目录

Day17.Java

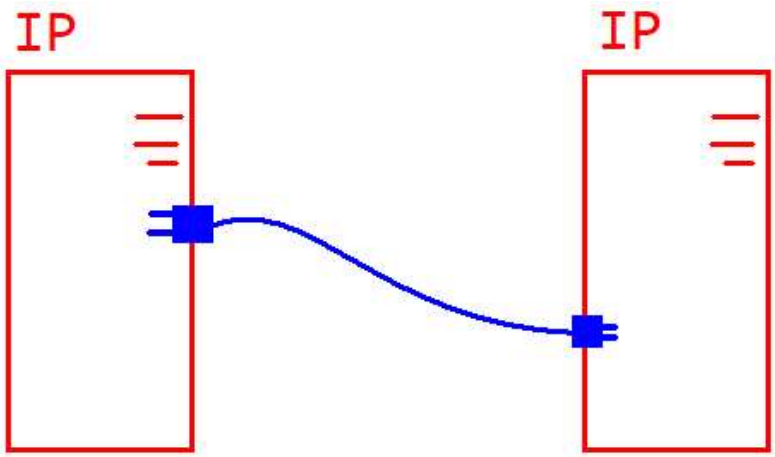
1 网络

- win+r，输入cmd

  ipconfig

  ping 192.168.12.xxx
  如果ping不通，让他把防火墙关掉

- Socket - 网络套接字
- 两台主机用IP地址可以找到对方
- 两台主机各选择一个端口，在端口上通信



1.1 ServerSocket

- 在服务器端，选择一个端口
- 在端口上启动服务，等待客户端发起连接
- 创建对象
  - new ServerSocket(端口号)
- 方法

- accept()
  暂停，等待客户端发起连接，并建立连接通道
  返回一个 Socket 对象，表示连接通道，插在服务器端的插头

- close()
  停止服务，释放占用的端口

1.2 Socket

- 表示连接通道两端的插头对象
- 客户端
  - new Socket(ip, 端口号)
- 方法
  - getInputStream()
  - getOutputStream()
    获得Socket连接中的两个方向的流

  - close()
    断开连接

  - setSoTimeout(毫秒值)
    设置接收数据等待超时时长
    超时会出现 SocketTimeoutException

网络

项目：day1701_网络
类：day1701.Server1
Client1

Server1

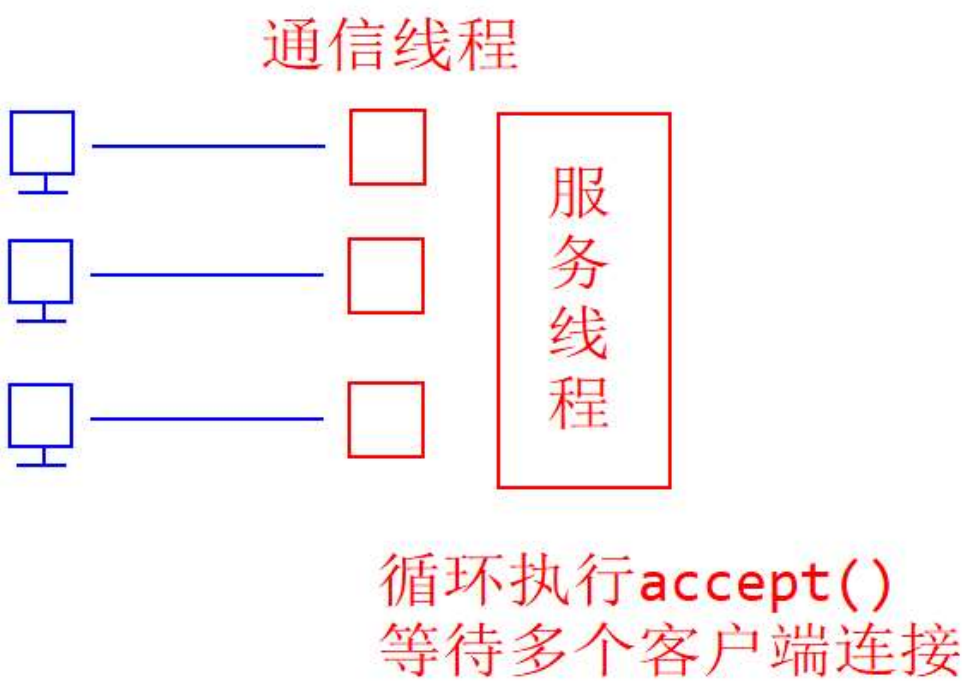
```
package day1701;

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

public class Server1 {
   public static void main(String[] args) throws Exception {
      //选择端口，启动服务
      ServerSocket ss = new ServerSocket(8000);
      System.out.println("服务已启动");

      //等待客户端发起连接，建立连接通道
      //并得到通道的服务器端插头
      Socket s = ss.accept();
      System.out.println("客户端已连接");
      //连接通道中，双向的流
      InputStream in = s.getInputStream();
      OutputStream out = s.getOutputStream();
      /* *)通信协议
       *     *) 通信流程
       *     *) 数据格式
       *  1) 从客户端接收，"hello" "aaa" "bbb" "ccc" ....
       *  2) 向客户端发送，"world" */
```

```
        for(int i=0;i<5;i++) {
            char c = (char) in.read();
            System.out.print(c);
        }
        out.write("world".getBytes());
        out.flush();
        s.close();
        ss.close();
    }
}
```

