

深圳大学实验报告

课程名称： 计算机网络

实验项目名称： Socket 网络编程

学院： 计算机与软件学院

专业： 软件工程

指导教师： 姚俊梅

报告人： 郑彦薇 学号： 2020151022 班级： 软件工程 01 班

实验时间： 2023 年 3 月 14 日--2023 年 4 月 9 日

实验报告提交时间： 2023/04/02

教务处制

实验目的：

1. 学习网络编程基本概念、InetAddress 的应用、URL 的应用、URLConnection 的应用
2. 了解简单网路应用的编程思路
3. 了解网络编程相关的一些库
4. 掌握 Socket 的 TCP 通信、Socket 的 UDP 通信
5. 掌握 Socket、ServerSocket 类和 DatagramPacket、DatagramSocket 类的使用

实验环境：

具有 Internet 连接的主机、java 编程语言、python 编程语言。

实验内容：

1. 使用 InetAddress 类的方法获取本地机的名称和 ip 地址。
2. 使用 InetAddress 类的方法获取网站 www.csdn.net 的 IP 地址，如果存在多个 IP 地址，要求全部返回。
3. 使用 URL 类下载深圳大学首页 <https://www.szu.edu.cn>，并统计下载得到的网页文件的大小。
4. 利用 Socket 类和 ServerSocket 类编写一个 C/S 程序，实现 C/S 通信。
5. 利用 Socket 类和 ServerSocket 类，编写一个 C/S 程序，实现网络文件传输。
6. 编写一个数据报通信程序，实现一对一的简单聊天功能。
7. 编写一个基于客户/服务器的网络聊天程序，实现多个用户的群聊。

实验步骤与结果：

一、使用 InetAddress 类的方法获取本地机的名称和 ip 地址

1. 编程思路：

在 java 中，使用 InetAddress 类中的 getLocalHost()方法就可以获得本机地址信息，包括本机 ip 地址和本地主机名。再分别使用 getHostAddress()方法和 getHostName()方法获得这两个值即可。编写程序及注释如下图所示：

```
public class LocalAdd {  
    public static void main(String[] args) throws Exception{  
        //获取本机地址信息  
        InetAddress local = InetAddress.getLocalHost();  
        //获取本地 ip 地址  
        String LocalAddress = local.getHostAddress();  
        //获取本地主机名  
        String LocalName = local.getHostName();  
        //输出信息  
        System.out.println("本地主机 IP:"+LocalAddress);  
        System.out.println("本地主机名:"+LocalName);  
    }  
}
```

2. 运行结果：

运行该 java 文件，可以得到运行结果如下：

```
本地主机IP:172.30.178.76
本地主机名:LAPTOP-QT7SKB71

进程已结束,退出代码0
```

二、使用 InetAddress 类的方法获取网站 www.csdn.net 的 IP 地址，如果存在多个 IP 地址，要求全部返回

1. 编程思路：

为获得指定域名的地址信息，我们可以使用 `getByName()` 方法。同样可以使用 `getHostAddress()` 方法获得地址信息中的 ip 地址，因为此处可能存在多个 ip 地址，我们可以定义一个列表用于存放所有的 ip 地址，再设置循环进行输出。编写程序及注释如下：

```
public class CSDNAdd {
    public static void main(String[] args) throws IOException{
        String csdnhost = "www.csdn.net";
        //获取指定域名地址信息
        InetAddress ip1 = InetAddress.getByName(csdnhost);

        //返回所有 ip 地址
        System.out.println("网站 www.csdn.net 的所有 IP 地址为:");
        InetAddress[] ip2 = InetAddress.getAllByName(csdnhost);
        for(int i=0;i<ip2.length;i++){
            System.out.println(i+1+":"+ip2[i].getHostAddress());
        }
    }
}
```

2. 运行结果：

运行该文件，可以得到运行结果如下：

```
网站www.csdn.net的所有IP地址为：
1:220.185.184.18

进程已结束,退出代码0
```

三、使用 URL 类下载深圳大学首页 <https://www.szu.edu.cn>，并统计下载得到的网页文件的大小

1. 编程思路：

在 URL 类中，可以使用 `openStream()` 方法获得 `InputStream` 对象，得到 `InputStream` 对象就可以对网站上的资源进行下载。设置文件名对下载文件进行存放，再对其大小进行统计输出。编写程序及注释如下：

```
public class TestUrl {
    public static void main(String[] args) throws Exception{
        URL url = new URL("https://www.baidu.com");
```

