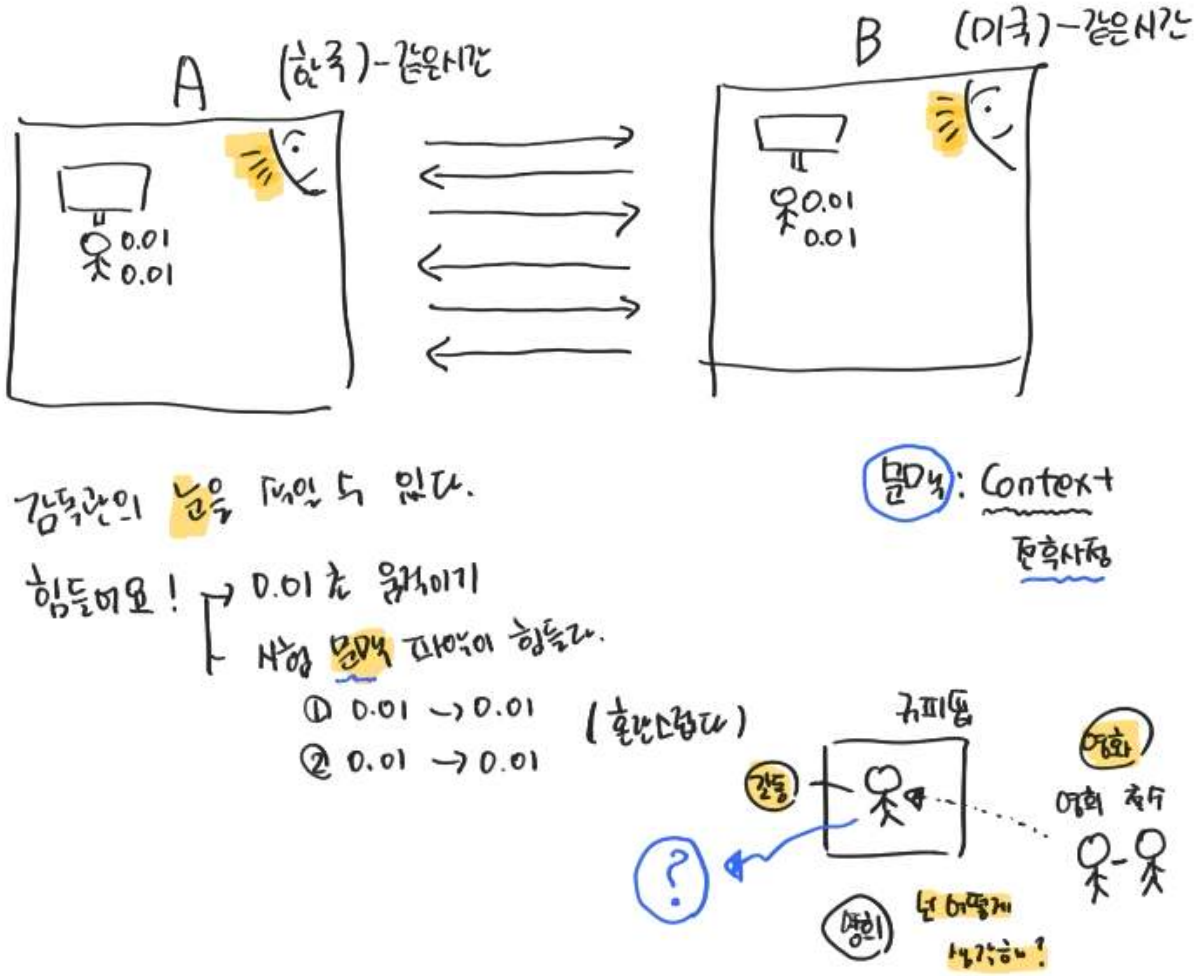
Chapter 7. 스레드와 소켓통신

62장. 스레드(번개맨 아저씨의 고민)

▷ 번개맨이 한국과 미국에서 동시에 시험을 치르고 싶다. 감독관이 눈을 한번 깜빡이는 시간 0.01초. 번개맨은 양쪽 시험장을 0.01초에 한 번씩 왔다 갔다 하면서 동시에 시험을 치르기로 한다.

▷ 멀티스레드 : 하나의 CPU가 두 가지 이상의 일을 동시에 수행하는 것.



▷ 자바는 기본적으로 main Thread 한 개만 갖고 있다.

63장. 스레드 실습

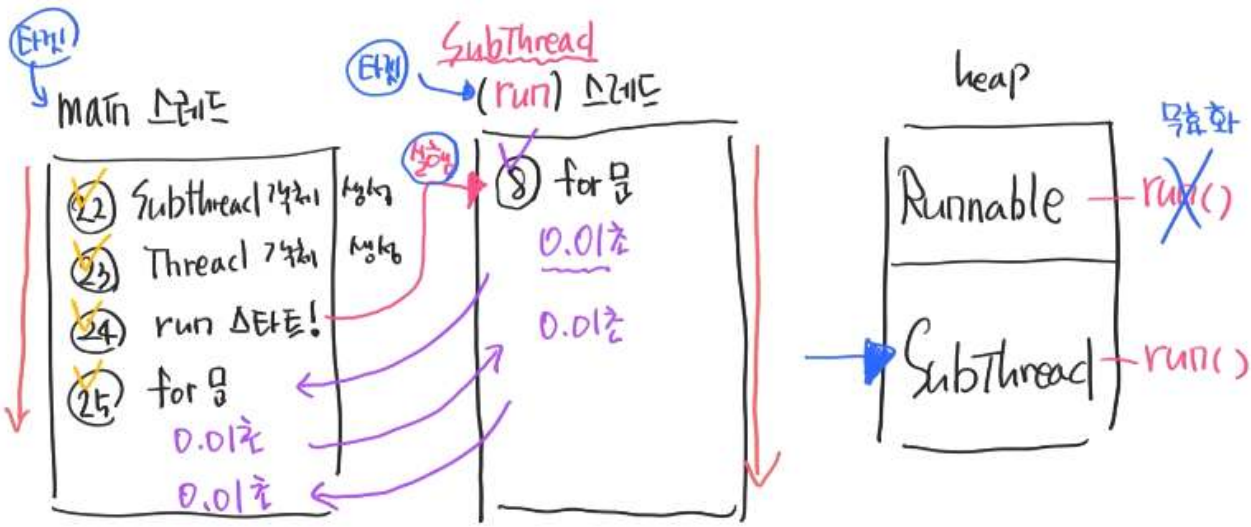
▷ 자바의 메인스레드

```
package ch07;
public class ThreadEx01{
  // 자바의 메인 스레드
  public static void main(String[] args){
    for(int i=1; i<=10; i++){
      try{
        System.out.println("메인스레드 : " + i);
        Thread.sleep(1000);  // 1초 : 1000
      } catch (InterruptedException e){
          e.printStackTrace();
      }
    }
  }
}
```

▷ 서브클래스 추가해서 자바의 서브 스레드로 사용

```
package ch07;

// SubThread → Runnable(다형성)
public SubThread implements Runnable{
  // 자바의 서브스레드
  @Override
  public void run(){
    for(int i=1; i<=5; i++){
      try{
        System.out.println("서브스레드 : " + i);
        Thread.sleep(1000);  // 1초 : 1000
      } catch (InterruptedException e){
          e.printStackTrace();
      }
    }
  }
}
```

