2017학년도 1학기

JAVA 프로그래밍 II

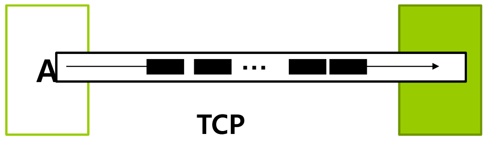
514770-1
2017년 봄학기
5/24/2017
박경신

Lab #7 (Networking)

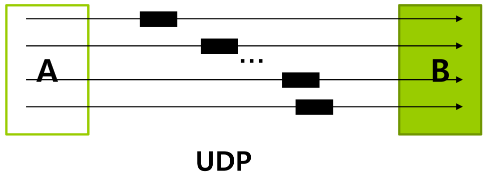
- 기존 요구사항 분석
  - Lab #6는 Thread, Runnable과 SwingWorker를 이용한 다양한 멀티스레드 기능을 사용
  - **Lab #7는 TCP/UDP 등 다양한 네트워크 프로그래밍 기능을 사용**
- TCP, UDP, HTTP, File Transfer

TCP vs UDP

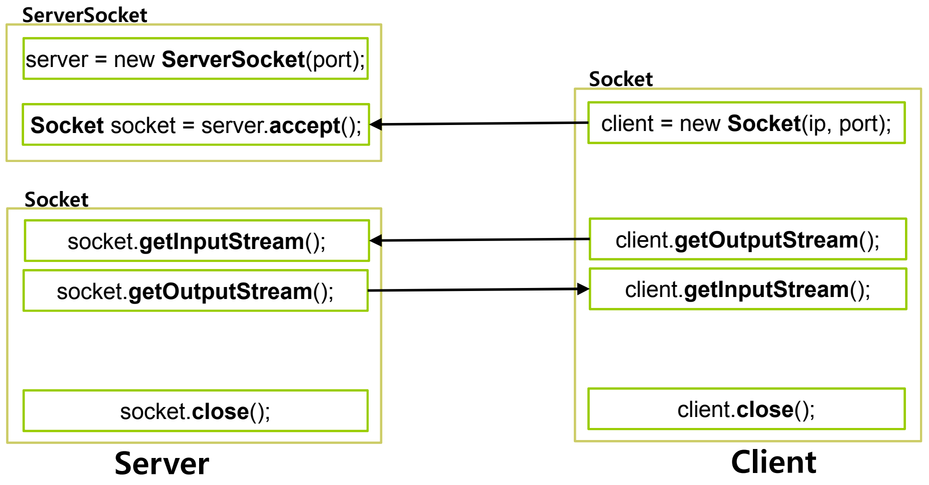
- **TCP** is a **connection-oriented reliable stream** transport protocol



- **UDP** is a **connectionless unreliable datagram** transport protocol



TCP Java Socket


ServerSocket
server = new ServerSocket(port);
Socket socket = server.accept();
Socket
socket.getInputStream();
socket.getOutputStream();
socket.close();
Server
Socket
client = new Socket(ip, port);
client.getOutputStream();
client.getInputStream();
client.close();
Client

Lab #7_1 TCP

- Lab#7_1에서는 TCPServer/TCPClient 클래스를 구현한다.

```
public class TCPServer {
    private ServerSocket serverSocket = null;
// 중간생략..
    public int init(int port) {
        serverSocket = new ServerSocket(port);
        if (serverSocket != null) return OK;
    }
    public TCPClient checkForNewConnections() {
        Socket s = serverSocket.accept();
        if (s != null) return new TCPClient(s);
    }
}
```

Lab #7_1 TCP

```
public class TCPClient {
    private Socket socket = null;
    public int connectToServer(String ip, int port) {
        socket = new Socket(ip, port);
        if (socket != null) return OK;
    }
    public void close() {        socket.close();        }
    public int read(byte[] buffer, int nbytes, boolean blocking) {
        InputStream is = socket.getInputStream();
        is.read(buffer, 0, nbytes);
    }
    public int write(byte[] buffer, int nbytes, boolean blocking) {
        OutputStream os = socket.getOutputStream();
        os.write(buffer, 0, nbytes);
    }
}
```

