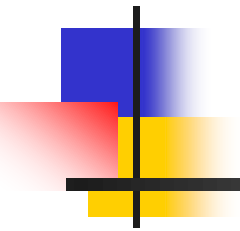
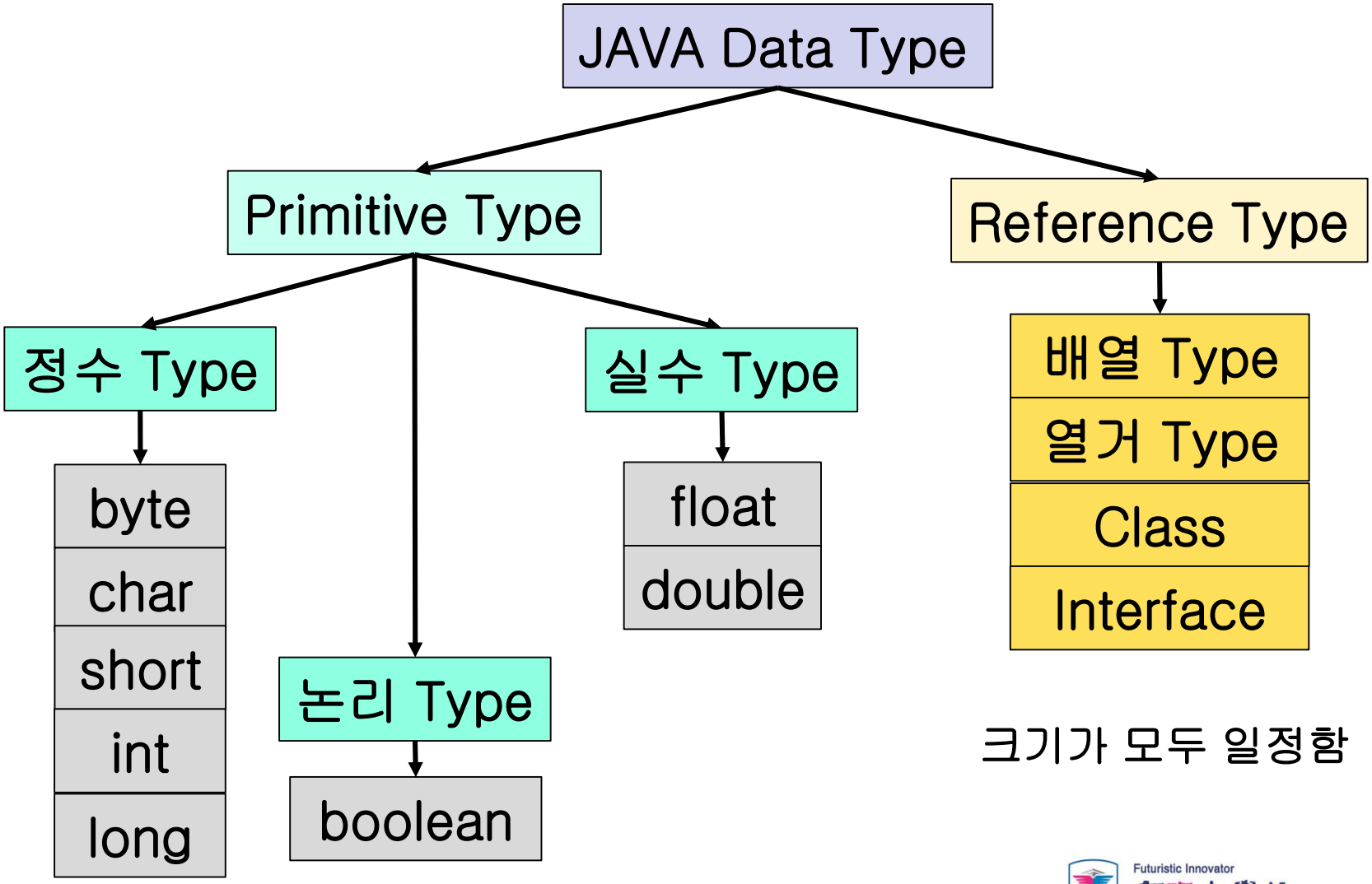


메소드(Method)와 배열

경복대학교
소프트웨어융합과
배희호 교수

Reference Type



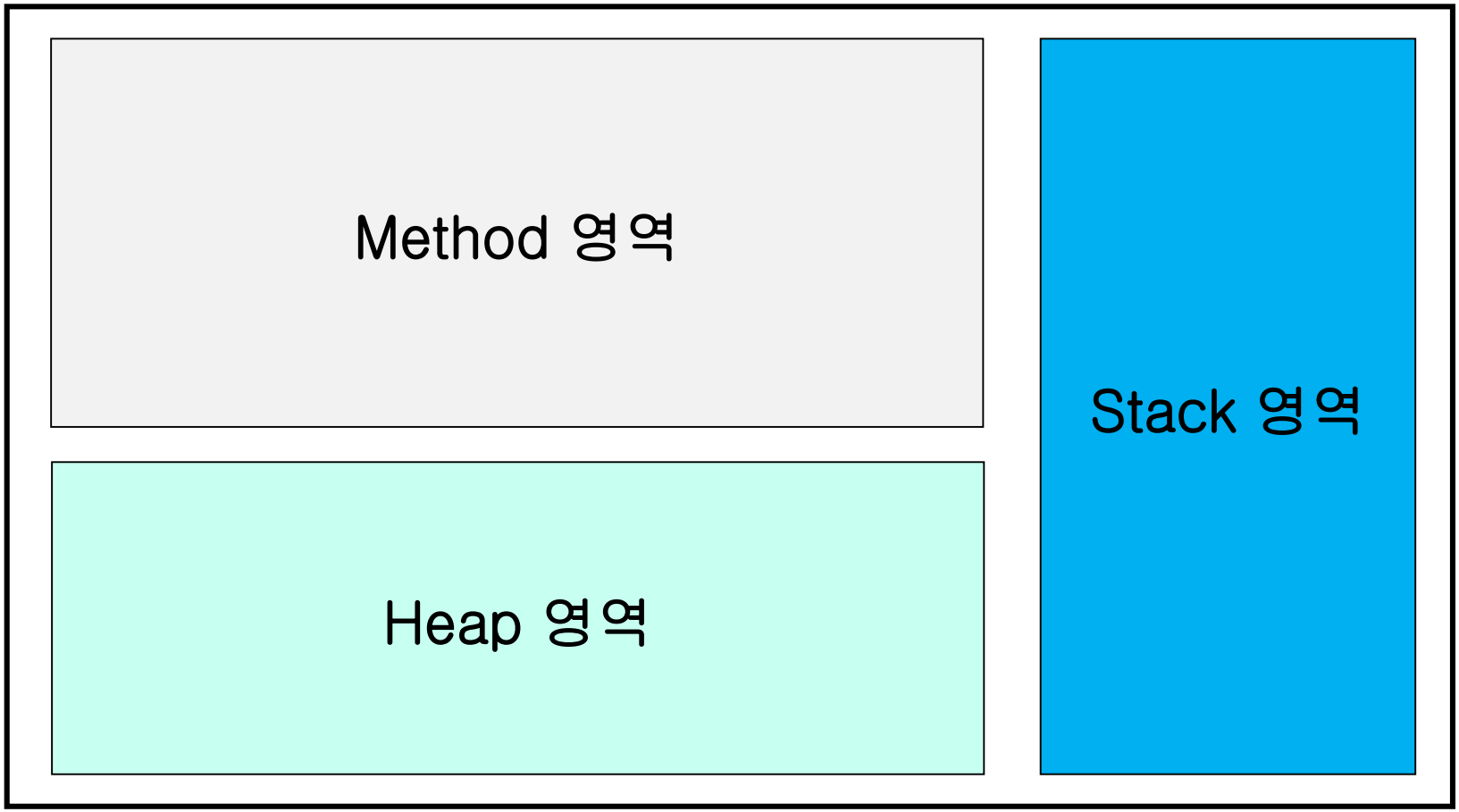
크기가 모두 일정함

Reference Type

- Reference Type 특징
  - Primitive Type과는 달리 실제 값이 저장되지 않고, Data가 저장된 공간의 Address(주소)를 저장
  - 즉, 실제 값은 다른 곳에 있으며 값이 있는 주소를 가지고 있어서 나중에 그 주소를 참조해서 값을 가져옴
  - Memory의 Heap(힙)에 실제 값을 저장하고, 그 참조 값(주소 값)을 갖는 변수는 Stack(스택)에 저장
  - Reference Type의 변수는 null로 초기화 시킬 수 있음

Reference Type

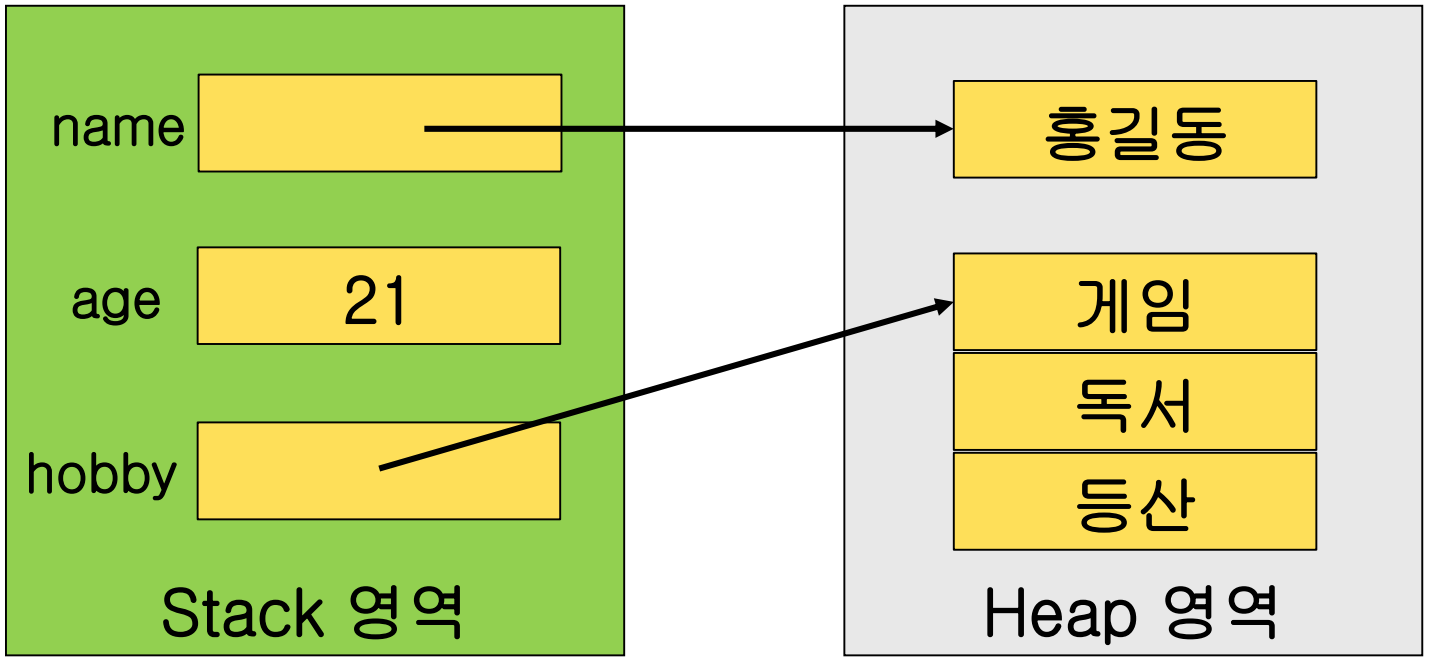
- JAVA RunTime Data Area



Reference Type

- 배열의 저장 구조

```
String name = “홍길동”;
int age = 21;
String[] hobby =  new String[]{“게임”, “독서”, “등산”};
```

