

경기과학기술대학교 · 인공지능학과

자바 프로그래밍

값을 여러 개 담는 그릇

이름을 열 개 짓는 대신,
번호가 붙은 칸을 만든다.

4주차

오늘의 강의

1 교시 배열의 선언과 생성, 2 교시 length 와 for-each 문, 3 교시 2차원 배열과 배열 리턴

1교시



배열 만들기

선언과 생성은 다른 일
인덱스는 0부터

2교시



length 와 for-each

배열은 크기를 스스로 안
다
훑기만 할 때 쓰는 반복

3교시



2차원과 리턴

행과 열
배열을 돌려주는 메소드

1교시

배열의 선언과 생성

이름 하나에 번호가 붙은 칸을 여러 개 만든다.

선언과 생성

인덱스 0부터

레퍼런스 공유

지난주에서 이어받는 것



for 문

배열을 처음부터 끝까지
훑는 도구



중첩 반복

3교시 2차원 배열에서
사용



조건문

배열에서 가장 큰 값을
고르는 자리

오늘 새로 배우는 것: 값을 여러 개 담는 그릇

반복문 고르기 기준

반복 횟수나 범위가 명확하면 for 문.

횟수를 처음부터 알 수 없고 반복하면서 조건이 달라지면 while 문이나 do-while 문.

변수 열 개로 더하면

```
int i0, i1, i2, i3, i4, i5, i6, i7, i8, i9;
```

```
sum = i0+i1+i2+i3+i4+i5+i6+i7+i8+i9;
```

1 —
2 —

- 변수 이름을 열 개나 따로 지어야 한다
- 더하는 식도 한 줄로 길게 늘어난다

배열로 바꾸면



변수 열 개

```
int i0, i1, ..., i9;  
sum = i0 + i1 + ... + i9;
```



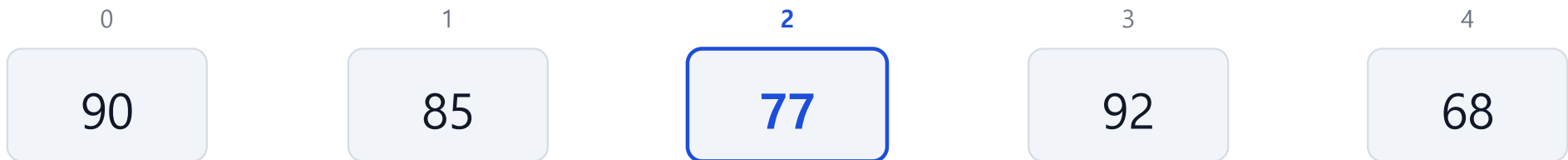
배열 + for 문

```
int i[] = new int[10];  
for(int n=0;n<10;n++) sum+=i[n];
```

핵심: 배열과 for 문을 쓰면 같은 일을 두 줄로 끝난다

배열이란

배열은 인덱스와 인덱스에 대응하는 데이터들로 이루어진 연속적인 자료 구조다.
배열에는 같은 종류의 데이터만 순차적으로 저장된다.



정수(int) 배열에는 정수만 저장된다. 실수를 넣으면 컴파일 오류가 난다.

1000개라면

1000개

변수 이름을 모두 지어야 한다

```
sum = i0 + i1 + ... + i999;
```

작성이 거의 불가능하다

서랍장과 서랍 번호

서랍장 하나에 서랍이 다섯 개 있다.

- 서랍장 이름 하나, 서랍마다 번호
- 배열 이름 하나에 원소 번호

서랍장



배열

서랍 번호



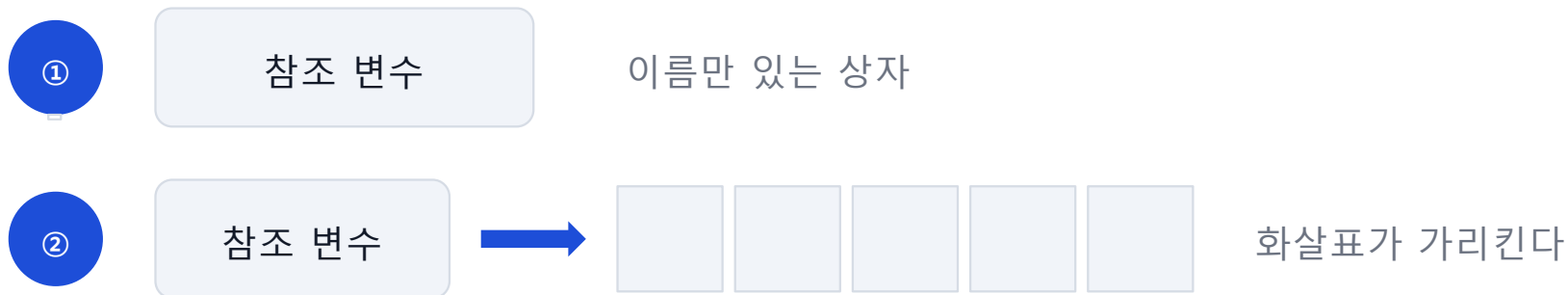
인덱스



배열은 두 단계

배열 선언은 레퍼런스 변수를 만드는 것이고, 배열 생성은 `new` 로 공간을 만드는 것이다.

