Java Networking Programming

Wang Yang
wyang AT njnet.edu.cn

Outline

- Basic of Networking Programming
- Java Networking Programming API
- Multi-thread for Java networking Programming

Basic Idioms

- Identity of Host
- Role
- Service
- Connection

Identity of Host

- How to identify a person
    - his identity card : 320102199301022312
        - unique in the world/country
        - can be used in world
        - not random, structured information
        - Hard for being remember and used by human
        - easy for being used by machine

Identity of Host

- How to identify a person
  - his name : 小明
    - unique in the local area
    - used in the class by call “WHO is 小明”
    - easy for human
    - easy but not directly for machine

Identity of Host

- How to identify a person
  - his address + his name : 小明
    - nearly unique in the world
    - used in the world by write to
      - “中国 江苏省 南京市 东南大学 软件学院 小明”
    - easy for human
    - easy but not directly for machine

Identity of Host

- How to identify a Host （主机）
  - IP Address : person’s identity card (58.192.10.2)
    - easy for machine using in the world
  - Hostname : person’s name “ Computer263T1”
    - easy for human remember and use locally
  - Domain Name : person’s address + Name (machine23.cose.seu.edu.cn)
    - easy for human remembering and using in the world

Identity of Host

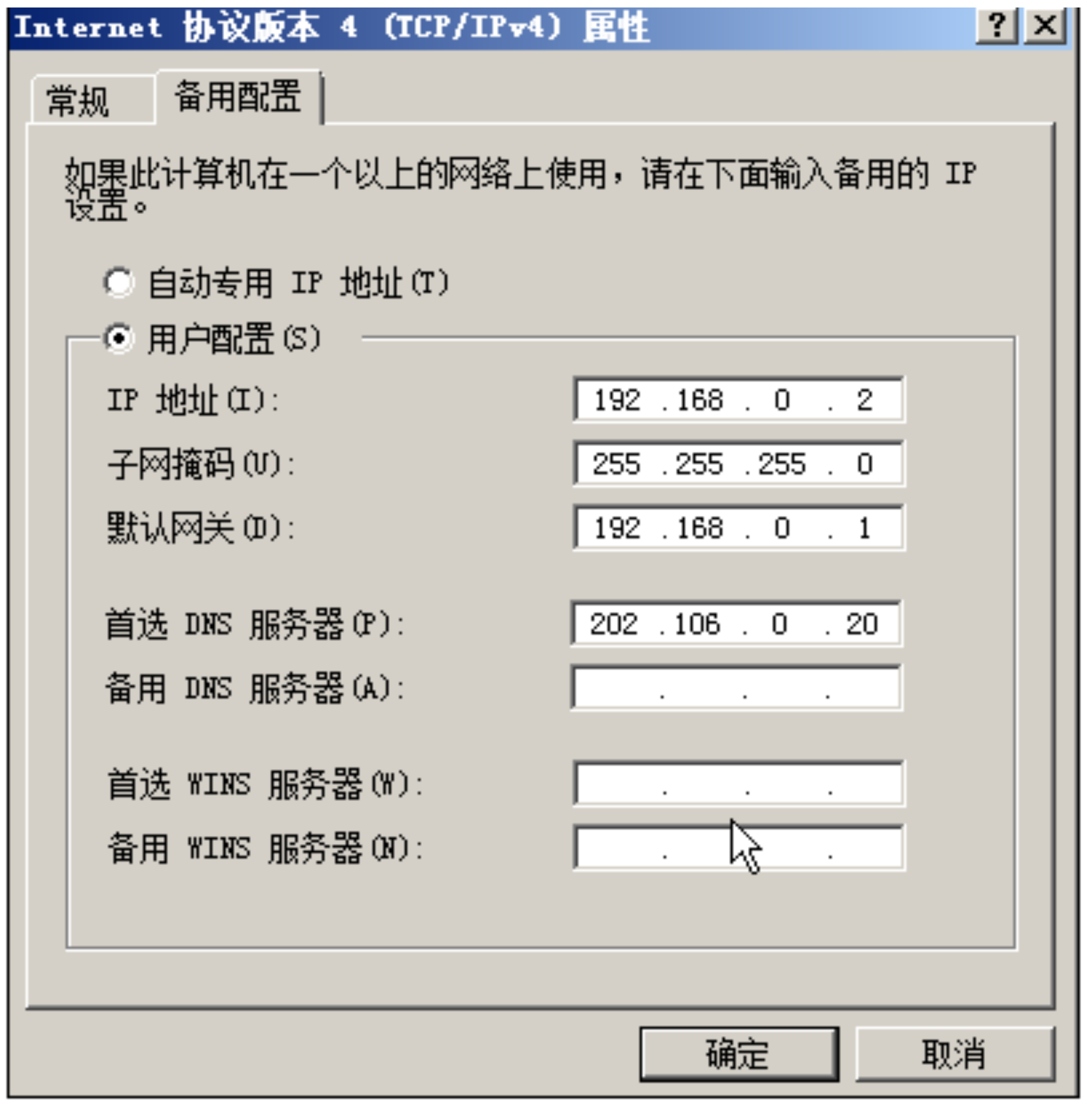
- What is a Host （主机）
  - an equipment, physically or virtually
  - Computer
  - cellphone
  - pad
  - any equipment connected to the Internet
    - car, camera, plane

