搭建一个 web 服务器:

Step 1：建立一个服务器和客户端：

建立一个服务端所需：ServerSocketChannel 和 SocketChannel;

在 ServerSocketChannel serversocketchannel = ServerSocketChannel.open()；该步建立了一个 ServerSocketChannel 对象，但是没有绑定端口;

在 serversocketchannel.bind(new InetSocketAddress(5000))；中绑定了一个端口（以 InetSocketAddress 对象的方式绑定）。

之后再利用 SocketChannel.open(new InetSocketAddress("iP",port))的方法建立了一个 SocketChannel，和指定 ip 和 port 的客户端建立了连接。

之前建立的 serversocketchannel 可以通过 .accept()的方式监听，这样建立的一个新的 SocketChannel 就是可以和客户端通信的 SocketChannel;

总结：先开放一个服务端 SocketChannel（等待客户端请求），绑定端口，客户端再建立一个连接，最后用先前建立的 ServerSocketChannel 对象监听客户端，得到的新 SocketChannel 就可以和客户端通信了。

Step2 ： 编写一个聊天客户端：

在写聊天客户端中，分别定义了几个方法：

```
public void go() throws IOException{
    SetupNetWork(); (可以在该方法中调用 SetupNetWork 方法。
}
Private SetupNetWork() throws IOException {
```

先 open 一个 SocketChannel 对象，绑定一个确定端口（用 InetSocketAddress 对象来实现）;

用 Channels.newWriter 可以创建一个 Writer 对象（Writer 是底层字节流和高级字符流的联系）

再将 PrintWriter 串联到 Writer 中。

该 PrintWriter 对象名叫 Writer。

}

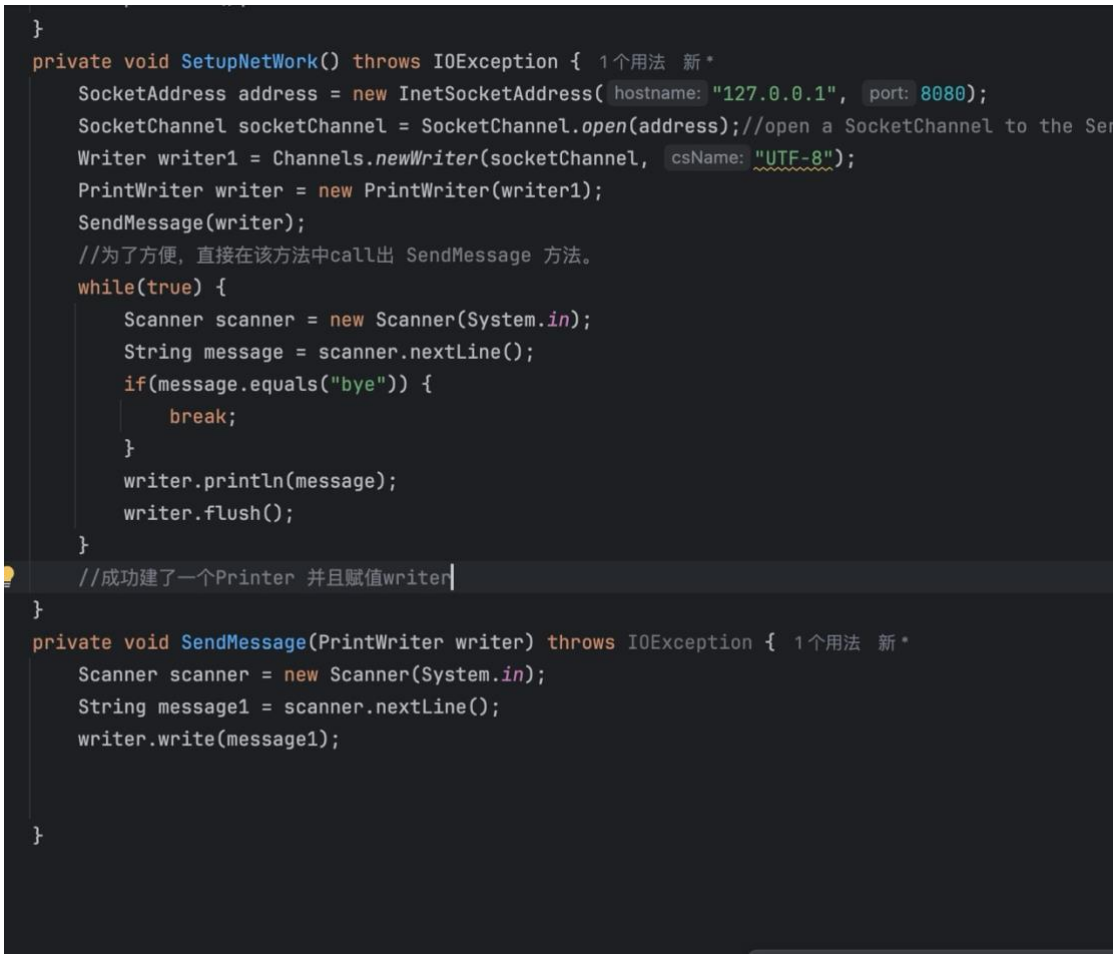
private void SendMessage(PrintWriter Writer) throws IOException {