선언만 한 배열 변수는 `null` 이고, 생성해야 원소에 값을 넣을 수 있다.



레퍼런스 변수 선언

```
int intArray[];
```

- 이 줄로는 배열 공간이 생기지 않는다
- `intArray` 의 값은 `null` 이다

선언하는 두 가지 방법

이 수업 방식

```
int intArray[];
```

변수 이름 뒤에 대괄호

=

다른 표기(같은 뜻)

```
int[] intArray;
```

타입 이름 뒤에 대괄호

이 수업은 교재를 따라 `int intArray[];` 로 통일한다.

선언에 크기를 적으면

내 작업공간/Ex01_DeclareError.java

```
public class Ex01_DeclareError {  
    public static void main(String[] args) {  
        int intArray[5];  
        intArray[0] = 10;  
        System.out.println(intArray[0]);  
    }  
}
```

컴파일 오류 화면

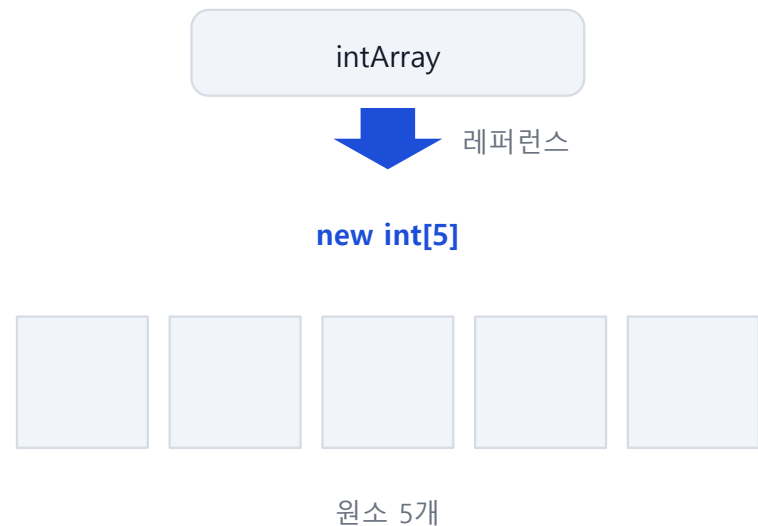
```
error: ']' expected
```

선언에 크기를 적으면 컴파일에서 막힌다

new 로 공간 만들기

```
intArray = new int[5];
```

- new가 원소 다섯 칸을 만든다
- [] 안에 만들 원소 개수를 적는다
- 레퍼런스를 intArray에 넘겨준다



선언과 생성을 한 줄로

① `int intArray[];`

+

② `intArray = new int[5];`



`int intArray[] = new int[5];`

만들면 0이 들어 있다

그래서 합을 구하는 코드가 처음부터 맞는다

타입

int · long

double

boolean

기본값

0

0.0

false

배열 만들어 값 넣어 보기

Ex02_ArrayFirst.java

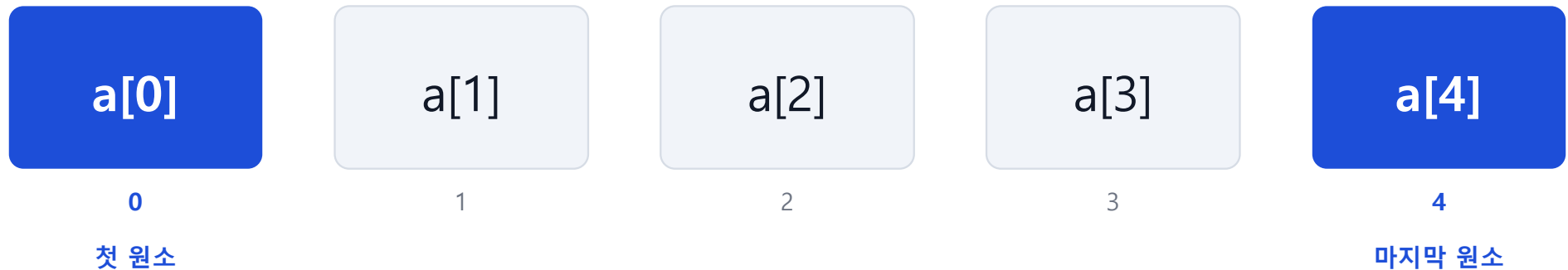
```
public class Ex02_ArrayFirst {  
    public static void main(String[] args) {  
        int score[];  
        score = new int[3];  
  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

배열 만들어 값 넣기 결과

90
85
100

세 칸에 넣은 값이 차례로 나온다

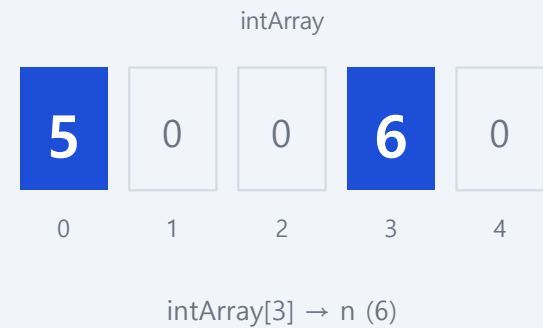
인덱스는 0부터



첫 원소는 $a[0]$, 마지막 원소는 $a[\text{크기}-1]$ 이다.

원소에 넣고 읽기