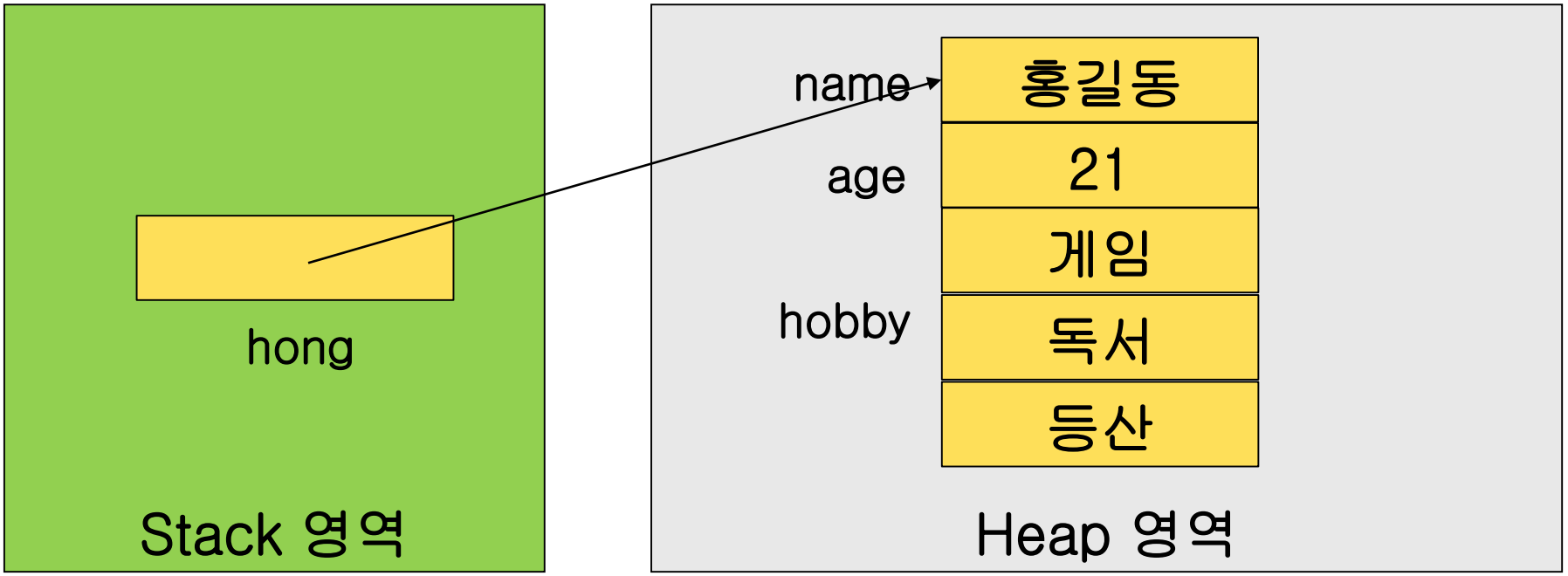


Reference Type 예제 1

```
public class Main {
    String name = "홍길동";
    int age = 21;
    String[] hobby = new String[] {"게임", "독서", "등산"};

    public static void main(String[] args) {
        Main hong = new Main();          // 객체

        System.out.printf("이름 : %s₩n", hong.name);
        System.out.printf("나이 : %s₩n", hong.age);
        System.out.printf("취미 : %s₩n", hong.hobby);
    }
}
```

Reference Type 예제 1

- Object의 저장 구조



매개변수로 배열

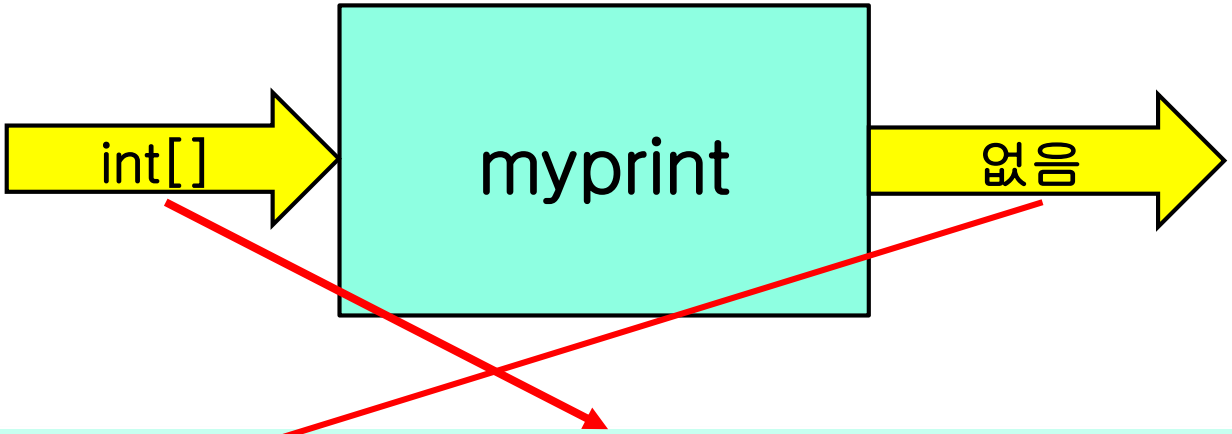
- 기본 Data Type의 값을 Method에 전달할 수 있듯이 배열을 Method의 매개변수로 전달할 수도 있음
  - 배열을 통째로 전달하는 방법은 없음
    - JAVA는 매개변수로 배열이 선언되는 것을 허용하지 않음
    - 따라서 Method의 매개변수로 배열을 전달하지 못함
  - 문자열 또한 통째로 전달하는 방법은 없음
    - 문자열은 배열의 형태로 표현되므로, 문자열도 Method의 인자로 전달하지 못함
  - 배열의 주소 값을 전달(Call By Reference)
    - JAVA는 배열을 통째로 전달하는 방법을 제공하지 않는 대신 배열의 주소 값을 전달하는 방법을 제공

매개변수로 배열

- Method의 매개변수(인자)로 배열을 전달하는 방법
  - 배열의 이름을 Method의 실 인자로 하여 Method를 호출하면, 그 배열의 첫 번째 원소의 주소가 Method로 전달됨
  - 호출되는 Method에서 대응하는 형식 인자는 그 배열의 Data Type과 동일하게 선언 함
    - 실 인자가 배열 명인 경우 형식 인자도 역시 동일한 형과 같은 크기의 배열로 선언되어야 함
  - 주의할 점으로 호출된 Method에서는 이 형식 인자를 이용하여 그 배열의 원소를 다루거나 배열 형태로 다룰 수 있다는 점임

매개변수로 배열 예제1

- 1차원 배열을 매개 변수로 받아 출력하는 myprint() 메소드 정의

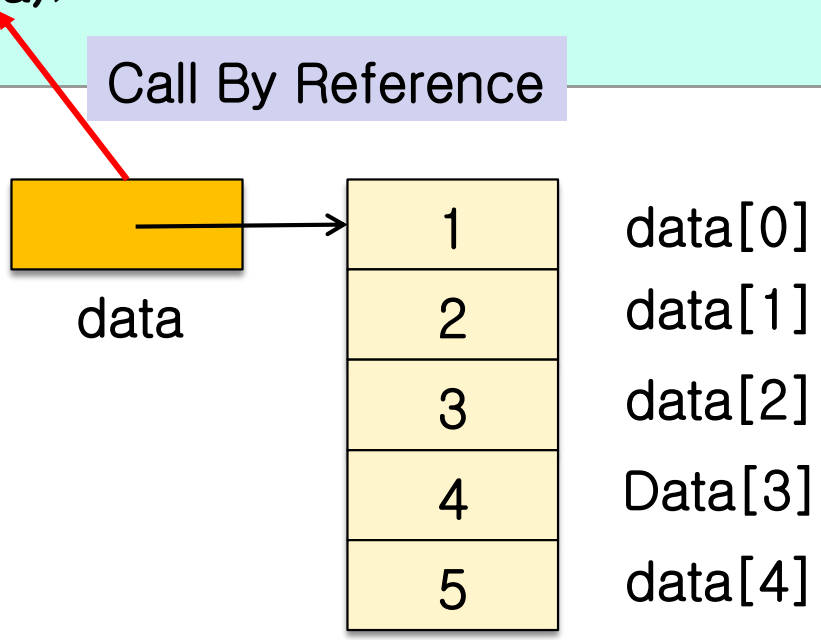



```
private static void myprint(int[] value) {
    for (int i = 0; i < value.length; i++)
        System.out.printf("%4d", value[i]);
    System.out.println();
}
```

Call By Reference

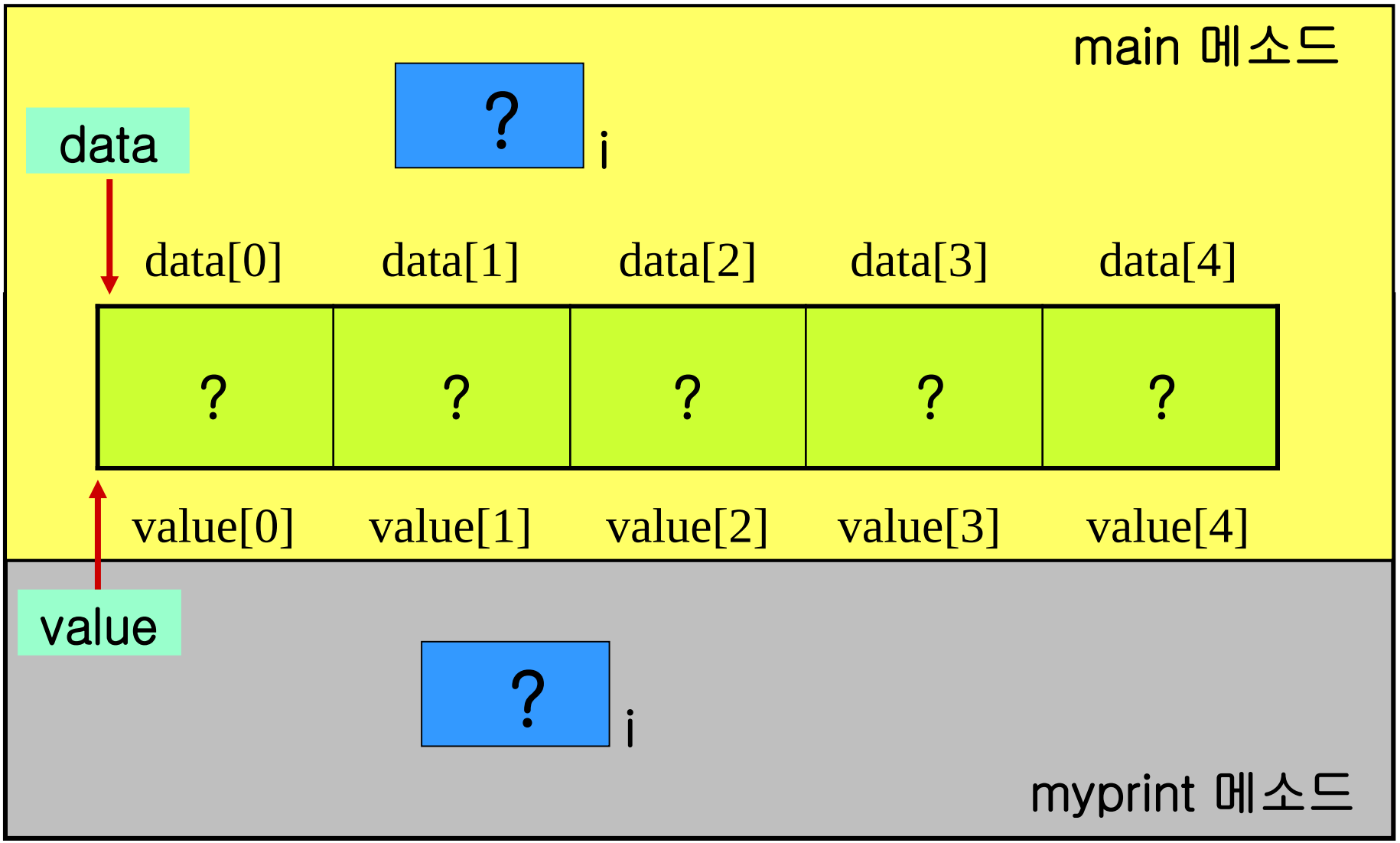
매개변수로 배열 예제1

```
public static void main(String[] args) {
    int[] data = {1, 2, 3, 4, 5};

    myprint(data);
}
```



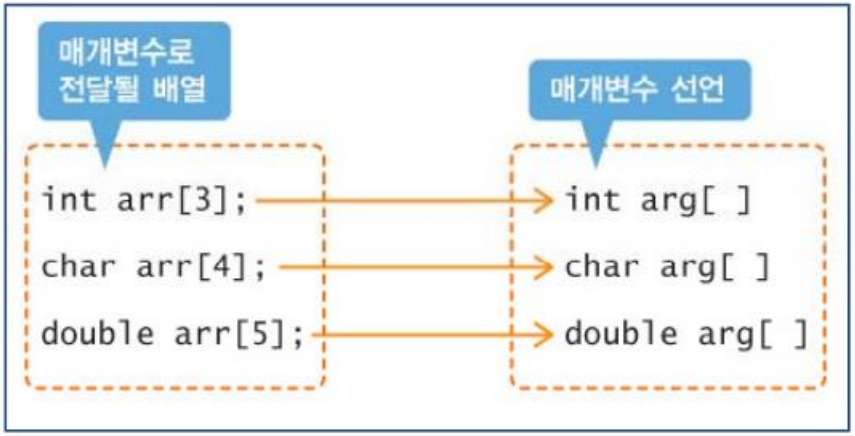
- ✓ 1차원 배열은 매개 변수로 개별 원소를 전달 할 수도 있고, 1차원 배열 전체를 전달할 수 있음
- ✓ 참조형 변수 data를 이용해서 가리키고 있는 배열

