10강 - 배열

학급의 성적표를 보고 각 학생별로 총점과 평균을 구해야 한다고 생각해 봅시다.

성적표가 아래와 같다면 3명씩 3과목이므로 총 9개의 변수가 필요할 것 입니다.

이름	국어	영어	수학
철수	92	81	76
영희	72	95	84
민혁	80	86	98

```
int kor1 = 92;
int kor2 = 72;
int kor3 = 80;
// ... 생략 ...
int math2 = 84;
int math3 = 98;
```

만약 30명의 학생에 대한 20과목에 대한 점수라고 가정한다면 프로그램은 좀 더 복잡해 지고 더 많은 변수를 소스코드상에 선언,할당 해야 할 것 입니다.

#01. 배열

같은 종류의 데이터를 그룹화 한 형태.

사물함(캐비넷)같은 구조를 갖고 있다고 여겨도 좋다.

1) 배열 만들기

배열의 선언

데이터 타입 뒤에 배열임을 의미하는 `[]`를 명시한다.

```
int[] a;
```

배열의 할당

값을 대입하는 것이 아니라 배열의 칸을 결정하는 것임에 유의한다.

new 예약어 뒤에 데이터 타입을 다시 한번 명시하고 배열의 칸 수를 `[]`를 사용하여 지정한다.

```
a = new int[3];
```

선언과 할당의 통합

```
int[] a = new int[3];
```

2) 배열의 활용

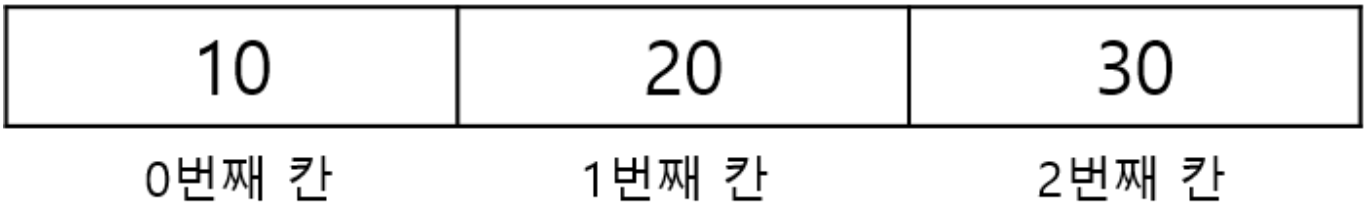
값 저장하기

크기가 결정된 배열은 0부터 시작되는 일련번호를 부여받는다. 이를 배열의 **인덱스**라고 한다.

값을 저장하기 위해서는 배열변수 이름 뒤에 `[0]` 형식으로 인덱스 번호를 지정하고 대입한다.

```
a[0] = 10;  // 인덱스가 0인 칸에 10을 대입
a[1] = 20;  // 인덱스가 1인 칸에 20을 대입
a[2] = 30;  // 인덱스가 2인 칸에 30을 대입
```

즉, 아래와 같이 구성된다고 생각할 수 있다.



위와 같이 행의 수가 1줄로만 구성되어 있는 형태의 배열을 **1차 배열** 이라고 한다.

3) 배열에 저장된 값 사용하기

인덱스 번호를 사용한다는 점을 제외하고는 일반 변수의 사용과 동일하다.

다른 변수에 복사하기

```
int k = a[0];   // k에 10이 복사됨
```

출력하기

```
System.out.println(a[1]);   // 20이 출력됨
```

연산하기

```
a[2] = a[0] * a[1]; // 인덱스가 2인 공간의 값이 200이 됨
```

4) 선언+할당+값대입을 한 번에 일괄 처리 하기

배열을 할당할 때 사이즈(칸 수)를 지정하지 않고 중괄호 {}를 열어서 배열의 원소값들을 직접 나열한다.

```
int[] a = new int[] { 10, 20, 30, 40 };
```

이 때 new int[]는 생략할 수 있다.

```
int[] a = { 10, 20, 30, 40 };
```

5) 배열의 크기 (혹은 길이)

배열이름.length는 배열의 칸 수를 반환한다.

```
int[] a = new int[5];
System.out.println(a.length);    // 5가 출력됨
```

> 배열의 인덱스는 항상 0부터 크기-1까지 1씩 증가하면서 존재한다.

5) 반복문을 통한 활용

인덱스가 0부터 크기-1까지 1씩 증가한다는 특성을 활용하여 **for**문과 함께 사용하면 배열의 모든 원소에 대한 일괄 처리가 가능하다.

```
int[] a = {10, 20, 30, 40, 50};

// i의 범위는 0부터 길이(5)보다 작은 4까지 1씩 증가
// --> 즉, i가 배열의 모든 인덱스를 순환한다.
for (int i=0; i < a.length; i++) {
    System.out.println(a[i]);
}
```

6) 배열 사용시의 주의사항

사이즈가 결정되지 않은 배열은 사용할 수 없다.

```
int[] a;
a[0] = 10;
```

- **출력결과**

```
Test.java:5: error: variable a might not have been initialized
        a[0] = 10;
        ^
1 error
```

배열은 초기값이 자동으로 할당된다.

배열의 크기만을 결정하고 원소를 대입하지 않은 경우 초기값이 자동으로 할당된다.

- 숫자형 - 0으로 초기화
- boolean형 - false로 초기화
- String형 - `null`로 초기화

> null이라는 개념은 객체라는 단원에서 다시 소개합니다.