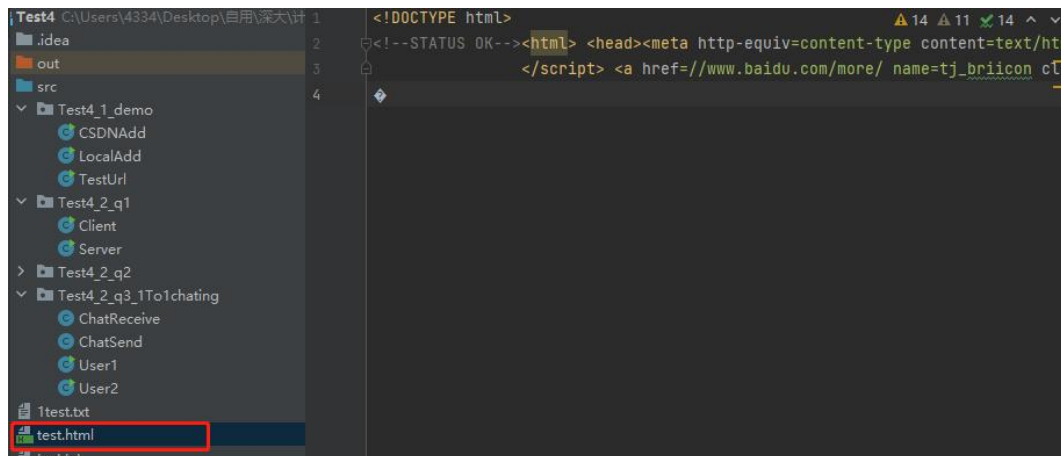
```

        InputStream in = url.openStream();//生成输入流
        FileOutputStream fout = new FileOutputStream(new File("test.html"));
        int a = 0;
        while(a>-1){
            a = in.read();//读入
            fout.write(a);//写出
        }
        // 统计文件的大小
        File file = new File("test.html");
        System.out.println("文件大小为:" + new DecimalFormat("#.00").format(file.length() /
1024.00)
                + "k");
    }
}

```

2. 运行结果:

运行该程序, 可以得到一个 test.html 文件如下:



统计得文件大小为:

文件大小为: 2.39k

进程已结束, 退出代码0

四、利用 Socket 类和 ServerSocket 类编写一个 C/S 程序, 实现 C/S 通信

1. 服务器端 server 编程思路:

在服务器端中, 首先使用 ServerSocket 类创建一个 server 对象, 并设置端口号 (如 6789), 使用 accept() 方法等待与客户端进行连接。

接着使用 BufferedReader 获取客户端得数据, 通过 ServerSocket 中的 getInputStream 方法将输入流与 BufferedReader 连接。

设置 while 循环, 不断接收数据并存放到 string 类的 str 对象中, 并创建一个 DataOutputStream 类, 将数据发送给客户端, 建立输出流。

在循环中, 对 str 接收的数据进行判断, 如果是 time 则按一定格式发送当前服务器的时间; 如果是 exit 则向客户端发送 “bye\n” 字段, 在输出一个 bye 之后退出循环并终止服务器。

编写程序及注释如下：

```
public class Server {
    public static void main(String[] args) {
        try {
            ServerSocket server = new ServerSocket(6789);
            System.out.println("服务器启动完毕");
            Socket socket=server.accept();//等待连入请求，并创建客户连接
            System.out.println("创建客户连接");
            BufferedReader reader;
            //创建与套接字连接的输入流，接收信息使用 BufferedReader 读取
            reader=new BufferedReader(new InputStreamReader(socket.getInputStream()));

            while (true) {
                String str = reader.readLine();
                DataOutputStream ToClient=new
DataOutputStream(socket.getOutputStream());
                //如果接收到“exit”，打印“bye”，退出
                if (str.equals("Exit")) {
                    ToClient.writeBytes("Bye\n");
                    System.out.println("Bye");
                    break;
                }
                //如果收到"Time"，将服务器端当前时间返回给客户端
                if(str.equals("Time")) {
                    Calendar calendar=Calendar.getInstance();
                    //设置日期格式
                    SimpleDateFormat dateformat= new
SimpleDateFormat("yyyy-MM-dd :hh:mm:ss ");
                    System.out.println("服务器当前的时间
为:"+dateformat.format(calendar.getTime()));
                    String str1="Current Client Time
is:"+dateformat.format(calendar.getTime());
                    ToClient.writeBytes(str1+"\n");
                }
            }
            reader.close();
            socket.close();
            server.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2. 客户端 client 编程思路：

在客户端中使用 `Socket()` 类创建一个 `ClientSocket` 对象，并设置端口号（与上述服务器端设置的端口号要相同）。接着创建一个 `DataOutputStream` 与 `ClientSocket` 的输出流进行连接，用于传送数据给服务器端。再创建一个 `BufferedReader` 用于接收服务器端的数据，同样与 `ClientSocket` 对象建立连接。

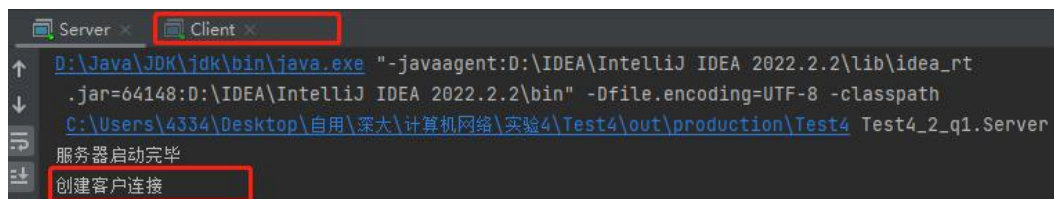
接下来设置一个循环，将用户输入的命令发送给服务器端。用户可以不断输入 `time` 命令并从服务器端中获得结果。当输入为 `exit` 时，退出循环并终止该程序。

编写程序及注释如下：

