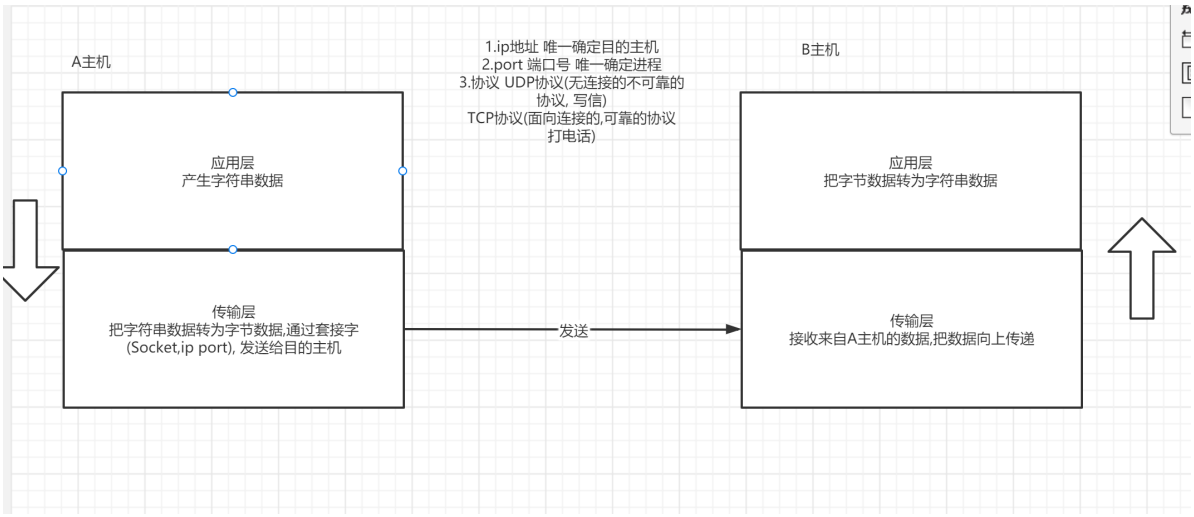
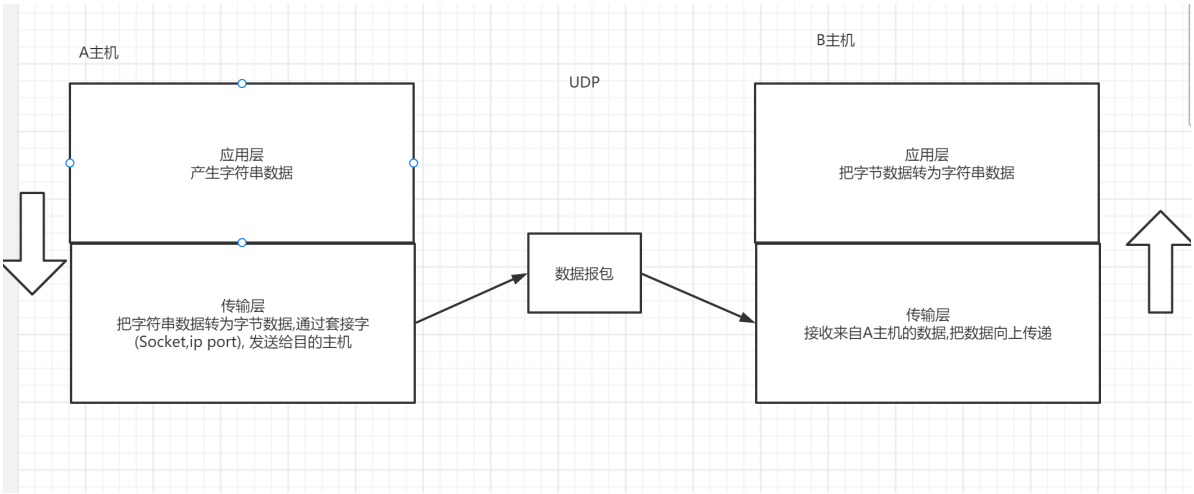
基本原理