```
int intArray[] = new int[5];  
intArray[0] = 5;  
intArray[3] = 6;  
int n = intArray[3]; // n은 6
```



중괄호로 초기화

정수 배열

```
int intArray[] = {4, 3, 2, 1, 0};
```

실수 배열

```
double doubleArray[] = {0.01, 0.02, 0.03, 0.04};
```

new를 쓰지 않아도 배열이 생긴다.

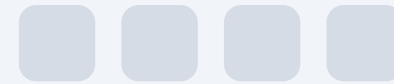
크기는 적은 개수가 정한다

intArray



값 5개 → 5칸

doubleArray



값 4개 → 4칸

- 중괄호 안에 적은 값 개수가 배열 크기를 정한다.
- 크기를 따로 적지 않는다

초기화한 배열 읽어 보기

Ex03_InitArray.java

```
public class Ex03_InitArray {  
    public static void main(String[] args) {  
        int intArray[] = {4, 3, 2, 1, 0};  
        double doubleArray[] = {0.01, 0.02, 0.03, 0.04};  
  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

초기화한 배열 결과

첫 원소 4
마지막 원소 0
실수 배열의 세 번째 0.03

첫 원소는 4, 마지막 원소는 0



선언만 하면 null

```
int intArray[];
```

선언만 했다. 아직 생성하지 않았다.

intArray



null

가리키는 배열이 없다.

1단계 선언만 끝났고, 2단계 생성은 없다.

생성하지 않고 쓰면

Ex04_NullArray.java

```
public class Ex04_NullArray {  
    public static void main(String[] args) {  
        int intArray[] = null;  
  
        System.out.println("배열 변수만 있습니다.");  
        System.out.println(intArray[0]);  
    }  
}
```

실행 오류 화면

`java.lang.NullPointerException`

배열을 만들지 않으면 실행 중에 멈춘다

범위를 넘어 접근하면

파일: Ex05_RangeError.java

```
public class Ex05_RangeError {  
    public static void main(String[] args) {  
        int intArray[] = new int[5];  
  
        intArray[0] = 5;  
        intArray[3] = 6;  
        System.out.println("인덱스 3 의 값은 " + intArray[3]);  
        System.out.println("인덱스 5 의 값은 " + intArray[5]);  
    }  
}
```

인덱스 오류 화면

```
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
```

인덱스 5는 다섯 칸 배열에 없다

컴파일 오류와 실행 오류

	컴파일 오류	실행 오류
언제 걸리나	실행하기 전에	실행하다가 그 줄에서
실습실 끝 줄	컴파일 오류로 실행하지 못했어요	실행이 오류로 끝났어요 (종료 코드 1)
오늘 본 예	선언에 크기를 적은 것	없는 배열.없는 인덱스를 쓴 것

예제 3-7 ArrayAccess

Ex06_ArrayAccess.java

```
import java.util.Scanner;

public class Ex06_ArrayAccess {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int intArray[]; // 배열에 대한 레퍼런스 선언
        intArray = new int[5]; // 배열 생성
        int max = 0; // 현재 가장 큰 수
        System.out.println("양수 5개를 입력하세요.");
        for (int i = 0; i < 5; i++) {
            intArray[i] = scanner.nextInt(); // 입력받은 정수를 배열에 저장
            if (intArray[i] > max) // 입력받은 수가 현재 제일 크면
                max = intArray[i]; // max 변경
        }
        System.out.print("가장 큰 수는 " + max + "입니다.");
        scanner.close();
    }
}
```

예제 3-7 실행 결과

양수 5개를 입력하세요.
1 39 78 100 99
가장 큰 수는 100입니다.

가장 큰 수는 100

max 가 바뀌는 자리

```
if(intArray[i] > max)
```

```
    max = intArray[i];
```

1 → 39 → 78 → 100

다섯 번째 입력 99는 100보다 작아 max가 그대로다

내 점수로 배열 채우기

Ex07_MyScores.java

```
public class Ex07_MyScores {  
    public static void main(String[] args) {  
        int score[] = new int[3];  
        int sum = 0;  
  
        // 여기서 부터 코드를 작성하세요.  
  
  
        // 여기 까지  
  
        System.out.println("세 과목 합은 " + sum);  
    }  
}
```

내 점수 합 결과

세 과목 합은 245

세 과목 합은 245

레퍼런스 치환

```
int myArray[] = intArray;
```

intArray



같은 배열

원소를 저장하는 공간 하나

myArray



두 이름이 같은 배열을 가리킨다

한쪽을 바꾸면 어떻게 될까

```
intArray[0] = 5;  
myArray[0] = 100;
```

배열 공유 확인해 보기

Ex08_ShareArray.java

```
public class Ex08_ShareArray {  
    public static void main(String[] args) {  
        int intArray[] = new int[5];  
        int myArray[] = intArray;  
  
        intArray[0] = 5;  
        myArray[0] = 100;  
  
        System.out.println("intArray[0] 은 " + intArray[0]);  
        System.out.println("myArray[0] 은 " + myArray[0]);  
    }  
}
```

