|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **□ 수행평가 - 빅데이터를 활용한 IoT 시스템 개발**  **(feat.스마트시티프로젝트)** | | | | | | |
|  |  |  | |  | |  |
| **과정명** | | 빅데이터를 활용한 IoT시스템 개발(feat. 스마트시티 프로젝트) | | | | |
| **교과목명** | | IoT제어 및 통신기술 | | **훈련교사** | | 이진만 |
| **과정명** | | Network 프로그래밍  IoT 센서연동  CAN 통신 이해 | | | | |
| **수행날짜** | | 2020.03.09 | 훈련생명 | | 유한솔 | |
| **과제개요** | | | | | | |
| 1. 안드로이드 디바이스를 중심으로 Client와 Web Application 연동 2. 시스템 구성도를 그리시오 3. 각 파트의 시스템을 개발 하시오 4. 시스템을 통합 하시오 5. 수행 결과를 입력 하시오 | | | | | | |
| **Pad 서버 및 클라이언트**  **package** com.example.pad;  **import** androidx.appcompat.app.AppCompatActivity;  **import** android.os.AsyncTask; **import** android.os.Bundle; **import** android.util.Log; **import** android.view.View; **import** android.widget.ArrayAdapter; **import** android.widget.Button; **import** android.widget.TextView; **import** android.widget.Toast;  **import** java.io.IOException; **import** java.io.InputStream; **import** java.io.ObjectInputStream; **import** java.io.ObjectOutputStream; **import** java.io.OutputStream; **import** java.net.ServerSocket; **import** java.net.Socket; **import** java.util.Collection; **import** java.util.HashMap; **import** java.util.Iterator;  **import** msg.Msg;  **public class** MainActivity **extends** AppCompatActivity {   TextView **tv**, **tvclient**, **servertv**;   Socket **ssocket**;  String **sip** = **"70.12.231.175"**; *// my computer  // String sip = "70.12.231.175";* **int sport** = 7777;   HashMap<String, ObjectOutputStream>  **maps** = **new** HashMap<String, ObjectOutputStream>();  HashMap<String, String>  **ids** = **new** HashMap<String, String>();  ServerSocket **serverSocket**;   **int port** = 8888;  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  makeUi();   }    **class** serverReady **extends** Thread{   **public** serverReady(){  **try** {  **serverSocket** = **new** ServerSocket(**port**);  Log.*d*(**"-----"**,**"ServerSocket created.."**);  } **catch** (IOException e) {  e.printStackTrace();  }  }   @Override  **public void** run() {  **while**(**true**) {   Socket socket = **null**;   Log.*d*(**"-----"**,**"Server Ready.."**);  **try** {  socket = **serverSocket**.accept();  Log.*d*(**"-----"**,**"client connected.."**);  **if**(socket.getInetAddress().toString().equals(**"/70.12.113.203"**)){  runOnUiThread(**new** Runnable() {  @Override  **public void** run() {  **tvclient**.setText(**"Connected"**);  }  });  }  **else if**(socket.getInetAddress().toString().equals(**"/70.12.231.175"**)) {  runOnUiThread(**new** Runnable() {  @Override  **public void** run() {  **servertv**.setText(**"Connected"**);  }  });  }   **new** Receiver(socket).start();  } **catch** (IOException e) {  e.printStackTrace();  }  }  }  }   **public void** setList(){ *// adapter = // new ArrayAdapter<String>( // MainActivity.this, // android.R.layout.simple\_list\_item\_1, // getIds() // ); // adapter.notifyDataSetChanged(); // listView.setAdapter(adapter);* }   **private void** makeUi() {  **tvclient** = findViewById(R.id.***tvclient***);  **servertv** = findViewById(R.id.***servertv***);  **tv** = findViewById(R.id.***tv***);   **new** ConnectThread(**sip**,**sport**,**null**).start();   }   **public void** ckbt(View v){  **if**(v.getId() == R.id.***startbt***){   Msg msg = **new** Msg(**"Admin"**,**"start"**,**null**);  **new** Sender(msg).start();  **new** serverReady().start();     }**else if**(v.getId() == R.id.***endbt***){  Msg msg = **new** Msg(**"Admin"**,**"stop"**,**null**);  **new** Sender(msg).start();  }   }   **public void** displayData(Msg msg){  **final** String txt = msg.getTxt();  runOnUiThread(**new** Runnable() {  @Override  **public void** run() {   **tv**.setText(txt);  }  });    }   **class** Receiver **extends** Thread{   InputStream **is**;  ObjectInputStream **ois**;   OutputStream **os**;  ObjectOutputStream **oos**;   Socket **socket**;  **public** Receiver(Socket socket) **throws** IOException {  **this**.