

**Faculty of Computing & Information Technology**

**Computer Science Department**

CPCS 371 Project

Student Book Sharing System

Computer Networks

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Section** |
| **Hussam Shawly** | **1742403** | **DT** |
| **Omar Al-Qurashi (Leader)** | **1742589** | **DT** |

**December 1, 2019**

**Table of Contents**

[**Introduction** 3](#_Toc26100425)

[**UML use case** 3](#_Toc26100426)

[**Documentation** 4](#_Toc26100427)

[Client: 4](#_Toc26100428)

[Server: 5](#_Toc26100429)

[Client Handler: 6](#_Toc26100430)

[**Sample Output:** 8](#_Toc26100431)

[**Conclusion:** 12](#_Toc26100432)

[**Appendix: Source Codes** 13](#_Toc26100433)

# **Introduction**

This system allows students to upload books on the system, they can also view the uploaded books and they can reserve any book, as long as it is not reserved by another student. To use these functions the students must register in the system and log in.

# **UML use case**

This diagram describes our UML use case for the system:

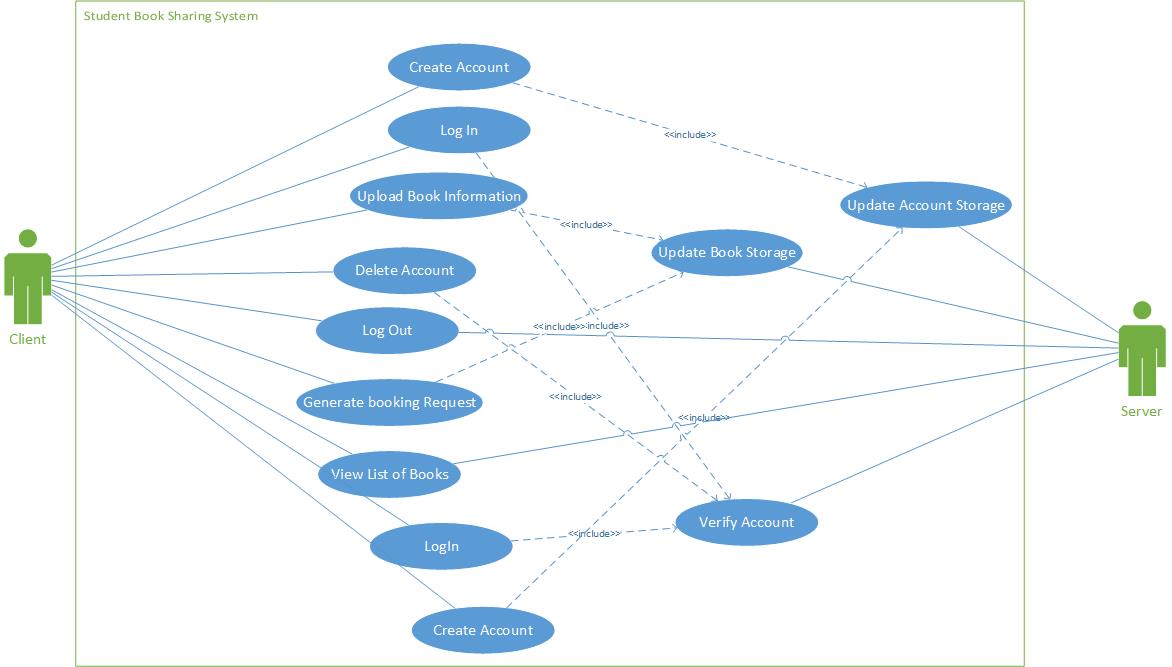
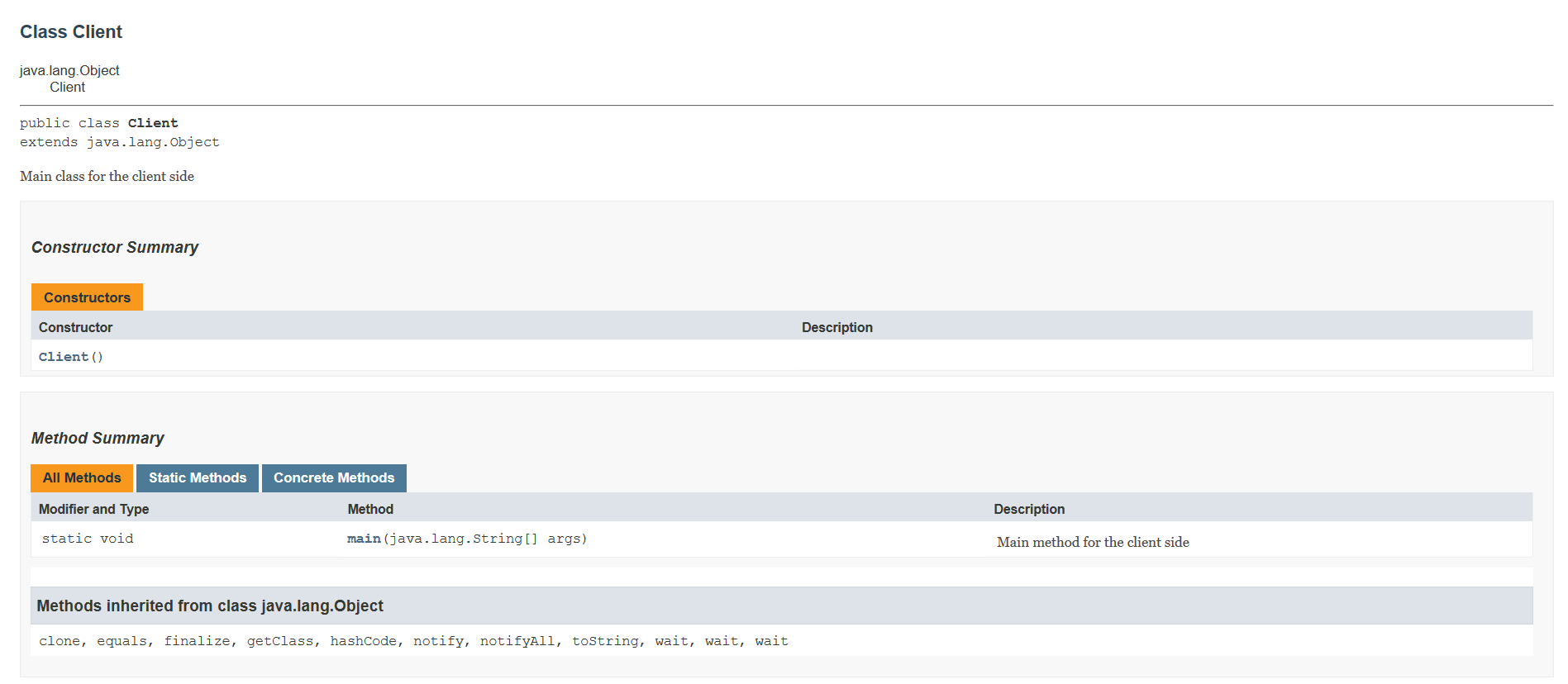