```
public class Client {  
    static String hostname = new String("localhost");  
    public static void main(String[] args) {  
        try {  
            //client 端创建一个 socket，与服务器连接  
            Socket ClientSocket = new Socket(hostname, 6789);  
  
            //创建与 socket 连接的输出流，定义为 out  
            DataOutputStream out = new DataOutputStream(ClientSocket.getOutputStream());  
            BufferedReader reader;  
            reader=new BufferedReader(new  
InputStreamReader(ClientSocket.getInputStream()));  
  
            String sentence;  
            String modifiedSentence;  
            while(true){ //可以多次输入 Time，直到输入 Exit 才结束  
                BufferedReader inFromUser = new BufferedReader(new  
InputStreamReader(System.in));  
                sentence = inFromUser.readLine();  
                out.writeBytes(sentence+'\n');  
                modifiedSentence = reader.readLine();  
                System.out.println("From Server:" + modifiedSentence);  
                if(sentence.equals("Exit")){  
                    break;  
                }  
            }  
            ClientSocket.close();  
            out.close();  
            reader.close();  
        } catch (UnknownHostException e) {  
            System.out.println("UnknownHost");  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

3. 运行结果：

运行时，先运行服务器端的程序，再运行客户端的程序。

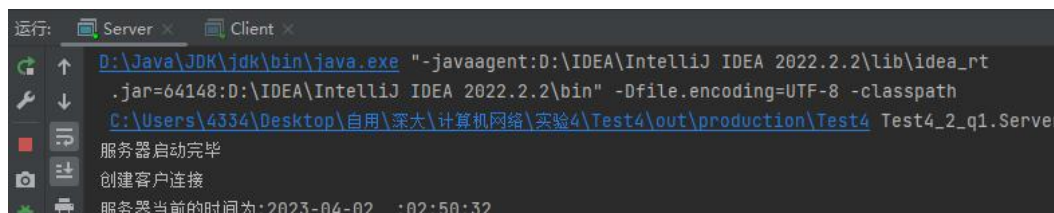


```
Server x Client x
D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\lib\idea_rt
.jar=64148:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -classpath
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\Test4 Test4_2_q1.Server
服务器启动完毕
创建客户连接
```

接着在客户端的运行台中输入“Time”命令，可以分别在客户端运行台和服务器端运行台中打印出相关信息。



```
运行: Server x Client x
D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\lib\idea_rt
.jar=64154:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -classpath
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\Test4 Test4_2_q1.Client
Time
From Server:Current Client Time is:2023-04-02 :02:50:32
```




```
运行: Server x Client x
D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\lib\idea_rt
.jar=64154:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -classpath
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\Test4 Test4_2_q1.Server
服务器启动完毕
创建客户连接
服务器当前的时间为:2023-04-02 :02:50:32
```

输入“Exit”，则输出 Bye 并终止程序。



```
运行: Server x Client x
D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\lib\idea_rt
.jar=64154:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -classpath
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\Test4 Test4_2_q1.Client
Time
From Server:Current Client Time is:2023-04-02 :02:50:32
Exit
From Server:Bye
进程已结束, 退出代码0
```



```
运行: Server x Client x
D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\lib\idea_rt
.jar=64148:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -classpath
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\Test4 Test4_2_q1.Server
服务器启动完毕
创建客户连接
服务器当前的时间为:2023-04-02 :02:50:32
Bye
进程已结束, 退出代码0
```

五、利用 Socket 类和 ServerSocket 类，编写一个 C/S 程序，实现网络文件传输

1. 服务器端 FileServer 编程思路：

通过线程处理执行命令实现网络文件的传输。首先继承线程接口 Runnable，并设置成员对象包括静态类型的端口号和 Socket 类的对象 s。

在构造函数 FileServer 中，使用父类 Runnable 的构造函数，对 Socket 类对象 s 进行复制。定义 server 方法，创建一个 Socket 对象，并设置端口号。进入一个循环，与客户端进行

连接并存放成员对象 s 中，输出线程数量，再调用线程并开启。

相关代码及注释如下：

```
private static final int MONITORPORT = 12345;
private Socket s;
public FileServer(Socket s) {
    super();
    this.s = s;
}
public static void server()
{
    try {
        ServerSocket ss = new ServerSocket(MONITORPORT);
        int i=0;
        while(true)
        {
            i++;
            Socket s = ss.accept(); //与客户端进行连接并存放成员对象 s 中
            System.out.println("服务器的线程"+i+"启动,与客户端 1 连接成功");
            new Thread(new FileServer(s)).start();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    FileServer.server();
}
```