배열 공유 결과

intArray[0] 은 100
myArray[0] 은 100

두 이름이 같은 배열을 가리킨다

1교시에서 본 것

1 배열은 같은 종류 값을 번호로 꺼내 쓰는 그릇

2 선언과 생성은 다른 일

3 인덱스는 0부터 크기-1까지

4 레퍼런스를 옮기면 배열은 하나다

1교시 실습 살펴보기



실습①

선언과 생성은
다른 두 단계다



실습②

중괄호 안 개수가
크기를 정한다



실습③

과목이 몇 개든
코드는 그대로다

2교시

length 와 for-each 문

배열이 자기 크기를 알려 주면 반복이 짧아진다.

length 필드

for-each

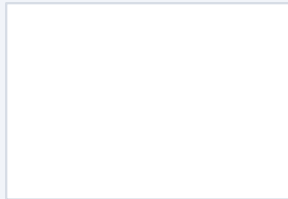
고르는 기준

배열은 크기를 스스로 안다

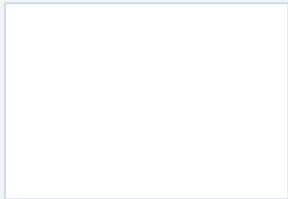
자바는 배열을 객체로 다룬다

배열 객체

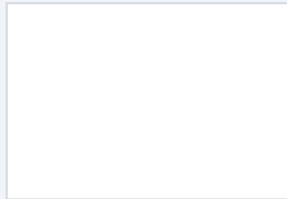
length = 5



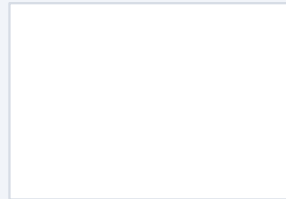
0



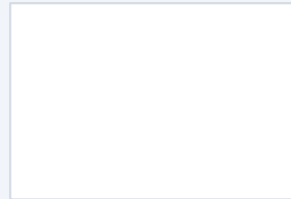
1



2



3



4

length 로 크기 알아내기

```
int intArray[] = new int[5];  
  
int size = intArray.length; // size는 5
```



괄호를 붙이지 않는다

length는 필드라서 값처럼 쓴다. String의 length()와는 다르다.

length 만큼 도는 for 문

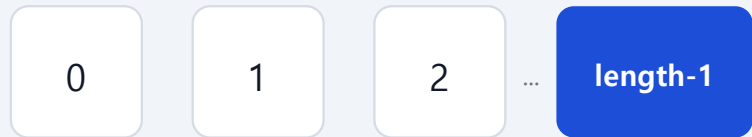
예시 코드

```
for(int i=0; i<intArray.length; i++)  
    System.out.println(intArray[i]);
```

조건식: $i < \text{intArray.length}$

0부터 length-1까지 돈다.

인덱스 범위



0 ~ length-1

크기를 따로 관리하지 않는다



변수로 따로 적어 둔다

크기가 바뀌면 변수도 같이 고쳐야 한다



배열에게 묻는다

배열이 들고 있는 값을 그때그때 확인한다

배열 전체 출력해 보기

Ex09_ArrayPrint.java

```
public class Ex09_ArrayPrint {  
    public static void main(String[] args) {  
        int intArray[] = {10, 20, 30, 40, 50, 60};  
  
        // 여기서 부터 코드를 작성하세요.  
  
  
        // 여기 까지  
    }  
}
```

배열 전체 출력 결과

배열 크기는 6
10 20 30 40 50 60

크기는 6, 원소는 여섯 개

5를 박아 두면

크기를 직접 적음

```
for (int i = 0; i < 5; i++) {  
    sum += intArray[i];  
}
```

늘리면 마지막 원소를 빠뜨린다

length를 씀

```
for (int i = 0; i < intArray.length; i++) {  
    sum += intArray[i];  
}
```

늘어나도 그대로 돈다

☕ 256번째 판에서 깨진 화면

1980년 오락실 게임 팩맨은 판 번호를 1바이트에 적었다.

256번째 판이 되자 판 번호가 255를 넘어 0으로 돌아갔다.

화면 오른쪽 절반이 글자와 기호로 뒤덮여 더 진행할 수 없었다. 이를 킬 스크린이라 부른다.

출처: 1980년 남코 팩맨 기판 분해 분석





크기를 넘어 그리면



게임이라면

화면이 깨진 채로 돈다.



자바라면

`ArrayIndexOutOfBoundsException`
실행을 멈추고 알려준다.

예제 3-8 ArrayLength

Ex10_ArrayLength.java

```
import java.util.Scanner;

public class Ex10_ArrayLength {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("5개의 정수를 입력하세요.");
        int intArray[] = new int[5]; // 배열의 선언과 생성
        double sum = 0.0;
        for (int i = 0; i < intArray.length; i++)
            intArray[i] = scanner.nextInt(); // 키보드에서 입력받은 정수 저장

        for (int i = 0; i < intArray.length; i++)
            sum += intArray[i]; // 배열에 저장된 정수 값을 더하기

        System.out.print("평균은 " + sum / intArray.length);
        scanner.close();
    }
}
```

예제 3-8 실행 결과

5개의 정수를 입력하세요.