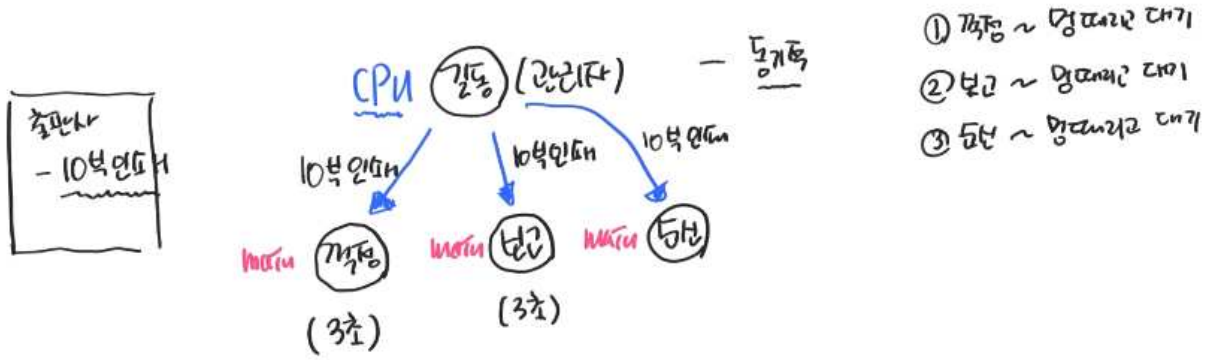
```
public class ThreadEx01{
  // 자바의 메인 스레드
  public static void main(String[] args){
    SubThread st = new SubThread();
    Thread t1 = new Thread(st);  // 타겟을 지정하는 곳

    t1.start;  // run 메서드 실행.

    for(int i=1; i<=5; i++){
      try{
        System.out.println("메인스레드 : " + i);
        Thread.sleep(1000);  // 1초 : 1000
      } catch (InterruptedException e){
          e.printStackTrace();
      }
    }
  }
}
```



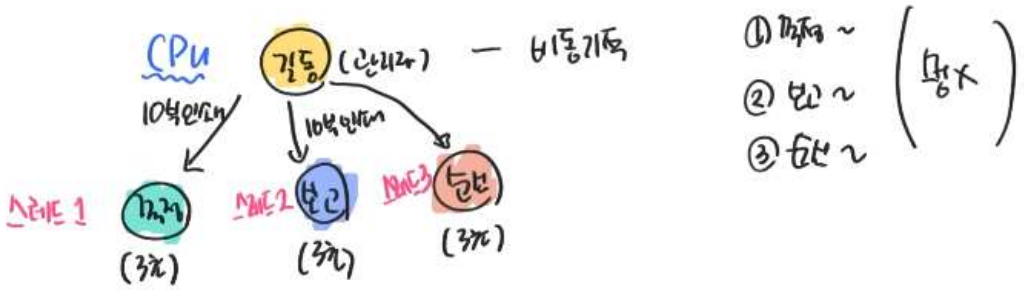
▶ context switching(문맥교환) : 두 개의 스레드가 번갈아 왔다 갔다 하면서 실행 됨. - 특별한 규칙은 없음.

64장. 동기와 비동기

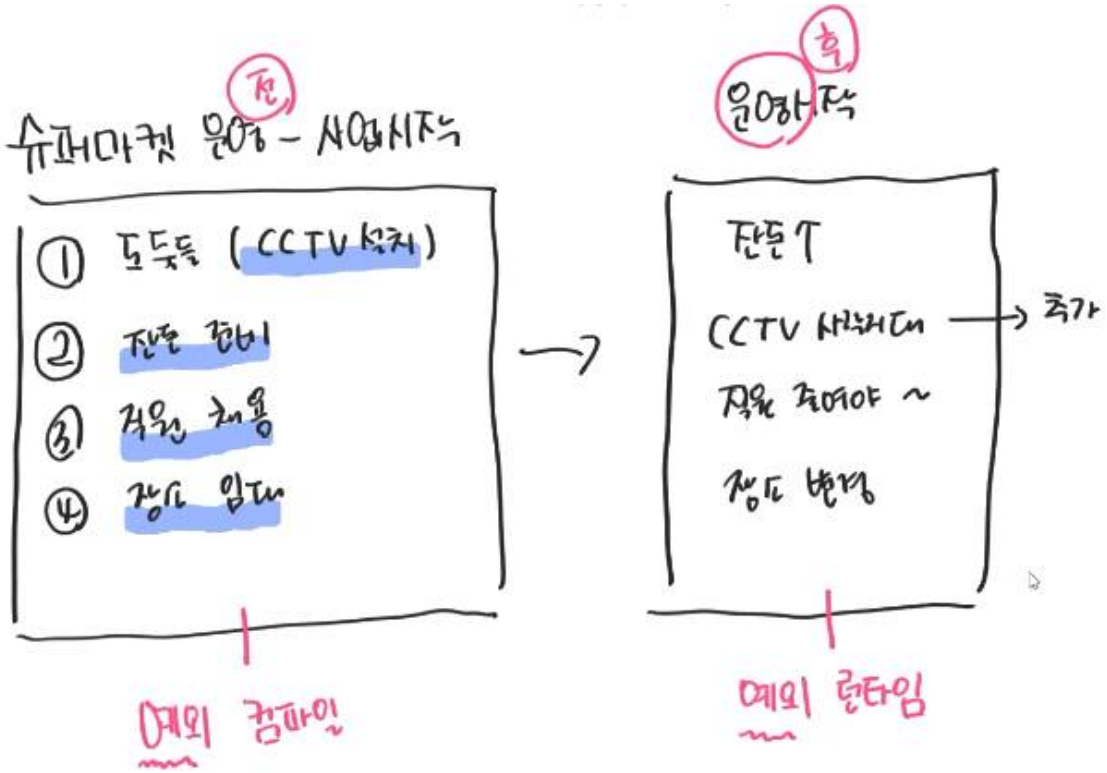
▷ 동기(Sync) : 일의 순서가 있다.



▷ 비동기(Async) : 일의 순서가 없다.



65장. 예외처리(컴파일, 런타임)



▷ 예외 예제

```
package ch07;
class 총{
   void shoot(){
      System.out.println("총을 발사했습니다!!");
   }
}

public class ExceptionEx01{
   public static void main(String[] args){
      // 컴파일 예외(Java가 알 수 있고, 처리도 함.)
      try{
         System.out.println("잠자기 시작");
         Thread.sleep(1000);  // 메인스레드 3초 동안 잠을 잔다.
         System.out.println("잠자기 끝");
      } catch (InterruptedException e){  // 방해!!
            e.printStackTrace();
      }
```

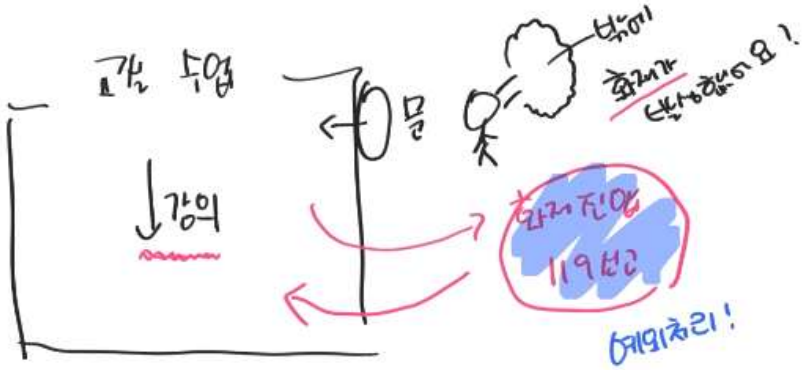
```
    // 런타임 예외(개발자가 알 수 있고, 또 처리도 해야 함.)
    int[] num = {1, 2, 3};
    try{
      System.out.println(num[3]);    // 실행시에 오류 발생
    } catch (ArrayIndexOutOfBoundsException e){  // 방해!!
      // try 했다가 예외가 발생하면 어떻게 처리할지를 정의하는 영역
        System.out.println("괜찮아 그냥 진행해");
        System.out.println(e.getMessage());    // 로그 파일로 남기거나, 데이터베이스에 저장함.
        e.printStackTrace();  // 오류를 추적해서 찍어줌.
    }

    String data = "안녕";
    System.out.println(data.length());  // 정상적으로 길이가 나타남.

    String nullData = null;
    System.out.println(data.length());  // NullPointerException 발생

    총 s = new 총;
    s.shoot();  // 메모리에 올라와서 정상 처리 됨.

    총 s1 = null;
    //s.shoot();  // 메모리에 올라와 지지 않아서 오류 생김. try catch 로 묶어줘야...
    try{
      s.shoot();
    } catch (NullPointerException e) {
        //Systeml.out.println("총이 없어서 발사하지 못했습니다.");
        Systeml.out.println("총이 없어서 총을 만들고 다시 시도하겠습니다.");
        s1 = new 총();
        s.shoot();
    }

    System.out.println("메인 스레드 종료");    // 오류 처리를 반드시 해야한다.
  }
}
```

