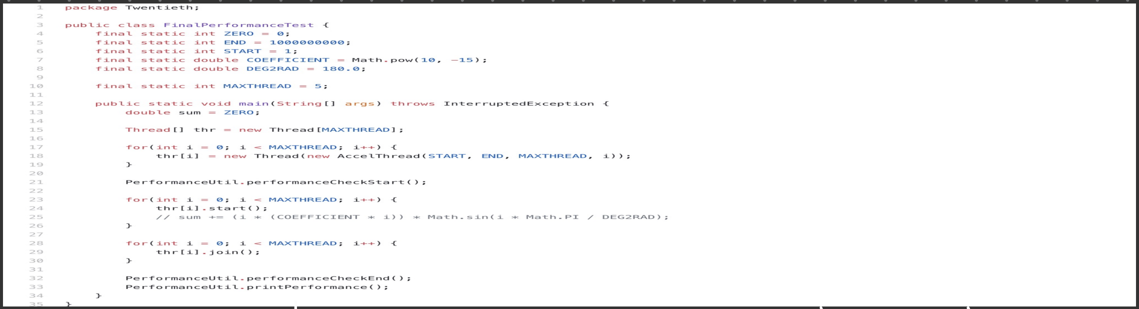
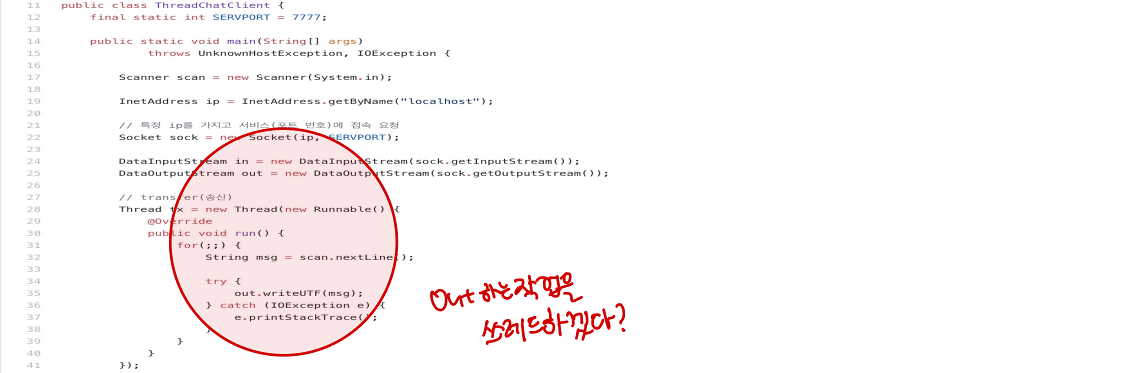
1 FinalPerformanceTest 에서 28번째 줄의 join()을 쓰는 이유가 실행되고있는 Thread가 끝나고 투입을 하라는 뜻인가요?



```
1   package Twentieth;
2
3   public class FinalPerformanceTest {
4       final static int ZERO = 0;
5       final static int END = 1000000000;
6       final static int START = 1;
7       final static double COEFFICIENT = Math.pow(10, -15);
8       final static double DEG2RAD = 180.0;
9
10      final static int MAXTHREAD = 5;
11
12      public static void main(String[] args) throws InterruptedException {
13          double sum = ZERO;
14
15          Thread[] thr = new Thread[MAXTHREAD];
16
17          for(int i = 0; i < MAXTHREAD; i++) {
18              thr[i] = new Thread(new AccelThread(START, END, MAXTHREAD, i));
19          }
20
21          PerformanceUtil.performanceCheckStart();
22
23          for(int i = 0; i < MAXTHREAD; i++) {
24              thr[i].start();
25              // sum += (i * (COEFFICIENT * i)) * Math.sin(i * Math.PI / DEG2RAD);
26          }
27
28          for(int i = 0; i < MAXTHREAD; i++) {
29              thr[i].join();
30          }
31
32          PerformanceUtil.performanceCheckEnd();
33          PerformanceUtil.printPerformance();
34      }
35  }
```

