МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО

ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ОТЧЕТ

По лабораторной работе № 6

Тема: Передача данных без установления соединения

Группа: АВТ-610

Студенты: Князев Р.И., Дунаев Н.Ю., Яковенко Е.В.

Преподаватель: Токарев В.Г.

Новосибирск 2018

1. Цели и задания работы

Задание в л.р.5 реализовать с использованием класса дейтаграмм протокола UDP без установления соединения (класс DatagramSocket). Флажком выделяется «первое/второе» приложение, отличающиеся только номером порта, на котором они прослушивают прием. Например, первое принимает данные на порте 7001, а передает на 7002, второе – наоборот. Для конвертации XML-строки, содержащей передаваемый объект, в массив байтов для передачи в дейтаграммный сокет использовать классы ByteArrayOutputStream/ByteArrayOutputStream в качестве классаисточника текстового потока.

Функционал программы:

**while** (**true**) {  
 *// Получение датаграммы* **datagramPacket** = **new** DatagramPacket(**new byte**[1024], 1024);  
 DatagramSocket datagramSocket = **new** DatagramSocket(9876);  
 datagramSocket.receive(**datagramPacket**);  
 **PORT** = **datagramPacket**.getPort();  
 **byte**[] newData = **datagramPacket**.getData();  
 **int** c = ByteBuffer.*wrap*(newData).getInt();  
 System.***out***.println(**"Message received: "** + c);  
  
 **switch** (c)  
 {  
 *// Получение объекта* **case** 1:  
 datagramSocket.receive(**datagramPacket**);  
 newData = **datagramPacket**.getData();  
 **int** type = ByteBuffer.*wrap*(newData).getInt();  
 datagramSocket.receive(**datagramPacket**);  
 newData = **datagramPacket**.getData();  
 **int** x = ByteBuffer.*wrap*(newData).getInt();  
 datagramSocket.receive(**datagramPacket**);  
 newData = **datagramPacket**.getData();  
 **int** y = ByteBuffer.*wrap*(newData).getInt();  
 ClassForSer classForSer = **new** ClassForSer(x, y, type);  
 **mas**.add(classForSer);  
 datagramSocket.close();  
 System.***out***.println(**"Объект получен, размер массива "** + **mas**.size() + **" объект х = "** + **mas**.get(**mas**.size()-1).**x** + **" y = "** +  
 **mas**.get(**mas**.size()-1).**y** + **" type = "** + **mas**.get(**mas**.size()-1).**type**);  
 **break**;  
  
 *// Отпавка объектов из mas* **case** 2:  
 DatagramSocket ds = **new** DatagramSocket();  
 sendSize(2705);  
 DatagramPacket datagramPacket;  
 **byte**[] bytes;  
  
 **for**(**int** i = 0; i < **mas**.size(); i++) {  
 bytes = ByteBuffer.*allocate*(4).putInt(**mas**.get(i).**type**).array();  
 datagramPacket = **new** DatagramPacket(bytes, bytes.**length**, **host**, 2705);  
 ds.send(datagramPacket);  
 bytes = ByteBuffer.*allocate*(4).putInt(**mas**.get(i).**x**).array();  
 datagramPacket = **new** DatagramPacket(bytes, bytes.**length**, **host**, 2705);  
 ds.send(datagramPacket);  
 bytes = ByteBuffer.*allocate*(4).putInt(**mas**.get(i).**y**).array();  
 datagramPacket = **new** DatagramPacket(bytes, bytes.**length**, **host**, 2705);  
 ds.send(datagramPacket);  
 }  
 ds.close();  
 datagramSocket.close();  
 System.***out***.println(**"Объекты отправлены"**);  
 **break**;  
  
 *// Получить размер вектора* **case** 3:  
 sendSize(2700);  
 datagramSocket.close();  
 System.***out***.println(**"Передан размер вектора: "** + **mas**.size());  
 **break**;  
  
 *// Очистить вектор* **case** 4:  
 **mas** = **new** Vector<>();  
 System.***out***.println(**"Вектор очищен"**);  
 datagramSocket.close();  
 **break**;  
 }  
  
}

Отправление размера массива:

**public void** sendSize(**int** port) **throws** Exception  
{  
 **byte**[] bytesS = ByteBuffer.*allocate*(4).putInt(**mas**.size()).array();  
 DatagramPacket dp = **new** DatagramPacket(bytesS, bytesS.**length**, **host**, port);  
 DatagramSocket dataSize = **new** DatagramSocket();  
 dataSize.send(dp);  
 dataSize.close();  
}