**socket** = socket;  **is** = socket.getInputStream();  **ois** = **new** ObjectInputStream(**is**);   **os** = socket.getOutputStream();  **oos** = **new** ObjectOutputStream(**os**);  **maps**.put(socket.getInetAddress().toString(),  **oos**);  **try** {  Log.*d*(**"==="**,**"receive thread"**);  Msg msg = (Msg) **ois**.readObject();  **ids**.put(socket.getInetAddress().toString(),  msg.getId());  } **catch** (ClassNotFoundException e) {  e.printStackTrace();  }  }   @Override  **public void** run() {  **while**(**ois** != **null**) {  Msg msg = **null**;  **try** {   msg = (Msg) **ois**.readObject();  **if**(msg.getTxt().equals(**"q"**)) {  System.***out***.println(  **ids**.get(**socket**.getInetAddress().toString())+**":Exit .."**);   **maps**.remove(  **socket**.getInetAddress().toString()  );   **ids**.remove(**socket**.getInetAddress().toString()  );  runOnUiThread(**new** Runnable() {  @Override  **public void** run() {  *// setList();* }  });  **break**;  }  *//sendMsg(msg);* displayData(msg);  } **catch** (Exception e) {  **maps**.remove(  **socket**.getInetAddress().toString()  );   **ids**.remove(**socket**.getInetAddress().toString()  );  runOnUiThread(**new** Runnable() {  @Override  **public void** run() {  *// setList();* }  });  **break**;  }  } *// end while* **try** {  **if**(**ois** != **null**) {  **ois**.close();  }  **if**(**socket** != **null**) {  **socket**.close();  }  }**catch**(Exception e) {  e.printStackTrace();  }  }   }   **class** Sender **extends** Thread{  Msg **msg**;  **public** Sender(Msg msg) {  **this**.**msg** = msg;  }  @Override  **public void** run() {   Collection<ObjectOutputStream>  cols = **maps**.values();  Iterator<ObjectOutputStream>  its = cols.iterator();  **while**(its.hasNext()) {  **try** {  its.next().writeObject(**msg**);  } **catch** (IOException e) {  e.printStackTrace();  }  }  }   }   **class** ConnectThread **extends** Thread {   String **ip**;  **int port**;  String **id**; *//id 추가* OutputStream **os**;  ObjectOutputStream **oos**;   **public** ConnectThread() {  }   **public** ConnectThread(String ip, **int** port, String id) {  **this**.**ip** = ip;  **this**.**port** = port;  *//* **this**.**id** = id;    }   @Override  **public void** run() {   *//Client.java의 Client()에서 try/catch가져오기* **try** {  Log.*d*(**"--------"**,**"run"**); *// Thread.sleep(1000);  // ssocket.setSoTimeout(2000);* **ssocket** = **new** Socket(**ip**, **port**); *//소켓만들고* **os** = **ssocket**.getOutputStream();  **oos** = **new** ObjectOutputStream(**os**);  Msg test = **new** Msg(**"test"**,**"test"**,**null**);  **oos**.writeObject(test);  Log.*d*(**"--------"**,**"run2"**);  runOnUiThread(**new** Runnable() {  @Override  **public void** run() {  **servertv**.setText(**"Connected Server"**); *//현상황찍고  //new Sender(new Msg("tablet","asdf",null)).start();* }  });    } **catch** (Exception e) {  Log.*d*(**"--------"**,**"ex"**);  **int** i =0;   **while** (**true**) {  i++; *//몇번 retry했는지 표시* Log.*d*(**"----"**,i+**""**);  e.printStackTrace();  **final int** finalI = i;  runOnUiThread(**new** Runnable() {  @Override  **public void** run() {  **servertv**.setText(**"Retry Connection"**+ finalI); *//현상황찍고* }  });  *//System.out.println("Retry.."); : 안드로이드에선 사용불가(log.d)  //현재 화면에 있는 text를 가져와서 뿌린다.* **try** {  Thread.