ㅣ

SEO JIHO (서지호)

# 과제: [HW1-nw22-socket]

* 해당 프로그램에 주석 (comment)를 한글로 단다. (추가할 수 있는 것은 최대한 추가)

# #개요

|  |  |
| --- | --- |
|  | [자바 응용프로그램을 통한 멀티 스레드 서버, 클라이언트 설계 및 프로토콜 설계] |

**Requirements**

* < 요구조건 #1: 서버의 멀티 스레드 지원>
* < 요구조건 #2: 단순 메시지 전송이 아닌 설계된 프로토콜을 통한 통신 지원>
* < 요구조건 #3: 정확한 사칙연산 수행>
* < 요구조건 #4: 아스키 코드 기반의 메시지>
* < 요구조건 #5: 서버 접속 정보가 담긴 파일을 읽고 서버에 접속하는 FILEIO 기능 구현>

# REPORT

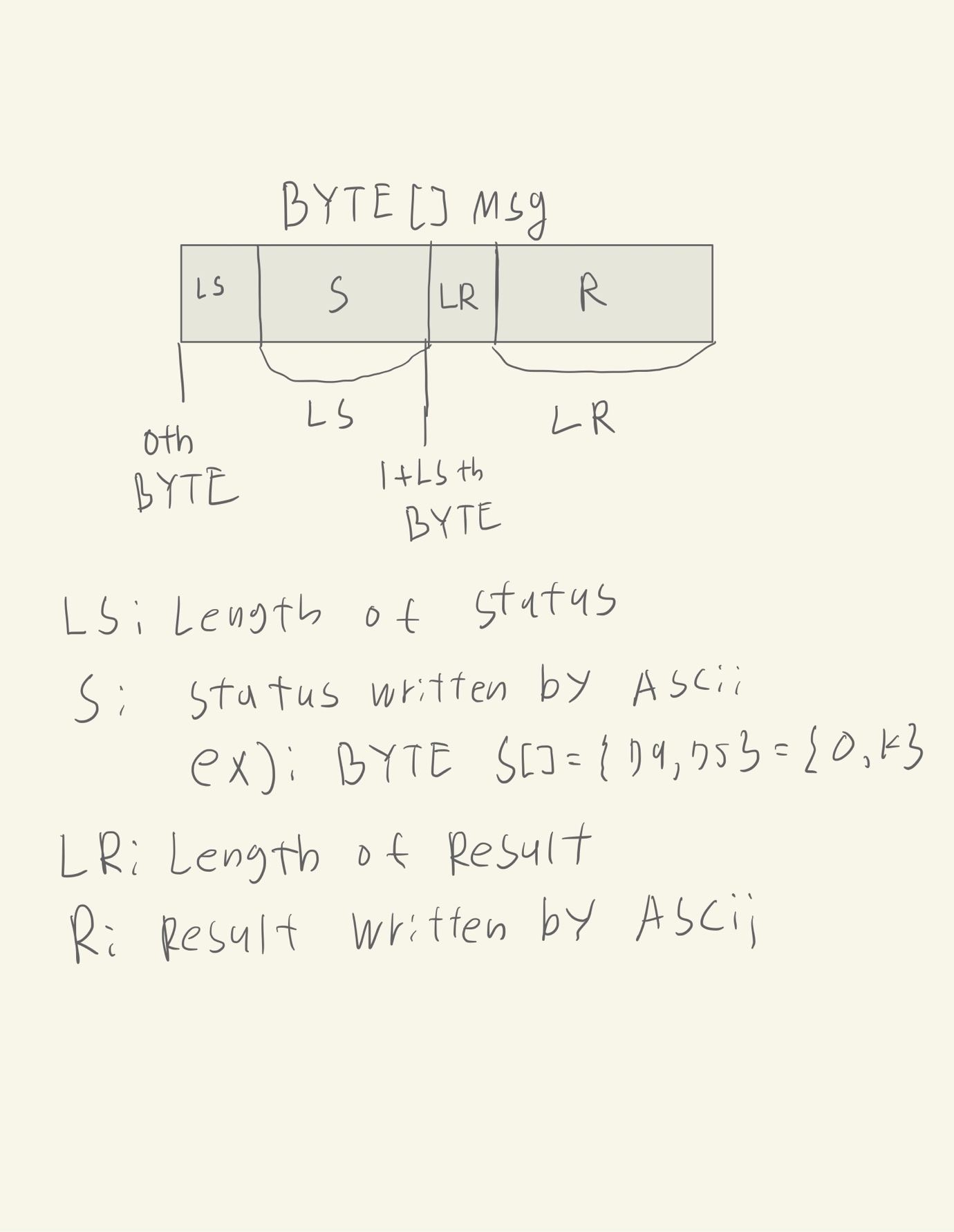
* 과제의 기본 요구 사항을 이곳에 원하는 만큼 작성 (e.g. 구조도, protocol 내용 상세 설명 등)
* 소스코드는 마지막에 제공하는 양식을 이용 하여 제출 (zip파일로도 제출)

