[디지털 컨버전스]
스마트 콘텐츠와 웹 융합 응용SW 개발자 양성과정

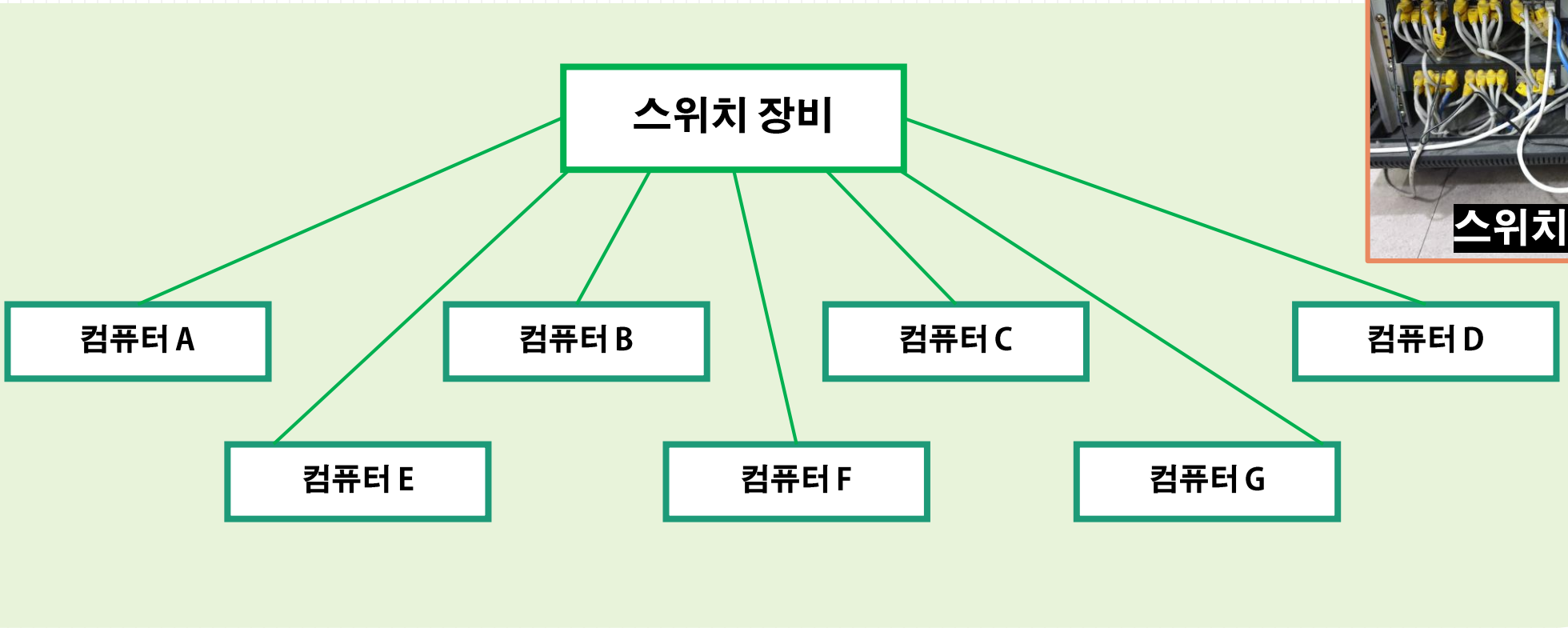
강사 : 이상훈
학생 : 임초롱

22회차

네트워크

링크 https://github.com/limcholong/LectureContents/tree/main/java/CholongLim/Day22/src

집에서는 공유기가 스위치 장비 역할을 하고 있다.

