10:
11:
12: /**
13:  * The Class LibraryClient.
14:  */
15:
16: /**
17:  * @author j.callado
18:  */
19: public class LibraryClient {
20:
21:     /**
22:      * Send message.
23:      *
24:      * @param spr the spr
25:      * @param mess the mess
26:      */
27:     public static void sendMessage(PrintWriter spr, String mess) {
28:         int len = mess.length();
29:         spr.print(String.format("%03d",len)+"":"+mess");
30:         spr.flush();
31:     }
32:
33:     /**
34:      * Recv message.
35:      *
36:      * @param sbr the sbr
37:      * @return the string
38:      */
39:     public static String recvMessage(BufferedReader sbr) {
40:         try {
41:             char[] slen = new char[4];
42:             sbr.read(slen,0,4);
43:             int len = Integer.parseInt(new String(slen,0,3));
44:             char[] sdata = new char[len];
45:             sbr.read(sdata,0,len);
46:             return new String(sdata);
47:         } catch (IOException e) {
48:             // TODO Auto-generated catch block
49:             e.printStackTrace();
50:             return null;
51:         }
52:     }
53:
54:     public static void loginNewUser(Socket s, BufferedReader sbr, PrintW
riter spr, BufferedReader br){
55:
56:         try{
57:
58:             LibraryClient.sendMessage(spr, "newUser");
59:             String resp = LibraryClient.recvMessage(sbr);
60:             if (resp!= null) {
61:                 System.out.println(resp);
62:                 if(resp.equals("ERR")){
63:                     System.out.println("SYSTEM LOG: Erro
r Received In loginNewUser Method");
64:                     System.out.println("Returning to Mai
n Menu");
65:                     return;
66:                 }

```

```

67:         }
68:
69:         System.out.println("Welcome to the Online Digital Li
brary Service!");
70:         System.out.println("Here, We Will Create Your Person
alized Account So You Can Return to Your Selections!");
71:         System.out.println("Please Create Your userName(a-zA
-Z0-9): ");
72:         String userName = br.readLine();
73:
74:         LibraryClient.sendMessage(spr, "login:String:"+userN
ame);
75:         resp = LibraryClient.recvMessage(sbr);
76:         if(resp!=null){
77:             System.out.println("Got:"+resp);
78:             if(resp.equals("ERR")){
79:                 System.out.println("SYSTEM LOG: Erro
r Received, Invalid userID Creation");
80:                 System.out.println("Returning to Mai
n Menu");
81:                 return;
82:             }
83:
84:
85:             System.out.println("Creat A Unique Password For Your
Account: ");
86:             String password = br.readLine();
87:
88:             LibraryClient.sendMessage(spr, "login:String:" + pas
sword);
89:             resp = LibraryClient.recvMessage(sbr);
90:             if(resp!=null){
91:                 System.out.println("Got:"+resp);
92:                 if(resp.equals("ERR")){
93:                     System.out.println("SYSTEM LOG: Erro
r Received, Invalid Password Creation");
94:                     System.out.println("Returning to Mai
n Menu");
95:                     return;
96:                 }
97:
98:
99:                 if(resp.equals("OK")){
100:                     System.out.println("You Are Now Logged In!")
101:                 }
102:
103:             } catch (UnknownHostException e) {
104:                 // TODO Auto-generated catch block
105:                 e.printStackTrace();
106:             } catch (IOException e) {
107:                 // TODO Auto-generated catch block
108:                 e.printStackTrace();
109:             }
110:         }
111:
112:
113:         public static void loginReturningUser(Socket s, BufferedReader sbr,
PrintWriter spr, BufferedReader br){
114:
115:             try{
116:
117:                 LibraryClient.sendMessage(spr, "returningUser");
118:                 String resp = LibraryClient.recvMessage(sbr);
119:                 if (resp!= null) {
120:                     System.out.println(resp);
121:                     if(resp.equals("ERR")){
122:                         System.out.println("SYSTEM LOG: Erro
r Received In ReturningUser Method");

```