*sleep*(1000); *// ssocket.setSoTimeout(2000);* **ssocket** = **new** Socket(**ip**, **port**); *//커넥션시도* runOnUiThread(**new** Runnable() {  @Override  **public void** run() {  **servertv**.setText(**"Connected Server"**); *//현상황찍고* **new** Sender(**new** Msg(**"tablet"**,**"asdf"**,**null**)).start();   }  });   } **catch** (Exception e1) {  e1.printStackTrace();  **this**.start();   }  }  }  *//retry일때도 sender가 형성되어야하므로 여기에 생성* **try** {   *//객체생성* SReceiverTask sreceiverTask = **new** SReceiverTask(**ssocket**); *//리시버만들고* sreceiverTask.execute();    } **catch** (IOException e) {  e.printStackTrace();  }  } *//end run* }   **class** SReceiverTask **extends** AsyncTask<Void, Msg, Void> {   InputStream **is**;  ObjectInputStream **ois**;   **public** SReceiverTask(Socket socket) **throws** IOException {  **is** = socket.getInputStream();  **ois** = **new** ObjectInputStream(**is**);  }    @Override  **protected** Void doInBackground(Void... voids) {  **while** (**ois** != **null**) {  Msg msg = **null**;  **try** {  msg = (Msg) **ois**.readObject();  publishProgress(msg);  } **catch** (Exception e) {  msg = **new** Msg(**"System"**, **"Server is dead"**, **null**);  publishProgress(msg);  **break**;   }  }  **return null**;   }    @Override  **protected void** onPostExecute(Void aVoid) {  **try** {  **if** (**ois** != **null**) {  **ois**.close();  }  **if** (**ssocket** != **null**) {  **ssocket**.close();  }  } **catch** (Exception e) {  e.printStackTrace();  }   }   *//doinbackground가 동작되는 동안 실행(던져주면받음)* @Override  **protected void** onProgressUpdate(Msg... values) {   String id = values[0].getId();  **if**(id.equals(**"Admin"**)){  String txt = values[0].getTxt(); *//서버에서 "1", "2" 이런식으로 온다.* **tv**.setText(txt);*//받았는지 확인* **new** Sender(values[0]).start(); *// if(ssocket != null){ // try { // ssocket.close(); // } catch (IOException e) { // e.printStackTrace(); // new ConnectThread(sip,sport,null).start(); // } // }   // 만약 패드와서버가 통신중에 서버가 죽으면 다시 reconnection을 요구한다.  //일단 소켓을 close 하고 다시 thread를 돌려 reconnection 하는 것이다.  // return;* }    Msg msg = **null**;  *// if(txt.trim().equals("0")){ // msg = new Msg("server","0",null); // }else{ // msg = new Msg("server","1",null); // } // sendMsg(msg);* }  } }  **Java App Client**  package msg;  import java.io.IOException;  import java.io.InputStream;  import java.io.ObjectInputStream;  import java.io.ObjectOutputStream;  import java.io.OutputStream;  import java.net.Socket;  import java.util.ArrayList;  import java.util.Scanner;  public class Client {  Socket socket;  Sender sender;  String vel;  boolean aflag = true;  public Client() {}  public Client(String address,int port) throws IOException {  try {  socket = new Socket(address, port);    }catch(Exception e) {  while(true) {  System.out.println("Retry..");  try {  Thread.sleep(1000);  socket = new Socket(address, port);  break;  } catch (Exception e1) {  //e1.printStackTrace();  }  }  }    System.out.println("Connected Server:"+address);    sender = new Sender(socket);    Msg msg = new Msg("Kwg", null, null);  sender.setMsg(msg);  new Thread(sender).start();    new Receiver(socket).start();  }    class Receiver extends Thread{  InputStream is;  ObjectInputStream ois;    public Receiver(Socket socket) throws IOException {  is = socket.getInputStream();  ois = new ObjectInputStream(is);  }  @Override  public void run() {  while(ois != null) {  Msg msg = null;  try {  msg = (Msg) ois.readObject();    if(msg.getIps() == null || msg.getIps().size() == 0) {  System.out.println(  msg.getId()+":"+msg.getTxt()  );  }else {  ArrayList<String> list  = msg.getIps();  System.out.println(list);  }    if(msg.getId().equals("Admin")) {  if(msg.getTxt().equals("stop")) {  vel="0";  aflag=false;  }    else if(msg.getTxt().equals("start")) {    aflag=true;  }    }    }catch(Exception e) {  System.out.println("Server Die");  break;  }  }    try {  if(ois != null) {  ois.close();  }  if(socket != null) {  socket.close();  }  }catch(Exception e) {  e.printStackTrace();  }  }    }      class Sender implements Runnable{  OutputStream os;  ObjectOutputStream oos;  Msg