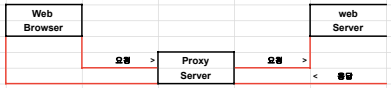
***Proxy 서버**



1. 필터링 → 접근 제어
2. 로깅 (logging)
3. 캐싱(caching) → 임시 보관

자체 대기열이라는 배열이 있음.
클라이언트의 요청을 대기열에 넣고 한개씩 한개씩 처리한다.
방식은 "큐(FIFO)" 방식
new ServerSocket(포트번호, 대기열크기)
대기열 => 50개 default
keyScan.nextLine(); 서버 대기열에서 대기
PID = 프로세스 id

println -> 운영체제에서 제공하는 기능을 호출하는 것 :

ServerSocket ss = new ServerSocket(8888, 2);
2 = 대기열의 숫자 설정하기

---

1) 소켓 생성
Socket socket = new Socket();

2) 연결할 서버의 주소를 준비
SocketAddress socketAddress = new InetSocketAddress("192.168.0.76", 8888);
> 추상클래스 레퍼런스 = 추상클래스를 상속 받은 서브클래스의 객체 주소를 담겠다.
> InetSocketAddress(서브클래스)

3) 서버와의 연결을 시도
> socket.connect(socketAddress, 3000);
연결 : connect // 3000 = 타임아웃

---

local&host loop back

App

Os NIC Hub

GateWay

***버퍼 사용시 입출력**

---

*TCP

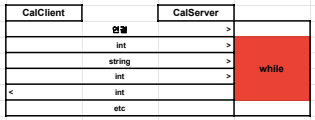
data 데이터를 전송하기 쉬운 크기로 잘라서 패킷으로 포장한다.
↓
Packet(패킷)
TCP/IP → 패킷 전송 →
↓

- 보내는이의 주소
- 받는이의 주소
- 데이터 길이
- 체크섬
- etc

TCP는 손실될 가능성이 있음.

---

*stateful 1



---

*stateful 2 : 서버측에 계산결과 유지

서버 ↔ 클라이언트

result = 0

사용

result

0

>클라이언트 연결이 유지하는 동안에는 result 변수도 유지된다.
> 즉, Client와의 작업 정보를 유지할 수 있다

---

*stateful 3 : 스레드

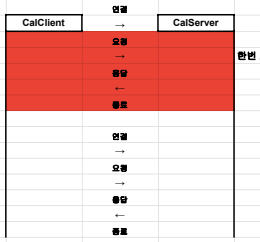
클라이언트 정보를 유지 가능 : stateful
별도의 클라이언트 기능을 실행 가능 : thread

main T : main() { ~~~~~~~~~~~~~~~~~~~~~~~~~> }

start()
T : run(){~~~~~~~~~~~~~~~~}
T : run(){~~~~~~~~~~~~~~~~}
T : run(){~~~~~~~~~~~~~~~~}