Identity of Host

- Host’ IP address 58.192.10.2
  - A way of uniquely addressing machines using 32 bit addresses: giving 4 billion possible addresses
  - The Internet consists of a large number of independent sub-networks (subnets 58.192.10)
  - A mechanism for relaying datagrams from one network to another (routing)
  - For routing to work each organization must have a well known prefix (58.192.*.* is in Southeast University)

Identity of Host

- How to know your IP Address

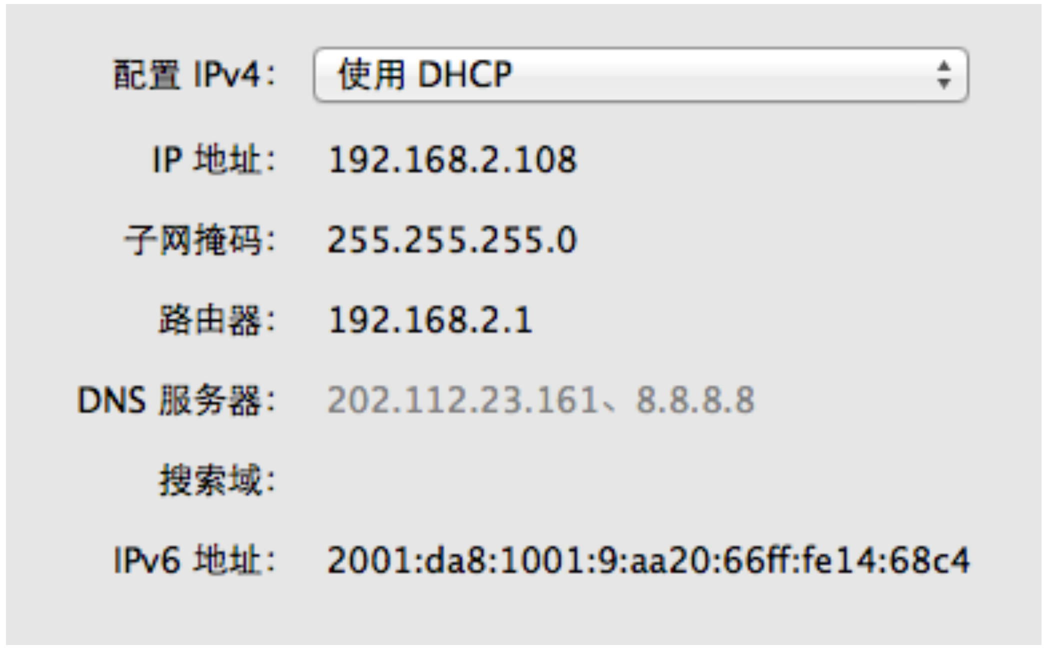


Internet 协议版本 4 (TCP/IPv4) 属性

如果此计算机在一个以上的网络上使用，请在下面输入备用的 IP 设置。

自动专用 IP 地址(T)
[✓ 用户配置(S)]

IP 地址(I):	192 . 168 . 0 . 2
子网掩码(U):	255 . 255 . 255 . 0
默认网关(D):	192 . 168 . 0 . 1
首选 DNS 服务器(P):	202 . 106 . 0 . 20
备用 DNS 服务器(A):	. . .
首选 WINS 服务器(W):	. . .
备用 WINS 服务器(N):	. . .



配置 IPv4:	使用 DHCP
IP 地址:	192.168.2.108
子网掩码:	255.255.255.0
路由器:	192.168.2.1
DNS 服务器:	202.112.23.161、8.8.8.8
搜索域:	
IPv6 地址:	2001:da8:1001:9:aa20:66ff:fe14:68c4

Identity of Host

- Next Generation IP Address
  - The solution is IPv6 which uses 128 bit addresses
  - Improves services such as multicasting and secure communication
  - Several addresses per m2 of the Earth’s surface
  - Not yet widely deployed by ISPs
  - Perhaps widely deployed in 2-4 years
  - Well written Java software should move to IPv6 without modification/recompilation

Identity of Host

- Hostname for a Host (Computer263T1)
  - used in local area network
  - a word combined of letters, number
  - the host will map the hostname to IP Address

Identity of Host

- Domainname for a Host
  - machine23.cose.seu.edu.cn
  - used in world
  - structured information, easy remembered by human
  - the host will map the DomainName to IP Address

Role

- Server
    - www.seu.edu.cn provide seu information services
    - ftp2.seu.edu.cn provide software download services
    - bbs.seu.edu.cn provide a forum for discussing campus affairs
- Client
    - people use services provided by the server
    - all of us

Service

- A host can provide different service
  - 58.192.10.2 can provide
    - Web information www.seu.edu.cn
    - ftp service ftp2.seu.edu.cn
    - smtp service mail.seu.edu.cn
  - we need to distinguish these service in one host
    - we use ports to map the service

Service

- What is a Port
  - a short integer : 0~65535
  - Well-known ports :
    - Web : 80, FTP: 21, SMTP:25
    - image you are calling well-known phone numbers : 114, 119, 120, …

Connection

- What is a Connection
  - A virtual Calling line between two machines
    - two host, two ports
      - 211.63.192.4 45412 — 58.192.10.2 80
    - imaging two person, two phone number
      - 刘欢欢 52090919 — 杨老师 52090904
      - All data will be transfer bidirectional in the connection

Java Networking Programming API

- java.net.*
  - we will learn these Classes and interfaces
  - Java.net.InetAddress
  - java.net.ServerSocket
  - java.net.Socket