2 3 4 5 9

평균은 4.6

평균은 4.6

평균이 4.6인 이유

```
double sum = 0.0;
```

sum을 double로 선언했으므로 나눗셈도 실수로 계산된다.

double
4.6

vs

int
4

배열을 넘길 때

	C/C++	자바
넘기는 것	배열과 크기를 따로	배열만
크기를 아는 법	같이 받은 size	x.length
호출	sum(a, 5)	sum(a)

훑기만 할 때

```
for (int i = 0; i < intArray.length; i++) {  
    System.out.println(intArray[i]);  
}
```

3번 등장

`i = 0 · i < length · i++`

1번 사용

`intArray[i]`

for-each 문의 구조

```
for(변수 : 배열레퍼런스) {  
    반복 작업문  
}
```

조건식도 증감식도 없다

변수

원소를 담을 변수

배열레퍼런스

배열 이름

for-each 를 읽는 법

```
for (int k : n) {  
    합 += k;  
}
```

 이렇게 읽는다

- for(int k : n) → 배열 n의 원소를 하나씩 k에 담아
- 배열 크기만큼 돌고 저절로 끝난다

같은 합을 두 문법으로

for-each

```
int sum = 0;
```

```
for(int k : n) {
```

```
    sum += k;
```

```
}
```

for

```
int sum = 0;
```

```
for(int i=0; i<n.length; i++) {
```

```
    int k = n[i];
```

```
    sum += k;
```

```
}
```

for-each 로 합 구해 보기

Ex11_ForEachSum.java

```
public class Ex11_ForEachSum {  
    public static void main(String[] args) {  
        int n[] = {1, 2, 3, 4, 5};  
        int sum = 0;  
  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
        System.out.println("합은 " + sum);  
    }  
}
```

for-each 합 결과

합은 15

합은 15

예제 3-9 foreachEx

Ex12_ForeachEx.java

```
public class Ex12_ForeachEx {  
    public static void main(String[] args) {  
        int[] n = {1, 2, 3, 4, 5};  
        int sum = 0;  
        for (int k : n) {           // k 는 n[0], n[1], ... n[4] 로 반복  
            System.out.print(k + " ");    // 반복되는 k 값 출력  
            sum += k;  
        }  
        System.out.println("합은 " + sum);  
  
        String f[] = {"사과", "배", "바나나", "체리", "딸기", "포도"};  
        for (String s : f)           // s 는 f[0], f[1], ... f[5] 로 반복  
            System.out.print(s + " ");  
    }  
}
```

예제 3-9 실행 결과

1 2 3 4 5 합은 15
사과 배 바나나 체리 딸기 포도

정수 배열과 문자열 배열이 같은 문법으로 돈다

문자열 배열도 같다

```
String f[] = {"사과", "배", ...};  
for (String s : f) {  
    System.out.println(s);  
}
```

콜론 왼쪽 변수의 타입은 배열 원소의 타입을 따른다.

- 정수 배열이면 int
- 문자열 배열이면 String



인덱스를 안 쓰면 안 틀린다

인덱스로 반복

```
for (int i = 0; i < n.length; i++) {  
    System.out.println(n[i]);  
}
```

for-each로 반복

```
for (int k : n) {  
    System.out.println(k);  
}
```

인덱스 없음

for-each 로 못 하는 것



몇 번째인지 모른다

지금 몇 번째 원소인지 알 수 없다



값을 못 바꾼다

원소 값을 바꿀 수 없다

인덱스가 필요하거나 값을 바꿔야 하면 for 문을 쓴다

for-each 로 원소를 바꾸면

```
public class Ex13_ForEachAssign {  
    public static void main(String[] args) {  
        int n[] = {1, 2, 3};  
  
        for (int k : n) {  
            k = k * 10;  
        }  
  
        for (int k : n) {  
            System.out.print(k + " ");  
        }  
    }  
}
```

파일: Ex13_ForEachAssign.java

원소를 바꾸려 한 결과

1 2 3

값이 그대로 1 2 3

배열을 도는 반복 고르기

배열을 처음부터 끝까지 훑기만 하면 for-each 문.

인덱스가 필요하거나 원소 값을 바꿔야 하면 for 문.

문제: 문자 10개 배열 선언

확인 문제

10개의 문자를 저장하는 배열 myChar를
선언하는 코드는?

다음 장에서 정답을 확인한다.

풀이: 두 가지 선언

```
char myChar[] = new char[10];
```

```
char[] myChar = new char[10];
```

선언만으로는 공간이 생기지 않아 new가 필요하다

1에서 10까지 배열에 담아 더하기

Ex14_SumTen.java

```
public class Ex14_SumTen {  
    public static void main(String[] args) {  
        int n[] = new int[10];  
        int sum = 0;  
  
        // 여기서 부터 코드를 작성하세요.  
  
  
        // 여기 까지  
  
        System.out.println("1부터 10까지의 합은 " + sum);  
    }  
}
```

1에서 10까지 합 결과

1부터 10까지의 합은 55

1부터 10까지의 합은 55

2교시에서 본 것

- 배열은 자기 크기를 length로 알려 준다
- 조건식에 length를 쓰면 크기가 바뀌어도 고칠 곳이 없다
- for-each는 원소를 하나씩 담아 끝까지 돈다
- 인덱스가 필요하거나 값을 바꿔야 하면 for 문

length

for-each

2교시 실습 살펴보기



실습④

길이 필드에는
괄호가 없다



실습⑤

끝내는 조건을
적지 않는다



실습⑥

for와 for-each를
목적따라 나눈다

3교시

2차원 배열과 배열 리턴

줄과 칸이 있는 값, 그리고 배열을 돌려주는 메소드.