UDP编程

传输原理

通过数据报包进行传输



发送端步骤

- 创建发送端的Socket对象
- 把要发送的数据封装成数据报包
- 通过send(数据报包) 方法发送出去
- close

接收端步骤

- 创建接收端的socket对象
- 创建一个用于接收的数据报包
- receive(空包)接收
- 解析数据,把数据从包里取出来
- close

DatagramSocket

此类表示用来**发送**和**接收**数据报包的套接字。

构造方法

DatagramSocket(int port)　　创建数据报套接字并将其绑定到本地主机上的指定端口。

成员方法

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。

DatagramPacket

此类表示数据报包。

构造方法

用于发送的

```
DatagramPacket(byte[] buf,  int offset, int length, InetAddress address, int port)　　构造数据报包，用来将长度为 length 偏移量为 offset  的包发送到指定主机上的指定端口号。
```

用于接收的

```
DatagramPacket(byte[] buf, int offset, int length)
构造 DatagramPacket，用来接收长度为 length 的包，在缓冲区中指定了偏移量。
```

成员方法

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。
int	getOffset() 返回将要发送或接收到的数据的偏移量。

案例

v1 发送端发送消息,接收端接收并打印

```
package _23network.com.cskaoyan.udp.v1;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

/**
 * @description: 接收端
 * @author: 景天
 * @date: 2022/7/29 16:18
 **/

public class Receiver {
    public static void main(String[] args) throws IOException {
        // - 创建接收端的socket对象
        DatagramSocket datagramSocket = new DatagramSocket(9999);
        //- 创建一个用于接收的数据报包
        // DatagramPacket(byte[] buf, int offset, int length)
        //构造 DatagramPacket，用来接收长度为 length 的包，在缓冲区中指定了偏移量。
        byte[] bytes = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(bytes, 0,
bytes.length);

        //- receive(空包)接收
        System.out.println("recevie before");
        datagramSocket.receive(receivePacket);
        System.out.println("recevie after");

        //- 解析数据,把数据从包里取出来
        //| 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。    |
        //| int    | getOffset()       返回将要发送或接收到的数据的偏移量。  |
        byte[] data = receivePacket.getData();
        int offset = receivePacket.getOffset();
        int length = receivePacket.getLength();
        String s = new String(data, offset, length);
        System.out.println("接收到了来自"+receivePacket.getSocketAddress()+
                "的消息: "+s);
        //- close
        datagramSocket.close();
    }
}



package _23network.com.cskaoyan.udp.v1;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
```

```java
/**
 * @description: 发送端
 * @author: 景天
 * @date: 2022/7/29 16:18
 **/
/*
发送端发送消息,接收端接收并打印
 */
public class Sender {
    public static void main(String[] args) throws IOException {
        // - 创建发送端的Socket对象
        // DatagramSocket(int port)
        // 创建数据报套接字并将其绑定到本地主机上的指定端口。
        DatagramSocket datagramSocket = new DatagramSocket(8888);
        // 来自应用层的数据
        String s = "hello udp";
        byte[] bytes = s.getBytes();
        //- 把要发送的数据封装成数据报包
        // DatagramPacket(byte[] buf,  int offset, int length, InetAddress
address, int port)

        // 构造数据报包，用来将长度为 length 偏移量为 offset  的包发送到指定主机上的指定端
口号。
        InetAddress targetIp = InetAddress.getByName("127.0.0.1");
        int port = 9999;
        DatagramPacket sendPacket = new DatagramPacket(bytes, 0, bytes.length,
targetIp, port);


        //- 通过send(数据报包) 方法发送出去
        datagramSocket.send(sendPacket);

        //- close
        datagramSocket.close();
    }
}
```