InetAddress

- Java has a class java.net.InetAddress which abstracts network addresses
- Serves three main purposes:
  - Encapsulates an address
  - Performs name lookup (converting a host name into an IP address)
  - Performs reverse lookup (converting the address into a host name)

InetAddress

- Abstraction of a network address
  - Currently uses IPv4 (a 32 bit address)
  - Will support other address formats in future
  - Allows an address to be obtained from a host name and vice versa
  - Is immutable (is a read-only object)
  - Create an InetAddress object with the address you need and throw it away when you have finished

InetAddress

- Static construction using a factory method
    - InetAddress getByName(String hostName)
        - hostName can be “host.domain.com.au”, or
        - hostName can be “130.95.72.134”
    - InetAddress getLocalHost()
- Some useful methods:
    - String getHostName()
        - Gives you the host name (for example “www.sun.com”)
    - String getHostAddress()
        - Gives you the address (for example “192.18.97.241”)
    - InetAddress getLocalHost()
    - InetAddress[] getAllByName(String hostName)

InetAddress

```
import java.net.InetAddress;
import java.net.UnknownHostExcepion;

public static void main(String[] args)
{
   try {
      InetAddress inet1 =
            InetAddress.getByName("asp.ee.uwa.edu.au");
      System.out.println(
            "HostAddress=" + inet1.getHostAddress());
      InetAddress inet2 =
            InetAddress.getByName("130.95.72.134");
      System.out.println("HostName=" + inet2.getHostName());
      if (inet1.equals(inet2))
         System.out.println("Addresses are equal");
   }
   catch (UnknownHostException uhe) {
      uhe.printStackTrace();
   }
}
```

Two Types of Socket

- java.net.ServerSocket is used by servers so that they can accept incoming connections
    - A server is a piece of software which advertises and then provides some service on request
- java.net.Socket is used by clients who wish to establish a connection to a (remote) server
    - A client is a piece of software (usually on a different machine) which makes use of some service

ServerSocket

- Listens on well-known port for incoming connections
- Creates a dynamically allocated Socket for each newly established connection
- Maintains a queue to ensure that prospective clients are not lost

ServerSocket

- Construction:
  - ServerSocket(int port, int backlog)
    - Allows up to backlog requests to queue waiting for the server to deal with them
- Some useful methods:
  - Socket accept()
    - Blocks waiting for a client to attempt to establish a connection
  - void close()
    - Called by the server when it is shutting down to ensure that any resources are deallocated
  - More details in the Javadoc (as always!)