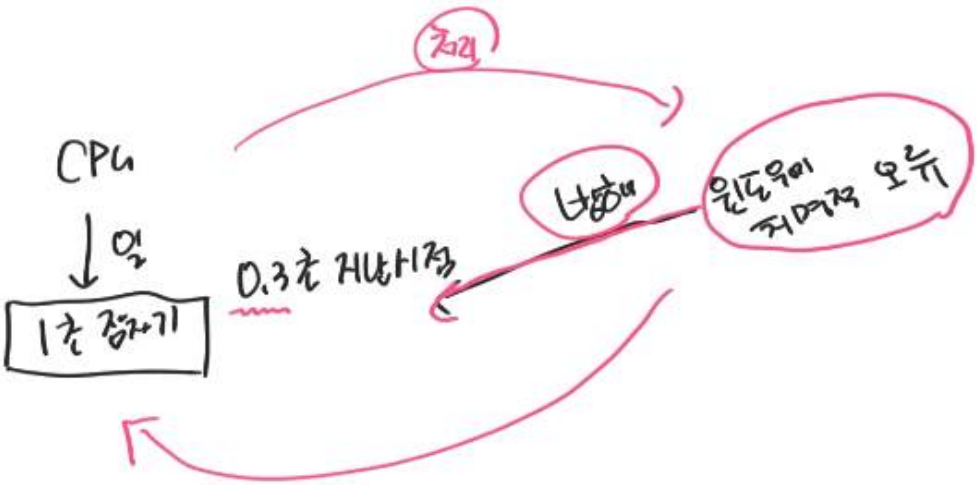
※ 예외의 종류는 굉장히 많으며 세분화 되어 있어서 모든 것을 일일이 처리를 해 주기가 불가능하다. 그래서 모든 예외의 부모클래스를 이용해 예외 처리를 한다.

① Compile Exception → Exception

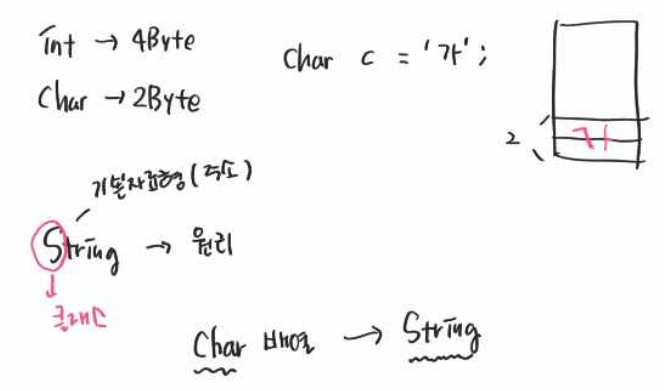
② Runtime Exception → RuntimeException

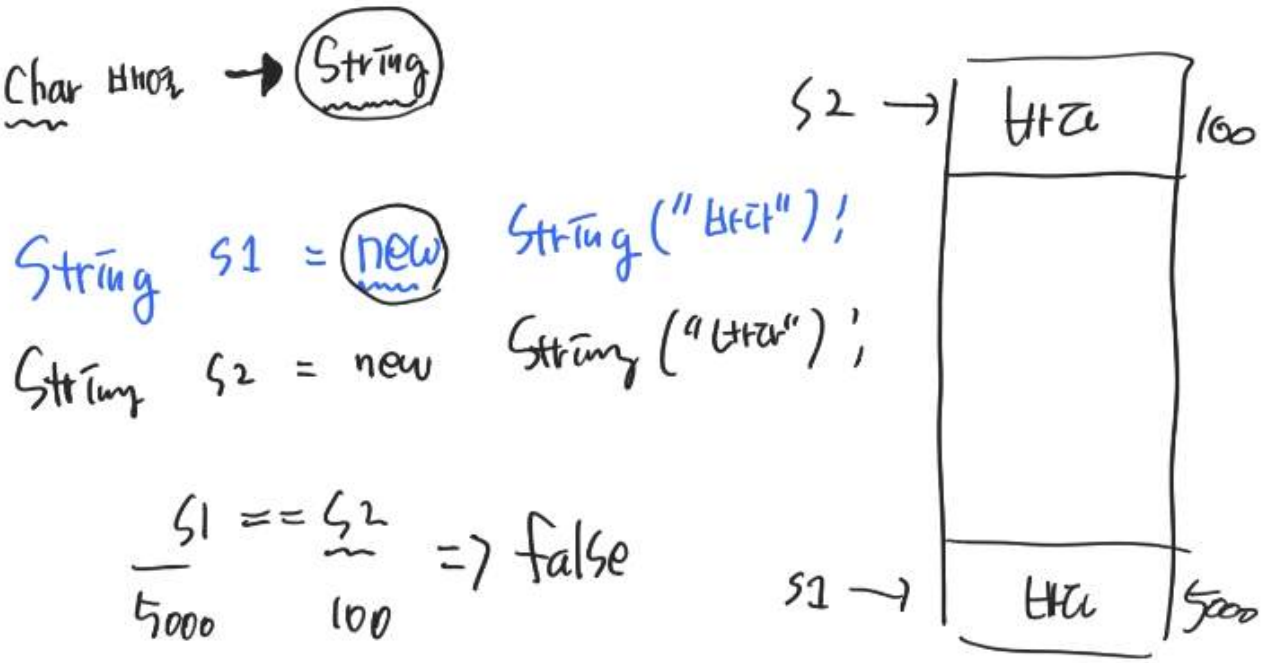
▶ 인터럽트 란?





