|  |
| --- |
| import java.net.\*;  import java.io.\*;  import java.util.\*;  class ChatServer  {  int port = 6000; //서버에서 포트번호는 6000으로 설정  ServerSocket server = null; //ServerSocket server 선언(ServerSocket 클래스는 TCP 서버의 역할을 한다. 클라이언트의 연결 요청을 기다리며 요청이 오면 요청을 수락한다.)  Socket child = null; //Socket child 선언(Socket 클래스는 연결된 클래스와 통신하는 역할을 한다.)  HashMap<String, PrintWriter> hm; //key를 String문자열로 지정,value을 PrintWriter로 지정한 HashMap을 hm을 선언  public ChatServer() { //생성자  ChatSverThread sr; //ChatSverTread의 sr객체 선언  Thread t; //쓰레드 객체 선언  try {  server = new ServerSocket(port); //바인딩 포트를 대입하여 sever 객체 생성  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("\* 채팅 서버 \*");  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("클라이언트의 접속을 기다립니다.");  hm = new HashMap<String, PrintWriter>(); //HashMap hm 객체 생성  //무한루프를 도는 while문의 과정은  //accept메소드를 통해 클라이언트의 소켓을 child에 넣어준다.  //클라이언트 소켓과 연결이 되었으면 클라이언트 소켓을 담은 child와 HashMap객체 생성한 hm을 대입한 ChatSverThread객체 sr을 생성해준다.  //스레드를 실행시키기 위해 객체 생성하는데 sr을 실행시켜준다는 의미로 sr을 넣어줍니다.  //그리고 스레드 객체 t를 실행해줍니다.  while (true) {  child = server.accept(); // 클라이언트의 소켓을 연결받습니다.  if (child != null) { // 클라이언트 소켓과 연결시  sr = new ChatSverThread(child, hm); // 채팅 스레드를 생성합니다.  t = new Thread(sr); // 채팅스레드를 시작합니다.  t.start();  }  }  } catch (Exception e) { //catch를 통해 예외처리해준다  e.printStackTrace();  }  }  public static void main(String[] args)  {  new ChatServer();  }  } |
| import java.net.\*;  import java.io.\*;  import java.util.\*;  public class ChatSverThread implements Runnable //스레드 실행하는 부분  {  Socket child; //클라이언트와 통신하기 위한 소켓  BufferedReader ois; //클라이언트로부터 데이터를 수신받기 위한 스트림  PrintWriter oos; //클라이언트에게 데이터를 송신하기 위한 스트림  String user\_id; //접속자의 아이디를 저장할 변수 선언  HashMap<String, PrintWriter> hm; //HashMap hm 선언  InetAddress ip; //ip주소를 저장할 변수 선언  String msg; //메세지를 저장할 변수 선언  public ChatSverThread(Socket s, HashMap<String, PrintWriter> h) { //생성자 부분(소켓 s와 HashMap h를 매개변수로 받아온다.)  child = s; // 넘겨받은 소캣 s를 child로 지정  hm = h; //넘겨받은 HashMap을 hm으로 지정  try {  ois = new BufferedReader(new InputStreamReader(child.getInputStream())); // 클라이언트로부터 데이터를 수신받기 위해 child로부터 입력스트림을 얻는 ois객체 생성  oos = new PrintWriter(child.getOutputStream()); //클라이언트로 메세지를 송신하기 위해 chile의 스트림데이터를 출력해주는 객체 oos 생성  user\_id = ois.readLine(); //클라이언트에게 버퍼에 저장되있는 것을 user\_id에 저장  ip = child.getInetAddress(); //child의 아이피주소를 ip에 저장  System.out.println(ip + "로부터 " + user\_id + "님이 접속하였습니다.");  broadcast(user\_id + "님이 접속하셨습니다."); //broadcast의 메소드에 문자열을 대입하여 실행해준다.  synchronized (hm) { //여러 스레드가 공유하는 hm을 동기화 시켜줌으로써 한 시점에서만 실행될 수 있게 해준다.  hm.put(user\_id, oos); //user\_id를 key로 설정하고, 출력스트림인 객체 oos로 value값으로 설정한다.(put메소드는 key와 value를 저장한다.)  }  }  catch (Exception e) { //catch를 통해 예외처리해준다  e.printStackTrace();  }  }  public void run() { //쓰레드의 실행되는 부분  String receiveData; //데이터를 받는 문자열 변수 선언  try {  while ((receiveData = ois.readLine()) != null) { //버퍼에 저장되어있는 값을 읽어들여와 receiveData에 저장 해주고, 버퍼에 저장되있는 값이 null이 아닐때까지 반복문 수행  if (receiveData.equals("/quit")) { //receiveData(버퍼로 입력받은값이) "/quit" 문자열일때  synchronized (hm) {  hm.remove(user\_id); //HashMap hm객체에 있는 user\_id를 제거해준다.  }  break; //그리고 반복문을 종료해준다.  }  //receiveData에 저장되있는 문자열 "/to"의 인덱스를 리턴해주는데 그 인덱스값이 0보다 크거나 같으면  //sendMsg메소드에 receiveData를 대입하여 실행한다.  else if (receiveData.indexOf("/to") >= 0) {  sendMsg(receiveData);  }  //위에 조건문에 해당이 되지 않는다면 아래 문장을 출력해주고, broadcast메소드에 매개변수를 대입해서 실행시켜 해당 문장 출력  else {  System.out.println(user\_id + " >> " + receiveData);  broadcast(user\_id + " >> " + receiveData);  }  }  }  catch (Exception e) { //catch를 통한 예외처리  e.printStackTrace();  }  finally { //try-catch를 지나 마지막에 반드시 수행되어야하는 finally블럭이다.  synchronized (hm) {  hm.remove(user\_id); //user\_id를 제거해준다.  }  broadcast(user\_id + "님이 퇴장했습니다."); //broadcast메소드를 이용해 해당 문장 출력  System.out.println(user\_id + "님이 퇴장했습니다.");  try {  if (child != null) { //만약 child(소켓)가 null이 아닐때  ois.close(); //BufferedReader와 InputStreamReader를 닫아준다.  oos.close(); //OutputStream을 닫아준다.  child.close(); //Socket도 닫아준다.  }  }  catch (Exception e) { //catch를 통한 예외처리  }  }  }  public void broadcast(String message) {  synchronized (hm) {  try {  for (PrintWriter oos : hm.values()) { //hm에 저장되어있는 값(PrintWriter)들을 하나씩 oos에 넣어준다.(for문 반복작업)  oos.println(message); //oos에 들어간 값(PrintWriter)을 이용해 매개변수로 받은 message를 출력해준다.  oos.flush(); //버퍼값을 비워준다.  }  } catch (Exception e) { //catch를 통해 예외처리해준다.  }  }  }  public void sendMsg(String message) {  int begin = message.indexOf(" ") + 1; //매개변수로 받은 message의 " " 인덱스를 반환한 값에 + 1을 하여 begin에 넣어준다.  int end = message.indexOf(" ", begin); //매개변수로 받은 message의 begin의 값(예를 들면 begin이 2이면 end에는 두번째 공백의 index값이 리턴된다.)번째의 " "의 인덱스를 end에 넣어준다.  if (end != -1) { //만약 end가 -1이 아니라면(문자열의 값이 있다면)  String id = message.substring(begin, end); //message의 begin과 end부분을 잘라서 id에 저장해준다.  String msg = message.substring(end + 1); //message의 end + 1한 나머지 부분을 msg에 저장해준다.  PrintWriter oos = hm.get(id); //hm에 id(key)를 넣어줌으로써 리턴된 value를 oos에 넣어준다.  try {  if (oos != null) { //oos가 null이 아니면  oos.println(user\_id + "님이 다음과 같은 귓속말을 보내셨습니다. : " + msg); //해당 문자출력  oos.flush(); //버퍼 값 비워준다.  }  } catch (Exception e) { //catch를 통해 예외처리해준다.  }  }  }  } |