행과 열

배열의 배열

리턴은 레퍼런스

표 모양으로 된 값

성적표처럼 줄과 칸이 있는 값이 있다.

- 학년별 1,2학기 성적
- 반별 과목 점수
- 좌석표

	1학기	2학기
1학년		
2학년		
3학년		
4학년		

2차원 배열 선언

변수 뒤에 대괄호 둘

```
int intArray[][];
```

대괄호 둘을 변수 뒤에 적는다.

=

타입 뒤에 대괄호 둘

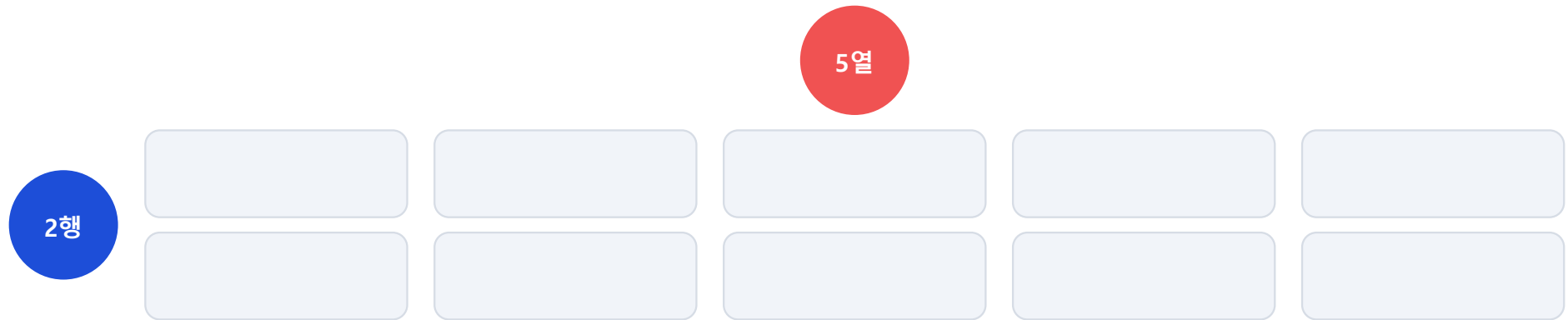
```
int[][] intArray;
```

대괄호 둘을 타입 옆에 적는다.

대괄호가 하나면 1차원, 둘이면 2차원이다.

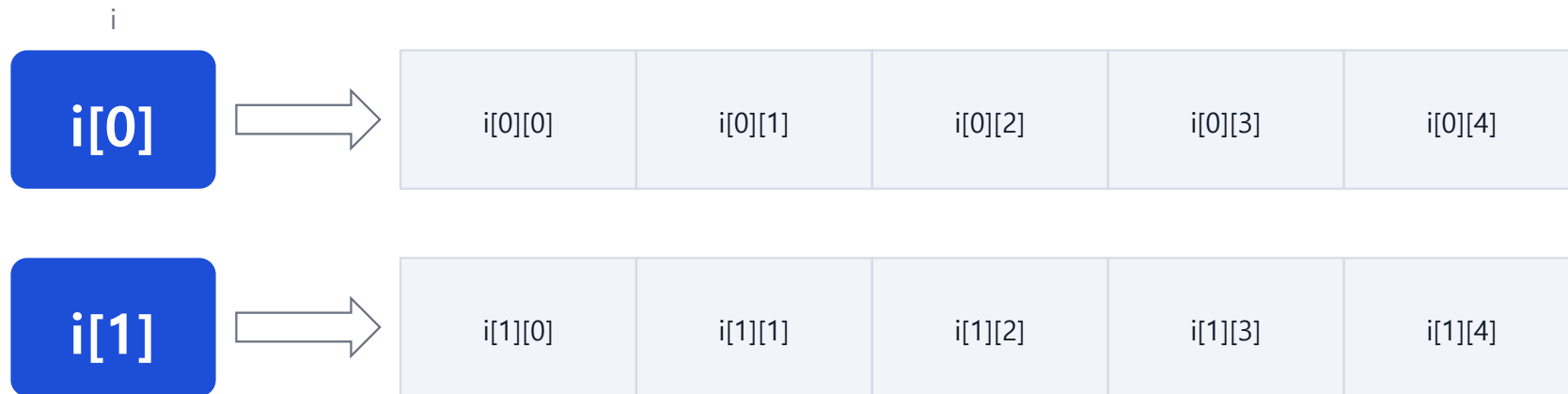
행과 열을 정해 만들기

```
intArray = new int[2][5];  
int intArray[][] = new int[2][5];
```



가로로 다섯 칸이 열리고, 세로로 두 줄이 행이다.

2차원 배열의 구조



행의 원소는 1차원 배열의 레퍼런스다

배열 크기와 인덱스

1차원		2차원
크기를 묻는 법	<code>a.length</code>	<code>score.length</code> 는 행 <code>score[0].length</code> 는 열
첫 인덱스	<code>a[0]</code>	<code>score[0][0]</code>
마지막 인덱스	<code>a[a.length-1]</code>	<code>score[score.length-1]</code> <code>[score[0].length-1]</code>
넘어가면	실행 오류	실행 오류

원소 하나를 집는 법

```
intArray[1][3] = 7;
```

대괄호 순서는 [행][열]이다.

intArray

	열 0	열 1	열 2	열 3
행 0				
행 1				7

2차원 배열 초기화

```
int intArray[][] = {{0,1,2},{3,4,5},{6,7,8}}; // 3x3
```

```
char charArray[][] = {{'a','b','c'},{'d','e','f'}}; // 2x3
```

```
double doubleArray[][] = {{0.01,0.02},{0.03,0.04}}; // 2x2
```

intArray[][] 예

{0, 1, 2}

{3, 4, 5}

{6, 7, 8}



중괄호 안에 중괄호를 행마다 하나씩 둔다

int-char-double 모두 같은 방식이다.