66장. String Constant Pool

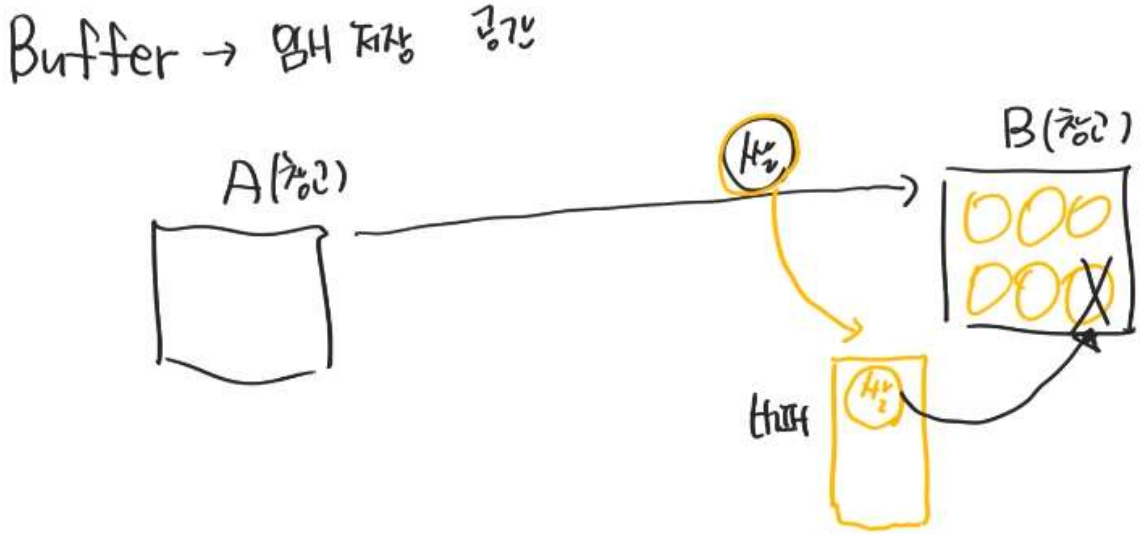




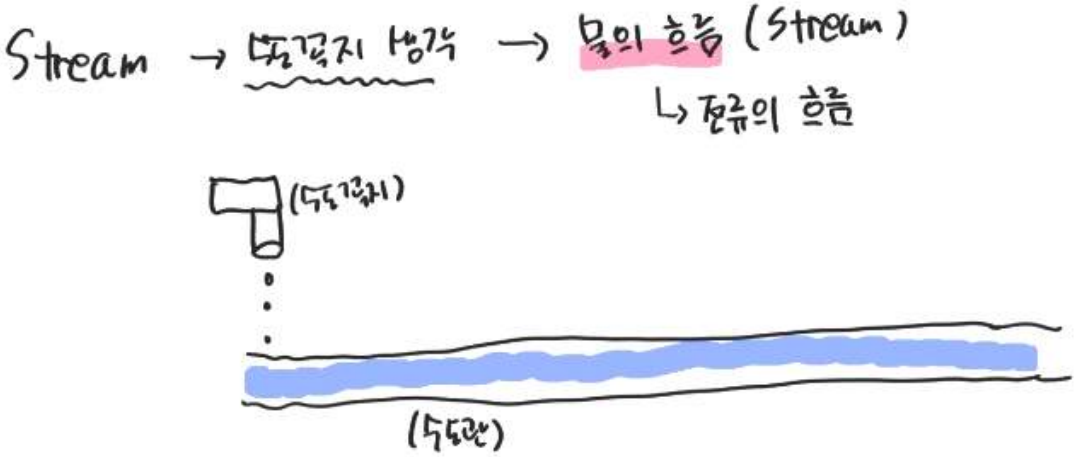
```
package ch07;
public class StringEx01{
  // 자바의 메인 스레드
  public static void main(String[] args){
    String s1 = new String("바다");
    String s2 = new String("바다");
    System.out.pirntln(s1);
    System.out.pirntln(s2);
    System.out.pirntln(s1==s2);
    // 비교해 보면 ... 주소가 다르게 생성되었기 때문에 둘은 같지 않다.

    String s3 = "바다";    //
    String s4 = "바다";
    System.out.pirntln(s3);
    System.out.pirntln(s4);
    System.out.pirntln(s3==s4);
    // 비교해 보면 ... 얘 둘은 같다.
    // 같은 문자열이면 같은 메모리 공간을 공유(Constant Pool)하기에 메모리 효율이 올라감.
    // 자주 변경하면 새로운 공간이 할당 됨.
    // 문자열의 비교는 반드시 equals로 함. → 얘는 주소와 값 두가지를 모두 비교한다.
    System.out.pirntln(s1.equals("바다"));
  }
}
```

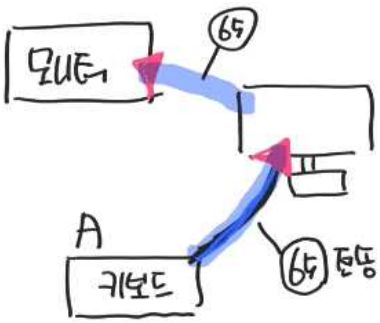
67장. 버퍼(Stream이란)

▷ 버퍼 : 임시공간



▷ Stream 이란?





컴퓨터 입장 (입력)

① 65 데이터 - Input

② 65 데이터 - Output (출력)

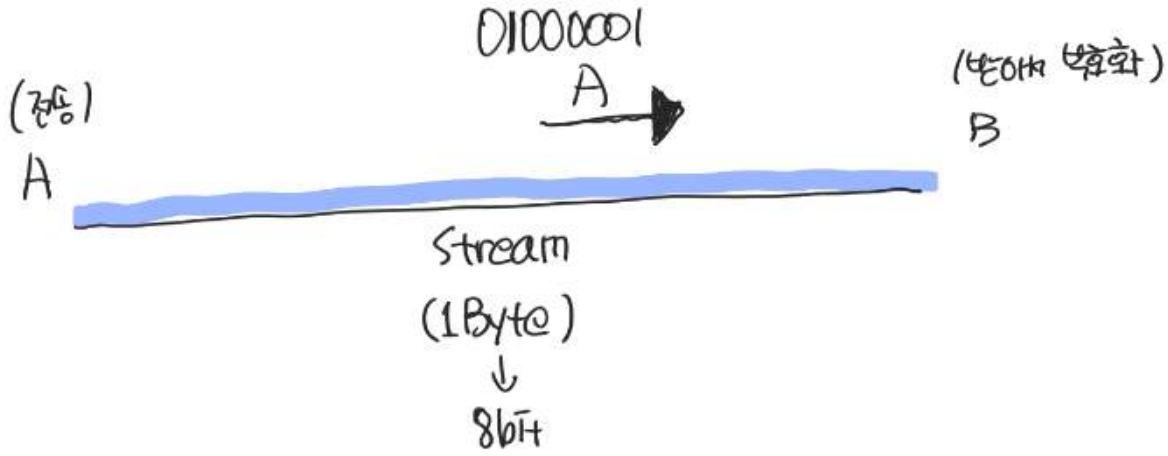
↓

I/O

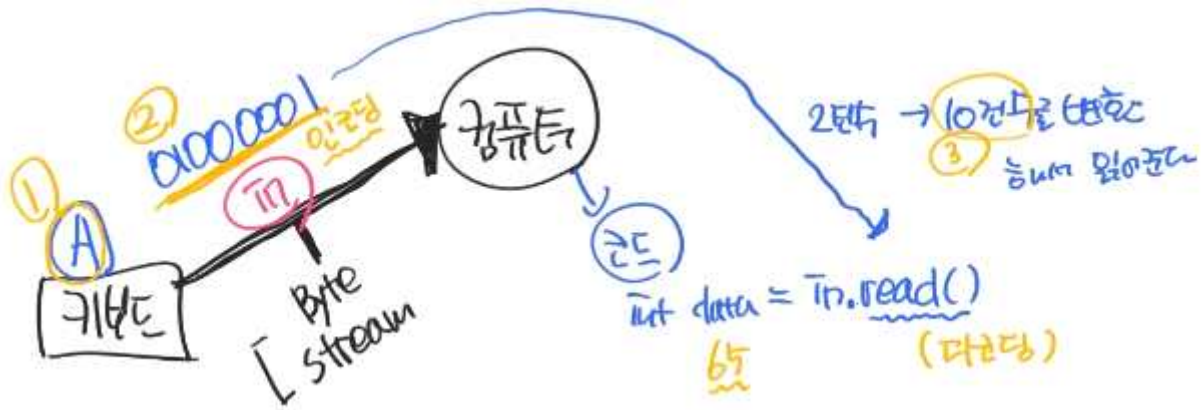
문자	숫자
A	65
B	66
C	67

아스키코드 표 발생!

▷ 통신에서의 전송



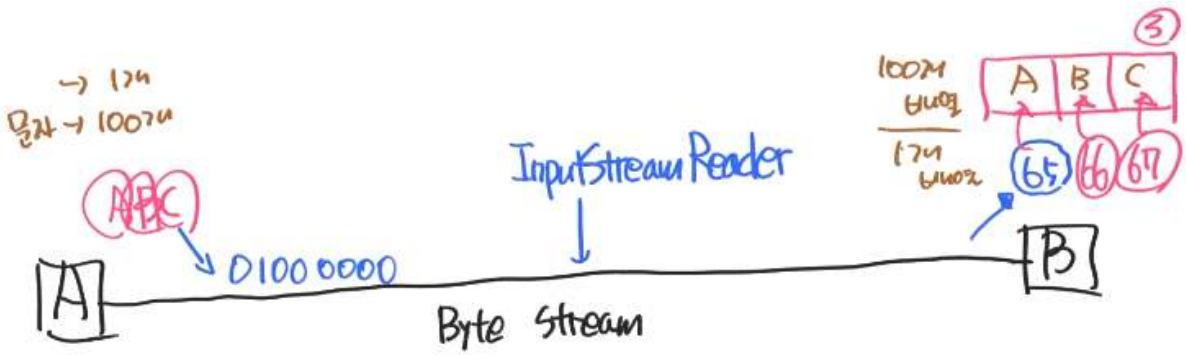
```
package ch07;
import java.io.InputStream;
public class StreamEx01{
   public static void main(String[] args){
      InputStream in = System.in;  // System.In → 키보드(Byte Stream)
      try{
         // 1. 키보드는 A를 인코딩해서 010000 으로 컴퓨터에게 전송
         // 2. ByteStream으로 흘러들어간다.(Input)
         // 3. read() 메서드로 0100000 → 65 로 디코딩한다.
         // 4. 65를 → 문자로 부호화 시킨다.
         int data = in.read();
         //System.out.println(data);
         System.out.println((char)data);  // 부호화
      } catch (Exception e){
          e.printStackTrace();
      }
   // 바이트스트림이기 때문에 한 번에 1바이트만 전송 가능하다.
   }
}
```



