深 圳 大 学 实 验 报 告

课程名称 计算机网络

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 杨术

报 告 人 张欣杰 学号 2020151091

实验时间 2023/03/27

提交时间 2023/04/03

教务处制

一、实验目的与要求

掌握 Socket 的 TCP 通信、 Socket 的 UDP 通信

二、实验内容与方法

实验内容：Socket、ServerSocket 类和 DatagramPacket 、DatagramSocket 类的使用
实验环境和要求：Windows 操作系统、Internet 链接、Java 开发环境
实验方法：

1. Java 对网络编程的支持
   java.net 包中的主要的类和可能产生的例外包括：
   1) 面向应用层的类：URL，URLConnection
   2) 面向传输层/IP 层的类：
      TCP 协议相关类：Socket、ServerSocket
      UDP 协议相关类：DatagramPacket、DatagramSocket、MulticastSocket
      表示 IP 的类：InetAddress
2. 套接字（socket）
   1) 用于实现网络上客户端程序和服务器程序之间的连接。
   2) 套接字负责网络上进程之间的通信
   3) 客户端程序可以向套接字里写入请求，然后服务器会处理这个请求，并把处理结果通过套接字送回。
   4) 服务器应用程序一般会侦听一个特定端口，用来等待客户的连接请求，当一个连接请求到达时，客户和服务器会建立一个通信连接，在连接过程中，客户被分配一个本地端口号并与一个 Socket 连接，客户通过写 Socket 来通知服务器，再通过读取 Socket 来获取服务器发送过来的信息。
   5) 类似地，服务器也获取一个本地端口号，它需要一个新的端口号来侦听原始端口上的其他连接请求。服务器也给它的本地端口连接一个 Socket，通过读写它来与客户通信。
   6) Socket 可以根据通信性质分类，这种性质对于用户是可见的。
   7) 应用程序一般仅在同一类的套接字之间进行通信。不过只要底层的通信协议允许，不同类型的套接字之间也可以通信。
   8) 目前可以使用两种套接字，即流套接字和数据报套接字：流套接字提供了双向的、有序的、无重复的并且无记录边界的数据流服务。TCP 是一种流套接字协议。据数报套接字支持双向的数据流，但并不保证是可靠、有序、无重复的，也就是说，一个以数据报套接字来接收信息的进程有可能发现信息重复了，或者和发出的顺序不同了。数据报套接字的一个重要特点是它保留了记录边界。UDP 即是一种数据报套接字协议。
3. 端口
   1) 是一个逻辑概念。每一个服务器都运行在该主机的一个对外开放的端口上
   2) 一个主机上可以有多种服务器，也就是有多个端口。程序员可以在创建自己的服务器程序时使用其它端口（即除了系统默认的端口）
   3) 端口常以数字编号，作为客户可指定一个端口号，用这个端口号来连接相应的服务器以接收服务
4. Socket 类的构造方法

public Socket(String host, int port)
public Socket(InetAddress address, int port)
public Socket(String host, int port, InetAddress localAddr, int localPort)

5. Socket 的常用方法

Socket 的输入/输出流管理；抛出例外 IOException。
public InputStream getInputStream()
public void shutdownInput()
public OutputStream getOutputStream()
public void shutdownOutput()
关闭 Socket：public void close() throws IOException
设置/获取 Socket 数据：

1) public InetAddress getInetAddress()：返回此套接字链接的地址对象
2) public InetAddress getLocalAddress()：返回此套接字本地的地址对象
3) public int getPort()：返回此套接字链接地址的端口

6. ServeSocket 类

1) 构造方法：
   public ServerSocket(int port)
   public ServerSocket(int port, int backlog)：支持指定数目的连接
   public ServerSocket(int port, int backlog, InetAddress bindAddr)：在指定的机器上运行
2) 主要方法：
   public Socket accept()：等待客户端的连接
   public void close()：关闭 Socket
3) 设置/获取 Socket 数据的方法：
   public InetAddress getInetAddress()
   public int getLocalPort()