1. Blueprint of Protocol

Type: Ascii base, byte Communication

Length: Dynamic(Depends on result’s length)

Structure and Components of Procotol:



Procedure of Client to handle this Message:

1. Read 1 byte from Input Stream so that check how long byte needs to be read
2. Read byte as much as value we got from procedure 1 and save bytes to new byte array named status
3. Read 1 byte from Input Stream so that check how long byte needs to be read(for length of result)
4. Read byte as much as value we got from procedure 3 and save bytes to new byte array named result
5. Appropriately Handle two array we’ve got from above procedure.

Code That reads Message from InputStream:

public static void recvMsg(InputStream is){

byte[] data = new byte[256]; //서버로 부터 받은 byte 배열 공간 생성

try{

is.read(data); //서버로 부터 받은 byte 배열을 data에 저장

}

catch(Exception e){

System.out.println(e);

}

byte sizeofcode=data[0]; //sizeofcode에 data의 첫번째 byte 저장

byte sizeofresult=data[sizeofcode+1]; //sizeofresult에 data[sizeofcode+1]에 있는 결과값의 길이 저장

String code=new String(data,1,sizeofcode); //code에 data의 1번째 byte부터 sizeofcode만큼의 byte를 String으로 변환하여 저장

System.out.println("Status: "+code);

if(sizeofresult!=0){

String result=new String(data,sizeofcode+2,sizeofresult); //result에 data의 sizeofcode+2번째 byte부터 sizeofresult만큼의 byte를 String으로 변환하여 저장

System.out.println("Result: "+result);

}

}

## SOURCE CODES

1. Client Source Code:

|  |
| --- |
| package HW1;  import java.io.\*;  import java.net.\*;  public class client {  public static void recvMsg(InputStream is){  byte[] data = new byte[256]; //서버로 부터 받은 byte 배열 공간 생성  try{  is.read(data); //서버로 부터 받은 byte 배열을 data에 저장  }  catch(Exception e){  System.out.println(e);  }  byte sizeofcode=data[0]; //sizeofcode에 data의 첫번째 byte 저장  byte sizeofresult=data[sizeofcode+1]; //sizeofresult에 data[sizeofcode+1]에 있는 결과값의 길이 저장  String code=new String(data,1,sizeofcode); //code에 data의 1번째 byte부터 sizeofcode만큼의 byte를 String으로 변환하여 저장  System.out.println("Status: "+code);  if(sizeofresult!=0){  String result=new String(data,sizeofcode+2,sizeofresult); //result에 data의 sizeofcode+2번째 byte부터 sizeofresult만큼의 byte를 String으로 변환하여 저장  System.out.println("Result: "+result);  }    }  public static void main(String[] args){  String ip="localhost"; // 기본 서버 주소와 포트를 하드코드방식으로 지정  String port="8080";  InputStream is=null; // 서버로부터 받은 메시지를 읽어들이기 위한 InputStream  BufferedReader stin = null; // 키보드로부터 입력받기 위한 BufferedReader  BufferedWriter toserver = null; // 서버로 메시지를 보내기 위한 BufferedWriter  Socket socket = null; // 서버와 통신하기 위한 소켓  try{  File file= new File("HW1/server\_info.dat"); // 서버 주소와 포트를 저장한 파일을 읽어들이기 위한 File 객체  FileReader filereader = new FileReader(file);  BufferedReader bufReader = new BufferedReader(filereader);  String line="";  line=bufReader.readLine();  ip=line;  line=bufReader.readLine();  port=line;  bufReader.close();  }  catch(FileNotFoundException e){ // 파일이 없을 경우  System.out.println("no file found, Run Program as default ip and port");  }  catch(Exception e){ //그 외의 에러 발생시  System.out.println(e);  }  try {  socket = new Socket(ip, Integer.parseInt(port)); // 서버와 통신을 위한 소켓 생성  is=socket.getInputStream(); // 서버로부터 받은 메시지를 읽어들이기 위한 InputStream  stin = new BufferedReader(new InputStreamReader(System.in)); // 키보드로부터 입력받기 위한 BufferedReader  toserver = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())); // 서버로 메시지를 보내기 위한 BufferedWriter  String outputMessage; // 서버로 보낼 메시지를 저장할 변수  while (true) {  outputMessage = stin.readLine(); // 키보드로부터 입력받은 메시지를 outputMessage에 저장  if (outputMessage.equalsIgnoreCase("bye")) { // bye를 입력받으면  toserver.write(outputMessage); //서버에게 bye를 보냄  toserver.flush(); // 버퍼를 비움  recvMsg(is); // 서버로부터 받은 메시지를 출력  break; //소켓 연결 종료  }  toserver.write(outputMessage); // bye가 아닌 다른 메시지를 입력받으면 서버로 메시지를 보냄  toserver.flush(); // 버퍼를 비움  recvMsg(is); // 서버로부터 받은 결과를 recvMsg 함수를 통해 출력      }  } catch (IOException e) {  System.out.println(e.getMessage());  } finally {  try {  socket.close();  } catch (IOException e) {  System.out.println("비정상적 종료");  }  }  }  } |