매개변수로 배열 예제1



매개변수로 배열 예제1

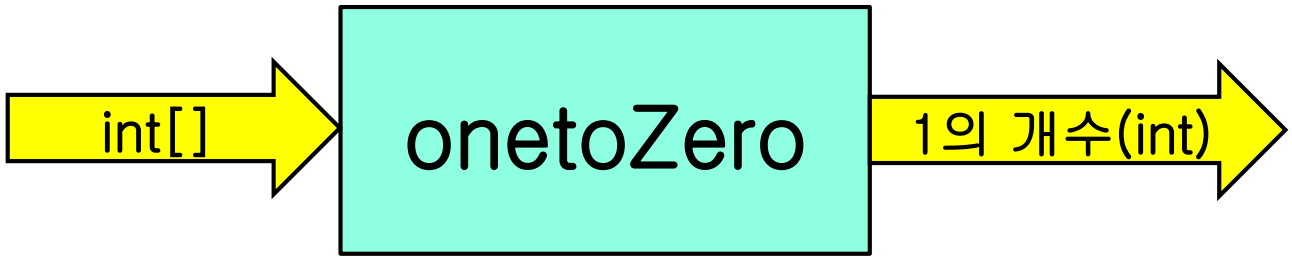
- 배열의 주소 값을 전달받는 매개변수의 선언 방식



- 매개변수의 이름은 배열의 이름처럼 사용 가능
  - 배열의 주소 값을 전달받은 매개변수는 배열의 이름처럼 배열 요소에 접근할 수 있음

매개변수로 배열 예제2

- 1차원 배열의 원소 값이 1인 원소의 개수을 반환하고, 값을 0으로 변경하는 onetoZero() 메소드를 정의




```
private static int onetoZero(int[] test) {
        int count = 0;
        for (int i = 0; i < test.length; i++) {
            if (test[i] == 1) {
                count++;
                test[i] = 0;
            }
        }
        return count;
    }
```

Call By Reference

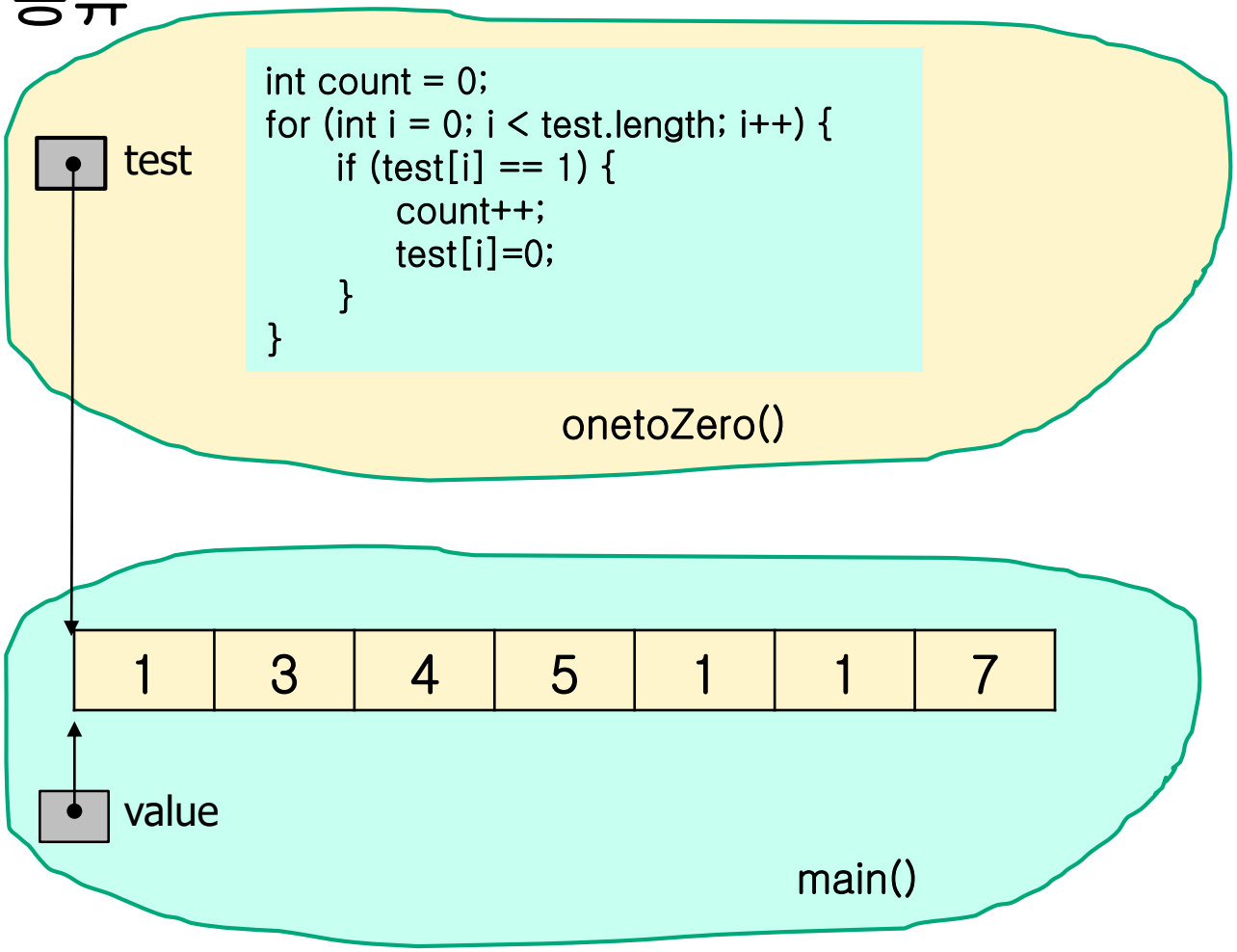
매개변수로 배열 예제2

```
public static void main(String[] args) {
    int num;
    int[] value = {1, 3, 4, 5, 1, 1, 7};

    display("Old Array : [", value);   // 혼합 형태 (value, Reference)

    num = onetoZero(value);
    System.out.printf("1의 개수 = %d₩n", num);
    display("New Array : [", value);
}
```

```
private static void display(String message, int[] value) {
    System.out.print(message);
    for (int i = 0; i < value.length; i++)
        System.out.print(value[i] + " ");
    System.out.print("]₩n");
}
```

매개변수로 배열 예제2

- 배열의 공유



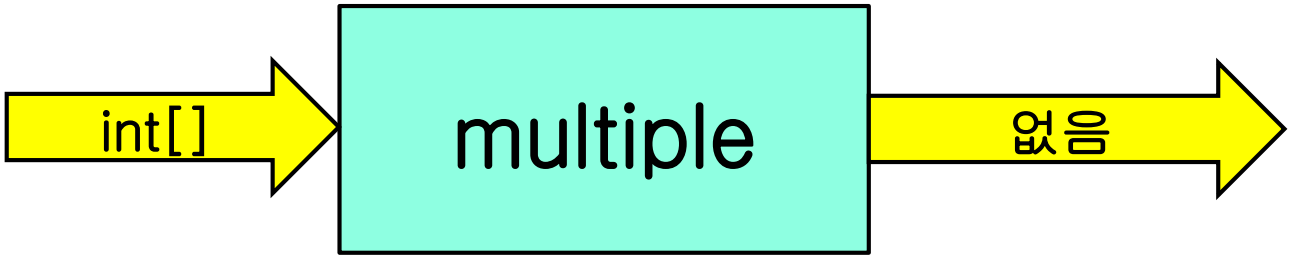
매개변수로 배열 예제2

- 보통 인수 (Call by Value)
  - Method에서 형식 인수의 값이 바뀌더라도 이 Method를 호출한 측에서 실 인수로 사용된 변수에는 영향 없음

- 배열 인수 (Call by Reference)
  - Method에서 사용되는 배열은 이 Method를 호출한 측에서 실 인수로 사용된 배열과 같은 배열 임
  - 따라서 Method에서 배열 원소가 변경되는 것은 호출한 측의 인수로 사용된 배열이 변경되는 것임
  - 배열 인수의 사용에 주의를 요함

매개변수로 배열 실습1

- 1차원 배열을 받아 원소의 값들을 2배로 변환하는 메소드 multiple() 정의



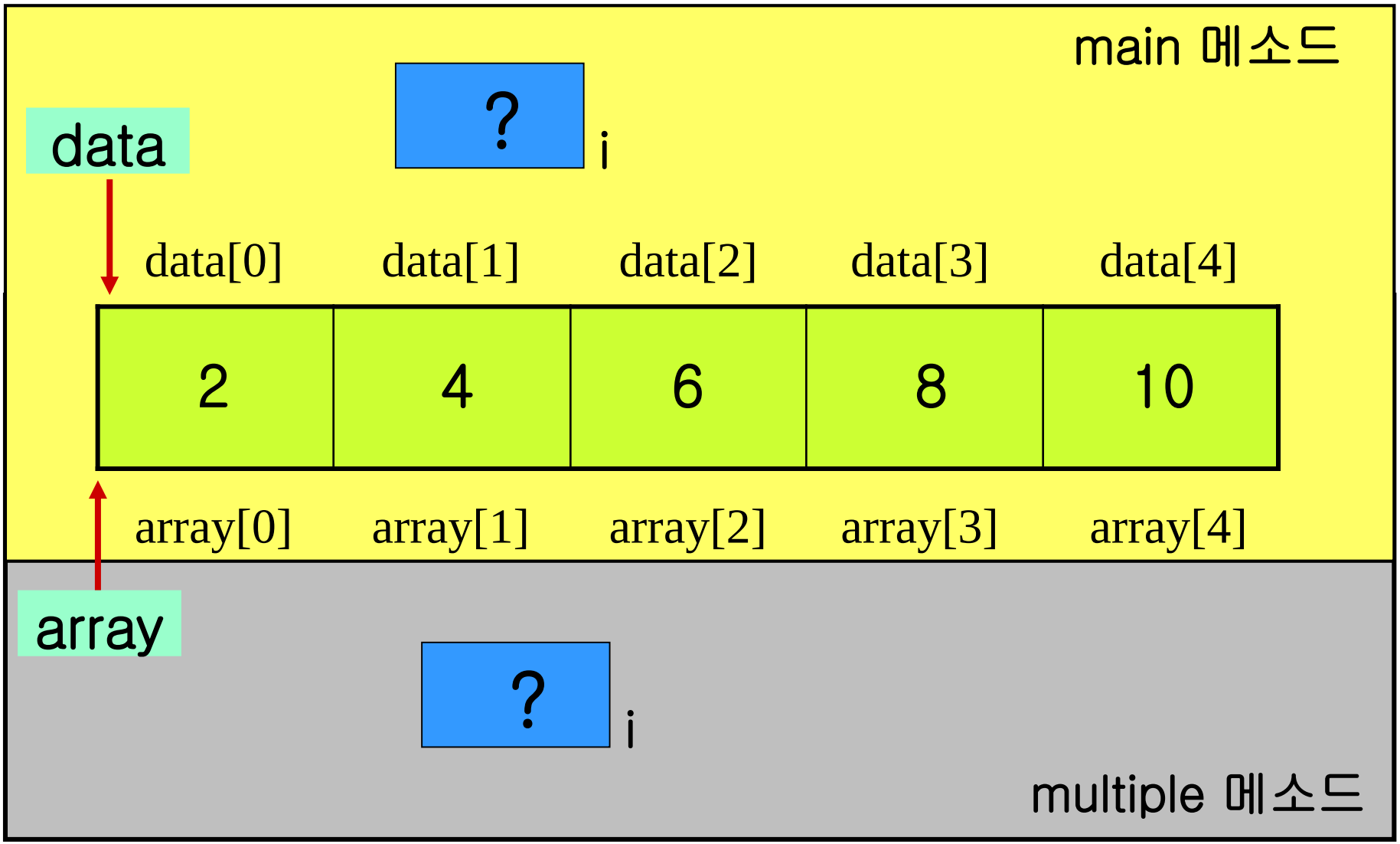

```
private static void multiple(int[ ] array) {
        for (int i = 0; i < array.length; i++)
                array[i] *= 2;
}
```