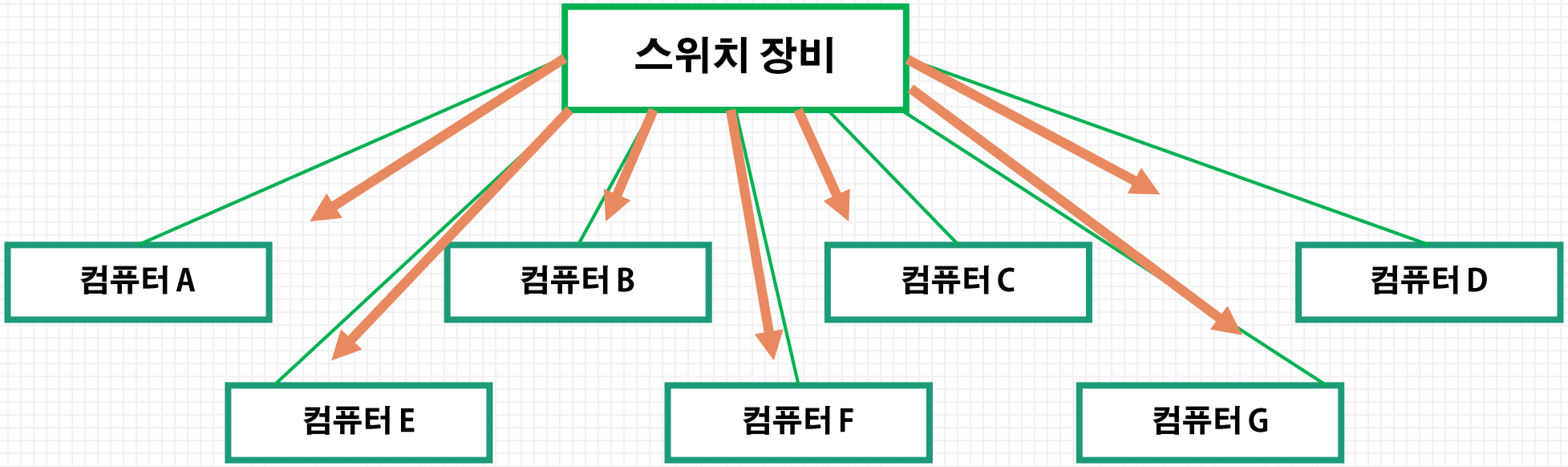


스위치 장비와 pc들은 위와 같은 형태로 연결되어 있다.

네트워크

링크 https://github.com/limcholong/LectureContents/tree/main/java/CholongLim/Day22/src

- ex. 컴퓨터 G가 컴퓨터 E를 찾는다.



➔ 전 구간으로 데이터를 뿌리는 것을 **BroadCasting(브로드 캐스팅)** 이라고 한다.
(그러나 계속 BroadCasting하면 스위치 장비가 과부하 된다. = DoS 공격 방식 중 하나)

네트워크

링크 https://github.com/limcholong/LectureContents/tree/main/java/CholongLim/Day22/src

IP의 종류에는 두 가지가 있다.
1. 공인 IP
2. 사설 IP

사설 IP192.168대역은 사설망이다. (NAT 프로토콜)
= IP확인을 위해서는 Git에서 ipconfig -all을 입력하여 IPv4 Address를 확인한다.

인터넷을 하기 위해서는 무조건 공인망이 있어야 한다.
공인 IP가 있어야 외부 인터넷 가능하다.
그걸 NAT 프로토콜이 하며 그걸 지원해주는 것이 공유기이다.

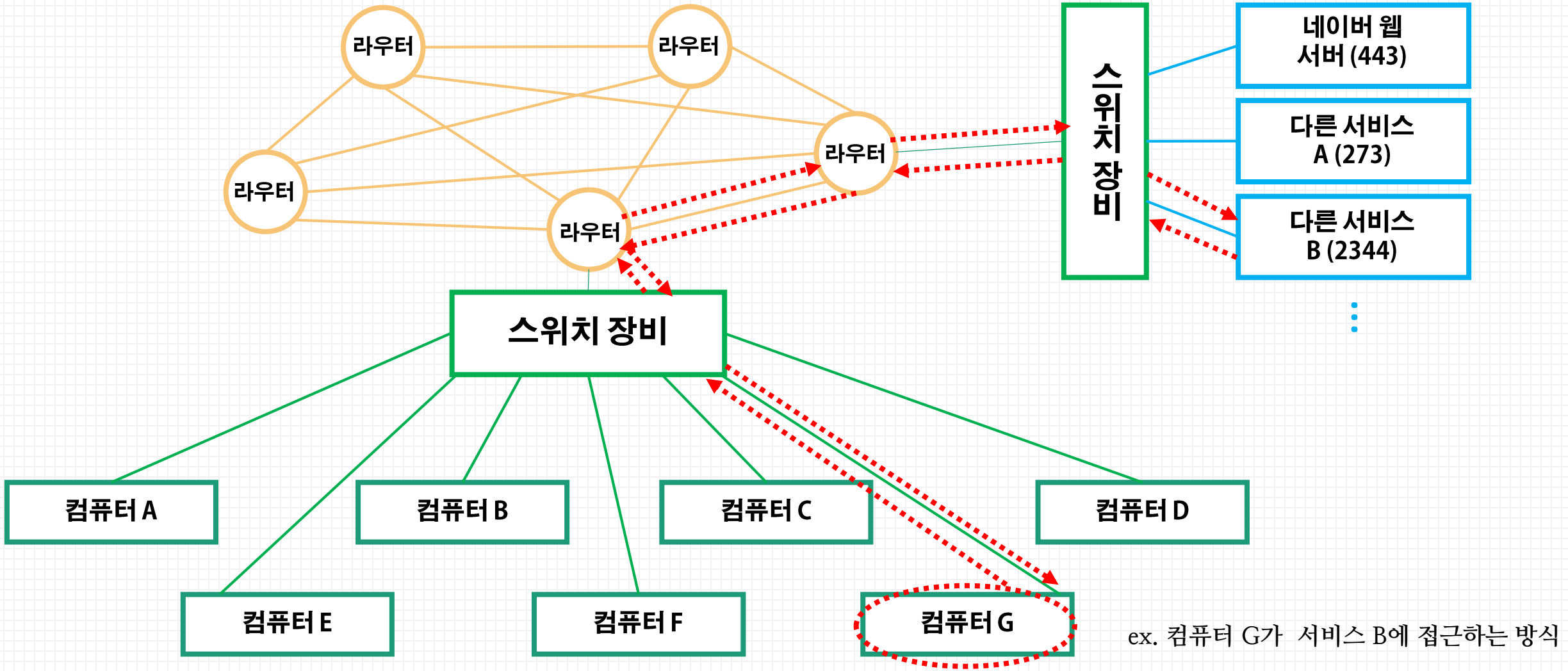
공유기의 역할이 하나의 공인 IP 를 가지고 여러 대의 사설 IP 가 인터넷을 할 수 있게 해준다.
(NAT 프로토콜이 이것을 서포트 해준다.)
즉, 사설 IP 를 사용하는 가정에서는 공유기가 스위치 역할을 해준다.