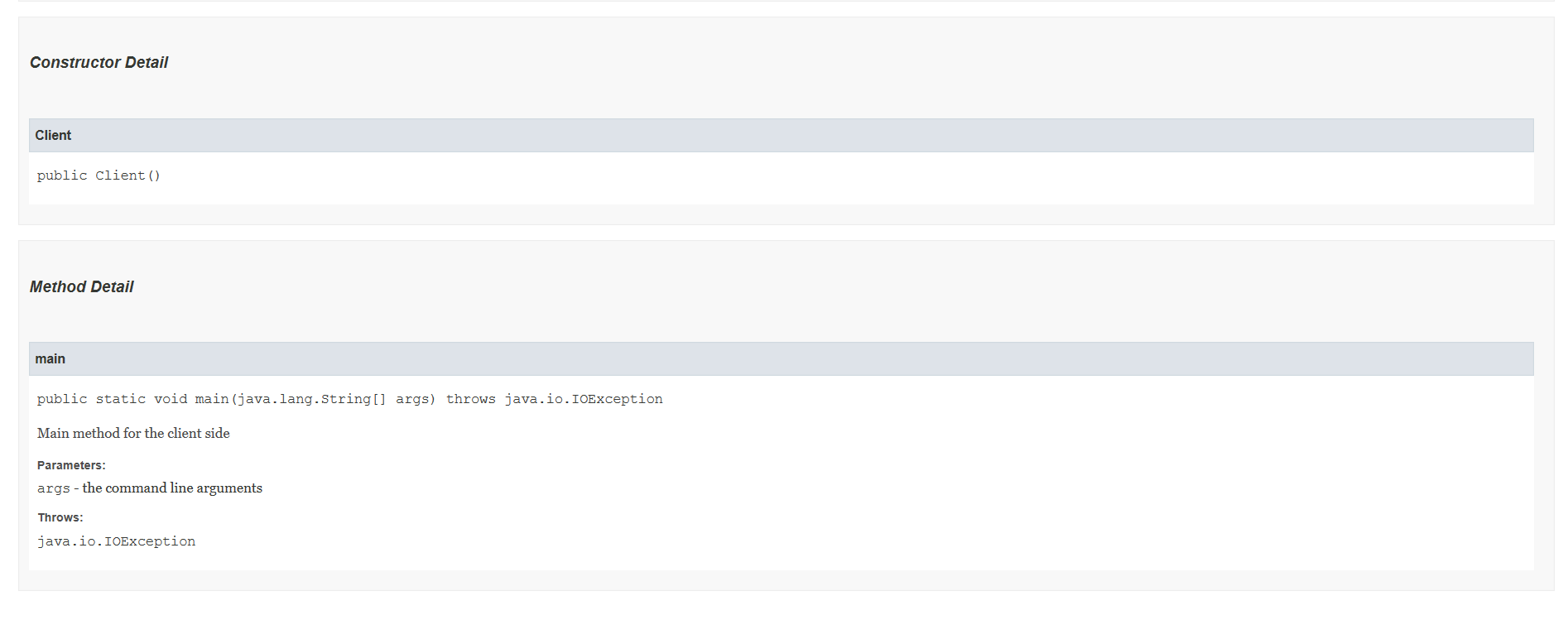
Figure 1

# **Documentation**

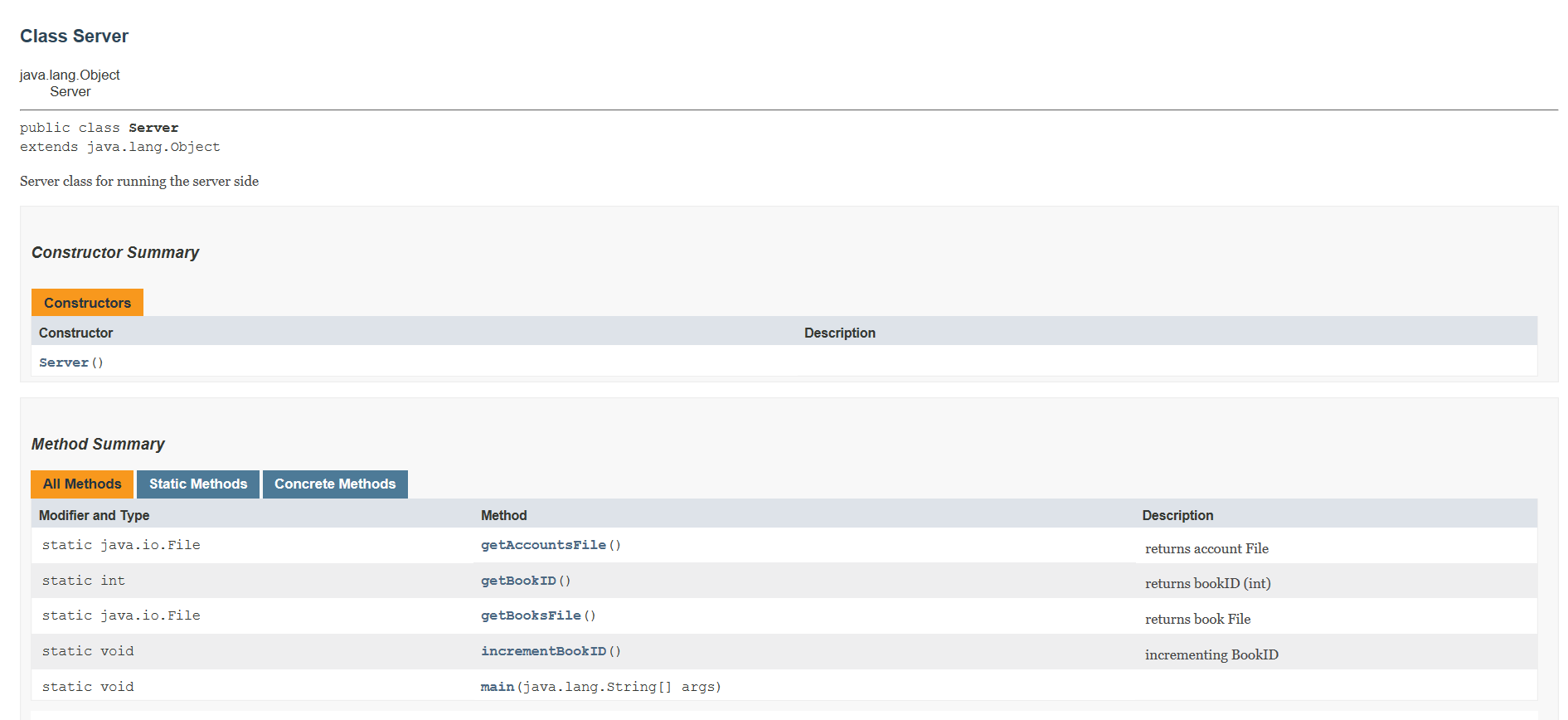
The following Javadoc describes our system:

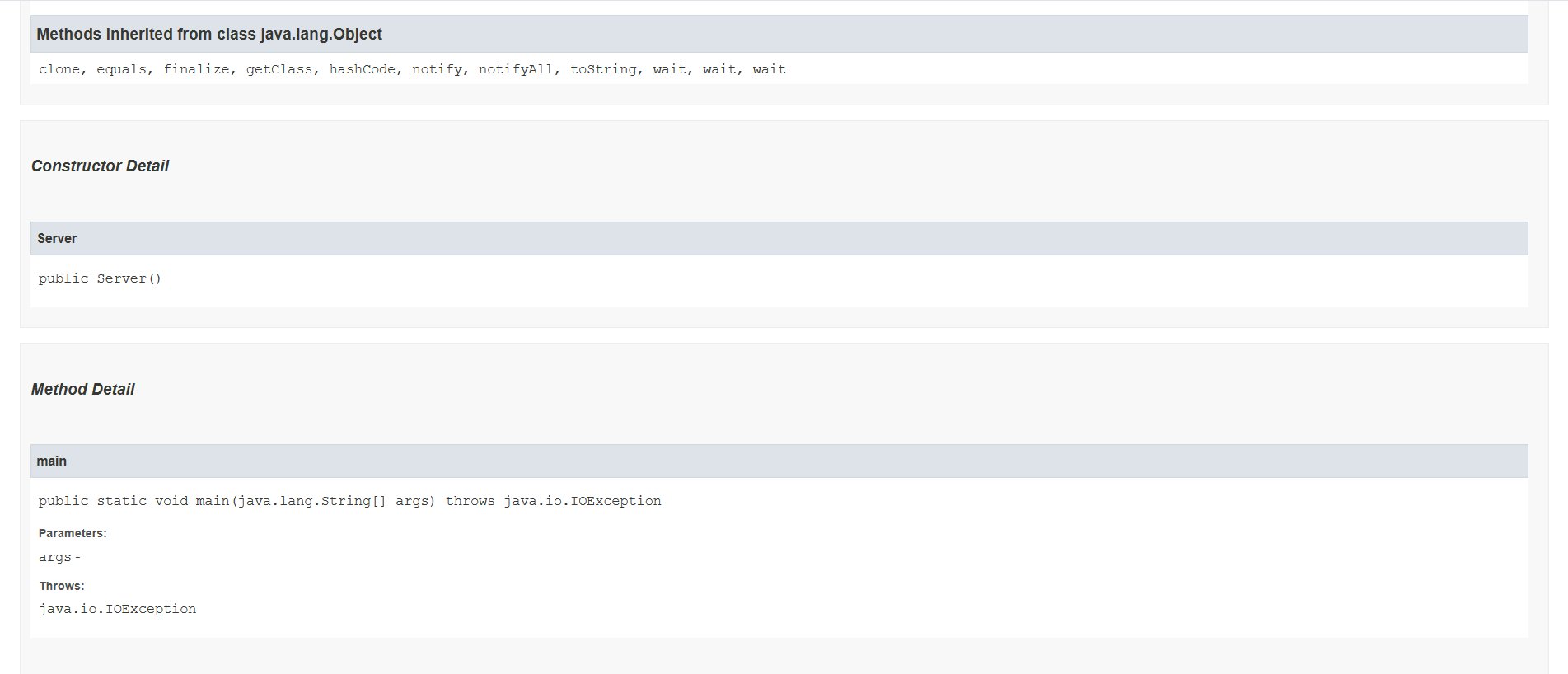
## Client:

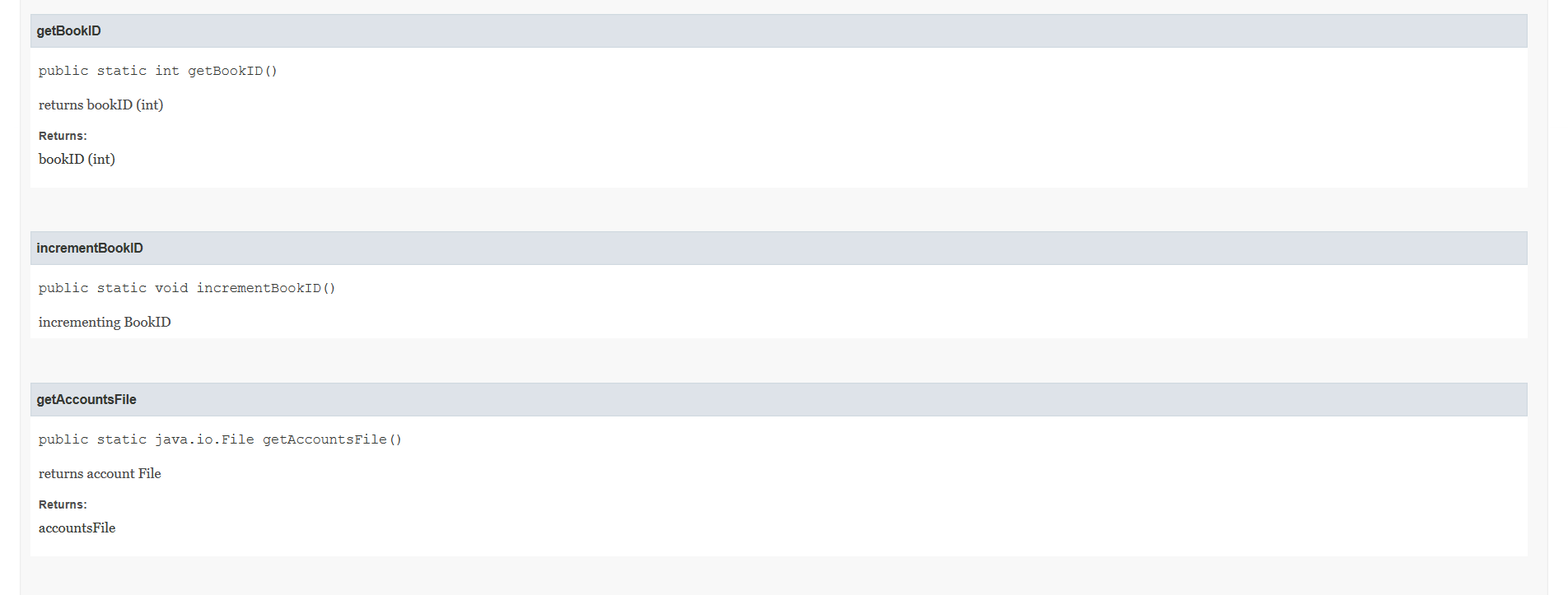
****

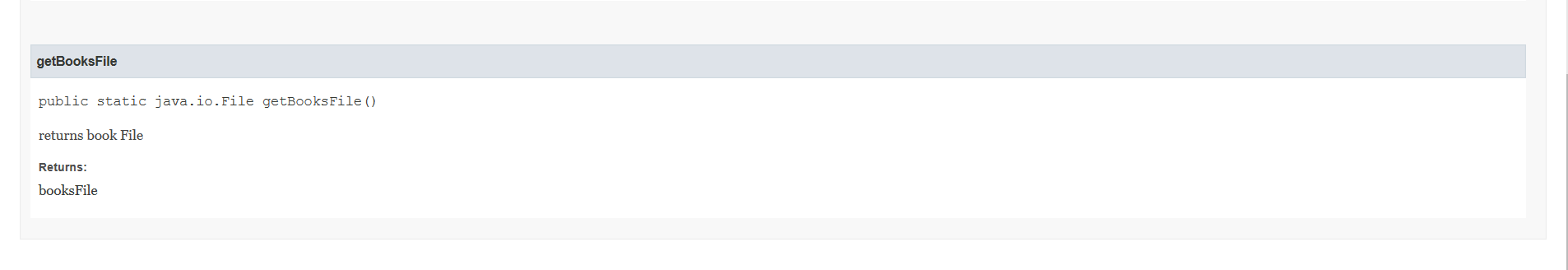
****

## Server:

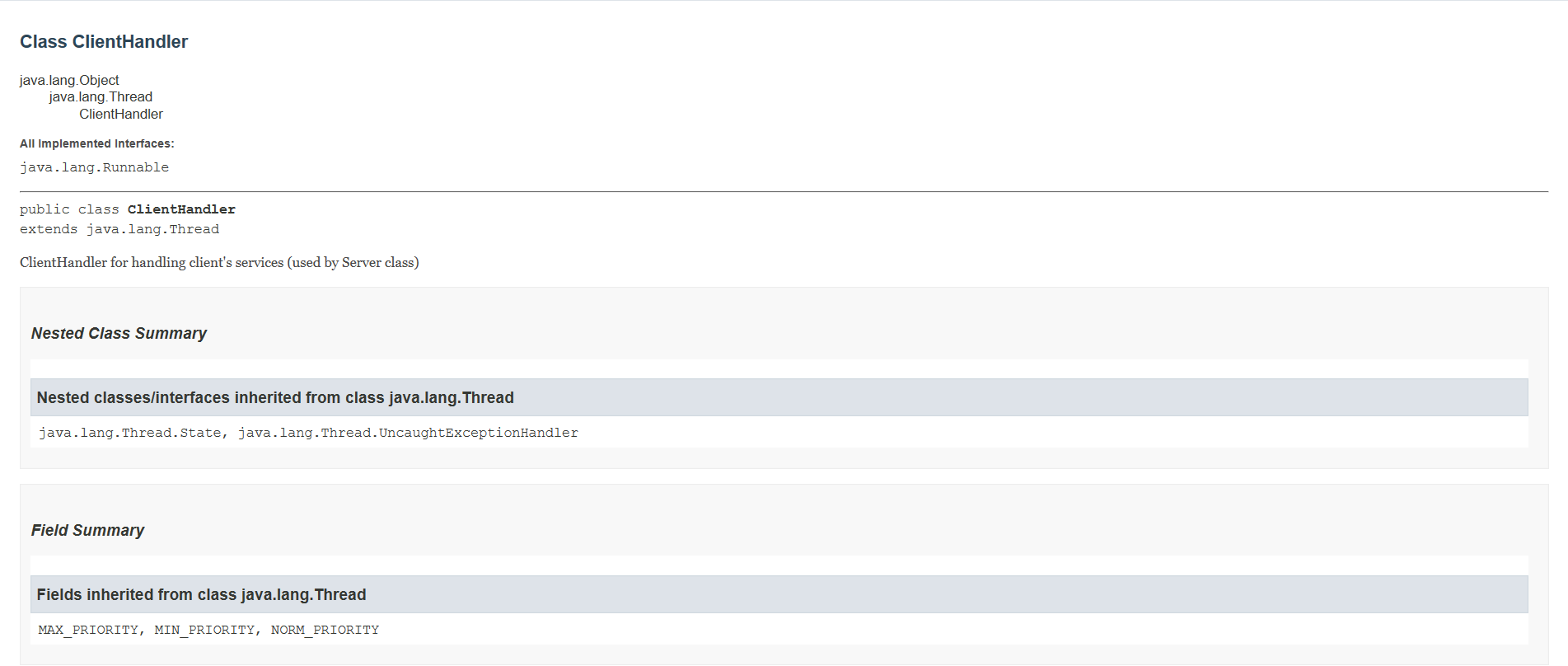
****

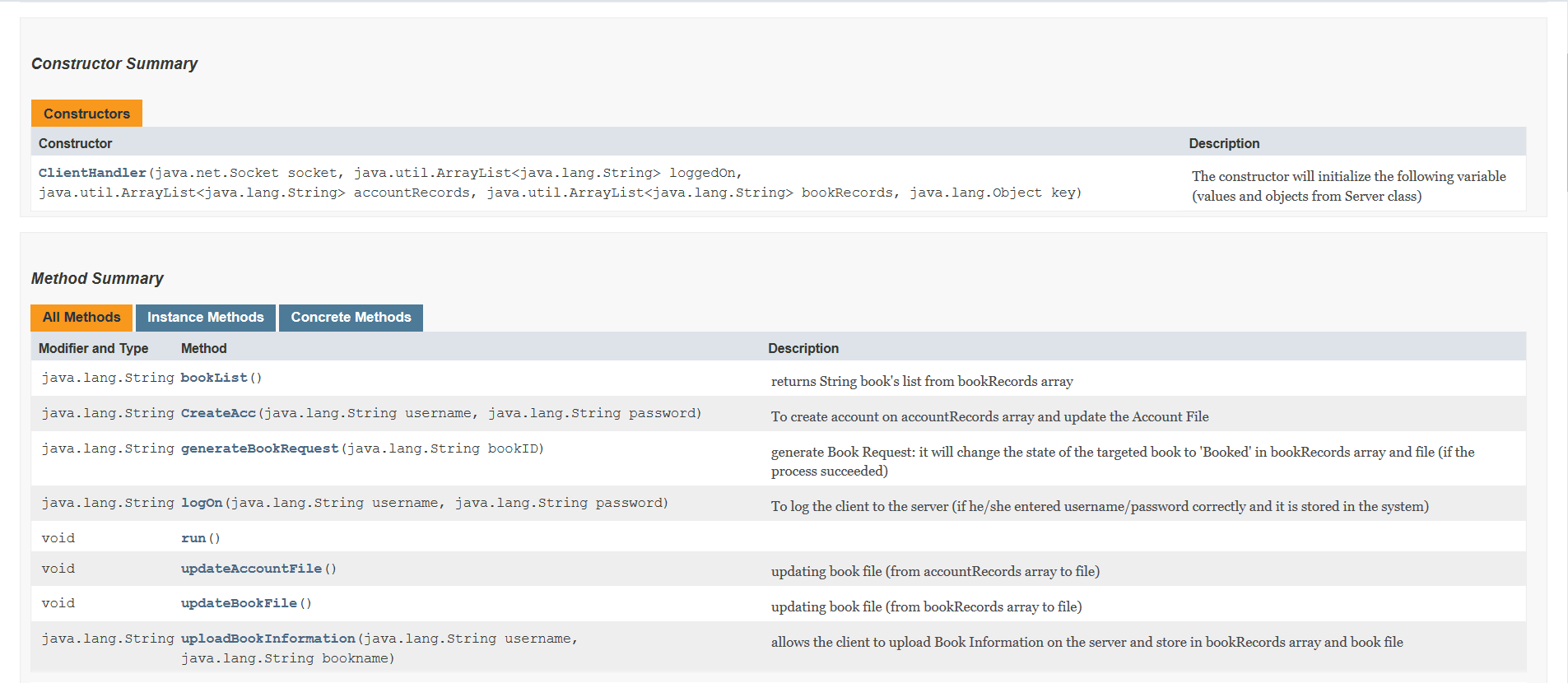