v2 使用工具类优化v1

```java
package utils;

import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * @description:
 * @author: 景天
 * @date: 2022/7/29 16:32
 **/

public class NetworkUtils {
```

```
    // 获取用于发送的数据报包的方法
    public static DatagramPacket getSendPacket(String msg,String ip,int port)
throws UnknownHostException {
        // 把要发送的数据封装成数据报包
        byte[] bytes = msg.getBytes();
        InetAddress targetIp = InetAddress.getByName(ip);
        DatagramPacket sendPacket = new DatagramPacket(bytes, 0, bytes.length,
targetIp, port);
        // 最终要返回数据报包
        return sendPacket;
    }

    // 获取用于接收的数据报包的方法
    public static DatagramPacket getReceivePacket() {
        byte[] bytes = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(bytes, 0,
bytes.length);

        // 最终要返回一个用于接收的包
        return receivePacket;

    }

    // 解析数据的方法
    public static String parseMsg(DatagramPacket packet) {
        byte[] data = packet.getData();
        int offset = packet.getOffset();
        int length = packet.getLength();
        String msg = new String(data, offset, length);
        return msg;
    }
}
```

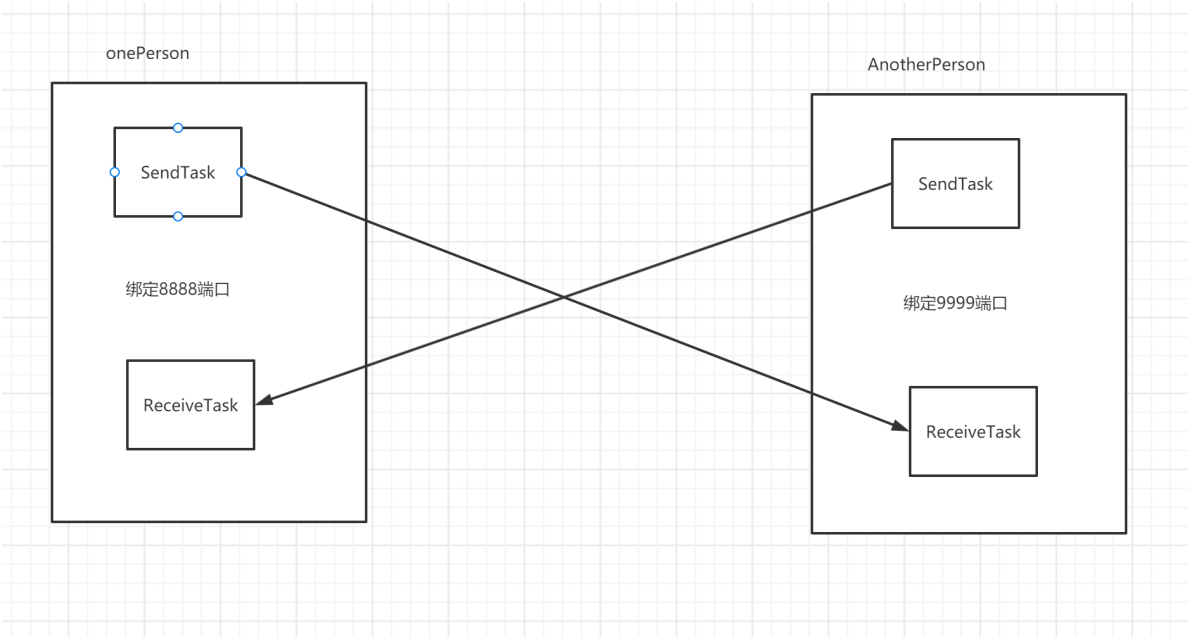
v3 发送端接收端相互发送

```
package _23network.com.cskaoyan.udp.v3;

import utils.NetworkUtils;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.util.Scanner;

/**
 * @description: 接收端
 * @author: 景天
 * @date: 2022/7/29 17:18
 **/

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

```
        // 创建接收端的socket对象
        DatagramSocket datagramSocket = new DatagramSocket(12306);
        // 创建Scanner对象
        Scanner scanner = new Scanner(System.in);
        // while
        while (true) {
            // 接收消息的逻辑
            // 创建用于接收的数据报包
            DatagramPacket receivePacket = NetworkUtils.getReceivePacket();
            // receive方法接收
            datagramSocket.receive(receivePacket);
            // parse
            String s = NetworkUtils.parseMsg(receivePacket);
            System.out.println(s);