존재하지 않는 인덱스에 접근할 경우

크기가 3인 배열에 대해 인덱스가 3인 위치에 접근한다면 에러가 발생한다.

```
int[] a = { 10, 20, 30 };
System.out.println(a[3]);
```

이 때 **컴파일(`javac`)**시에는 아무런 문제가 없고, 컴파일 된 결과물을 **실행(`java`)**할 때 에러가 발생한다.

문제가 없는 부분은 정상적으로 실행되고 문제가 있는 행을 실행하는 시점에 에러가 발생함

#02. 2차 배열

`열(=칸)`의 개념만 존재하는 1차 배열에 `행(=줄)`의 개념을 추가한 형태.

1차 배열의 각 원소가 또 다른 배열로 구성된 형태

1) 배열의 생성

선언

변수 선언시 데이터 타입 뒤에 행과 열을 의미하는 대괄호(`[]`)를 각각 명시

```
int[][] myarr;
```

할당

`new` 키워드 뒤에 데이터 타입을 명시하고 대괄호 안에 행과 열의 수를 결정해 준다.

```
myarr = new int[2][3];    // 2행3열
```

선언과 할당의 통합

```
int[][] myarr = new int[2][3];
```

2) 배열 원소에 값을 대입하기

각 원소에 직접 대입하기

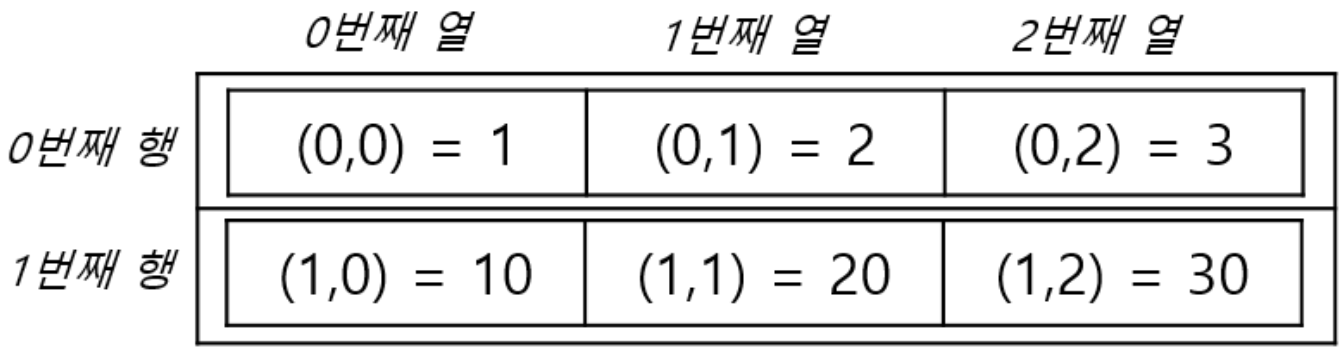
2행 3열인 경우 **행**의 인덱스는 **0부터 1까지**, **열**의 인덱스는 **0부터 2까지** 존재한다.

배열에 저장된 원소에 접근하기 위해서는 변수이름 뒤에 **행,열의 순서로 인덱스를 명시**한다.

대입 방법은 일반 변수와 동일하다.

```
myarr[0][0] = 1;    // 0행0열
myarr[0][1] = 2;    // 0행1열
myarr[0][2] = 3;    // 0행2열

myarr[1][0] = 10;    // 1행0열
myarr[1][1] = 20;    // 1행1열
myarr[1][2] = 30;    // 1행2열
```



선언, 할당, 값 대입을 일괄 처리하기

```
int[][] myarr = new int[][] { {1, 2, 3}, {10, 20, 30} };
```

`new int[][]`는 생략 가능

```
int[][] myarr = { {1, 2, 3}, {10, 20, 30} };
```

2차 배열의 의미

2차 배열은 1차 배열 안에 각각 개별적인 1차 배열이 원소로서 포함되는 형태로 이해해야 한다.

```
// 독립적인 1차 배열 구성
int[] arr1 = {1, 2, 3};
int[] arr2 = {10, 20, 30};

// 2차 배열은 1차 배열 안에 또 다른 각각의 배열이 포함되는 형태.
int[][] twoArray = { arr1, arr2 };
```

3) 가변배열

2차 배열로 포함되는 배열이 각각 크기가 다른 형태

TwoDimArray1.java 예제와 같이 항상 배열의 모든 행이 동일한 열로 구성되는 것은 아니다. (모든 줄의 칸 수가 같다는 보장은 없다는 의미)

```
int[][] myarr = new int[][] { {1, 2, 3, 4}, {10, 20} };
```

> 가변배열이 자주 등장하는 것은 아니다. 95% 이상의 경우가 모든 행의 열 크기가 동일한 경우이다.

4) 2차 배열의 행, 열 크기 구하기

2차 배열에 대한 길이를 직접 구하면 **행의 크기**를 알 수 있다.

```
int rowSize = myarr.length;
```

2차 배열의 모든 행에 대한 열 크기가 항상 동일하다는 보장이 없기 때문에 열의 크기는 **각 행마다 개별적**으로 구해야 한다.

```
int colSize0 = myarr[0].length; // 0번째 행의 열 크기
int colSize1 = myarr[1].length; // 1번째 행의 열 크기
```

5) 2차 배열과 반복문

배열의 모든 원소 스캔하기

2차 배열의 모든 원소를 반복문으로 스캔하기 위해서는 **중첩 반복문**을 사용해야 한다.

이 때 부모 반복문은 **행**에 대해 관여하고, 자식 반복문은 **열**에 대해 관여한다.