****

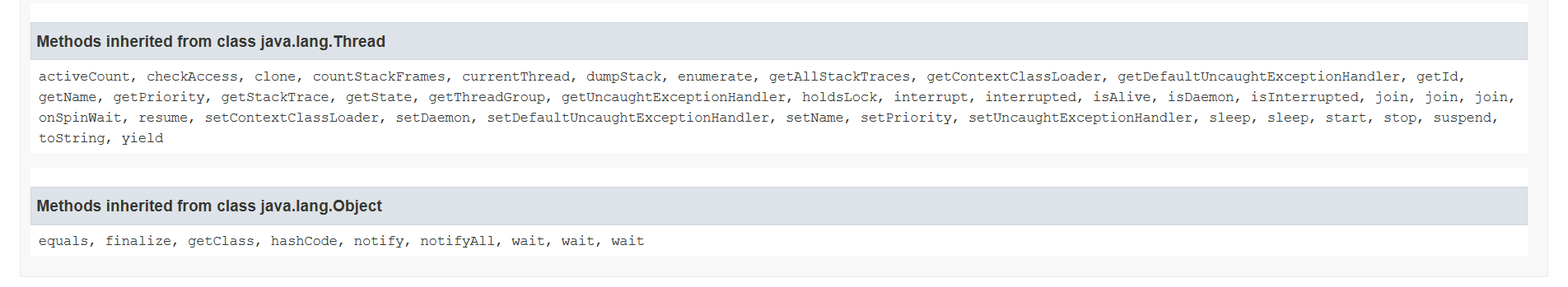
****

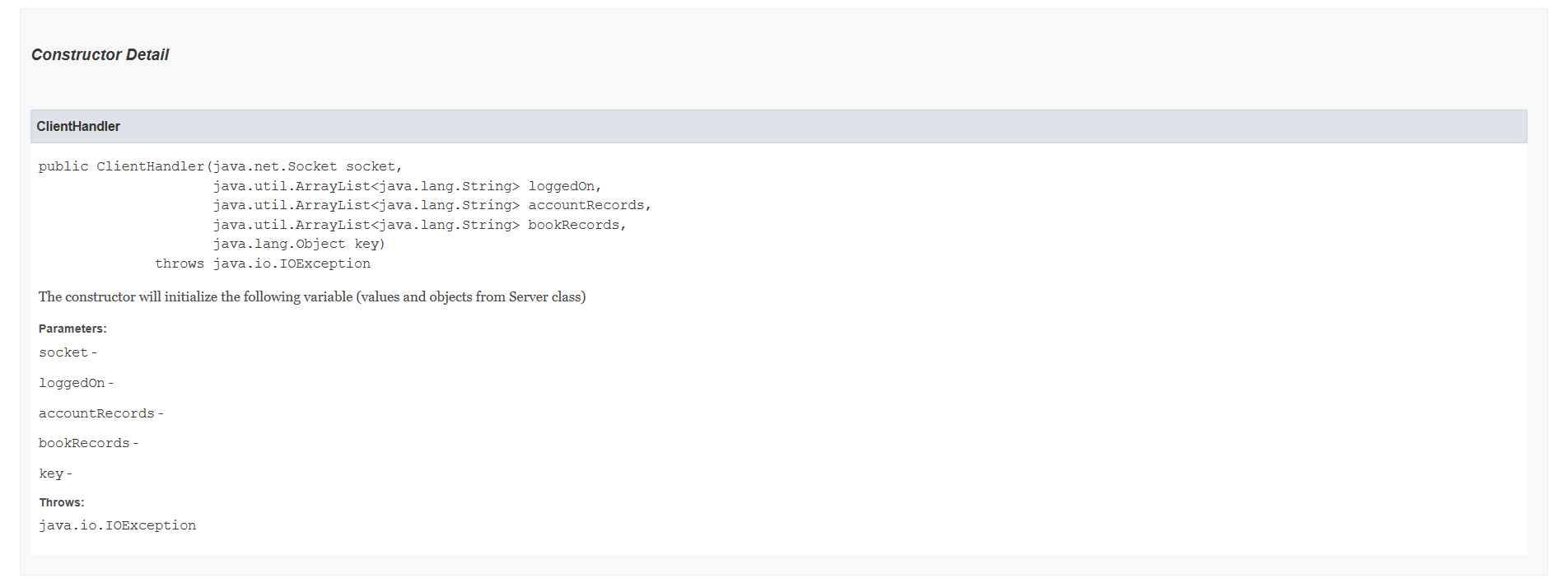
****

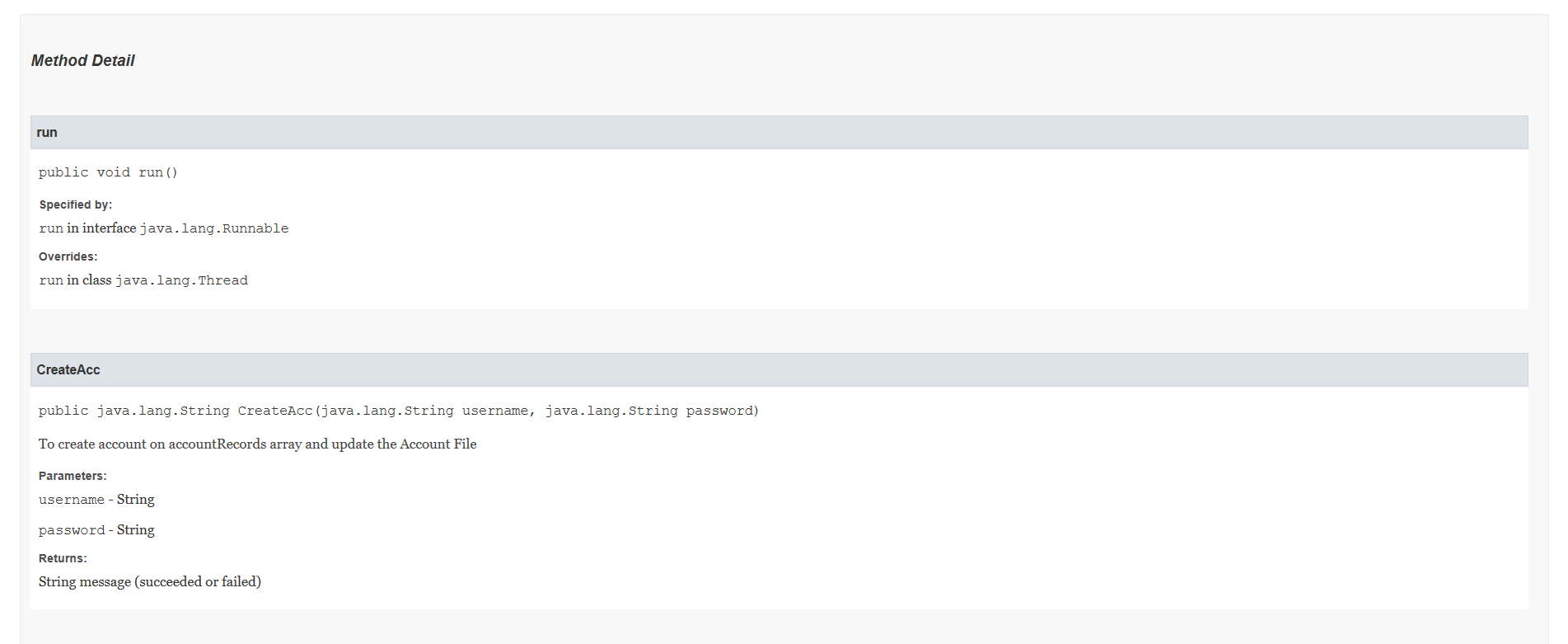
## Client Handler:

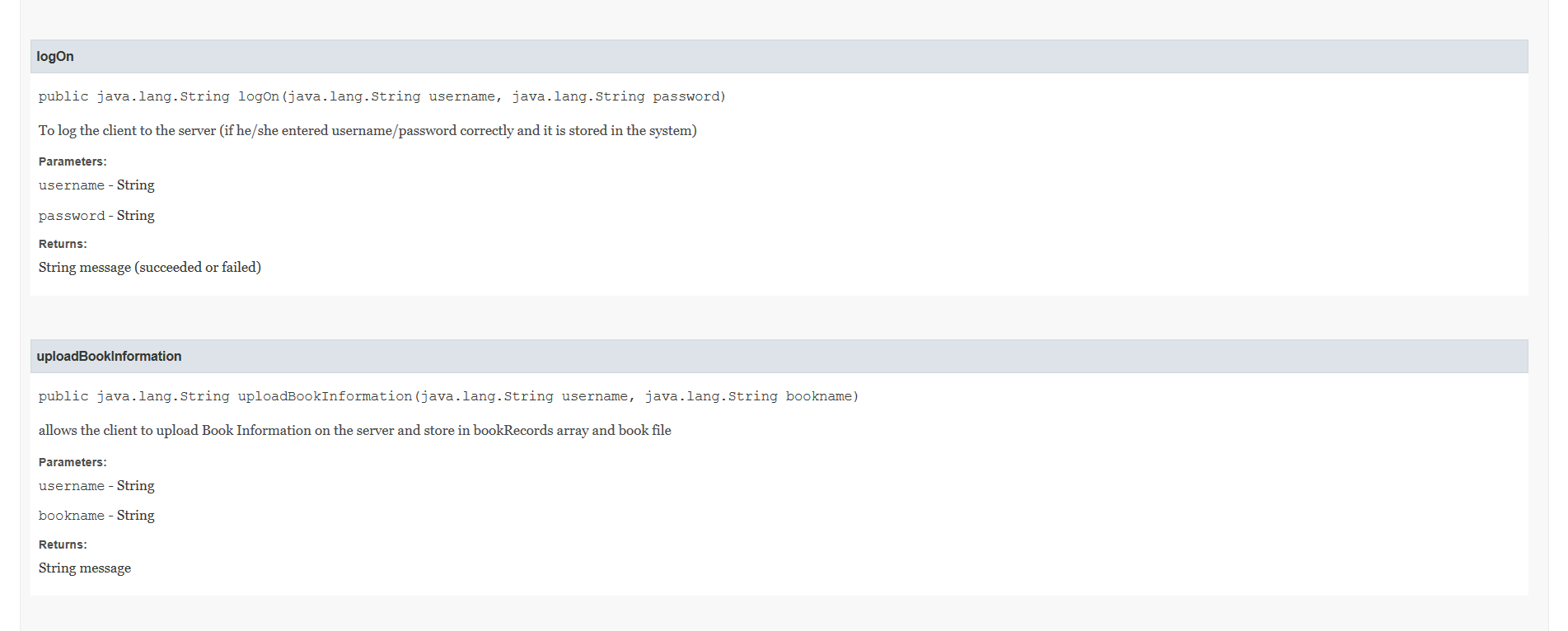
****

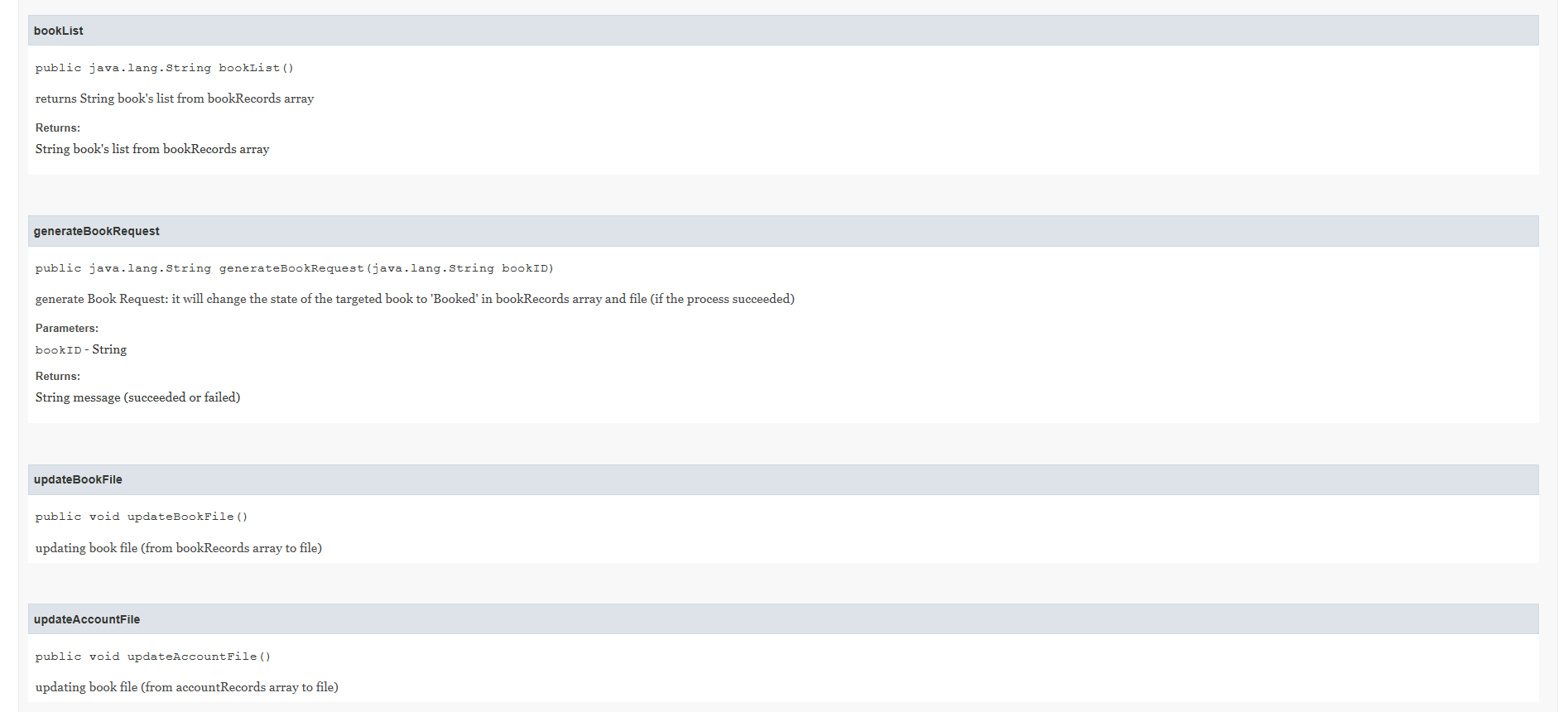
****

****

****

****

****

****

# **Sample Output:**

The sample output of the system will be presented in the next page.

|  |
| --- |
| Output1:  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>java Client  ----- Welcome to Student Book Sharing System ------  01;omar;1234#  20;OK#  02;omar;1234#  20;OK#  03;omar;Android#  23; Invalid format#  03;omar,Android#  22;Information is uploaded successfully on the system #  04;LISTBOOKS#  24;0,Android,omar,Available#  03;omar,CPCS371#  22;Information is uploaded successfully on the system #  03;omar,Software Engineering#  22;Information is uploaded successfully on the system #  04;LISTBOOKS#  24;0,Android,omar,Available#  24;1,CPCS371,omar,Available#  24;2,Software Engineering,omar,Available#  05;omar;3#  16; This book is not available#  05;omar;1#  25;Book has been reserved successfully #  04;LISTBOOKS#  24;0,Android,omar,Available#  24;1,CPCS371,omar,Booked#  24;2,Software Engineering,omar,Available#  06;omar;1234#  26;omar is logged off successfully#  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>java Client  ----- Welcome to Student Book Sharing System ------  02;omar;1234#  20;OK#  06;omar;1234#  26;omar is logged off successfully#  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>  Output2:  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src> java Client  ----- Welcome to Student Book Sharing System ------  02;omar;1234#  17;error (someone logged in by this username)#  01;ahmed;1111#  20;OK#  02;ahmed;1231#  21;Invalid user (or bad password)#  02;ahmed;1111#  20;OK#  04;LISTBOOKS#  24;0,Android,omar,Available#  24;1,CPCS371,omar,Booked#  24;2,Software Engineering,omar,Available#  05;ahmed;1  23; Invalid format#  05;ahmed;1#  26; This book is not available for reservation#  05;ahmed;0#  25;Book has been reserved successfully #  07;ahmed;1234#  21;Invalid user (or bad password)#  07;ahmed;1111#  27;ahmed#  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>  Output 3:  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>java Client  ----- Welcome to Student Book Sharing System ------  01;khaled;a1b2#  20;OK#  02;khaled;a1b2#  20;OK#  04;LISTBOOKS#  24;0,Android,omar,Booked#  24;1,CPCS371,omar,Booked#  24;2,Software Engineering,omar,Available#  03;khaled;Compiler Course#  23; Invalid format#  03;khaled,Compiler Course#  22;Information is uploaded successfully on the system #  06;khaled;a1b2#  26;khaled is logged off successfully#  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>  Output 4:  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>java Client  ----- Welcome to Student Book Sharing System ------  01;khaled;a1b2#  20;OK#  02;khaled;a1b2#  20;OK#  04;LISTBOOKS#  24;0,Android,omar,Booked#  24;1,CPCS371,omar,Booked#  24;2,Software Engineering,omar,Available#  03;khaled;Compiler Course#  23; Invalid format#  03;khaled,Compiler Course#  22;Information is uploaded successfully on the system #  06;khaled;a1b2#  26;khaled is logged off successfully#  