为实现文件数据的传输，还需定义 Run 方法作为线程运行时需要执行的程序。在该方法中将文件里得数据传输到客户端的文件中。

Run 方法的代码及注释如下：

```
public void run() {
    byte[] buf = new byte[100];
    OutputStream os=null;
    FileInputStream fins=null;
    try {
        os = s.getOutputStream();
        String fileName = "test.txt";

        System.out.println("要传输的文件为: "+fileName);
        os.write(fileName.getBytes());
        System.out.println("开始传输文件");
        fins = new FileInputStream(fileName);
        int data;
```



```

        while(-1!=(data = fins.read()))
        {
            os.write(data);
        }
        System.out.println("文件传输结束");
    } catch (IOException e) {
        e.printStackTrace();
    } finally
    {
        try {
            if(fins!=null) fins.close();
            if(os!=null) os.close();
            this.s.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}

```

2. 客户端 FileClient 编程思路:

首先设置相对应的 IP 地址、本机端口号和服务器端口号。与服务器进行连接，创建输入流和输出流分别与服务器端和文件进行连接，将服务器接收到的数据传输给文件。

具体编程及注释如下:

```

public class FileClient {
    private static final String SERVERIP = "127.0.0.1";
    private static final int SERVERPORT = 12345;
    private static final int CLIENTPORT = 54321;

    public static void main(String[] args) {
        byte[] buf = new byte[100];
        Socket s = new Socket();
        try {
            s.connect(new InetSocketAddress(SERVERIP, SERVERPORT), CLIENTPORT);
            System.out.println("与服务器连接成功");
            InputStream is = s.getInputStream();
            int len = is.read(buf);
            String fileName = new String(buf, 0, len);
            System.out.println("接收到的文件为: " + fileName);
            System.out.println("保存为: " + "1" + fileName);
            //创建输出流，修改文件名
            FileOutputStream fos = new FileOutputStream("1" + fileName);
            int data;
            while (-1 != (data = is.read())) {
                fos.write(data);
            }
        }
    }
}

```

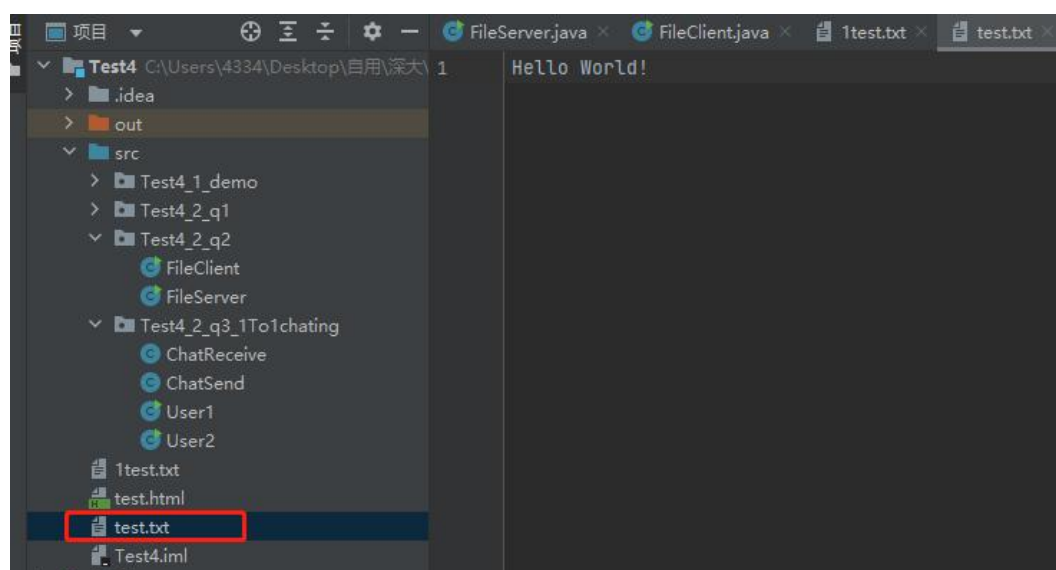
```

        is.close();
        s.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

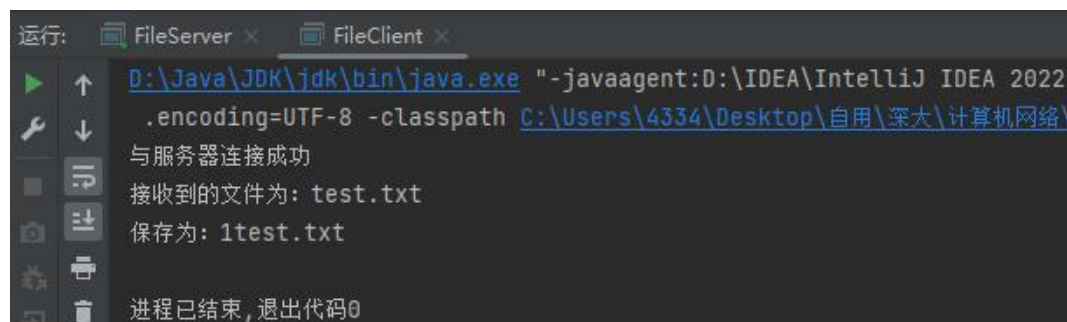
```

3. 运行结果:

在项目中创建一个 test 文本文件，用于传输。同样需要先运行服务器端程序，再运行客户端程序。



第一次运行 FileClient:

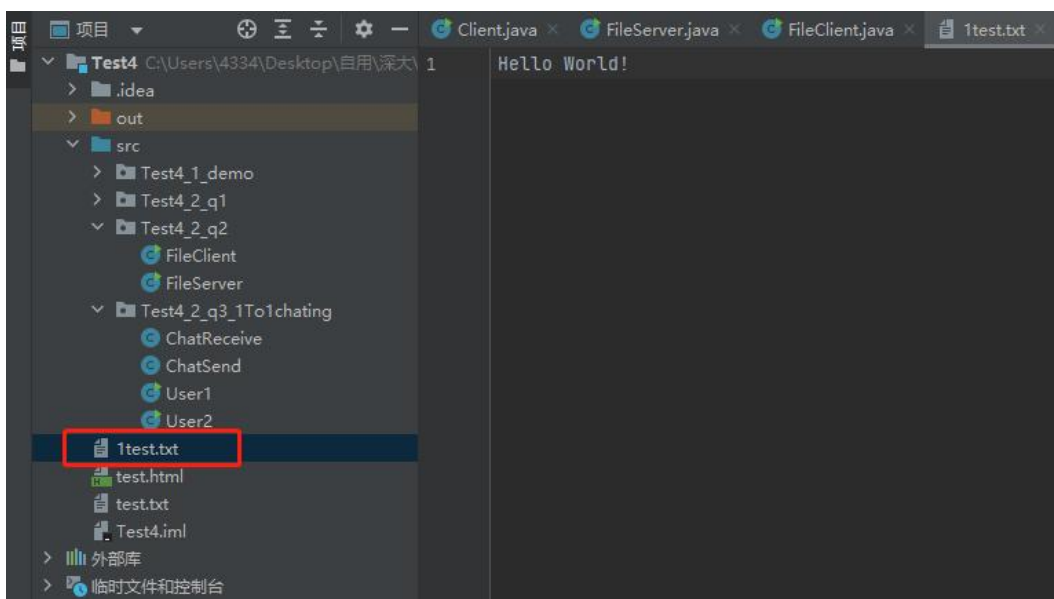


第二次运行 FileClient:



```
运行: FileServer x FileClient x
D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.
.encoding=UTF-8 -classpath C:\Users\4334\Desktop\自用\深大\计算机网络\实
服务器的线程1启动,与客户端1连接成功
要传输的文件为: test.txt
开始传输文件
文件传输结束
服务器的线程2启动,与客户端1连接成功
要传输的文件为: test.txt
开始传输文件
文件传输结束
```

得到的 1test 文本文件:



六、编写一个数据报通信程序，实现一对一的简单聊天功能

1. 发送端 ChatSend 编程思路:

程序的关键在于使用 DatagramSocket 指定端口并创建发送端，再使用 DatagramPacket 将数据进行封装，指定包裹的传输目的地址。另外，在包裹发送完成后，需要对资源进行释放。具体代码及相关注释如下：

```
public class ChatSend implements Runnable {
    private DatagramSocket client; // 发送端
    private BufferedReader br; // 输入流
    private String toIp; // 发送目标 ip
    private int toPort; // 发送目标端口 和接收端使用端口一致
    public ChatSend(int port, String toIp, int toPort) {
        this.toIp = toIp;
        this.toPort = toPort;
        try {
            client = new DatagramSocket(port);
            br = new BufferedReader(new InputStreamReader(System.in));
```

```

        } catch (SocketException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        while (true) {
            try {
                // 准备数据 一定转成字节数组
                String data = br.readLine();
                byte[] datas = data.getBytes();
                // 封装成 DatagramPacket 包裹, 需要指定地址
                DatagramPacket packet = new DatagramPacket(datas, 0, datas.length,
                    new InetSocketAddress(this.toIp, this.toPort));
                // 输出发送时间
                SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                System.out.println("发送 " + format.format(new Date()));
                // 发送包裹
                client.send(packet);
                if (data.equals("bye")) {
                    break;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        // 释放资源
        client.close();
    }
}

```

2. 接收端 ChatReceive 编程思路:

程序同样需要使用 DatagramSocket 指定端口并创建接收端, 以及使用 DatagramPacket 将容器封装成包裹。使用 DatagramSocket 中的 receive 方法阻塞式接收封装的包裹, 再使用 getData 方法和 getLength 方法对数据进行分析。最后同样需要释放资源。具体代码及相关注释如下:

```

public class ChatReceive implements Runnable {
    private DatagramSocket server; // 接收端
    private int port; // 接收端使用端口
    private String from; // 发送方标记
    public ChatReceive(int port, String from) {
        this.from = from;
        try {
            server = new DatagramSocket(port);

```

```

    } catch (SocketException e) {
        e.printStackTrace();
    }
}

@Override
public void run() {
    while (true) {
        try {
            // 准备容器封装成 DatagramPacket 包裹
            byte[] container = new byte[1024 * 60];
            DatagramPacket packet = new DatagramPacket(container, container.length);
            // 阻塞式接收包裹
            server.receive(packet);
            // 输出接收时间
            SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");

            System.out.println("接收 " + format.format(new Date()));
            // 分析数据
            byte[] datas = packet.getData();
            String msg = new String(datas, 0, packet.getLength());
            System.out.println(from + ":" + msg);
            if(msg.equals("bye")) {
                break;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // 释放资源
    server.close();
}
}