```

123:                System.out.println("Returning to Mai
n Menu");
124:                return;
125:            }
126:        }
127:        System.out.println("Welcome Back! Please Provide You
r Existing Account Credentials Below");
128:        System.out.println("Enter Your Existing userName: ")
;
129:        String userName = br.readLine();
130:        LibraryClient.sendMessage(spr, "login:String:"+ user
Name);
131:        resp = LibraryClient.recvMessage(sbr);
132:        if(resp!=null){
133:            System.out.println("Got:"+resp);
134:            if(resp.equals("ERR")){
135:                System.out.println("SYSTEM LOG: Error
Received, Invalid userID Provided");
136:                System.out.println("Returning to Mai
n Menu");
137:                return;
138:            }
139:            System.out.println("Enter your password: ");
140:            String password = br.readLine();
141:            LibraryClient.sendMessage(spr, "login:String:"+passw
ord);
142:            resp = LibraryClient.recvMessage(sbr);
143:            if(resp!=null){
144:                System.out.println("Got:"+resp);
145:                if(resp.equals("ERR")){
146:                    System.out.println("SYSTEM LOG: Error
Received, Invalid Password Provided");
147:                    System.out.println("Returning to Mai
n Menu");
148:                    return;
149:                }
150:                if(resp.equals("OK")){
151:                    System.out.println("You Are Now Logged In!")
;
152:                }
153:            } catch (UnknownHostException e) {
154:                // TODO Auto-generated catch block
155:                e.printStackTrace();
156:            } catch (IOException e) {
157:                // TODO Auto-generated catch block
158:                e.printStackTrace();
159:            }
160:        }
161:    }
162:    /**
163:     * User sends the login request.
164:     *
165:     * @param s the s
166:     * @param sbr the sbr
167:     * @param spr the spr
168:     * @param br the br
169:     */
170:    public static void login(Socket s, BufferedReader sbr, PrintWriter s
pr, BufferedReader br){
171:        try{
172:            boolean back = false;
173:            LibraryClient.sendMessage(spr, "login");
174:            String resp = LibraryClient.recvMessage(sbr);
175:            if (resp!= null) {
176:                System.out.println(resp);
177:                if(resp.equals("ERR")){
178:                    System.out.println("SYSTEM LOG: Error
Received, Returning to Main Menu");
179:                    return;
180:                }
181:                while(!back){
182:                    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");
183:                    String userChoice = br.readLine();
184:                    switch(userChoice) {
185:                        case "1":
186:                            System.out.println("DEBUG: Entering
loginNewUser Method");
187:                            loginNewUser(s, sbr, spr, br);
188:                            return;
189:                        case "2":
190:                            System.out.println("DEBUG: Entering
loginReturningUser Method");
191:                            loginReturningUser(s, sbr, spr, br);
192:                            return;
193:                        case "3":
194:                            back = true;
195:                            System.out.println("Returning to Mai
n Menu");
196:                            return;
197:                        default:
198:                            System.out.println("Please Select a
Valid Option(1-3)");
199:                            break;
200:                    }
201:                    System.out.println("SYSTEM LOG: Exiting login Method
, Returning to Main Menu");
202:                } catch (UnknownHostException e) {
203:                    // TODO Auto-generated catch block
204:                    e.printStackTrace();
205:                } catch (IOException e) {
206:                    // TODO Auto-generated catch block
207:                    e.printStackTrace();
208:                }
209:            }
210:            /**
211:             * Checks if it is logged in by asking the server to see if it has m
aintained the authentication token.
212:             *
213:             * @param sbr the sbr
214:             * @param spr the spr
215:             * @return true, if is logged in
216:             */
217:            public static boolean isLoggedIn(BufferedReader sbr, PrintWriter spr

```



