Java2 Lab 14
(TCP/UDP)

[Experimental Objective]

Master the basic methods of TCP and UDP network programming

【UDP】

DatagramPacket:

The data sent and received by UDP communication needs to use the DatagramPacket class. The instance object of this class is equivalent to a container for encapsulating data sent or received in UDP communication.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。

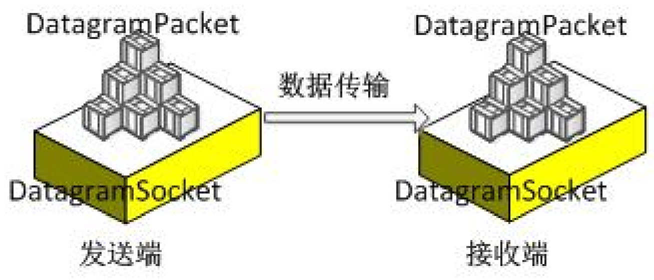
InetAddress	**getAddress**() 返回某台机器的 IP 地址，此数据报将要发往该机器或者是从该机器接收到的。
int	**getPort**() 返回某台远程主机的端口号，此数据报将要发往该主机或者是从该主机接收到的。
byte[]	**getData**() 返回数据缓冲区。
int	**getLength**() 返回将要发送或接收到的数据的长度。

DatagramSocket

The DatagramSocket class acts like a dock, and can send and receive DatagramPacket packets using instance objects of this class.

	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。



[案例1 UDP]

UDP uses DatagramSocket for simple data transfer.
Write the sender and receiver programs locally to implement string transmission.
Sending end:

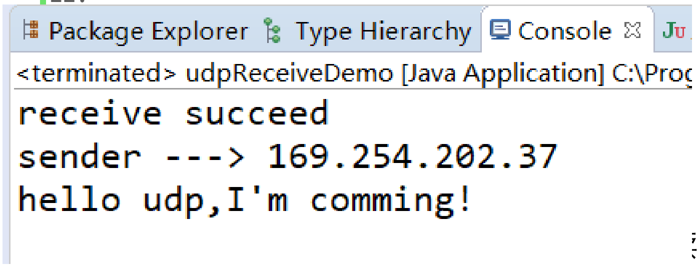
```
    The steps are:  Create a sender Socket object
                   Create data and package
                    send data
                   Release resources
```

```java
public class udpSendDemo {
    public static void main(String[] args) throws IOException  {
        DatagramSocket ds = new DatagramSocket();
        String s = "hello udp,I'm comming!";
        byte[] bys = s.getBytes();
        int length = bys.length;
        InetAddress address = InetAddress.getLocalHost();
        int port = 8888;
        DatagramPacket dp = new DatagramPacket(bys,length,address,port);
        ds.send(dp);
        ds.close();
    }
}
```

Receiving end:：

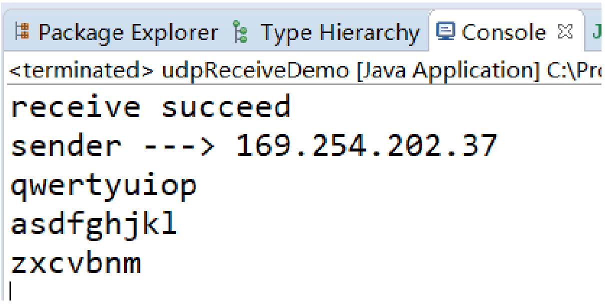
```
The steps are as follows: Create a Receive Socket object
        Receive data
        Analytical data
        Output Data
         Release resources
```

```java
public class  udpReceiveDemo {
    public static void  main（String [] args）  throws  IOException {

        DatagramSocket ds =  new  DatagramSocket（8888 ）;

        byte [] bys =  new byte [ 1024 ];
        DatagramPacket dp =  new  DatagramPacket（bys, bys.length）;

        ds.receive（DP）;
        System.out.println（“接收成功” ）;

        InetAddress address = dp.getAddress（）;
        byte [] data = dp.getData（）;
        int  length = dp.getLength（）;

        System.out.println（“sender --->”  + address.getHostAddress（））;
        System.out.println（new  String（bys, 0 , length））;

        ds.close（）;

    }

```



```
<terminated> udpReceiveDemo [Java Application] C:\Prog
receive succeed
sender ---> 169.254.202.37
hello udp,I'm comming!
```

使用UDP协议，在发送端上传一
出。



Package Explorer | Type Hierarchy | Console | J

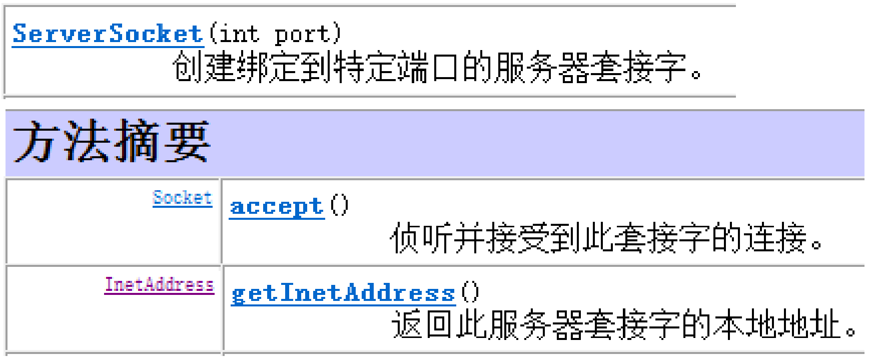
```
<terminated> udpReceiveDemo [Java Application] C:\Pr
receive succeed
sender ---> 169.254.202.37
qwertyuiop
asdfghjkl
zxcvbnm
```

【TCP】

UDP only has a sender and a receiver. It does not distinguish between the client and the server. The computer can send data arbitrarily.