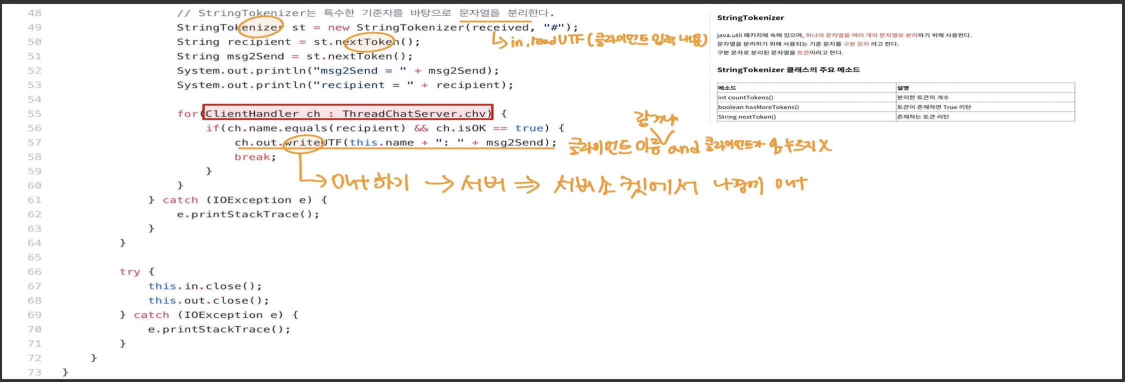
2. ThreadClient에서 28번째줄 Thread 할당부분 해석 한번더 부탁드리고, 소괄호 안에다 Runnable을 넣는 구조가 잘이해가안됩니다.



```
11  public class ThreadChatClient {
12      final static int SERVPORT = 7777;
13
14      public static void main(String[] args)
15              throws UnknownHostException, IOException {
16
17          Scanner scan = new Scanner(System.in);
18
19          InetAddress ip = InetAddress.getByName("localhost");
20
21          // 특정 ip를 가지고 서비스(포트 번호)에 접속 요청
22          Socket sock = new Socket(ip, SERVPORT);
23
24          DataInputStream in = new DataInputStream(sock.getInputStream());
25          DataOutputStream out = new DataOutputStream(sock.getOutputStream());
26
27          // transfer(송신)
28          Thread tx = new Thread(new Runnable() {
29              @Override
30              public void run() {
31                  for(;;) {
32                      String msg = scan.nextLine();
33
34                      try {
35                          out.writeUTF(msg);
36                      } catch (IOException e) {
37                          e.printStackTrace();
38                      }
39                  }
40              }
41          });
```

Out 하는작업을 스레드하겠다?

3. 아래의 빨간 박스가 무엇을 뜻하는지 한번더 알려주시면 감사하겠습니다.



```
48                  // StringTokenizer는 특수한 기준자를 바탕으로 문자열을 분리한다.
49                  StringTokenizer st = new StringTokenizer(received, "#");
50                  String recipient = st.nextToken();
51                  String msg2Send = st.nextToken();
52                  System.out.println("msg2Send = " + msg2Send);
53                  System.out.println("recipient = " + recipient);
54
55                  for(ClientHandler ch : ThreadChatServer.chv) {
56                      if(ch.name.equals(recipient) && ch.isOK == true) {
57                          ch.out.writeUTF(this.name + ": " + msg2Send);
58                          break;
59                      }
60                  }
61              } catch (IOException e) {
62                  e.printStackTrace();
63              }
64          }
65
66          try {
67              this.in.close();
68              this.out.close();
69          } catch (IOException e) {
70              e.printStackTrace();
71          }
72      }
73  }
```

↳ in.readUTF (클라이언트 입력 내용)

