1. **TCP**

public class TCPSyntax {

public static void main(String[] args) throws IOException, ClassNotFoundException {

String host = "";

int port = 2207;

Socket socket = new Socket(host, port);

**// DataInputStream/DataOutputStream**

// Hỗ trợ read/write kiểu dữ liệu nguyên thuỷ

DataInputStream dis = new DataInputStream(socket.getInputStream());

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

String sendData1 = "";

dos.writeUTF(sendData1);

int receive1 = dis.readInt();

int res1 = 0;

dos.writeInt(res1);

**// BufferReader/BufferWriter**

// Hỗ trợ read/write kiểu dữ liệu nguyên thuỷ

BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

String sendData2 = "";

bw.write(sendData2);

// bw.flush giúp xoá bộ nhớ đệm để đẩy dữ liệu đi (bw.write) ngay cả khi chưa đủ kích thước của bộ đệm

bw.newLine();

bw.flush();

String receive2 = br.readLine();

String res2 = "";

bw.write(res2);

bw.flush();

**// InputStream/OutputStream**

// Không hỗ trợ kiểu dữ liệu nguyên thuỷ, chỉ làm việc với kiểu dữ liệu byte

InputStream is = new DataInputStream(socket.getInputStream());

OutputStream os = new DataOutputStream(socket.getOutputStream());

String sendData3 = "";

os.write(sendData3.getBytes());

byte[] buff = new byte[1024];

is.read(buff);

String receive3 = new String(buff);

String res3 = "";

os.write(res3.getBytes());

**// ObjectInputStream/ObjectOutputStream**

ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());

ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());

// dùng writeObject

String sendData4 = "";

oos.writeObject(sendData4);

// nhớ ép kiểu

Student st = (Student) ois.readObject();

oos.writeObject(st);

ois.close();

oos.close();

//

socket.close();

}

}

1. **UDP**

public class Main {

public static void main(String[] args) throws IOException, ClassNotFoundException {

String msv = "";

String host = "";

int port = 2207;

InetAddress IPAddress = InetAddress.getByName(host);

DatagramSocket client = new DatagramSocket();

client.send(new DatagramPacket(msv.getBytes(), msv.length(), IPAddress, port));

byte[] receiveBuff = new byte[1024];

DatagramPacket dp = new DatagramPacket(receiveBuff, receiveBuff.length);

client.receive(dp);

ByteArrayInputStream bis = new ByteArrayInputStream(dp.getData());

ObjectInputStream ois = new ObjectInputStream(bis);

Student933 st1 = (Student933) ois.readObject();

ByteArrayOutputStream bos = new ByteArrayOutputStream();

ObjectOutputStream oos = new ObjectOutputStream(bos);

oos.writeObject(st1);

byte[] sendBuff = bos.toByteArray();

client.send(new DatagramPacket(sendBuff, sendBuff.length, IPAddress, port));

client.close();

}

}

1. **GCD**

int gcd(int a, int b){

// Lặp tới khi 1 trong 2 số bằng 0

while (a\*b != 0){

if (a > b){

a %= b; // a = a % b

}else{

b %= a;

}

}

return a + b; // return a + b, bởi vì lúc này hoặc a hoặc b đã bằng 0.

}

1. **Compare**

ArrayList<Integer> list = new ArrayList<>();

for (String s : arr) {

list.add(Integer.parseInt(s));

}

Collections.sort(list, new Comparator<Integer>() {

@Override

public int compare(Integer o1, Integer o2) {

return o1 < o2 ? -1 : 1;

}

});

1. **Treeset**

* các phần tử sẽ được đặt theo thứ tự tăng dần tự

String[] arr = data[1].split(",");

TreeSet<Integer> set = new TreeSet<>();

for (String s : arr) {

set.add(Integer.parseInt(s));

}

1. **LinkedHashSet**

* duy trì thứ tự chèn của các phần tử

LinkedHashSet<String> set = new LinkedHashSet<>();

for (String s : arr) {

set.add(s);

}

Iterator<String> iterator = set.iterator();

while (iterator.hasNext()) {

res += iterator.next();

}

1. **UDP Object**
   1. **Product937**

public static class Product937 implements Serializable {

private static final long serialVersionUID = 937L;

public String id;

public String code;

public String name;

public int quantity;

public Product937(String id, String code, String name, int quantity) {

this.id = id;

this.code = code;

this.name = name;

this.quantity = quantity;

}

}

public static void main(String[] args) throws IOException {

String request = ";B20DCCN535;937";

DatagramSocket socket = new DatagramSocket();

send(socket, request);

DatagramPacket packet = new DatagramPacket(new byte[1024], 1024);

socket.receive(packet);

ByteArrayInputStream bais = new ByteArrayInputStream(packet.getData());

ObjectInputStream ois = new ObjectInputStream(bais);

