实验 7 网络编程

实验目的

1. 掌握 Socket 通信
2. 掌握 UDP 数据报通信
3. 掌握网络编程中多线程的应用

实验步骤

步骤 1：创建工程，工程名以学号姓名方式命名，“2020010505-薛飞宇”。
步骤 2：为每道实习题目创建对应的包，如“work1”、“work2”……；
步骤 3：按照要求创建源代码，进行编写，注意编码格式，如缩进、命名规范等；
步骤 4：规范书写实习报告；
步骤 5：实现与测试。附上代码或以附件的形式提交，同时贴上必要的代码运行截图；
步骤 6：及时总结心得体会与备忘。

实验内容

实验题 1 掌握 Socket 类的使用

在两台机器上分别运行以下服务器端程序和客户端程序，截图观察程序的多次运行结果，回答问题。

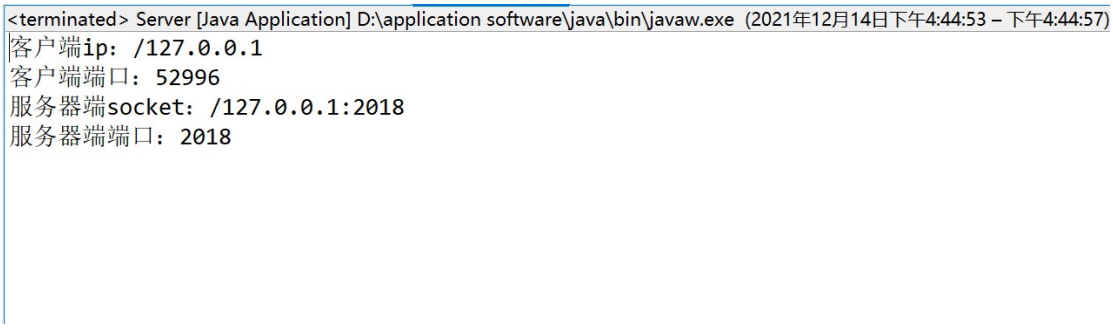

图 1.1 实验 1 Server

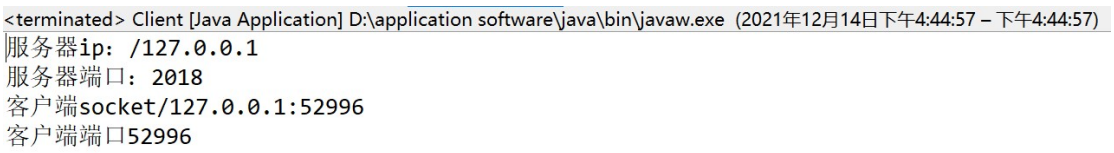

图 1.2 实验 1 Client

根据实验结果，添加相关注释如下图：

```
    ServerSocket sSocket  = null;
    Socket socket = null;
    try {
        sSocket = new ServerSocket(2018);
        //将服务器端程序设置端口为2018
    } catch (IOException e) {
        // TODO Auto-generated catch block
        System.out.println("端口已被占用。");
    }
    try {
        socket = sSocket.accept();
        //等待客户端的连接请求，建立相应的socket
        System.out.println("客户端ip: "+socket.getInetAddress());
        //获取客户端的ip地址
        System.out.println("客户端端口: "+socket.getPort());
        //获取客户端程序的端口
        System.out.println("服务器端socket: "+socket.getLocalSocketAddress());
        //获取服务器端的socket
        System.out.println("服务器端端口: "+socket.getLocalPort());
        //获取服务器端的端口
        socket.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        System.out.println("连接客户端异常。");
```