같거나

클라이언트 이름 and 클라이언트가 ok 누르지 X

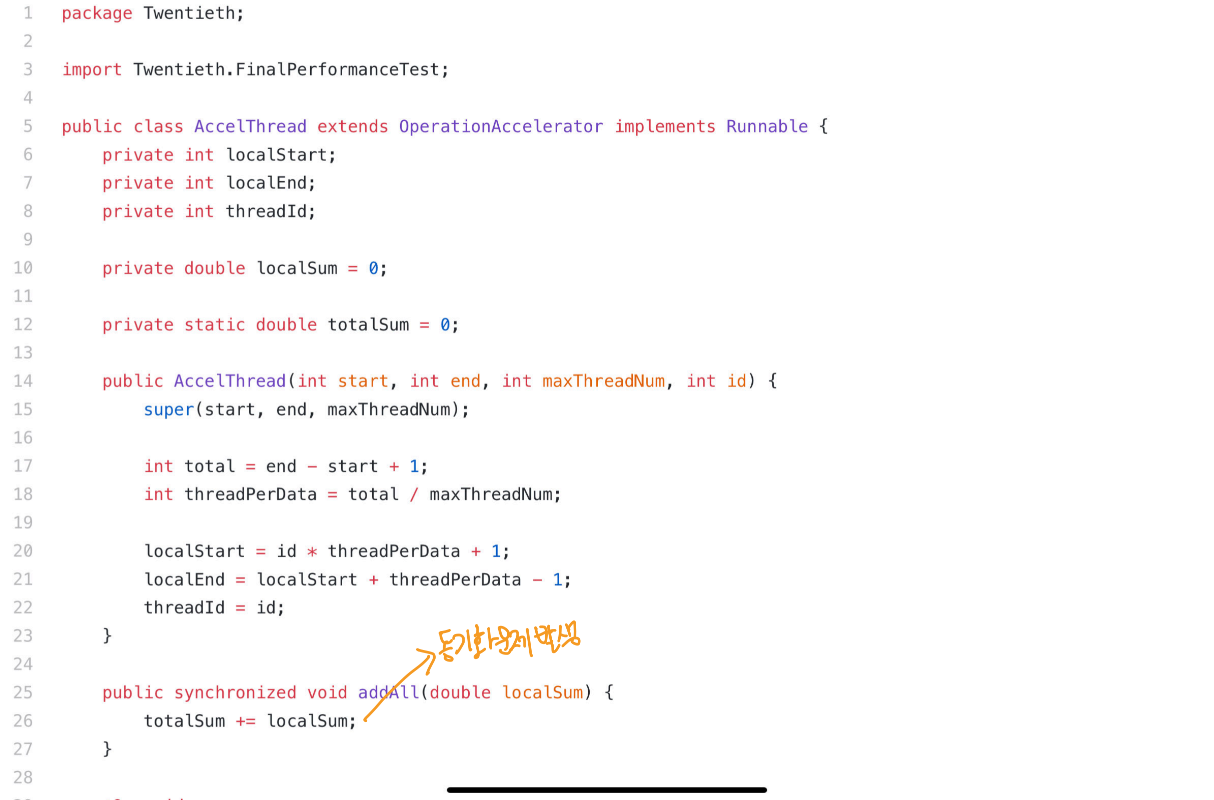
↳ Out하기 → 서버 ⇒ 서버소켓에서 나중에 out

StringTokenizer

java.util 패키지에 속해 있으며, 하나의 문자열을 여러 개의 문자열로 분리하기 위해 사용한다.
문자열을 분리하기 위해 사용되는 기준 문자를 구분 문자 라고 한다.
구분 문자로 분리된 문자열을 토큰이라고 한다.