2차원 배열 채워 보기

Ex15_TwoDimFirst.java

```
public class Ex15_TwoDimFirst {  
    public static void main(String[] args) {  
        int intArray[][] = new int[2][5];  
  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

2차원 배열 채우기 결과

```
0 1 2 3 4
1 2 3 4 5
```

두 줄 다섯 칸이 행과 열대로 찍힌다



1부터 세던 언어

C 에서 배열 이름은 첫 칸의 주소이고, 인덱스는 첫 칸에서 몇 칸 떨어졌는가를 뜻했다. 첫 칸은 0칸 떨어져 있다.

- 1957년 포트란은 크기 10짜리 배열이면 번호가 1부터 10까지
- 자바는 C의 방식을 그대로 물려받았다

출처: IBM FORTRAN 명세(1957) · Dennis Ritchie, 1993





0부터 세는 약속

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]

행도 열도 0부터 시작한다

- 첫 칸은 [0][0]이다
- 마지막 칸은 [1][4]다

예제 3-10 ScoreAverage · Ex16_ScoreAverage.java

교수 시연

```
public class Ex16_ScoreAverage {
    public static void main(String[] args) {
        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 sum = 0;

        for (int year = 0; year < score.length; year++)    // 각 학년별로 반복
            for (int term = 0; term < score[year].length; term++)    // 각 학년의 학기별로 반복
                sum += score[year][term];    // 전체 평점 더하기

        int n = score.length;    // 배열의 행 개수, 4
        int m = score[0].length;    // 배열의 열 개수, 2
        System.out.println("4년 전체 평점 평균은 " + sum / (n * m));
    }
}
```

예제 3-10 실행 결과

4년 전체 평점 평균은 3.725

4년 전체 평점 평균은 3.725

왜 `score[year].length` 인가

≡ 바깥 반복

- `score.length`
- 행 개수 4개

[] 안쪽 반복

- `score[year].length`
- 열 개수 2개

2차원 배열은 배열의 배열이다. 행마다 열 개수를 따로 묻는다.

중첩 for 가 만나는 자리

1

바깥 for 시작
행 하나를 고정한다

2

안쪽 for 전체 순회
열 0~length-1을 돈다

3

바깥 for 다음 행
같은 과정을 반복한다

	열0	열1
행0	1	2
행1	3	4

숫자는 방문 순서다. 바깥이 행, 안쪽이 열이다.

2차원 배열과 for-each

예시 코드

```
for(double row[] : score)
    for(double s : row)
        sum += s;
```

row는 한 행(1차원 배열)

정리

- 바깥 반복이 꺼내는 건 행 하나
- 안쪽 반복은 행의 원소 하나씩
- 원소 값을 sum에 더한다

내 성적표로 평균 내기

Ex17_MyScoreTable.java

```
public class Ex17_MyScoreTable {  
    public static void main(String[] args) {  
        double score[][] = {{4.0, 3.0},    // 1학년  
                             {3.5, 4.5}};    // 2학년  
  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

내 성적표 평균 결과

1학년 평균은 3.5

2학년 평균은 4.0

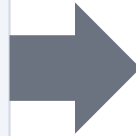
학년마다 한 줄씩 평균이 나온다

메소드가 배열을 돌려줄 때



지금까지

메소드는 값 하나만 돌려줬다.



이제부터

배열도 돌려줄 수 있다.

메소드는 5주차에서 다룬다. 지금은 배열을 돌려받는 모습만 본다.

배열을 리턴하는 메소드

그림 3-15

```
int[] makeArray() {  
    int temp[] = new int[4];  
    return temp;  
}
```

리턴 타입: int[]



- 리턴 타입에 int[] 사용
- temp 배열을 그대로 반환

돌아오는 것은 레퍼런스



생각하기 쉬운 그림

배열 전체가 통째로 복사되어
돌아온다고 생각하기 쉽다



실제로 일어나는 일

실은 레퍼런스 하나만
복사되어 돌아온다

1교시에서 본 레퍼런스 치환과 같은 원리다.

리턴받는 세 단계



메소드가 끝나도 리턴받은 배열은 사라지지 않는다.

리턴 타입에 크기를 적지 않는다



리턴 타입엔 크기 없음

```
int[] makeArray()
```

컴파일된다



리턴 타입에 크기 적음

```
int[4] makeArray()
```

컴파일 오류다

배열 선언에서 크기를 못 적는 것과 같은 규칙이다.