| import java.net.\*;  import java.io.\*;  import java.util.\*;  public class ChatClient {  String ipAddress; //접속을 요청할 서버의 아이피 주소  static final int port = 6000; //접속을 요청할 서버의 포트번호  Socket client = null; //클라이언트 소켓  BufferedReader read; //키보드로부터 메세지를 읽어올 입력스트림  PrintWriter oos; //서버에 데이터를 전송받기 oos  BufferedReader ois; //서버로부터 데이터를 전송 받기위한 ois  String sendData; //서버로 보낼 데이터를 저장하기 위한 변수  String receiveData; //서버로부터 받은 데이터를 받기 위한 변수  String user\_id;  ReceiveDataThread rt; //ReceiveDataTread는 접속을 계속 유지하면서 데이터 송수신을 하기 위한 클래스이다.  boolean endflag = false;  public ChatClient(String id, String ip) { //생성자 부분  user\_id = id; //매개변수로 받은 id를 user\_id로 저장한다.  ipAddress = ip; //매개변수로 받은 ip를 ipAddress로 저장한다.  try {  System.out.println("\*\*\*\* 클라이언트\*\*\*\*\*");  client = new Socket(ipAddress, port); //접속할 아이피주소와 포트번호 이용하여 client소켓 생성한다.  read = new BufferedReader(new InputStreamReader(System.in)); //키보드로부터 입력한 값을 스트림 데이터로 받아와 버퍼에 저장한 read객체 생성한다.  ois = new BufferedReader(new InputStreamReader(client.getInputStream())); //서버로부터 데이터를 수신받기 위해 client로부터 입력스트림을 얻는 ois객체 생성  oos = new PrintWriter(client.getOutputStream()); //서버로 메세지를 송신하기 위해 client의 스트림데이터를 출력해주는 객체 oos 생성  oos.println(user\_id); //user\_id를 출력해준다.  oos.flush(); //버퍼값을 비워준다.  rt = new ReceiveDataThread(client, ois); //서버가 보낸 데이터를 수신받기 위한 스레드 객체 생성  Thread t = new Thread(rt); //스레드 실행  t.start();  while (true) { //무한 반복문 수행하는 while문  sendData = read.readLine(); //버퍼에 있는 값을 읽어 들여 sendData에 저장한다.  oos.println(sendData); //서버에 send데이터를 송신한다.  oos.flush(); //버퍼값을 비워준다.  if (sendData.equals("/quit")) { //보내는 데이터가 "/quit" 일때  endflag = true; //endflag를 true로 변환해주고 접속을 끊어준다.  break;  }  }  System.out.print("클라이트의 접속을 종료합니다. ");  System.exit(0); //프로그램 종료  } catch (Exception e) { //catch를 통한 예외처리  if (!endflag)  e.printStackTrace();  } finally {  try {  ois.close();//BufferedReader와 InputStreamReader를 닫아준다.  oos.close();//OutputStream을 닫아준다.  client.close();//Socket도 닫아준다.  System.exit(0);//프로그램 종료  } catch (IOException e2) { //catch를 통한 예외처리  e2.printStackTrace();  }  }  }  public static void main(String[] args) {  if (args.length < 2) {  System.out.print("USAGE : java ChatClient 사용자\_id 서버\_ip");  System.exit(0);  }  new ChatClient(args[0], args[1]);  }  } |
| import java.io.BufferedReader;  import java.io.IOException;  import java.net.Socket;  class ReceiveDataThread implements Runnable {  Socket client; //서버와 통신하기 위한 소켓  BufferedReader ois; //서버로부터 데이터를 수신받기 위한 스트림  String receiveData; //서버로부터 수신받은 데이터를 저장하기 위한 변수  public ReceiveDataThread(Socket s, BufferedReader ois) { //생성자 부분(소켓 객체와 입력스트림이 전달된다.)  client = s; //매개변수로 받은 소켓 s를 client에 넣어줌  this.ois = ois; //매개변수로 받은 입력스트림 ois를 ois에 넣어준다.  }  public void run() { //쓰레드 실행부분  try {  while ((receiveData = ois.readLine()) != null) //입력 스티림을 통해 데이터를 가져와 receiveData에 넣어주고 receiveData가 null이 아닐때까지 반복문 수행해준다.  System.out.println(receiveData); //데이터 출력  } catch (Exception e) { //catch를 통한 예외처리  e.printStackTrace();  } finally {  try {  ois.close(); //BufferedReader 닫아준다.  client.close(); //소켓 닫아준다  } catch (IOException e2) { //catch를 통한 예외처리  e2.printStackTrace();  }  }  }  } |