图 1.3 服务器端程序注释

```
public class Client {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Socket socket = new Socket();
        //新建一个socket对象
        InetSocketAddress isa = new InetSocketAddress("127.0.0.1",2018);
        //新建一个Server的socket地址的对象
        try {
            socket.connect(isa);
            //将socket连接Server的地址
            System.out.println("服务器ip: "+socket.getInetAddress());
            //获取服务器的ip地址
            System.out.println("服务器端口: "+socket.getPort());
            //获取服务器的端口
            System.out.println("客户端socket: "+socket.getLocalSocketAddress());
            //获取客户端的socket
            System.out.println("客户端端口: "+socket.getLocalPort());
            //获取客户端的端口
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("连接服务器异常。");
        }
    }
```

图 1.4 客户端程序注释

下面是本程序的 UML 类图。

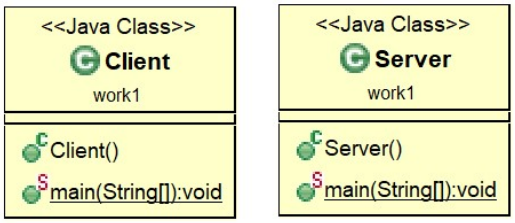

图 1.5 实验 1UML 类图

实验题 2 使用 TCP 协议实现文件传送

在当前工程的根目录新建两个文件夹:“source”和“destination”,在 source 中放置两个文件:“cartoon.jpg”以及“file.docx”。这两个文件见实 习附件。实现一个服务器端程序 Server，服务器端可以通过文件流直接读取文 件内容。实现一个客户端程序 Client，客户端程序运行后，提示用户输入文件 名。如，用户在客户端输入“file.docx”并回车，则程序通过 TCP 协议读取服务器端的数据，将该文件传送一个副本保存在“destination”目录中。