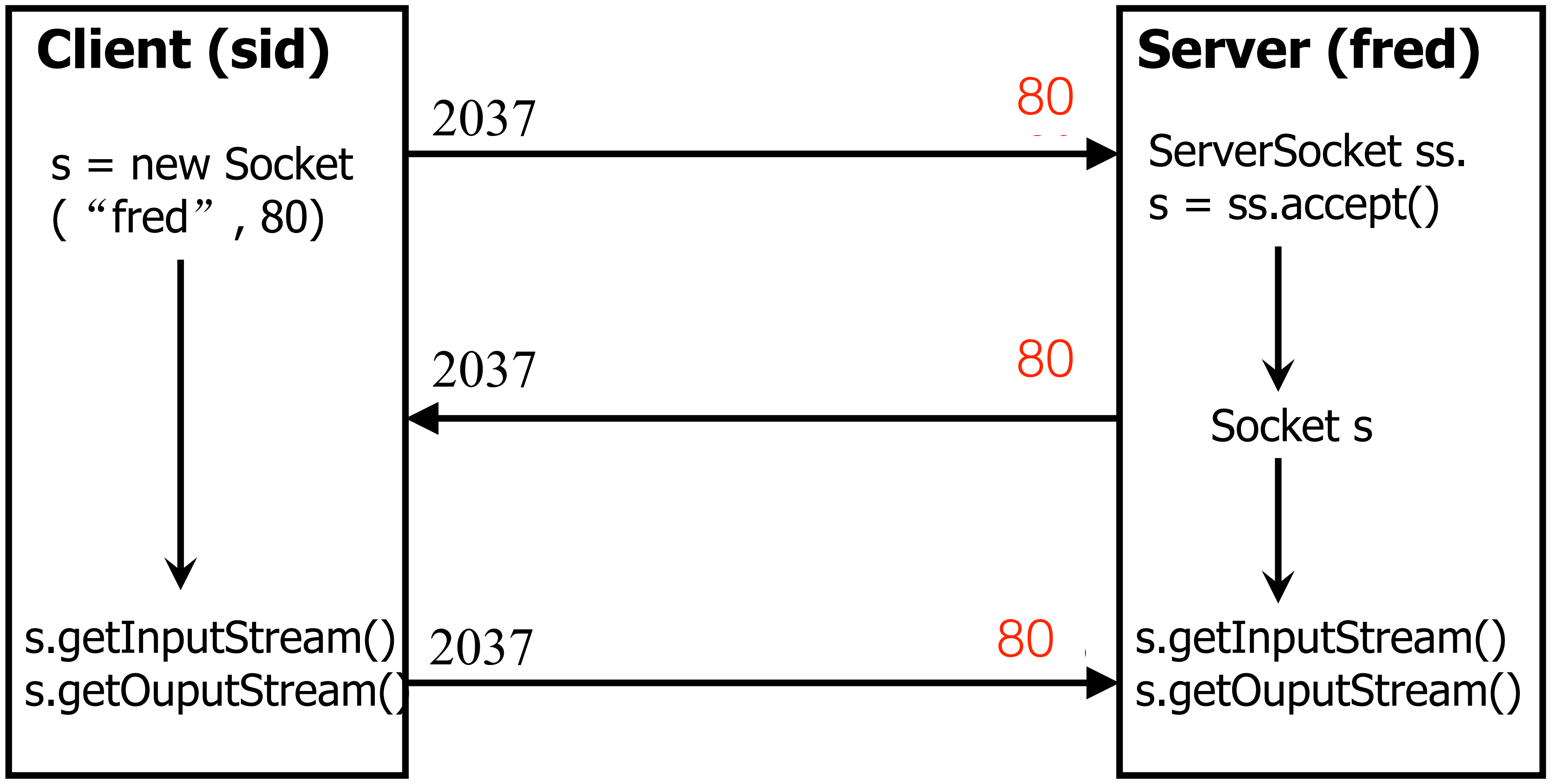
Socket

- Provides access to network streams
- Bi-directional communication between sender and receiver
- Can be used to connect to a remote address and port by using the constructor:
  - Socket(String remoteHost, int port)
- Also used to accept an incoming connection (see ServerSocket)

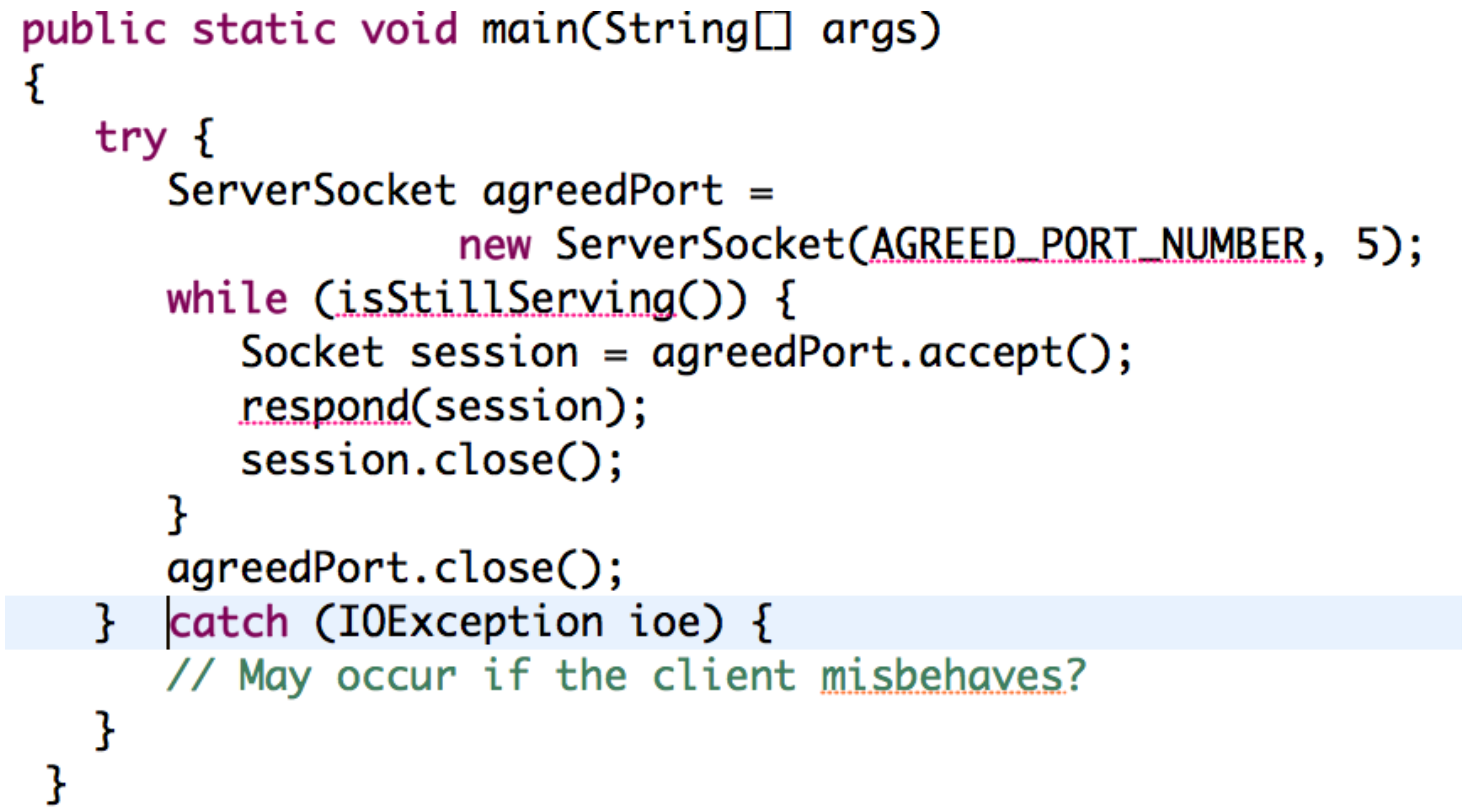
Socket

- Can obtain access to input and output streams
- Input stream allows reception of data from the other party
    - InputSteam getInputStream()
- Output stream allows dispatch of data to the other party
    - OutputStream getOutputStream()