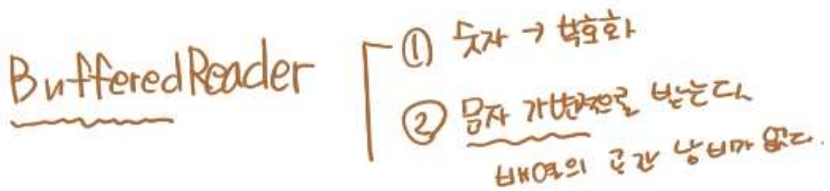
68장. 버퍼(BufferedReader 란)

▷ InputStreamReader

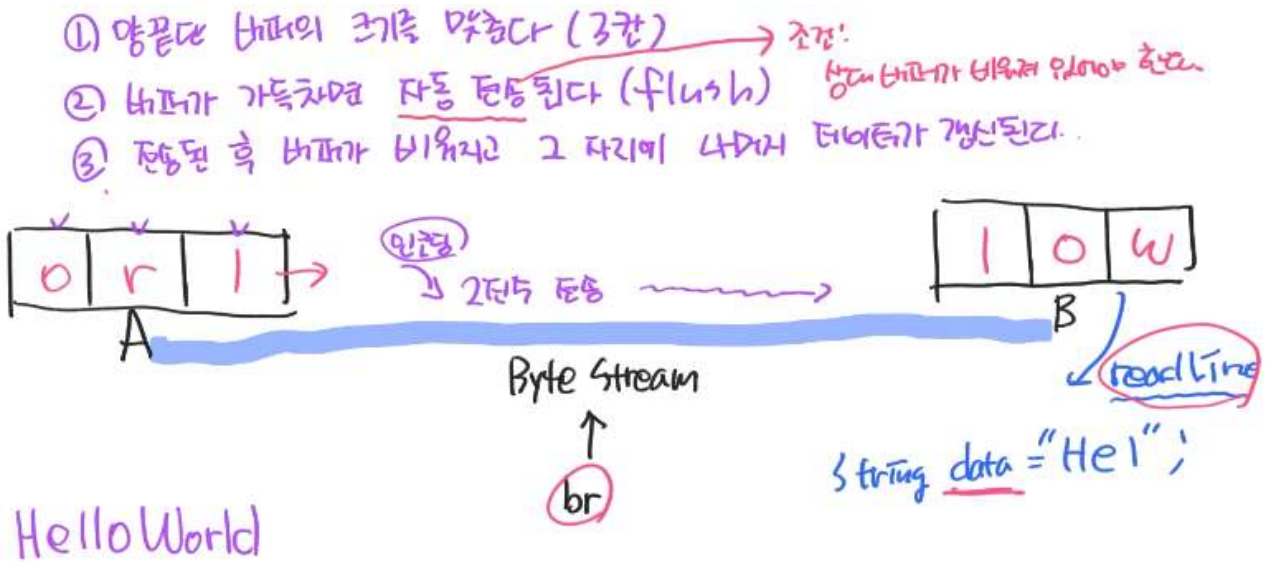
```
package ch07;
public class StreamEx02{
   public static void main(String[] args){
      InputStream in = System.in;  // System.In → 키도드(Byte Stream)
      InputStreamReader ir = new InputStreamReader(in);
      // 65 → A 로 바까주는 애!!
      // 얘는 가변적으로 데이터를 못 받음. → 불편
      // 배열로 처리하면 가능한데.... 몇개가 들어올지 어찌 아는가?
      try{
         // A 만 딸랑 들어오거나 넘치면...?
         // 두가지 해결 한 것이 BufferedReader이고 얘를 사용해야 함.
         char[] data = new char[1000];
         ir.read(data);
         System.out.println(data);  // 부호화
      } catch (Exception e){
          e.printStackTrace();
      }
   // 바이트스트림이기 때문에 한번에 1바이트만 전송 가능하다.
   }
}
```