本题稍有难度，但是经过我的缜密分析，问题已经迎刃而解。

下面是测试结果。

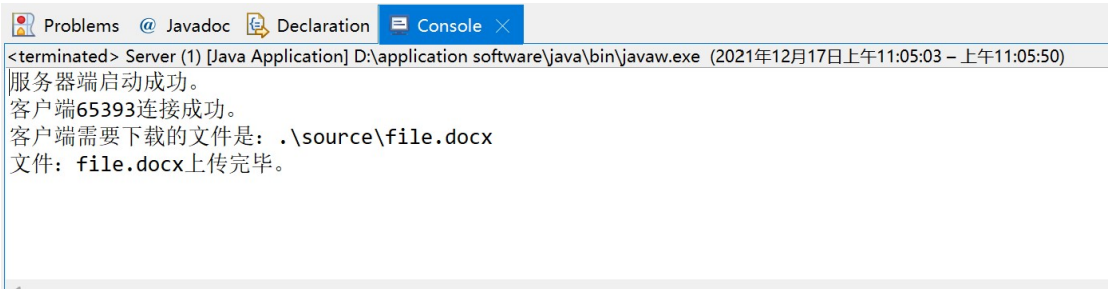

图 2.1 实验 2 结果

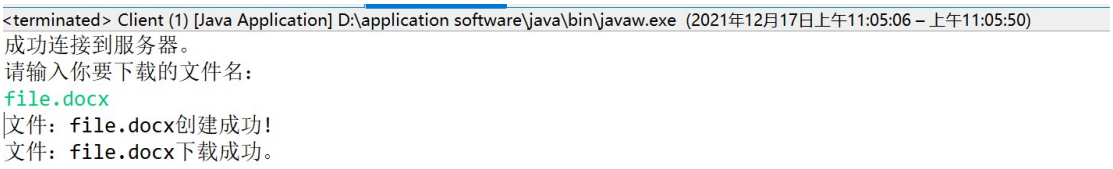

图 2.2 实验 2 结果

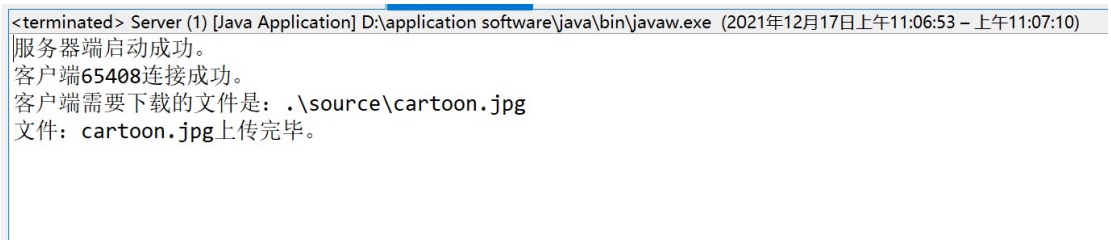

图 2.3 实验 2 结果

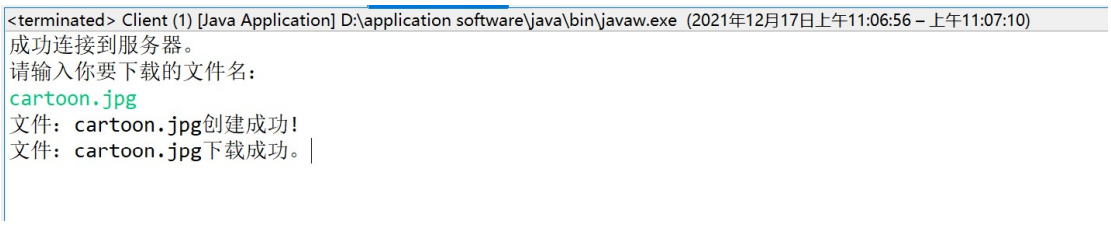

图 2.4 实验 2 结果

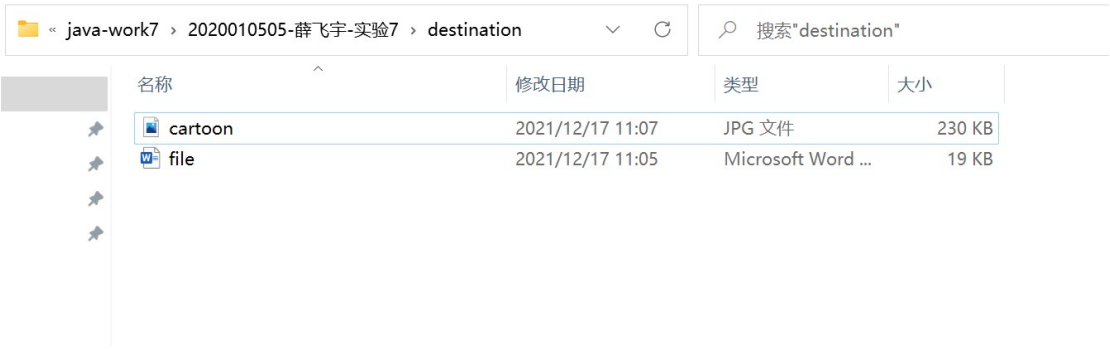

图 2.5 实验 2 结果