UDP.Product937 product = null;

try {

product = (UDP.Product937) ois.readObject();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

assert product != null;

String quantity = String.valueOf(product.quantity);

String reversed = new StringBuilder(quantity).reverse().toString();

product.quantity = Integer.parseInt(reversed);

String[] names = Arrays.stream(product.name.split(" ")).toArray(String[]::new);

String tmp = names[0];

names[0] = names[names.length - 1];

names[names.length - 1] = tmp;

product.name = String.join(" ", names);

// ByteArrayOutputStream baos = new ByteArrayOutputStream();

// ObjectOutputStream oos = new ObjectOutputStream(baos);

// oos.writeObject(product);

// oos.flush();

// byte[] buffer = baos.toByteArray();

// socket.send(new DatagramPacket(buffer, buffer.length, packet.getAddress(), packet.getPort()));

socket.close();

}

* 1. **Student933**

public class A933 {

public static void main(String[] args) {

try {

DatagramSocket socket = new DatagramSocket();

InetAddress host = InetAddress.getByName("localhost");

int port = 2207;

String studentCode =

byte[] write = studentCode.getBytes();

DatagramPacket sendPacket = new DatagramPacket(write, write.length, host, port);

socket.send(sendPacket);

byte[] read = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(read, read.length);

socket.receive(receivePacket);

ByteArrayInputStream inputStream = new ByteArrayInputStream(read);

ObjectInputStream ois = new ObjectInputStream(inputStream);

Student933 student933 = (Student933) ois.readObject();

String nameStudent = student933.getName();

student933.setName( chuanHoaChuoi(nameStudent));

student933.setEmail(chuannHoaEmail(nameStudent));

ByteArrayOutputStream outputStream = new ByteArrayOutputStream();

ObjectOutputStream oos = new ObjectOutputStream(outputStream);

oos.writeObject(student933);

byte[] write1 = outputStream.toByteArray();

DatagramPacket sendPacket1 = new DatagramPacket(write1, write1.length, host, port);

socket.send(sendPacket1);

} catch (Exception e) {

e.printStackTrace();

}

}

public static String chuanHoaChuoi(String st){

String[] s= st.trim().toLowerCase().split("\\s+");

st="";

for(int i=0;i<s.length;i++)

st+=s[i].substring(0,1).toUpperCase() + s[i].substring(1) +" ";

st = st.substring(0,st.length()-1);

return st;

}

public static String chuannHoaEmail(String st) {

String[] s= st.trim().toLowerCase().split("\\s+");

st = s[s.length - 1];

for(int i=0 ; i < s.length-1 ; i++)

st += s[i].substring(0,1);

st += "@ptit.edu.vn";

return st;

1. **TCP Object (917)**

public class ObjectStream {

public static void main(String[] args) throws IOException, ClassNotFoundException {

final Socket socket = new Socket("localhost", 2208);

/\* ============ !!! NHỚ KHỞI TẠO OutputStream trước InputStream !!! ======================= \*/

final ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());

final ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());

/\* == !!! CHỈ DÙNG MỖI readObject() xuyên suốt trong bài, ko được dùng readUTF... !!! ===== \*/

oos.writeObject("B20DCCN535;917");

Product917 product917 = (Product917) ois.readObject();

String[] nameWords = product917.getName().split(" ");

String tmp = nameWords[0];

nameWords[0] = nameWords[nameWords.length - 1];

nameWords[nameWords.length - 1] = tmp;

String newName = String.join(" ", nameWords);

product917.setName(newName);

String num = String.valueOf(product917.getQuantity());

String newNum = new StringBuilder().append(num).reverse().toString();

product917.setQuantity(Integer.parseInt(newNum));

oos.writeObject(product917);

socket.close();

}

}

1. **Web service (961) -** nhớ chạy clean and build trước khi bấm run

public class WS\_BAITHI {

public static void main(String[] args) {

// TODO code application logic here

ws.NumberS\_Service service = new ws.NumberS\_Service();

ws.NumberS port = service.getNumberSPort();

String str = port.getNumber("");

System.out.println(str);

String id = str.trim().split(";")[0];

String string = str.trim().split(";")[1];

System.out.println(id);

System.out.println(string);

String[] s = string.trim().split(",");

for (int i = 0; i < s.length; i++) {

for (int j = 0; j < s.length - 1; j++) {

String so1 = s[j].trim() + s[j + 1].trim();

String so2 = s[j + 1].trim() + s[j].trim();

if (Long.parseLong(so1) < Long.parseLong(so2)) {

String sodoi = s[j];

s[j] = s[j + 1];

s[j + 1] = sodoi;

}

}

}

String kq = "";

for (int i = 0; i < s.length; i++) {

kq += s[i].trim();

}

System.out.println(kq);

port.greatestNumber(Integer.parseInt(id), kq);