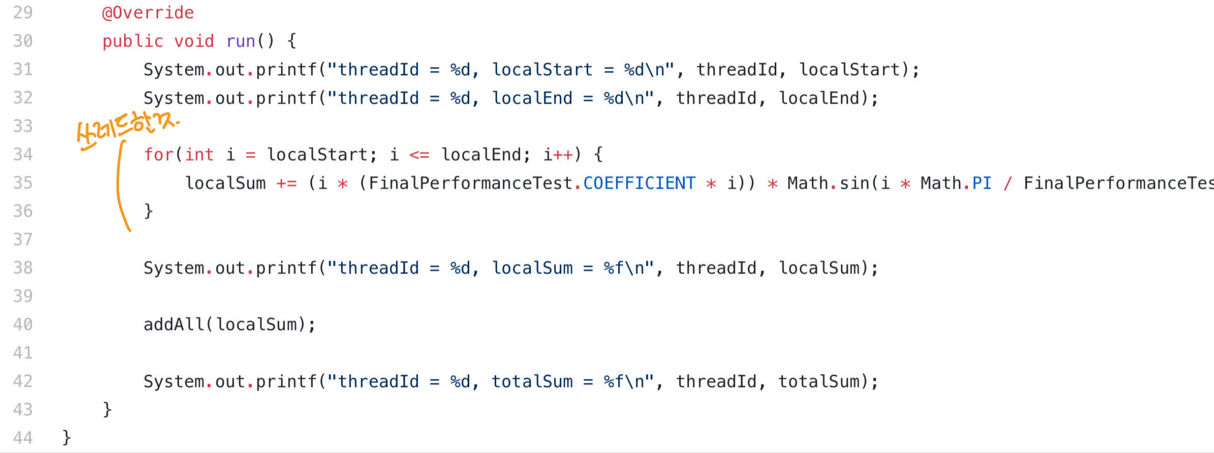
StringTokenizer 클래스의 주요 메소드

메소드	설명
int countTokens()	분리한 토큰의 개수
boolean hasMoreTokens()	토큰이 존재하면 True 리턴
String nextToken()	존재하는 토큰 리턴



```java
package Twentieth;

import Twentieth.FinalPerformanceTest;

public class AccelThread extends OperationAccelerator implements Runnable {
    private int localStart;
    private int localEnd;
    private int threadId;

    private double localSum = 0;

    private static double totalSum = 0;

    public AccelThread(int start, int end, int maxThreadNum, int id) {
        super(start, end, maxThreadNum);

        int total = end - start + 1;
        int threadPerData = total / maxThreadNum;

        localStart = id * threadPerData + 1;
        localEnd = localStart + threadPerData - 1;
        threadId = id;
    }

    public synchronized void addAll(double localSum) {
        totalSum += localSum;
    }