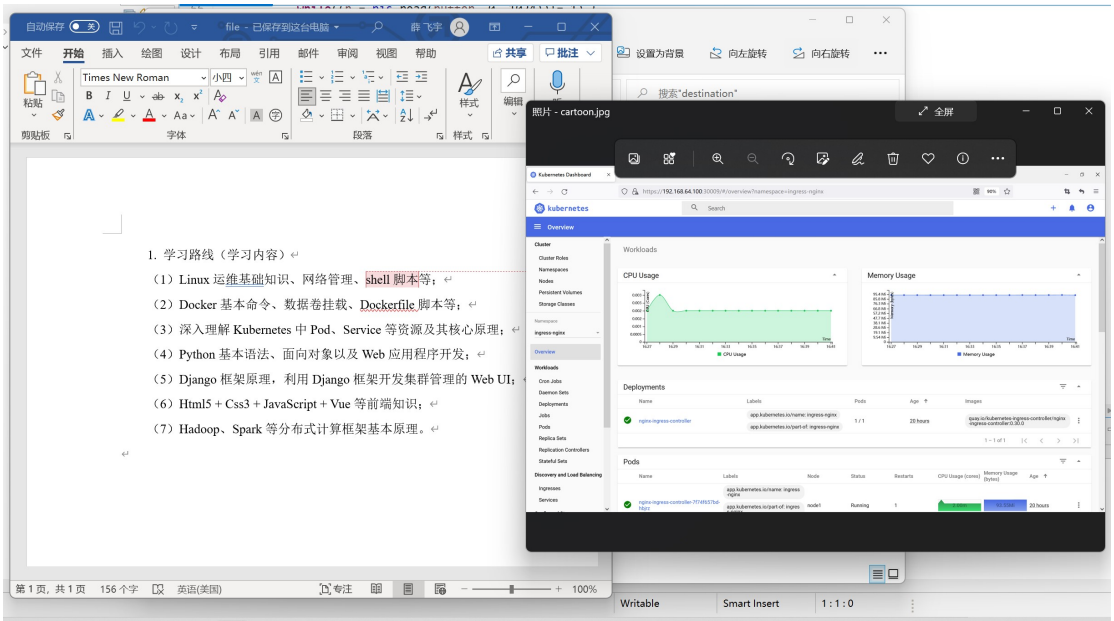

图 2.6 实验 2 结果

本题最主要的难点就是输入输出流和网络结合，所以要解决这道题，首先要理清楚这些对象之间的关系。

下面是我画的对象之间的关系模型图。

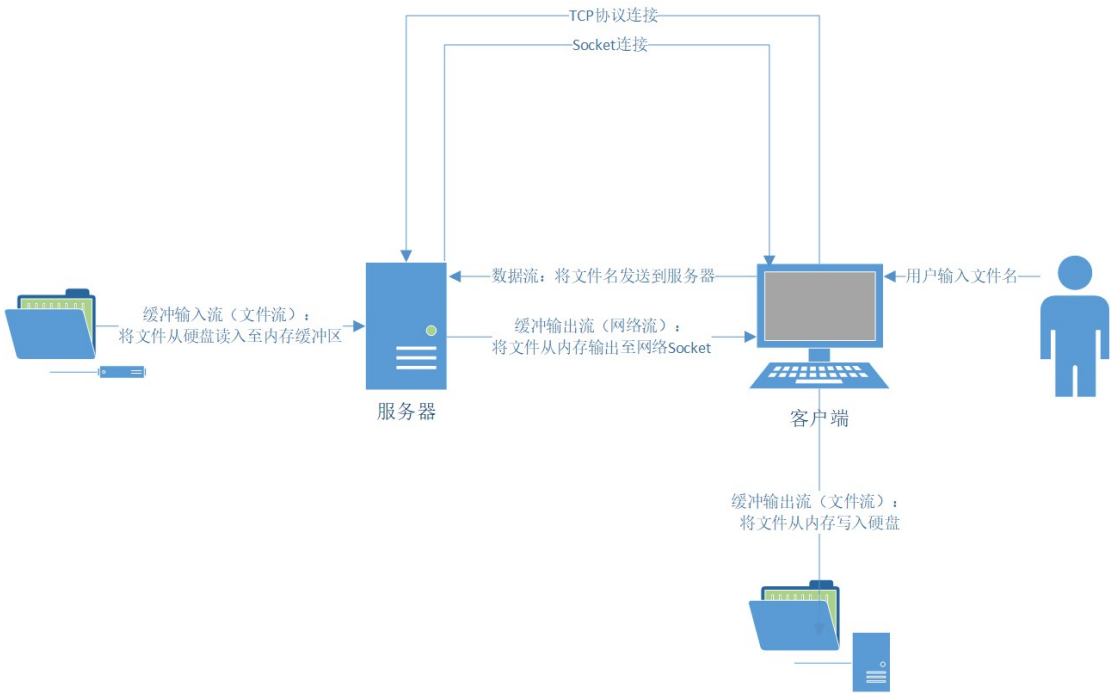

图 2.7 网络文件模型图

首先，通过 Socket 将服务器和客户端连接起来，用户在客户端将文件名输入到内存中，客户端通过数据流将这个文件名输出到服务器端的内存中，然后根据这个文件名建立相应的文件输入流，将文件内容读入至内存中，再通过网络输出到客户端内存中去，客户端再建立相应的文件输出流，将文件内容写入硬盘。