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>  Output 5:  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>java Client  ----- Welcome to Student Book Sharing System ------  omar;qw  29;Invalid message#  02;omar,qw#  23; Invalid format#  02;omar,1234#  23; Invalid format#  02;omar;1234#  20;OK#  04;LISTBOOKS#  24;0,Android,omar,Booked#  24;1,CPCS371,omar,Booked#  24;2,Software Engineering,omar,Available#  24;3,Compiler Course,khaled,Available#  24;4,The Book,omar,Available#  04;LISTBOOKS  23; Invalid format#  04,LISTBOOKS#  23; Invalid format#  07;omar;1234#  27;omar#  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>  Output 6:  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>java Client  ----- Welcome to Student Book Sharing System ------  03;omar;1234#  18;You should login first#  05;omar;1234#  18;You should login first#  02;omar;1234#  17;error (someone logged in by this username)#  01;ahmed;1233#  20;OK#  02;ahmed;1233#  20;OK#  03;ahmed,Linear Algebra#  22;Information is uploaded successfully on the system #  04;LISTBOOKS#  24;0,Android,omar,Booked#  24;1,CPCS371,omar,Booked#  24;2,Software Engineering,omar,Available#  24;3,Compiler Course,khaled,Available#  24;4,The Book,omar,Available#  24;5,Linear Algebra,ahmed,Available#  05;ahmed;5#  25;Book has been reserved successfully #  04;LISTBOOKS#  24;0,Android,omar,Booked#  24;1,CPCS371,omar,Booked#  24;2,Software Engineering,omar,Available#  24;3,Compiler Course,khaled,Available#  24;4,The Book,omar,Available#  24;5,Linear Algebra,ahmed,Booked#  06;ahmed;5#  21;Invalid user (or bad password)#  06;ahmed,1234#  23; Invalid format#  07;ahmed,1234#  23; Invalid format#  07;ahmed;5#  21;Invalid user (or bad password)#  06;ahmed;1234#  21;Invalid user (or bad password)#  06;ahmed;1233#  26;ahmed is logged off successfully#  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>  output of the server:  C:\Users\omar-\Documents\NetBeansProjects\Client\_Server\src>java Server  Waiting for client on port 9090...  Just connected to /127.0.0.1:55001  Just connected to /127.0.0.1:55003  Client /127.0.0.1:55003 Waiting for key...  Client /127.0.0.1:55003 holding the key...  Client /127.0.0.1:55003 left the key  Client /127.0.0.1:55001 Waiting for key...  Client /127.0.0.1:55001 holding the key...  Client /127.0.0.1:55001 left the key  Client /127.0.0.1:55001 Waiting for key...  Client /127.0.0.1:55001 holding the key...  Client /127.0.0.1:55001 left the key  Client /127.0.0.1:55001 Waiting for key...  Client /127.0.0.1:55001 holding the key...  Client /127.0.0.1:55001 left the key  Client /127.0.0.1:55001 Waiting for key...  Client /127.0.0.1:55001 holding the key...  Client /127.0.0.1:55001 left the key  Client /127.0.0.1:55001 Waiting for key...  Client /127.0.0.1:55001 holding the key...  Client /127.0.0.1:55001 left the key  ---Client /127.0.0.1:55001 Exited from system---  Client /127.0.0.1:55003 Waiting for key...  Client /127.0.0.1:55003 holding the key...  Client /127.0.0.1:55003 left the key  ---Client /127.0.0.1:55003 Exited from system--- |