예제 3-11 ReturnArray

Ex18_ReturnArray.java

```
public class Ex18_ReturnArray {
    static int[] makeArray() {           // 일차원 정수 배열 리턴
        int temp[] = new int[4];        // 배열 생성
        for (int i = 0; i < temp.length; i++)
            temp[i] = i;                 // 배열 초기화, 0, 1, 2, 3
        return temp;                    // 배열 리턴
    }

    public static void main(String[] args) {
        int intArray[];                  // 배열 레퍼런스 변수 선언
        intArray = makeArray();           // 메소드가 리턴한 배열 참조
        for (int i = 0; i < intArray.length; i++)
            System.out.print(intArray[i] + " ");
    }
}
```

예제 3-11 실행 결과

0 1 2 3

0 1 2 3 이 그대로 돌아왔다

메소드가 끝나도 배열은 남는다

메소드 안에서

temp



0 1 2 3

메소드가 끝난 후

temp 사라짐

intArray



0 1 2 3

메소드 안 배열과 동일

2차원 배열을 리턴하는 메소드

1차원

int[]



2차원

int[][]

```
int[][] make2DArray() {  
    return new int[3][4];  
}
```

```
int[][] int2Array = make2DArray();
```

문제: 옳은 선언과 생성 고르기

① `int [] a = new int [];`

② `int a [5] = new int [5];`

③ `int a [] = new int [5];`

④ `int [5] a = {1,2,3,4,5};`

풀이: 셋이 틀린 이유

정답은 ③이다.

①

생성할 때 크기를 적어야 한다.

②

선언에는 크기를 적을 수 없다.

④

선언에는 크기를 적을 수 없고, 중괄호 초기화엔 `new`도 크기도 없다.

문제: 리턴 타입 채우기

문자 배열을 리턴하는 메소드다. 빈칸을 채운다.

```
_____ makeCharArray() {  
    char [] c = new char[4];  
    _____  
}
```

풀이

리턴 타입

`char[]`

마지막 줄

`return c;`

리턴 타입에는 크기를 적지 않는다 · 교재 p.116

종합 실습으로 할 일

1

점수 배열 받기

배열을 만들어 돌려주는 메소드에서 점수 배열을 받는다

2

최고점·합 계산

for 문으로 최고점과 합을 구한다

3

과목 수 세기

과목 수는 length로 센다

점수 배열로 최고점과 평균

Ex19_MakeScores.java

```
public class Ex19_MakeScores {
    static int[] makeScores() {
        int temp[] = {88, 95, 70, 100, 62};
        return temp;
    }

    public static void main(String[] args) {
        int score[] = makeScores();
        int max = score[0];
        int sum = 0;

        // 여기서 부터 코드를 작성하세요.

        // 여기 까지

        System.out.println("과목 수는 " + score.length);
        System.out.println("최고점은 " + max);
        System.out.println("평균은 " + (double) sum / score.length);
    }
}
```

종합 실습 결과

과목 수는 5
최고점은 100
평균은 83.0

과목 수 5, 최고점 100, 평균 83.0

같은 일을 for-each 로

Ex20_ScoreForEach.java

```
public class Main {  
    public static void main(String[] args) {  
        int[] score = {75, 90, 100, 68, 82};  
        int max = score[0], sum = 0;  
        for (int s : score) {  
            if (s > max) max = s; sum += s;  
        }  
        int subject = score.length;  
        System.out.println("과목 수 " + subject + ", 최고점 " + max + ", 평균 " + ((double) sum / subject));  
    }  
}
```

3교시에서 본 것

01

2차원 배열은 배열의 배열

02

행 개수는 length, 열 개수는 행의 length

03

메소드는 배열을 돌려줄 수 있다

04

돌려받는 것은 레퍼런스이지 복사본이 아니다

3교시 실습 살펴보기

실습⑦

안쪽 반복이 끝나야 한 학년 줄이 끝난다.

실습⑧

학년이 바뀔 때마다 합을 0으로 되돌린다.

실습⑨

0으로 두면 모든 점수가 그보다 작을 때 값이 틀린다. 배열의 첫 값을 후보로 쓴다.

내 점수 배열 만들기

미션 조건

- 다섯 과목 점수를 배열에 담아 최고점.평균 출력
- 같은 계산을 for·for-each 두 가지로 작성
- 제출: 코드와 실행 화면

배열 과제에서 볼 세 가지

배열 크기를 length 로 세었나
인덱스가 0부터 크기-1까지인가
값만 훑는 반복에 for-each 를 썼나

안 될 때 여기부터

틀림

- 선언에 크기를 적음
- 만들지 않은 배열을 씀
- 인덱스를 크기까지 돌림

맞음

- ✓ 크기는 new 뒤에 적는다
- ✓ new 로 만든 뒤에 값을 넣는다
- ✓ 조건식은 $i < a.length$

실행이 오류로 끝났어요 가 뜨면 예외 이름 줄부터 읽는다

오늘 공부한 내용

배열은 같은 종류 값을 번호로 꺼내 쓰는 그릇

선언은 이름, 생성은 공간. 선언만 하면 `null`

인덱스는 0부터 크기-1까지, 넘으면 실행 오류

`length` 는 배열이 들고 있는 크기

훅기만 하면 `for-each`, 인덱스가 필요하면 `for`

다음 주엔 클래스와 객체

오늘 쓴 배열도 사실은 객체였습니다



다음 주 예고

- 클래스 정의와 객체 생성
- 필드와 메소드



반드시 지참

- 내 점수 배열 과제 (제출)
- 두 가지 반복으로 작성