How it all fits together

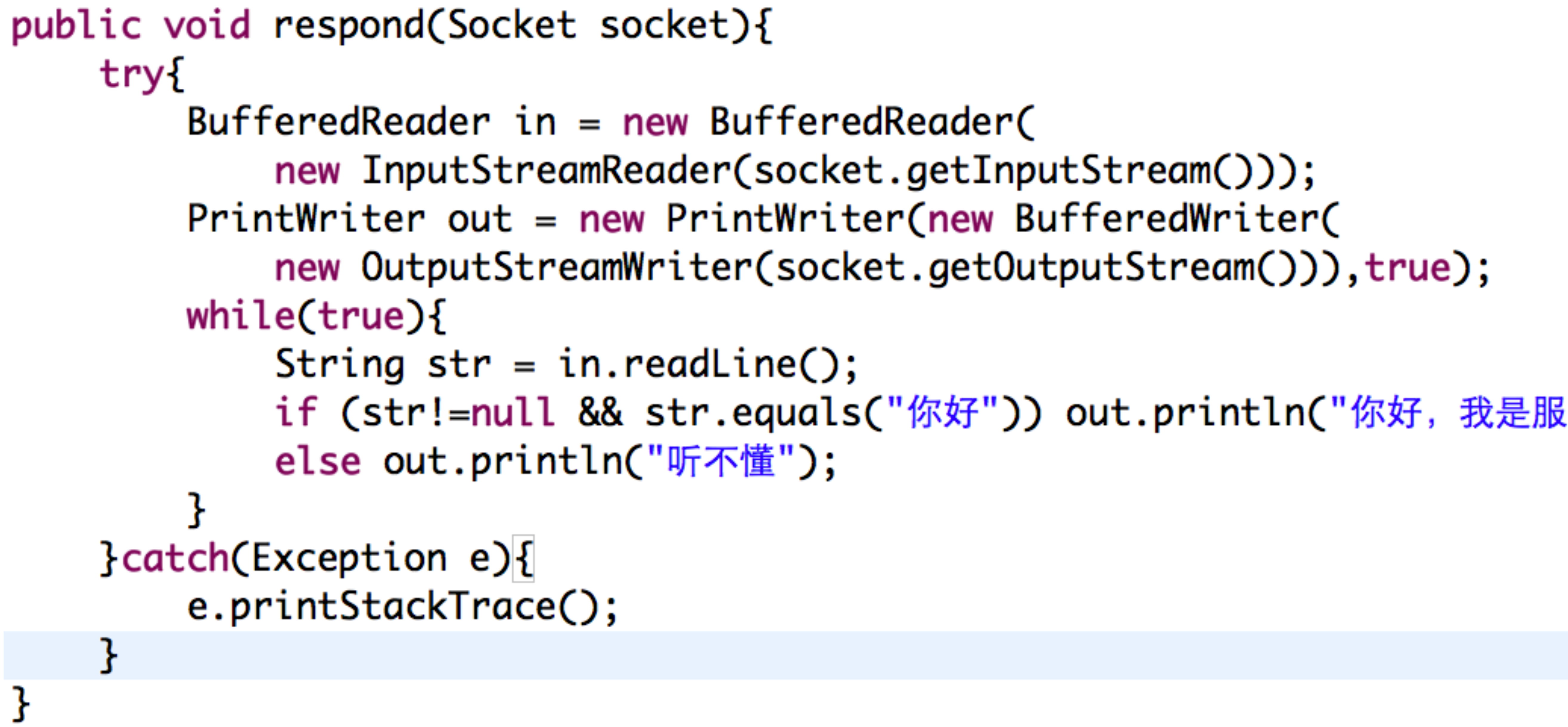


A Simple Server



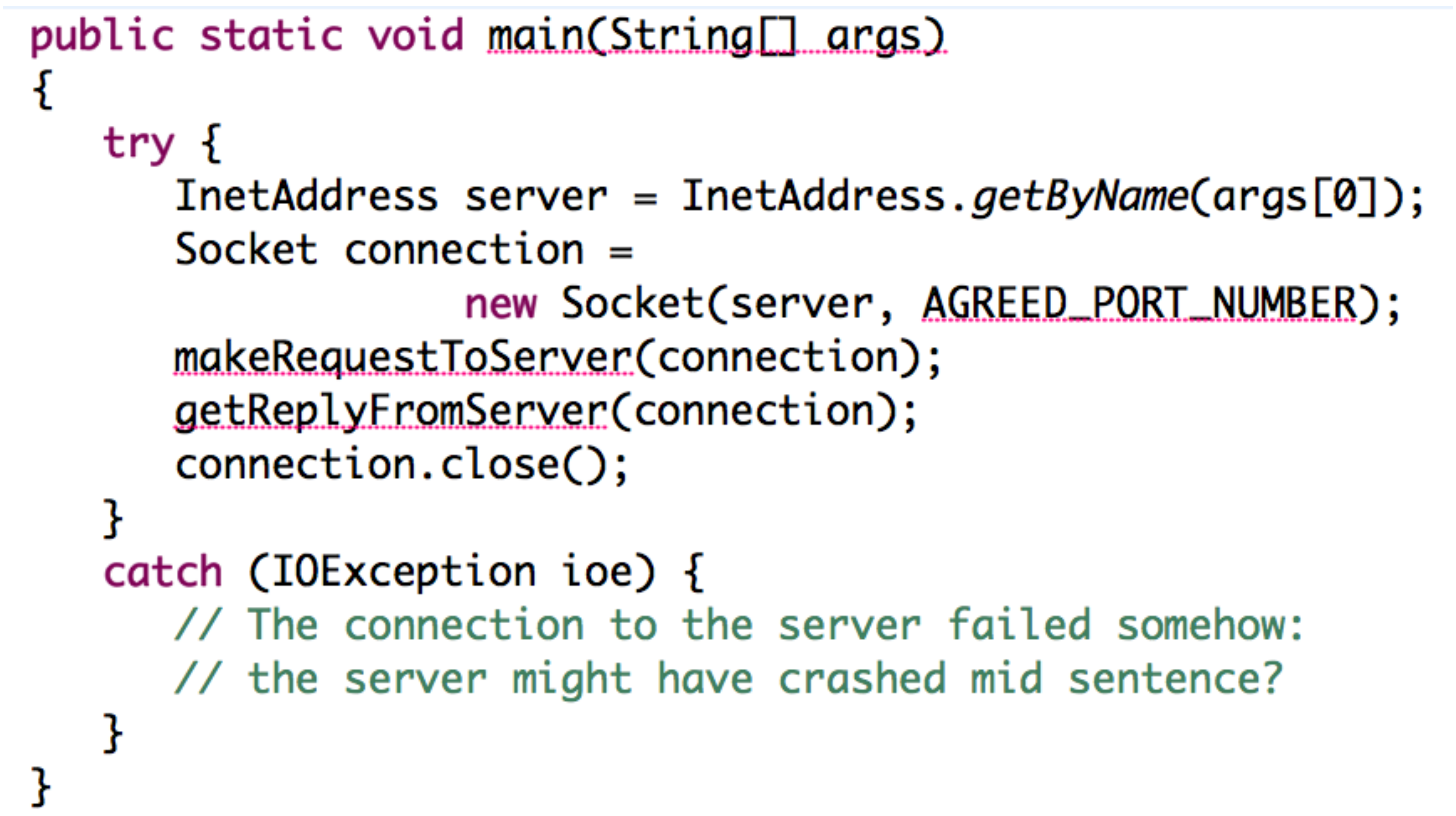
```
public static void main(String[] args)
{
   try {
      ServerSocket agreedPort =
                  new ServerSocket(AGREED_PORT_NUMBER, 5);
      while (isStillServing()) {
         Socket session = agreedPort.accept();
         respond(session);
         session.close();
      }
      agreedPort.close();
   }  catch (IOException ioe) {
      // May occur if the client misbehaves?
   }
 }
```

A Simple Server