            // 发送的逻辑
            // 键盘接收数据
            String msg = scanner.nextLine();
            // 封装成数据报包
            DatagramPacket sendPacket = NetworkUtils.getSendPacket(msg,
"127.0.0.1", 11111);

            // send发送
            datagramSocket.send(sendPacket);

        }


    }
}


package _23network.com.cskaoyan.udp.v3;

import utils.NetworkUtils;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.util.Scanner;

/**
 * @description: 发送端
 * @author: 景天
 * @date: 2022/7/29 16:48
 **/
/*
发送端接收端相互发送
 */
public class Sender {
    public static void main(String[] args) throws IOException {
        // 创建发送端的socket对象
        DatagramSocket datagramSocket = new DatagramSocket(11111);
        // 创建scanner对象
        Scanner scanner = new Scanner(System.in);
```

```
        // 多次发送
        // 循环
        while (true) {
            // 发送的逻辑
            // 键盘接收数据
            String s = scanner.nextLine();
            // 把数据封装成包
            DatagramPacket sendPacket = NetworkUtils.getSendPacket(s,
"127.0.0.1", 12306);
            // send
            datagramSocket.send(sendPacket);

            // 接收的逻辑
            // 创建的用于接收的数据报包
            DatagramPacket receivePacket = NetworkUtils.getReceivePacket();
            // receive接收
            datagramSocket.receive(receivePacket);
            // 解析
            String msg = NetworkUtils.parseMsg(receivePacket);
            // 打印
            System.out.println(msg);

        }


    }
}
```

v4 使用多线程优化v3



思路:

定义2个类OnePerson AnotherPerson

每个人里面都有2个任务运行在2个线程中 send receive

SendTask

- 实现Runnable接口
- 定义成员变量 DatagreamSocket ip port
- run方法里面发送

ReceiveTask

- 实现Runnable接口
- 定义成员变量 DatagreamSocket
- run里面去接收

```
package _23network.com.cskaoyan.udp.v4;

import utils.NetworkUtils;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

/**
 * @description: 接收任务
 * @author: 景天
 * @date: 2022/7/29 17:41
 **/

public class ReceiveTask implements Runnable{

    // 定义成员变量 DatagreamSocket
    DatagramSocket datagramSocket;

    public ReceiveTask(DatagramSocket datagramSocket) {
        this.datagramSocket = datagramSocket;
    }