# **Conclusion:**

In summery the book sharing system allows multiple students to access the system and do the following:

* Create Account
* Log In
* Upload book information
* View list of books
* Generate booking request
* Log out
* Delete account

# **Appendix: Source Codes**

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.io.File;

import java.io.PrintWriter;

import java.util.Scanner;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\* Server public class for running the server side, ClientHandler class for handling client's services (used by Server class)

\* @author omar-

\*/

public class Server {

/\*\*

\* Main method for the server side

\* @param args the command line arguments

\*/

private static int bookID = 0; // used when uploading book

private static Scanner accountRead; // reading account file

private static Scanner bookRead; // reading book file

private static ArrayList<String> loggedOn = new ArrayList<>(); // for tracking logged On users

private static ArrayList<String> accountRecords = new ArrayList<>();

private static ArrayList<String> bookRecords = new ArrayList<>();

private static File accountsFile = new File("accounts.txt");

private static File booksFile = new File("books.txt");

private static Socket socket;

private static Object key = new Object(); // used for synchronization purposes...

/\*\*

\*

\* @param args

\* @throws IOException

\*/

public static void main(String[] args) throws IOException {

if(!accountsFile.exists()) // if this file does not exist:

{

accountsFile.createNewFile();

System.out.println("Accounts File has been created");

}

if(!booksFile.exists()) // if this file does not exist:

{

booksFile.createNewFile();

System.out.println("Books File has been created");

}

accountRead = new Scanner(accountsFile);

bookRead = new Scanner(booksFile);

while(bookRead.hasNext()){ // storing book's records to bookRecords array:

bookRecords.add(bookRead.nextLine());

}

if(bookRecords.size() != 0){ // this part for initializing 'bookID' variable to last current book id + 1:

bookID = Integer.parseInt(bookRecords.get(bookRecords.size() - 1).split("[,]")[0]); // last Book ID

bookID++;

}

while(accountRead.hasNext()){ // storing account's records to accountRead array:

accountRecords.add(accountRead.nextLine());

}

ServerSocket listener = new ServerSocket(9090);

System.out.println("Waiting for client on port " + listener.getLocalPort() + "...");

while(true){ // forever, always waiting for new clients

socket = listener.accept();

ClientHandler newClient = new ClientHandler(socket, loggedOn, accountRecords, bookRecords, key);

newClient.start(); // start the thread

System.out.println("Just connected to " + socket.getRemoteSocketAddress());

}

}

/\*\*

\* returns bookID (int)

\* @return bookID (int)

\*/

public static int getBookID() {

return bookID;

}

/\*\*

\* incrementing BookID

\*/

public static void incrementBookID() {

bookID++;

}

/\*\*

\* returns account File

\* @return accountsFile

\*/

public static File getAccountsFile() {

return accountsFile;

}

/\*\*

\* returns book File

\* @return booksFile

\*/

public static File getBooksFile() {

return booksFile;

}

}

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

class ClientHandler extends Thread{ // (extends Thread) to use run() method

// the following variable will be initialized by the constructor (value and object)

private ArrayList<String> loggedOn;

private ArrayList<String> accountRecords;

private ArrayList<String> bookRecords;

private Socket clientSocket;

private DataInputStream dis;

private DataOutputStream out;

private Object key;

/\*\*

\* The constructor will initialize the following variable (values and objects from Server class)

\* @param socket

\* @param loggedOn

\* @param accountRecords

\* @param bookRecords

\* @param key

\* @throws IOException

\*/

public ClientHandler(Socket socket, ArrayList<String> loggedOn

, ArrayList<String> accountRecords

, ArrayList<String> bookRecords

, Object key) throws IOException{

clientSocket = socket;

dis=new DataInputStream(clientSocket.getInputStream());

out = new DataOutputStream(clientSocket.getOutputStream());

this.loggedOn = loggedOn;

this.accountRecords = accountRecords;

this.bookRecords = bookRecords;

this.key = key;

}

@Override

public void run() {

String[] token; // to handle the command from the Client object

String command = "";

String currentUser = ""; // storing the username of this running user

boolean isLogin = false; // used in the condition of the second while loop

boolean isLogout = false; // used in the condition of the first and second while loop

String message = "----- Welcome to Student Book Sharing System ------"; // telling the user that he/she connected to server

while(!isLogout) // if not logged out yet

{

try { // for IOException:

out.writeUTF(message); // sending message to the client

command = dis.readUTF(); // for receiving message from server.

}catch(IOException e){}

if(!command.endsWith("#")){ // not end with '#' character

message = " 23; Invalid format#";

continue;

}

command = command.substring(0,command.length() - 1); // remove '#'

token = command.split("[;]"); // so dealing the command's part will be easier for "switch" part

switch(token[0])

{

case "01":

if(token.length == 3) // three parts

{

System.out.println("Client "+clientSocket.getRemoteSocketAddress()+" Waiting for key..."); // for monitoring purposes...

synchronized(key){ // no other threads can edit critical section until this thread finish

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" holding the key..."); // for monitoring purposes...

message = CreateAcc(token[1], token[2]); //To create account on accountRecords array and update the Account File

}

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" left the key"); // for monitoring purposes...

}

else

{

message = " 23; Invalid format#";

}

break;

case "02":

if(token.length == 3) // three parts

{

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" Waiting for key..."); // for monitoring purposes...

synchronized(key){ // no other threads can edit critical section until this thread finish

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" holding the key..."); // for monitoring purposes...

message = logOn(token[1], token[2]); /\*

To log the client to the server

(if he/she entered username/password correctly and it is stored in the system)

\*/

}

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" left the key"); // for monitoring purposes...

if(message.equals(" 20;OK#")){

currentUser = token[1];

isLogin = true;

}

}

else

{

message = " 23; Invalid format#";

}

break;

case "03":

case "04":

case "05":

case "06":

case "07": message = " 18;You should login first#";

break;

default:

message = " 29;Invalid message#";

}

while(!isLogout && isLogin) // if not logged out yet and logged in

{

try { // for IOException:

out.writeUTF(message); // for sending message to server.

command = dis.readUTF(); // for receiving message from server.