Client1

```
package day1701;

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

public class Client1 {
    public static void main(String[] args) throws Exception {
        //新建Socket对象
        //客户端的插头
        Socket s = new Socket("127.0.0.1", 8000);
        //取出流
        InputStream in = s.getInputStream();
        OutputStream out = s.getOutputStream();
        //1.发"hello"  2.收"world"
        out.write("hello".getBytes());
        out.flush();
        for(int i=0;i<5;i++) {
            char c = (char) in.read();
            System.out.print(c);
        }
        s.close();
    }
}
```

1.3 服务器端线程模型



回声服务

*）支持多个客户端同时连接
*）客户端发送的数据，服务器收到后，原封不动的发回

项目：day1702_回声服务
类：day1702.EchoServer
EchoClient

EchoServer

```
package day1702;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class EchoServer {
   /*
    * 用来启动服务器
    * 启动一个“服务线程”，
    * 来循环等待多个客户端连接
    */
   public void start() {
      new Thread() {
         @Override
         public void run() {
            try {
               //在8000端口上启动服务
               ServerSocket ss = new ServerSocket(8000);
               System.out.println("服务器已启动");
               //循环等待客户端连接
               while(true) {
                  Socket s = ss.accept();
                  //启动通信线程，对连接通道s执行通信过程
```

```
                    TongXinThread t = new TongXinThread(s);
                    t.start();
                }
            } catch (Exception e) {
                System.out.println(
                 "服务无法在8000端口上启动，或者服务已停止");
                e.printStackTrace();
            }
        }
    }.start();
  }

  class TongXinThread extends Thread {
    Socket s;
    public TongXinThread(Socket s) {
      this.s = s;
    }

    @Override
    public void run() {
      try {
                                        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream(), "UTF-8"));
          PrintWriter out = new PrintWriter(new OutputStreamWriter(s.getOutputStream(),
"UTF-8"));
          String line;
          while((line = in.readLine()) != null) {
            out.println(line);
            out.flush();
          }
          //网络断开
        } catch (Exception e) {
          //网络断开
        }
        //网络断开
        System.out.println("一个客户端已断开");
    }
  }

  public static void main(String[] args) {
    EchoServer s = new EchoServer();
    s.start();
  }

}
```

EchoClient

```
package day1702;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
```

```
import java.net.UnknownHostException;
import java.util.Scanner;

public class EchoClient {
   public static void main(String[] args) throws Exception {
      Socket s = new Socket("127.0.0.1", 8000);
      BufferedReader in = new BufferedReader(
       new InputStreamReader(
       s.getInputStream(), "UTF-8"));
      PrintWriter out = new PrintWriter(
       new OutputStreamWriter(
       s.getOutputStream(), "UTF-8"));
      while(true) {
         System.out.println("输入: ");
         String msg = new Scanner(System.in).nextLine();
         out.println(msg);
         out.flush();
         msg = in.readLine();
         System.out.println("收到: "+msg);
      }

   }
}
```