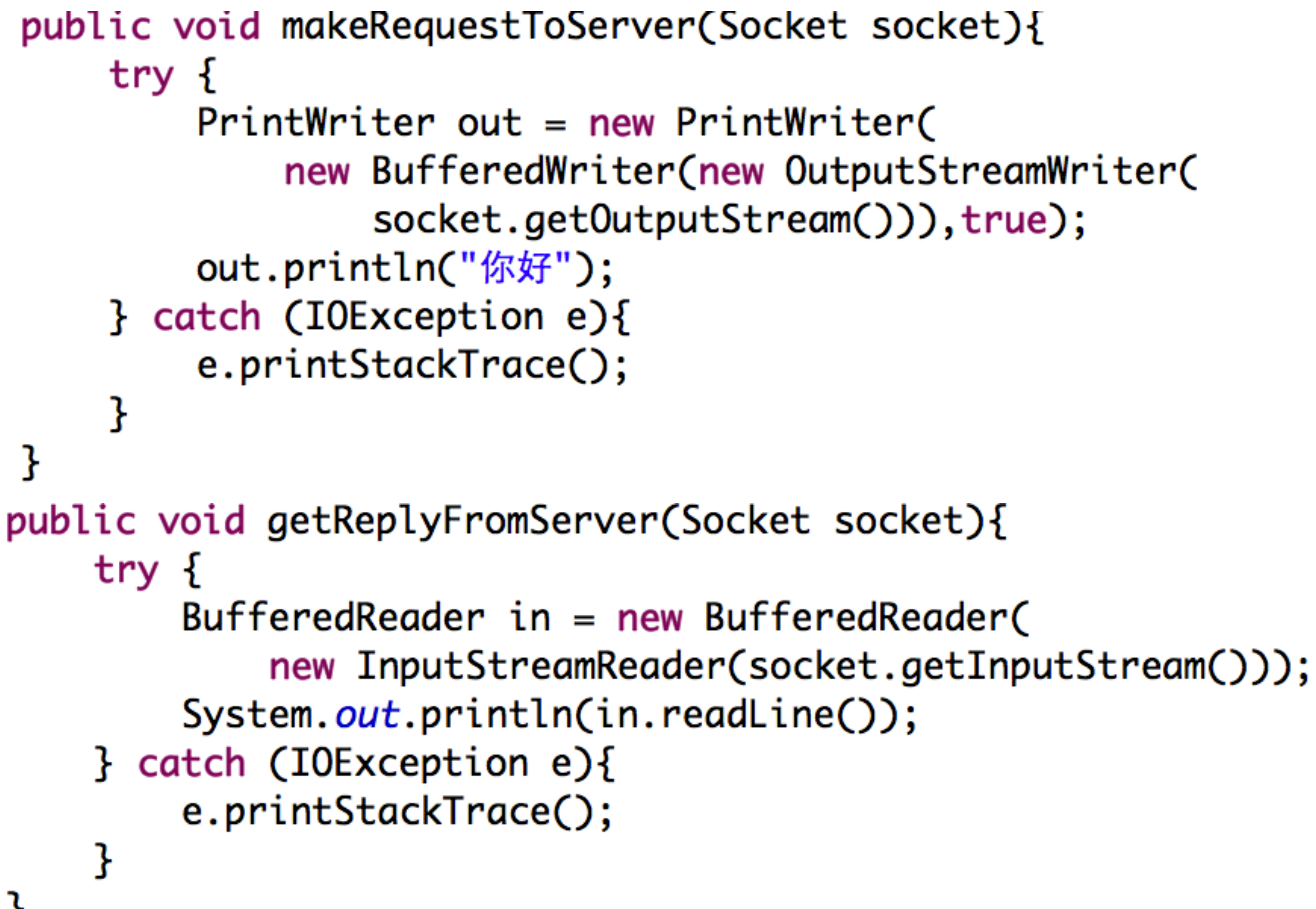
```java
public void respond(Socket socket){
    try{
        BufferedReader in = new BufferedReader(
            new InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(new BufferedWriter(
            new OutputStreamWriter(socket.getOutputStream())),true);
        while(true){
            String str = in.readLine();
            if (str!=null && str.equals("你好")) out.println("你好，我是服
            else out.println("听不懂");
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

A Simple Client