*//Отправка объекта серверу***if**(e.getKeyCode() == KeyCode.NUMPAD8.getCode())  
{  
 **try** {  
 **int** value = 1;  
 **byte**[] bytes = ByteBuffer.allocate(4).putInt(value).array();  
 DatagramSocket ds = **new** DatagramSocket();  
 InetAddress addr = InetAddress.getByName(**"127.0.0.1"**);  
 DatagramPacket datagramPacket = **new** DatagramPacket(bytes, bytes.length, addr, 9876);  
 ds.send(datagramPacket);  
 **int** type = 1;  
 **if**(mas.get(0) **instanceof** Picture)  
 type = 0;  
 bytes = ByteBuffer.allocate(4).putInt(type).array();  
 datagramPacket = **new** DatagramPacket(bytes, bytes.length, addr, 9876);  
 ds.send(datagramPacket);  
  
 bytes = ByteBuffer.allocate(4).putInt(mas.get(0).x).array();  
 datagramPacket = **new** DatagramPacket(bytes, bytes.length, addr, 9876);  
 ds.send(datagramPacket);  
  
 bytes = ByteBuffer.allocate(4).putInt(mas.get(0).y).array();  
 datagramPacket = **new** DatagramPacket(bytes, bytes.length, addr, 9876);  
 ds.send(datagramPacket);  
  
 ds.close();  
 mas.remove(0);  
 wind.setMas(mas);  
 }**catch** (Exception ee)  
 {  
 ee.printStackTrace();  
 }  
}  
*// Получение размера вектора***if**(e.getKeyCode() == KeyCode.NUMPAD7.getCode())  
{  
 **try** {  
 DatagramSocket gss = **new** DatagramSocket(2700);  
 DatagramPacket getPack = **new** DatagramPacket(**new byte**[1024], 1024);  
 **int** value = 3;  
 **byte**[] bytes = ByteBuffer.allocate(4).putInt(value).array();  
 DatagramSocket dss = **new** DatagramSocket();  
 InetAddress addr = InetAddress.getByName(**"127.0.0.1"**);  
 DatagramPacket datagramPacket = **new** DatagramPacket(bytes, bytes.length, addr, 9876);  
 dss.send(datagramPacket);  
 gss.receive(getPack);  
 **int** num = ByteBuffer.wrap(getPack.getData()).getInt();  
 System.out.println(num);  
 gss.close();  
 dss.close();  
 }**catch** (Exception e1)  
 {  
 e1.printStackTrace();  
 }  
  
}  
*//Очистить вектор на сервере***if**(e.getKeyCode() == KeyCode.NUMPAD6.getCode())  
{  
 **try** {  
 DatagramPacket getPack = **new** DatagramPacket(**new byte**[1024], 1024);  
 **int** value = 4;  
 **byte**[] bytes = ByteBuffer.allocate(4).putInt(value).array();  
 DatagramSocket dss = **new** DatagramSocket();  
 InetAddress addr = InetAddress.getByName(**"127.0.0.1"**);  
 DatagramPacket datagramPacket = **new** DatagramPacket(bytes, bytes.length, addr, 9876);  
 dss.send(datagramPacket);  
 }**catch** (Exception s)  
 {  
 s.printStackTrace();  
 }  
  
}  
*//получение объектов с сервера***if**(e.getKeyCode() == KeyCode.NUMPAD9.getCode())  
{  
 **try** {  
  
 DatagramSocket gs = **new** DatagramSocket(2705);  
 DatagramPacket getPack = **new** DatagramPacket(**new byte**[1024], 1024);  
 **int** value = 2;  
 **byte**[] bytes = ByteBuffer.allocate(4).putInt(value).array();  
 DatagramSocket ds = **new** DatagramSocket();  
 InetAddress addr = InetAddress.getByName(**"127.0.0.1"**);  
 DatagramPacket datagramPacket = **new** DatagramPacket(bytes, bytes.length, addr, 9876);  
 ds.send(datagramPacket);  
 gs.receive(getPack);  
 **int** num = ByteBuffer.wrap(getPack.getData()).getInt();  
 Object obj;  
 **while**(num-->0) {  
 gs.receive(getPack);  
 **int** type = ByteBuffer.wrap(getPack.getData()).getInt();  
 gs.receive(getPack);  
 **int** x = ByteBuffer.wrap(getPack.getData()).getInt();  
 gs.receive(getPack);  
 **int** y = ByteBuffer.wrap(getPack.getData()).getInt();  
 **if** (type == 0)  
 obj = **new** Picture(x, y);  
 **else if** (type == 1)  
 obj = **new** SinFunc(x, y);  
 **else** obj = **new** Picture();  
 mas.add(obj);  
 }  
 wind.setMas(mas);  
 gs.close();  
 ds.close();  
  
 }**catch** (Exception exx)  
 {  
 exx.printStackTrace();  
 }  
}

**Выводы:**

Входе лабораторной работы мы познакомились с передачей данных по UDP-протоколу.