msg;    public Sender(Socket socket) throws IOException {  os = socket.getOutputStream();  oos = new ObjectOutputStream(os);  }  public void setMsg(Msg msg) {  this.msg = msg;  }  @Override  public void run() {  if(oos != null) {  try {  oos.writeObject(msg);  } catch (IOException e) {  e.printStackTrace();  }  }  }    }      public void startClient() {    while(true) {    try {  Msg msg = new Msg("kim","0",null);  if(aflag) {  vel=Math.floor(Math.random()\*30+60)+"";  msg.setTxt(vel);  }  System.out.println("vel: "+vel);    sender.setMsg(msg);  new Thread(sender).start();  Thread.sleep(4000);  } catch (InterruptedException e) {  e.printStackTrace();  break;  }    }  try {  socket.close();  } catch (IOException e) {  e.printStackTrace();  }  System.out.println("End Client.");    }      public static void main(String[] args) {  Client client = null;  try {  client = new Client("70.12.231.197", 8888);  client.startClient();  } catch (IOException e) {  e.printStackTrace();  }    }  }  **Web Server**  package server;  import java.io.IOException;  import java.io.InputStream;  import java.io.ObjectInputStream;  import java.io.ObjectOutputStream;  import java.io.OutputStream;  import java.net.ServerSocket;  import java.net.Socket;  import java.util.Collection;  import java.util.HashMap;  import java.util.Iterator;  import msg.Msg;  public class Server {  HashMap<String, ObjectOutputStream> maps = new HashMap<>();  ServerSocket serverSocket;  boolean aflag = true;  public Server() {  }  public Server(int port) throws IOException {  serverSocket = new ServerSocket(port);  System.out.println("Start Server");  Runnable r = new Runnable() {  @Override  public void run() {  while (aflag) {  Socket socket = null;  try {  System.out.println("Server Ready..");  socket = serverSocket.accept();  System.out.println("Client Ready..");  System.out.println(socket.getInetAddress());  makeOut(socket);  System.out.println("makeOut");  new Receiver(socket).start();  } catch (IOException e) {  e.printStackTrace();  }  }  }  };  new Thread(r).start();  }  public void makeOut(Socket socket) throws IOException {  OutputStream os;  ObjectOutputStream oos;  os = socket.getOutputStream();  oos = new ObjectOutputStream(os);  maps.put(socket.getInetAddress().toString(), oos);  System.out.println("접속자수:" + maps.size());  }  class Receiver extends Thread {  InputStream is;  ObjectInputStream ois;  Socket socket;  public Receiver(Socket socket) {  System.out.println("hi Receiver");  this.socket = socket;  try {  is = socket.getInputStream();  ois = new ObjectInputStream(is);  } catch (IOException e) {  e.printStackTrace();  }  System.out.println("접속자수:" + maps.size());  }  @Override  public void run() {  Msg msg = null;  while (ois != null) {  try {  msg = (Msg) ois.readObject();  System.out.println(msg.getId() + ":" + msg.getTxt());  if (msg.getTxt().equals("q")) {  System.out.println(msg.getId() + ":Exit ..");  maps.remove(socket.getInetAddress().toString());  System.out.println("접속자수:" + maps.size());  break;  }  sendMsg(msg);  } catch (Exception e) {  maps.remove(socket.getInetAddress().toString());  System.out.println(socket.getInetAddress() + ":Exit ..");  System.out.println("접속자수:" + maps.size());  break;  }  }  // sendMsg(msg);  try {  if (ois != null) {  ois.close();  }  if (socket != null) {  socket.close();  }  } catch (Exception e) {  e.printStackTrace();  }  }  }  class Sender extends Thread {  Msg msg;  public Sender(Msg msg) {  this.msg = msg;  }  @Override  public void run() {  // HashMap에 있는 oos를 꺼낸다음  // for문을 돌리면서 전송 한다.  