聊天室

项目: day1703_聊天室
类: day1703.ChatServer
ChatClient

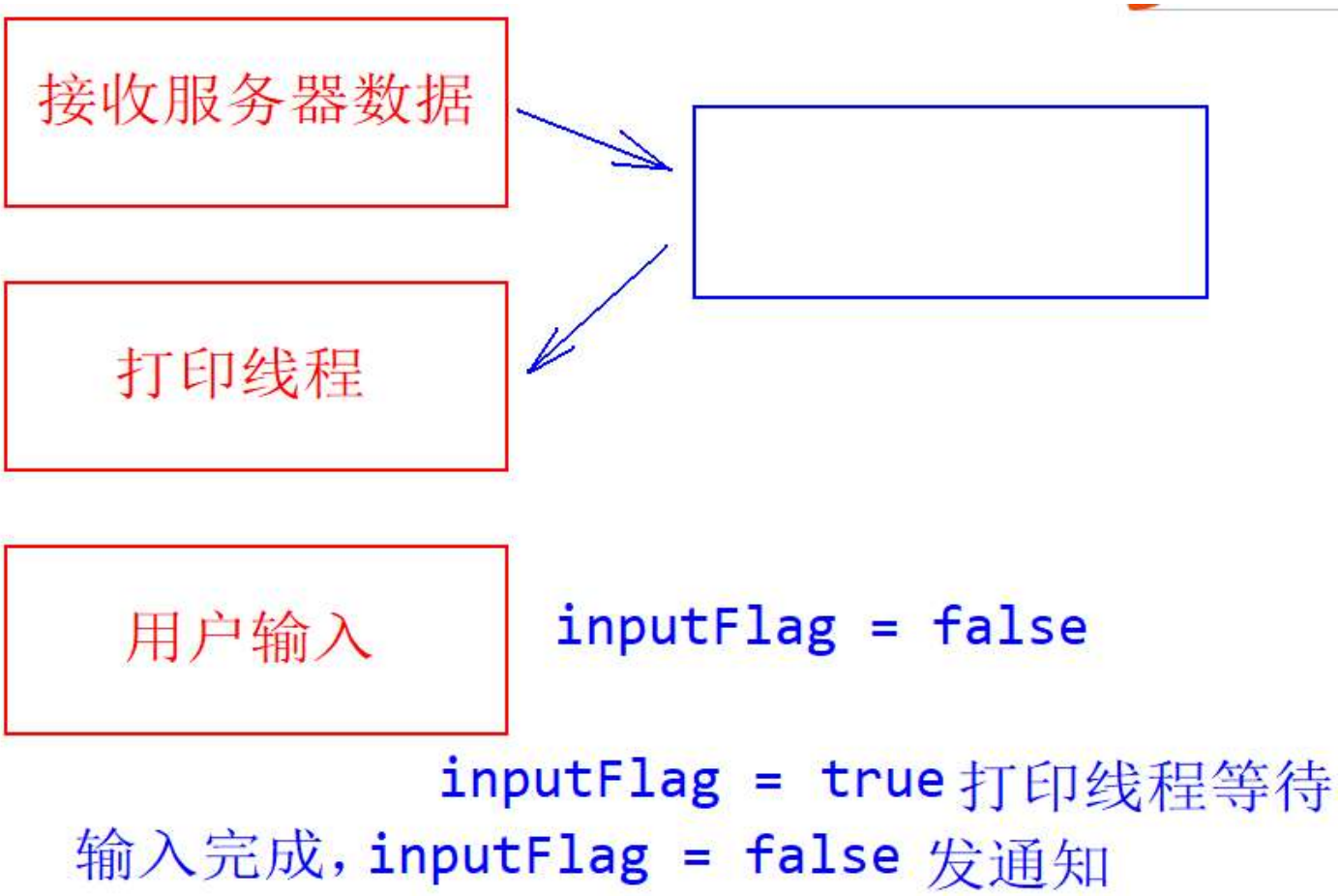
ChatServer

```
package day1703;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.net.SocketTimeoutException;
import java.util.ArrayList;
import java.util.List;

public class ChatServer {
   private List<TongXinThread> list = new ArrayList<>();

   public void start() {
      new Thread() {
         @Override
         public void run() {
            try {
                ServerSocket ss =
                   new ServerSocket(8000);
```

```
                System.out.println("聊天室服务器已经启动");
                while(true) {
                   Socket s = ss.accept();
                   TongXinThread t =
                    new TongXinThread(s);
                   t.start();
                   synchronized (list) {
                      list.add(t);
                   }
                }
             } catch (Exception e) {
                System.out.println(
                 "服务无法在8000端口上启动，或服务已结束");
             }
          }
       }.start();
    }

    class TongXinThread extends Thread {
       private Socket s;
       BufferedReader in;
       PrintWriter out;
       String name;
       public TongXinThread(Socket s) {
          this.s = s;
          try {
             s.setSoTimeout(3000);
          } catch (SocketException e) {
          }
       }

       public void send(String msg) {
          out.println(msg);
          out.flush();
       }

       public void sendAll(String msg) {
          synchronized (list) {
             for (TongXinThread t : list) {
                t.send(msg);
             }
          }
       }

       @Override
       public void run() {
          try {
             in = new BufferedReader(
              new InputStreamReader(
              s.getInputStream(), "UTF-8"));
             out = new PrintWriter(
              new OutputStreamWriter(
              s.getOutputStream(), "UTF-8"));

             name = in.readLine();//先接收客户端的昵称
             send(name+", 欢迎进入激情聊天室！");//发送欢迎信息
             //群发上线消息
             synchronized (list) {
                sendAll(name+"进入了聊天室，在线人数："
                      +list.size());
             }

             int count = 0;
```

```
            String line;
            while(true) {
                try {
                   line = in.readLine();
                   count = 0;
                } catch (SocketTimeoutException e) {
                   count++;
                   if(count == 3) {
                      send("您已经被踢出聊天室! ");
                      s.close();
                      break;
                   }
                   send("**** 请积极参与激情聊天 "+count+"/3 ****");
                   continue;
                }
                if(line == null) break;

                sendAll(name+"说:  "+line);

                String ip =
                 ((InetSocketAddress)
                   s.getRemoteSocketAddress()).getHostString();

                System.out.println(name+"("+ip+")说:  "+line);
            }
            //断开
         } catch (Exception e) {
            //断开
         }
         //断开
         synchronized (list) {
            list.remove(this);
            sendAll(name+"离开了聊天室, 在线人数: "
                   +list.size());
         }
      }
   }

   public static void main(String[] args) {
      ChatServer server = new ChatServer();
      server.start();
   }

}
```