```

3. 创建两个用户对象 user1、user2:

user1、user2 中保证接收和发送目的地址对应。

```

public class User1 {
    public static void main(String[] args) throws SocketException{
        new Thread(new ChatSend( port: 9999, toIp: "localhost", toPort: 8888)).start();
        new Thread(new ChatReceive( port: 7777, from: "User2")).start();
    }
}

```

```

public class User2 {
    public static void main(String[] args) throws SocketException{
        new Thread(new ChatReceive( port: 8888, from: "User1")).start();
        new Thread(new ChatSend( port: 6666, toIp: "localhost", toPort: 7777)).start();
    }
}

```

4. 运行 user1.java、user2.java，模拟用户之间的对话，在控制台实现简单的一对一聊天：

运行: User1 x User2 x

```

D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\bin\idea_rt.jar=50348:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -c
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\T
Hello!Here is user1
发送 2023-04-02 16:00:19
接收 2023-04-02 16:00:31
User2:Hi!Here is user2

```

运行: User1 x User2 x

```

D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\bin\idea_rt.jar=50353:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -c
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\T
接收 2023-04-02 16:00:19
User1:Hello!Here is user1
Hi!Here is user2
发送 2023-04-02 16:00:31

```

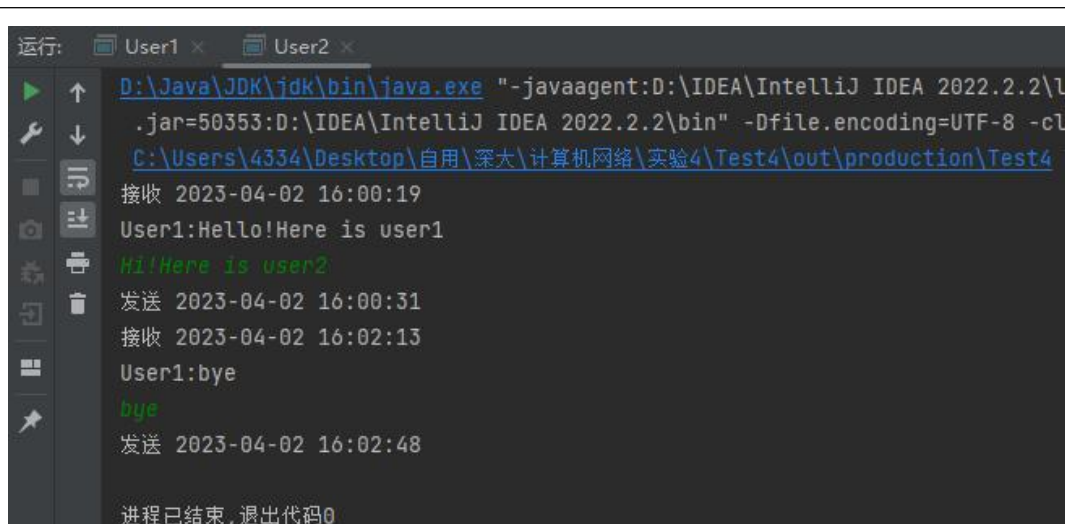
用户发送“bye”结束聊天：

运行: User1 x User2 x

```

D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\bin\idea_rt.jar=50348:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -c
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\T
Hello!Here is user1
发送 2023-04-02 16:00:19
接收 2023-04-02 16:00:31
User2:Hi!Here is user2
bye
发送 2023-04-02 16:02:13
接收 2023-04-02 16:02:48
User2:bye
进程已结束,退出代码0

```



```
运行: User1 x User2 x
D:\Java\JDK\jdk\bin\java.exe "-javaagent:D:\IDEA\IntelliJ IDEA 2022.2.2\l
.jar=50353:D:\IDEA\IntelliJ IDEA 2022.2.2\bin" -Dfile.encoding=UTF-8 -cl
C:\Users\4334\Desktop\自用\深大\计算机网络\实验4\Test4\out\production\Test4
接收 2023-04-02 16:00:19
User1:Hello!Here is user1
Hi!Here is user2
发送 2023-04-02 16:00:31
接收 2023-04-02 16:02:13
User1:bye
bye
发送 2023-04-02 16:02:48
进程已结束,退出代码0
```

七、编写一个基于客户/服务器的网络聊天程序，实现多个用户的群聊

在 python 环境下实现简易聊天室的创建。在多人聊天室中，消息的传送通过客户端→服务器端→客户端实现，采用多线程，一个负责发送，一个负责接收。程序之间数据的传送通过 encode()编码和 decode()解码实现。