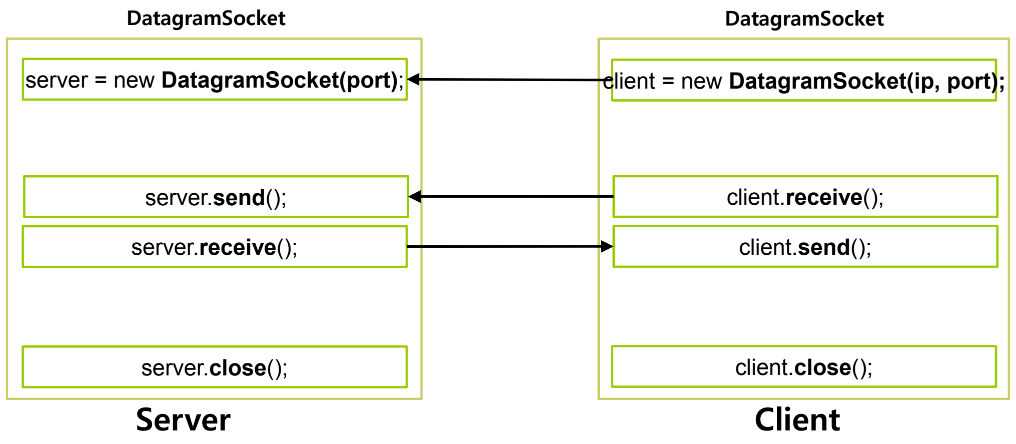
Lab #7_1 TCP

```
public class TCPServerTest {
    TCPServer server = null; TCPClient client = null;
    byte[] sendData = new byte[512]; bytes[] receiveData = new byte[512];
    public TCPServerTest(int port) {    server = new TCPServer(port);    }
    public void listen() {
        client = server.checkForNewConnections();
    }
    public String receive() {
        client.read(receiveData, 512, true);
        return Utility.getReceivedString(receiveData);
    }
    public void send(String s) {
        Utility.paddingBytesArray(sendData, 512, s.getBytes(), s.length());
        client.write(sendData, 512, true);
    }
}
```

Lab #7_1 TCP

```
public class TCPClientTest {
    TCPClient client = null;
    byte[] sendData = new byte[512]; bytes[] receiveData = new byte[512];
    public TCPClientTest(String ip, int port) {    client = new TCPClient(ip, port);    }
    public String receive() {
        client.read(receiveData, 512, true);
        return Utility.getReceivedString(receiveData);
    }
    public void send(String s) {
        Utility.paddingBytesArray(sendData, 512, s.getBytes(), s.length());
        client.write(sendData, 512, true);
    }
}
```

UDP Java Socket



Lab #7_2 UDP

- Lab#7_2에서는 UDPSocket 클래스를 구현한다.
  - *public class UDPSocket {*

```
    private DatagramSocket socket = null;
    public UDPSocket(int port) {
        socket = new DatagramSocket(port);
    }
    public UDPSocket(String ip, int port) {
        socket = new DatagramSocket(); setSendAddress(ip, port);
    }
    public void setSendAddress(String ip, int port) {
        destAddr = InetAddress.getByName(ip);       destPort = port;
    }
    public void copyReceiveAddressToSendAddress() {
        setSendAddress(receivedAddr, receivedPort);
    }
```

Lab #7_2 UDP

```
    public int send(byte[] buffer, int size) {
        DatagramPacket dp = new DatagramPacket(buffer, size, destAddr, destPort);
        socket.send(dp);
    }
    public byte[] receive(int size) {
        byte[] buffer = new byte[size];
        DatagramPacket dp = new DatagramPacket(buffer, size);
        socket.receive(dp);
        receivedAddr = dp.getAddress();
        receivedPort = dp.getPort();
    }
}
```

Lab #7_2 UDP

- *public class UDPServerTest {*

```
    UDPSocket server = null;
    byte[] sendData = new byte[512];
    public UDPServerTest(int port) {    server = new UDPSocket(port);  }
    public void copyReceiveAddressToSendAddress() {
        server.copyReceiveAddressToSendAddress();
    }
    public String receive() {
        byte[] receiveData = server.receive(512);
        return Utility.getReceivedString(receiveData);
    }
    public void send(String s) {
        Utility.paddingBytesArray(sendData, 512, s.getBytes(), s.length());
        server.send(sendData, 512);
    }
}
```

Lab #7_2 UDP

```
public class UDPClientTest {
  UDPSocket client = null;
  byte[] sendData = new byte[512];
  public UDPClientTest(String ip, int port) {    client = new UDPSocket(ip, port);       }
  public String receive() {
      byte[] receiveData = client.receive(512);
      return Utility.getReceivedString(receiveData);
  }
  public void send(String s) {
      Utility.paddingBytesArray(sendData, 512, s.getBytes(), s.length());
      client.send(sendData, 512);
  }
}
```

Lab #7_3 HTTP

- Lab#7_3에서는 HTTPClient 클래스를 구현한다.

```
public class HTTPClient {
  private void requestGet(String url) throws Exception {
  URL obj = new URL(url);
  HttpURLConnection con = (HttpURLConnection) obj.openConnection();
  con.setRequestMethod("GET"); con.setRequestProperty("User-Agent", USER_AGENT);
  int responseCode = con.getResponseCode();
  System.out.println("Sending 'GET' request to URL : " + url);
  System.out.println("Response Code : " + responseCode);
  BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream())
  String inputLine;
  StringBuffer response = new StringBuffer();
  while ((inputLine = in.readLine()) != null) { response.append(inputLine); }
  in.close(); System.out.println(response.toString());
  }
```