这个实验成功了，但是我还没有时间去完成多线程的通信任务。

下面是本实验的 UML 类图。

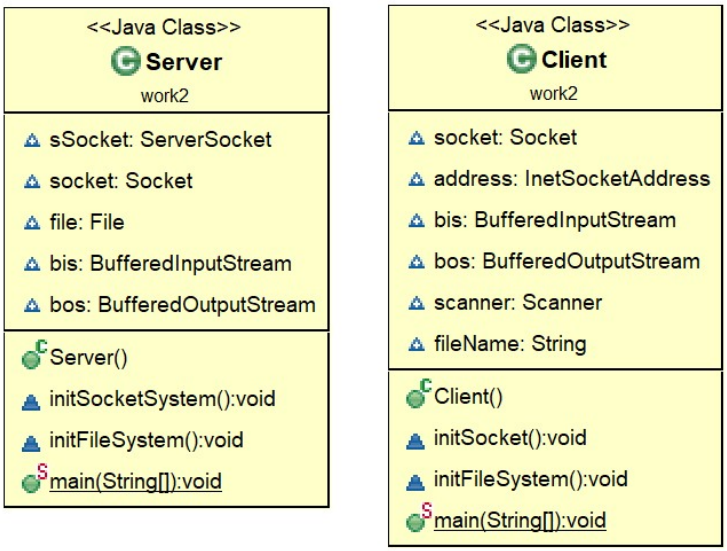

图 2.8 实验 2UML 类图

实验题 3 对象流在网络中的应用

现有一个 Triangle（三角形）类，服务器用于接收客户端传来的 Triangle 对象，计算三角形面积，或者判断三角形是否合法，将计算结果（字符串）传给客户端。