}catch(IOException e){}

if(!command.endsWith("#")){ // not end with '#' character

message = " 23; Invalid format#";

continue;

}

command = command.substring(0,command.length() - 1); // remove '#'

token = command.split("[;]"); // so dealing the command's part will be easier in "switch" part

switch(token[0])

{

case "01":

case "02":

message = " 11;Not possible on login mode#";;

break;

case "03":

if(token.length == 2 && token[1].contains(currentUser + ",")) // two parts + is he/she the current user or not

{

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" Waiting for key..."); // for monitoring purposes...

synchronized(key){ // no other threads can edit critical section until this thread finish

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" holding the key..."); // for monitoring purposes...

token = token[1].split(","); // get book name and username

message = uploadBookInformation(token[0], token[1]);

}

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" left the key"); // for monitoring purposes...

}

else

{

message = " 23; Invalid format#";

}

break;

case "04":

if(token.length == 2 && token[1].equals("LISTBOOKS")) // two parts + is "LISTBOOKS" wrote correctly?

{

message = bookList(); //String book's list from bookRecords array

}

else

{

message = " 23; Invalid format#";

}

break;

case "05":

if(token.length == 3 && token[1].equals(currentUser)) // three parts + is he/she the current user or not

{

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" Waiting for key..."); // for monitoring purposes...

synchronized(key){ // no other threads can edit critical section until this thread finish

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" holding the key..."); // for monitoring purposes...

message = generateBookRequest(token[2]);

}

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" left the key"); // for monitoring purposes...

}

else

{

message = " 23; Invalid format#";

}

break;

case "06":

if(token.length == 3)

{ // three parts

if(token[1].equals(currentUser) && accountRecords.contains(token[1] + "," + token[2]))

{ // Is he/she the current user or not + given username,password correct?

message = " 26;"+token[1]+" is logged off successfully#";

loggedOn.remove(token[1]);

isLogout = true;

}

else

{

message = " 21;Invalid user (or bad password)#";

}

}

else

{

message = " 23; Invalid format#";

}

break;

case "07":

if(token.length == 3)

{ // three parts

if(token[1].equals(currentUser) && accountRecords.contains(token[1] + "," + token[2]))

{ // Is he/she the current user or not + given username,password correct?

message = " 27;"+token[1]+"#";

accountRecords.remove(token[1] + "," + token[2]);

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" Waiting for key..."); // for monitoring purposes...

synchronized(key){ // no other threads can edit critical section until this thread finish

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" holding the key..."); // for monitoring purposes...

updateAccountFile(); // updating book file (from accountRecords array to file)

}

System.out.println("Client "+clientSocket.getRemoteSocketAddress() +" left the key"); // for monitoring purposes...

loggedOn.remove(token[1]);

isLogout = true;

}

else

{

message = " 21;Invalid user (or bad password)#";

}

}

else

{

message = " 23; Invalid format#";

}

break;

default:

message = " 29;Invalid message#";

}

}

}

try {

System.out.println("---Client "+clientSocket.getRemoteSocketAddress() +" Exited from system---"); // for monitoring purposes...

out.writeUTF(message); // message to client

clientSocket.close();

}catch(IOException e){}

}

/\*\*

\* To create account on accountRecords array and update the Account File

\* @param username String

\* @param password String

\* @return String message (succeeded or failed)

\*/

public String CreateAcc(String username, String password)

{

if(!accountRecords.contains(username + "," + password)) // if the account not exist in the system

{

accountRecords.add(username+","+password);

updateAccountFile(); // updating book file (from accountRecords array to file)

return " 20;OK#";

}

else

{

return " 19;Account already exists in system#";

}

}

/\*\*

\* To log the client to the server (if he/she entered username/password correctly and it is stored in the system)

\* @param username String

\* @param password String

\* @return String message (succeeded or failed)

\*/

public String logOn(String username, String password)

{

if(accountRecords.contains(username + "," + password)) // if the account exist in the system

{

if(!loggedOn.contains(username)) // if the account not logged in

{

loggedOn.add(username);

return " 20;OK#";

}

else

{

return " 17;error (someone logged in by this username)#";

}

}

else

{

return " 21;Invalid user (or bad password)#";

}

}

/\*\*

\* allows the client to upload Book Information on the server and store in bookRecords array and book file

\* @param username String

\* @param bookname String

\* @return String message

\*/

public String uploadBookInformation(String username, String bookname)

{

bookRecords.add(Server.getBookID()+","+bookname+","+username+",Available");

updateBookFile(); // updating book file (from bookRecords array to file)

Server.incrementBookID(); // incrementing BookID

return " 22;Information is uploaded successfully on the system #";

}

/\*\*

\* returns String book's list from bookRecords array

\* @return String book's list from bookRecords array

\*/

public String bookList(){

String bookList = "";

for (int i = 0; i < bookRecords.size(); i++) {

bookList += " 24;" + bookRecords.get(i) + "#\n";

}

if(bookRecords.size() == 0){ // if there is no book

bookList = " 24;#";

}

return bookList;

}

/\*\*

\* generate Book Request: it will change the state of the targeted book to 'Booked' in bookRecords array and file (if the process succeeded)

\* @param bookID String

\* @return String message (succeeded or failed)

\*/

public String generateBookRequest(String bookID){

for (int i = 0; i < bookRecords.size(); i++) {

if (bookRecords.get(i).startsWith(bookID + ",")) {

if (bookRecords.get(i).endsWith("Available")) {

String temp = bookRecords.get(i);

temp = temp.substring(0, temp.length() - 9);

temp = temp + "Booked";

bookRecords.set(i, temp);

updateBookFile(); // updating book file (from bookRecords array to file)

return " 25;Book has been reserved successfully #";

}else {

return " 26; This book is not available for reservation#";

}

}

}

return " 16; This book is not available#";

}

/\*\*

\* updating book file (from bookRecords array to file)

\*/

public void updateBookFile() {

try {PrintWriter fileWrite = new PrintWriter(Server.getBooksFile()); // to delete content of the file, then write

for (int j = 0; j < bookRecords.size(); j++) {

fileWrite.println(bookRecords.get(j));

}

fileWrite.flush();

fileWrite.close();

} catch(IOException e){};

}

/\*\*

\* updating book file (from accountRecords array to file)

\*/

public void updateAccountFile() {

try {PrintWriter fileWrite = new PrintWriter(Server.getAccountsFile()); // to delete content of the file, then write

for (int i = 0; i < accountRecords.size(); i++) {

fileWrite.println(accountRecords.get(i));

}

fileWrite.flush();

fileWrite.close();

} catch(IOException e){};

}

}

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ConnectException;

import java.net.Socket;

import java.util.Scanner;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\* Main class for the client side

\* @author omar-

\*/

public class Client {

/\*\*

\* Main method for the client side

\* @param args the command line arguments

\* @throws java.io.IOException

\*/

public static void main(String[] args) throws IOException {

Scanner input = new Scanner(System.in);

try{ // handling the exceptions:

Socket socket = new Socket("127.0.0.1", 9090);

DataOutputStream dout=new DataOutputStream(socket.getOutputStream()); // for sending message to server.

DataInputStream in = new DataInputStream(socket.getInputStream()); // for receiving message from server

String commingGoingMsg = "";

while(true)

{

commingGoingMsg =in.readUTF(); // for receiving message from server.

System.out.println(commingGoingMsg); // printing the received message from server.

if(commingGoingMsg.contains("off successfully#")

|| commingGoingMsg.startsWith(" 27;")) // means if logout successfully...

{

break;

}

/\*

commingGoingMsg: taking input from user and delete the...

extra space in the beginnig and in the end of the file

\*/

commingGoingMsg = input.nextLine().trim();

dout.writeUTF(commingGoingMsg); // for sending message to server.

dout.flush(); // make sure that message is sent

}

socket.close(); // when finish

} catch(ConnectException e){ // handling ConnectException:

System.out.println(" Cannot connect to server, please try again later...");

} catch(Exception e){ // handling Unknown exception:

System.out.println(" (Something happend in client side. Please try again...)");

}

}

}