```

){
241:
242:         //try{
243:
244:         boolean loggedIn = false;
245:
246:         System.out.println("Checking Login Status...");
247:
248:         LibraryClient.sendMessage(spr, "isLoggedIn");
249:         String resp = LibraryClient.recvMessage(sbr);
250:         if (resp != null) {
251:             System.out.println(resp);
252:             if(resp.equals("OK")){
253:                 loggedIn = true;
254:                 System.out.println("You Are Logged In!");
255:                 //return loggedIn;
256:             } else if(resp.equals("ERR")){
257:                 System.out.println("SYSTEM LOG: Error Received, You Must Login First!");
258:             }
259:         }
260:
261:         //System.out.println("You Are Logged In!");
262:         //LibraryClient.sendMessage(spr, "OK");
263:
264:         /*
265:         } catch (UnknownHostException e) {
266:             // TODO Auto-generated catch block
267:             e.printStackTrace();
268:         } catch (IOException e) {
269:             // TODO Auto-generated catch block
270:             e.printStackTrace();
271:         }*/
272:         System.out.println(loggedIn);
273:         return loggedIn;
274:
275:     }
276:
277:     /**
278:     * Send logout request to the server.
279:     * Note: Upon receiving an Error when logged in, the user will be logged out to ensure that his/her information cannot be modified and
280:     * any other unwarranted behavior.
281:     *
282:     * @param s the s
283:     * @param sbr the sbr
284:     * @param spr the spr
285:     * @param br the br
286:     */
287:     public static void logout(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){
288:
289:         Boolean tokenActive = false;
290:         tokenActive = isLoggedIn(sbr, spr);
291:         System.out.println(tokenActive);
292:         if(tokenActive){
293:
294:             LibraryClient.sendMessage(spr, "logout");
295:             String resp = LibraryClient.recvMessage(sbr);
296:             if (resp!= null) {
297:                 System.out.println(resp);
298:                 if(resp.equals("ERR")){
299:                     System.out.println("SYSTEM LOG: Error Received in logout Method");
300:                     System.out.println("You Are Logged Out!");
301:                     System.out.println("Returning to Main Menu");
302:                     return;
303:
304:                 }
305:
306:                 if(resp.equals("OK")){
307:                     System.out.println("You Are Logged Out!");
308:                 }
309:
310:                 System.out.println("Exiting Client");
311:                 return;
312:             }
313:             return;
314:         }
315:
316:         /**
317:         * Send Author specific search request to the server.
318:         *
319:         * @param s the s
320:         * @param sbr the sbr
321:         * @param spr the spr
322:         * @param br the br
323:         */
324:         public static void authorSearch(Socket s, BufferedReader sbr, PrintWriter spr, BufferedReader br){
325:
326:             try{
327:                 LibraryClient.sendMessage(spr, "search:String:author");
328:
329:                 String resp = LibraryClient.recvMessage(sbr);
330:                 System.out.println("DEBUG: sent author Request");
331:                 System.out.println("Got: "+resp);
332:                 if(resp != null){
333:                     if(resp.equals("ERR")){
334:                         System.out.println("SYSTEM LOG: Error Received From author Request In authorSearch Method");
335:                         System.out.println("Returning to Main Menu");
336:                         return;
337:                     }
338:
339:                     System.out.println("Please Enter The Name Of The Author:");
340:
341:                     String authorName = br.readLine();
342:
343:                     LibraryClient.sendMessage(spr, "search:authorString:"+authorName);
344:
345:                     resp = LibraryClient.recvMessage(sbr);
346:                     LibraryClient.sendMessage(spr, "search:authorString:sendResult");//
347:                     resp = LibraryClient.recvMessage(sbr);//
348:                     if(resp!=null){
349:                         System.out.println("Got:"+resp);
350:                         if(resp.equals("ERR")){
351:                             System.out.println("SYSTEM LOG: Error Received From authorName Request In authorSearch Method");
352:                             System.out.println("Returning to Main Menu");
353:                             return;
354:                         }
355:
356:                         String[] respArray = resp.trim().split(",");
357:                         System.out.println("Displaying Top Author Results");
358:                         for(int i = 0; i < respArray.length; i++){
359:                             System.out.println("Result #" + i + ": " + respArray[i]);
360:                         }
361:                         System.out.println("Please Note The BookID For Your Borrow");
362:                     }
363:                 }
364:             }
365:         }

```

4

[illegible]