1. Server Source Code
2. package HW1;
3. import java.io.\*;
4. import java.net.ServerSocket;
5. import java.net.Socket;
6. import java.text.SimpleDateFormat;
7. import java.util.\*;
8. class MsgSend{ //메시지를 보내는 클래스 , 2개의 메소드를 포함
9. //1. sendMsg : 정상적인 메시지를 보내는 메소드
10. public int sendMsg(OutputStream os,String status,String result){
11. try{
12. byte[] code = status.getBytes();
13. byte[] data = result.getBytes();
14. int sizeofdata=data.length;
15. int sizeofcode=code.length;
16. byte[] packet= new byte[sizeofdata+sizeofcode+2];
17. packet[0]=(byte)sizeofcode;
18. for(int i=1; i<sizeofcode+1; i++){
19. packet[i]=code[i-1];
20. }
21. packet[sizeofcode+1]=(byte)sizeofdata;
22. for(int i=sizeofcode+2; i<sizeofdata+sizeofcode+2; i++){
23. packet[i]=data[i-sizeofcode-2];
24. }
26. os.write(packet);
27. os.flush();
29. }
30. catch(Exception e){
31. System.out.println(e);
32. }
33. return 0;
34. }
35. //2. sendMsg : 결과값을 포함하지 않는 비정상적 메세지를 보내는 메소드
36. public int sendMsg(OutputStream os,String status){
37. try{
38. byte[] code = status.getBytes();
40. int sizeofdata=0;
41. int sizeofcode=code.length;
42. byte[] packet= new byte[sizeofdata+sizeofcode+2];
43. packet[0]=(byte)sizeofcode;
44. for(int i=1; i<sizeofcode+1; i++){
45. packet[i]=code[i-1];
46. }
47. packet[sizeofcode+1]=(byte)sizeofdata;
48. os.write(packet);
49. os.flush();
51. }
52. catch(Exception e){
53. System.out.println(e);
54. }
55. return 0;
56. }
57. }
58. class ClientThread extends Thread
59. {
60. Socket socket;
61. int id;
63. //생성자를 통해 입력받은 소켓과 클라이언트(쓰레드)의 id를 저장
64. ClientThread (Socket socket, int id)
65. {
66. this.socket = socket;
67. this.id = id;
68. }