    @Override
    public void run() {
        // 只接收消息
        while (true) {
            // 创建用于接收的数据报包
            DatagramPacket receivePacket = NetworkUtils.getReceivePacket();
            // receive
            try {
                datagramSocket.receive(receivePacket);
                // parse
                String msg = NetworkUtils.parseMsg(receivePacket);
                // sout
                System.out.println("接收到了来自" +
receivePacket.getSocketAddress() +
                        "的消息: " + msg);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

```
    }
}
```

```
package _23network.com.cskaoyan.udp.v4;

import utils.NetworkUtils;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.UnknownHostException;
import java.util.Scanner;

/**
 * @description: 发送任务
 * @author: 景天
 * @date: 2022/7/29 17:38
 **/

public class SendTask implements Runnable {
    // 定义成员变量 DatagreamSocket  ip  port
    DatagramSocket datagramSocket;
    String ip;
    int port;

    public SendTask(DatagramSocket datagramSocket, String ip, int port) {
        this.datagramSocket = datagramSocket;
        this.ip = ip;
        this.port = port;
    }

    @Override
    public void run() {
        // 创建Scanner
        Scanner scanner = new Scanner(System.in);
        // 只发送消息
        while (true) {
            // 接收键盘数据
            String s = scanner.nextLine();
            // 封装成数据报包
            try {
                DatagramPacket sendPacket = NetworkUtils.getSendPacket(s, ip,
port);
                // send
                datagramSocket.send(sendPacket);
            } catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

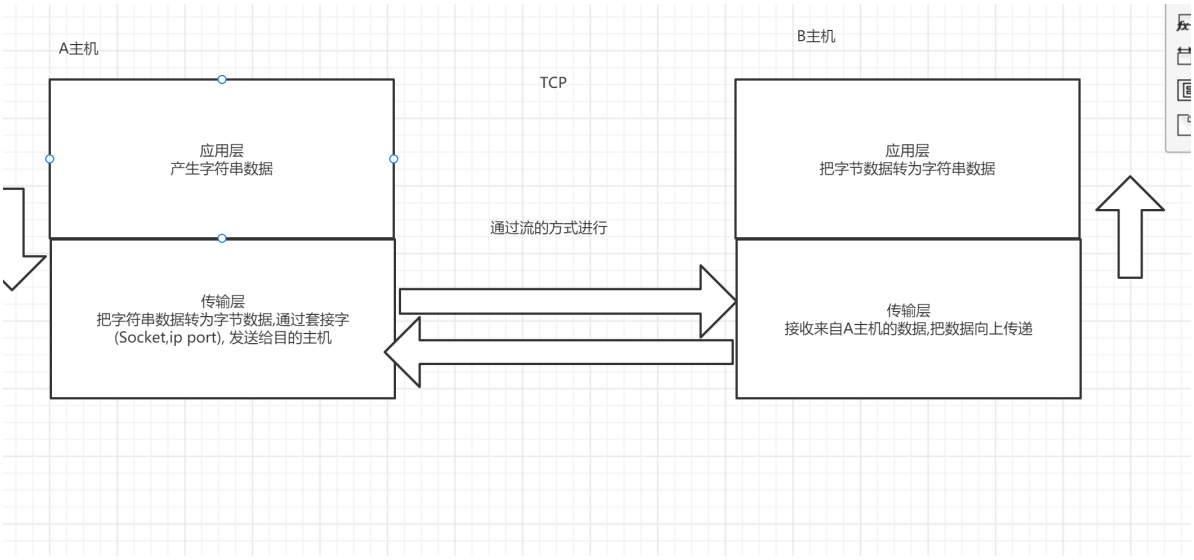
```
}


package _23network.com.cskaoyan.udp.v4;

import java.io.IOException;
import java.net.DatagramSocket;

/**
 * @description: 一个人
 * @author: 景天
 * @date: 2022/7/29 17:44
 **/

public class AnotherPerson {
    public static void main(String[] args) throws IOException {
        // 创建DatagramSocket对象
        DatagramSocket datagramSocket = new DatagramSocket(9999);
        // 创建发送任务
        // 创建接收任务
        // 创建线程
        // start启动
        new Thread(new SendTask(datagramSocket, "127.0.0.1", 8888)).start();
        new Thread(new ReceiveTask(datagramSocket)).start();
    }
}

package _23network.com.cskaoyan.udp.v4;

import java.io.IOException;
import java.net.DatagramSocket;