ChatClient




```
package day1703;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.LinkedList;
import java.util.Scanner;

public class ChatClient {
   Socket s;
   BufferedReader in;
   PrintWriter out;
   LinkedList<String> list = new LinkedList<>();
   boolean inputFlag = false;//没有输入

   public void start() {
      try {
         s = new Socket("192.168.12.108", 8000);
         in = new BufferedReader(
          new InputStreamReader(
          s.getInputStream(), "UTF-8"));
         out = new PrintWriter(
          new OutputStreamWriter(
          s.getOutputStream(), "UTF-8"));

         System.out.print("输入昵称: ");
         String name = new Scanner(System.in).nextLine();
         out.println(name);
         out.flush();
         //从服务器接收数据
         new Thread() {
            public void run() {
```

```
                receive();
            }
        }.start();
        //让用户输入聊天内容
        new Thread() {
            public void run() {
                input();
            }
        }.start();
        //打印线程
        new Thread() {
            public void run() {
                print();
            }
        }.start();
    } catch (Exception e) {
        System.out.println("无法连接服务器");
        e.printStackTrace();
    }
}

protected void print() {
    while(true) {
        synchronized (list) {
            while(list.size() == 0 || inputFlag) {
                try {
                    list.wait();
                } catch (InterruptedException e) {
                }
            }
            String msg = list.removeFirst();
            System.out.println(msg);
        }
    }
}
//在接收线程中执行
protected void receive() {
    try {
        String line;
        while((line = in.readLine()) != null) {
            synchronized (list) {
                list.add(line);
                list.notifyAll();
            }
        }
        //断开
    } catch (Exception e) {
        //断开
    }
    //断开
    System.out.println("已经与服务器断开连接");
}

//在输入线程中执行
protected void input() {
    System.out.println("**** 按回车输入聊天内容 ****");
    while(true) {
        new Scanner(System.in).nextLine();
        inputFlag = true;

        System.out.print("输入聊天内容: ");
        String s = new Scanner(System.in).nextLine();
        out.println(s);
```

```
            out.flush();

            inputFlag = false;
            synchronized (list) {
               list.notifyAll();
            }
         }
      }


      public static void main(String[] args) {
         ChatClient c = new ChatClient();
         c.start();
      }
   }
```

2 作业

- 重写
  - 回声，或聊天室