1. 服务器端 server.py

服务器端用于处理用户的操作，包括登录、聊天以及退出。当用户在客户端输入名称登录时，服务器端对输入的名称进行判断，再输出指定的信息进行判断反馈。对于用户发送的信息，服务器端访问用户列表，将信息发送到其他用户界面。当用户执行退出时，同样访问用户列表对信息进行发送。具体编程及注释如下：

```
# 服务器地址
HOST='0.0.0.0'
PORT= 8888
ADDR = (HOST,PORT)
# 用字典储存用户
user = {} # 储存用户信息 {name:address} 格式

# 处理进入聊天室
def login(socketed, name, address):
    # 判断姓名是否已存在或者姓名为 admin
    if name in user or 'admin' in name:
        socketed.sendto('NO'.encode(), address)
    else:
        socketed.sendto('YES'.encode(), address)
        # 循环通知其他人
        msg = '%s had joined the chat!' % name
        for other in user:
            socketed.sendto(msg.encode(), user[other])
        user[name] = address # 增加该用户

# 处理聊天
```



```

def chat(socketed, name,content):
    msg = "[%s]: %s" % (name, content)
    for other in user:
        if other == name:
            continue # 除去自己
        socketed.sendto(msg.encode(), user[other])

# 处理退出
def exits(socketed, name):
    del user[name] # 删除用户
    msg = '%s has left the chat!' % name
    for other in user:
        socketed.sendto(msg.encode(), user[other])

# 处理客户端请求
def request(socketed):
    # 循环接收各种客户端请求
    while True:
        # 接收所有客户端请求
        data, address = socketed.recvfrom(1024)
        # 对数据结构进行简单解析
        tmp = data.decode().split(' ',2)

        if tmp[0] == 'LOGIN':
            # 进入函数
            login(socketed, tmp[1], address)
        elif tmp[0] == 'CHAT':
            # 聊天函数
            chat(socketed, tmp[1],tmp[2])
        elif tmp[0] == 'EXIT':
            # 退出函数
            exits(socketed, tmp[1])

# 程序启动函数
def main():
    # UDP 套接字
    socketed = socket(AF_INET, SOCK_DGRAM)
    socketed.bind(ADDR)
    # 创建子进程
    process=Process(target=request,args=(socketed,))
    process.daemon=True # 父进程退出子进程也退出
    process.start()
    while True:
        # 发送管理员消息

```



```

content = input("message from admin: ")
# 服务端退出
if content == "EXIT":
    break
msg = "CHAT message from admin: " + content
# 从父进程发送给子进程
socketed.sendto(msg.encode(), ADDR)

```

2. 客户端 client.py

客户端进行用户的登录、聊天和退出。用户在客户端输入相应信息，由服务器端进行处理。具体编程及注释如下：

```

# 服务器地址
ADDR = ('127.0.0.1', 8888)

# 处理登录
def login(socketed):
    while True:
        # 进入聊天室
        name = input('Please input your nickname: ')

        # 发送姓名给服务端
        msg = 'LOGIN' + name
        # encode() 函数将目标字符串编写为目标二进制数据 bytes 类型，编码过程
        socketed.sendto(msg.encode(), ADDR)
        # 接收服务端结果反馈
        data, address = socketed.recvfrom(1024)
        # decode() 函数将编码后的字符串转换为原始字符串，解码过程
        if data.decode() == 'YES':
            print('Welcome %s ,input<bye> to quit the chat!' % (name,))
            return name
        else:
            print('%s has exited' % (name,))

# 子进程，循环接收消息
def chat(socketed, name):
    while True:
        data, address = socketed.recvfrom(1024 * 10)
        # 美化打印内容
        msg = "\n" + data.decode() + "\n[%s]: " % name
        print(msg, end="") # end="" 不换行

# 父进程，循环发送消息
def receive(socketed, name):
    while True:

```

```

# try 处理异常退出
try:
    news = input('[%s]: ' % (name,))
except KeyboardInterrupt:
    news = 'EXIT'

if news == 'bye':
    msg = 'EXIT ' + name
    socketed.sendto(msg.encode(), ADDR)
    sys.exit('%s has left the chat!' % (name,))

msg = 'CHAT %s %s' % (name, news)
# 发送编码结果到指定位置
socketed.sendto(msg.encode(), ADDR)

# 程序启动函数
def main():
    socketed = socket(AF_INET, SOCK_DGRAM)
    name = login(socketed) # 进入聊天室

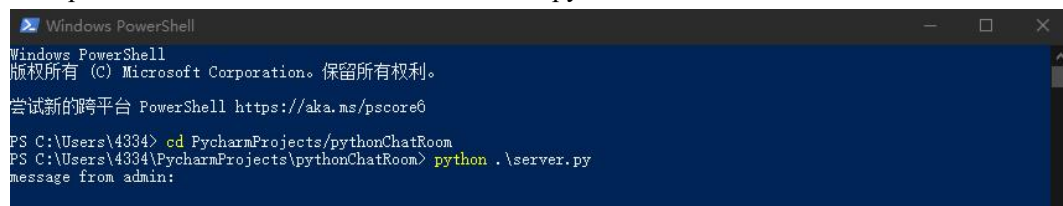
    # 创建子进程 用于接收消息
    process = Process(target=chat, args=(socketed, name))
    process.daemon = True # 父进程退出子进程也退出
    process.start()

    # 父进程发送消息
    receive(socketed, name)
    socketed.close()
    process.join()