매개변수로 배열 실습1

```
public static void main(String[] args) {
        int[ ] data = {1, 2, 3, 4, 5};

        display(data, "초기");          // 혼합 형태
        multiple(data);                 // call by reference
        display(data, "이후");
}
```

```
private static void display(int[ ] test, String message) {
        System.out.printf("₩n %s 데이터 리스트₩n", message);
        for (int i = 0; i < test.length; i++)
            System.out.printf(" data[%d] = %d₩n", i, test[i]);
   }
}
```

매개변수로 배열 실습1



매개변수로 배열 실습1[심화]

```
public static void main(String[] args) {
    int[] data = {1, 2, 3, 4, 5};

    display(data, "초기");
    for (int i = 0; i < data.length; i++)
        data[i] = multiple(data[i]);      // call by value
    display(data, "이후");
}
```

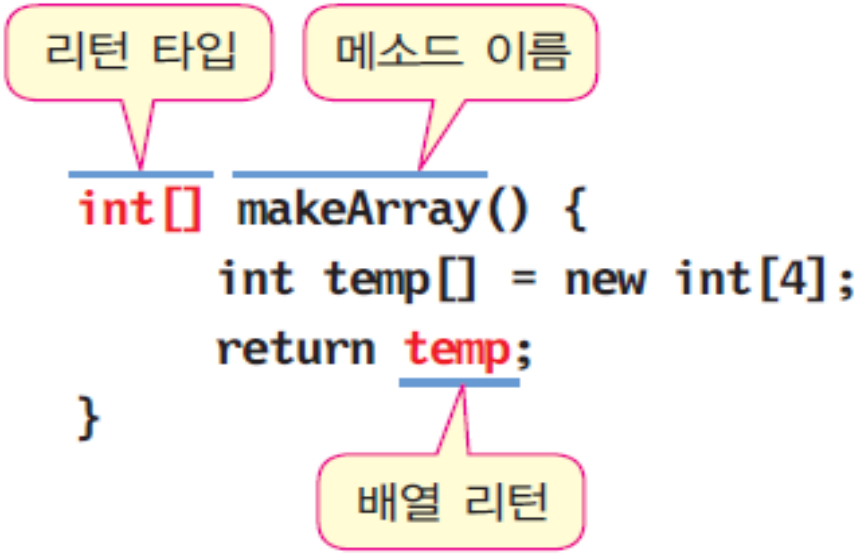
```
private static int multiple(int data) {
    data *= 2;

    return data;
}
```

반환 값으로의 배열

- Method는 여러 개의 입력 값을 가질 수 있음. 그렇다면 여러 개의 값을 반환하고 싶다면?
  - JAVA는 문법적으로 그런 기능을 제공하지 않음
  - Method에서 여러 개의 값의 반환을 위해서는 하나의 변수에 여러 개의 값을 담아서 출력하면 됨
    - 배열을 사용하거나 사용자 정의 변수(클래스)를 사용하면 됨

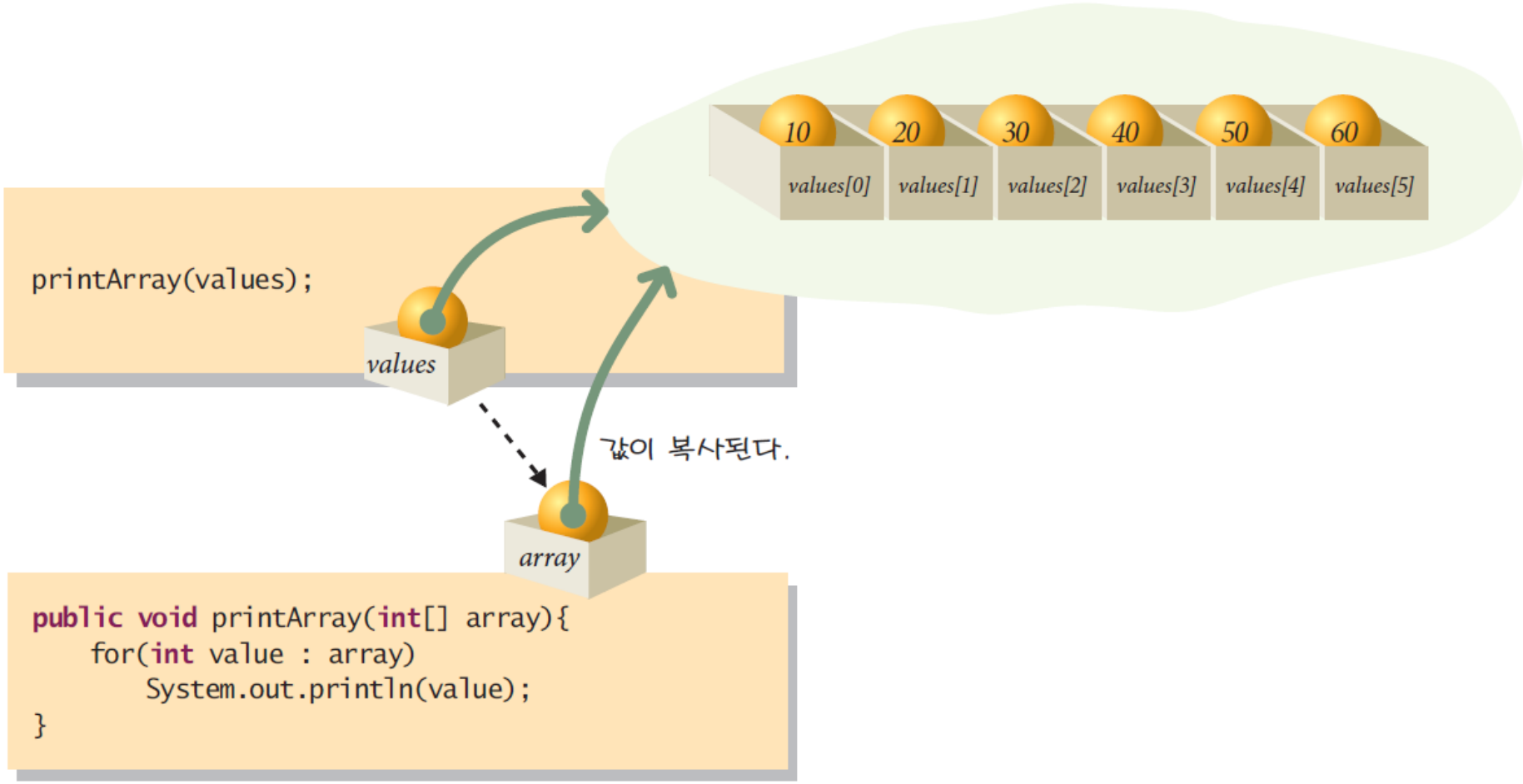
반환 값으로의 배열

- Method는 배열을 반환할 수도 있음
- Method가 반환하는 배열
  - Method가 반환하는 배열의 Data Type과 차원은 반환 받는 배열 Reference의 Type과 차원에 일치해야 함
  - 반환 Type에 배열의 크기를 지정하지 않음
  - 배열 객체를 반환하기 위해 Method의 반환 형에 배열 기호 [ ]를 명시



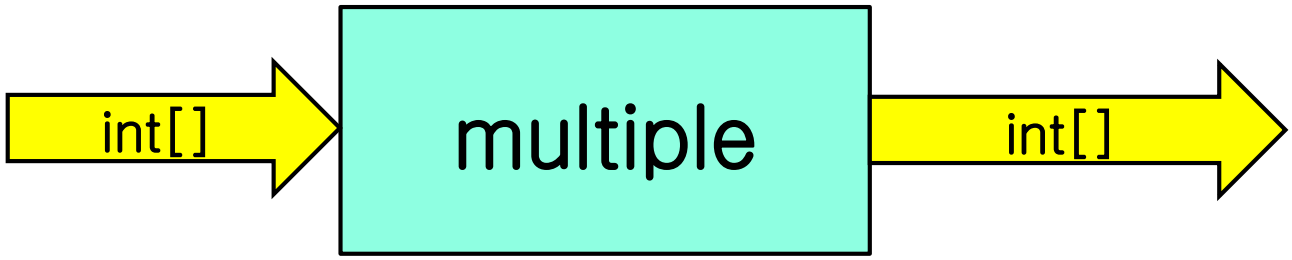
반환값으로의 배열

- Method와 배열



반환값으로의 배열 실습

- main() 메소드에서 multiple() 메소드의 1차원 배열을 받아서 출력하는 프로그램을 만들어보자




```
private static int[] multiple(int[] test){
    int[] temp = new int[test.length];

    for (int i = 0; i < temp.length; i++){
        temp[i] = test[i] * 2;
    }
    return temp;
}
```

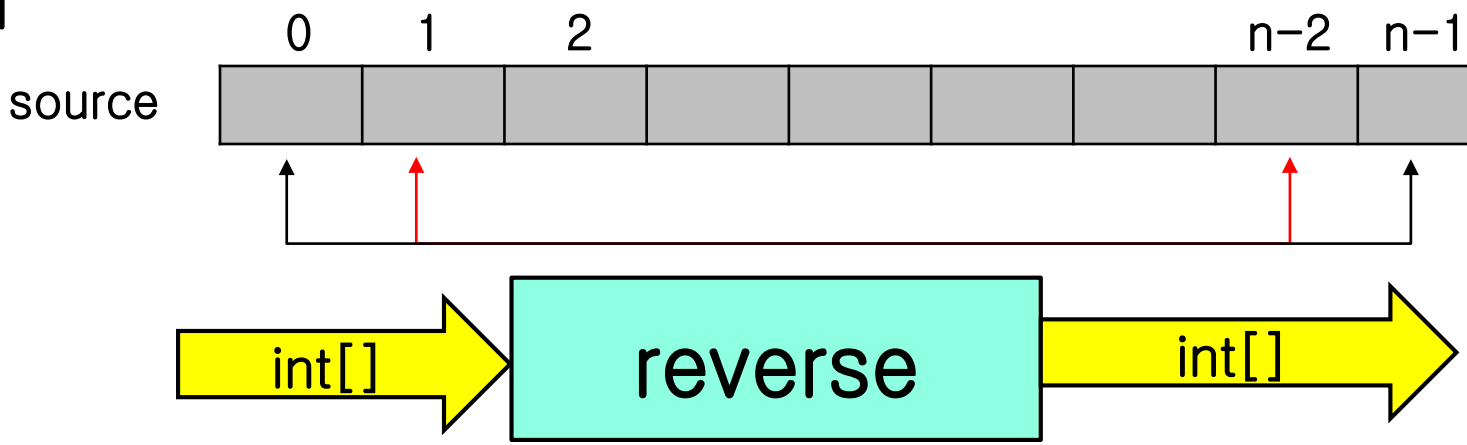
반환값으로의 배열 실습

```
public static void main(String[] args) {
    int[ ] data = {1, 2, 3, 4, 5};

    display(data, "초기");
    data = multiple(data);                    // call by reference
    display(data, "이후");
}
```

```
private static void display(int[ ] test, String message) {
    System.out.printf("₩n %s 데이터 리스트₩n", message);
    for (int i = 0; i < test.length; i++)
        System.out.printf(" data[%d] = %d₩n", i, test[i]);
}
```

반환값으로의 배열[심화]

