Java2 Lab 14
(TCP/UDP)

[Experimental Objective]

掌握TCP和UDP网络编程的基本方法

【一、UDP】

DatagramPacket:

UDP 通信发送和接收的数据需要使用 DatagramPacket 类，该类的实例对象就相当于一个集装箱，用于封装 UDP 通信中发送或者接收的数据。

Api:

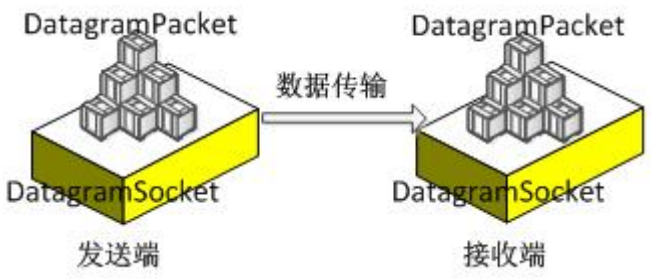
	DatagramPacket(byte[] buf, int length, InetAddress address, int port) 构造数据报包，用来将长度为 length 的包发送到指定主机上的指定端口号。
InetAddress	**getAddress**() 返回某台机器的 IP 地址，此数据报将要发往该机器或者是从该机器接收到的。
int	**getPort**() 返回某台远程主机的端口号，此数据报将要发往该主机或者是从该主机接收到的。
byte[]	**getData**() 返回数据缓冲区。
int	**getLength**() 返回将要发送或接收到的数据的长度。

DatagramSocket

DatagramSocket类的作用就类似于码头，使用这个类的实例对象就可以发送和接收DatagramPacket数据包

	DatagramSocket() 构造数据报套接字并将其绑定到本地主机上任何可用的端口。

void	**receive**(DatagramPacket p) 从此套接字接收数据报包。
void	**send**(DatagramPacket p) 从此套接字发送数据报包。



11849062@mail.sustc.edu.cn

[案例1 UDP]

UDP使用DatagramSocket进行简单的数据传输。
在本地写发送端和接收端程序，实现字符串的传输。

发送端：

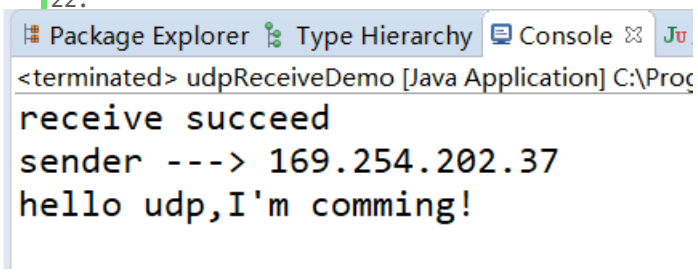
步骤为：创建发送端 Socket 对象
创建数据并打包
发送数据
释放资源

```
1.  public class udpSendDemo {
2.      public static void main(String[] args) throws IOException  {
3.          DatagramSocket ds = new DatagramSocket();
4.          String s = "hello udp,I'm comming!";
5.          byte[] bys = s.getBytes();
6.          int length = bys.length;
7.          InetAddress address = InetAddress.getLocalHost();
8.          int port = 8888;
9.          DatagramPacket dp = new DatagramPacket(bys,length,address,port);
10.         ds.send(dp);
11.         ds.close();
12.     }
13. }
```

接收端:

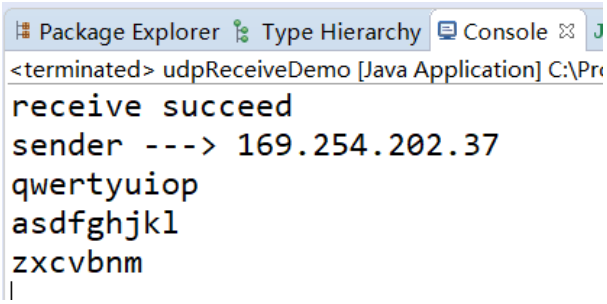
步骤为：创建接收端 Socket 对象
接收数据
解析数据
输出数据
释放资源

```
1.  public class  udpReceiveDemo {
2.      public static void  main（String [] args）  throws  IOException {
3.
4.          DatagramSocket ds =  new  DatagramSocket（8888 ）;
5.
6.          byte [] bys =  new byte [ 1024 ];
7.          DatagramPacket dp =  new  DatagramPacket（bys，bys.length）;
8.
9.          ds.receive（DP）;
10.         System.out.println（“接收成功” ）;
11.
12.         InetAddress address = dp.getAddress（）;
13.         byte [] data = dp.getData（）;
14.         int  length = dp.getLength（）;
15.
16.         System.out.println（“sender --->”  + address.getHostAddress（））;
17.         System.out.println（new  String（bys，0 ，length））;
18.
19.         ds.close（）;
20.
21.     }
22.
```



