**西安邮电大学**

**网络空间安全学院**

**实验报告**

**（2019/ 2020学年第1学期）**

实验名称： 信息安全专业课程设计A

学生姓名： XXXXXX c

专 业： 安全XXXX班

学 号： 8位学号（班内序号）

指导教师： XXX

**西安邮电大学网络空间安全学院XXXX（实验课程名）**

**考核表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学生姓名** |  | **班级/学号** | | **/** | |
| **承担任务实验室（单位）** | **信息安全与信息对抗实验教学中心** | **所在部门** | | **网安院XXX** | |
| **实施时间** | **年 月 日 — 年 月 日** | | | | |
| **实验内容** |  | | | | |
| **指导教师（师傅）姓名** |  | **职务或职称** | |  | |
| **指导教师（师傅）**  **对学生的评价** | **评价点** | **评价内容** | | | **评价得分** |
| **平时成绩**  **（20分）** | **课堂出勤、课堂纪律及学习态度**  **遵守实验室的规章制度等情况** | | |  |
| **实验能力**  **（30分）** | **文件检索或软硬件设计能力**  **分析、解决实际问题能力** | | |  |
| **实验验收**  **（20分）** | **实验最终完成质量情况**  **实验演示与答辩情况** | | |  |
| **实验报告**  **（30分）** | **语言及文字表达能力**  **实验报告结构及书写格式规范** | | |  |
| **总评成绩** |  | | **五级制** |  | |
| **指导教师（师傅）评语** | **指导教师（师傅）签字**  **年 月 日** | | | | |

1. **实验目的和要求**

**（实验报告正文可根据具体实验适当调整标题与格式。要求每次实验结束后需提交一次实验报告，全部实验结束后各次实验报告统一装订，附考核表一份，手写还是打印由课程教师确定）**

**正文要求小四号宋体，1.5倍行距。**

页面设置：上－2.5厘米，下－2厘米，左－2.5厘米，右－2厘米，装订线－0厘米，页眉－2厘米，页角－2厘米。

一级标题（章）：用1、2、3、4、5、6……的序号，小三号的加黑的黑体字，居中。一级标题（章）之间另起一页。

二级标题（节）：用1.1、1.2、1,3、1.4……的序号，四号的加黑的黑体字，居中。和一级标题之间空一行。

三级标题：用1.1.1、1.1.2、1.1.3、1.1.4……的序号，小四号的加黑的黑体字，靠左。和二级标题之间空一行。

四级标题：用1.1.1.1、1.1.1.2、1.1.1.3、1.1.1.4……的序号，小四号的宋体字，靠左。

作为章节和段落的序号划分至多到四级。

页码：用阿拉伯数字连续编页，用小五号Times New Roman字号，居中，数字两侧不加任何修饰。

**插图：**插图包括图解、示意图、构造图、曲线图、框图、流程图、布置图、地图、照片、图版等。插图注明项有图号、图题、图例。图号编码用章序号。如“图2.1”表示第2章第1图。图号与图题文字留一字空格置于无色文本框内，放在图的正下方居中，最好和图组合在一起，图题用5号或小5号字，字体用宋体。图中标注符号文字字号不大于图题的字号。

为确保打印足够清晰的图，还需注意以下几点：若是扫描图，则要保证有大于300dpi的分辨率；若是编程软件绘制的图，则应先画出足够大的图，再通过拷贝屏幕的方式，其它根据具体情况而定。

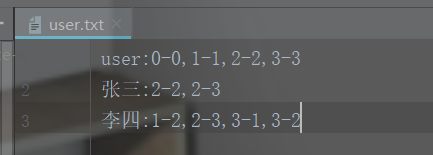
**表：**表的一般格式是数据依序竖排，内容和项目由左至右横读，通版排版。表号也用章序号编号码，如：表2.1是第2章中的第1表。表应有表题，与表号之间空1字，置于无色文本框内，放在表的上方居中，用5号或小5号宋体。表中的内容和项目字号不大于图题的字号。