그렇다면 어떻게 구글 / 네이버 등에 접속할 수 있을까 ?

네트워크

링크 https://github.com/limcholong/LectureContents/tree/main/java/CholongLim/Day22/src

Router(라우터)의 이용



ex. 컴퓨터 G가 서비스 B에 접근하는 방식

네트워크

링크 https://github.com/limcholong/LectureContents/tree/main/java/CholongLim/Day22/src

Router(라우터)가 하는 일은 무엇일까?
요청하는 IP주소는 Router가 보고 Router가 어디로 갈지 판단한다.
(네이버의 IP주소를 보고 연결 판단을 해준다.)

스위치 장비가 하는 일은 무엇일까?
하드웨어 주소를 보고 맥주소를 본다.
(Physical Address)

〈 결론 〉

웹 서비스에서 가장 중요한 것은 두 가지이다.
1. 포트번호(port)
2. 공인 IP

61번 문제 복습

링크 https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day22/src/Quiz61.java

```java
import java.math.BigInteger;

// 숫자가 굉장히 커지므로 BigInteger를 사용하도록 한다.
// 1 ~ 1000억까지 짝수들의 합을 계산하는 스레드 1개
// 1 ~ 1000억까지 7의 배수들의 합을 계산하는 스레드 1개
// 1 ~ 1000까지 11의 배수를 곱하는 스레드 1개
// 총 3개의 스레드를 만들어서 이들 각각의 결과를 출력하고
// 이 결과의 합을 출력하도록 프로그래밍 해보자!

class ThreadManager {
    final static int MAXTHREAD = 3;
    // 짝수 합, 7의 배수 합, 11 배수 곱을 계산할 각각의 스레드
    final static BigInteger START = new BigInteger( val: "1");
    // 시작 값 : 1 (이유 : 1부터 1000억까지의 수 중에서 계산값을 구함)
    final static BigInteger END = new BigInteger( val: "100000000000");
    // 끝 값 : 100000000000 (이유 : 1부터 1000억까지의 수 중에서 계산값을 구함)

    final static int EVEN = 2;
    // 짝수
    final static int SEVEN = 7;
    // 7의 배수
    final static int ELEVEN = 11;
    // 11의 배수

    final static int[] OPTION_ARR = { EVEN, SEVEN, ELEVEN };
    // 옵션 배열 = { 2 , 7 , 11 }

    Thread[] thr;

    public ThreadManager () {
        thr = new Thread[MAXTHREAD];
        // 스레드는 총 3개

        for (int i = 0; i < MAXTHREAD; i++) {
            thr[i] = new Thread(new DistributedThread(START, END, i, OPTION_ARR[i]));
            // thr[0] = new Thread ( new DistributedThread(1, 30.add(1), 0, EVEN ))
            // thr[1] = new Thread ( new DistributedThread(1, 30.add(1), 1, SEVEN ))
            // thr[2] = new Thread ( new DistributedThread(1, 30.add(1), 2, ELEVEN ))
        }
    }

    public void calcEachBigInteger () {
        calcEachBigIntegerStart();
    }

    public void calcEachBigIntegerStart () {
        for (int i = 0; i < MAXTHREAD; i++) {
            thr[i].start();
            // 스레드 3개 시작
        }
    }

    public void calcEachBigIntegerJoin () throws InterruptedException {
        for (int i = 0; i < MAXTHREAD; i++) {
            thr[i].join();
            // join을 통해 먼저 끝나는 스레드가 있다면 대기
        }
    }
}
```