```

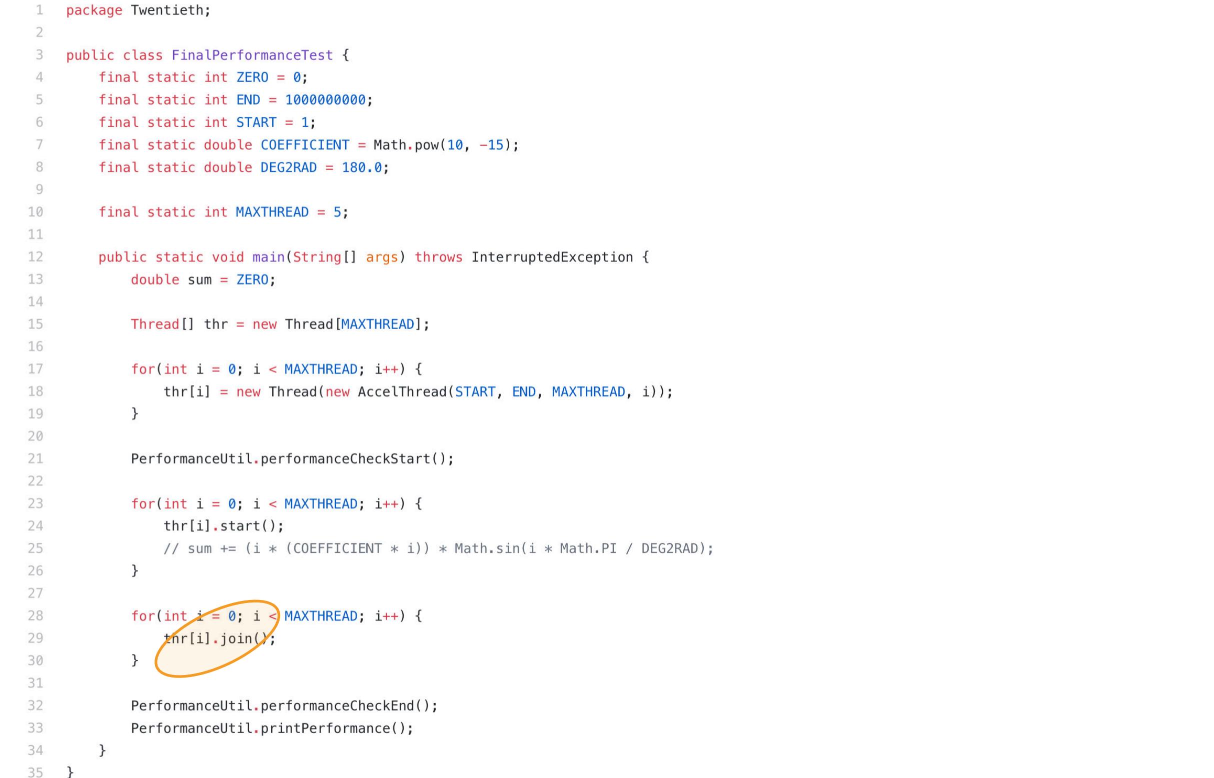
동기화문제발생



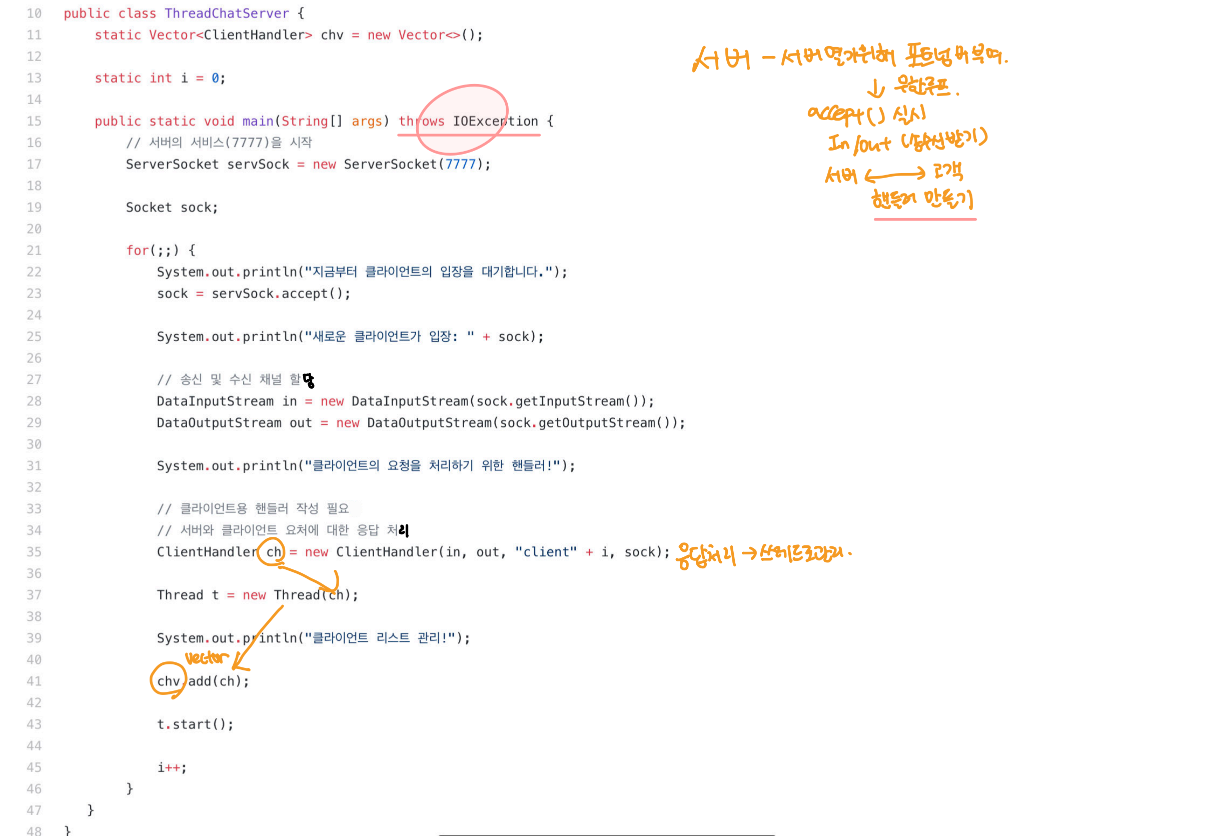
```java
    @Override
    public void run() {
        System.out.printf("threadId = %d, localStart = %d\n", threadId, localStart);
        System.out.printf("threadId = %d, localEnd = %d\n", threadId, localEnd);
```