**公式：**公式包括数学、物理和信息学等。正文中引用的公式、算式或方程式等可以按章序号用阿拉伯数字编号（式号），如：式（2.1）表示第2章第1式，公式一般单行居中排版与上下文分开，式号与公式同行，居右排版。

1. **实验原理**

**2.1 通信：通过socket实现服务器与客户端间远程通信，并通过多线程使服务端可以同时服务多个客户端**

**2.2 用户验证：其中包括账号验证与密码验证；将所有用户信息（账号，密码）都存储到文件中，当有客户端请求登录时，将登录信息与文件中的信息进行一一比对，并返回客户端相应状态码**



**2.3 图形校验实现：这里没有使用jui技术，而是简单的用二维数组来存储；二维数组是boolean类型，将所有正确的点标记为true即可，同样的用户文件中也只是存储为true的点的位置；**

**因此，在客户端与服务端通信时，约定传递的格式是字符串 x-y形式的字符串**

1. **实验内容及步骤**

**3.1. Server**

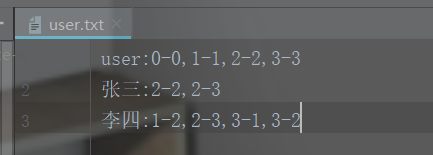
**ServerMain：负责启动服务端**

1. **import** java.io.IOException;
3. **public** **class** ServerMain {
5. **public** **static** **void** main(String[] args) {
6. **try** {
7. **new** Server().start();
8. } **catch** (IOException e) {
9. e.printStackTrace();
10. }
11. }
12. }

**ServerThread：服务端线程，负责处理用户请求，这里主要做登录验证**

1. **读取用户名**
2. **读取用户发来的验证码：在服务端维护一个二维数组，解析客户端发来的数据（x-y格式的字符串）成x，y索引，然后修改服务端的二维数组**
3. **发送变化后的数组信息：首先遍历二维数组找到为true的位置，然后将这些索引（x，y）封装成x1-y1,x2-y2,x3-y3的字符串，最后将字符串发送给客户端**
4. **登录验证：当用户输入verify时开始进行登录验证；首先，读取user.txt中的用户信息；**

**然后，解析用户信息username:x-y,x1-y1 成用户名与索引信息；最后分别判断用户与密码是否正确，并返回不同的状态码**



1. **import** java.io.\*;
2. **import** java.net.Socket;
3. **import** java.util.Arrays;
4. **import** java.util.logging.Logger;
6. **public** **class** ServerThread **extends** Thread {
8. **private** **static** String name = ServerThread.**class**.getName();
9. **private** **static** Logger log = Logger.getLogger(name);
11. **private** **static** **final** Integer ERROR\_USER = -1;
12. **private** **static** **final** Integer ERROR\_PWD = 0;
13. **private** **static** **final** Integer SUCCESS = 1;
15. **private** Socket conn;
17. **public** ServerThread(Socket conn) {
18. **this**.conn = conn;
19. log.info(conn.getInetAddress().getHostAddress() + "尝试连接ing。。。");
20. }
22. @Override
23. **public** **void** run() {
24. **try** {
25. write(read());
26. } **catch** (IOException e) {
27. e.printStackTrace();
28. } **finally** {
29. **try** {
30. conn.close();
31. } **catch** (IOException e) {
32. e.printStackTrace();
33. }
34. }
35. }
37. **public** **int** read() **throws** IOException {
38. InputStream inputStream = conn.getInputStream();
39. BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(inputStream));
40. OutputStream outputStream = conn.getOutputStream();
41. PrintWriter printWriter = **new** PrintWriter(outputStream);
43. String username = reader.readLine();
45. **boolean**[][] chart = **new** **boolean**[4][4];
47. String position;
48. **while** (!(position = reader.readLine()).equalsIgnoreCase("verify")) {
49. String[] split = position.split("-");
50. **int** x = Integer.valueOf(split[0]), y = Integer.valueOf(split[1]);
51. System.out.println("x:" + x + " y:" + y);
52. chart[x][y] = chart[x][y] ? **false** : **true**;
53. System.out.println("char[x][y]" + chart[x][y]);
54. String data = "";
55. **for** (**int** i = 0; i < 4; i++) {
56. **for** (**int** j = 0; j < 4; j++) {
57. **if** (chart[i][j]) {
58. String s = i + "-" + j;
59. data = data + s + ",";
60. }
61. }
62. }
63. **if** (! data.equals(""))
64. data = data.substring(0,data.length() - 1);
65. printWriter.println(data);
66. printWriter.flush();
67. }
69. **return** check(username, chart);
70. }
72. **public** **int** check(String username, **boolean** [][] chart) **throws** IOException {
73. **boolean**[][] rightChart = parseUserFile(username);
74. **if** (rightChart == **null**)
75. **return** ERROR\_USER;
76. **for** (**int** i = 0; i < rightChart.length; i++) {
77. **if** (! Arrays.equals(rightChart[i],chart[i]))
78. **return** ERROR\_PWD;
79. }
80. **return** SUCCESS;
81. }
83. **public** **boolean**[][] parseUserFile(String username) **throws** IOException {
84. File file = **new** File("server07/src/user.txt");
85. FileReader fileReader = **new** FileReader(file);
86. BufferedReader reader = **new** BufferedReader(fileReader);
88. **boolean** findUser = **false**;
89. **boolean**[][] rightChart = **new** **boolean**[4][4];
90. String line;
91. **while** ((line = reader.readLine()) != **null**) {
92. String[] split = line.split(":");
93. **if** (! split[0].equals(username))
94. **continue**;
95. findUser = **true**;
96. String[] poses = split[1].split(",");
97. **for** (**int** i = 0; i < poses.length; i++) {
98. String[] pos = poses[i].split("-");
99. **int** x = Integer.valueOf(pos[0]);
100. **int** y = Integer.valueOf(pos[1]);
101. rightChart[x][y] = **true**;
102. }
103. }
105. **return** findUser ? rightChart : **null**;
106. }
108. **public** **void** write(**int** checkCode) **throws** IOException{
109. OutputStream outputStream = conn.getOutputStream();
110. PrintWriter printWriter = **new** PrintWriter(outputStream);
112. **if** (checkCode == SUCCESS) {
113. log.info(conn.getInetAddress().getHostAddress() + " 登陆成功");
114. printWriter.println(checkCode);
115. printWriter.flush();
116. } **else** {
117. log.info(conn.getInetAddress().getHostAddress() + " 登录失败");
118. printWriter.println(checkCode);
119. printWriter.flush();
120. }
122. printWriter.close();
123. outputStream.close();
124. }
125. }