/**
 * @description: 一个人
 * @author: 景天
 * @date: 2022/7/29 17:44
 **/

public class OnePerson {
    public static void main(String[] args) throws IOException {
        // 创建DatagramSocket对象
        DatagramSocket datagramSocket = new DatagramSocket(8888);
        // 创建发送任务
        // 创建接收任务
        // 创建线程
        // start启动
        new Thread(new SendTask(datagramSocket, "127.0.0.1", 9999)).start();
        new Thread(new ReceiveTask(datagramSocket)).start();
    }
}
```

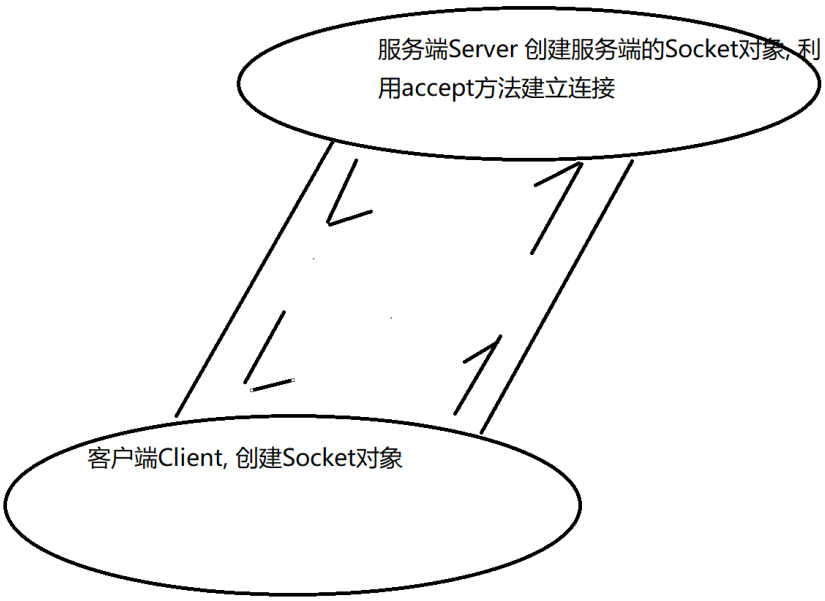
异常

- java.net.BindException: Address already in use: Cannot bind 端口号重复了

TCP编程

传输原理





客户端步骤(Client)

- 创建客户端socket对象(Socket)
- 从socket中获取输入输出流
- 利用输入输出流进行读写操作
- close

服务端步骤(Server)

- 创建服务端的socket对象(ServerSocket)
- 利用accept方法建立连接, 得到socke对象
- 从socket中获取输入输出流

- 利用输入输出流进行读写操作
- close

Socket

此类实现客户端套接字

构造方法

Socket(String host，int port)　　创建一个流套接字并将其连接到指定主机上的指定端口号。

成员方法

InputStream	getInputStream() 返回此套接字的输入流。

OutputStream	getOutputStream() 返回此套接字的输出流。

void	shutdownOutput() 禁用此套接字的输出流。
	socket的半关闭`

ServerSocket

此类实现服务器套接字

构造方法

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

成员方法

Socket	accept() 侦听并接受到此套接字的连接。

案例

v1 客户端发送消息,服务端接收并打印

```
package _23network.com.cskaoyan.tcp.v1;

import java.io.IOException;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * @description: 服务端
 * @author: 景天
```

```
 * @date: 2022/7/30 9:59
 **/

public class Server {
    public static void main(String[] args) throws IOException {
        // - 创建服务端的socket对象(ServerSocket)
        ServerSocket serverSocket = new ServerSocket(8888);
        //- 利用accept方法建立连接, 得到socke对象
        Socket socket = serverSocket.accept();
        //- 从socket中获取输入输出流
        InputStream in = socket.getInputStream();
        //- 利用输入输出流进行读写操作
        byte[] bytes = new byte[1024];
        int readCount = in.read(bytes);
        String s = new String(bytes, 0, readCount);
        InetAddress inetAddress = socket.getInetAddress();
        int port = socket.getPort();
        System.out.println("接收到了来自" + inetAddress + ":" + port + "消息 " +
s);
        //- close
        socket.close();
        serverSocket.close();
    }
}

package _23network.com.cskaoyan.tcp.v1;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;

/**
 * @description: 客户端
 * @author: 景天
 * @date: 2022/7/30 9:59
 **/
/*
客户端发送消息,服务端接收并打印
 */
public class Client {
    public static void main(String[] args) throws IOException {
        // - 创建客户端socket对象(Socket)
        Socket socket = new Socket("127.0.0.1", 8888);
        //- 从socket中获取输入输出流
        OutputStream out = socket.getOutputStream();
        //- 利用输入输出流进行读写操作
        out.write("hello tcp".getBytes());
        //- close
        socket.close();
    }
}
```

v2 多个客户端发送,服务端接收(多线程处理)

```
package _23network.com.cskaoyan.tcp.v2;

import java.io.IOException;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

/**
 * @description: 服务端
 * @author: 景天
 * @date: 2022/7/30 10:09
 **/

public class Server {
    public static void main(String[] args) throws IOException {
        // 创建服务端socket对象
        ServerSocket serverSocket = new ServerSocket(9999);
        // 创建线程池
        ExecutorService pool = Executors.newFixedThreadPool(2);
        // while
        while (true) {
            // accept方法建立连接 得到socket对象
            Socket socket = serverSocket.accept();
            // 放到线程里面去处理
            // new Thread(new ConnectTask(socket)).start();
            pool.submit(new ConnectTask(socket));