스레드한것.

```java
        for(int i = localStart; i <= localEnd; i++) {
            localSum += (i * (FinalPerformanceTest.COEFFICIENT * i)) * Math.sin(i * Math.PI / FinalPerformanceTes
        }

        System.out.printf("threadId = %d, localSum = %f\n", threadId, localSum);

        addAll(localSum);

        System.out.printf("threadId = %d, totalSum = %f\n", threadId, totalSum);
    }
}
```



```java
package Twentieth;

public class FinalPerformanceTest {
    final static int ZERO = 0;
    final static int END = 1000000000;
    final static int START = 1;
    final static double COEFFICIENT = Math.pow(10, -15);
    final static double DEG2RAD = 180.0;

    final static int MAXTHREAD = 5;

    public static void main(String[] args) throws InterruptedException {
        double sum = ZERO;

        Thread[] thr = new Thread[MAXTHREAD];

        for(int i = 0; i < MAXTHREAD; i++) {
            thr[i] = new Thread(new AccelThread(START, END, MAXTHREAD, i));
        }

        PerformanceUtil.performanceCheckStart();

        for(int i = 0; i < MAXTHREAD; i++) {
            thr[i].start();
            // sum += (i * (COEFFICIENT * i)) * Math.sin(i * Math.PI / DEG2RAD);
        }

        for(int i = 0; i < MAXTHREAD; i++) {
            thr[i].join();
        }

        PerformanceUtil.performanceCheckEnd();
        PerformanceUtil.printPerformance();
    }
}
```

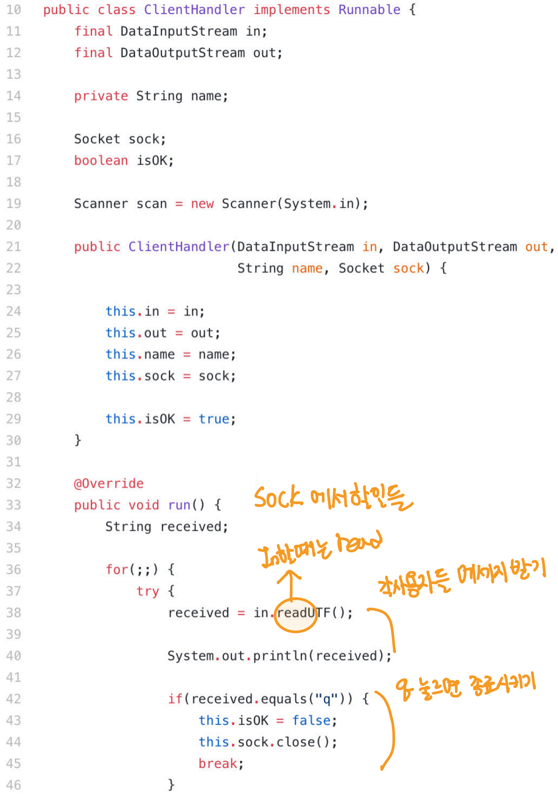


```java
public class ThreadChatServer {
    static Vector<ClientHandler> chv = new Vector<>();

    static int i = 0;

    public static void main(String[] args) throws IOException {
        // 서버의 서비스(7777)을 시작
        ServerSocket servSock = new ServerSocket(7777);

        Socket sock;

        for(;;) {
            System.out.println("지금부터 클라이언트의 입장을 대기합니다.");
            sock = servSock.accept();

            System.out.println("새로운 클라이언트가 입장: " + sock);

            // 송신 및 수신 채널 할당
            DataInputStream in = new DataInputStream(sock.getInputStream());
            DataOutputStream out = new DataOutputStream(sock.getOutputStream());

            System.out.println("클라이언트의 요청을 처리하기 위한 핸들러!");

            // 클라이언트용 핸들러 작성 필요
            // 서버와 클라이언트 요처에 대한 응답 처리
            ClientHandler ch = new ClientHandler(in, out, "client" + i, sock);

            Thread t = new Thread(ch);

            System.out.println("클라이언트 리스트 관리!");

            chv.add(ch);

            t.start();

            i++;
        }
    }
}
```

서버 - 서버 열기위해 포트넘버부여.
↓ 무한루프.
accept() 실시
In/out (송수신받기)
서버 ⟷ 고객
핸들러 만들기

응답처리 → 쓰레드로관리.