对象流相比文件流还是简单一点，因为对象流只是在服务器和客户端的内存间进行信息的传输，并不涉及硬盘的读写。

先通过 WindowBuilder 将客户端的界面搭建好。

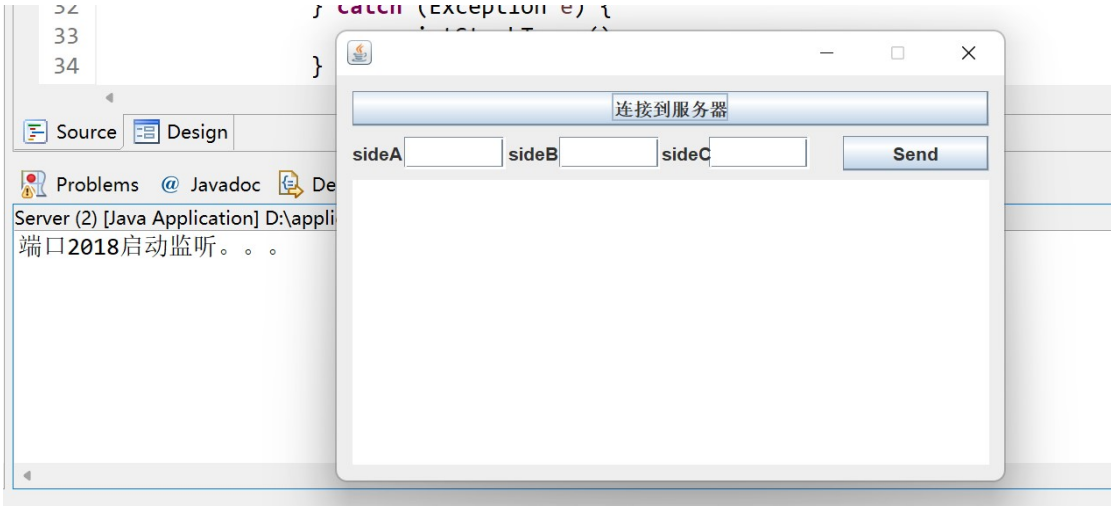

图 3.1 客户端界面

然后，在两个按钮上注册相应的事件监听器。用户点击连接到服务器，Client 的一个对象就调用 initSocket()方法，里面封装了 connect()方法。用户点击 Send 按钮，就新建一个相应的 Triangle 对象，由于已经实现了 Serializable 接口，直接将该对象通过对象流发送至服务器，服务完成对这个对象的处理工作后，将字符串信息通过对象流传输到客户端的界面中去，整个任务就完成了。