- 1차원 배열의 원소를 거꾸로(reverse) 복사하는 메소드를 정의

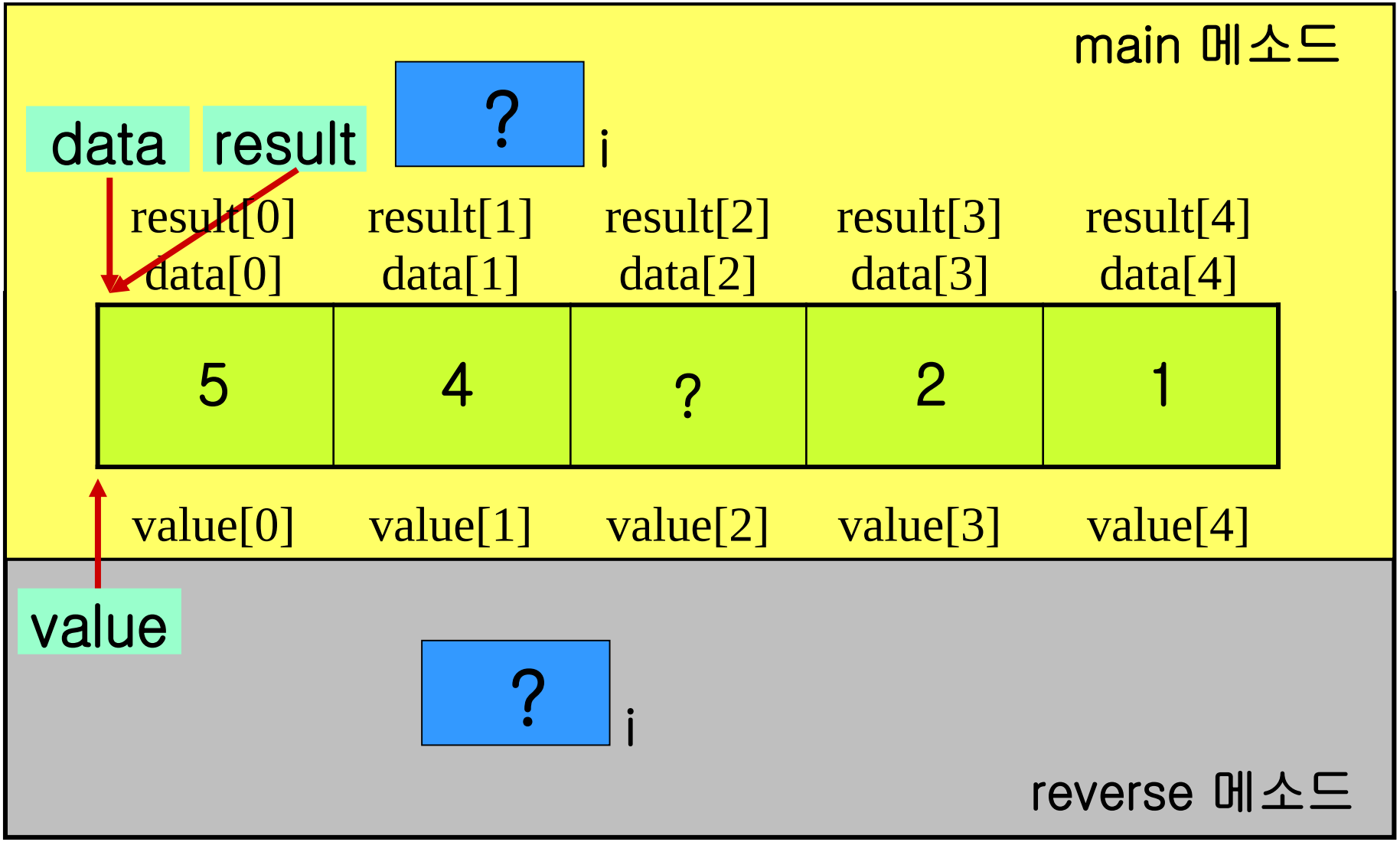



```
public static int[] reverse(int[] value) {
    for (int i = 0; i < value.length / 2; i++) {
        int temp = value[value.length - 1 - i];
        value[value.length - 1 - i] = value[i];
        value[i] = temp;
    }

    return value;
}
```

반환값으로의 배열[심화]

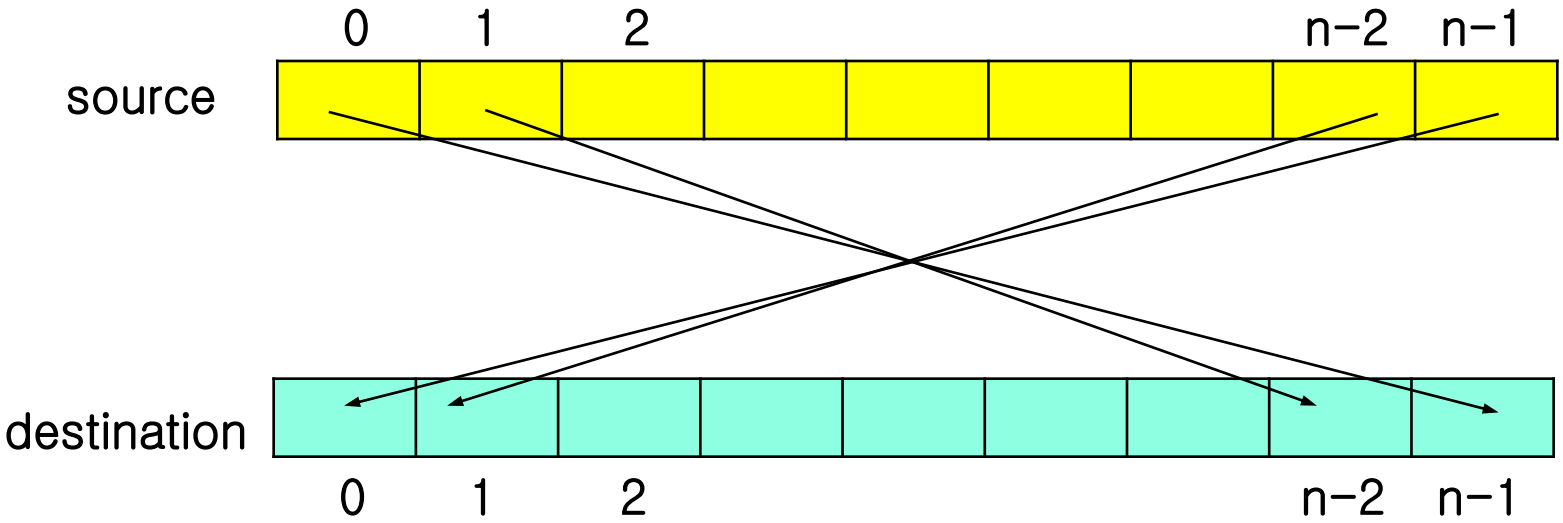
```
public static void main(String[] args) {
    int[] data = new int[]{1, 2, 3, 4, 5};
    int[] result;

    for (int i = 0; i < result.length; i++) {
        System.out.printf("%4d", data[i]);
    }
    System.out.println();

    result = reverse(data);
    for (int i = 0; i < result.length; i++) {
        System.out.printf("%4d", result[i]);
    }
}
```

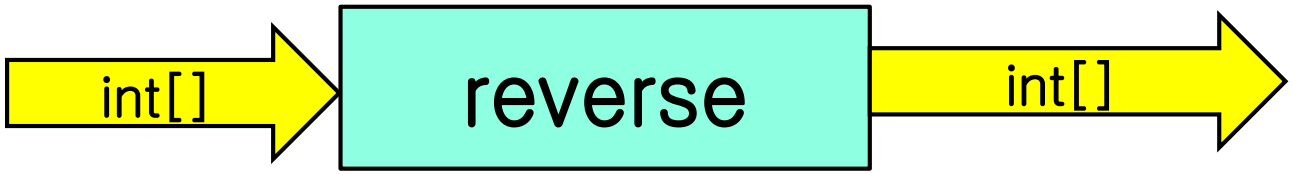
반환값으로의 배열[심화]



반환값으로의 배열[심화]

- 1차원 배열의 원소를 거꾸로 복사하는 reverse() 메소드를 정의





반환값으로의 배열[심화]

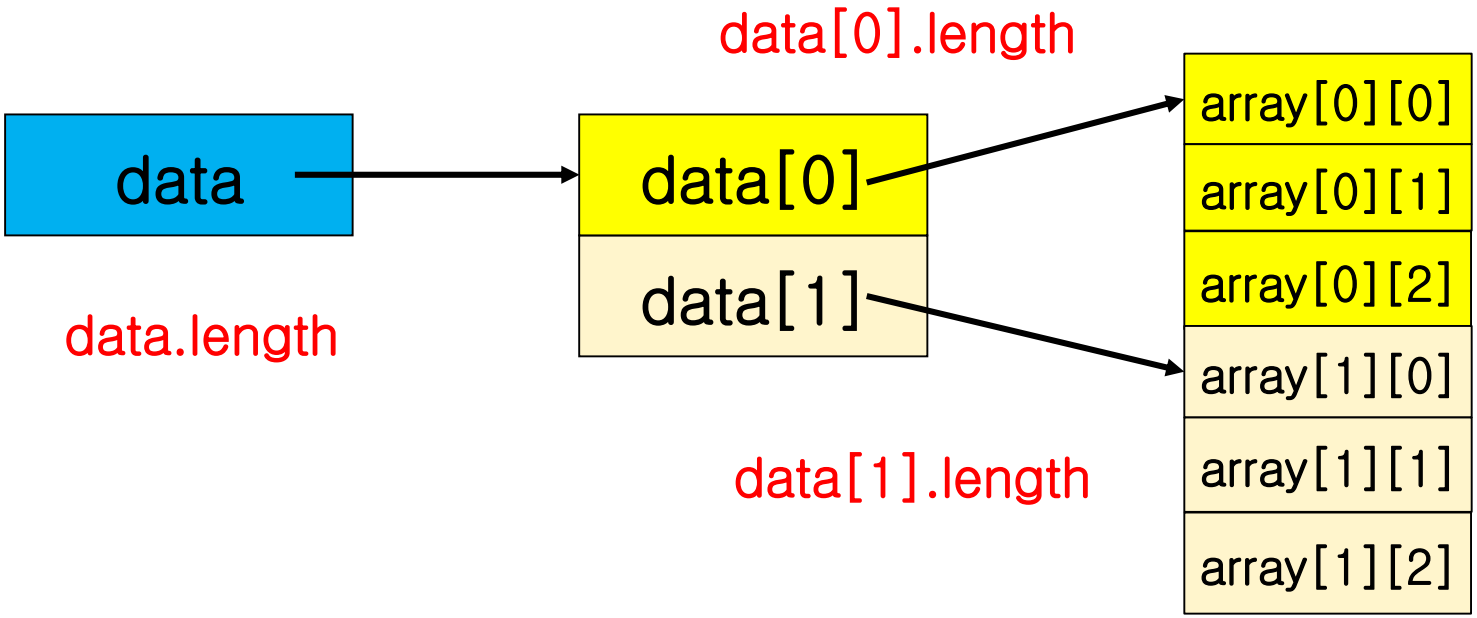
```
private static int[] reversecopy(int[] source) {
    int[] destination = new int[source.length];

    for (int from = 0, from < source.length - 1; from++) {
        destination[source.length - 1 - from] = source[from];
    }

    return destination;
}
```

메소드와 다차원 배열

- 1차원 배열을 메소드의 매개변수로 사용할 수 있듯이 다차원 배열을 메소드의 매개변수로 사용할 수 있음
- 1차원 배열은 1차원 배열을 전달하든지, 또는 각 원소의 값을 전달할 수 있음
- 2차원 배열은 2차원 배열을 전달할 수도 있지만, 또 따른 방법은 부분적인 1차원 배열을 전달할 수도 있고, 각 원소의 값을 전달할 수도 있음