```

3. 运行程序，模拟多人同时聊天

打开 powershell，进入项目，首先运行 server.py 程序。



```

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\4334> cd PycharmProjects\pythonChatRoom
PS C:\Users\4334\PycharmProjects\pythonChatRoom> python .\server.py
message from admin:

```

打开 3 个 powershell 界面，进入项目，都运行 client.py 程序，模拟 3 个用户的群聊。

```
Windows PowerShell
PS C:\Users\4334\PchamProjects\pythonChatRoom> python .\client.py
Please input your nickname: Leonard
Welcome Leonard ,input('quit' to quit the chat!
[Leonard]:
Sheldon has joined the chat!
[Leonard]:
[Sheldon]: Hello?
[Leonard]: Hello, sheldon! I'm Leonard
[Leonard]:
[Sheldon]: Hi, Leonard. Only we two in this chatroom?
[Leonard]: uh... it seems so...
[Sheldon]: alright...
[Leonard]:
penny has joined the chat!
[Leonard]: oh! We have a new friend!
[Sheldon]: Welcome!
[Leonard]:
[penny]: Hello, Leonard!
[Leonard]:
[penny]: Hello, Sheldon
[Leonard]:
[penny]: I'm penny
[Leonard]: Hi, penny
[Leonard]:
[Sheldon]: That's insane! I think we can go out for a drink!
[Leonard]:
[penny]: I agree
[Leonard]: sanel
[Leonard]:
[Sheldon]: well, shall we meet at the Central Park?
[Leonard]: ok
[Sheldon]:
[penny]: ok
[Leonard]:
[penny]: see you later
[Leonard]:
[penny]: bye
[Leonard]:
[Sheldon]: bye
[Leonard]:
Sheldon has left the chat!
[Leonard]:
penny has left the chat!
[Leonard]: quit
Leonard has left the chat!
PS C:\Users\4334\PchamProjects\pythonChatRoom>

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/powershell
PS C:\Users\4334\PchamProjects\pythonChatRoom> python .\client.py
Please input your nickname: Sheldon
Welcome Sheldon ,input('quit' to quit the chat!
[Sheldon]:
[Leonard]: Hello, sheldon! I'm Leonard
[Sheldon]:
[Leonard]: Hi, Leonard. Only we two in this chatroom?
[Sheldon]:
[Leonard]: uh... it seems so...
[Sheldon]: alright...
[Sheldon]:
penny has joined the chat!
[Sheldon]:
[Leonard]: oh! We have a new friend!
[Sheldon]: Welcome!
[Sheldon]:
[penny]: Hello, Leonard!
[Sheldon]:
[penny]: Hello, Sheldon
[Leonard]:
[penny]: I'm penny
[Sheldon]:
[penny]: Hello, Leonard!
[Sheldon]:
[Leonard]: Hi, Sheldon
[Sheldon]:
[penny]: I'm penny
[Leonard]:
[Sheldon]: That's insane! I think we can go out for a drink!
[Sheldon]:
[penny]: I agree
[Sheldon]:
[penny]: sanel
[Sheldon]:
[Leonard]: well, shall we meet at the Central Park?
[Sheldon]:
[penny]: ok
[Sheldon]:
[penny]: see you later
[Sheldon]:
[penny]: bye
[Sheldon]:
Sheldon has left the chat!
[Sheldon]:
penny has left the chat!
PS C:\Users\4334\PchamProjects\pythonChatRoom>

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/powershell
PS C:\Users\4334\PchamProjects\pythonChatRoom> python .\client.py
Please input your nickname: penny
Welcome penny ,input('quit' to quit the chat!
[penny]:
[Leonard]: oh! We have a new friend!
[penny]:
[Sheldon]: Welcome!
[penny]:
[Sheldon]: Hello, Leonard!
[penny]:
[Sheldon]: I'm penny
[penny]:
[Leonard]: Hi, penny
[penny]:
[Sheldon]: That's insane! I think we can go out for a drink!
[penny]: I agree
[penny]:
[Leonard]: sanel
[penny]:
[Sheldon]: well, shall we meet at the Central Park?
[Leonard]:
[penny]: ok
[penny]:
[penny]: see you later
[penny]:
[penny]: bye
[Leonard]:
[Sheldon]: bye
[penny]:
[Sheldon]: bye
[penny]:
Sheldon has left the chat!
[penny]:
penny has left the chat!
PS C:\Users\4334\PchamProjects\pythonChatRoom>
```

实验小结：

本次实验包含了对 InetAddress 类、URL 类的学习和使用，以及基于 TCP 和 UDP 的网络传输和通信。

通过本次实验，学会了如何使用 Socket 网络编程工具对指定网站的 IP 地址进行获取、访问网页信息并进行下载。

另外，通过对 C/S 通信、网络文件传输、简易聊天和简易聊天室的实现，学习了服务器端和客户端程序之间的协作，更加了解在解决实际问题时两者分别应负责的工作。

指导教师批阅意见：

成绩评定：

指导教师签字：

年 月 日

备注：