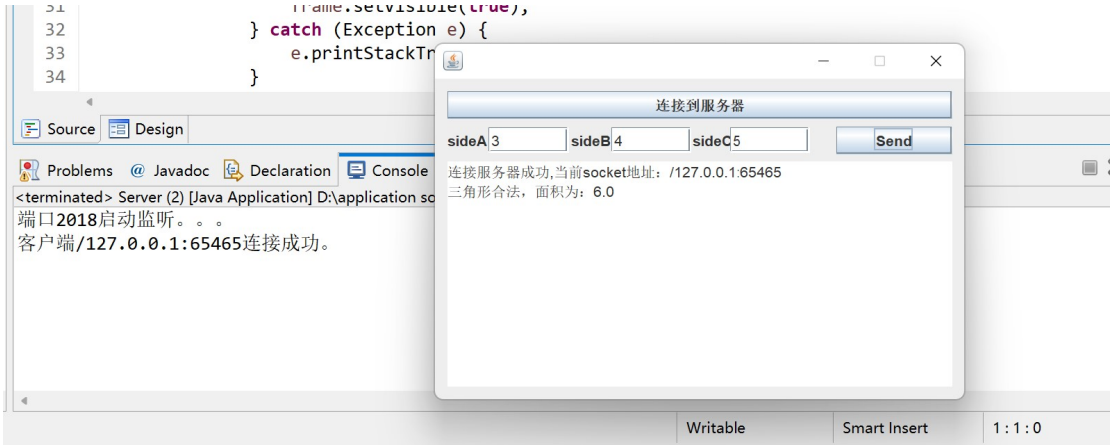

图 3.2 实验 3 测试结果

下面是本实验的 UML 类图。

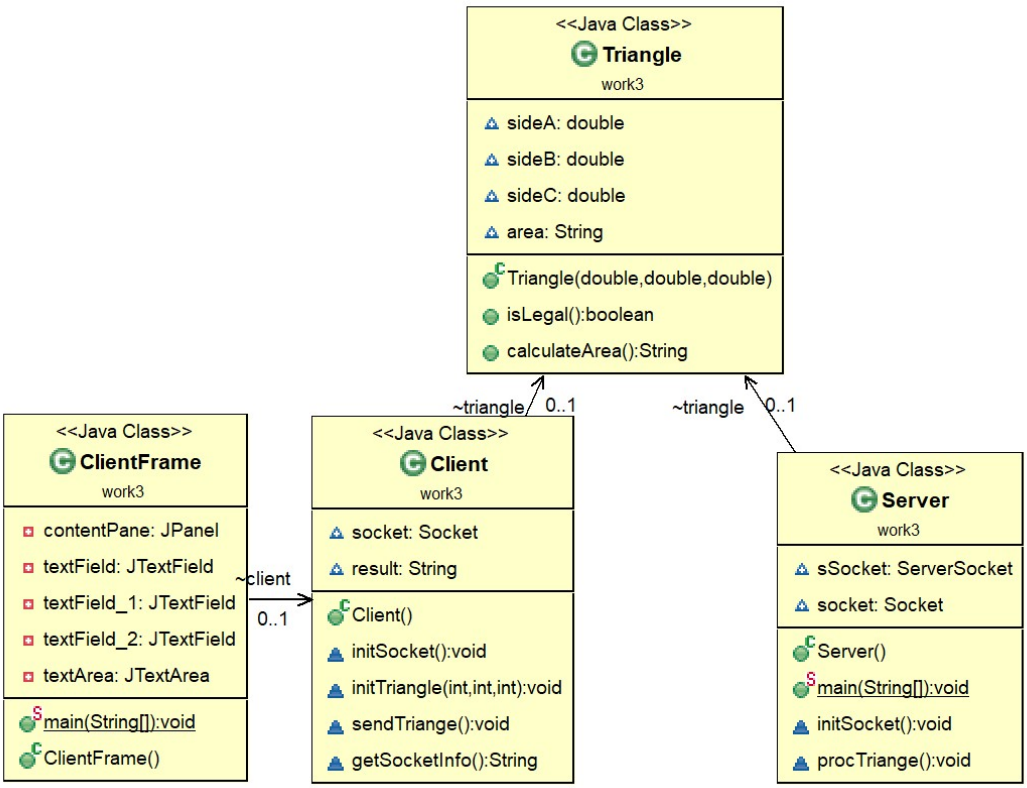

图 3.2 实验 3UML 类图

但是本实验我依然没有完成多线程的任务。

实验题 4 使用 UDP 数据报进行聊天

用 UDP 数据报传输数据，实现两个终端的消息通信。使用 Java 命令分别运行服务器端和客户端。

基本要求：实现两个终端 UDPA 和 UDPB，两个终端可以互发字符串消息。若某一端（如 A）发送了一个“TIME”字符串，则另一端（如 B）将其视作 一个命令，将目前的系统时间立即发送过去。如果接收的字符串不是“TIME”命令，则只接收显示即可。

UDP 编程和 TCP 编程的差别还是挺大的。TCP 编程是建立在 Socket 连接的基础之上的，因此 TCP 编程之前都要做的事是服务器端开启一个监听端口，然后客户端调用 connect()方法去连接服务器的监听端口。

然而，UDP 编程是基于数据报的。在两个终端进行的通信之前，它们需要通过 DatagramSocket 指定接收数据报的端口。然后用 DatagramPacket 封装要传递信息的字节数组以及要发送的套接字地址，通过 DatagramSocket 的 send()和 receive()方法进行发送和接收。