TCP communication strictly distinguishes between client and server. In communication, the client must connect to the server to communicate. The server cannot actively connect to the client, and the server program needs to be started before waiting for the client to connect. .

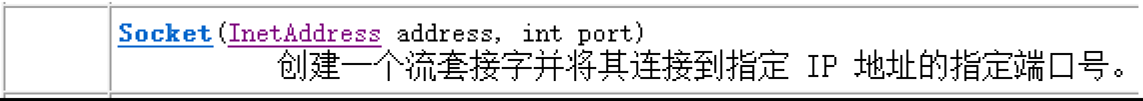
Two classes are provided in the JDK for implementing TCP programs. One is the ServerSocket class, which is used to represent the server side, and the other is the Socket class, which is used to represent the client.**ServerSocket:**



ServerSocket(int port)
创建绑定到特定端口的服务器套接字。

方法摘要	
Socket	accept() 侦听并接受到此套接字的连接。
InetAddress	getInetAddress() 返回此服务器套接字的本地地址。

Socket:



Socket(InetAddress address, int port)
创建一个流套接字并将其连接到指定 IP 地址的指定端口号。

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

[Case2 TCP]

Using the TCP protocol, a string is sent on the client, and the server receives and displays

Server:

```
Steps: Receive data using the TCP protocol
  Create a Receive Socket object
  Monitor (block)
  Get the input stream object
         retrieve data
         Output Data
  Release resources
```

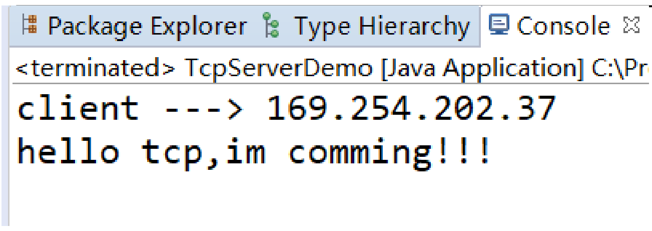
```java
public class TcpServerDemo {
    public static void main(String[] args) throws IOException {

        ServerSocket ss = new ServerSocket(10086);
        Socket s = ss.accept();
        InputStream is = s.getInputStream();

        byte[] bys = new byte[1024];
        int len;
        len = is.read(bys);

        InetAddress address = s.getInetAddress();
        System.out.println("client ---> " + address.getHostAddress());
        System.out.println(new String(bys, 0, len));

        s.close();
        // ss.close();  //服务端实际一般不关
    }
}
```

client:

```
Steps: Send data using TCP protocol
Create a sender Socket object (create a connection)
Get the output stream object
send data
Release resources
```

```java
public class TcpClientDemo {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket(InetAddress.getLocalHost(),10086);
        OutputStream os = s.getOutputStream();
        String str = "hello tcp,im comming!!!";
        os.write(str.getBytes());
        //释放资源
        //os.close();
        s.close();
    }
}
```

result:



```
<terminated> TcpServerDemo [Java Application] C:\Pr
client ---> 169.254.202.37
hello tcp,im comming!!!
```

[Exercise2 TCP]

Please simulate the user login process on the basis of Case 2.

Encapsulate the username and password into a User class, providing the corresponding constructor and getter/setter methods

Create a new UserDB class to define a collection, add the following User object to the collection:

new User("zhangsan","123456");

new User("lisi","654321");

Feature Tip:

Client:

1. Prompt the user to enter the user name and password, and send the user name and password entered by the user to the server.

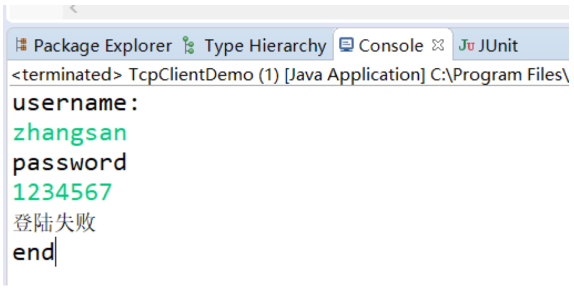
2. Receive the result of the server verifying the username and password.

Server:

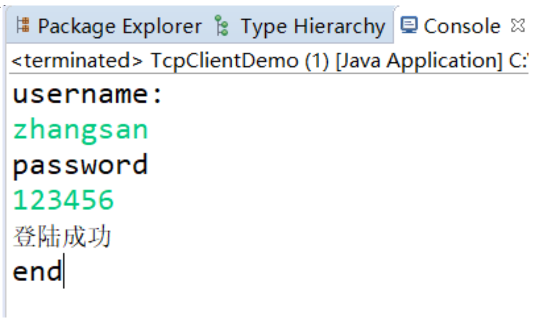
The server encapsulates the username and password sent by the client into a User object.

If the User object is included in the collection, write "Login Successful" to the client.

Otherwise write "login failed" to the client



```
<terminated> TcpClientDemo (1) [Java Application] C:\Program Files\
username:
zhangsan
password
1234567
登陆失败
end
```



```
<terminated> TcpClientDemo (1) [Java Application] C:
username:
zhangsan
password
123456
登陆成功
end
```