메소드와 다차원 배열

- 2차원 배열 = 배열의 배열
  - 2차원 배열은 배열 요소로 1차원 배열을 갖음



메소드와 다차원 배열

- 기존에 작성했던 메소드(multiple() : 1차원 배열을 매개변수로 받음)를 사용하여 2차원 배열에 적용하기

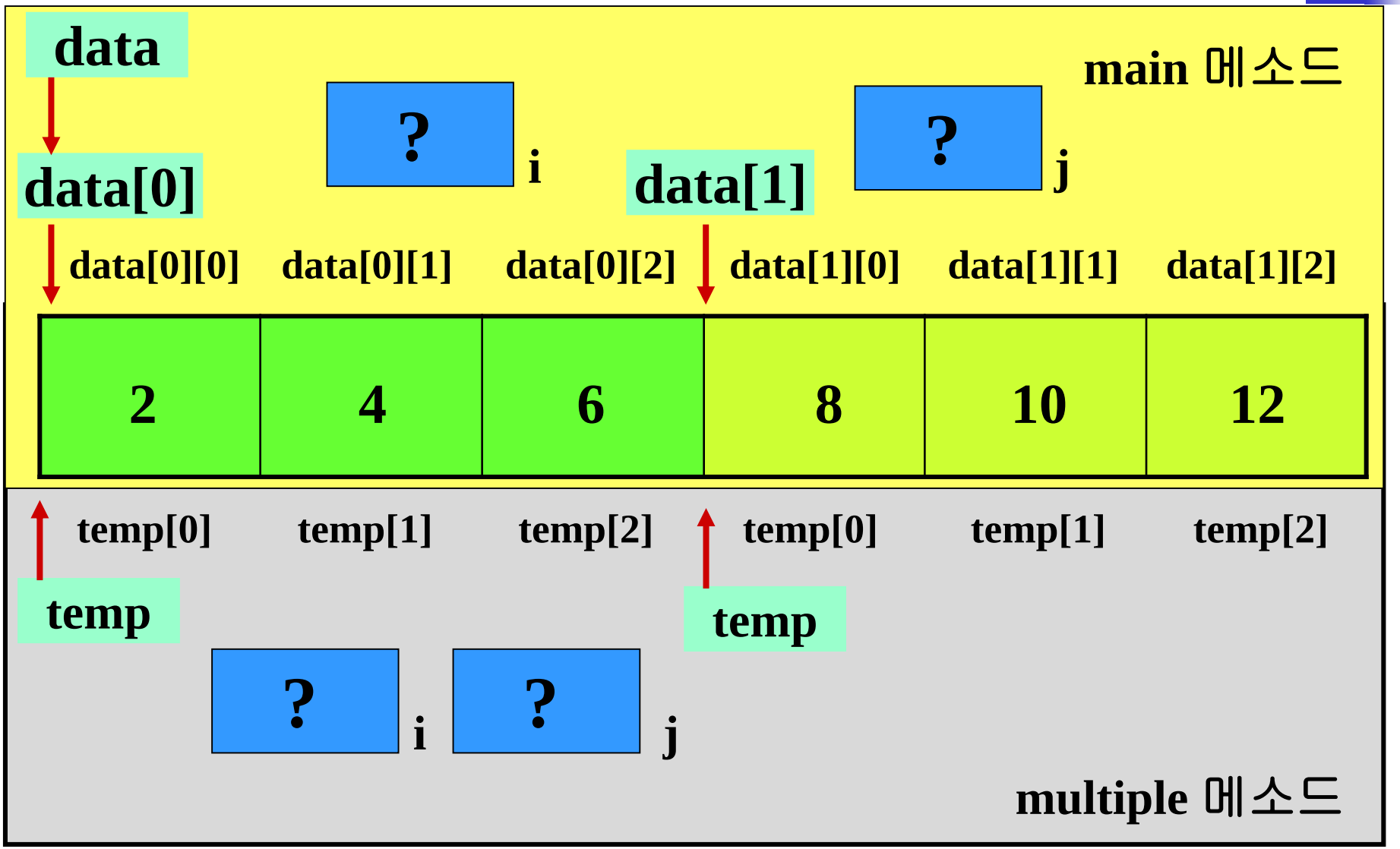
```
private static void multiple(int[] temp) {
    for (int i = 0; i < temp.length; i++)
        temp[i] *= 2;
}
```

메소드와 다차원 배열

```
public static void main(String[] args) {
        int[ ][] data = {{1, 2, 3}, {4, 5, 6}};

        display(data, "초기");
        for (int i = 0; i < data.length, i++)
                multiple(data[i]);
        display(data, "이후");
}
```

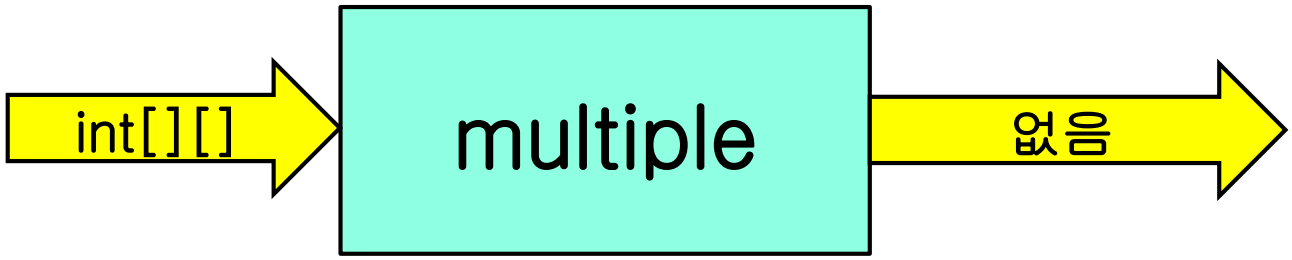
```
private static void display(int[ ][ ] test, String message) {
   System.out.printf("₩n %s 데이터 리스트₩n", message);
   for (int i = 0; i < test.length; i++)
      for (int j = 0; j < test[i].length; j++)
         System.out.printf(" data[%d][%d] = %d₩n", i, j, test[i][j]);
}
```

메소드와 다차원 배열



메소드와 다차원 배열

- 2차원 배열을 전달 받아 2배의 값으로 변경하는 multiple() 메소드를 정의



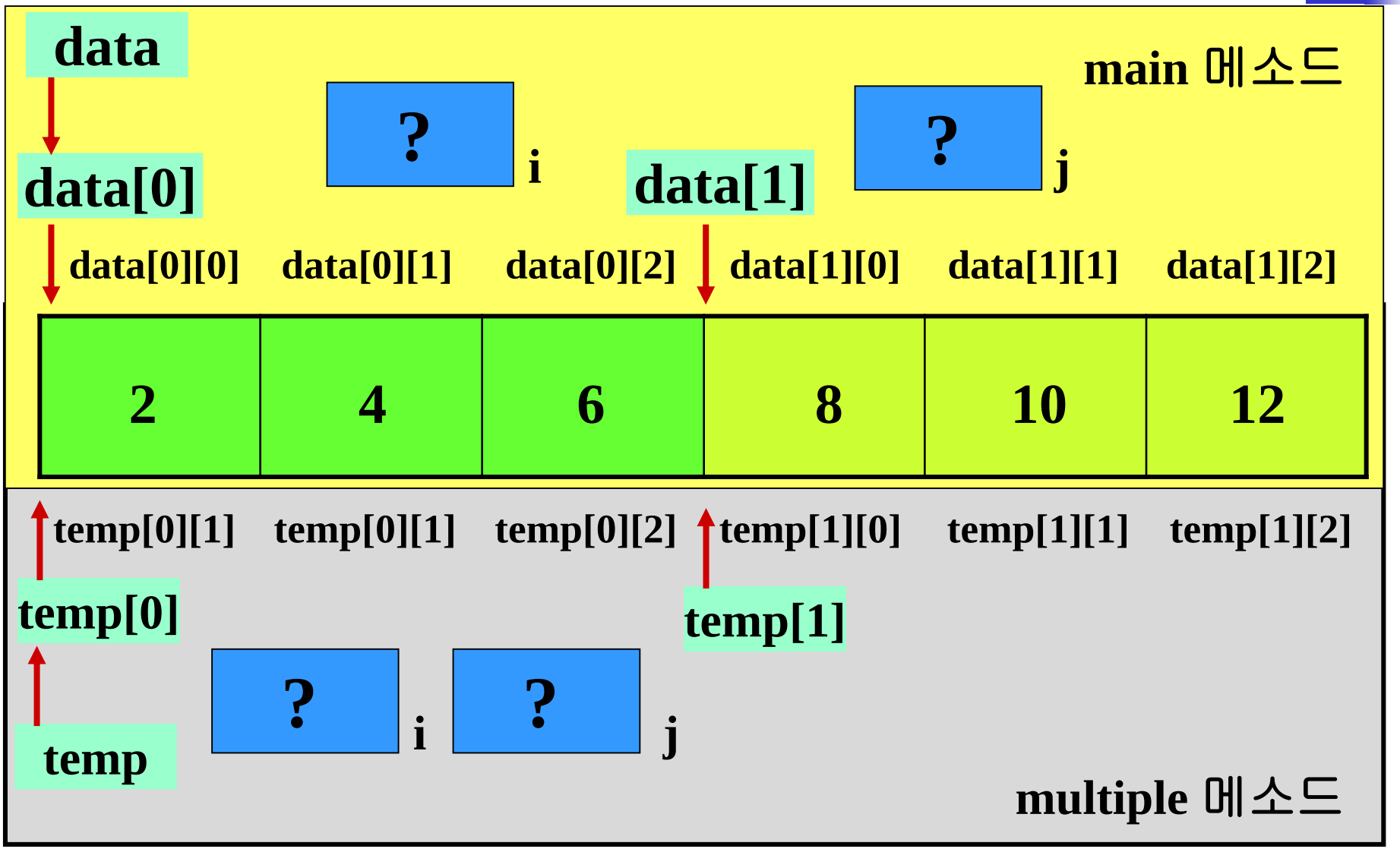

```
private static void multiple(int[ ][ ] temp) {
    for (int i = 0; i < temp.length; i++)
        for (int j = 0; j < temp[i].length; j++)
            temp[i][j] *= 2;
}
```

메소드와 다차원 배열

```
public static void main(String[] args) {
    int[][] data = {{1, 2, 3}, {4, 5, 6}};

    display(data, "초기");
    multiple(data);
    display(data, "이후");
}
```

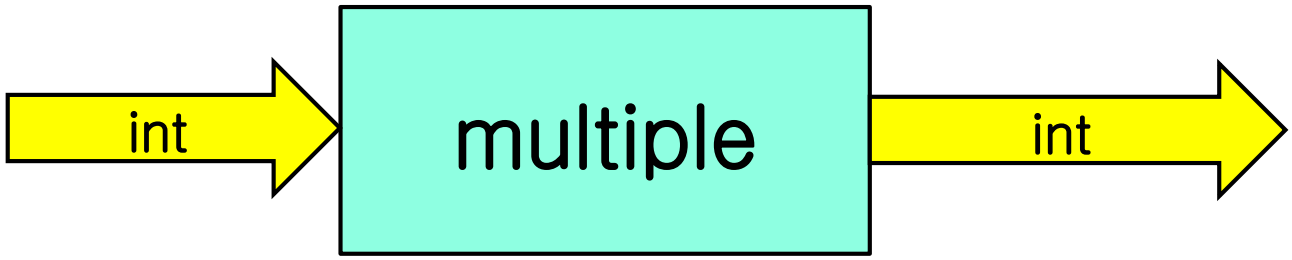
```
private static void display(int[][] test, String message) {
    System.out.printf("₩n %s 데이터 리스트₩n", message);
    for (int i = 0; i < test.length; i++)
        for (int j = 0; j < test[i].length; j++)
            System.out.printf(" data[%d][%d] = %d₩n", i, j, test[i][j]);
}
```

메소드와 다차원 배열



메소드와 다차원 배열

- 기본적인 int data를 받아 2배로 반환하는 multiple() 메소드 정의



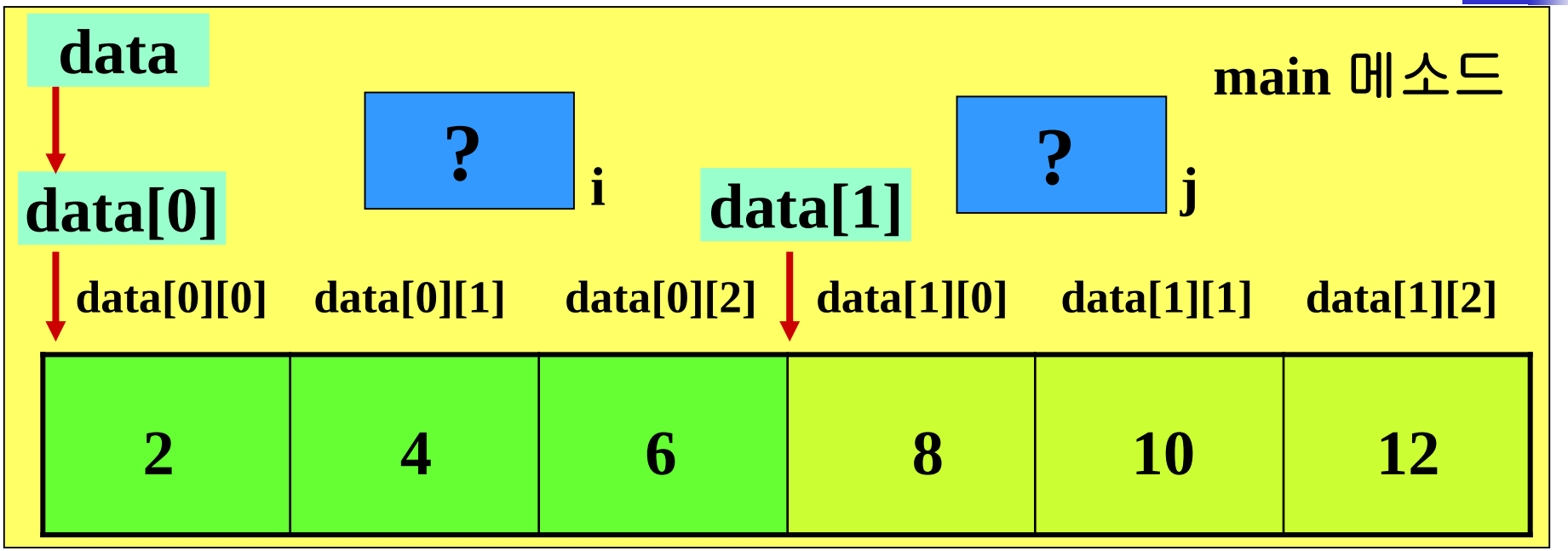

```
private static int multiple(int data) {

   return data *= 2;
}
```