61번 문제 복습

링크 https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day22/src/Quiz61.java

```java
class DistributedThread implements Runnable {
    // Runnable 인터페이스
    // run( ) 매서드만 채워주면 Runnable 인터페이스 구현은 간단하다.
    // Thread 상속받은 클래스처럼 start( )매서드는 없다.
    // 별도로 Thread 생성,구현한 Runnable 인터페이스를 인자로 넘겨주어야 한다.

    BigInteger start;
    BigInteger end;

    int threadIdx;
    int option;

    static final BigInteger ONE = new BigInteger( val: "1");

    BigInteger localSum;
    static BigInteger totalSum;

    public DistributedThread (BigInteger start, BigInteger end, int threadIdx, int option) {
        this.start = start;
        //this.end = end.add(BigInteger.ONE);
        this.end = new BigInteger( val: "30").add(BigInteger.ONE);
        // end = 100000000000 값이 너무 커서 계산 확인이 안된다.
        // end = 30으로 재설정
        // 30.add(1)

        this.threadIdx = threadIdx;
        this.option = option;

        localSum = BigInteger.ZERO;
        totalSum = BigInteger.ZERO;
    }
```

```java
    private synchronized void addAll () {
        totalSum = totalSum.add(localSum);
//        totalSum += localSum; 이 안된다.
        // totalSum = totalSum + localSum
    }

    @Override
    public void run() {

        for (BigInteger i = start; i.compareTo(end) == -1; i = i.add(ONE)) {
//            compareTo 란 무엇인가 ?
//            두개의 값을 비교하여 int값으로 반환해주는 함수이다.
            // 문자열 비교와 숫자의 비교 두 방식이 존재하며,
            // 숫자의 비교 : 크다(1), 같다(0), 작다(-1)
            // 문자열 비교 : 같다(0), 그 외 양수 / 음수값

            // 기준값.compareTo(비교대상)
            // 즉 i.compareTo(end) == -1 이란
            // i 값이 end값과 비교하였을때 작을 때
            // i < end와 같은 의미이지만, BigInteger에서는 쓸 수 없다.

            // i = i.add(ONE) 는 i++와 의미가 같다.



            /* DEBUG 메시지
            System.out.println("threadIdx: " + threadIdx +
                    ", BigInteger 기반 비교를 시작합니다. 현재 i = " + i +
                    ", start = " + start + ", end = " + end +
                    ", i + ONE = " + i.add(ONE)); */

            // if (i % option == 0) {
```

61번 문제 복습

링크 https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day22/src/Quiz61.java

```
125             if (
126                     (i.mod(new BigInteger(String.valueOf(option))).
127                             compareTo(BigInteger.ZERO)
128                     ) == 0
129             ) {
130 //              mod는 무엇인가 ?
131                 // 나머지 연산자(Modulus Operator)
132                 //i.mod(new BigInteger(String.valueOf(option))).compareTo(BigInteger.ZERO)) == 0
133                 // 만약 (i % ( EVEN.compareTo(0) )) == 0
134                 // (i % ( 2.compareTo(0) )) == 0
135                 // 계산식을 어떻게 생각하면 좋을지 이해가 안갑니다..
136                 // 이 부분 설명 한번만 부탁드립니다.
137
138
139                 /* DEBUG 메시지
140                 System.out.println("threadIdx: " + threadIdx +
141                         ", option: " + option +
142                         ", 내가 찾는 숫자는: " + i); */
143                 localSum = localSum.add(i);
144
145                 // System.out.println("threadIdx: " + threadIdx + ", sum = " + sum);
146             }
147         }
148
149         addAll();
150
151         System.out.println("totalSum = " + totalSum);
152     }
153 }
```

```
155 public class Quiz61 {
156     public static void main(String[] args) {
157         ThreadManager tm = new ThreadManager();
158
159         tm.calcEachBigInteger();
160     }
161 }
```

```
"C:\Program Files\
totalSum = 70
totalSum = 103
totalSum = 343

Process finished w
```