7. UDP

UDP 通信是一种无连接的数据报通信。使用该协议，两个程序进行通信时不用建立连接；数据以独立的包为单位发送，包的容量限定在 64KB 以内；每个数据报需要有完整的收/发地址，可以随时进行收/发数据报，但不保证传送顺序和内容准确；数据报可能会被丢失、延误等。UDP 通信是不可靠的通信，但通信速度较快，常常被应用在某些要求实时交互，准确性要求不高，但传输速度要求较高的场合（如视频会议系统等）。

Java 中，基于 UDP 协议实现网络通信的类有三个：

1) 用于表示通信数据的数据报类：DatagramPacket
2) 用于进行端到端通信的类：DatagramSocket
3) 用于广播通信的类：MulticastSocket

三、实验步骤与过程

1. 利用 Socket 类和 ServerSocket 类编写一个 C/S 程序，实现 C/S 通信

在服务器端，首先使用 new ServerSocket(port)方法监听指定端口，即指定 port 参数，我在本次实验中选择监听的端口为 6666 端口。如果客户端连接成功，则在控制台中打印出连接成功信息，然后使用 getInputStream()方法接受客户端数据，在接收到客户端数据之后，将接收到的客户端数据打印出来。在客户端代码中，首先需

要使用 new socket(host，port)方法连接到本机的服务器端口，然后向指定的端口发送数据。

Server 代码：

```
//监听6666端口
ServerSocket ss = new ServerSocket( port: 6666);
//等待客户端连接
Socket s = ss.accept();
//如果客户端连接成功，打印出来
System.out.println("Client connected:"+s);
//接收客户端的数据
DataInputStream dis = new DataInputStream(s.getInputStream());
String str = (String)dis.readUTF();
System.out.println("message= "+str);
//关闭资源
dis.close();
s.close();
ss.close();
```

Client 代码：

```
//连接到本机的6666端口
Socket s = new Socket( host: "localhost", port: 6666);
//向本机的6666端口发送数据
DataOutputStream dos = new DataOutputStream(s.getOutputStream());
dos.writeUTF( str: "Hello World!");
//关闭资源
dos.flush();
dos.close();
s.close();
```

运行结果：

首先运行服务器端代码，再运行客户端代码，然后再查看服务器端代码的输出。

```
Client connected:Socket[addr=/127.0.0.1,port=61958,localport=6666]
message= Hello World!
```

2. 客户端向服务器端发送 Time 命令，服务器端接受到该字符串后将服务器端当前时间返回给客户端；客户端向服务器端发送 Exit 命令，服务器端向客户端返回“Bye”后退出。

   此题代码与第一题思路基本一致，只不过在服务器端中增加一个持续出发条件，即 while(True)进行持续输入操作，当输入为“Exit”时跳出循环然后结束程序。在服务器端，服务器端的代码也一样，设置 while(True)持续接受客户端发来的数据，如果获取的数据为“Time”则返回服务器当前时间，如果数据为“Exit”则向客户端发送“Bye”报文，然后跳出循环结束程序。