vector



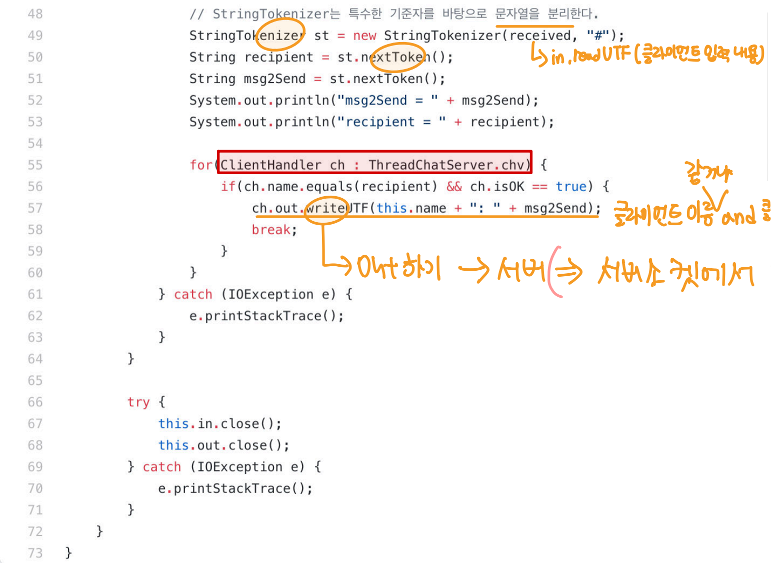
```java
public class ClientHandler implements Runnable {
    final DataInputStream in;
    final DataOutputStream out;

    private String name;

    Socket sock;
    boolean isOK;

    Scanner scan = new Scanner(System.in);

    public ClientHandler(DataInputStream in, DataOutputStream out,
                         String name, Socket sock) {

        this.in = in;
        this.out = out;
        this.name = name;
        this.sock = sock;

        this.isOK = true;
    }

    @Override
    public void run() {
        String received;

        for(;;) {
            try {
                received = in.readUTF();

                System.out.println(received);

                if(received.equals("q")) {
                    this.isOK = false;
                    this.sock.close();
                    break;
                }
```

sock 에서 할일들

In할때는 read

각사용자들 메세지 받기

q 눌르면 종료시키기



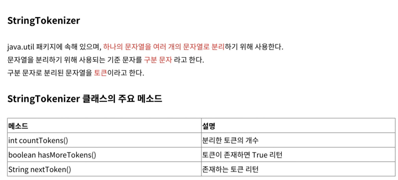
```
                // StringTokenizer는 특수한 기준자를 바탕으로 문자열을 분리한다.
                StringTokenizer st = new StringTokenizer(received, "#");
                String recipient = st.nextToken();
                String msg2Send = st.nextToken();
                System.out.println("msg2Send = " + msg2Send);
                System.out.println("recipient = " + recipient);

                for(ClientHandler ch : ThreadChatServer.chv) {
                    if(ch.name.equals(recipient) && ch.isOK == true) {
                        ch.out.writeUTF(this.name + ": " + msg2Send);
                        break;
                    }
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        try {
            this.in.close();
            this.out.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

↳ in.readUTF (클라이언트 입력 내용)

같거나

클라이언트 이름 and 클라이언트가 ok 누르지 X

→ 아웃하기 → 서버 (⇒ 서버소켓에서 나중에 아웃)

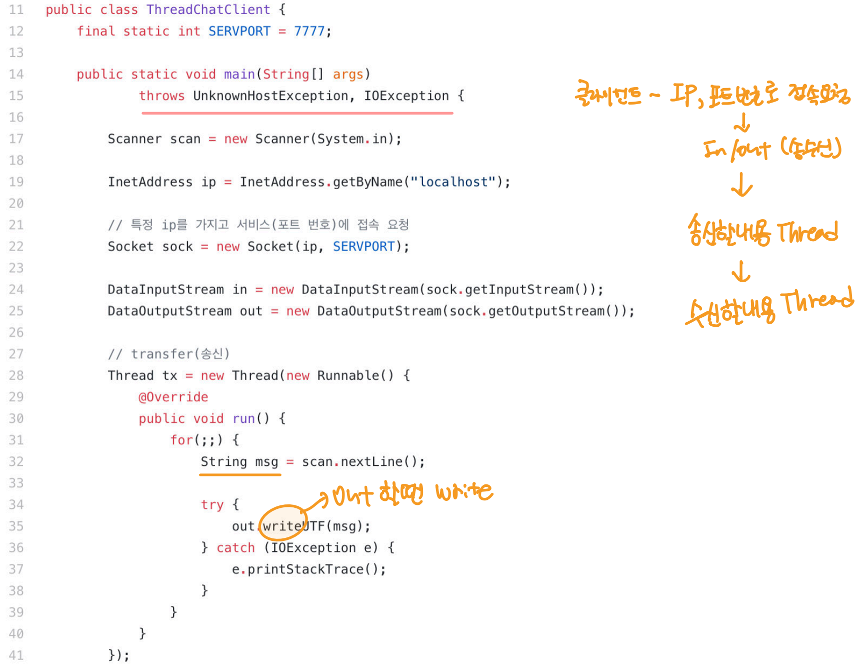


StringTokenizer

java.util 패키지에 속해 있으며, 하나의 문자열을 여러 개의 문자열로 분리하기 위해 사용한다.
문자열을 분리하기 위해 사용되는 기준 문자를 구분 문자 라고 한다.
구분 문자로 분리된 문자열을 토큰이라고 한다.

StringTokenizer 클래스의 주요 메소드

메소드	설명
int countTokens()	분리한 토큰의 개수
boolean hasMoreTokens()	토큰이 존재하면 True 리턴
String nextToken()	존재하는 토큰 리턴