        }
    }
}

class ConnectTask implements Runnable{
    // 定义成员变量
    Socket socket;

    public ConnectTask(Socket socket) {
        this.socket = socket;
    }

    @Override
    public void run() {
        // 数据接收
        while (true) {
            try {
                InputStream in = socket.getInputStream();
                byte[] bytes = new byte[1024];
                int readCount = in.read(bytes);
                String s = new String(bytes, 0, readCount);
                InetAddress inetAddress = socket.getInetAddress();
                int port = socket.getPort();
```

```
                System.out.println(Thread.currentThread().getName()+
                        "接收到了来自" + inetAddress + ":" + port + "消息 " + s);
                //- close
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}


package _23network.com.cskaoyan.tcp.v2;

import java.io.IOException;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.Socket;
import java.util.Scanner;

/**
 * @description: 客户端
 * @author: 景天
 * @date: 2022/7/30 10:09
 **/

public class Client {
    public static void main(String[] args) throws IOException {
        // 创建客户端socket对象
        Socket socket = new Socket("127.0.0.1", 9999);
        // 创建scanner对象
        Scanner scanner = new Scanner(System.in);
        // while
        while (true) {
            // 接收数据
            String s = scanner.nextLine();
            // 获取输出流
            OutputStream out = socket.getOutputStream();

            // write数据
            out.write(s.getBytes());

        }

    }
}
```

v3 客户端发送对象(序列化),服务端接收

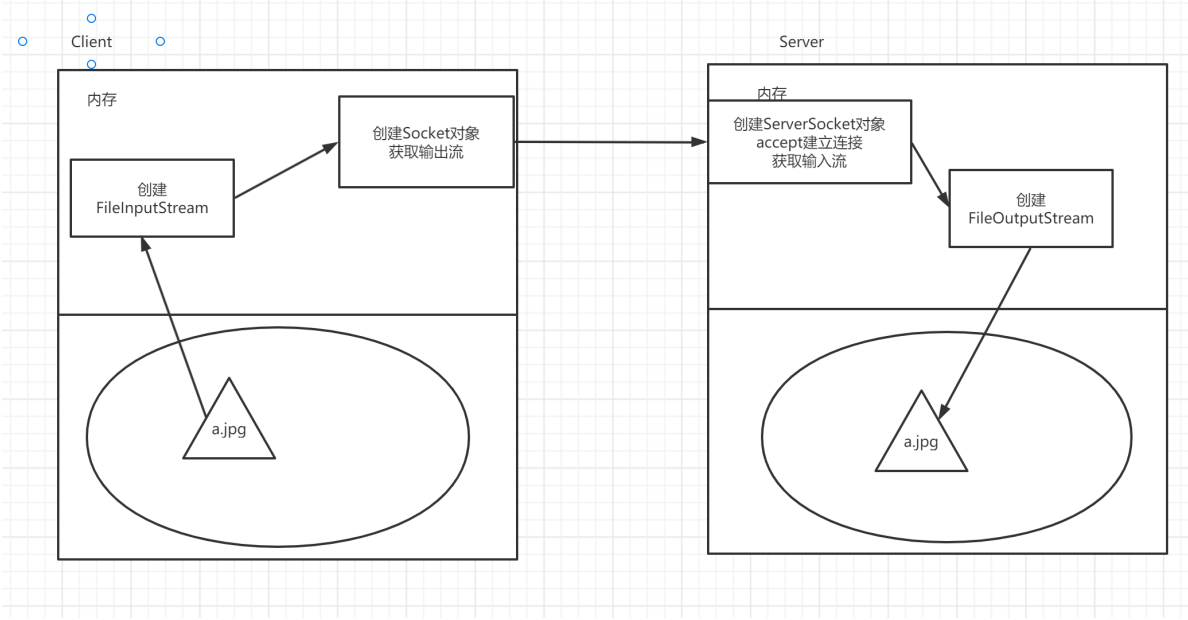
```
package _23network.com.cskaoyan.tcp.v3;

import java.io.IOException;
import java.io.InputStream;
```

```java
import java.io.ObjectInputStream;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * @description: 服务端
 * @author: 景天
 * @date: 2022/7/30 10:52
 **/

public class Server {
    public static void main(String[] args) throws IOException,
ClassNotFoundException {
        // 创建服务端的socket对象
        ServerSocket serverSocket = new ServerSocket(8888);