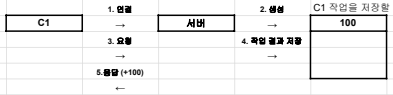
---

*Stateless 1



한번 요청에 한번 응답

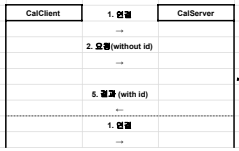
*Stateless 2 : 클라이언트 구분하기



---

*Stateless 2 : 클라이언트 구분하기

Collaborator Diagram = Sequence Diag = 시간 흐름에 따라 실행 과정을 표현기에 적합

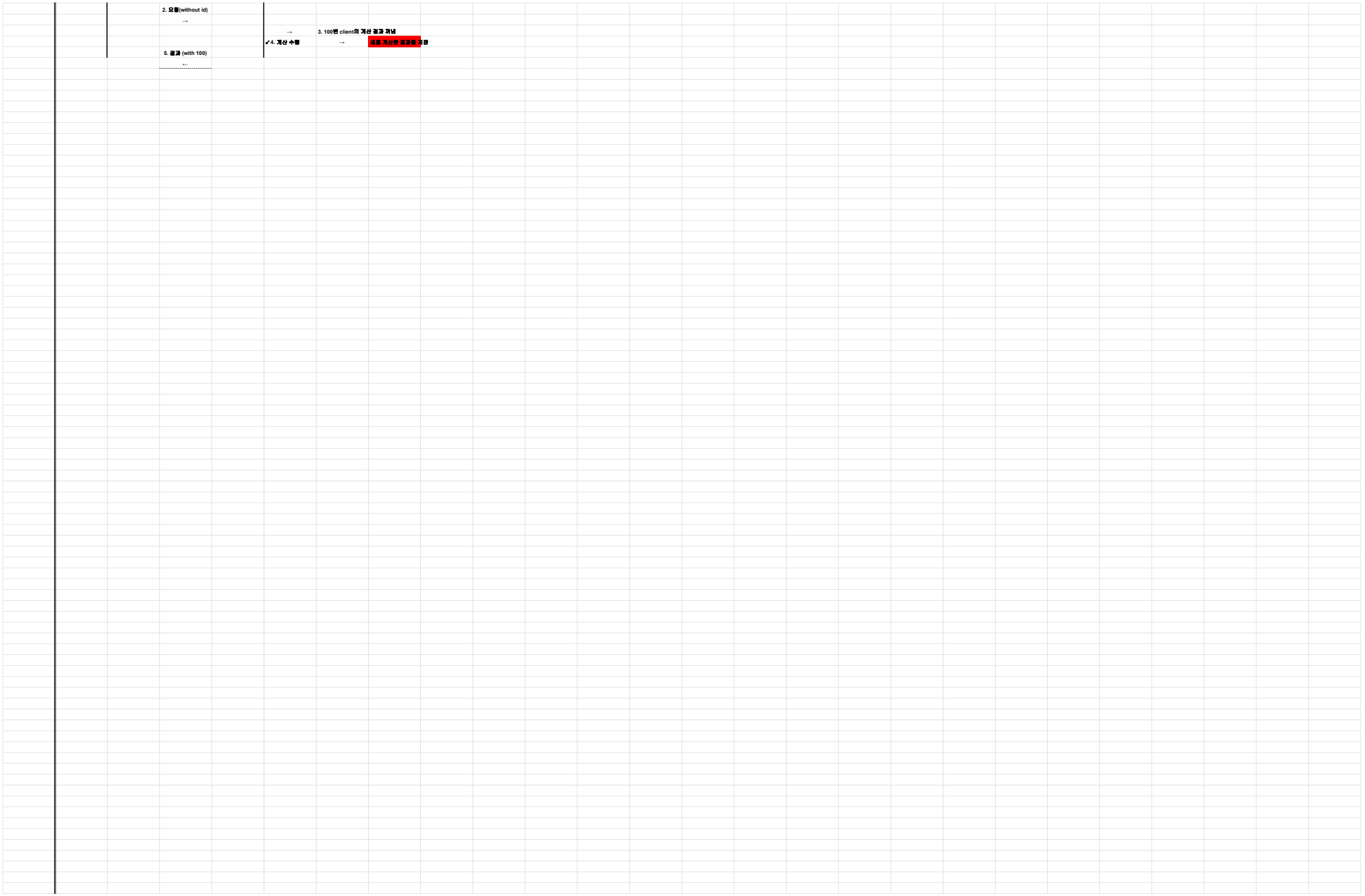


웹프로그래밍에서 세션id(sessoin id)라고 부른다

3. client id 생성 = 예) 100

4. 계산 수행

key	value
client id	계산결과
100	


2. 요청(without id)
→
→
3. 100번 client의 계산 결과 캐냄
↙4. 계산 수행
→
새로 계산한 결과를 저장
5. 결과 (with 100)
←