**Server：根据客户端个数启动ServerThread**

1. **import** java.io.IOException;
2. **import** java.net.ServerSocket;
3. **import** java.net.Socket;
4. **import** java.util.logging.Logger;
6. **public** **class** Server {
8. **private** **static** String name = Server.**class**.getName();
9. **private** **static** Logger log = Logger.getLogger(name);

12. **public** **void** start() **throws** IOException {
13. ServerSocket server = **new** ServerSocket(8080);
14. log.info("服务器在8080端口启动");
16. **while** (**true**) {
17. Socket conn = server.accept();
18. **new** ServerThread(conn).start();
19. }
20. }
21. }

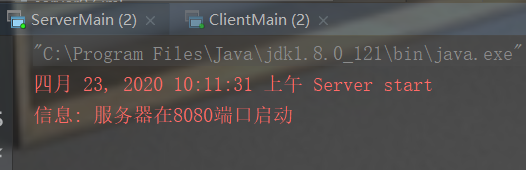
**3.2 Client**

**ClientMain：启动客户端**

1. **import** java.io.IOException;
3. **public** **class** ClientMain {
5. **public** **static** **void** main(String[] args) {
6. **try** {
7. **new** Client().start();
8. } **catch** (IOException | ClassNotFoundException e) {
9. e.printStackTrace();
10. }
11. }
12. }