▷ BufferedReader

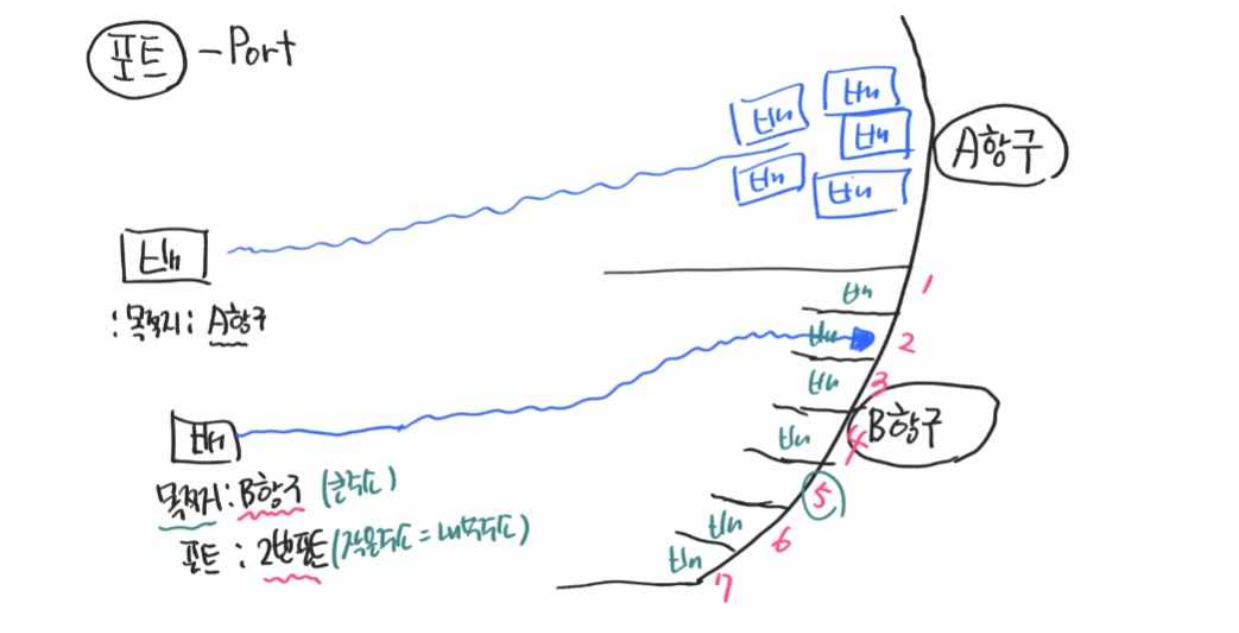


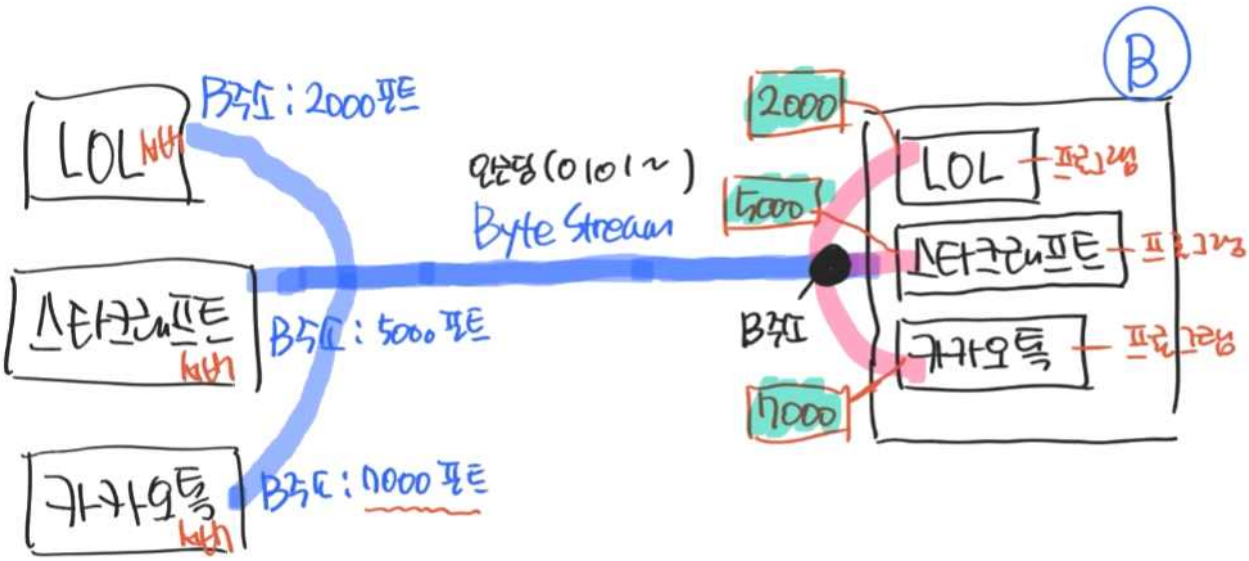
```
package ch07;
public class StreamEx03{
   public static void main(String[] args){
      InputStream in = System.in;  // System.In → 키도드(Byte Stream)
      InputStreamReader ir = new InputStreamReader(in);
      BufferedReader br = new BufferedReader(ir);
      try{
         String data = br.readLine();
         System.out.println(data);
      } catch (Exception e){
          System.out.println(e.getMessage());
      }
   }
}
```



69장. 소켓통신의 개념

▷ 소켓통신





소켓통신 (양끝단 포트는 두고 통신)

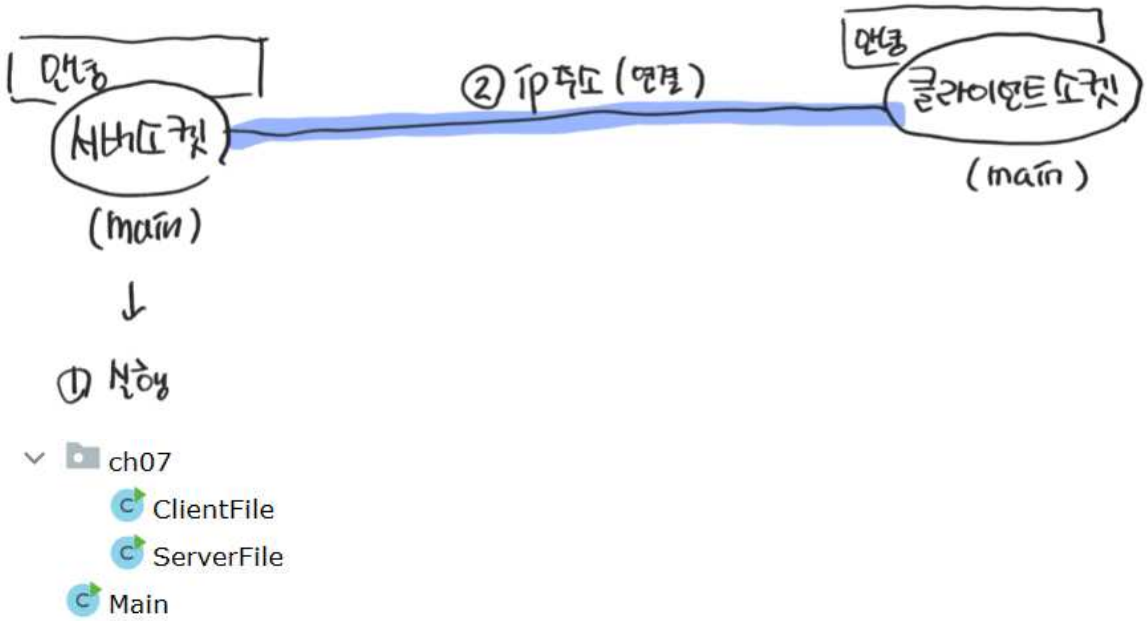
0 ~ 65535번! → 65536개 → 2^{16} → 2Byte → 16bit