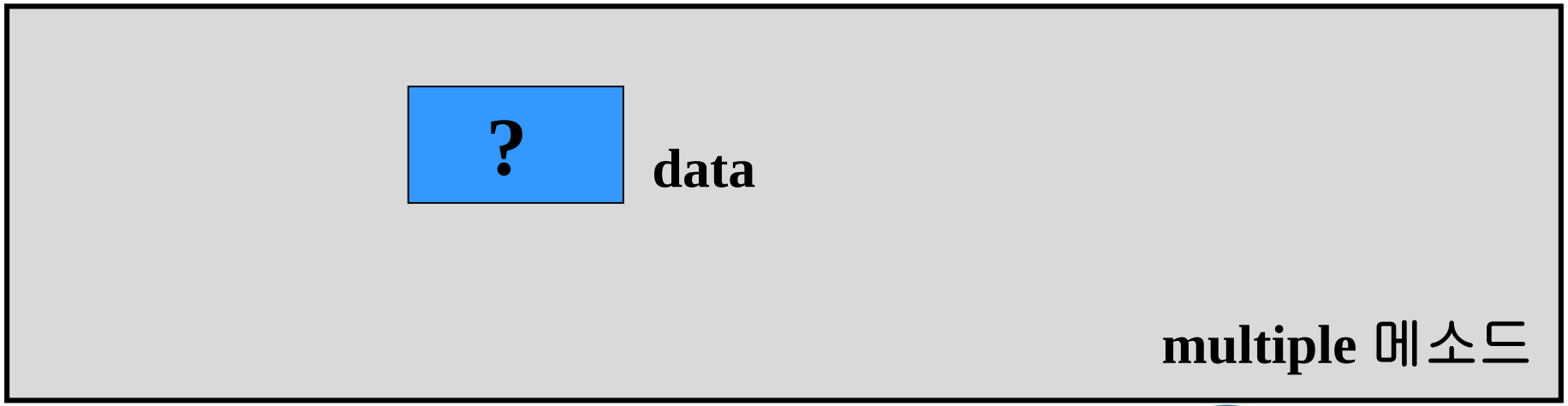
메소드와 다차원 배열

```
public static void main(String[] args) {
    int[ ][] data = {{1, 2, 3}, {4, 5, 6}};

    display(data, "초기");
    for (int i = 0; i < data.length; i++)
        for (int j = 0; j < data[i].length; j++)
            data[i][j] = multiple(data[i][j]);
    display(data, "이후");
}
```

```
private static void display(int[ ][ ] test, String message) {
    System.out.printf("₩n %s 데이터 리스트₩n", message);
    for (int i = 0; i < test.length; i++)
        for (int j = 0; j < test[i].length; j++)
            System.out.printf(" data[%d][%d] = %d₩n", i, j, test[i][j]);
}
```

메소드와 다차원 배열





Varargs(Variable arguments)

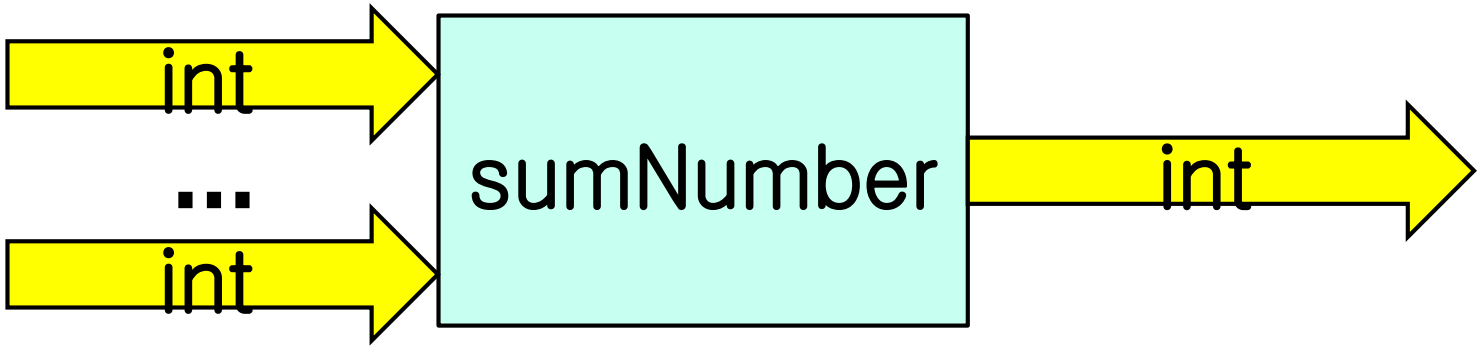
- 가변 인자라고 부르는 Varargs 또한 JDK 5에서 도입된 기능
- 이 Varargs는 필요에 따라 매개변수의 개수를 가변적으로 조정할 수 있는 기능
- 모든 인자 개수에 대해 함수를 작성할 수는 없으므로, JDK 5 이전에는 일정 개수 이상의 인자를 처리 할 때는 보통 Collection이나 Array를 사용했음

Varargs(Variable arguments)

- Varargs를 사용하면서 주의해야할 점
  - Varargs로 선언된 함수를 호출하면, 컴파일러가 배열을 함수 실행 직전에 생성하도록 코드를 변경하는데, 이것이 문제가 될 수 있음
  - 특정 메소드를 실행하는데 인자의 개수가 일정 이상으로 증가할 필요가 없다면, 단순하게 인자의 개수가 다른 메소드를 Overloading으로 생성하는게 성능상 더 좋음
  - 특히, 반복문 안에서 실행하는 경우에는 매 실행 시마다 배열을 만들어야 하기 때문에 더 문제임

Varargs 예제 1

- 매개변수의 개수가 정해지지 않은 받은 데이터를 더하여 합을 구하는 sumNumber() 메소드를 정의




```
private static int sumNumber(int... num) {
    int total = 0;

    for (int i = 0; i < num.length; i++) //전달인자의 개수만큼 반복
        total += num[i];

    return total;
}
```

Varargs 예제 1

```
private static void numPrint(int... num) {
    for (int i = 0; i < num.length - 1; i++)
        System.out.printf("%d + ", num[i]);
    System.out.print("₩b₩b₩b = ");
    System.out.printf("%d₩n", num[num.length - 1]);
}
```

```
public static void main(String[] args) {
    numPrint(10, 20, 30, 40, sumNumber(10, 20, 30, 40));
    numPrint(100, 250, 130, sumNumber(100, 250, 130));
    numPrint(50, 40, sumNumber(50, 40));
    numPrint(60, sumNumber(60));
}
```

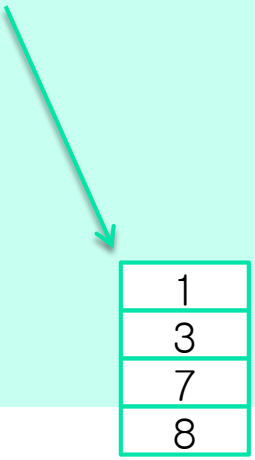
Varargs 예제 2

- 메소드를 구현할 때 파라미터 수를 미리 정하지 않을 수 있음 (가변수 파라미터)

```
class Adder {
    public int add(int... values) { ... }
}

Adder adder = new Adder();
adder.add(1, 3, 7);
adder.add(1, 2);
adder.add(1, 3, 7, 8);
```

메소드 매개변수의 개수

```
class Adder {
   public int add(int... values) { ... }
}

Adder adder = new Adder();
adder.add(1, 3, 7);
adder.add(1, 2);
adder.add(1, 3, 7, 8);
```



① 인자들이 배열로 만들어진다

② 배열을 가리키는 참조가 파라미터로 복사된다

메소드 매개변수의 개수

```
class Adder
    public int add(int... values) {
            int sum = 0;
            for (int i = 0; i < values.length; i++)
                    sum += values[i];
            return sum;
    }
```

1
3
7
8

메소드 매개변수의 개수

```
public class VarArgsTest {
        public static void main(String args[ ]) {
            VarArgs test = new VarArgs( );
            test.sub(1);
            test.sub(2, 3, 4, 5, 6);
            test.sub( );
        }
}
```

가변인자는 항상 맨 뒤에 와야한다
myMethod(int i, String... strings) 이런 식으로는 가능하지만,
myMethod(String... strings, int i) 이렇게 먼저와 버리면 안된다

2023년 오늘이 무슨 요일?

- 2023년 1월 1일은 일요일이다. 2023년 A월 B일은 무슨 요일일까요?

2023년 오늘이 무슨 요일?

- 문제 분석
  - 입력
    - 월(month) – int (1 ~ 12)
    - 일(day) – int (1 ~ 28, 29, 30, 31)
  - 출력
    - 요일명(weekName) – String
      - 월요일, 화요일, 수요일, 목요일, 금요일, 토요일, 일요일
  - 계산 방법
    - 오늘이 그해의 몇 번째 날인지를 계산하여, 7로 나누어 나머지를 가지고 요일을 계산

2023년 오늘이 무슨 요일?

```
public class Main {
    private static int[] days = {31, 28, 31, 30, 31, 30, 31, 31, 30,
                                                        31, 30, 31};

    public static void main(String[] args) throws IOException {
        int year = 2023, day, month;
        if (leapYear(year))
            days[1] = 29;
        month = inputMonth();
        day = inputDay(month);
        System.out.println(year + "년 " + month + "월 " + day + "일은 " +
                dayName(month, day) + "입니다.");
    }
```

2023년 오늘이 무슨 요일?

```
private static String dayName(int month, int day) {
    int count = 0;
    if (month > 1) {
        for (int i = 1; i < month; i++) {
            count += days[i - 1];
        }
    }
    count += day;
    int week = count % 7;
```

2023년 오늘이 무슨 요일?

```
    switch (week) {
        case 1:
            return "일요일";
        case 2:
            return "월요일";
        case 3:
            return "화요일";
        case 4:
            return "수요일";
        case 5:
            return "목요일";
        case 6:
            return "금요일";
        default:
            return "토요일";
    }
}
```

2023년 오늘이 무슨 요일?

```
private static int inputMonth() throws IOException {
    Scanner keyboard = new Scanner(System.in);
    int month;
    while (true) {
        System.out.print("월을 입력하세요 (1 ~ 12) : ");
        month = keyboard.nextInt();
        if (month >= 1 && month <= 12)
            break;
        else {
            System.err.print("Error - 월의 범위를 확인하세요");
            System.in.read();
        }
    }
    return month;
}
```

2023년 오늘이 무슨 요일?

```
private static int inputDay(int month) throws IOException {
    Scanner keyboard = new Scanner(System.in);
    int day;
    while (true) {
        System.out.printf("일을 입력하세요(1 ~ %d) : ", days[month - 1]);
        day = keyboard.nextInt();
        if (day >= 1 && day <= days[month - 1])
            break;
        else {
            System.err.print("Error - 날자의 범위를 확인하세요");
            System.in.read();
        }
    }
    return day;
}
```

2023년 오늘이 무슨 요일?

```
private static boolean leapYear(int year) {
    boolean result = false;
    if (((year % 4 == 0) && (year % 100 != 0)) || year % 400 == 0)
        result = true;
    return result;
}
```

2023년 오늘이 무슨 요일?(II)

```
public static void main(String[] args) throws IOException {
    int year = 2023, day, month;
    int[] days = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    if (leapYear(year))
        days[1] = 29;
    month = inputMonth();
    day = inputDay(month, days);
    System.out.println(year + "년 " + month + "월 " + day + "일은 " +
            dayName(month, day, days) + "요일 입니다." );
}
```

2023년 오늘이 무슨 요일?(II)

```
private static boolean leapYear(int year) {
    boolean result = false;
    if (((year % 4 == 0) && (year % 100 != 0)) || year % 400 == 0)
        result = true;
    return result;
}
```

2023년 오늘이 무슨 요일?(II)

```
private static int inputMonth() throws IOException {
    Scanner keyboard = new Scanner(System.in);
    int month;
    while (true) {
        System.out.print("월을 입력하세요 (1 ~ 12) : ");
        month = keyboard.nextInt();
        if (month >= 1 && month <= 12)
            break;
        else {
            System.err.print("Error - 월의 범위를 확인하세요");
            System.in.read();
        }
    }
    return month;
}
```

2023년 오늘이 무슨 요일?(II)

```
private static int inputDay(int month, int[] days) throws IOException {
    Scanner keyboard = new Scanner(System.in);
    int day;
    while (true) {
        System.out.printf("일을 입력하세요(1 ~ %d) : ", days[month - 1]);
        day = keyboard.nextInt();
        if (day >= 1 && day <= days[month - 1])
            break;
        else {
            System.err.print("Error - 날자의 범위를 확인하세요");
            System.in.read();
        }
    }
    return day;
}
```

2023년 오늘이 무슨 요일?(II)

```
private static String dayName(int month, int day, int[] days) {
    int count = 0;
    if (month > 1) {
        for (int i = 1; i < month; i++) {
            count += days[i - 1];
        }
    }
    count += day;
    int week = count % 7;
```

2023년 오늘이 무슨 요일?(II)

```
    switch (week) {
        case 1:
            return "일요일";
        case 2:
            return "월요일";
        case 3:
            return "화요일";
        case 4:
            return "수요일";
        case 5:
            return "목요일";
        case 6:
            return "금요일";
        default:
            return "토요일";
    }
}
```

오늘은 ?