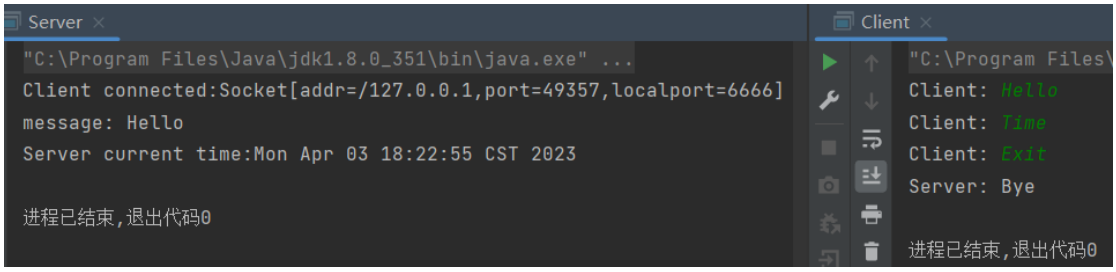
   Server 端主要代码

```java
while(true){
    String str = (String)dis.readUTF();
    switch (str){
        case "Time":
            System.out.println("Server current time:"+new Date());
            break;
        case "Exit":
            //向客户端发送报文
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            dos.writeUTF( str: "Bye");
            dos.flush();
            dos.close();
            break;
        default:
            System.out.println("message: "+str);
            break;
    }
    if(str.equals("Exit"))
        break;
}
```

Client 端主要代码：

```java
while(true){
    System.out.print("Client: ");
    String str = sc.nextLine();
    dos.writeUTF(str);
    if (str.equals("Exit")){
        //接受服务器端的数据
        DataInputStream dis = new DataInputStream(s.getInputStream());
        System.out.println("Server: "+dis.readUTF());
        dis.close();
        break;
    }
}
```

代码运行效果：



3. 编写完整程序；一个服务器端程序，一个客户端程序。服务器端和客户端都需要打印出接受到的消息和发出的命令
   完成本题只需要在原来服务器端代码的基础上加入一个回显即可

```
String str = (String)dis.readUTF();
System.out.println("Instruction from Client: "+str);
switch (str){
```

运行效果：

4. 编写数据报通信程序，实现简单的聊天功能

首先定义聊天程序所需要的组件：

```
private final JButton exit_button; // 退出按钮
6 个用法
private final JButton send_button; // 发送按钮
6 个用法
private final JButton clear_button; // 清除按钮
2 个用法
private final JTextField IP; // IP 地址输入框
1 个用法
private final JTextField port; // 端口号输入框
5 个用法
private final JTextField send_message; // 发送消息输入框
8 个用法
private final JTextArea textArea; // 显示聊天记录的文本框
2 个用法
private final int s_port; // 发送端口号
2 个用法
private final int r_port; // 接收端口号
2 个用法
private final String sender; // 发送方用户名
2 个用法
private final String receiver; // 接收方用户名
3 个用法
private DatagramSocket socket; // DatagramSocket 对象
```

定义 send_message 方法，用于发送消息：

```
public void send_message() throws UnknownHostException {
    String ip=IP.getText(); // 获取 IP 地址
    InetAddress address=InetAddress.getByName(ip); // 将 IP 地址转换为 InetAddress 对象
    byte[] buf=send_message.getText().getBytes(); // 获取要发送的消息
    DatagramPacket packet=new DatagramPacket(buf,buf.length,address,s_port); // 创建 DatagramPacket 对象
    textArea.append(sender+":"+send_message.getText()+"\n"); // 在文本框中显示消息
    try {
        socket.send(packet); // 发送消息
    } catch (IOException e) {
        e.printStackTrace();
    }
    send_message.setText(null); // 清空发送消息的输入框
}
```

定义 get_message 方法，用于接受消息：

```
public void get_message() throws IOException {
    try {
        socket=new DatagramSocket(r_port);
    }catch (SocketException e){
        e.printStackTrace();
    }
    byte[] buf=new byte[1024];
    final DatagramPacket packet=new DatagramPacket(buf,buf.length);
    Runnable runnable=new Runnable() {
        @Override
        public void run() {
            while (true){
                try {
                    Thread.sleep( millis: 100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                try {
                    socket.receive(packet);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                String message=new String(packet.getData(), offset: 0,packet.getLength());
                textArea.append(receiver+":"+message+"\n");
            }
        }
    };
    new Thread(runnable).start();
}
```