要想利用前一个方法的成员对象，需要利用方法的引用传递，然后再利用 Writer 对象打印出我们想要的内容。

注意想要输入的时候可以用 Scanner 对象来从键盘处得到信息，最后用一个 While 循环，直到输入“bye”的时候跳出循环

}

在代码中体现为：



```
}
private void SetupNetWork() throws IOException {  1个用法  新*
    SocketAddress address = new InetSocketAddress( hostname: "127.0.0.1",  port: 8080);
    SocketChannel socketChannel = SocketChannel.open(address);//open a SocketChannel to the Ser
    Writer writer1 = Channels.newWriter(socketChannel,  csName: "UTF-8");
    PrintWriter writer = new PrintWriter(writer1);
    SendMessage(writer);
    //为了方便，直接在该方法中call出 SendMessage 方法。
    while(true) {
        Scanner scanner = new Scanner(System.in);
        String message = scanner.nextLine();
        if(message.equals("bye")) {
            break;
        }
        writer.println(message);
        writer.flush();
    }
    //成功建了一个Printer 并且赋值writer
}
private void SendMessage(PrintWriter writer) throws IOException {  1个用法  新*
    Scanner scanner = new Scanner(System.in);
    String message1 = scanner.nextLine();
    writer.write(message1);


}
```

Step 3：编写一个聊天服务器：

首先新建一个 ArrayList<>的对象，该列表只包含 PrintWriter 对象，且不可被重新赋值。

在 public void go（）方法中，利用 Step1 中方法构建一个服务端

在 public void tellEveryOne（String message）中利用加强 for 将列表内容遍历出来，

实现消息的打印。

在 public class ClientHandler implements Runnable 中，日该类实现 Runnable 接口，从而自定义一个控制类，覆盖重写 run 方法，运行函数。

什么叫 Runnable 接口？

Runnable 接口是用来定义线程执行的任务，与 Thread 类配合使用，实现多线程编程，避免了 Thread 类直接继承带来的单继承限制问题。

在 Runnable 的实现类中，需要实现 Run 方法并且将实例传给 Thread 的构造函数，然后利用 Thread 类的 start 方法来启动线程。

在 go 中通过 BufferedReader 来读取数据，用一个字符串型变量来接，可以把读到的结果打印出来，检测是否搭建成功。

在循环中，读取到“886”的时候跳出循环。

该部分在代码体现为：

```
    SimpleChatServer server = new SimpleChatServer();
    server.go();
}
public void go() throws IOException {  1个用法  新*
    try {
        ServerSocketChannel serversocketChannel = ServerSocketChannel.open();
        serversocketChannel.socket().bind(new InetSocketAddress( port: 8080));//建一个ServerSocket
        SocketChannel socketChannel = serversocketChannel.accept();//新的SocketChannel可以和客户端
        while (true) {
            Reader reader = Channels.newReader(socketChannel,  csName: "UTF-8");
            BufferedReader bufferedReader = new BufferedReader(reader);
            //在go中新建一个BufferedReader对象
            String msg = bufferedReader.readLine();
            tellEveryone( message: "收到信息", bufferedReader);
            if("886".equals(msg)){
                break;
            }
            System.out.println(msg);
            Writer writer = Channels.newWriter(socketChannel,  csName: "UTF-8");
            PrintWriter writer1 = new PrintWriter(writer);
            String s = bufferedReader.readLine();
            writer1.write(s);
            writer1.write( s: "收到信息");
            writer1.flush();
        }

    }catch (Exception e){
        e.printStackTrace();
    }
```

最后实现的向服务端发送信息的运行结果截图为：

```
 com.data.send.SympleChatClientA
Hello!
今天天气真好!
bye

进程已结束，退出代码为 0
```

```
 com.data.send.SimpleChatServer
Hello! 今天天气真好!
```