0 ~ 1023 포트 X

1024 ~ 이상

70장. 소켓통신(서버소켓 만들기) - 일대일

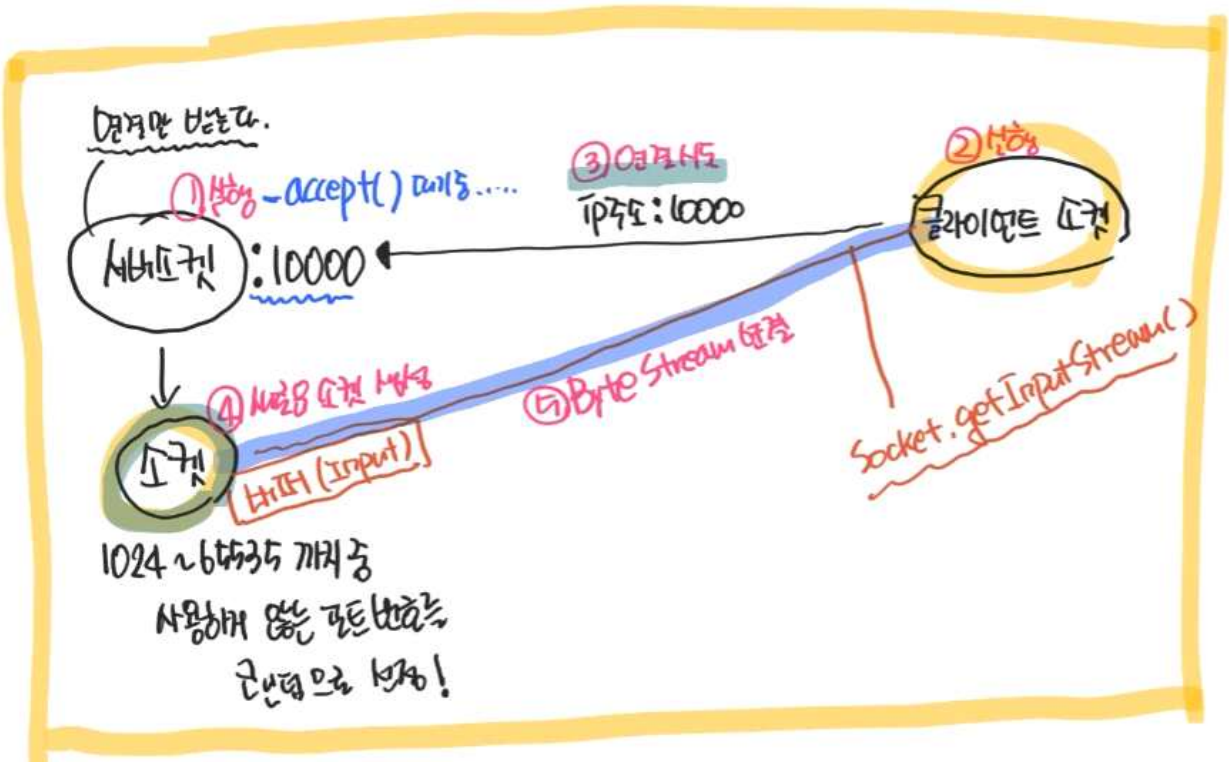
▷ 실습개요



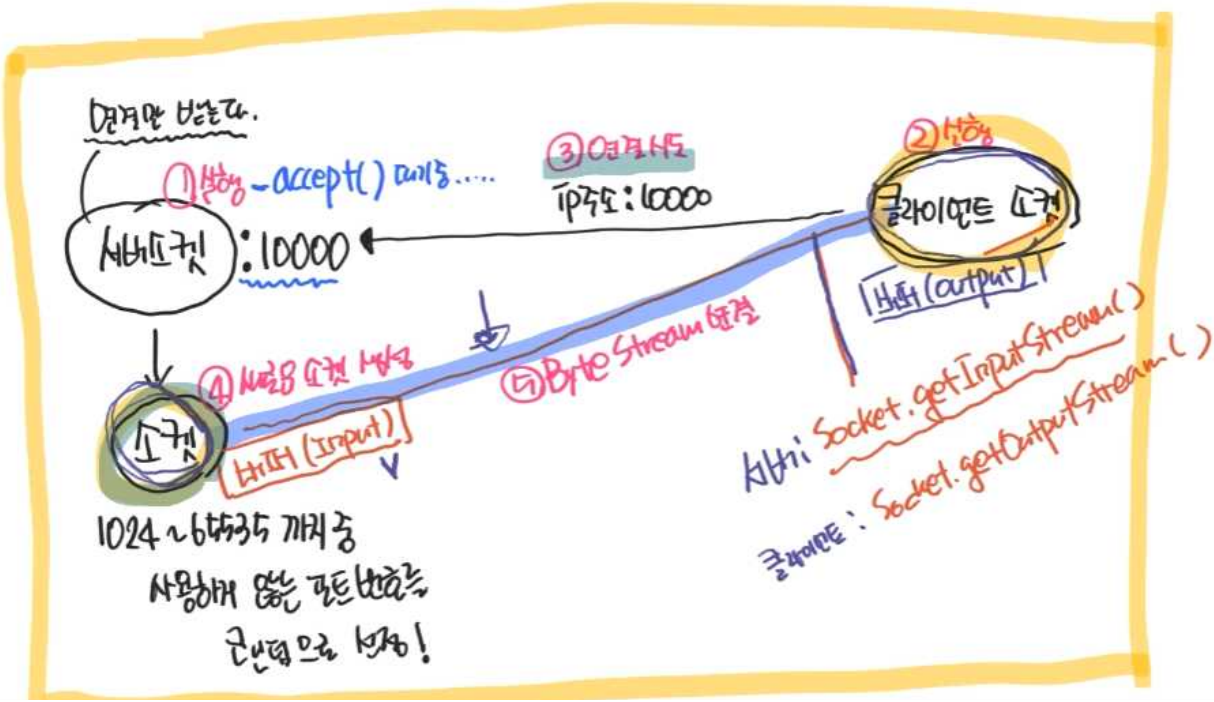
▶ ServerFile.java

```
public class ServerFile {
    // 클라이언트 연결을 받는 소켓
    ServerSocket serverSocket;
    //실제 통신을 하는 소켓
    Socket socket;
    BufferedReader br;
    public ServerFile(){
        System.out.println("1. 서버소켓 시작-------------------------------------");
        try{
            serverSocket = new ServerSocket(10000);
            System.out.println("2. 서버소켓 생성완료----------------------");
            System.out.println("3. 클라이언트 연결 대기 중----------------------");
            socket =  serverSocket.accept();
            System.out.println("4. 클라이언트 연결 완료 -----------------------------");
            // 클라이언트가 보내는 메시지를 읽어 들이는 버퍼 생성
            br = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String msg = br.readLine();
            System.out.println("4. 클라이언트로부터 받은 메시지 : " + msg);
        } catch (Exception e){
            System.out.println("서버소켓 에러 발생함 : " + e.getMessage());
        }
```

```
    }
    public static void main(String[] args) {
        new ServerFile();
    }
}
```



71장. 소켓통신(클라이언트소켓 만들기)

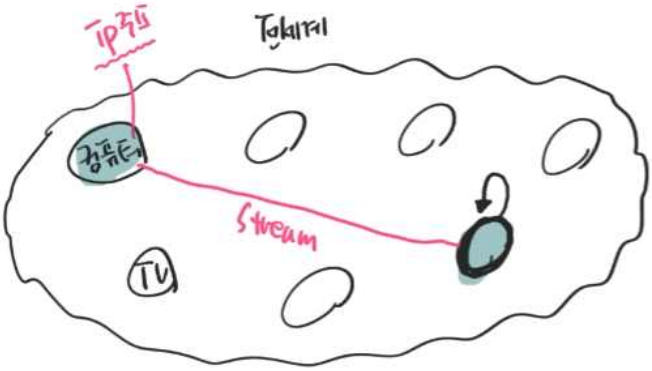


▷ ClientFile.java

```
public ClientFile(){
        // 소켓연결하기
        try{
            System.out.println("1. 클라이언트 소켓 시작-----------------------------------");
            socket = new Socket("127.0.0.1", 10000);
            // 서버 소켓의 accept() 메서드가 호출 됨.
            System.out.println("2. 버퍼(write) 연결완료---------------------------------");
            bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            // 서버로 메시지를 보내기 위한 키보드 연결
            System.out.println("3. 키보드 스트림 + 버퍼(read) 연결완료-------------------");
            br = new BufferedReader(new InputStreamReader(System.in));

            System.out.println("4. 키보드 메시지 입력 대기 중-------------------------------");
            String keyboardMsg = br.readLine();;
            // 반드시 메시지 끝을 알려줘야 한다.  ₩n
            bw.write(keyboardMsg + "₩n");
            // 버퍼가 크기 때문에 차려면 오래 걸린다. 자주 비워준다.
            bw.flush();
        } catch (Exception e) {
            System.out.println("클라이언트 소켓 에러 발생 함 : " + e.getMessage());
```

```
        }
    }
    public static void main(String[] args) {
        new ClientFile();
    }
}
```

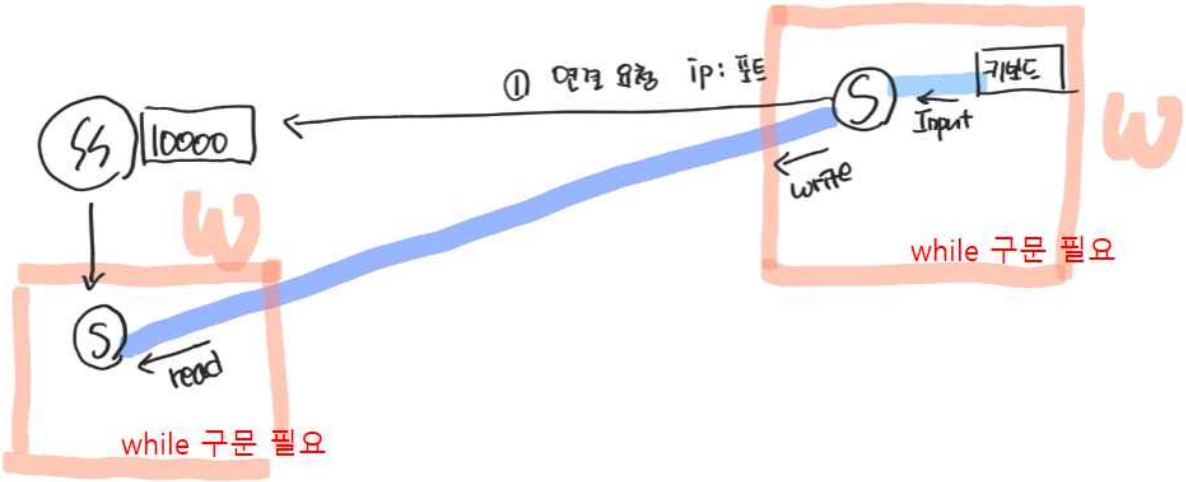
IPv4 체제 : 000.000.000.000
↓ ↓ ↓ ↓
0~255 " " "
↓ ↓ ↓ ↓
2^8 2^8 2^8 2^8
$= 2^{32} =$ 42억9천



ex) 210.10.8.254
접속 : 유일하다

자기 자신의 ip
: 110.5.8.7
: 본인 컴퓨터를 나타내는 주소 : 127.0.0.1] 같음!
: localhost

72장. 소켓통신(메시지 지속적으로 받기)