```

n Menu");
475:         return;
476:     }
477:     resp = resp.replace(":", "");
478:     String[] respArray = resp.trim().split(",");
479:     System.out.println("Displaying Top Keyword R
esults");
480:     for(int i = 0; i < respArray.length; i++){
481:         System.out.println("Result #" + i + ":
"+ respArray[i]);
482:     }
483:     System.out.println("Please Note The BookID F
or Your Borrow");
484:     }
485:
486:     return;
487:
488:
489:     } catch (UnknownHostException e) {
490:         // TODO Auto-generated catch block
491:         e.printStackTrace();
492:     } catch (IOException e) {
493:         // TODO Auto-generated catch block
494:         e.printStackTrace();
495:     }
496:     System.out.println("DEBUG: keywordSearch Not Entered!");
497:     return;
498:
499: }
500:
501: /**
502:  * Prompt user to select specific Search function.
503:  *
504:  * @param s the s
505:  * @param sbr the sbr
506:  * @param spr the spr
507:  * @param br the br
508:  */
509: public static void search(Socket s, BufferedReader sbr, PrintWriter
spr, BufferedReader br){
510:
511:     try{
512:         boolean back = false;
513:
514:         LibraryClient.sendMessage(spr, "search");
515:         String resp = LibraryClient.recvMessage(sbr);
516:         if (resp != null) {
517:             System.out.println("Got: "+resp);
518:             if(resp.equals("ERR")){
519:                 System.out.println("SYSTEM LOG: Erro
r Received From search Request In search Method");
520:                 System.out.println("Returning to Mai
n Menu");
521:                 return;
522:             }
523:         }
524:         //resp = LibraryClient.recvMessage(sbr);//new
525:         while(!back){
526:             System.out.println("Enter 1-4 To Select Your
Search (Case-Sensitive):" + '\n' + "1) Author" + '\n'
+ "2) Title" + '\n' + "3) Ke
yword" + '\n' + "4) Back to Main Menu");
527:             String userChoice = br.readLine();
528:
529:             switch(userChoice) {
530:                 case "1":
531:                     System.out.println("DEBUG: Entering
authorSearch Method");
532:                     authorSearch( s, sbr, spr, br);

```

```

534:         return;
535:     case "2":
536:         System.out.println("DEBUG: Entering
titleSearch Method");
537:         titleSearch(s, sbr, spr, br);
538:         return;
539:     case "3":
540:         System.out.println("DEBUG: Entering
keywordSearch Method");
541:         keywordSearch(s, sbr, spr, br);
542:         return;
543:     case "4":
544:         back = true;
545:         resp = recvMessage(sbr);
546:         return;
547:     default:
548:         System.out.println("Please Select a
Valid Option(1-4). Note: Search is Case-Sensitive");
549:         break;
550:     }
551: }
552:
553:     } catch (UnknownHostException e) {
554:         // TODO Auto-generated catch block
555:         e.printStackTrace();
556:     } catch (IOException e) {
557:         // TODO Auto-generated catch block
558:         e.printStackTrace();
559:     }
560:
561: }
562:
563:     return;
564: }
565:
566: /**
567:  * Send Borrow request to the server.
568:  *
569:  * @param s the s
570:  * @param sbr the sbr
571:  * @param spr the spr
572:  * @param br the br
573:  */
574: public static void borrow(Socket s, BufferedReader sbr, PrintWriter
spr, BufferedReader br){
575:
576:     Boolean tokenActive = false;
577:     tokenActive = isLoggedIn(sbr, spr);
578:     System.out.println(tokenActive);
579:     if(tokenActive){
580:
581:         try{
582:
583:             LibraryClient.sendMessage(spr, "borrow");
584:             String resp = LibraryClient.recvMessage(sbr);
585:
586:             if(resp!=null){
587:                 System.out.println("Got: "+resp);
588:                 if(resp.equals("ERR")){
589:                     System.out.println("SYSTEM L
OG: Error Received From borrow Request In borrow Method");
590:                     System.out.println("Returnin
g to Main Menu");
591:                     return;
592:                 }
593:             }
594:
595:             System.out.println("Please Enter The BookID

```