Collection<ObjectOutputStream> cols = maps.values();  Iterator<ObjectOutputStream> its = cols.iterator();  while (its.hasNext()) {  try {  its.next().writeObject(msg);  System.out.println("sender : " + msg.getTxt());  } catch (IOException e) {  e.printStackTrace();  }  }  }  }  class Sender2 extends Thread {  Msg msg;  public Sender2(Msg msg) {  this.msg = msg;  }  @Override  public void run() {  String ip = msg.getTid();  try {  maps.get(ip).writeObject(msg);  } catch (IOException e) {  e.printStackTrace();  }  }  }  public void sendMsg(Msg msg) {  System.out.println("hihi");  String ip = msg.getTid();  Sender sender = new Sender(msg);  sender.start();  }  public static void main(String[] args) {  Server server = null;  try {  server = new Server(7777);  } catch (IOException e) {  e.printStackTrace();  }  }  }  **Web Server\_Client**  package client;  import java.io.IOException;  import java.io.InputStream;  import java.io.ObjectInputStream;  import java.io.ObjectOutputStream;  import java.io.OutputStream;  import java.net.Socket;  import java.util.ArrayList;  import java.util.Random;  import msg.Msg;  public class Client {  Socket socket;  Sender sender;  public Client() {  }  public Client(String address, int port, Msg msg) throws IOException {  try {  socket = new Socket(address, port);  } catch (Exception e) {  while (true) {  System.out.println("Retry..");  try {  Thread.sleep(1000);  socket = new Socket(address, port);  break;  } catch (Exception e1) {  }  }  }  System.out.println("Connected Server:" + address);  sender = new Sender(socket);  sender.setMsg(msg);  new Thread(sender).start();  }  class Sender implements Runnable {  OutputStream os;  ObjectOutputStream oos;  Msg msg;  public Sender(Socket socket) throws IOException {  os = socket.getOutputStream();  oos = new ObjectOutputStream(os);  }  public void setMsg(Msg msg) {  this.msg = msg;  }  @Override  public void run() {  if (oos != null) {  try {  System.out.println(msg.getTxt());  oos.writeObject(msg);  } catch (IOException e) {  if (oos != null) {  try {  oos.close();  } catch (IOException e1) {  e1.printStackTrace();  }  return;  }  }  }  }  }  public static void main(String[] args) {  // Client client = null;  // try {  // client = new Client("70.12.225.90", 8888);  // //client.startClient2();  // } catch (IOException e) {  // e.printStackTrace();  // }  }  }  **Servlet**  package com.sds;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import client.Client;  import msg.Msg;  /\*\*  \* Servlet implementation class WebClientServlet  \*/  @WebServlet({ "/WebClientServlet", "/webclient" })  public class WebClientServlet extends HttpServlet {  private static final long serialVersionUID = 1L;    Client client;    /\*\*  \* @see HttpServlet#HttpServlet()  \*/  public WebClientServlet() {  }  /\*\*  \* @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)  \*/  protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {    System.out.println("hi servlet");    String ip = request.getParameter("ip");  String speed = request.getParameter("speed");    System.out.println(ip + " " + speed);    Msg msg = new Msg("Admin", speed, ip);  try {  client = new Client("70.12.113.248", 7777, msg);  } catch (IOException e) {  e.printStackTrace();  }  }  }  **Html**  <!DOCTYPE html>  <html>  <head>  <meta charset="EUC-KR">  <title>2020</title>  </head>  <body>  <h1>Test Page</h1>  <form action="webclient" method="post">  IP<input type="text" name="ip"><br>  SPEED<input type="text" name="speed"><br>  <input type="submit" value="send">  </form>  </body>  </html>  **수행 결과**   * 웹 서버와 앱 서버가 Pad에 연결될 시 연결되었는지 여부를 표시함. * Start button을 누르면 랜덤의 숫자를 데이터 출력란에 표시함 * 웹페이지에서 값을 입력하면 데이터 출력란에 랜덤값 대신 입력된 값이 출력됨 * End button을 누르면 출력란 값이 0으로 변경됨      * 웹페이지      * 웹페이지에서 웹서버를 통해 Pad서버로 연결된 모습      * 웹서버에 웹페이지와, Pad서버 2개가 접속된 모습      * Pad 서버에 랜덤값과 웹페이지에 입력된 값이 출력되는 모습      * 웹서버와 연결되었음을 표시      * 웹서버로부터 웹페이지에 입력된 값을 받음      * 앱 클라이언트와 연결되었고, 그로부터 랜덤값을 전달받고 있음      * 랜덤값 출력을 멈추고, 웹페이지 입력값 을 출력함      * End 버튼을 누르면 출력값이 0으로 변경됨 | | | | | | |