下面是本实验的测试结果。

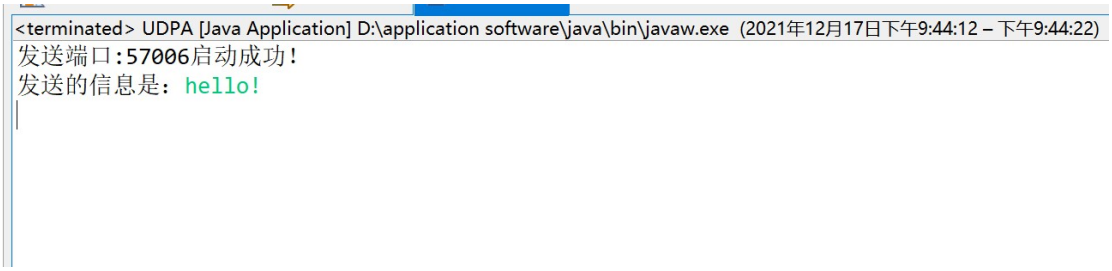

图 4.1 实验 4 测试

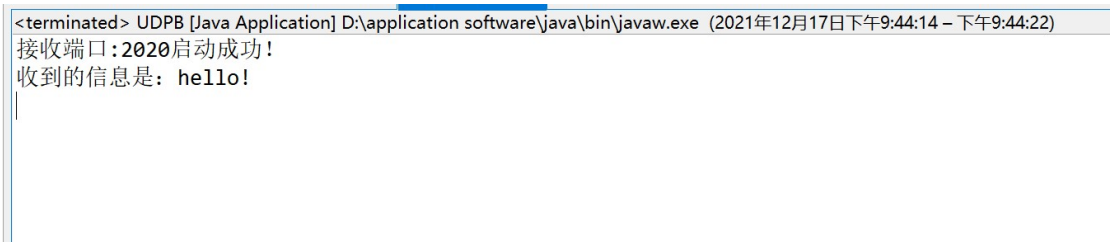

图 4.2 实验 4 测试

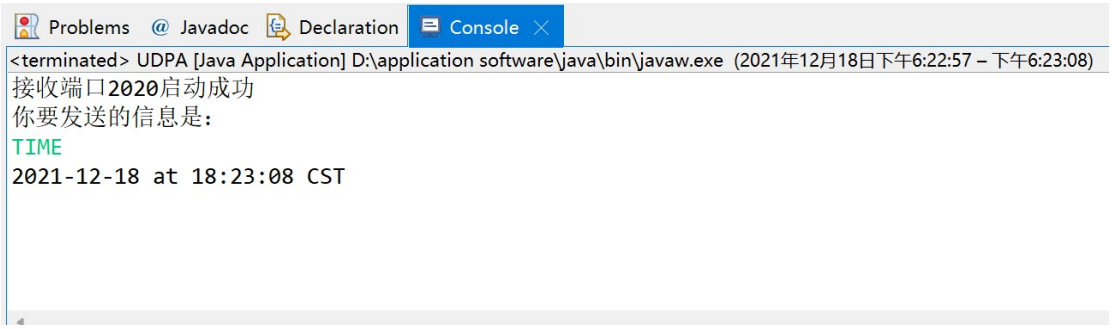

图 4.3 实验 4 测试

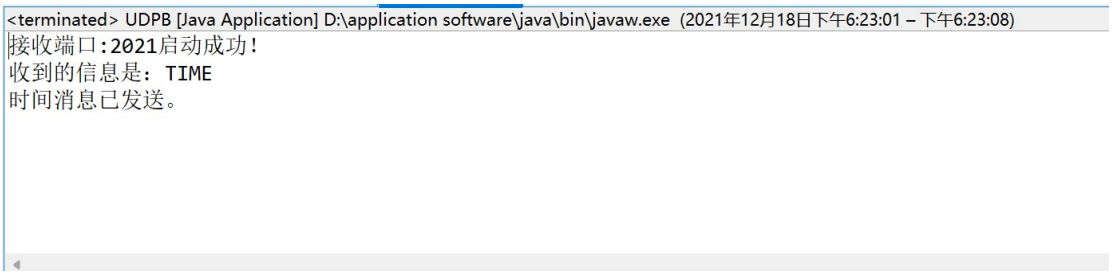

图 4.4 实验 4 测试

下面是本实验的 UML 类图。

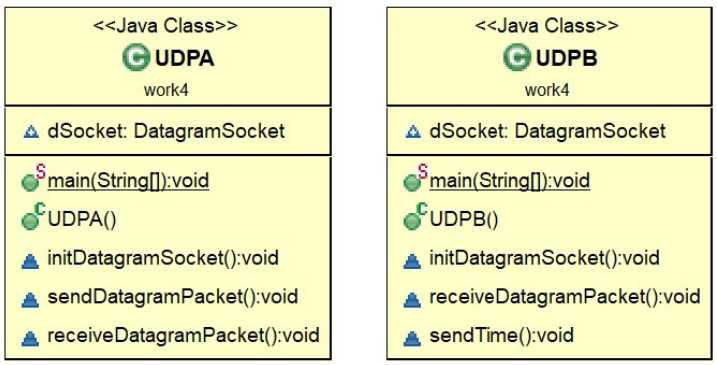

图 4.5 实验 4UML 测试

本次实验尚未解决的疑难：

多线程与网络的结合。

心得体会：

本次实验，我掌握 Java 网络编程的概念和实现，掌握、使用 TCP 协议进行 Socket 连接以及通信，在此基础上完成了网络文件的上传和下载。与 Java 的 Swing 结合，开发了带有图形界面的网络对象传输系统。掌握和使用 UDP 协议实现了两个终端之间的聊天。