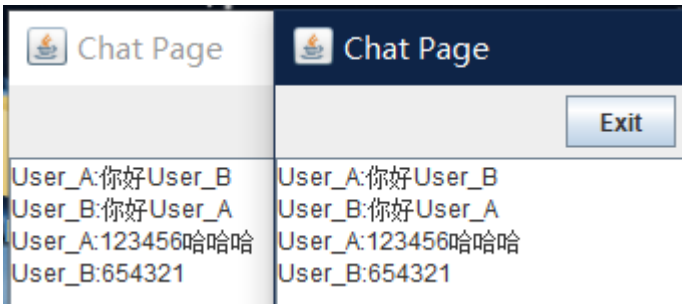
然后构造聊天框主函数 ChaPage：

```java
public ChatPage(int send_port,int receive_port,String sender,String receiver) throws IOException {
    this.sender=sender;  // 发送方名称
    this.receiver=receiver;  // 接收方名称
    this.s_port=send_port;  // 发送端口号
    this.r_port=receive_port;  // 接收端口号
    this.setTitle("Chat Page");  // 设置聊天页面的标题
    setBounds( x: 100, y: 100, width: 500, height: 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());
    JPanel panel=new JPanel();
    panel.setLayout(new FlowLayout());
    add(panel,BorderLayout.NORTH);
    exit_button=new JButton( text: "Exit");
    exit_button.setSize( width: 50, height: 20);
    send_button=new JButton( text: "Send");
    send_button.setSize( width: 50, height: 20);
    clear_button=new JButton( text: "Clear");
    clear_button.setSize( width: 50, height: 20);
    panel.add(exit_button);
    panel.add(send_button);
    panel.add(clear_button);
    textArea=new JTextArea();
    textArea.setLineWrap(true);
    textArea.setWrapStyleWord(true);
    JScrollPane scrollPane=new JScrollPane(textArea);  // 创建带滚动条的文本域
    add(scrollPane,BorderLayout.CENTER);  // 把文本域添加到聊天页面的中央
    JPanel panel1=new JPanel();  // 创建一个新的面板
    BorderLayout borderLayout=new BorderLayout();  // 创建边框布局
    borderLayout.setHgap(10);  // 设置水平间距为10
    panel1.setLayout(borderLayout);  // 设置面板的布局管理器为边框布局
```

实现 actionPerformed 方法，对按钮事件进行处理

```java
public void actionPerformed(ActionEvent e){
    if(e.getSource()==exit_button){ // 点击了退出按钮
        System.exit( status: 0);
    }
    if(e.getSource()==send_button){ // 点击了发送按钮
        try {
            send_message();
        } catch (IOException ioException) {
            ioException.printStackTrace();
        }
    }
    if(e.getSource()==clear_button){ // 点击了清除按钮
        textArea.setText("");
    }
}
```

代码运行效果：



四、实验结论或体会

本次实验是关于网络编程的基础内容，包括 TCP 和 UDP 协议的通信、Socket 和 ServerSocket 类的使用以及图形用户界面的设计等。通过这次实验，我学习到了以下内容：

1. TCP 协议和 UDP 协议的区别和特点，以及在不同场景下如何选择合适的协议；
2. Socket 和 ServerSocket 类的基本用法，包括创建套接字、监听和接受连接、发送和接收数据等操作；
3. 图形用户界面的设计原理和基本组件，包括窗口、标签、按钮、文本框、对话框等，并掌握了如何使用 Java Swing 库创建 GUI 界面；
4. 了解如何将网络通信和 GUI 界面结合起来，实现基于 TCP 和 UDP 协议的聊天程序。

通过本次实验，我深入了解了网络编程的基本知识和技能，并掌握了使用 Java 语言进行网络编程和 GUI 设计的方法。同时，在实验过程中，我也遇到了一些问题，例如 Socket 编程时需要注意的线程安全问题、UDP 协议下数据包的大小限制等等，通过查阅资料和与同学的讨论，我逐渐解决了这些问题，对网络编程的理解也更加深入了一步。

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。