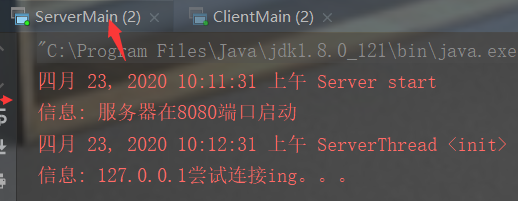
**Client**

1. **读取用户输入（ip，端口，用户名）**
2. **向服务端发送请求，即登陆信息**
3. **依次读取用户输入（set（x，y）），同时进行输入校验，若格式错误或索引越界则重新输入；解析输入并封装成x-y形式的字符串，然后发送给服务端**
4. **等verify后接收服务端发送来的状态码，解析后打印不同登录信息**
5. **import** java.io.\*;
6. **import** java.net.Socket;
7. **import** java.util.HashMap;
8. **import** java.util.Map;
9. **import** java.util.Scanner;
10. **import** java.util.logging.Logger;
12. **public** **class** Client {
14. **private** Socket client = **null**;
16. **private** **static** String name = Client.**class**.getName();
17. **private** **static** Logger log = Logger.getLogger(name);
19. **private** **static** **final** Integer ERROR\_USER = -1;
20. **private** **static** **final** Integer ERROR\_PWD = 0;
21. **private** **static** **final** Integer SUCCESS = 1;
23. **public** **void** start() **throws** IOException, ClassNotFoundException {
24. Map<String, Object> input = input();
25. client = **new** Socket((String)input.get("ip"), (Integer)input.get("port"));
26. login((String)input.get("username"));
27. client.close();
28. }
30. **public** Map<String,Object> input() {
31. HashMap<String, Object> map = **new** HashMap<>();
33. Scanner in = **new** Scanner(System.in);
34. System.out.print("服务器IP：");
35. map.put("ip",in.next());
36. System.out.print("服务器端口：");
37. map.put("port",in.nextInt());
38. System.out.print("用户名：");
39. map.put("username",in.next());
41. **return** map;
42. }
44. **public** **void** login(String username) **throws** IOException{
45. OutputStream outputStream = client.getOutputStream();
46. PrintWriter printWriter = **new** PrintWriter(outputStream);
47. InputStream inputStream = client.getInputStream();
48. BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(inputStream));
50. printWriter.println(username);
52. printWriter.flush();
53. printChart("");
55. Scanner in = **new** Scanner(System.in);
56. **while** (**true**) {
57. System.out.print("输入(verify确定)：");
58. String input = in.nextLine();
59. **if** ( input.equals("verify")) {
60. printWriter.println(input);
61. printWriter.flush();
62. **break**;
63. }
64. **if** ((input = createMsg(input)) == "") {
65. **continue**;
66. }
67. printWriter.println(input);
68. printWriter.flush();
70. String str = reader.readLine();
71. printChart(str);
72. }
74. **int** loginCode;
75. **if** ((loginCode = Integer.valueOf(reader.readLine())) == SUCCESS) {
76. log.info("登录成功！！！");
77. } **else** **if** (loginCode == ERROR\_PWD){
78. log.info("登录失败，密码错误");
79. } **else** {
80. log.info("登录失败，无该用户");
81. }
83. printWriter.close();
84. outputStream.close();
85. reader.close();
86. inputStream.close();
87. }
89. **public** **void** printChart(String pos) {
90. **boolean**[][] chart = **new** **boolean**[4][4];
91. **if** (pos != **null** && ! pos.isEmpty()) {
92. String[] split = pos.split(",");
93. **for** (**int** i = 0; i < split.length; i++) {
94. String[] split1 = split[i].split("-");
95. **int** x = Integer.valueOf(split1[0]);
96. **int** y = Integer.valueOf(split1[1]);
97. chart[x][y] = **true**;
98. }
99. }
101. **for** (**int** i = 0; i < 4; i++) {
102. **for** (**int** j = 0; j < 4; j++) {
103. **if** (chart[i][j]) {
104. System.out.print(" x ");
105. } **else** {
106. System.out.print(" \* ");
107. }
108. }
109. System.out.println();
110. }
111. }
113. **public** String createMsg(String input) {
114. **try** {
115. input = input.substring(input.indexOf("(") + 1,input.indexOf(")"));
116. String[] split = input.split(",");
117. **int** x = Integer.valueOf(split[0]) - 1;
118. **int** y = Integer.valueOf(split[1]) - 1;
119. **if** (x > 3 || x < 0  || y > 3 || y < 0) {
120. **throw** **new** Exception();
121. }
122. **return** x + "-" + y;
123. } **catch** (Exception e) {
124. System.out.println("输入错误（格式：set(x,y)）！请重新输入");
125. **return** "";
126. }
127. }
128. }
129. **实验结果分析**
130. **启动服务端**