```java
public class ThreadChatClient {
    final static int SERVPORT = 7777;

    public static void main(String[] args)
            throws UnknownHostException, IOException {

        Scanner scan = new Scanner(System.in);

        InetAddress ip = InetAddress.getByName("localhost");

        // 특정 ip를 가지고 서비스(포트 번호)에 접속 요청
        Socket sock = new Socket(ip, SERVPORT);

        DataInputStream in = new DataInputStream(sock.getInputStream());
        DataOutputStream out = new DataOutputStream(sock.getOutputStream());

        // transfer(송신)
        Thread tx = new Thread(new Runnable() {
            @Override
            public void run() {
                for(;;) {
                    String msg = scan.nextLine();

                    try {
                        out.writeUTF(msg);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
```

out 하면 write

클라이언트 ~ IP, 포트번호로 접속요청

↓

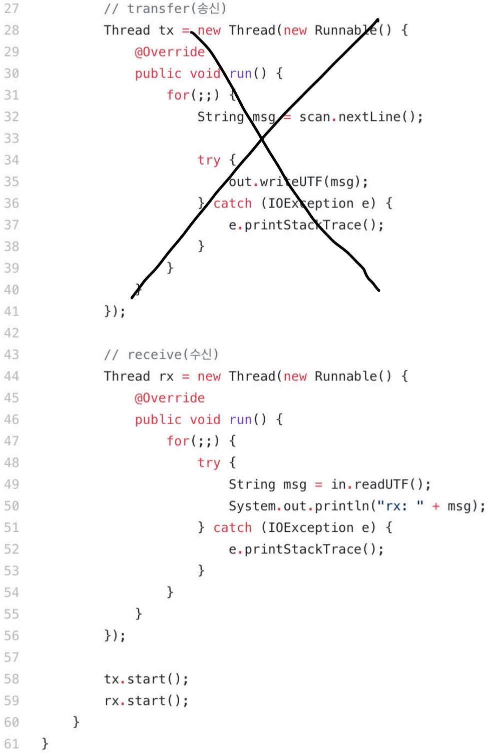
In/out (송수신)

↓

송신할내용 Thread

↓

수신할내용 Thread



```
        // transfer(송신)
        Thread tx = new Thread(new Runnable() {
            @Override
            public void run() {
                for(;;) {
                    String msg = scan.nextLine();

                    try {
                        out.writeUTF(msg);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        });

        // receive(수신)
        Thread rx = new Thread(new Runnable() {
            @Override
            public void run() {
                for(;;) {
                    try {
                        String msg = in.readUTF();
                        System.out.println("rx: " + msg);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        });

        tx.start();
        rx.start();
    }
}
```