Lab #7_4 ImageTransfer

- Lab#7_4에서는 C:/JAVA/IMG1~5.jpg를 보내는 ImageTransferServer/Client 클래스를 구현한다.

```
public class ImageTransferClient implements Runnable {
  TCPClient client = null; byte[] imageInByte; byte[] sizeBuf;
  public ImageTransferClient(String ip, int port) {   client = new TCPClient(ip, port); }
  public void run() { int i = 0;
      while(i<5) { i++;
          BufferedImage image = ImageIO.read(new File(imagefile));
          imageInByte = convertImageToByteArray(image));
          ByteArrayOutputStream bos = new ByteArrayOutputStream(4);
          dos.writeInt(imageInByte.length);
          sizeBuf = bos.toByteArray();
          client.write(sizeBuf, 4, true);
          client.write(imageInByte, imageInByte.length, true);
          }
      }
  }
}
```

Lab #7_4 ImageTransfer

```
public class ImageTransferServer implements Runnable {
  TCPServer server = null; TCPClient client = null; byte[] imageInByte; byte[] sizeBuf;
  public ImageTransferServer(int port) {   server = new TCPServer(port);       }
  public void listen() {    client = server.checkForNewConnections();    }
  public void run() { int imageSize = 0; int i = 0;
     while(i<5) { i++;
        int nread = client.read(sizeBuf, 4, true);
        if (nread == SIZE_OF_INT) {
           DataInputStream is = new DataInputStream(new ByteArrayInputStream(sizeBuf));
           imageSize = is.readInt();
        }
        if (imageSize > 0) {
           imageInByte = new byte[imageSize];
           nread = client.read(imageInByte, imageSize, true);
           convertByteArrayToImage(imageInByte, "jpg", "outputfilename.jpg");
        }
     }
  }
,
```

Lab #7_5 PhotoToggleButton & ImageTransfer

- Lab#7_5에서는 Lab5_4에 networking을 추가해서 ImageServerFrame 클래스는 client로부터 이미지를 받으면 메인프레임에 나타나게한다.

```
public class ImageUploadClient implements Runnable {
  Socket socket = null; String imagefile = null;
  public ImageUploadClient(Socket socket, String imagefile) {
     this.socket = socket; this.imagefile = imagefile;
  }
  public void run() {
    BufferedImage image = ImageIO.read(new File(imagefile));
    imageIO.write(image, "jpg", socket.getOutputStream()); // write
  }
  public static void main(String[] args) {
     Socket socket = new Socket(ip, port);
     ImageUploadClient client = new ImageUploadClient(socket, filename);
     new Thread(client).start();
  }
}
```

Lab #7_5 PhotoToggleButton & ImageTransfer

```
public class ImageUploadServer implements Runnable {
  Socket socket = null; String dir = "C:/JAVA";
  public ImageUploadServer(Socket socket) {    this.socket = socket;  }
  public void run() {
    InputStream is = socket.getInputStream(); // read
    BufferedImage img = ImageIO.read(ImageIO.createImageInputStream(is));
    ImageIO.write(img, "jpg", new File("uploadedfilename.jpg"));
    is.close();
  }
  public static void main(String[] args) {
     ServerSocket server = new ServerSocket(port);
     while(true) { Socket socket = new serverSocket.accept();
        ImageUploadServer server = new ImageUploadServer(socket);
        new Thread(client).start();
     }
  }
}
```

Lab #7_5 PhotoToggleButton & ImageTransfer

```
public class ImageServerThread implements Runnable {
  ServerSocket serverSocket = nul; Socket socket = null; ImageServerFrame frame;
  public ImageServerThread(ImageServerFrame frame) {
     this.frame = frame; serverSocket = new ServerSocket(port);
  } public void setDone() { done = true; socket.close(); }
  public void run() {
     while(!done) {
        socket = new serverSocket.accept();
        InputStream is = socket.getInputStream(); // read
        BufferedImage img = ImageIO.read(ImageIO.createImageInputStream(is));
        String path = dir + " /" + System.currentTimeMillis() + ".jpg";
        String name= path.substring(path.lastIndexOf('/')+1, path.lastIndexOf('.'));
        ImageIO.write(img, "jpg", new File(path));
        is.close();
        frame.loadButton(name, path);
     }
  }
}
```

Lab #7_5 PhotoToggleButton & ImageTransfer

```
public class ImageServerFrame implements Runnable {
  ImageServerThread upload = null; // networking
  public ImageServerFrame() {
     for(int i=0; i<5; i++) { loadButton("img"+(i+1), "dir+"/IMG"+(i+1)+".jpg"); }
     addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
           upload.setDone(); e.getWindow().dispose();
        }
     });
     upload = new ImageServerThread(this);
     new Thread(upload).start();
  }
  public void loadButton(String name, String path) {
     PhotoToggleButton b = new PhotoToggleButton(name, path);
     buttons.add(b); panel.add(b);
     revalidate(); // revalidate event
  }
}
```

과제 제출

- Lab07_1 ~ Lab07_5와 보고서를 전체적으로 묶어서 e-learning에 과제 제출
- 각 Lab마다 **본인이 추가로 작성한 코드**와 설명을 중점적으로 보고할 것!