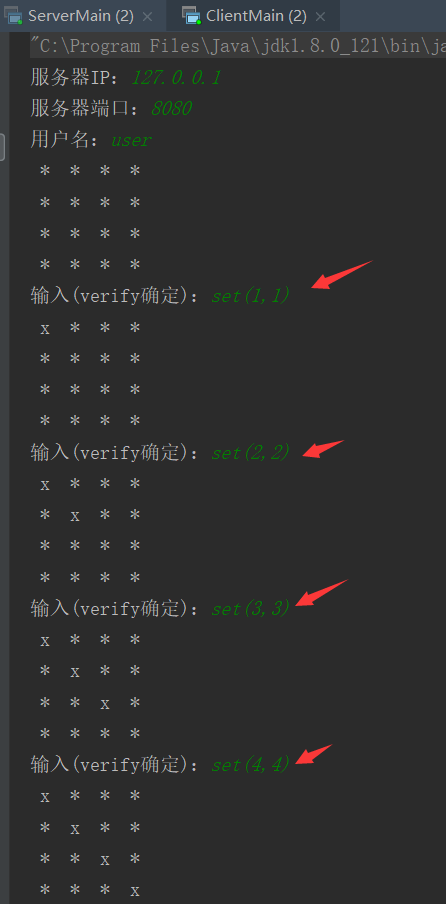


1. **启动客户端，并输入ip端口等信息，观察客户端与服务端**

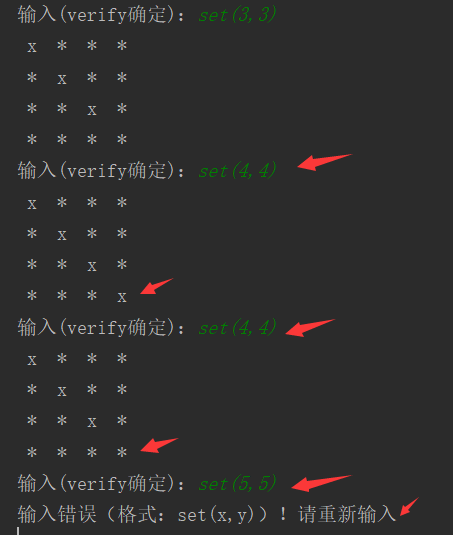




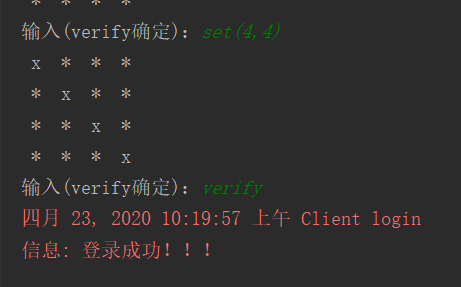
1. **客户端输入依次输入图形密码**

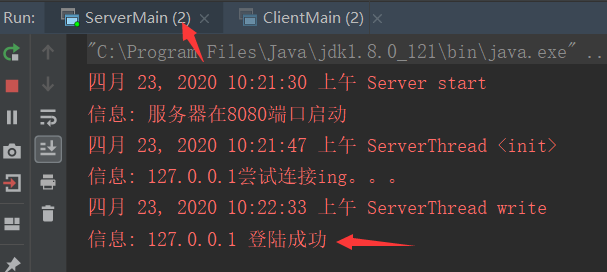


**演示输入错误，与回退**

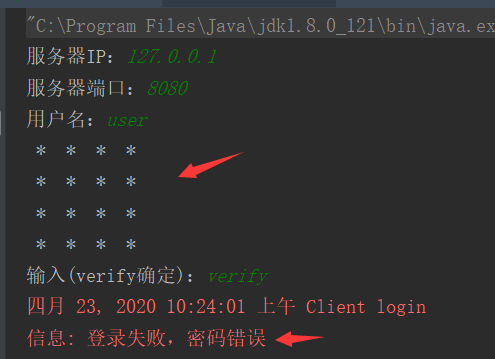


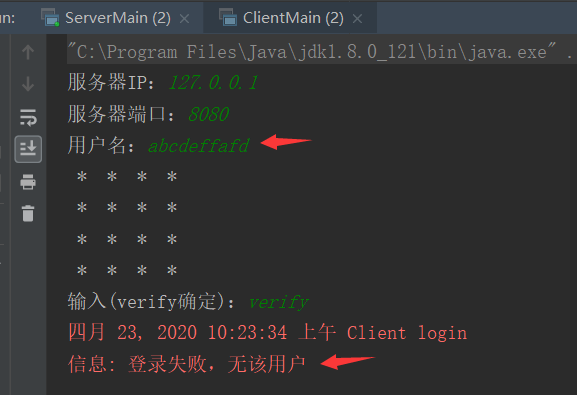
1. **verify进行登录校验，本次输入的username与密码正确**





1. **演示两种错误情况，账号错误与密码错误**







**五．实验总结及心得体会**