71. @Override
72. public void run ()
73. {
74. try
75. {
76. //메세지를 보내주는 객체 생성
77. MsgSend msgsend = new MsgSend();
78. while (true)
79. {
81. OutputStream os=socket.getOutputStream(); //클라이언트에게 메세지를 보내기 위한 스트림
82. InputStream IS = socket.getInputStream(); //클라이언트로부터 메세지를 받기 위한 스트림
83. byte[] bt = new byte[256]; //클라이언트로부터 받은 메세지를 저장할 배열
84. int size = IS.read(bt); //클라이언트로부터 받은 메세지의 크기를 저장
86. String output = new String(bt, 0, size, "UTF-8"); //받은 메세지를 String으로 변환
87. String[] command= output.split(" "); //받은 메세지를 공백을 기준으로 나눔
89. if(command.length<2){
90. if(command[0].equals("bye")){ //bye 명령어를 받으면 종료
91. msgsend.sendMsg(os,"ENDBYCLIENT");
92. System.out.println(" Thread " + id + " is closed. ");
93. break;
94. }
95. else{ //명령어가 2개 이하인 경우 에러메세지를 보냄
96. msgsend.sendMsg(os,"WCMD");
97. continue;
98. }
99. }
100. else if(command.length>2){
101. msgsend.sendMsg(os,"TooMany");
102. continue;
103. }
104. String[] args=command[1].split(",");
105. if(args.length>2){ //계산의 인자가 2개 초과인 경우 에러메세지를 보냄
106. msgsend.sendMsg(os,"TooMany");
107. continue;
108. }
109. if(args.length<2){ //계산의 인자가 2개 미만인 경우 에러메세지를 보냄
110. msgsend.sendMsg(os,"TooFew");
111. continue;
112. }
113. double result=0; //계산 결과를 저장할 변수
114. if(command[0].toUpperCase().equals("ADD")){
115. result=Double.parseDouble(args[0])+Double.parseDouble(args[1]);
116. }
117. else if(command[0].toUpperCase().equals("MINUS")){
118. result=Double.parseDouble(args[0])-Double.parseDouble(args[1]);
119. }
120. else if(command[0].toUpperCase().equals("DIV")){
121. if(Double.compare(Double.parseDouble(args[1]),0)==0){//나누는 수가 0인 경우 에러메세지를 보냄
122. msgsend.sendMsg(os,"DIVZERO");
123. continue;
124. }
125. result=Double.parseDouble(args[0])/Double.parseDouble(args[1]);
126. }
127. else if(command[0].toUpperCase().equals("MUL")){
128. result=Double.parseDouble(args[0])\*Double.parseDouble(args[1]);
129. }
130. else{
131. msgsend.sendMsg(os,"WCMD"); //명령어가 존재하지 않는 경우 에러메세지를 보냄
132. continue;
133. }
134. String result\_string; //계산 결과를 문자열로 변환할 변수
135. if(result!=(int)result){ //결과가 정수가 아니라면
136. result\_string=Double.toString(result); //Double형으로 변환
137. }else{ //결과가 정수라면
138. result\_string=Integer.toString((int)result); //Integer형으로 변환
139. }
141. System.out.println("Thread " + id + " > result:" + result);
142. msgsend.sendMsg(os, "OK", result\_string); //OK 코드와 계산 결과를 보냄
144. }
145. } catch (IOException e)
146. {
147. System.out.println(" Thread " + id + " is closed. ");
148. }
149. }
150. }
151. class ConnectThread extends Thread
152. {
153. ServerSocket serverSocket;
154. int count = 1;
156. ConnectThread (ServerSocket serverSocket) //생성자를 통해 서버소켓을 받음
157. {
158. System.out.println(server.getTime() + " Server opened"); //서버가 열렸다는 메세지 출력
159. this.serverSocket = serverSocket; //서버소켓을 저장
160. }
162. @Override
163. public void run ()
164. {
165. try
166. {
167. while (true) //계속 새로운 클라이언트의 연결을 수락하고 새 소켓을 cLIENTtHREAD에 넘겨줌
168. {
169. Socket socket = serverSocket.accept(); //클라이언트의 연결을 수락
170. System.out.println(" Thread " + count + " is started.");
171. ClientThread clientThread = new ClientThread(socket, count);
172. clientThread.start(); //새로운 클라이언트의 연결을 수락하고 새 소켓을 cLIENTtHREAD에 넘겨줌
173. count++;
174. }
175. } catch (IOException e)
176. {
177. System.out.println(e);
178. System.out.println(" SERVER CLOSE ");
179. }
180. }
181. }
182. public class server
183. {
184. public static void main (String[] args)
185. {
187. ServerSocket serverSocket = null;
188. try
189. { // 서버소켓을 생성, 8080 포트와 binding
190. serverSocket = new ServerSocket(8080); // 생성자 내부에 bind()가 있고, bind() 내부에 listen() 있음
191. ConnectThread connectThread = new ConnectThread(serverSocket); // 서버소켓을 connectThread에 넘겨줌
192. connectThread.start(); // connectThread 시작

195. } catch (IOException e)
196. {
197. e.printStackTrace();
198. }
200. }
202. static String getTime ()
203. {
204. SimpleDateFormat f = new SimpleDateFormat("[hh : mm : ss ]");
205. return f.format(new Date());
206. }
207. }

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | 덧셈 |
|  | |

|  |  |
| --- | --- |
| #2 | 곱셈 |
|  | |

|  |  |
| --- | --- |
| #3 | 나눗셈 |
|  | |

|  |  |
| --- | --- |
| #4 | 잘못된 명령어 입력시 |
|  | |

|  |  |
| --- | --- |
| #1 | 클라이언트가 bye 명령어를 보낼 시 |
|  | |