```java
public static void main(String[] args)
{
    try {
        InetAddress server = InetAddress.getByName(args[0]);
        Socket connection =
                    new Socket(server, AGREED_PORT_NUMBER);
        makeRequestToServer(connection);
        getReplyFromServer(connection);
        connection.close();
    }
    catch (IOException ioe) {
        // The connection to the server failed somehow:
        // the server might have crashed mid sentence?
    }
}
```

A Simple Client

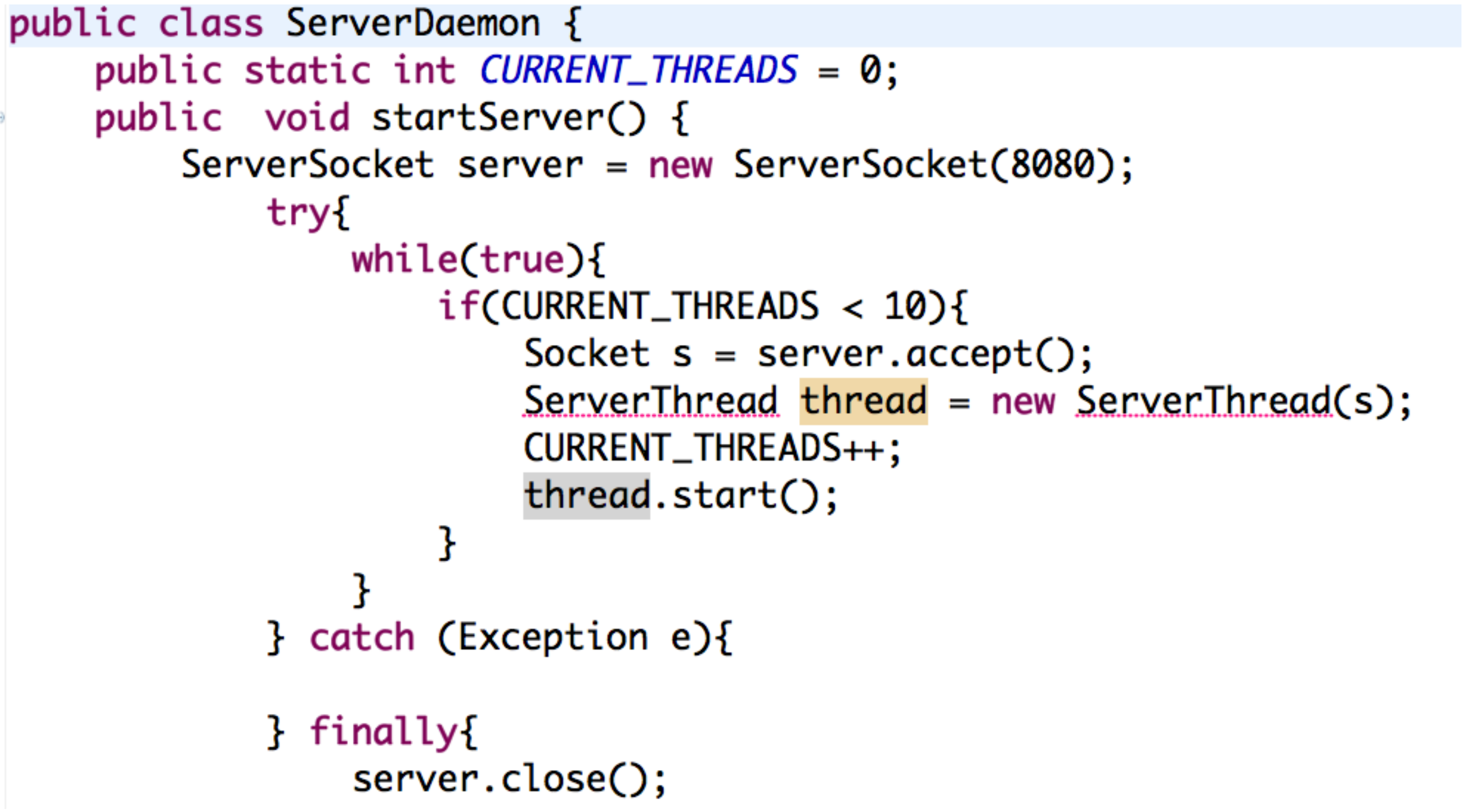