[练习1 UDP]

请在案例一的基础上，结合之前学过的I/O流，使用UDP协议，在发送端上传一个txt文档（message.txt），在接收端将文档内容读出。



【二、TCP】

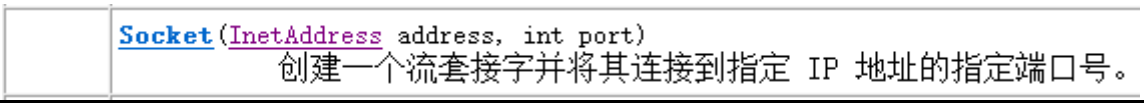
UDP 中只有发送端和接收端，不区分客户端与服务器端，计算机之间可以任意地发送数据。

而 TCP 通信是严格区分客户端与服务器端的，在通信时，必须先由客户端去连接服务器端才能实现通信，服务器端不可以主动连接客户端，并且服务器端程序需要事先启动，等待客户端的连接。

在 JDK 中提供了两个类用于实现 TCP 程序，一个是 ServerSocket 类，用于表示服务器端，一个是 Socket 类，用于表示客户端。

ServerSocket:

Socket:



方法声明	功能描述
int getPort()	该方法返回一个 int 类型对象，该对象是 Socket 对象与服务器端连接的端口号
InetAddress getLocalAddress()	该方法用于获取 Socket 对象绑定的本地 IP 地址，并将 IP 地址封装成 InetAddress 类型的对象返回
void close()	该方法用于关闭 Socket 连接，结束本次通信。在关闭 socket 之前，应将与 socket 相关的所有的输入/输出流全部关闭，这是因为一个良好的程序应该在执行完毕时释放所有的资源

InputStream getInputStream()	该方法返回一个 InputStream 类型的输入流对象，如果该对象是由服务器端的 Socket 返回，就用于读取客户端发送的数据，反之，用于读取服务器端发送的数据
OutputStream getOutputStream()	该方法返回一个 OutputStream 类型的输出流对象，如果该对象是由服务器端的 Socket 返回，就用于向客户端发送数据，反之，用于向服务器端发送数据

[案例2 TCP]

使用TCP协议，在客户端发送一行字符串，服务端进行接收并显示

服务端：

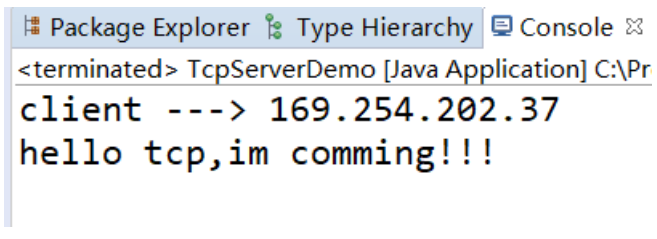
```
步骤：    使用 TCP 协议接收数据
          创建接收端 Socket 对象
          监听（阻塞）
          获取输入流对象
          获取数据
          输出数据
          释放资源
```

```
1.  public class TcpServerDemo {
2.      public static void main(String[] args) throws IOException {
3.
4.          ServerSocket ss = new ServerSocket(10086);
5.          Socket s = ss.accept();
6.          InputStream is = s.getInputStream();
7.
8.          byte[] bys = new byte[1024];
9.          int len;
10.         len = is.read(bys);
11.
12.         InetAddress address = s.getInetAddress();
13.         System.out.println("client ---> " + address.getHostAddress());
14.         System.out.println(new String(bys, 0, len));
15.
16.         s.close();
17.         // ss.close();  //服务端实际一般不关
18.     }
19. }
```

客户端：

```
步骤：    使用 TCP 协议发送数据
          创建发送端 Socket 对象（创建连接）
          获取输出流对象
          发送数据
          释放资源
```

效果如下：




```
1.  public class TcpClientDemo {
2.      public static void main(String[] args) throws IOException {
3.          Socket s = new Socket(InetAddress.getLocalHost(),10086);
4.          OutputStream os = s.getOutputStream();
```

```
5.          String str = "hello tcp,im comming!!!";
6.          os.write(str.getBytes());
7.          //释放资源
8.          //os.close();
9.          s.close();
10.     }
11. }
```

[练习2 TCP]

请在案例2的基础上，模拟用户登陆流程。

将用户名和密码封装到一个User类中,提供对应的构造方法和getter/setter方法
新建一个UserDB类里面定义一个集合,在集合中添加以下User对象：
new User("zhangsan","123456");
new User("lisi","654321");

功能提示：
客户端:
1.提示用户输入用户名和密码,将用户输入的用户名和密码发送给服务端
2.接收服务端验证完用户名和密码的结果

服务端:
服务端将客户端发送过来的用户名密码封装成 User 对象
集合中如果包括这个 User 对象,向客户端写入” 登录成功”
否则向客户端写入”登录失败”