```

Of Your Book Request(Only Five Books Can Be Out At Once!) :");
596:         String bookID = br.readLine();
597:
598:         LibraryClient.sendMessage(spr, "borrow:Strin
g:"+bookID);
599:         resp = LibraryClient.recvMessage(sbr);
600:         if(resp!=null){
601:             System.out.println("Got: "+resp);
602:             if(resp.equals("ERR")){
603:                 System.out.println("SYSTEM L
OG: Error Received From bookID Request In borrow Method");
604:                 System.out.println("Your boo
kList might be full. Please Limit to Five Borrow");
605:                 System.out.println("Returnin
g to Main Menu");
606:                 return;
607:             }
608:             System.out.println("Here Is Your Req
uest: "+resp);
609:             System.out.println("Please Enjoy You
r Selection!");
610:         }
611:         return;
612:
613:     } catch (UnknownHostException e) {
614:         // TODO Auto-generated catch block
615:         e.printStackTrace();
616:     } catch (IOException e) {
617:         // TODO Auto-generated catch block
618:         e.printStackTrace();
619:     }
620:
621:     } else {
622:         System.out.println("Returning to the Main Menu");
623:         return;
624:     }
625: }
626:
627: /**
628:  * Send Return book request to the server.
629:  *
630:  * @param s the s
631:  * @param sbr the sbr
632:  * @param spr the spr
633:  * @param br the br
634:  */
635: public static void returnBook(Socket s, BufferedReader sbr, PrintWri
ter spr, BufferedReader br){
636:
637:     boolean isLoggedIn = isLoggedIn(sbr, spr);
638:
639:     if(isLoggedIn){
640:
641:         try{
642:
643:             LibraryClient.sendMessage(spr, "returnBook")
;
644:             String resp = LibraryClient.recvMessage(sbr)
;
645:             if(resp!=null){
646:                 System.out.println("Got: "+resp);
647:                 if(resp.equals("ERR")){
648:                     System.out.println("SYSTEM L
OG: Error Received From returnBook Request In returnBook Method");
649:                     System.out.println("Returnin
g to Main Menu");
650:                     return;
651:                 }
652:             }

```

```

653:
654:         System.out.println("Please Enter The BookID
Of\nThe Book You Wish to Return :");
655:         String bookID = br.readLine();
656:
657:         LibraryClient.sendMessage(spr, "returnBook:S
tring:"+bookID);
658:         resp = LibraryClient.recvMessage(sbr);
659:         if(resp!=null){
660:             System.out.println(resp);
661:             if(resp.equals("ERR")){
662:                 System.out.println("SYSTEM L
OG: Error Received From bookID Request In returnBook Method");
663:                 System.out.println("Returnin
g to Main Menu");
664:                 return;
665:             }
666:             if(resp.equals("OK")){
667:                 System.out.println("Your Boo
k Has Been Returned!\nPlease Search For Your Next Selection!");
668:             }
669:             //Just in case something else comes
670:             //System.out.println("Error Received
over the wire besides OK or ERR
, Exiting Program");
671:
672:         }
673:         return;
674:
675:     } catch (UnknownHostException e) {
676:         // TODO Auto-generated catch block
677:         e.printStackTrace();
678:     } catch (IOException e) {
679:         // TODO Auto-generated catch block
680:         e.printStackTrace();
681:     }
682:
683:     //System.out.println("DEBUG: returnBook Not Entered!
");
684:
685:
686:     } else if(!isLoggedIn){
687:         System.out.println("Returning to the Main Menu");
688:         return;
689:     } else {
690:         System.out.println("SYSTEM LOG: Unknown Error Receiv
ed, Exiting Program");
691:
692:     }
693:
694:
695:
696: /**
697:  * List currently checked out books from the server.
698:  *
699:  * @param s the s
700:  * @param sbr the sbr
701:  * @param spr the spr
702:  * @param br the br
703:  */
704: public static void listCurrentBooks(Socket s, BufferedReader sbr, Pr
intWriter spr, BufferedReader br){
705:
706:
707:     //try{
708:
709:         LibraryClient.sendMessage(spr, "listCurrentBooks");
710:         String resp = LibraryClient.recvMessage(sbr);
711:         if(resp!=null){

```