```java
public void makeRequestToServer(Socket socket){
    try {
        PrintWriter out = new PrintWriter(
            new BufferedWriter(new OutputStreamWriter(
                socket.getOutputStream())),true);
        out.println("你好");
    } catch (IOException e){
        e.printStackTrace();
    }
}
public void getReplyFromServer(Socket socket){
    try {
        BufferedReader in = new BufferedReader(
            new InputStreamReader(socket.getInputStream()));
        System.out.println(in.readLine());
    } catch (IOException e){
        e.printStackTrace();
    }
}
```

Problems with the simple server

- If one people wan’t quit the connection:
  - the ServerSocket will be blocked by this guy
  - backlog’ people will wait in the queue
  - other people will be dropped
  - we can use multithread to solve this problem

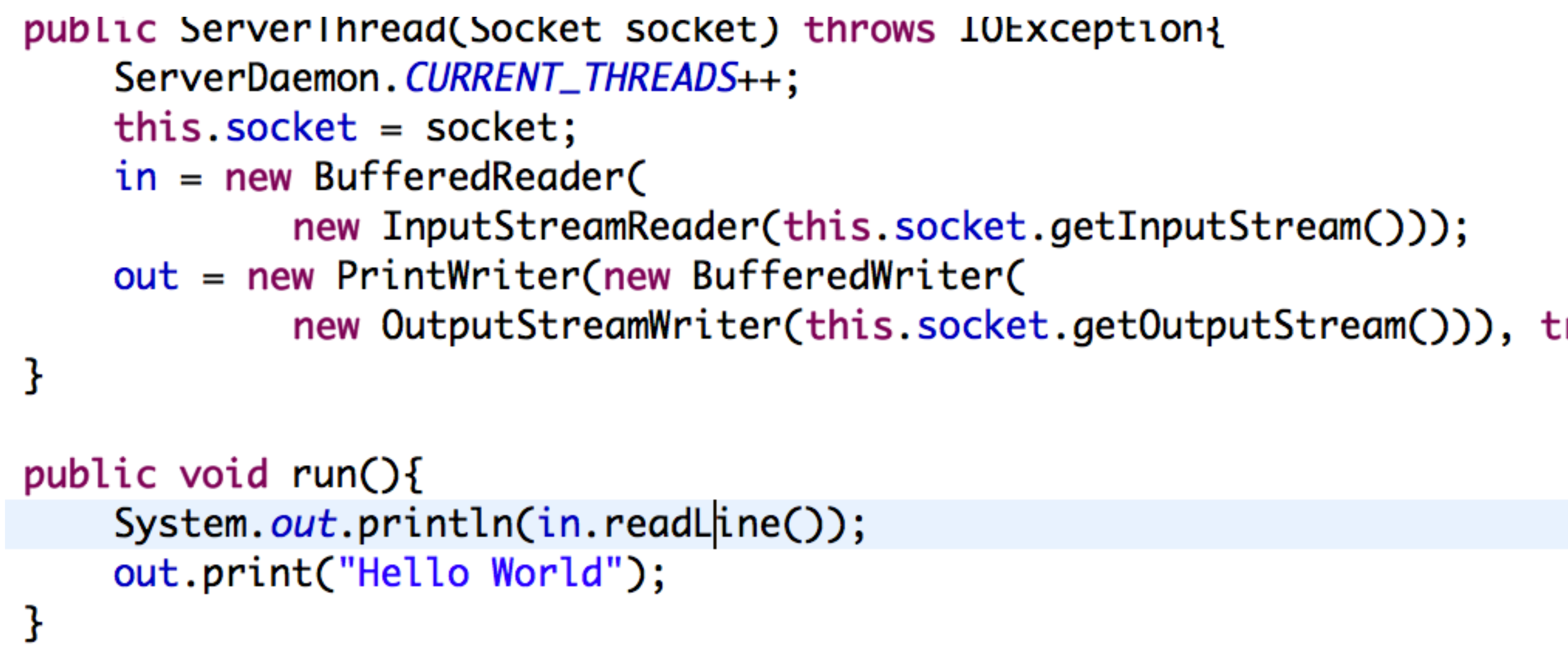
Mutli-Thread Server



```
public class ServerDaemon {
    public static int CURRENT_THREADS = 0;
    public  void startServer() {
        ServerSocket server = new ServerSocket(8080);
            try{
                while(true){
                    if(CURRENT_THREADS < 10){
                        Socket s = server.accept();
                        ServerThread thread = new ServerThread(s);
                        CURRENT_THREADS++;
                        thread.start();
                    }
                }
            } catch (Exception e){

            } finally{
                server.close();
```

Multi-Thread Server



```
public ServerThread(Socket socket) throws IOException{
    ServerDaemon.CURRENT_THREADS++;
    this.socket = socket;
    in = new BufferedReader(
            new InputStreamReader(this.socket.getInputStream()));
    out = new PrintWriter(new BufferedWriter(
            new OutputStreamWriter(this.socket.getOutputStream())), tr
}

public void run(){
    System.out.println(in.readLine());
    out.print("Hello World");
}
```