▷ ServerFile.java

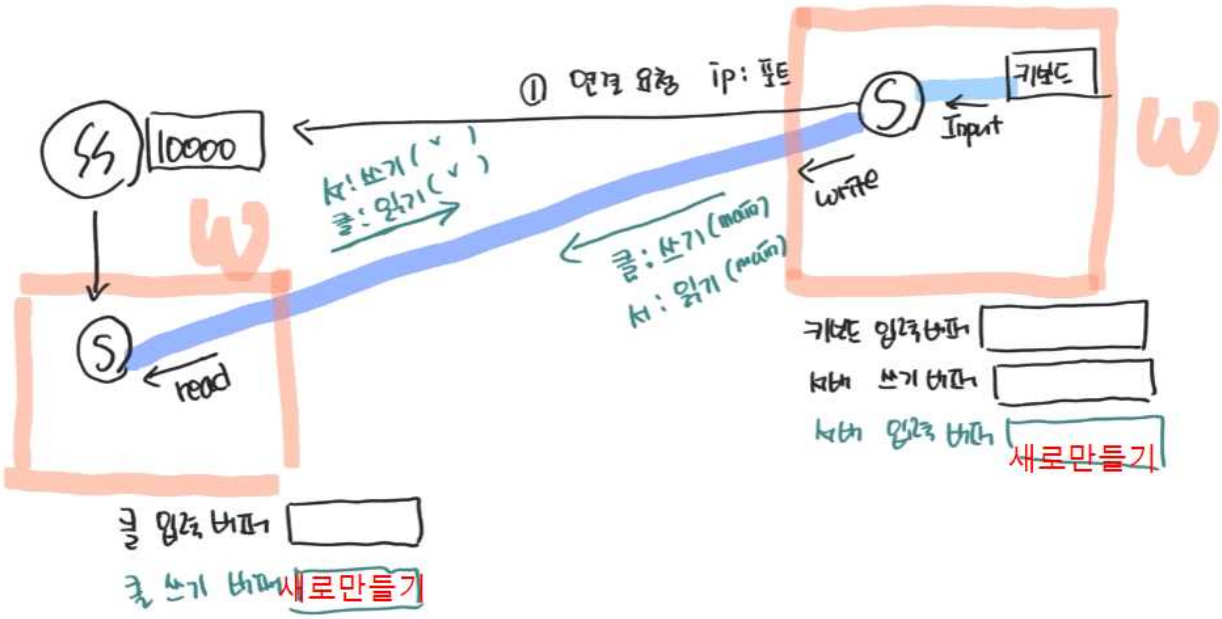
```
package org.example.socket2;
public class ServerFile {
    // 클라이언트 연결을 받는 소켓
    ServerSocket serverSocket;
    //실제 통신을 하는 소켓
    Socket socket;
    BufferedReader br;
    public ServerFile(){
        System.out.println("1. 서버소켓 시작-----------------------------------");
        try{
            serverSocket = new ServerSocket(10000);
            System.out.println("2. 서버소켓 생성완료 : 클라이언트 접속 대기 중----------------------");
            socket =  serverSocket.accept();  // 클라이언트 접속 대기중....
            System.out.println("3. 클라이언트 연결 완료 ----------------------------");
            //외부로 부터 읽어들이는 버퍼를 달았다.
            br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            while(true){
                String msg = br.readLine();
                System.out.println("4. 클라이언트로부터 받은 메시지 : " + msg);
            }
        } catch (Exception e){
            System.out.println("서버소켓 에러 발생함 : " + e.getMessage());
        }

    }
```

```
    public static void main(String[] args) {
        new ServerFile();
    }
}
```

▷ ClientFile.java

```
package org.example.socket2;
public class ClientFile {
    Socket socket;
    BufferedWriter bw;
    BufferedReader br;

    public ClientFile(){
        // 소켓연결하기
        try{
            System.out.println("1. 클라이언트 소켓 시작-----------------------------------");
            socket = new Socket("127.0.0.1", 10000); // 서버 소켓의 accept() 메서드가 호출 됨.
            System.out.println("2. 버퍼(write) 연결완료---------------------------------");
            bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            // 키보드 연결
            System.out.println("3. 키보드 스트림 + 버퍼(read) 연결완료--------------------");
            br = new BufferedReader(new InputStreamReader(System.in));
            while(true){
                System.out.println("4. 키보드 메시지 입력 대기 중--------------------------------");
                String keyboardMsg = br.readLine();;
                // 메시지 끝을 알려줘야 한다.  ₩n
                bw.write(keyboardMsg + "₩n");
                bw.flush();
            }
        } catch (Exception e) {
            System.out.println("클라이언트 소켓 에러 발생 함 : " + e.getMessage());
        }
    }
    public static void main(String[] args) {
        new ClientFile();
    }
}
```

73장. 소켓통신(양방향통신)-Thread



▷ ServerFile.java

```
package org.example.ch07.socket3;

public class ServerFile {
    // 클라이언트 연결을 받는 소켓
    ServerSocket serverSocket;
    //실제 통신을 하는 소켓
    Socket socket;
    BufferedReader br;
    // 새로운 스레드가 필요하다.
    BufferedWriter bw;
    BufferedReader keyboard;
    public ServerFile(){
        System.out.println("1. 서버소켓 시작-----------------------------------");
        try{
            serverSocket = new ServerSocket(10000);
            System.out.println("2. 서버소켓 생성완료 :
                                클라이언트 접속 대기 중-------------------------------");
            socket =  serverSocket.accept();  // 클라이언트 접속 대기중....
            System.out.println("3. 클라이언트 연결 완료 -------------------------------");
            //외부로 부터 읽어들이는 버퍼를 달았다.
            br = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            keyboard = new BufferedReader(new InputStreamReader(System.in));
            bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
```

```
            // write 스레드 실행(글쓰기)
            WriteThread wt = new WriteThread();
            Thread t1 = new Thread(wt);
            t1.start();

              // main 스레드 역할(글읽기)
            while(true){
                String msg = br.readLine();
                System.out.println("4. 클라이언트로부터 받은 메시지 : " + msg);
            }
        } catch (Exception e){
            System.out.println("서버소켓 에러 발생함 : " + e.getMessage());
        }
    }
    // 내부 클래스(여기에서만 쓸거니까)
    class WriteThread implements Runnable{
        @Override
        public void run() {
            while (true){
                try {
                    String keyboardMsg = keyboard.readLine();
                    bw.write(keyboardMsg + "₩n");
                    bw.flush();
                } catch (Exception e){
                    System.out.println("서버소켓 쪽에서 키보드 입력받는 중
                                        오류가 발생했습니다." + e.getMessage());
                }
            }
        }
    }
    public static void main(String[] args) {
        new ServerFile();
    }
}
```

▷ ClientFile.java

```
package org.example.ch07.socket3;

public class ClientFile {
    Socket socket;
    BufferedWriter bw;
    BufferedReader keyboard;
    BufferedReader br;

    public ClientFile(){
        // 소켓연결하기
        try{
            System.out.println("1. 클라이언트 소켓 시작-----------------------------------");
            socket = new Socket("127.0.0.1", 10000); // 서버 소켓의 accept() 메서드가 호출 됨.
            System.out.println("2. 버퍼(write) 연결완료---------------------------------");
            bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            // 키보드 연결
            System.out.println("3. 키보드 스트림 + 버퍼(read) 연결완료--------------------");
            keyboard = new BufferedReader(new InputStreamReader(System.in));

            br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            // 새로운 스레드의 역할(글 읽기)
            ReadThread rt = new ReadThread();
            Thread t1 = new Thread(rt);
            t1.start();

            // 메인 스레드의 역할(글 쓰기)
            while(true){
                System.out.println("4. 키보드 메시지 입력 대기 중-------------------------------");
                String keyboardMsg = keyboard.readLine();;
                // 메시지 끝을 알려줘야 한다.  ₩n
                bw.write(keyboardMsg + "₩n");
                bw.flush();
            }
        } catch (Exception e) {
            System.out.println("클라이언트 소켓 에러 발생 함 : " + e.getMessage());
        }
    }
```

```
    class ReadThread implements Runnable{

        @Override
        public void run() {
            while (true) {
                try {
                    String msg = br.readLine();
                    System.out.println("서버로 부터 받은 메시지 : " + msg);
                } catch (Exception e) {
                    System.out.println("클라이언트 소켓 쪽에서 메시지를  입력받는 중
                            오류가 발생했습니다." + e.getMessage());
                }
            }
        }
    }
    public static void main(String[] args) {
        new ClientFile();
    }
}
```