        // accept方法建立连接 得到socket对象
        Socket socket = serverSocket.accept();
        // 获取输入流
        InputStream inputStream = socket.getInputStream();
        // 对输入流进行包装 对象流
        ObjectInputStream in = new ObjectInputStream(inputStream);
        // readObject()
        Student student = (Student) in.readObject();
        // 打印
        System.out.println(student);
        // close
        socket.close();
        serverSocket.close();
    }
}


package _23network.com.cskaoyan.tcp.v3;

import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.Socket;

/**
 * @description: 客户端
 * @author: 景天
 * @date: 2022/7/30 10:52
 **/

public class Client {
    public static void main(String[] args) throws IOException {
        // 创建客户端Socket对象
        Socket socket = new Socket("127.0.0.1", 8888);
        // 创建学生对象
        Student student = new Student("张三", 20);
        // 获取输出流 OutputStream
        OutputStream outputStream = socket.getOutputStream();
        // ObjectOutputStream对OutputStream进行包装
```

```
        ObjectOutputStream out = new ObjectOutputStream(outputStream);
        // writeObject(对象)
        out.writeObject(student);
        // close
        socket.close();
    }
}
```

v4 客户端上传文件到服务端

思路:




```
package _23network.com.cskaoyan.tcp.v4;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

/**
 * @description: 客户端
 * @author: 景天
 * @date: 2022/7/30 11:11
 **/

public class Client {
    public static void main(String[] args) throws IOException {
        // 创建客户端socket对象
        Socket socket = new Socket("127.0.0.1", 11111);
        // 创建输入流对象
        FileInputStream in = new FileInputStream("D:\\b.txt");
        // 获取输出流
        OutputStream out = socket.getOutputStream();
        // 边读边写
```

```
        int readCount;
        byte[] bytes = new byte[1024];
        while ((readCount = in.read(bytes)) != -1) {
            out.write(bytes,0,readCount);
        }
        // 循环结束 文件发送成功
        System.out.println("while end");
        // 禁用此套接字的输出流
        socket.shutdownOutput();
        // 接收来自服务端的反馈消息
        InputStream inputStream = socket.getInputStream();
        byte[] bytes1 = new byte[1024];
        System.out.println("read before");
        int readCount2 = inputStream.read(bytes1);
        System.out.println("read after");

        System.out.println(new String(bytes1,0,readCount2));
        // close
        in.close();
        socket.close();
    }
}


package _23network.com.cskaoyan.tcp.v4;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * @description: 服务端
 * @author: 景天
 * @date: 2022/7/30 11:11
 **/

public class Server {
    public static void main(String[] args) throws IOException {
        // 创建服务端的socket对象
        ServerSocket serverSocket = new ServerSocket(11111);
        // accept建立连接 得到socket对象
        Socket socket = serverSocket.accept();
        // 创建自己的输出流
        FileOutputStream out = new FileOutputStream("copy_b.txt");
        // 从socket中或取输入流
        InputStream in = socket.getInputStream();

        // 边读边写
        int readCount;
        byte[] bytes = new byte[1024];
        while ((readCount = in.read(bytes)) != -1) {
            out.write(bytes,0,readCount);
```

```
        }
        System.out.println("while end");
        // 保存成功
        // 给客户端反馈消息
        OutputStream outputStream = socket.getOutputStream();
        outputStream.write("file upload successful!".getBytes());

        // close
        out.close();
        socket.close();
        serverSocket.close();
    }
}
```

异常:

- java.net.ConnectException: Connection refused: connect 应该先启动服务端