- 년 월 일을 입력 받아 다음과 같은 정보를 출력하는 프로그램을 만들어보자

년 월 일 입력 : 2023 11 13<Enter>
2023년은 계묘년(토끼띠) 입니다.
2023년 11(November)월 13일은 월요일(Monday) 입니다.
2023년 11월 13일은 2023년의 317번째 날입니다.

- Hint
  - 1900년 1월 1일부터 입력
  - 1900년 1월 1일 월요일 임

최대값과 최소값

- 10개의 정수를 입력 받아 합과, 최대값과 최소값을 찾는 Program을 사용자 정의 Method를 이용하여 작성하여라.

최대값과 최소값

- 문제 이해

1	45
2	56
3	76
4	98
5	54
6	34
7	29
8	92
9	88
10	69

합= 641
최대값 = 98
최소값 = 29

1단계: max = min = 45
2단계: max = 56 min = 45
3단계: max = 76 min = 45
4단계: max = 98 min = 45
……
6단계: max = 98 min = 29
……

최대값과 최소값

- 문제 해결 방법
  - 최대값, 최소값 구하기
    - 처음 입력된 데이터를 max와 min의 초기값으로 넣음
    - 다음 데이터들 중 max보다 큰 값이 들어오면 max에는 그 값을 대치
    - min보다 작은 값이 들어오면 min에는 그 값을 대치
    - 최종적으로 max에는 가장 큰 값이, min에는 가장 작은 값이 남게 됨
  - 10개의 데이터가 입력되고 나면 최종적으로 구해진 합과 최대값, 최소값을 출력

최대값과 최소값

- 입출력 변수 선정
  - int data;
    - 10개의 데이터 값을 입력할 변수
  - int max;
    - 최대값을 저장할 변수
    - 입력된 데이터가 max에 있는 값보다 크면 max에 그 값을 저장
  - int min;
    - 최소값을 구할 변수
    - 입력된 데이터가 min에 있는 값보다 작으면 min에 그 값을 저장
  - int sum = 0;
    - 합을 저장할 변수
  - int i;
    - 반복 제어 변수로 사용

최대값과 최소값

- 실행 시 화면
  - 입력 형태
    - 프로그램이 실행될 때 다음과 같은 메시지를 내 보내어 10개의 데이터를 입력
    - “1 번째 데이터를 입력 하시오 ?”
    - “2 번째 데이터를 입력 하시오 ?”
    - ……
  - 출력 형태
    - 최대값, 최소값, 합을 구한 후 다음과 같이 출력
    - “합 = 641”
    - “최대값 = 98”
    - “최소값 = 29”

최대값과 최소값

- 가상언어표현

i = 0에서 9까지 변하는 동안

data 값을 입력한다

sum = sum + data // data값을 누적 시킨다

i = 0이면 max = data, min = data로 대치. 아부분을 반복수행

그렇지 않으면 data > max 이면 max = data를 대치

data < min 이면 min = data를 대치

sum, max, min을 출력한다.

최대값과 최소값

```
public static void main(String[] args) {
    int[] data = {90, 80, 70, 100, 80, 90, 80, 100, 60, 50};
    int total, max, min;

    dataRead(data);

    total = total(data);
    max = max(data);
    min = min(data);

    dataPrint(data, total, max, min);
}
```

최대값과 최소값

```
private static void dataRead(int[] data) {
    Scanner keyboard = new Scanner(System.in);
    int i = 0;
    while (true) {
        System.out.printf("(%d/%d) 번째 Data 입력 : ", i + 1, data.length);
        data[i] = keyboard.nextInt();
        i++;
        if (i >= data.length)
            break;
    }
}
```

최대값과 최소값

```
private static int total(int[] data) {
    int total = 0;
    for (int i = 0; i < data.length; i++)
        total += data[i];
    return total;
}
```

```
private static void dataPrint(int[] data, int total, int max, int min) {
    for (int i = 0; i < data.length; i++)
        System.out.printf("%2d번째 Data = %d₩n", i + 1, data[i]);
    System.out.printf("₩n 합계 = %d₩n", total);
    System.out.printf(" MAX = %d₩n", max);
    System.out.printf(" MIN = %d₩n", min);
}
```

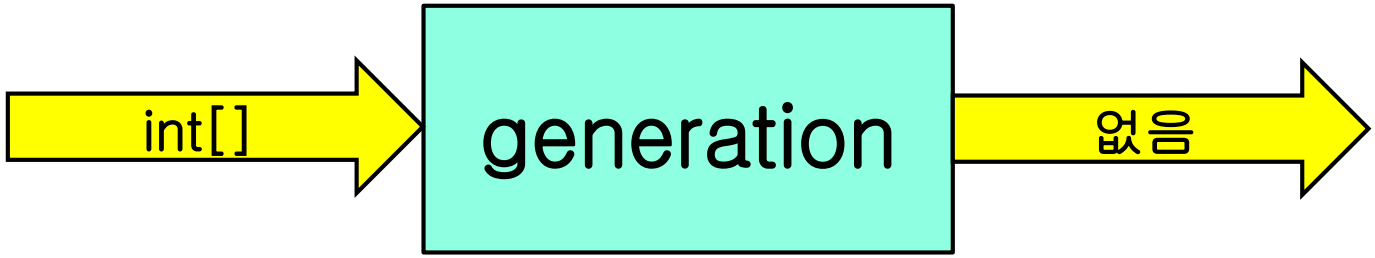
최대값과 최소값

```
private static int max(int[] data) {
    int max = data[0];
    for (int i = 1; i < data.length; i++)
        if (max < data[i])
            max = data[i];
    return max;
}
```

```
private static int min(int[] data) {
    int min = data[0];
    for (int i = 1; i < data.length; i++)
        if (min > data[i])
            min = data[i];
    return min;
}
```

Lotto 번호 생성

- 중복이 없는 6개의 숫자를 랜덤으로 뽑기
  - 해당 인덱스 위치에 랜덤의 숫자를 뽑고 나서 이전까지 뽑은 값들이 있는지 비교 후, 중복 값이 있으면 해당 위치의 값을 다시 뽑고 다시 비교 중복 값이 없으면 다음을 뽑는다



Lotto 번호 생성

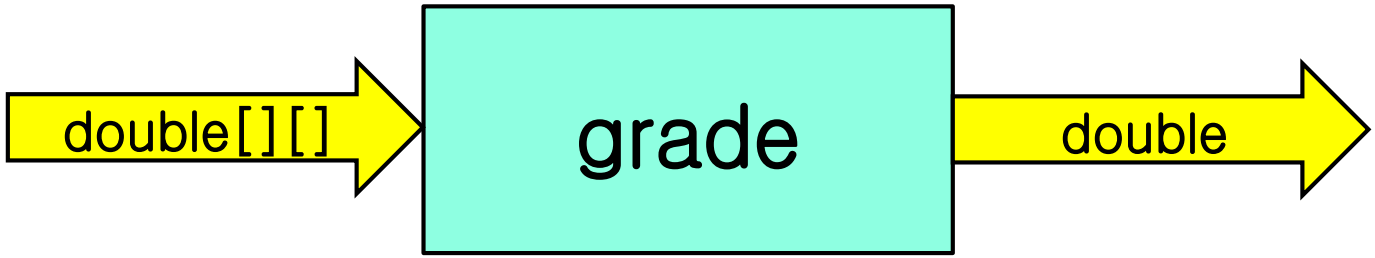
```
public static void main(String[] args) {
    int[] lotto = new int[6];

    generation(lotto);

    output(lotto);
}
```

Lotto 번호 생성

```
private static void output(int[] lotto) {
    System.out.print("Lotto 번호 : ");
    for (int i = 0; i < lotto.length; i++)
        System.out.print(lotto[i] + ", ");
    System.out.printf("₩b₩b₩n");
}

private static void generation(int[] lotto) {
    Random random = new Random(System.currentTimeMillis());
    for (int i = 0; i < lotto.length; i++) {
        lotto[i] = random.nextInt(45) + 1;
        for (int j = 0; j < i; j++) {
            if (lotto[i] == lotto[j]) {
                i--;
            }
        }
    }
}
```

대학의 평균 평점

- 대학의 평균 평점을 구하는 grade()를 정의하여라.




```
private static double process(double[][] grade) {
    double sum = 0;

    for(int year = 0; year < grade.length; year++)     //학년별
        for(int term = 0; term < grade[year].length;term++) // 학기별
            sum += grade[year][term];

    return (sum / (grade.length * grade[0].length));
}
```

대학의 평균 평점

```
public static void main(String[] args) throws IOException {
    double[][] score = {{3.3, 3.4}, // 1학년 1, 2학기 평점
                        {3.5, 3.6}, // 2학년 1, 2학기 평점
                        {3.7, 4.0}, // 3학년 1, 2학기 평점
                        {4.1, 4.2} }; // 4학년 1, 2학기 평점
    double avg;

    input(score);
    avg = grade(score);

    System.out.printf("%d학년 전체 평점 평균 : %.4.1f₩n",
                                                score.length, avg);
}
```

대학의 평균 평점

```
private static void input(double[][] grade) throws IOException {
    Scanner keyboard = new Scanner(System.in);

    for (int year = 0; year < grade.length; year++) //각 학년별로 반복
        for (int term = 0; term < grade[year].length; term++) {
            System.out.printf("%d학년 %d학기 평점 입력 : ", year + 1,
                                                        term + 1);
            grade[year][term] = keyboard.nextDouble();
            if (grade[year][term] < 0.0 || grade[year][term] > 4.5) {
                System.err.println("입력 오류");
                System.in.read();
                term--;
            }
        }
}
```

암호화/복호화

- 다음은 알파벳과 숫자를 아래에 주어진 암호표로 암호화하고 복호화하는 메소드를 정의하고, 프로그램하여라
  - ASCII 97 ~ 122
    - a b c d e f g h i j k l m n o p q r s t u v w x y z
    - ` ~ ! @ # $ % ^ & * ( ) - _ + = | [ ] { } ; : , . /
  - ASCII 48 ~ 57
    - 0 1 2 3 4 5 6 7 8 9
    - q w e r t y u i o p

암호화/복호화

```
public class Main {
    static char[] abcCode = {'`', '~', '!', '@', '#', '$', '%', '^', '&',
                             '*', '(', ')', '-', '_', '+', '=', '|', '[',
                             ']', '{', '}', ';', ':', ',', '.', '/'};
    static char[] numCode = {'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p'};

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        String source;
        String encryption,  decryption;

        System.out.print("문자열 입력 : ");
        source = keyboard.nextLine();
        encryption = encrypt(source);
        decryption = decrypt(encryption);

        System.out.println("원 본 : " + source);
        System.out.println("암호화 : " + encryption);
        System.out.println("복호화 : " + decryption);
    }
```

암호화/복호화

```
private static String decrypt(String encryption) {
    String result = "";
    for (int i = 0; i < encryption.length(); i++) {
        char ch = encryption.charAt(i);
        int index = 0;
        if (ch >= 'a' && ch <= 'z') {
            for (int j = 0; j < numCode.length; j++) {
                if (ch == numCode[j]) {
                    index = j;
                    break;
                }
            }
            index += 48;
            result += (char) index;
```

암호화/복호화

```
            } else {
                for (int j = 0; j < abcCode.length; j++) {
                    if (ch == abcCode[j]) {
                        index = j;
                        break;
                    }
                }
                index += 97;
                result += (char) index;
            }
        }
        return result;
    }
```

암호화/복호화

```
    private static String encrypt(String source) {
        String result = "";

        for (int i = 0; i < source.length(); i++) {
            char ch = source.charAt(i);
            int index = ch;
            if (ch >= 'a' && ch <= 'z') {
                result += abcCode[index - 97];
            } else if (ch >= '0' && ch <= '9') {
                result += numCode[index - 48];
            }
        }
        return result;
    }
}
```