```

712:         System.out.println(resp);
713:         if(resp.equals("ERR")){
714:             System.out.println("You Do Not Currently Hav
e Any Books Checked Out");
715:             System.out.println("Please Search For A Sele
ction Or Borrow Once You Have Found One");
716:             System.out.println("Returning to Main Menu")
;
717:             return;
718:         }
719:         String[] respArray = resp.trim().split(",");
720:         System.out.println("Displaying Your Currently Checke
d Out Books:");
721:         for(int i = 0; i < respArray.length; i++){
722:             System.out.println("Result #" + i + ": " + respAr
ray[i]);
723:         }
724:         System.out.println("Please Note The BookID For Your
Borrow or Return");
725:     }
726:
727:
728:     return;
729:
730:     //System.out.println("DEBUG: listCurrentBooks Not Entered!")
;
731: }
732:
733: /**
734:  * The main method.
735:  *
736:  * @param args the arguments
737:  */
738: public static void main(String[] args) {
739:     // TODO Auto-generated method stub
740:
741:     Boolean exit = false;
742:
743:
744:     try {
745:         Socket s = new Socket("localhost", 45000);
746:         BufferedReader sbr = new BufferedReader(new InputStr
eamReader(s.getInputStream()));
747:         PrintWriter spr = new PrintWriter(s.getOutputStream(
));
748:         BufferedReader br = new BufferedReader(new InputStre
amReader(System.in));
749:         while(!exit){
750:             sendMessage(spr, "reset");
751:             String resp = recvMessage(sbr);
752:             System.out.println("Please Select 1-7 As An
Option:");
753:             System.out.println("1) Login"
+ '\n' + "2) Logout/exit" +
'\n' + "3) Search (Case-Sensitive)" + '\n' + "4) Borrow" + '\n' + "5) Return"
+ '\n' + "6) List Checked Ou
t Books\nEnter a Number: ");
754:             String userChoice = br.readLine();
755:             //String resp;
756:             switch(userChoice) {
757:                 case "1":
758:                     login( s, sbr, spr, br);
759:                     break;
760:                 case "2":
761:                     sendMessage(spr, "logout");
762:
763:             }
764:
765:             if(resp.equals("ERR")){
766:                 System.out.println("You Do Not Currently Hav
e Any Books Checked Out");
767:                 System.out.println("Please Search For A Sele
ction Or Borrow Once You Have Found One");
768:                 System.out.println("Returning to Main Menu")
;
769:                 return;
770:             }
771:             String[] respArray = resp.trim().split(",");
772:             System.out.println("Displaying Your Currently Checke
d Out Books:");
773:             for(int i = 0; i < respArray.length; i++){
774:                 System.out.println("Result #" + i + ": " + respAr
ray[i]);
775:             }
776:             System.out.println("Please Note The BookID For Your
Borrow or Return");
777:         }
778:
779:         return;
780:     } catch (IOException e) {
781:         // TODO Auto-generated catch block
782:         e.printStackTrace();
783:     }
784: }
785:
786: }
787:
788: }
789:
790: }
791:
792: }
793:
794: }
795:
796: }
797:
798: }
799:
800: }
801:
802: }
803:
804: }
805:
806: }
807:
808: }
809:
810: }
811:
812: }
813:
814: }
815:
816: }
817:
818: }
819:
820: }
821:
822: }

```